Flutter resumen documentación

# Scrolling widgets

## custom scroll view

A [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) that creates custom scroll effects using [slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html).

A [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) lets you supply [slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) directly to create various scrolling effects, such as lists, grids, and expanding headers. For example, to create a scroll view that contains an expanding app bar followed by a list and a grid, use a list of three slivers: [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html), [SliverList](https://api.flutter.dev/flutter/widgets/SliverList-class.html), and [SliverGrid](https://api.flutter.dev/flutter/widgets/SliverGrid-class.html).

[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)s in these [slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) must produce [RenderSliver](https://api.flutter.dev/flutter/rendering/RenderSliver-class.html) objects.

To control the initial scroll offset of the scroll view, provide a [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html) with its [ScrollController.initialScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/initialScrollOffset.html) property set.

By default, if items are inserted at the "top" of a scrolling container like [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) or [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html), the top item and all of the items below it are scrolled downwards. In some applications, it's preferable to have the top of the list just grow upwards, without changing the scroll position. This example demonstrates how to do that with a [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) with two [SliverList](https://api.flutter.dev/flutter/widgets/SliverList-class.html) children, and the [CustomScrollView.center](https://api.flutter.dev/flutter/widgets/ScrollView/center.html) set to the key of the bottom SliverList. The top one SliverList will grow upwards, and the bottom SliverList will grow downwards.

### Accessibility

A [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) can allow Talkback/VoiceOver to make announcements to the user when the scroll state changes. For example, on Android an announcement might be read as "showing items 1 to 10 of 23". To produce this announcement, the scroll view needs three pieces of information:

* The first visible child index.
* The total number of children.
* The total number of visible children.

The last value can be computed exactly by the framework, however the first two must be provided. Most of the higher-level scrollable widgets provide this information automatically. For example, [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) provides each child widget with a semantic index automatically and sets the semantic child count to the length of the list.

To determine visible indexes, the scroll view needs a way to associate the generated semantics of each scrollable item with a semantic index. This can be done by wrapping the child widgets in an [IndexedSemantics](https://api.flutter.dev/flutter/widgets/IndexedSemantics-class.html).

This semantic index is not necessarily the same as the index of the widget in the scrollable, because some widgets may not contribute semantic information. Consider a [ListView.separated](https://api.flutter.dev/flutter/widgets/ListView/ListView.separated.html): every other widget is a divider with no semantic information. In this case, only odd numbered widgets have a semantic index (equal to the index ~/ 2). Furthermore, the total number of children in this example would be half the number of widgets. (The [ListView.separated](https://api.flutter.dev/flutter/widgets/ListView/ListView.separated.html) constructor handles this automatically; this is only used here as an example.)

The total number of visible children can be provided by the constructor parameter semanticChildCount. This should always be the same as the number of widgets wrapped in [IndexedSemantics](https://api.flutter.dev/flutter/widgets/IndexedSemantics-class.html).

### Persisting the scroll position during a session

Scroll views attempt to persist their scroll position using [PageStorage](https://api.flutter.dev/flutter/widgets/PageStorage-class.html). This can be disabled by setting [ScrollController.keepScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/keepScrollOffset.html) to false on the [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html). If it is enabled, using a [PageStorageKey](https://api.flutter.dev/flutter/widgets/PageStorageKey-class.html) for the [key](https://api.flutter.dev/flutter/widgets/Widget/key.html) of this widget is recommended to help disambiguate different scroll views from each other.

### Constructors

[CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView/CustomScrollView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? center, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) anchor = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> slivers = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) that creates custom scroll effects using slivers.

const

### Properties

[anchor](https://api.flutter.dev/flutter/widgets/ScrollView/anchor.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The relative position of the zero scroll offset.

finalinherited

[cacheExtent](https://api.flutter.dev/flutter/widgets/ScrollView/cacheExtent.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

The viewport has an area before and after the visible area to cache items that are about to become visible when the user scrolls.

finalinherited

[center](https://api.flutter.dev/flutter/widgets/ScrollView/center.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

The first child in the [GrowthDirection.forward](https://api.flutter.dev/flutter/rendering/GrowthDirection.html) growth direction.

finalinherited

[clipBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

finalinherited

[controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

An object that can be used to control the position to which this scroll view is scrolled.

finalinherited

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

finalinherited

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[keyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/keyboardDismissBehavior.html) → [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html)

[ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) the defines how this [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) will dismiss the keyboard automatically.

finalinherited

[physics](https://api.flutter.dev/flutter/widgets/ScrollView/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the scroll view should respond to user input.

finalinherited

[primary](https://api.flutter.dev/flutter/widgets/ScrollView/primary.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Whether this is the primary scroll view associated with the parent [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html).

finalinherited

[restorationId](https://api.flutter.dev/flutter/widgets/ScrollView/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

finalinherited

[reverse](https://api.flutter.dev/flutter/widgets/ScrollView/reverse.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the scroll view scrolls in the reading direction.

finalinherited

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/scrollBehavior.html) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)?

A [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) that will be applied to this widget individually.

finalinherited

[scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) along which the scroll view's offset increases.

finalinherited

[semanticChildCount](https://api.flutter.dev/flutter/widgets/ScrollView/semanticChildCount.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)?

The number of children that will contribute semantic information.

finalinherited

[shrinkWrap](https://api.flutter.dev/flutter/widgets/ScrollView/shrinkWrap.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the extent of the scroll view in the [scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html) should be determined by the contents being viewed.

finalinherited

[slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>

The slivers to place inside the viewport.

final

## draggable scrollable sheet

A container for a [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) that responds to drag gestures by resizing the scrollable until a limit is reached, and then scrolling.

This widget can be dragged along the vertical axis between its [minChildSize](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/minChildSize.html), which defaults to 0.25 and [maxChildSize](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/maxChildSize.html), which defaults to 1.0. These sizes are percentages of the height of the parent container.

The widget coordinates resizing and scrolling of the widget returned by builder as the user drags along the horizontal axis.

The widget will initially be displayed at its initialChildSize which defaults to 0.5, meaning half the height of its parent. Dragging will work between the range of minChildSize and maxChildSize (as percentages of the parent container's height) as long as the builder creates a widget which uses the provided [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html). If the widget created by the [ScrollableWidgetBuilder](https://api.flutter.dev/flutter/widgets/ScrollableWidgetBuilder.html) does not use the provided [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html), the sheet will remain at the initialChildSize.

By default, the widget will stay at whatever size the user drags it to. To make the widget snap to specific sizes whenever they lift their finger during a drag, set [snap](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snap.html) to true. The sheet will snap between [minChildSize](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/minChildSize.html) and [maxChildSize](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/maxChildSize.html). Use [snapSizes](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snapSizes.html) to add more sizes for the sheet to snap between.

The snapping effect is only applied on user drags. Programmatically manipulating the sheet size via [DraggableScrollableController.animateTo](https://api.flutter.dev/flutter/widgets/DraggableScrollableController/animateTo.html) or [DraggableScrollableController.jumpTo](https://api.flutter.dev/flutter/widgets/DraggableScrollableController/jumpTo.html) will ignore [snap](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snap.html) and [snapSizes](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snapSizes.html).

By default, the widget will expand its non-occupied area to fill available space in the parent. If this is not desired, e.g. because the parent wants to position sheet based on the space it is taking, the [expand](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/expand.html) property may be set to false.

### Constructors

[DraggableScrollableSheet](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/DraggableScrollableSheet.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) initialChildSize = 0.5, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) minChildSize = 0.25, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxChildSize = 1.0, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) expand = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) snap = false, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? snapSizes, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)? snapAnimationDuration, [DraggableScrollableController](https://api.flutter.dev/flutter/widgets/DraggableScrollableController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shouldCloseOnMinExtent = true, required [ScrollableWidgetBuilder](https://api.flutter.dev/flutter/widgets/ScrollableWidgetBuilder.html) builder})

Creates a widget that can be dragged and scrolled in a single gesture.

const

### Properties

[builder](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/builder.html) → [ScrollableWidgetBuilder](https://api.flutter.dev/flutter/widgets/ScrollableWidgetBuilder.html)

The builder that creates a child to display in this widget, which will use the provided [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html) to enable dragging and scrolling of the contents.

final

[controller](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/controller.html) → [DraggableScrollableController](https://api.flutter.dev/flutter/widgets/DraggableScrollableController-class.html)?

A controller that can be used to programmatically control this sheet.

final

[expand](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/expand.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the widget should expand to fill the available space in its parent or not.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[initialChildSize](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/initialChildSize.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The initial fractional value of the parent container's height to use when displaying the widget.

final

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[maxChildSize](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/maxChildSize.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The maximum fractional value of the parent container's height to use when displaying the widget.

final

[minChildSize](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/minChildSize.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The minimum fractional value of the parent container's height to use when displaying the widget.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[shouldCloseOnMinExtent](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/shouldCloseOnMinExtent.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the sheet, when dragged (or flung) to its minimum size, should cause its parent sheet to close.

final

[snap](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snap.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the widget should snap between [snapSizes](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snapSizes.html) when the user lifts their finger during a drag.

final

[snapAnimationDuration](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snapAnimationDuration.html) → [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)?

Defines a duration for the snap animations.

final

[snapSizes](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet/snapSizes.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>?

A list of target sizes that the widget should snap to.

final

## grid view

A scrollable, 2D array of widgets.

The main axis direction of a grid is the direction in which it scrolls (the [scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html)).

The most commonly used grid layouts are [GridView.count](https://api.flutter.dev/flutter/widgets/GridView/GridView.count.html), which creates a layout with a fixed number of tiles in the cross axis, and [GridView.extent](https://api.flutter.dev/flutter/widgets/GridView/GridView.extent.html), which creates a layout with tiles that have a maximum cross-axis extent. A custom [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) can produce an arbitrary 2D arrangement of children, including arrangements that are unaligned or overlapping.

To create a grid with a large (or infinite) number of children, use the [GridView.builder](https://api.flutter.dev/flutter/widgets/GridView/GridView.builder.html) constructor with either a [SliverGridDelegateWithFixedCrossAxisCount](https://api.flutter.dev/flutter/rendering/SliverGridDelegateWithFixedCrossAxisCount-class.html) or a [SliverGridDelegateWithMaxCrossAxisExtent](https://api.flutter.dev/flutter/rendering/SliverGridDelegateWithMaxCrossAxisExtent-class.html) for the [gridDelegate](https://api.flutter.dev/flutter/widgets/GridView/gridDelegate.html).

To use a custom [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html), use [GridView.custom](https://api.flutter.dev/flutter/widgets/GridView/GridView.custom.html).

To create a linear array of children, use a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html).

To control the initial scroll offset of the scroll view, provide a [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html) with its [ScrollController.initialScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/initialScrollOffset.html) property set.

### Transitioning to [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html)

A [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) is basically a [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) with a single [SliverGrid](https://api.flutter.dev/flutter/widgets/SliverGrid-class.html) in its [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) property.

If [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) is no longer sufficient, for example because the scroll view is to have both a grid and a list, or because the grid is to be combined with a [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html), etc, it is straight-forward to port code from using [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) to using [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) directly.

The [key](https://api.flutter.dev/flutter/widgets/Widget/key.html), [scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html), [reverse](https://api.flutter.dev/flutter/widgets/ScrollView/reverse.html), [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html), [primary](https://api.flutter.dev/flutter/widgets/ScrollView/primary.html), [physics](https://api.flutter.dev/flutter/widgets/ScrollView/physics.html), and [shrinkWrap](https://api.flutter.dev/flutter/widgets/ScrollView/shrinkWrap.html) properties on [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) map directly to the identically named properties on [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html).

The [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) property should be a list containing just a [SliverGrid](https://api.flutter.dev/flutter/widgets/SliverGrid-class.html).

The [childrenDelegate](https://api.flutter.dev/flutter/widgets/GridView/childrenDelegate.html) property on [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) corresponds to the [SliverGrid.delegate](https://api.flutter.dev/flutter/widgets/SliverMultiBoxAdaptorWidget/delegate.html) property, and the [gridDelegate](https://api.flutter.dev/flutter/widgets/GridView/gridDelegate.html) property on the [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) corresponds to the [SliverGrid.gridDelegate](https://api.flutter.dev/flutter/widgets/SliverGrid/gridDelegate.html) property.

The [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html), [GridView.count](https://api.flutter.dev/flutter/widgets/GridView/GridView.count.html), and [GridView.extent](https://api.flutter.dev/flutter/widgets/GridView/GridView.extent.html) constructors' children arguments correspond to the [childrenDelegate](https://api.flutter.dev/flutter/widgets/GridView/childrenDelegate.html) being a [SliverChildListDelegate](https://api.flutter.dev/flutter/widgets/SliverChildListDelegate-class.html) with that same argument. The [GridView.builder](https://api.flutter.dev/flutter/widgets/GridView/GridView.builder.html) constructor's itemBuilder and childCount arguments correspond to the [childrenDelegate](https://api.flutter.dev/flutter/widgets/GridView/childrenDelegate.html) being a [SliverChildBuilderDelegate](https://api.flutter.dev/flutter/widgets/SliverChildBuilderDelegate-class.html) with the matching arguments.

