Flutter resumen documentación

# INTERACTION MODEL WIDGETS

# TOUCH INTERACTION

## ABSORB POINTER

A widget that absorbs pointers during hit testing.

When [absorbing](https://api.flutter.dev/flutter/widgets/AbsorbPointer/absorbing.html) is true, this widget prevents its subtree from receiving pointer events by terminating hit testing at itself. It still consumes space during layout and paints its child as usual. It just prevents its children from being the target of located events, because it returns true from [RenderBox.hitTest](https://api.flutter.dev/flutter/rendering/RenderBox/hitTest.html).

When [ignoringSemantics](https://api.flutter.dev/flutter/widgets/AbsorbPointer/ignoringSemantics.html) is true, the subtree will be invisible to the semantics layer (and thus e.g. accessibility tools).

### Semantics

Using this class may also affect how the semantics subtree underneath is collected.

If [absorbing](https://api.flutter.dev/flutter/widgets/AbsorbPointer/absorbing.html) is true, pointer-related [SemanticsAction](https://api.flutter.dev/flutter/dart-ui/SemanticsAction-class.html)s are removed from the semantics subtree. Otherwise, the subtree remains untouched.

The usages of [ignoringSemantics](https://api.flutter.dev/flutter/widgets/AbsorbPointer/ignoringSemantics.html) are deprecated and not recommended. This property was introduced to workaround the semantics behavior of the [IgnorePointer](https://api.flutter.dev/flutter/widgets/IgnorePointer-class.html) and its friends before v3.8.0-12.0.pre.

Before that version, entire semantics subtree is dropped if [absorbing](https://api.flutter.dev/flutter/widgets/AbsorbPointer/absorbing.html) is true. Developers can only use [ignoringSemantics](https://api.flutter.dev/flutter/widgets/AbsorbPointer/ignoringSemantics.html) to preserver the semantics subtrees.

After that version, with [absorbing](https://api.flutter.dev/flutter/widgets/AbsorbPointer/absorbing.html) set to true, it only prevents semantics user actions in the semantics subtree but leaves the other [SemanticsProperties](https://api.flutter.dev/flutter/semantics/SemanticsProperties-class.html) intact. Therefore, the [ignoringSemantics](https://api.flutter.dev/flutter/widgets/AbsorbPointer/ignoringSemantics.html) is no longer needed.

If [ignoringSemantics](https://api.flutter.dev/flutter/widgets/AbsorbPointer/ignoringSemantics.html) is true, the semantics subtree is dropped. Therefore, the subtree will be invisible to assistive technologies.

If [ignoringSemantics](https://api.flutter.dev/flutter/widgets/AbsorbPointer/ignoringSemantics.html) is false, the semantics subtree is collected as usual.

### Constructors

[AbsorbPointer](https://api.flutter.dev/flutter/widgets/AbsorbPointer/AbsorbPointer.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) absorbing = true, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Use ExcludeSemantics or create a custom absorb pointer widget instead. ' 'This feature was deprecated after v3.8.0-12.0.pre.') [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? ignoringSemantics, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a widget that absorbs pointers during hit testing.

const

### Properties

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

Whether this widget absorbs pointers during hit testing.

final

[child](https://api.flutter.dev/flutter/widgets/SingleChildRenderObjectWidget/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

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

Whether the semantics of this render object is ignored when compiling the semantics tree.

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

[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

## DISMISSIBLE

A widget that can be dismissed by dragging in the indicated [direction](https://api.flutter.dev/flutter/widgets/Dismissible/direction.html).

Dragging or flinging this widget in the [DismissDirection](https://api.flutter.dev/flutter/widgets/DismissDirection.html) causes the child to slide out of view. Following the slide animation, if [resizeDuration](https://api.flutter.dev/flutter/widgets/Dismissible/resizeDuration.html) is non-null, the Dismissible widget animates its height (or width, whichever is perpendicular to the dismiss direction) to zero over the [resizeDuration](https://api.flutter.dev/flutter/widgets/Dismissible/resizeDuration.html).

Backgrounds can be used to implement the "leave-behind" idiom. If a background is specified it is stacked behind the Dismissible's child and is exposed when the child moves.