The [GridView.count](https://api.flutter.dev/flutter/widgets/GridView/GridView.count.html) and [GridView.extent](https://api.flutter.dev/flutter/widgets/GridView/GridView.extent.html) constructors create custom grid delegates, and have equivalently named constructors on [SliverGrid](https://api.flutter.dev/flutter/widgets/SliverGrid-class.html) to ease the transition: [SliverGrid.count](https://api.flutter.dev/flutter/widgets/SliverGrid/SliverGrid.count.html) and [SliverGrid.extent](https://api.flutter.dev/flutter/widgets/SliverGrid/SliverGrid.extent.html) respectively.

The [padding](https://api.flutter.dev/flutter/widgets/BoxScrollView/padding.html) property corresponds to having a [SliverPadding](https://api.flutter.dev/flutter/widgets/SliverPadding-class.html) in the [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) property instead of the grid itself, and having the [SliverGrid](https://api.flutter.dev/flutter/widgets/SliverGrid-class.html) instead be a child of the [SliverPadding](https://api.flutter.dev/flutter/widgets/SliverPadding-class.html).

Once code has been ported to use [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html), other slivers, such as [SliverList](https://api.flutter.dev/flutter/widgets/SliverList-class.html) or [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html), can be put in the [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) list.

### Persisting the scroll position during a session

Scroll views attempt to persist their scroll position using [PageStorage](https://api.flutter.dev/flutter/widgets/PageStorage-class.html). This can be disabled by setting [ScrollController.keepScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/keepScrollOffset.html) to false on the [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html). If it is enabled, using a [PageStorageKey](https://api.flutter.dev/flutter/widgets/PageStorageKey-class.html) for the [key](https://api.flutter.dev/flutter/widgets/Widget/key.html) of this widget is recommended to help disambiguate different scroll views from each other.

### Padding

By default, [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) will automatically pad the limits of the grid's scrollable to avoid partial obstructions indicated by [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html)'s padding. To avoid this behavior, override with a zero [padding](https://api.flutter.dev/flutter/widgets/BoxScrollView/padding.html) property.

### Constructors

[GridView](https://api.flutter.dev/flutter/widgets/GridView/GridView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) gridDelegate, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId})

Creates a scrollable, 2D array of widgets with a custom [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html).

[GridView.builder](https://api.flutter.dev/flutter/widgets/GridView/GridView.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) gridDelegate, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? itemCount, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets that are created on demand.

[GridView.count](https://api.flutter.dev/flutter/widgets/GridView/GridView.count.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [int](https://api.flutter.dev/flutter/dart-core/int-class.html) crossAxisCount, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) mainAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) crossAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) childAspectRatio = 1.0, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets with a fixed number of tiles in the cross axis.

[GridView.custom](https://api.flutter.dev/flutter/widgets/GridView/GridView.custom.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) gridDelegate, required [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html) childrenDelegate, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets with both a custom [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html) and a custom [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html).

const

[GridView.extent](https://api.flutter.dev/flutter/widgets/GridView/GridView.extent.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxCrossAxisExtent, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) mainAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) crossAxisSpacing = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) childAspectRatio = 1.0, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, 2D array of widgets with tiles that each have a maximum cross-axis extent.

### Properties

[anchor](https://api.flutter.dev/flutter/widgets/ScrollView/anchor.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The relative position of the zero scroll offset.

finalinherited

[cacheExtent](https://api.flutter.dev/flutter/widgets/ScrollView/cacheExtent.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

The viewport has an area before and after the visible area to cache items that are about to become visible when the user scrolls.

finalinherited

[center](https://api.flutter.dev/flutter/widgets/ScrollView/center.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

The first child in the [GrowthDirection.forward](https://api.flutter.dev/flutter/rendering/GrowthDirection.html) growth direction.

finalinherited

[childrenDelegate](https://api.flutter.dev/flutter/widgets/GridView/childrenDelegate.html) → [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html)

A delegate that provides the children for the [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html).

final

[clipBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

finalinherited

[controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

An object that can be used to control the position to which this scroll view is scrolled.

finalinherited

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

finalinherited

[gridDelegate](https://api.flutter.dev/flutter/widgets/GridView/gridDelegate.html) → [SliverGridDelegate](https://api.flutter.dev/flutter/rendering/SliverGridDelegate-class.html)

A delegate that controls the layout of the children within the [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html).

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[keyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/keyboardDismissBehavior.html) → [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html)

[ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) the defines how this [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) will dismiss the keyboard automatically.

finalinherited

[padding](https://api.flutter.dev/flutter/widgets/BoxScrollView/padding.html) → [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)?

The amount of space by which to inset the children.

finalinherited

[physics](https://api.flutter.dev/flutter/widgets/ScrollView/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the scroll view should respond to user input.

finalinherited

[primary](https://api.flutter.dev/flutter/widgets/ScrollView/primary.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Whether this is the primary scroll view associated with the parent [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html).

finalinherited

[restorationId](https://api.flutter.dev/flutter/widgets/ScrollView/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

finalinherited

[reverse](https://api.flutter.dev/flutter/widgets/ScrollView/reverse.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the scroll view scrolls in the reading direction.

finalinherited

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/scrollBehavior.html) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)?

A [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) that will be applied to this widget individually.

finalinherited

[scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) along which the scroll view's offset increases.

finalinherited

[semanticChildCount](https://api.flutter.dev/flutter/widgets/ScrollView/semanticChildCount.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)?

The number of children that will contribute semantic information.

finalinherited

[shrinkWrap](https://api.flutter.dev/flutter/widgets/ScrollView/shrinkWrap.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the extent of the scroll view in the [scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html) should be determined by the contents being viewed.

finalinherited

## list view

A scrollable list of widgets arranged linearly.

[ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) is the most commonly used scrolling widget. It displays its children one after another in the scroll direction. In the cross axis, the children are required to fill the [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html).

If non-null, the [itemExtent](https://api.flutter.dev/flutter/widgets/ListView/itemExtent.html) forces the children to have the given extent in the scroll direction.

If non-null, the [prototypeItem](https://api.flutter.dev/flutter/widgets/ListView/prototypeItem.html) forces the children to have the same extent as the given widget in the scroll direction.

Specifying an [itemExtent](https://api.flutter.dev/flutter/widgets/ListView/itemExtent.html) or an [prototypeItem](https://api.flutter.dev/flutter/widgets/ListView/prototypeItem.html) is more efficient than letting the children determine their own extent because the scrolling machinery can make use of the foreknowledge of the children's extent to save work, for example when the scroll position changes drastically.

You can't specify both [itemExtent](https://api.flutter.dev/flutter/widgets/ListView/itemExtent.html) and [prototypeItem](https://api.flutter.dev/flutter/widgets/ListView/prototypeItem.html), only one or none of them.

There are four options for constructing a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html):

1. The default constructor takes an explicit [List<Widget>](https://api.flutter.dev/flutter/dart-core/List-class.html) of children. This constructor is appropriate for list views with a small number of children because constructing the [List](https://api.flutter.dev/flutter/dart-core/List-class.html) requires doing work for every child that could possibly be displayed in the list view instead of just those children that are actually visible.
2. The [ListView.builder](https://api.flutter.dev/flutter/widgets/ListView/ListView.builder.html) constructor takes an [IndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/IndexedWidgetBuilder.html), which builds the children on demand. This constructor is appropriate for list views with a large (or infinite) number of children because the builder is called only for those children that are actually visible.
3. The [ListView.separated](https://api.flutter.dev/flutter/widgets/ListView/ListView.separated.html) constructor takes two [IndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/IndexedWidgetBuilder.html)s: itemBuilder builds child items on demand, and separatorBuilder similarly builds separator children which appear in between the child items. This constructor is appropriate for list views with a fixed number of children.
4. The [ListView.custom](https://api.flutter.dev/flutter/widgets/ListView/ListView.custom.html) constructor takes a [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html), which provides the ability to customize additional aspects of the child model. For example, a [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html) can control the algorithm used to estimate the size of children that are not actually visible.

To control the initial scroll offset of the scroll view, provide a [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html) with its [ScrollController.initialScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/initialScrollOffset.html) property set.

By default, [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) will automatically pad the list's scrollable extremities to avoid partial obstructions indicated by [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html)'s padding. To avoid this behavior, override with a zero [padding](https://api.flutter.dev/flutter/widgets/BoxScrollView/padding.html) property.

### Creation

While laying out the list, visible children's elements, states and render objects will be created lazily based on existing widgets (such as when using the default constructor) or lazily provided ones (such as when using the [ListView.builder](https://api.flutter.dev/flutter/widgets/ListView/ListView.builder.html) constructor).

### Destruction

When a child is scrolled out of view, the associated element subtree, states and render objects are destroyed. A new child at the same position in the list will be lazily recreated along with new elements, states and render objects when it is scrolled back.

### Destruction mitigation

In order to preserve state as child elements are scrolled in and out of view, the following options are possible:

* Moving the ownership of non-trivial UI-state-driving business logic out of the list child subtree. For instance, if a list contains posts with their number of upvotes coming from a cached network response, store the list of posts and upvote number in a data model outside the list. Let the list child UI subtree be easily recreate-able from the source-of-truth model object. Use [StatefulWidget](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html)s in the child widget subtree to store instantaneous UI state only.
* Letting [KeepAlive](https://api.flutter.dev/flutter/widgets/KeepAlive-class.html) be the root widget of the list child widget subtree that needs to be preserved. The [KeepAlive](https://api.flutter.dev/flutter/widgets/KeepAlive-class.html) widget marks the child subtree's top render object child for keepalive. When the associated top render object is scrolled out of view, the list keeps the child's render object (and by extension, its associated elements and states) in a cache list instead of destroying them. When scrolled back into view, the render object is repainted as-is (if it wasn't marked dirty in the interim).

This only works if addAutomaticKeepAlives and addRepaintBoundaries are false since those parameters cause the [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) to wrap each child widget subtree with other widgets.

* Using [AutomaticKeepAlive](https://api.flutter.dev/flutter/widgets/AutomaticKeepAlive-class.html) widgets (inserted by default when addAutomaticKeepAlives is true). [AutomaticKeepAlive](https://api.flutter.dev/flutter/widgets/AutomaticKeepAlive-class.html) allows descendant widgets to control whether the subtree is actually kept alive or not. This behavior is in contrast with [KeepAlive](https://api.flutter.dev/flutter/widgets/KeepAlive-class.html), which will unconditionally keep the subtree alive.

As an example, the [EditableText](https://api.flutter.dev/flutter/widgets/EditableText-class.html) widget signals its list child element subtree to stay alive while its text field has input focus. If it doesn't have focus and no other descendants signaled for keepalive via a [KeepAliveNotification](https://api.flutter.dev/flutter/widgets/KeepAliveNotification-class.html), the list child element subtree will be destroyed when scrolled away.

[AutomaticKeepAlive](https://api.flutter.dev/flutter/widgets/AutomaticKeepAlive-class.html) descendants typically signal it to be kept alive by using the [AutomaticKeepAliveClientMixin](https://api.flutter.dev/flutter/widgets/AutomaticKeepAliveClientMixin-mixin.html), then implementing the [AutomaticKeepAliveClientMixin.wantKeepAlive](https://api.flutter.dev/flutter/widgets/AutomaticKeepAliveClientMixin/wantKeepAlive.html) getter and calling [AutomaticKeepAliveClientMixin.updateKeepAlive](https://api.flutter.dev/flutter/widgets/AutomaticKeepAliveClientMixin/updateKeepAlive.html).

### Transitioning to [**CustomScrollView**](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html)

A [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) is basically a [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) with a single [SliverList](https://api.flutter.dev/flutter/widgets/SliverList-class.html) in its [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) property.

If [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) is no longer sufficient, for example because the scroll view is to have both a list and a grid, or because the list is to be combined with a [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html), etc, it is straight-forward to port code from using [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) to using [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) directly.

The [key](https://api.flutter.dev/flutter/widgets/Widget/key.html), [scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html), [reverse](https://api.flutter.dev/flutter/widgets/ScrollView/reverse.html), [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html), [primary](https://api.flutter.dev/flutter/widgets/ScrollView/primary.html), [physics](https://api.flutter.dev/flutter/widgets/ScrollView/physics.html), and [shrinkWrap](https://api.flutter.dev/flutter/widgets/ScrollView/shrinkWrap.html) properties on [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) map directly to the identically named properties on [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html).