The widget calls the [onDismissed](https://api.flutter.dev/flutter/widgets/Dismissible/onDismissed.html) callback either after its size has collapsed to zero (if [resizeDuration](https://api.flutter.dev/flutter/widgets/Dismissible/resizeDuration.html) is non-null) or immediately after the slide animation (if [resizeDuration](https://api.flutter.dev/flutter/widgets/Dismissible/resizeDuration.html) is null). If the Dismissible is a list item, it must have a key that distinguishes it from the other items and its [onDismissed](https://api.flutter.dev/flutter/widgets/Dismissible/onDismissed.html) callback must remove the item from the list.

### Constructors

[Dismissible](https://api.flutter.dev/flutter/widgets/Dismissible/Dismissible.html)({required [Key](https://api.flutter.dev/flutter/foundation/Key-class.html) key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? background, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? secondaryBackground, [ConfirmDismissCallback](https://api.flutter.dev/flutter/widgets/ConfirmDismissCallback.html)? confirmDismiss, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onResize, [DismissUpdateCallback](https://api.flutter.dev/flutter/widgets/DismissUpdateCallback.html)? onUpdate, [DismissDirectionCallback](https://api.flutter.dev/flutter/widgets/DismissDirectionCallback.html)? onDismissed, [DismissDirection](https://api.flutter.dev/flutter/widgets/DismissDirection.html) direction = DismissDirection.horizontal, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)? resizeDuration = const Duration(milliseconds: 300), [Map](https://api.flutter.dev/flutter/dart-core/Map-class.html)<[DismissDirection](https://api.flutter.dev/flutter/widgets/DismissDirection.html), [double](https://api.flutter.dev/flutter/dart-core/double-class.html)> dismissThresholds = const <DismissDirection, double>{}, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) movementDuration = const Duration(milliseconds: 200), [double](https://api.flutter.dev/flutter/dart-core/double-class.html) crossAxisEndOffset = 0.0, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html) behavior = HitTestBehavior.opaque})

Creates a widget that can be dismissed.

const

### Properties

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

A widget that is stacked behind the child. If secondaryBackground is also specified then this widget only appears when the child has been dragged down or to the right.

final

[behavior](https://api.flutter.dev/flutter/widgets/Dismissible/behavior.html) → [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html)

How to behave during hit tests.

final

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

The widget below this widget in the tree.

final

[confirmDismiss](https://api.flutter.dev/flutter/widgets/Dismissible/confirmDismiss.html) → [ConfirmDismissCallback](https://api.flutter.dev/flutter/widgets/ConfirmDismissCallback.html)?

Gives the app an opportunity to confirm or veto a pending dismissal.

final

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

Defines the end offset across the main axis after the card is dismissed.

final

[direction](https://api.flutter.dev/flutter/widgets/Dismissible/direction.html) → [DismissDirection](https://api.flutter.dev/flutter/widgets/DismissDirection.html)

The direction in which the widget can be dismissed.

final

[dismissThresholds](https://api.flutter.dev/flutter/widgets/Dismissible/dismissThresholds.html) → [Map](https://api.flutter.dev/flutter/dart-core/Map-class.html)<[DismissDirection](https://api.flutter.dev/flutter/widgets/DismissDirection.html), [double](https://api.flutter.dev/flutter/dart-core/double-class.html)>

The offset threshold the item has to be dragged in order to be considered dismissed.

final

[dragStartBehavior](https://api.flutter.dev/flutter/widgets/Dismissible/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

[movementDuration](https://api.flutter.dev/flutter/widgets/Dismissible/movementDuration.html) → [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)

Defines the duration for card to dismiss or to come back to original position if not dismissed.

final

[onDismissed](https://api.flutter.dev/flutter/widgets/Dismissible/onDismissed.html) → [DismissDirectionCallback](https://api.flutter.dev/flutter/widgets/DismissDirectionCallback.html)?

Called when the widget has been dismissed, after finishing resizing.

final

[onResize](https://api.flutter.dev/flutter/widgets/Dismissible/onResize.html) → [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)?

Called when the widget changes size (i.e., when contracting before being dismissed).

final

[onUpdate](https://api.flutter.dev/flutter/widgets/Dismissible/onUpdate.html) → [DismissUpdateCallback](https://api.flutter.dev/flutter/widgets/DismissUpdateCallback.html)?

Called when the dismissible widget has been dragged.

final

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

The amount of time the widget will spend contracting before [onDismissed](https://api.flutter.dev/flutter/widgets/Dismissible/onDismissed.html) is called.

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

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

A widget that is stacked behind the child and is exposed when the child has been dragged up or to the left. It may only be specified when background has also been specified.

final

## DRAG TARGET

A widget that receives data when a [Draggable](https://api.flutter.dev/flutter/widgets/Draggable-class.html) widget is dropped.

When a draggable is dragged on top of a drag target, the drag target is asked whether it will accept the data the draggable is carrying. If the user does drop the draggable on top of the drag target (and the drag target has indicated that it will accept the draggable's data), then the drag target is asked to accept the draggable's data.

### Constructors

[DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget/DragTarget.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [DragTargetBuilder](https://api.flutter.dev/flutter/widgets/DragTargetBuilder.html)<T> builder, [DragTargetWillAccept](https://api.flutter.dev/flutter/widgets/DragTargetWillAccept.html)<T>? onWillAccept, [DragTargetWillAcceptWithDetails](https://api.flutter.dev/flutter/widgets/DragTargetWillAcceptWithDetails.html)<T>? onWillAcceptWithDetails, [DragTargetAccept](https://api.flutter.dev/flutter/widgets/DragTargetAccept.html)<T>? onAccept, [DragTargetAcceptWithDetails](https://api.flutter.dev/flutter/widgets/DragTargetAcceptWithDetails.html)<T>? onAcceptWithDetails, [DragTargetLeave](https://api.flutter.dev/flutter/widgets/DragTargetLeave.html)<T>? onLeave, [DragTargetMove](https://api.flutter.dev/flutter/widgets/DragTargetMove.html)<T>? onMove, [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html) hitTestBehavior = HitTestBehavior.translucent})

Creates a widget that receives drags.

const

### Properties

[builder](https://api.flutter.dev/flutter/widgets/DragTarget/builder.html) → [DragTargetBuilder](https://api.flutter.dev/flutter/widgets/DragTargetBuilder.html)<T>

Called to build the contents of this widget.

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

[hitTestBehavior](https://api.flutter.dev/flutter/widgets/DragTarget/hitTestBehavior.html) → [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html)

How to behave during hit testing.

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

[onAccept](https://api.flutter.dev/flutter/widgets/DragTarget/onAccept.html) → [DragTargetAccept](https://api.flutter.dev/flutter/widgets/DragTargetAccept.html)<T>?

Called when an acceptable piece of data was dropped over this drag target.

final

[onAcceptWithDetails](https://api.flutter.dev/flutter/widgets/DragTarget/onAcceptWithDetails.html) → [DragTargetAcceptWithDetails](https://api.flutter.dev/flutter/widgets/DragTargetAcceptWithDetails.html)<T>?

Called when an acceptable piece of data was dropped over this drag target.

final

[onLeave](https://api.flutter.dev/flutter/widgets/DragTarget/onLeave.html) → [DragTargetLeave](https://api.flutter.dev/flutter/widgets/DragTargetLeave.html)<T>?

Called when a given piece of data being dragged over this target leaves the target.

final

[onMove](https://api.flutter.dev/flutter/widgets/DragTarget/onMove.html) → [DragTargetMove](https://api.flutter.dev/flutter/widgets/DragTargetMove.html)<T>?

Called when a [Draggable](https://api.flutter.dev/flutter/widgets/Draggable-class.html) moves within this [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html).

final

[onWillAccept](https://api.flutter.dev/flutter/widgets/DragTarget/onWillAccept.html) → [DragTargetWillAccept](https://api.flutter.dev/flutter/widgets/DragTargetWillAccept.html)<T>?

Called to determine whether this widget is interested in receiving a given piece of data being dragged over this drag target.

final

[onWillAcceptWithDetails](https://api.flutter.dev/flutter/widgets/DragTarget/onWillAcceptWithDetails.html) → [DragTargetWillAcceptWithDetails](https://api.flutter.dev/flutter/widgets/DragTargetWillAcceptWithDetails.html)<T>?

Called to determine whether this widget is interested in receiving a given piece of data being dragged over this drag target.

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

## DRAGGABLE

A widget that can be dragged from to a [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html).