The [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) property should be a list containing either:

* a [SliverList](https://api.flutter.dev/flutter/widgets/SliverList-class.html) if both [itemExtent](https://api.flutter.dev/flutter/widgets/ListView/itemExtent.html) and [prototypeItem](https://api.flutter.dev/flutter/widgets/ListView/prototypeItem.html) were null;
* a [SliverFixedExtentList](https://api.flutter.dev/flutter/widgets/SliverFixedExtentList-class.html) if [itemExtent](https://api.flutter.dev/flutter/widgets/ListView/itemExtent.html) was not null; or
* a [SliverPrototypeExtentList](https://api.flutter.dev/flutter/widgets/SliverPrototypeExtentList-class.html) if [prototypeItem](https://api.flutter.dev/flutter/widgets/ListView/prototypeItem.html) was not null.

The [childrenDelegate](https://api.flutter.dev/flutter/widgets/ListView/childrenDelegate.html) property on [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) corresponds to the [SliverList.delegate](https://api.flutter.dev/flutter/widgets/SliverMultiBoxAdaptorWidget/delegate.html) (or [SliverFixedExtentList.delegate](https://api.flutter.dev/flutter/widgets/SliverMultiBoxAdaptorWidget/delegate.html)) property. The [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) constructor's children argument corresponds to the [childrenDelegate](https://api.flutter.dev/flutter/widgets/ListView/childrenDelegate.html) being a [SliverChildListDelegate](https://api.flutter.dev/flutter/widgets/SliverChildListDelegate-class.html) with that same argument. The [ListView.builder](https://api.flutter.dev/flutter/widgets/ListView/ListView.builder.html) constructor's itemBuilder and itemCount arguments correspond to the [childrenDelegate](https://api.flutter.dev/flutter/widgets/ListView/childrenDelegate.html) being a [SliverChildBuilderDelegate](https://api.flutter.dev/flutter/widgets/SliverChildBuilderDelegate-class.html) with the equivalent arguments.

The [padding](https://api.flutter.dev/flutter/widgets/BoxScrollView/padding.html) property corresponds to having a [SliverPadding](https://api.flutter.dev/flutter/widgets/SliverPadding-class.html) in the [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) property instead of the list itself, and having the [SliverList](https://api.flutter.dev/flutter/widgets/SliverList-class.html) instead be a child of the [SliverPadding](https://api.flutter.dev/flutter/widgets/SliverPadding-class.html).

[CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html)s don't automatically avoid obstructions from [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html) like [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html)s do. To reproduce the behavior, wrap the slivers in [SliverSafeArea](https://api.flutter.dev/flutter/widgets/SliverSafeArea-class.html)s.

Once code has been ported to use [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html), other slivers, such as [SliverGrid](https://api.flutter.dev/flutter/widgets/SliverGrid-class.html) or [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html), can be put in the [CustomScrollView.slivers](https://api.flutter.dev/flutter/widgets/CustomScrollView/slivers.html) list.

### Special handling for an empty list

A common design pattern is to have a custom UI for an empty list. The best way to achieve this in Flutter is just conditionally replacing the [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) at build time with whatever widgets you need to show for the empty list state:

### Selection of list items

[ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) has no built-in notion of a selected item or items. For a small example of how a caller might wire up basic item selection, see [ListTile.selected](https://api.flutter.dev/flutter/material/ListTile/selected.html).

[ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html)s are often decorated with [Scrollbar](https://api.flutter.dev/flutter/material/Scrollbar-class.html)s and overscroll indicators, which are managed by the inherited [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html). Placing a [ScrollConfiguration](https://api.flutter.dev/flutter/widgets/ScrollConfiguration-class.html) above a ScrollView can modify these behaviors for that ScrollView, or can be managed app-wide by providing a ScrollBehavior to [MaterialApp.scrollBehavior](https://api.flutter.dev/flutter/material/MaterialApp/scrollBehavior.html) or [CupertinoApp.scrollBehavior](https://api.flutter.dev/flutter/cupertino/CupertinoApp/scrollBehavior.html).

### Persisting the scroll position during a session

Scroll views attempt to persist their scroll position using [PageStorage](https://api.flutter.dev/flutter/widgets/PageStorage-class.html). This can be disabled by setting [ScrollController.keepScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/keepScrollOffset.html) to false on the [controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html). If it is enabled, using a [PageStorageKey](https://api.flutter.dev/flutter/widgets/PageStorageKey-class.html) for the [key](https://api.flutter.dev/flutter/widgets/Widget/key.html) of this widget is recommended to help disambiguate different scroll views from each other.

See also:

* [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html), which is a scrollable widget that has a single child.
* [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html), which is a scrolling list of child widgets that are each the size of the viewport.
* [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html), which is a scrollable, 2D array of widgets.
* [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html), which is a scrollable widget that creates custom scroll effects using slivers.
* [ListBody](https://api.flutter.dev/flutter/widgets/ListBody-class.html), which arranges its children in a similar manner, but without scrolling.
* [ScrollNotification](https://api.flutter.dev/flutter/widgets/ScrollNotification-class.html) and [NotificationListener](https://api.flutter.dev/flutter/widgets/NotificationListener-class.html), which can be used to watch the scroll position without using a [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html).
* The [catalog of layout widgets](https://flutter.dev/widgets/layout/).
* Cookbook: [Use lists](https://flutter.dev/docs/cookbook/lists/basic-list)
* Cookbook: [Work with long lists](https://flutter.dev/docs/cookbook/lists/long-lists)
* Cookbook: [Create a horizontal list](https://flutter.dev/docs/cookbook/lists/horizontal-list)
* Cookbook: [Create lists with different types of items](https://flutter.dev/docs/cookbook/lists/mixed-list)
* Cookbook: [Implement swipe to dismiss](https://flutter.dev/docs/cookbook/gestures/dismissible)

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [StatelessWidget](https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html)
* [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html)
* [BoxScrollView](https://api.flutter.dev/flutter/widgets/BoxScrollView-class.html)
* ListView

### Constructors

[ListView](https://api.flutter.dev/flutter/widgets/ListView/ListView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [ItemExtentBuilder](https://api.flutter.dev/flutter/rendering/ItemExtentBuilder.html)? itemExtentBuilder, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, linear array of widgets from an explicit [List](https://api.flutter.dev/flutter/dart-core/List-class.html).

[ListView.builder](https://api.flutter.dev/flutter/widgets/ListView/ListView.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [ItemExtentBuilder](https://api.flutter.dev/flutter/rendering/ItemExtentBuilder.html)? itemExtentBuilder, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? itemCount, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, linear array of widgets that are created on demand.

[ListView.custom](https://api.flutter.dev/flutter/widgets/ListView/ListView.custom.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, [ItemExtentBuilder](https://api.flutter.dev/flutter/rendering/ItemExtentBuilder.html)? itemExtentBuilder, required [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html) childrenDelegate, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a scrollable, linear array of widgets with a custom child model.

const

[ListView.separated](https://api.flutter.dev/flutter/widgets/ListView/ListView.separated.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, required [IndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/IndexedWidgetBuilder.html) separatorBuilder, required [int](https://api.flutter.dev/flutter/dart-core/int-class.html) itemCount, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addAutomaticKeepAlives = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addRepaintBoundaries = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) addSemanticIndexes = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a fixed-length scrollable linear array of list "items" separated by list item "separators".

### Properties

[anchor](https://api.flutter.dev/flutter/widgets/ScrollView/anchor.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The relative position of the zero scroll offset.

finalinherited

[cacheExtent](https://api.flutter.dev/flutter/widgets/ScrollView/cacheExtent.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

The viewport has an area before and after the visible area to cache items that are about to become visible when the user scrolls.

finalinherited

[center](https://api.flutter.dev/flutter/widgets/ScrollView/center.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

The first child in the [GrowthDirection.forward](https://api.flutter.dev/flutter/rendering/GrowthDirection.html) growth direction.

finalinherited

[childrenDelegate](https://api.flutter.dev/flutter/widgets/ListView/childrenDelegate.html) → [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html)

A delegate that provides the children for the [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html).

final

[clipBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

finalinherited

[controller](https://api.flutter.dev/flutter/widgets/ScrollView/controller.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

An object that can be used to control the position to which this scroll view is scrolled.

finalinherited

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

finalinherited

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[itemExtent](https://api.flutter.dev/flutter/widgets/ListView/itemExtent.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

If non-null, forces the children to have the given extent in the scroll direction.

final

[itemExtentBuilder](https://api.flutter.dev/flutter/widgets/ListView/itemExtentBuilder.html) → [ItemExtentBuilder](https://api.flutter.dev/flutter/rendering/ItemExtentBuilder.html)?

If non-null, forces the children to have the corresponding extent returned by the builder.

final

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[keyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/keyboardDismissBehavior.html) → [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html)

[ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) the defines how this [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) will dismiss the keyboard automatically.

finalinherited

[padding](https://api.flutter.dev/flutter/widgets/BoxScrollView/padding.html) → [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)?

The amount of space by which to inset the children.

finalinherited

[physics](https://api.flutter.dev/flutter/widgets/ScrollView/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the scroll view should respond to user input.

finalinherited

[primary](https://api.flutter.dev/flutter/widgets/ScrollView/primary.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Whether this is the primary scroll view associated with the parent [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html).

finalinherited

[prototypeItem](https://api.flutter.dev/flutter/widgets/ListView/prototypeItem.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)?

If non-null, forces the children to have the same extent as the given widget in the scroll direction.

final

[restorationId](https://api.flutter.dev/flutter/widgets/ScrollView/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

finalinherited

[reverse](https://api.flutter.dev/flutter/widgets/ScrollView/reverse.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the scroll view scrolls in the reading direction.

finalinherited

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollView/scrollBehavior.html) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)?

A [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) that will be applied to this widget individually.

finalinherited

[scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) along which the scroll view's offset increases.

finalinherited

[semanticChildCount](https://api.flutter.dev/flutter/widgets/ScrollView/semanticChildCount.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)?

The number of children that will contribute semantic information.

finalinherited

[shrinkWrap](https://api.flutter.dev/flutter/widgets/ScrollView/shrinkWrap.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the extent of the scroll view in the [scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html) should be determined by the contents being viewed.

finalinherited

## nested scroll view

A scrolling view inside of which can be nested other scrolling views, with their scroll positions being intrinsically linked.

The most common use case for this widget is a scrollable view with a flexible [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) containing a [TabBar](https://api.flutter.dev/flutter/material/TabBar-class.html) in the header (built by [headerSliverBuilder](https://api.flutter.dev/flutter/widgets/NestedScrollView/headerSliverBuilder.html), and with a [TabBarView](https://api.flutter.dev/flutter/material/TabBarView-class.html) in the [body](https://api.flutter.dev/flutter/widgets/NestedScrollView/body.html), such that the scrollable view's contents vary based on which tab is visible.

### Motivation

In a normal [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html), there is one set of slivers (the components of the scrolling view). If one of those slivers hosted a [TabBarView](https://api.flutter.dev/flutter/material/TabBarView-class.html) which scrolls in the opposite direction (e.g. allowing the user to swipe horizontally between the pages represented by the tabs, while the list scrolls vertically), then any list inside that [TabBarView](https://api.flutter.dev/flutter/material/TabBarView-class.html) would not interact with the outer [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html). For example, flinging the inner list to scroll to the top would not cause a collapsed [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) in the outer [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) to expand.

[NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html) solves this problem by providing custom [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)s for the outer [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) and the inner [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html)s (those inside the [TabBarView](https://api.flutter.dev/flutter/material/TabBarView-class.html), hooking them together so that they appear, to the user, as one coherent scroll view.

### [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html)s with [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html)s

Using a [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) in the outer scroll view, or [headerSliverBuilder](https://api.flutter.dev/flutter/widgets/NestedScrollView/headerSliverBuilder.html), of a [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html) may require special configurations in order to work as it would if the outer and inner were one single scroll view, like a [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html).

### Pinned [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html)s

A pinned [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) works in a [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html) exactly as it would in another scroll view, like [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html). When using [SliverAppBar.pinned](https://api.flutter.dev/flutter/material/SliverAppBar/pinned.html), the app bar remains visible at the top of the scroll view. The app bar can still expand and contract as the user scrolls, but it will remain visible rather than being scrolled out of view.

This works naturally in a [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html), as the pinned [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) is not expected to move in or out of the visible portion of the viewport. As the inner or outer [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html)s are moved, the app bar persists as expected.

If the app bar is floating, pinned, and using an expanded height, follow the floating convention laid out below.

### Floating [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html)s

When placed in the outer scrollable, or the [headerSliverBuilder](https://api.flutter.dev/flutter/widgets/NestedScrollView/headerSliverBuilder.html), a [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) that floats, using [SliverAppBar.floating](https://api.flutter.dev/flutter/material/SliverAppBar/floating.html) will not be triggered to float over the inner scroll view, or [body](https://api.flutter.dev/flutter/widgets/NestedScrollView/body.html), automatically.

This is because a floating app bar uses the scroll offset of its own [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) to dictate the floating action. Being two separate inner and outer [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html)s, a [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) in the outer header is not aware of changes in the scroll offset of the inner body.

In order to float the outer, use [NestedScrollView.floatHeaderSlivers](https://api.flutter.dev/flutter/widgets/NestedScrollView/floatHeaderSlivers.html). When set to true, the nested scrolling coordinator will prioritize floating in the header slivers before applying the remaining drag to the body.

Furthermore, the floatHeaderSlivers flag should also be used when using an app bar that is floating, pinned, and has an expanded height. In this configuration, the flexible space of the app bar will open and collapse, while the primary portion of the app bar remains pinned.

### Snapping [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html)s

Floating [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html)s also have the option to perform a snapping animation. If [SliverAppBar.snap](https://api.flutter.dev/flutter/material/SliverAppBar/snap.html) is true, then a scroll that exposes the floating app bar will trigger an animation that slides the entire app bar into view. Similarly if a scroll dismisses the app bar, the animation will slide the app bar completely out of view.

It is possible with a [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html) to perform just the snapping animation without floating the app bar in and out. By not using the [NestedScrollView.floatHeaderSlivers](https://api.flutter.dev/flutter/widgets/NestedScrollView/floatHeaderSlivers.html), the app bar will snap in and out without floating.

The [SliverAppBar.snap](https://api.flutter.dev/flutter/material/SliverAppBar/snap.html) animation should be used in conjunction with the [SliverOverlapAbsorber](https://api.flutter.dev/flutter/widgets/SliverOverlapAbsorber-class.html) and [SliverOverlapInjector](https://api.flutter.dev/flutter/widgets/SliverOverlapInjector-class.html) widgets when implemented in a [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html). These widgets take any overlapping behavior of the [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) in the header and redirect it to the [SliverOverlapInjector](https://api.flutter.dev/flutter/widgets/SliverOverlapInjector-class.html) in the body. If it is missing, then it is possible for the nested "inner" scroll view below to end up under the [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html) even when the inner scroll view thinks it has not been scrolled.

### Snapping and Floating [**SliverAppBar**](https://api.flutter.dev/flutter/material/SliverAppBar-class.html)s

Currently, [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html) does not support simultaneously floating and snapping the outer scrollable, e.g. when using [SliverAppBar.floating](https://api.flutter.dev/flutter/material/SliverAppBar/floating.html) & [SliverAppBar.snap](https://api.flutter.dev/flutter/material/SliverAppBar/snap.html) at the same time.

### Stretching [**SliverAppBar**](https://api.flutter.dev/flutter/material/SliverAppBar-class.html)s

Currently, [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html) does not support stretching the outer scrollable, e.g. when using [SliverAppBar.stretch](https://api.flutter.dev/flutter/material/SliverAppBar/stretch.html).

See also:

* [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html), for examples on different configurations like floating, pinned and snap behaviors.
* [SliverOverlapAbsorber](https://api.flutter.dev/flutter/widgets/SliverOverlapAbsorber-class.html), a sliver that wraps another, forcing its layout extent to be treated as overlap.
* [SliverOverlapInjector](https://api.flutter.dev/flutter/widgets/SliverOverlapInjector-class.html), a sliver that has a sliver geometry based on the values stored in a [SliverOverlapAbsorberHandle](https://api.flutter.dev/flutter/widgets/SliverOverlapAbsorberHandle-class.html).

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [StatefulWidget](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html)
* NestedScrollView

### Constructors

[NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView/NestedScrollView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, required [NestedScrollViewHeaderSliversBuilder](https://api.flutter.dev/flutter/widgets/NestedScrollViewHeaderSliversBuilder.html) headerSliverBuilder, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) body, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) floatHeaderSlivers = false, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior})

Creates a nested scroll view.

const

### Properties

[body](https://api.flutter.dev/flutter/widgets/NestedScrollView/body.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

The widget to show inside the [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html).

final

[clipBehavior](https://api.flutter.dev/flutter/widgets/NestedScrollView/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

final

[controller](https://api.flutter.dev/flutter/widgets/NestedScrollView/controller.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

An object that can be used to control the position to which the outer scroll view is scrolled.

final

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/NestedScrollView/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

final

[floatHeaderSlivers](https://api.flutter.dev/flutter/widgets/NestedScrollView/floatHeaderSlivers.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether or not the [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html)'s coordinator should prioritize the outer scrollable over the inner when scrolling back.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[headerSliverBuilder](https://api.flutter.dev/flutter/widgets/NestedScrollView/headerSliverBuilder.html) → [NestedScrollViewHeaderSliversBuilder](https://api.flutter.dev/flutter/widgets/NestedScrollViewHeaderSliversBuilder.html)

A builder for any widgets that are to precede the inner scroll views (as given by [body](https://api.flutter.dev/flutter/widgets/NestedScrollView/body.html)).

final

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[physics](https://api.flutter.dev/flutter/widgets/NestedScrollView/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the scroll view should respond to user input.

final

[restorationId](https://api.flutter.dev/flutter/widgets/NestedScrollView/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

final

[reverse](https://api.flutter.dev/flutter/widgets/NestedScrollView/reverse.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the scroll view scrolls in the reading direction.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollBehavior](https://api.flutter.dev/flutter/widgets/NestedScrollView/scrollBehavior.html) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)?

A [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) that will be applied to this widget individually.

final

[scrollDirection](https://api.flutter.dev/flutter/widgets/NestedScrollView/scrollDirection.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) along which the scroll view's offset increases.

final

## notification listener

A widget that listens for [Notification](https://api.flutter.dev/flutter/widgets/Notification-class.html)s bubbling up the tree.

Notifications will trigger the [onNotification](https://api.flutter.dev/flutter/widgets/NotificationListener/onNotification.html) callback only if their [runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) is a subtype of T.

To dispatch notifications, use the [Notification.dispatch](https://api.flutter.dev/flutter/widgets/Notification/dispatch.html) method.

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [ProxyWidget](https://api.flutter.dev/flutter/widgets/ProxyWidget-class.html)
* NotificationListener

### Constructors

[NotificationListener](https://api.flutter.dev/flutter/widgets/NotificationListener/NotificationListener.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, [NotificationListenerCallback](https://api.flutter.dev/flutter/widgets/NotificationListenerCallback.html)<T>? onNotification})

Creates a widget that listens for notifications.

const

### Properties

[child](https://api.flutter.dev/flutter/widgets/ProxyWidget/child.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

The widget below this widget in the tree.

finalinherited

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[onNotification](https://api.flutter.dev/flutter/widgets/NotificationListener/onNotification.html) → [NotificationListenerCallback](https://api.flutter.dev/flutter/widgets/NotificationListenerCallback.html)<T>?

Called when a notification of the appropriate type arrives at this location in the tree.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

### Methods

[createElement](https://api.flutter.dev/flutter/widgets/NotificationListener/createElement.html)() → [Element](https://api.flutter.dev/flutter/widgets/Element-class.html)

Inflates this configuration to a concrete instance.

override

[debugDescribeChildren](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)() → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)>

Returns a list of DiagnosticsNode objects describing this node's children.

inherited

[debugFillProperties](https://api.flutter.dev/flutter/widgets/Widget/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://api.flutter.dev/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

Add additional properties associated with the node.

inherited

[noSuchMethod](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a nonexistent method or property is accessed.

inherited

[toDiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html)? name, [DiagnosticsTreeStyle](https://api.flutter.dev/flutter/foundation/DiagnosticsTreeStyle.html)? style}) → [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [DiagnosticsNode.toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticsNode/toStringDeep.html).

inherited

[toString](https://api.flutter.dev/flutter/foundation/Diagnosticable/toString.html)({[DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.info}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A string representation of this object.

inherited

[toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixLineOne = '', [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? prefixOtherLines, [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants.

inherited

[toStringShallow](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) joiner = ', ', [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object.

inherited

[toStringShort](https://api.flutter.dev/flutter/widgets/Widget/toStringShort.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A short, textual description of this widget.

inherited

### Operators

[operator ==](https://api.flutter.dev/flutter/widgets/Widget/operator_equals.html)([Object](https://api.flutter.dev/flutter/dart-core/Object-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

The equality operator.

inherited

## page view

A scrollable list that works page by page.

Each child of a page view is forced to be the same size as the viewport.

You can use a [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html) to control which page is visible in the view. In addition to being able to control the pixel offset of the content inside the [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html), a [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html) also lets you control the offset in terms of pages, which are increments of the viewport size.

The [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html) can also be used to control the [PageController.initialPage](https://api.flutter.dev/flutter/widgets/PageController/initialPage.html), which determines which page is shown when the [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html) is first constructed, and the [PageController.viewportFraction](https://api.flutter.dev/flutter/widgets/PageController/viewportFraction.html), which determines the size of the pages as a fraction of the viewport size.

### Persisting the scroll position during a session

Scroll views attempt to persist their scroll position using [PageStorage](https://api.flutter.dev/flutter/widgets/PageStorage-class.html). For a [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html), this can be disabled by setting [PageController.keepPage](https://api.flutter.dev/flutter/widgets/PageController/keepPage.html) to false on the [controller](https://api.flutter.dev/flutter/widgets/PageView/controller.html). If it is enabled, using a [PageStorageKey](https://api.flutter.dev/flutter/widgets/PageStorageKey-class.html) for the [key](https://api.flutter.dev/flutter/widgets/Widget/key.html) of this widget is recommended to help disambiguate different scroll views from each other.

See also:

* [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html), which controls which page is visible in the view.
* [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html), when you need to make a single child scrollable.
* [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html), for a scrollable list of boxes.
* [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html), for a scrollable grid of boxes.
* [ScrollNotification](https://api.flutter.dev/flutter/widgets/ScrollNotification-class.html) and [NotificationListener](https://api.flutter.dev/flutter/widgets/NotificationListener-class.html), which can be used to watch the scroll position without using a [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html).

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [StatefulWidget](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html)
* PageView

### Constructors

[PageView](https://api.flutter.dev/flutter/widgets/PageView/PageView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.horizontal, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html)? controller, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) pageSnapping = true, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onPageChanged, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[], [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) allowImplicitScrolling = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) padEnds = true})

Creates a scrollable list that works page by page from an explicit [List](https://api.flutter.dev/flutter/dart-core/List-class.html) of widgets.

[PageView.builder](https://api.flutter.dev/flutter/widgets/PageView/PageView.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.horizontal, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html)? controller, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) pageSnapping = true, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onPageChanged, required [NullableIndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/NullableIndexedWidgetBuilder.html) itemBuilder, [ChildIndexGetter](https://api.flutter.dev/flutter/widgets/ChildIndexGetter.html)? findChildIndexCallback, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? itemCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) allowImplicitScrolling = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) padEnds = true})

Creates a scrollable list that works page by page using widgets that are created on demand.

[PageView.custom](https://api.flutter.dev/flutter/widgets/PageView/PageView.custom.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.horizontal, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html)? controller, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) pageSnapping = true, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>? onPageChanged, required [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html) childrenDelegate, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) allowImplicitScrolling = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) padEnds = true})

Creates a scrollable list that works page by page with a custom child model.

### Properties

[allowImplicitScrolling](https://api.flutter.dev/flutter/widgets/PageView/allowImplicitScrolling.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Controls whether the widget's pages will respond to [RenderObject.showOnScreen](https://api.flutter.dev/flutter/rendering/RenderObject/showOnScreen.html), which will allow for implicit accessibility scrolling.

final

[childrenDelegate](https://api.flutter.dev/flutter/widgets/PageView/childrenDelegate.html) → [SliverChildDelegate](https://api.flutter.dev/flutter/widgets/SliverChildDelegate-class.html)

A delegate that provides the children for the [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html).

final

[clipBehavior](https://api.flutter.dev/flutter/widgets/PageView/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

final

[controller](https://api.flutter.dev/flutter/widgets/PageView/controller.html) → [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html)

An object that can be used to control the position to which this page view is scrolled.

final

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/PageView/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[onPageChanged](https://api.flutter.dev/flutter/widgets/PageView/onPageChanged.html) → [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[int](https://api.flutter.dev/flutter/dart-core/int-class.html)>?

Called whenever the page in the center of the viewport changes.

final

[padEnds](https://api.flutter.dev/flutter/widgets/PageView/padEnds.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether to add padding to both ends of the list.

final

[pageSnapping](https://api.flutter.dev/flutter/widgets/PageView/pageSnapping.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Set to false to disable page snapping, useful for custom scroll behavior.

final

[physics](https://api.flutter.dev/flutter/widgets/PageView/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the page view should respond to user input.

final

[restorationId](https://api.flutter.dev/flutter/widgets/PageView/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

final

[reverse](https://api.flutter.dev/flutter/widgets/PageView/reverse.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the page view scrolls in the reading direction.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollBehavior](https://api.flutter.dev/flutter/widgets/PageView/scrollBehavior.html) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)?

A [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) that will be applied to this widget individually.

final

[scrollDirection](https://api.flutter.dev/flutter/widgets/PageView/scrollDirection.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) along which the scroll view's offset increases with each page.

final

### Methods

[createElement](https://api.flutter.dev/flutter/widgets/StatefulWidget/createElement.html)() → [StatefulElement](https://api.flutter.dev/flutter/widgets/StatefulElement-class.html)

Creates a [StatefulElement](https://api.flutter.dev/flutter/widgets/StatefulElement-class.html) to manage this widget's location in the tree.

inherited

[createState](https://api.flutter.dev/flutter/widgets/PageView/createState.html)() → [State](https://api.flutter.dev/flutter/widgets/State-class.html)<[PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html)>

Creates the mutable state for this widget at a given location in the tree.

override

[debugDescribeChildren](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)() → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)>

Returns a list of DiagnosticsNode objects describing this node's children.

inherited

[debugFillProperties](https://api.flutter.dev/flutter/widgets/Widget/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://api.flutter.dev/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

Add additional properties associated with the node.

inherited

[noSuchMethod](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a nonexistent method or property is accessed.

inherited

[toDiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html)? name, [DiagnosticsTreeStyle](https://api.flutter.dev/flutter/foundation/DiagnosticsTreeStyle.html)? style}) → [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [DiagnosticsNode.toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticsNode/toStringDeep.html).

inherited

[toString](https://api.flutter.dev/flutter/foundation/Diagnosticable/toString.html)({[DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.info}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A string representation of this object.

inherited

[toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixLineOne = '', [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? prefixOtherLines, [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants.

inherited

[toStringShallow](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) joiner = ', ', [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object.

inherited

[toStringShort](https://api.flutter.dev/flutter/widgets/Widget/toStringShort.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A short, textual description of this widget.

inherited

### Operators

[operator ==](https://api.flutter.dev/flutter/widgets/Widget/operator_equals.html)([Object](https://api.flutter.dev/flutter/dart-core/Object-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

The equality operator.

inherited

## refreshed indicator

A widget that supports the Material "swipe to refresh" idiom.

When the child's [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) descendant overscrolls, an animated circular progress indicator is faded into view. When the scroll ends, if the indicator has been dragged far enough for it to become completely opaque, the [onRefresh](https://api.flutter.dev/flutter/material/RefreshIndicator/onRefresh.html) callback is called. The callback is expected to update the scrollable's contents and then complete the [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html) it returns. The refresh indicator disappears after the callback's [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html) has completed.

The trigger mode is configured by [RefreshIndicator.triggerMode](https://api.flutter.dev/flutter/material/RefreshIndicator/triggerMode.html).

### Refresh indicator does not show up

The [RefreshIndicator](https://api.flutter.dev/flutter/material/RefreshIndicator-class.html) will appear if its scrollable descendant can be overscrolled, i.e. if the scrollable's content is bigger than its viewport. To ensure that the [RefreshIndicator](https://api.flutter.dev/flutter/material/RefreshIndicator-class.html) will always appear, even if the scrollable's content fits within its viewport, set the scrollable's [Scrollable.physics](https://api.flutter.dev/flutter/widgets/Scrollable/physics.html) property to [AlwaysScrollableScrollPhysics](https://api.flutter.dev/flutter/widgets/AlwaysScrollableScrollPhysics-class.html):

ListView(

physics: const AlwaysScrollableScrollPhysics(),

// ...

)

A [RefreshIndicator](https://api.flutter.dev/flutter/material/RefreshIndicator-class.html) can only be used with a vertical scroll view.

See also:

* [material.io/design/platform-guidance/android-swipe-to-refresh.html](https://material.io/design/platform-guidance/android-swipe-to-refresh.html)
* [RefreshIndicatorState](https://api.flutter.dev/flutter/material/RefreshIndicatorState-class.html), can be used to programmatically show the refresh indicator.
* [RefreshProgressIndicator](https://api.flutter.dev/flutter/material/RefreshProgressIndicator-class.html), widget used by [RefreshIndicator](https://api.flutter.dev/flutter/material/RefreshIndicator-class.html) to show the inner circular progress spinner during refreshes.
* [CupertinoSliverRefreshControl](https://api.flutter.dev/flutter/cupertino/CupertinoSliverRefreshControl-class.html), an iOS equivalent of the pull-to-refresh pattern. Must be used as a sliver inside a [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) instead of wrapping around a [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) because it's a part of the scrollable instead of being overlaid on top of it.

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [StatefulWidget](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html)
* RefreshIndicator

### Constructors

[RefreshIndicator](https://api.flutter.dev/flutter/material/RefreshIndicator/RefreshIndicator.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) displacement = 40.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) edgeOffset = 0.0, required [RefreshCallback](https://api.flutter.dev/flutter/material/RefreshCallback.html) onRefresh, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [ScrollNotificationPredicate](https://api.flutter.dev/flutter/widgets/ScrollNotificationPredicate.html) notificationPredicate = defaultScrollNotificationPredicate, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsLabel, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsValue, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) strokeWidth = RefreshProgressIndicator.defaultStrokeWidth, [RefreshIndicatorTriggerMode](https://api.flutter.dev/flutter/material/RefreshIndicatorTriggerMode.html) triggerMode = RefreshIndicatorTriggerMode.onEdge})

Creates a refresh indicator.

const

[RefreshIndicator.adaptive](https://api.flutter.dev/flutter/material/RefreshIndicator/RefreshIndicator.adaptive.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) displacement = 40.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) edgeOffset = 0.0, required [RefreshCallback](https://api.flutter.dev/flutter/material/RefreshCallback.html) onRefresh, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [ScrollNotificationPredicate](https://api.flutter.dev/flutter/widgets/ScrollNotificationPredicate.html) notificationPredicate = defaultScrollNotificationPredicate, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsLabel, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsValue, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) strokeWidth = RefreshProgressIndicator.defaultStrokeWidth, [RefreshIndicatorTriggerMode](https://api.flutter.dev/flutter/material/RefreshIndicatorTriggerMode.html) triggerMode = RefreshIndicatorTriggerMode.onEdge})

Creates an adaptive [RefreshIndicator](https://api.flutter.dev/flutter/material/RefreshIndicator-class.html) based on whether the target platform is iOS or macOS, following Material design's [Cross-platform guidelines](https://material.io/design/platform-guidance/cross-platform-adaptation.html).

const

### Properties

[backgroundColor](https://api.flutter.dev/flutter/material/RefreshIndicator/backgroundColor.html) → [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?

The progress indicator's background color. The current theme's [ThemeData.canvasColor](https://api.flutter.dev/flutter/material/ThemeData/canvasColor.html) by default.

final

[child](https://api.flutter.dev/flutter/material/RefreshIndicator/child.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

The widget below this widget in the tree.

final

[color](https://api.flutter.dev/flutter/material/RefreshIndicator/color.html) → [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?

The progress indicator's foreground color. The current theme's [ColorScheme.primary](https://api.flutter.dev/flutter/material/ColorScheme/primary.html) by default.

final

[displacement](https://api.flutter.dev/flutter/material/RefreshIndicator/displacement.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The distance from the child's top or bottom [edgeOffset](https://api.flutter.dev/flutter/material/RefreshIndicator/edgeOffset.html) where the refresh indicator will settle. During the drag that exposes the refresh indicator, its actual displacement may significantly exceed this value.

final

[edgeOffset](https://api.flutter.dev/flutter/material/RefreshIndicator/edgeOffset.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The offset where [RefreshProgressIndicator](https://api.flutter.dev/flutter/material/RefreshProgressIndicator-class.html) starts to appear on drag start.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[notificationPredicate](https://api.flutter.dev/flutter/material/RefreshIndicator/notificationPredicate.html) → [ScrollNotificationPredicate](https://api.flutter.dev/flutter/widgets/ScrollNotificationPredicate.html)

A check that specifies whether a [ScrollNotification](https://api.flutter.dev/flutter/widgets/ScrollNotification-class.html) should be handled by this widget.

final

[onRefresh](https://api.flutter.dev/flutter/material/RefreshIndicator/onRefresh.html) → [RefreshCallback](https://api.flutter.dev/flutter/material/RefreshCallback.html)

A function that's called when the user has dragged the refresh indicator far enough to demonstrate that they want the app to refresh. The returned [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html) must complete when the refresh operation is finished.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[semanticsLabel](https://api.flutter.dev/flutter/material/RefreshIndicator/semanticsLabel.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

The [SemanticsProperties.label](https://api.flutter.dev/flutter/semantics/SemanticsProperties/label.html) for this progress indicator.

final

[semanticsValue](https://api.flutter.dev/flutter/material/RefreshIndicator/semanticsValue.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

The [SemanticsProperties.value](https://api.flutter.dev/flutter/semantics/SemanticsProperties/value.html) for this progress indicator.

final

[strokeWidth](https://api.flutter.dev/flutter/material/RefreshIndicator/strokeWidth.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

Defines [strokeWidth](https://api.flutter.dev/flutter/material/RefreshIndicator/strokeWidth.html) for RefreshIndicator.

final

[triggerMode](https://api.flutter.dev/flutter/material/RefreshIndicator/triggerMode.html) → [RefreshIndicatorTriggerMode](https://api.flutter.dev/flutter/material/RefreshIndicatorTriggerMode.html)

Defines how this [RefreshIndicator](https://api.flutter.dev/flutter/material/RefreshIndicator-class.html) can be triggered when users overscroll.

final

## reorderable list view

A list whose items the user can interactively reorder by dragging.

This sample shows by dragging the user can reorder the items of the list. The [onReorder](https://api.flutter.dev/flutter/material/ReorderableListView/onReorder.html) parameter is required and will be called when a child widget is dragged to a new position.

By default, on [TargetPlatformVariant.desktop](https://api.flutter.dev/flutter/flutter_test/TargetPlatformVariant/TargetPlatformVariant.desktop.html) platforms each item will have a drag handle added on top of it that will allow the user to grab it to move the item. On [TargetPlatformVariant.mobile](https://api.flutter.dev/flutter/flutter_test/TargetPlatformVariant/TargetPlatformVariant.mobile.html), no drag handle will be added, but when the user long presses anywhere on the item it will start moving the item. Displaying drag handles can be controlled with [ReorderableListView.buildDefaultDragHandles](https://api.flutter.dev/flutter/material/ReorderableListView/buildDefaultDragHandles.html).

All list items must have a key.

This example demonstrates using the [ReorderableListView.proxyDecorator](https://api.flutter.dev/flutter/material/ReorderableListView/proxyDecorator.html) callback to customize the appearance of a list item while it's being dragged.

This example demonstrates using the [ReorderableListView.proxyDecorator](https://api.flutter.dev/flutter/material/ReorderableListView/proxyDecorator.html) callback to customize the appearance of a [Card](https://api.flutter.dev/flutter/material/Card-class.html) while it's being dragged.

### Constructors

[ReorderableListView](https://api.flutter.dev/flutter/material/ReorderableListView/ReorderableListView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children, required [ReorderCallback](https://api.flutter.dev/flutter/widgets/ReorderCallback.html) onReorder, void onReorderStart([int](https://api.flutter.dev/flutter/dart-core/int-class.html) index)?, void onReorderEnd([int](https://api.flutter.dev/flutter/dart-core/int-class.html) index)?, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [ItemExtentBuilder](https://api.flutter.dev/flutter/rendering/ItemExtentBuilder.html)? itemExtentBuilder, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, [ReorderItemProxyDecorator](https://api.flutter.dev/flutter/widgets/ReorderItemProxyDecorator.html)? proxyDecorator, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) buildDefaultDragHandles = true, [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html)? padding, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? header, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? footer, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? scrollController, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) anchor = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? autoScrollerVelocityScalar})

Creates a reorderable list from a pre-built list of widgets.

[ReorderableListView.builder](https://api.flutter.dev/flutter/material/ReorderableListView/ReorderableListView.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [IndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/IndexedWidgetBuilder.html) itemBuilder, required [int](https://api.flutter.dev/flutter/dart-core/int-class.html) itemCount, required [ReorderCallback](https://api.flutter.dev/flutter/widgets/ReorderCallback.html) onReorder, void onReorderStart([int](https://api.flutter.dev/flutter/dart-core/int-class.html) index)?, void onReorderEnd([int](https://api.flutter.dev/flutter/dart-core/int-class.html) index)?, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? itemExtent, [ItemExtentBuilder](https://api.flutter.dev/flutter/rendering/ItemExtentBuilder.html)? itemExtentBuilder, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? prototypeItem, [ReorderItemProxyDecorator](https://api.flutter.dev/flutter/widgets/ReorderItemProxyDecorator.html)? proxyDecorator, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) buildDefaultDragHandles = true, [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html)? padding, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? header, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? footer, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? scrollController, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) shrinkWrap = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) anchor = 0.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? cacheExtent, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? autoScrollerVelocityScalar})

Creates a reorderable list from widget items that are created on demand.

const

### Properties

[anchor](https://api.flutter.dev/flutter/material/ReorderableListView/anchor.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)

The relative position of the zero scroll offset.

final

[autoScrollerVelocityScalar](https://api.flutter.dev/flutter/material/ReorderableListView/autoScrollerVelocityScalar.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

The velocity scalar per pixel over scroll.

final

[buildDefaultDragHandles](https://api.flutter.dev/flutter/material/ReorderableListView/buildDefaultDragHandles.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

If true: on desktop platforms, a drag handle is stacked over the center of each item's trailing edge; on mobile platforms, a long press anywhere on the item starts a drag.

final

[cacheExtent](https://api.flutter.dev/flutter/material/ReorderableListView/cacheExtent.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

The viewport has an area before and after the visible area to cache items that are about to become visible when the user scrolls.

final

[clipBehavior](https://api.flutter.dev/flutter/material/ReorderableListView/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

final

[dragStartBehavior](https://api.flutter.dev/flutter/material/ReorderableListView/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

final

[footer](https://api.flutter.dev/flutter/material/ReorderableListView/footer.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)?

A non-reorderable footer item to show after the items of the list.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[header](https://api.flutter.dev/flutter/material/ReorderableListView/header.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)?

A non-reorderable header item to show before the items of the list.

final

[itemBuilder](https://api.flutter.dev/flutter/material/ReorderableListView/itemBuilder.html) → [IndexedWidgetBuilder](https://api.flutter.dev/flutter/widgets/IndexedWidgetBuilder.html)

Called, as needed, to build list item widgets.

final

[itemCount](https://api.flutter.dev/flutter/material/ReorderableListView/itemCount.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The number of items in the list.

final

[itemExtent](https://api.flutter.dev/flutter/material/ReorderableListView/itemExtent.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

If non-null, forces the children to have the given extent in the scroll direction.

final

[itemExtentBuilder](https://api.flutter.dev/flutter/material/ReorderableListView/itemExtentBuilder.html) → [ItemExtentBuilder](https://api.flutter.dev/flutter/rendering/ItemExtentBuilder.html)?

If non-null, forces the children to have the corresponding extent returned by the builder.

final

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[keyboardDismissBehavior](https://api.flutter.dev/flutter/material/ReorderableListView/keyboardDismissBehavior.html) → [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html)

[ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) the defines how this [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) will dismiss the keyboard automatically.

final

[onReorder](https://api.flutter.dev/flutter/material/ReorderableListView/onReorder.html) → [ReorderCallback](https://api.flutter.dev/flutter/widgets/ReorderCallback.html)

A callback used by the list to report that a list item has been dragged to a new location in the list and the application should update the order of the items.

final

[onReorderEnd](https://api.flutter.dev/flutter/material/ReorderableListView/onReorderEnd.html) → (void Function([int](https://api.flutter.dev/flutter/dart-core/int-class.html) index)?)

A callback that is called when the dragged item is dropped.

final

[onReorderStart](https://api.flutter.dev/flutter/material/ReorderableListView/onReorderStart.html) → (void Function([int](https://api.flutter.dev/flutter/dart-core/int-class.html) index)?)

A callback that is called when an item drag has started.

final

[padding](https://api.flutter.dev/flutter/material/ReorderableListView/padding.html) → [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html)?

The amount of space by which to inset the list contents.

final

[physics](https://api.flutter.dev/flutter/material/ReorderableListView/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the scroll view should respond to user input.

final

[primary](https://api.flutter.dev/flutter/material/ReorderableListView/primary.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Whether this is the primary scroll view associated with the parent [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html).

final

[prototypeItem](https://api.flutter.dev/flutter/material/ReorderableListView/prototypeItem.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)?

If non-null, forces the children to have the same extent as the given widget in the scroll direction.

final

[proxyDecorator](https://api.flutter.dev/flutter/material/ReorderableListView/proxyDecorator.html) → [ReorderItemProxyDecorator](https://api.flutter.dev/flutter/widgets/ReorderItemProxyDecorator.html)?

A callback that allows the app to add an animated decoration around an item when it is being dragged.

final

[restorationId](https://api.flutter.dev/flutter/material/ReorderableListView/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

final

[reverse](https://api.flutter.dev/flutter/material/ReorderableListView/reverse.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the scroll view scrolls in the reading direction.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollController](https://api.flutter.dev/flutter/material/ReorderableListView/scrollController.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

An object that can be used to control the position to which this scroll view is scrolled.

final

[scrollDirection](https://api.flutter.dev/flutter/material/ReorderableListView/scrollDirection.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) along which the scroll view's offset increases.

final

[shrinkWrap](https://api.flutter.dev/flutter/material/ReorderableListView/shrinkWrap.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the extent of the scroll view in the [scrollDirection](https://api.flutter.dev/flutter/material/ReorderableListView/scrollDirection.html) should be determined by the contents being viewed.

final

## scroll configuration

Controls how [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) widgets behave in a subtree.

The scroll configuration determines the [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html) and viewport decorations used by descendants of [child](https://api.flutter.dev/flutter/widgets/ProxyWidget/child.html).

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [ProxyWidget](https://api.flutter.dev/flutter/widgets/ProxyWidget-class.html)
* [InheritedWidget](https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html)
* ScrollConfiguration

### Constructors

[ScrollConfiguration](https://api.flutter.dev/flutter/widgets/ScrollConfiguration/ScrollConfiguration.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) behavior, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child})

Creates a widget that controls how [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) widgets behave in a subtree.

const

### Properties

[behavior](https://api.flutter.dev/flutter/widgets/ScrollConfiguration/behavior.html) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)

How [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) widgets that are descendants of [child](https://api.flutter.dev/flutter/widgets/ProxyWidget/child.html) should behave.

final

[child](https://api.flutter.dev/flutter/widgets/ProxyWidget/child.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

The widget below this widget in the tree.

finalinherited

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

### Methods

[createElement](https://api.flutter.dev/flutter/widgets/InheritedWidget/createElement.html)() → [InheritedElement](https://api.flutter.dev/flutter/widgets/InheritedElement-class.html)

Inflates this configuration to a concrete instance.

inherited

[debugDescribeChildren](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)() → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)>

Returns a list of DiagnosticsNode objects describing this node's children.

inherited

[debugFillProperties](https://api.flutter.dev/flutter/widgets/ScrollConfiguration/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://api.flutter.dev/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

Add additional properties associated with the node.

override

[noSuchMethod](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a nonexistent method or property is accessed.

inherited

[toDiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html)? name, [DiagnosticsTreeStyle](https://api.flutter.dev/flutter/foundation/DiagnosticsTreeStyle.html)? style}) → [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [DiagnosticsNode.toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticsNode/toStringDeep.html).

inherited

[toString](https://api.flutter.dev/flutter/foundation/Diagnosticable/toString.html)({[DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.info}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A string representation of this object.

inherited

[toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixLineOne = '', [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? prefixOtherLines, [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants.

inherited

[toStringShallow](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) joiner = ', ', [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object.

inherited

[toStringShort](https://api.flutter.dev/flutter/widgets/Widget/toStringShort.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A short, textual description of this widget.

inherited

[updateShouldNotify](https://api.flutter.dev/flutter/widgets/ScrollConfiguration/updateShouldNotify.html)(covariant [ScrollConfiguration](https://api.flutter.dev/flutter/widgets/ScrollConfiguration-class.html) oldWidget) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the framework should notify widgets that inherit from this widget.

override

### Operators

[operator ==](https://api.flutter.dev/flutter/widgets/Widget/operator_equals.html)([Object](https://api.flutter.dev/flutter/dart-core/Object-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

The equality operator.

inherited

### Static Methods

[of](https://api.flutter.dev/flutter/widgets/ScrollConfiguration/of.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)

The [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) for [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) widgets in the given [BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html).

## scrollable

A widget that manages scrolling in one dimension and informs the [Viewport](https://api.flutter.dev/flutter/widgets/Viewport-class.html) through which the content is viewed.

[Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) implements the interaction model for a scrollable widget, including gesture recognition, but does not have an opinion about how the viewport, which actually displays the children, is constructed.

It's rare to construct a [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) directly. Instead, consider [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) or [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html), which combine scrolling, viewporting, and a layout model. To combine layout models (or to use a custom layout mode), consider using [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html).

The static [Scrollable.of](https://api.flutter.dev/flutter/widgets/Scrollable/of.html) and [Scrollable.ensureVisible](https://api.flutter.dev/flutter/widgets/Scrollable/ensureVisible.html) functions are often used to interact with the [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) widget inside a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) or a [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html).

To further customize scrolling behavior with a [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html):

1. You can provide a [viewportBuilder](https://api.flutter.dev/flutter/widgets/Scrollable/viewportBuilder.html) to customize the child model. For example, [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html) uses a viewport that displays a single box child whereas [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) uses a [Viewport](https://api.flutter.dev/flutter/widgets/Viewport-class.html) or a [ShrinkWrappingViewport](https://api.flutter.dev/flutter/widgets/ShrinkWrappingViewport-class.html), both of which display a list of slivers.
2. You can provide a custom [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html) that creates a custom [ScrollPosition](https://api.flutter.dev/flutter/widgets/ScrollPosition-class.html) subclass. For example, [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html) uses a [PageController](https://api.flutter.dev/flutter/widgets/PageController-class.html), which creates a page-oriented scroll position subclass that keeps the same page visible when the [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) resizes.

### Persisting the scroll position during a session

Scrollables attempt to persist their scroll position using [PageStorage](https://api.flutter.dev/flutter/widgets/PageStorage-class.html). This can be disabled by setting [ScrollController.keepScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/keepScrollOffset.html) to false on the [controller](https://api.flutter.dev/flutter/widgets/Scrollable/controller.html). If it is enabled, using a [PageStorageKey](https://api.flutter.dev/flutter/widgets/PageStorageKey-class.html) for the [key](https://api.flutter.dev/flutter/widgets/Widget/key.html) of this widget (or one of its ancestors, e.g. a [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html)) is recommended to help disambiguate different [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html)s from each other.

See also:

* [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html), which is a commonly used [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) that displays a scrolling, linear list of child widgets.
* [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html), which is a scrolling list of child widgets that are each the size of the viewport.
* [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html), which is a [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) that displays a scrolling, 2D array of child widgets.
* [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html), which is a [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) that creates custom scroll effects using slivers.
* [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html), which is a scrollable widget that has a single child.
* [ScrollNotification](https://api.flutter.dev/flutter/widgets/ScrollNotification-class.html) and [NotificationListener](https://api.flutter.dev/flutter/widgets/NotificationListener-class.html), which can be used to watch the scroll position without using a [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html).

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [StatefulWidget](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html)
* Scrollable

**Constructors**

[Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable/Scrollable.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [AxisDirection](https://api.flutter.dev/flutter/painting/AxisDirection.html) axisDirection = AxisDirection.down, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, required [ViewportBuilder](https://api.flutter.dev/flutter/widgets/ViewportBuilder.html) viewportBuilder, [ScrollIncrementCalculator](https://api.flutter.dev/flutter/widgets/ScrollIncrementCalculator.html)? incrementCalculator, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? semanticChildCount, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)? scrollBehavior, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge})

Creates a widget that scrolls.

const

### Properties

[axis](https://api.flutter.dev/flutter/widgets/Scrollable/axis.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The axis along which the scroll view scrolls.

read-only

[axisDirection](https://api.flutter.dev/flutter/widgets/Scrollable/axisDirection.html) → [AxisDirection](https://api.flutter.dev/flutter/painting/AxisDirection.html)

The direction in which this widget scrolls.

final

[clipBehavior](https://api.flutter.dev/flutter/widgets/Scrollable/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

final

[controller](https://api.flutter.dev/flutter/widgets/Scrollable/controller.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

An object that can be used to control the position to which this widget is scrolled.

final

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/Scrollable/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

final

[excludeFromSemantics](https://api.flutter.dev/flutter/widgets/Scrollable/excludeFromSemantics.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the scroll actions introduced by this [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) are exposed in the semantics tree.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[incrementCalculator](https://api.flutter.dev/flutter/widgets/Scrollable/incrementCalculator.html) → [ScrollIncrementCalculator](https://api.flutter.dev/flutter/widgets/ScrollIncrementCalculator.html)?

An optional function that will be called to calculate the distance to scroll when the scrollable is asked to scroll via the keyboard using a [ScrollAction](https://api.flutter.dev/flutter/widgets/ScrollAction-class.html).

final

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[physics](https://api.flutter.dev/flutter/widgets/Scrollable/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the widgets should respond to user input.

final

[restorationId](https://api.flutter.dev/flutter/widgets/Scrollable/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollBehavior](https://api.flutter.dev/flutter/widgets/Scrollable/scrollBehavior.html) → [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html)?

A [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) that will be applied to this widget individually.

final

[semanticChildCount](https://api.flutter.dev/flutter/widgets/Scrollable/semanticChildCount.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)?

The number of children that will contribute semantic information.

final

[viewportBuilder](https://api.flutter.dev/flutter/widgets/Scrollable/viewportBuilder.html) → [ViewportBuilder](https://api.flutter.dev/flutter/widgets/ViewportBuilder.html)

Builds the viewport through which the scrollable content is displayed.

final

### Methods

[createElement](https://api.flutter.dev/flutter/widgets/StatefulWidget/createElement.html)() → [StatefulElement](https://api.flutter.dev/flutter/widgets/StatefulElement-class.html)

Creates a [StatefulElement](https://api.flutter.dev/flutter/widgets/StatefulElement-class.html) to manage this widget's location in the tree.

inherited

[createState](https://api.flutter.dev/flutter/widgets/Scrollable/createState.html)() → [ScrollableState](https://api.flutter.dev/flutter/widgets/ScrollableState-class.html)

Creates the mutable state for this widget at a given location in the tree.

override

[debugDescribeChildren](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)() → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)>

Returns a list of DiagnosticsNode objects describing this node's children.

inherited

[debugFillProperties](https://api.flutter.dev/flutter/widgets/Scrollable/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://api.flutter.dev/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

Add additional properties associated with the node.

override

[noSuchMethod](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a nonexistent method or property is accessed.

inherited

[toDiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html)? name, [DiagnosticsTreeStyle](https://api.flutter.dev/flutter/foundation/DiagnosticsTreeStyle.html)? style}) → [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [DiagnosticsNode.toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticsNode/toStringDeep.html).

inherited

[toString](https://api.flutter.dev/flutter/foundation/Diagnosticable/toString.html)({[DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.info}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A string representation of this object.

inherited

[toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixLineOne = '', [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? prefixOtherLines, [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants.

inherited

[toStringShallow](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) joiner = ', ', [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object.

inherited

[toStringShort](https://api.flutter.dev/flutter/widgets/Widget/toStringShort.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A short, textual description of this widget.

inherited

### Operators

[operator ==](https://api.flutter.dev/flutter/widgets/Widget/operator_equals.html)([Object](https://api.flutter.dev/flutter/dart-core/Object-class.html) other) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

The equality operator.

inherited

### Static Methods

[ensureVisible](https://api.flutter.dev/flutter/widgets/Scrollable/ensureVisible.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[double](https://api.flutter.dev/flutter/dart-core/double-class.html) alignment = 0.0, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) duration = Duration.zero, [Curve](https://api.flutter.dev/flutter/animation/Curve-class.html) curve = Curves.ease, [ScrollPositionAlignmentPolicy](https://api.flutter.dev/flutter/widgets/ScrollPositionAlignmentPolicy.html) alignmentPolicy = ScrollPositionAlignmentPolicy.explicit}) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<void>

Scrolls the scrollables that enclose the given context so as to make the given context visible.

[maybeOf](https://api.flutter.dev/flutter/widgets/Scrollable/maybeOf.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[Axis](https://api.flutter.dev/flutter/painting/Axis.html)? axis}) → [ScrollableState](https://api.flutter.dev/flutter/widgets/ScrollableState-class.html)?

The state from the closest instance of this class that encloses the given context, or null if none is found.

[of](https://api.flutter.dev/flutter/widgets/Scrollable/of.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[Axis](https://api.flutter.dev/flutter/painting/Axis.html)? axis}) → [ScrollableState](https://api.flutter.dev/flutter/widgets/ScrollableState-class.html)

The state from the closest instance of this class that encloses the given context.

[recommendDeferredLoadingForContext](https://api.flutter.dev/flutter/widgets/Scrollable/recommendDeferredLoadingForContext.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[Axis](https://api.flutter.dev/flutter/painting/Axis.html)? axis}) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Provides a heuristic to determine if expensive frame-bound tasks should be deferred for the context at a specific point in time.

## scrollbar

A Material Design scrollbar.

To add a scrollbar to a [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html), wrap the scroll view widget in a [Scrollbar](https://api.flutter.dev/flutter/material/Scrollbar-class.html) widget.

A scrollbar thumb indicates which portion of a [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) is actually visible.

By default, the thumb will fade in and out as the child scroll view scrolls. When [thumbVisibility](https://api.flutter.dev/flutter/material/Scrollbar/thumbVisibility.html) is true, the scrollbar thumb will remain visible without the fade animation. This requires that the [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html) associated with the Scrollable widget is provided to [controller](https://api.flutter.dev/flutter/material/Scrollbar/controller.html), or that the [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html) is being used by that Scrollable widget.

If the scrollbar is wrapped around multiple [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html)s, it only responds to the nearest ScrollView and shows the corresponding scrollbar thumb by default. The [notificationPredicate](https://api.flutter.dev/flutter/material/Scrollbar/notificationPredicate.html) allows the ability to customize which [ScrollNotification](https://api.flutter.dev/flutter/widgets/ScrollNotification-class.html)s the Scrollbar should listen to.

If the child [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) is infinitely long, the [RawScrollbar](https://api.flutter.dev/flutter/widgets/RawScrollbar-class.html) will not be painted. In this case, the scrollbar cannot accurately represent the relative location of the visible area, or calculate the accurate delta to apply when dragging on the thumb or tapping on the track.

### Interaction

Scrollbars are interactive and can use the [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html) if a [controller](https://api.flutter.dev/flutter/material/Scrollbar/controller.html) is not set. Interactive Scrollbar thumbs can be dragged along the main axis of the [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) to change the [ScrollPosition](https://api.flutter.dev/flutter/widgets/ScrollPosition-class.html). Tapping along the track exclusive of the thumb will trigger a [ScrollIncrementType.page](https://api.flutter.dev/flutter/widgets/ScrollIncrementType.html) based on the relative position to the thumb.

When using the [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html), it must not be attached to more than one [ScrollPosition](https://api.flutter.dev/flutter/widgets/ScrollPosition-class.html). [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html)s that have not been provided a [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html) and have a [ScrollView.scrollDirection](https://api.flutter.dev/flutter/widgets/ScrollView/scrollDirection.html) of [Axis.vertical](https://api.flutter.dev/flutter/painting/Axis.html) will automatically attach their ScrollPosition to the PrimaryScrollController. Provide a unique ScrollController to each [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) in this case to prevent having multiple ScrollPositions attached to the PrimaryScrollController.

### Automatic Scrollbars on Desktop Platforms

Scrollbars are added to most [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html) widgets by default on [TargetPlatformVariant.desktop](https://api.flutter.dev/flutter/flutter_test/TargetPlatformVariant/TargetPlatformVariant.desktop.html) platforms. This is done through [ScrollBehavior.buildScrollbar](https://api.flutter.dev/flutter/widgets/ScrollBehavior/buildScrollbar.html) as part of an app's [ScrollConfiguration](https://api.flutter.dev/flutter/widgets/ScrollConfiguration-class.html). Scrollables that do not use the [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html) or have a [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html) provided to them will receive a unique ScrollController for use with the Scrollbar. In this case, only one Scrollable can be using the PrimaryScrollController, unless [interactive](https://api.flutter.dev/flutter/material/Scrollbar/interactive.html) is false. To prevent [Axis.vertical](https://api.flutter.dev/flutter/painting/Axis.html) Scrollables from using the PrimaryScrollController, set [ScrollView.primary](https://api.flutter.dev/flutter/widgets/ScrollView/primary.html) to false. Scrollable widgets that do not have automatically applied Scrollbars include

* [EditableText](https://api.flutter.dev/flutter/widgets/EditableText-class.html)
* [ListWheelScrollView](https://api.flutter.dev/flutter/widgets/ListWheelScrollView-class.html)
* [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html)
* [NestedScrollView](https://api.flutter.dev/flutter/widgets/NestedScrollView-class.html)
* [DropdownButton](https://api.flutter.dev/flutter/material/DropdownButton-class.html)

Default Scrollbars can be disabled for the whole app by setting a [ScrollBehavior](https://api.flutter.dev/flutter/widgets/ScrollBehavior-class.html) with scrollbars set to false.

Dynamically changes to a [CupertinoScrollbar](https://api.flutter.dev/flutter/cupertino/CupertinoScrollbar-class.html), an iOS style scrollbar, by default on the iOS platform.

The color of the Scrollbar thumb will change when [MaterialState.dragged](https://api.flutter.dev/flutter/material/MaterialState.html), or [MaterialState.hovered](https://api.flutter.dev/flutter/material/MaterialState.html) on desktop and web platforms. These stateful color choices can be changed using [ScrollbarThemeData.thumbColor](https://api.flutter.dev/flutter/material/ScrollbarThemeData/thumbColor.html).

A scrollbar track can be added using [trackVisibility](https://api.flutter.dev/flutter/material/Scrollbar/trackVisibility.html). This can also be drawn when triggered by a hover event, or based on any [MaterialState](https://api.flutter.dev/flutter/material/MaterialState.html) by using [ScrollbarThemeData.trackVisibility](https://api.flutter.dev/flutter/material/ScrollbarThemeData/trackVisibility.html).

The [thickness](https://api.flutter.dev/flutter/material/Scrollbar/thickness.html) of the track and scrollbar thumb can be changed dynamically in response to [MaterialState](https://api.flutter.dev/flutter/material/MaterialState.html)s using [ScrollbarThemeData.thickness](https://api.flutter.dev/flutter/material/ScrollbarThemeData/thickness.html).

See also:

* [RawScrollbar](https://api.flutter.dev/flutter/widgets/RawScrollbar-class.html), a basic scrollbar that fades in and out, extended by this class to add more animations and behaviors.
* [ScrollbarTheme](https://api.flutter.dev/flutter/material/ScrollbarTheme-class.html), which configures the Scrollbar's appearance.
* [CupertinoScrollbar](https://api.flutter.dev/flutter/cupertino/CupertinoScrollbar-class.html), an iOS style scrollbar.
* [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html), which displays a linear, scrollable list of children.
* [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html), which displays a 2 dimensional, scrollable array of children.

### Constructors

[Scrollbar](https://api.flutter.dev/flutter/material/Scrollbar/Scrollbar.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? thumbVisibility, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? trackVisibility, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? thickness, [Radius](https://api.flutter.dev/flutter/dart-ui/Radius-class.html)? radius, [ScrollNotificationPredicate](https://api.flutter.dev/flutter/widgets/ScrollNotificationPredicate.html)? notificationPredicate, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? interactive, [ScrollbarOrientation](https://api.flutter.dev/flutter/widgets/ScrollbarOrientation.html)? scrollbarOrientation, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Use ScrollbarThemeData.trackVisibility to resolve based on the current state instead. ' 'This feature was deprecated after v3.4.0-19.0.pre.') [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? showTrackOnHover})

Creates a Material Design scrollbar that by default will connect to the closest Scrollable descendant of [child](https://api.flutter.dev/flutter/material/Scrollbar/child.html).

const

### Properties

[child](https://api.flutter.dev/flutter/material/Scrollbar/child.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

The widget below this widget in the tree.

final

[controller](https://api.flutter.dev/flutter/material/Scrollbar/controller.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

The [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html) used to implement Scrollbar dragging.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[interactive](https://api.flutter.dev/flutter/material/Scrollbar/interactive.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Whether the Scrollbar should be interactive and respond to dragging on the thumb, or tapping in the track area.

final

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[notificationPredicate](https://api.flutter.dev/flutter/material/Scrollbar/notificationPredicate.html) → [ScrollNotificationPredicate](https://api.flutter.dev/flutter/widgets/ScrollNotificationPredicate.html)?

A check that specifies whether a [ScrollNotification](https://api.flutter.dev/flutter/widgets/ScrollNotification-class.html) should be handled by this widget.

final

[radius](https://api.flutter.dev/flutter/material/Scrollbar/radius.html) → [Radius](https://api.flutter.dev/flutter/dart-ui/Radius-class.html)?

The [Radius](https://api.flutter.dev/flutter/dart-ui/Radius-class.html) of the scrollbar thumb's rounded rectangle corners.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollbarOrientation](https://api.flutter.dev/flutter/material/Scrollbar/scrollbarOrientation.html) → [ScrollbarOrientation](https://api.flutter.dev/flutter/widgets/ScrollbarOrientation.html)?

Dictates the orientation of the scrollbar.

final

[showTrackOnHover](https://api.flutter.dev/flutter/material/Scrollbar/showTrackOnHover.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Controls if the track will show on hover and remain, including during drag.

final

[thickness](https://api.flutter.dev/flutter/material/Scrollbar/thickness.html) → [double](https://api.flutter.dev/flutter/dart-core/double-class.html)?

The thickness of the scrollbar in the cross axis of the scrollable.

final

[thumbVisibility](https://api.flutter.dev/flutter/material/Scrollbar/thumbVisibility.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Indicates that the scrollbar thumb should be visible, even when a scroll is not underway.

final

[trackVisibility](https://api.flutter.dev/flutter/material/Scrollbar/trackVisibility.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Indicates that the scrollbar track should be visible.

final

### Methods

[build](https://api.flutter.dev/flutter/material/Scrollbar/build.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

Describes the part of the user interface represented by this widget.

override

[createElement](https://api.flutter.dev/flutter/widgets/StatelessWidget/createElement.html)() → [StatelessElement](https://api.flutter.dev/flutter/widgets/StatelessElement-class.html)

Creates a [StatelessElement](https://api.flutter.dev/flutter/widgets/StatelessElement-class.html) to manage this widget's location in the tree.

inherited

[debugDescribeChildren](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)() → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)>

Returns a list of DiagnosticsNode objects describing this node's children.

inherited

[debugFillProperties](https://api.flutter.dev/flutter/widgets/Widget/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://api.flutter.dev/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

Add additional properties associated with the node.

inherited

[noSuchMethod](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a nonexistent method or property is accessed.

inherited

[toDiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html)? name, [DiagnosticsTreeStyle](https://api.flutter.dev/flutter/foundation/DiagnosticsTreeStyle.html)? style}) → [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [DiagnosticsNode.toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticsNode/toStringDeep.html).

inherited

[toString](https://api.flutter.dev/flutter/foundation/Diagnosticable/toString.html)({[DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.info}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A string representation of this object.

inherited

[toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixLineOne = '', [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? prefixOtherLines, [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants.

inherited

[toStringShallow](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) joiner = ', ', [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object.

inherited

[toStringShort](https://api.flutter.dev/flutter/widgets/Widget/toStringShort.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A short, textual description of this widget.

inherited

## single child scroll view

A box in which a single widget can be scrolled.

This widget is useful when you have a single box that will normally be entirely visible, for example a clock face in a time picker, but you need to make sure it can be scrolled if the container gets too small in one axis (the scroll direction).

It is also useful if you need to shrink-wrap in both axes (the main scrolling direction as well as the cross axis), as one might see in a dialog or pop-up menu. In that case, you might pair the [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html) with a [ListBody](https://api.flutter.dev/flutter/widgets/ListBody-class.html) child.

When you have a list of children and do not require cross-axis shrink-wrapping behavior, for example a scrolling list that is always the width of the screen, consider [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html), which is vastly more efficient than a [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html) containing a [ListBody](https://api.flutter.dev/flutter/widgets/ListBody-class.html) or [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) with many children.

### Sample code: Using [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html) with a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html)

Sometimes a layout is designed around the flexible properties of a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), but there is the concern that in some cases, there might not be enough room to see the entire contents. This could be because some devices have unusually small screens, or because the application can be used in landscape mode where the aspect ratio isn't what was originally envisioned, or because the application is being shown in a small window in split-screen mode. In any case, as a result, it might make sense to wrap the layout in a [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html).

Doing so, however, usually results in a conflict between the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), which typically tries to grow as big as it can, and the [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html), which provides its children with an infinite amount of space.

To resolve this apparent conflict, there are a couple of techniques, as discussed below. These techniques should only be used when the content is normally expected to fit on the screen, so that the lazy instantiation of a sliver-based [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) or [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html) is not expected to provide any performance benefit. If the viewport is expected to usually contain content beyond the dimensions of the screen, then [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html) would be very expensive (in which case [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) may be a better choice than [Column](https://api.flutter.dev/flutter/widgets/Column-class.html)).

### Centering, spacing, or aligning fixed-height content

If the content has fixed (or intrinsic) dimensions but needs to be spaced out, centered, or otherwise positioned using the [Flex](https://api.flutter.dev/flutter/widgets/Flex-class.html) layout model of a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), the following technique can be used to provide the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) with a minimum dimension while allowing it to shrink-wrap the contents when there isn't enough room to apply these spacing or alignment needs.

A [LayoutBuilder](https://api.flutter.dev/flutter/widgets/LayoutBuilder-class.html) is used to obtain the size of the viewport (implicitly via the constraints that the [SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html) sees, since viewports typically grow to fit their maximum height constraint). Then, inside the scroll view, a [ConstrainedBox](https://api.flutter.dev/flutter/widgets/ConstrainedBox-class.html) is used to set the minimum height of the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html).

The [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) has no [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) children, so rather than take on the infinite height from its [BoxConstraints.maxHeight](https://api.flutter.dev/flutter/rendering/BoxConstraints/maxHeight.html), (the viewport provides no maximum height constraint), it automatically tries to shrink to fit its children. It cannot be smaller than its [BoxConstraints.minHeight](https://api.flutter.dev/flutter/rendering/BoxConstraints/minHeight.html), though, and It therefore becomes the bigger of the minimum height provided by the [ConstrainedBox](https://api.flutter.dev/flutter/widgets/ConstrainedBox-class.html) and the sum of the heights of the children.

If the children aren't enough to fit that minimum size, the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) ends up with some remaining space to allocate as specified by its [Column.mainAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/mainAxisAlignment.html) argument.

### Expanding content to fit the viewport

The following example builds on the previous one. In addition to providing a minimum dimension for the child [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), an [IntrinsicHeight](https://api.flutter.dev/flutter/widgets/IntrinsicHeight-class.html) widget is used to force the column to be exactly as big as its contents. This constraint combines with the [ConstrainedBox](https://api.flutter.dev/flutter/widgets/ConstrainedBox-class.html) constraints discussed previously to ensure that the column becomes either as big as viewport, or as big as the contents, whichever is biggest.

Both constraints must be used to get the desired effect. If only the [IntrinsicHeight](https://api.flutter.dev/flutter/widgets/IntrinsicHeight-class.html) was specified, then the column would not grow to fit the entire viewport when its children were smaller than the whole screen. If only the size of the viewport was used, then the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) would overflow if the children were bigger than the viewport.

The widget that is to grow to fit the remaining space so provided is wrapped in an [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) widget.

This technique is quite expensive, as it more or less requires that the contents of the viewport be laid out twice (once to find their intrinsic dimensions, and once to actually lay them out). The number of widgets within the column should therefore be kept small. Alternatively, subsets of the children that have known dimensions can be wrapped in a [SizedBox](https://api.flutter.dev/flutter/widgets/SizedBox-class.html) that has tight vertical constraints, so that the intrinsic sizing algorithm can short-circuit the computation when it reaches those parts of the subtree.

### Persisting the scroll position during a session

Scroll views attempt to persist their scroll position using [PageStorage](https://api.flutter.dev/flutter/widgets/PageStorage-class.html). This can be disabled by setting [ScrollController.keepScrollOffset](https://api.flutter.dev/flutter/widgets/ScrollController/keepScrollOffset.html) to false on the [controller](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/controller.html). If it is enabled, using a [PageStorageKey](https://api.flutter.dev/flutter/widgets/PageStorageKey-class.html) for the [key](https://api.flutter.dev/flutter/widgets/Widget/key.html) of this widget is recommended to help disambiguate different scroll views from each other.

See also:

* [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html), which handles multiple children in a scrolling list.
* [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html), which handles multiple children in a scrolling grid.
* [PageView](https://api.flutter.dev/flutter/widgets/PageView-class.html), for a scrollable that works page by page.
* [Scrollable](https://api.flutter.dev/flutter/widgets/Scrollable-class.html), which handles arbitrary scrolling effects.

Inheritance

* [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)
* [DiagnosticableTree](https://api.flutter.dev/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)
* [StatelessWidget](https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html)
* SingleChildScrollView

### Constructors

[SingleChildScrollView](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/SingleChildScrollView.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Axis](https://api.flutter.dev/flutter/painting/Axis.html) scrollDirection = Axis.vertical, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reverse = false, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? primary, [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)? physics, [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)? controller, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId, [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) keyboardDismissBehavior = ScrollViewKeyboardDismissBehavior.manual})

Creates a box in which a single widget can be scrolled.

const

### Properties

[child](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/child.html) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)?

The widget that scrolls.

final

[clipBehavior](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/clipBehavior.html) → [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)

The content will be clipped (or not) according to this option.

final

[controller](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/controller.html) → [ScrollController](https://api.flutter.dev/flutter/widgets/ScrollController-class.html)?

An object that can be used to control the position to which this scroll view is scrolled.

final

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/dragStartBehavior.html) → [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html)

Determines the way that drag start behavior is handled.

final

[hashCode](https://api.flutter.dev/flutter/widgets/Widget/hashCode.html) → [int](https://api.flutter.dev/flutter/dart-core/int-class.html)

The hash code for this object.

read-onlyinherited

[key](https://api.flutter.dev/flutter/widgets/Widget/key.html) → [Key](https://api.flutter.dev/flutter/foundation/Key-class.html)?

Controls how one widget replaces another widget in the tree.

finalinherited

[keyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/keyboardDismissBehavior.html) → [ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html)

[ScrollViewKeyboardDismissBehavior](https://api.flutter.dev/flutter/widgets/ScrollViewKeyboardDismissBehavior.html) the defines how this [ScrollView](https://api.flutter.dev/flutter/widgets/ScrollView-class.html) will dismiss the keyboard automatically.

final

[padding](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/padding.html) → [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)?

The amount of space by which to inset the child.

final

[physics](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/physics.html) → [ScrollPhysics](https://api.flutter.dev/flutter/widgets/ScrollPhysics-class.html)?

How the scroll view should respond to user input.

final

[primary](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/primary.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)?

Whether this is the primary scroll view associated with the parent [PrimaryScrollController](https://api.flutter.dev/flutter/widgets/PrimaryScrollController-class.html).

final

[restorationId](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/restorationId.html) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)?

Restoration ID to save and restore the scroll offset of the scrollable.

final

[reverse](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/reverse.html) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the scroll view scrolls in the reading direction.

final

[runtimeType](https://api.flutter.dev/flutter/dart-core/Object/runtimeType.html) → [Type](https://api.flutter.dev/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

read-onlyinherited

[scrollDirection](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/scrollDirection.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) along which the scroll view's offset increases.

final

### Methods

[build](https://api.flutter.dev/flutter/widgets/SingleChildScrollView/build.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context) → [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)

Describes the part of the user interface represented by this widget.

override

[createElement](https://api.flutter.dev/flutter/widgets/StatelessWidget/createElement.html)() → [StatelessElement](https://api.flutter.dev/flutter/widgets/StatelessElement-class.html)

Creates a [StatelessElement](https://api.flutter.dev/flutter/widgets/StatelessElement-class.html) to manage this widget's location in the tree.

inherited

[debugDescribeChildren](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)() → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)>

Returns a list of DiagnosticsNode objects describing this node's children.

inherited

[debugFillProperties](https://api.flutter.dev/flutter/widgets/Widget/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://api.flutter.dev/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

Add additional properties associated with the node.

inherited

[noSuchMethod](https://api.flutter.dev/flutter/dart-core/Object/noSuchMethod.html)([Invocation](https://api.flutter.dev/flutter/dart-core/Invocation-class.html) invocation) → dynamic

Invoked when a nonexistent method or property is accessed.

inherited

[toDiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html)? name, [DiagnosticsTreeStyle](https://api.flutter.dev/flutter/foundation/DiagnosticsTreeStyle.html)? style}) → [DiagnosticsNode](https://api.flutter.dev/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [DiagnosticsNode.toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticsNode/toStringDeep.html).

inherited

[toString](https://api.flutter.dev/flutter/foundation/Diagnosticable/toString.html)({[DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.info}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A string representation of this object.

inherited

[toStringDeep](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringDeep.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) prefixLineOne = '', [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? prefixOtherLines, [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants.

inherited

[toStringShallow](https://api.flutter.dev/flutter/foundation/DiagnosticableTree/toStringShallow.html)({[String](https://api.flutter.dev/flutter/dart-core/String-class.html) joiner = ', ', [DiagnosticLevel](https://api.flutter.dev/flutter/foundation/DiagnosticLevel.html) minLevel = DiagnosticLevel.debug}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object.

inherited

[toStringShort](https://api.flutter.dev/flutter/widgets/Widget/toStringShort.html)() → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

A short, textual description of this widget.

inherited