When a draggable widget recognizes the start of a drag gesture, it displays a [feedback](https://api.flutter.dev/flutter/widgets/Draggable/feedback.html) widget that tracks the user's finger across the screen. If the user lifts their finger while on top of a [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html), that target is given the opportunity to accept the [data](https://api.flutter.dev/flutter/widgets/Draggable/data.html) carried by the draggable.

The [ignoringFeedbackPointer](https://api.flutter.dev/flutter/widgets/Draggable/ignoringFeedbackPointer.html) defaults to true, which means that the [feedback](https://api.flutter.dev/flutter/widgets/Draggable/feedback.html) widget ignores the pointer during hit testing. Similarly, [ignoringFeedbackSemantics](https://api.flutter.dev/flutter/widgets/Draggable/ignoringFeedbackSemantics.html) defaults to true, and the [feedback](https://api.flutter.dev/flutter/widgets/Draggable/feedback.html) also ignores semantics when building the semantics tree.

On multitouch devices, multiple drags can occur simultaneously because there can be multiple pointers in contact with the device at once. To limit the number of simultaneous drags, use the [maxSimultaneousDrags](https://api.flutter.dev/flutter/widgets/Draggable/maxSimultaneousDrags.html) property. The default is to allow an unlimited number of simultaneous drags.

This widget displays [child](https://api.flutter.dev/flutter/widgets/Draggable/child.html) when zero drags are under way. If [childWhenDragging](https://api.flutter.dev/flutter/widgets/Draggable/childWhenDragging.html) is non-null, this widget instead displays [childWhenDragging](https://api.flutter.dev/flutter/widgets/Draggable/childWhenDragging.html) when one or more drags are underway. Otherwise, this widget always displays [child](https://api.flutter.dev/flutter/widgets/Draggable/child.html)

### Constructors

[Draggable](https://api.flutter.dev/flutter/widgets/Draggable/Draggable.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) feedback, T? data, [Axis](https://api.flutter.dev/flutter/painting/Axis.html)? axis, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? childWhenDragging, [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html) feedbackOffset = Offset.zero, [DragAnchorStrategy](https://api.flutter.dev/flutter/widgets/DragAnchorStrategy.html) dragAnchorStrategy = childDragAnchorStrategy, [Axis](https://api.flutter.dev/flutter/painting/Axis.html)? affinity, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? maxSimultaneousDrags, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onDragStarted, [DragUpdateCallback](https://api.flutter.dev/flutter/widgets/DragUpdateCallback.html)? onDragUpdate, [DraggableCanceledCallback](https://api.flutter.dev/flutter/widgets/DraggableCanceledCallback.html)? onDraggableCanceled, [DragEndCallback](https://api.flutter.dev/flutter/widgets/DragEndCallback.html)? onDragEnd, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onDragCompleted, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) ignoringFeedbackSemantics = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) ignoringFeedbackPointer = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) rootOverlay = false, [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html) hitTestBehavior = HitTestBehavior.deferToChild, [AllowedButtonsFilter](https://api.flutter.dev/flutter/gestures/AllowedButtonsFilter.html)? allowedButtonsFilter})

Creates a widget that can be dragged to a [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html).

const

### Properties

[affinity](https://api.flutter.dev/flutter/widgets/Draggable/affinity.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)?

Controls how this widget competes with other gestures to initiate a drag.

final

[allowedButtonsFilter](https://api.flutter.dev/flutter/widgets/Draggable/allowedButtonsFilter.html) → [AllowedButtonsFilter](https://api.flutter.dev/flutter/gestures/AllowedButtonsFilter.html)?

Called when interaction starts. This limits the dragging behavior for custom clicks (such as scroll click). Its parameter comes from [PointerEvent.buttons](https://api.flutter.dev/flutter/gestures/PointerEvent/buttons.html).

final

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

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) to restrict this draggable's movement, if specified.

final

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

The widget below this widget in the tree.

final

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

The widget to display instead of [child](https://api.flutter.dev/flutter/widgets/Draggable/child.html) when one or more drags are under way.

final

[data](https://api.flutter.dev/flutter/widgets/Draggable/data.html) → T?

The data that will be dropped by this draggable.

final

[dragAnchorStrategy](https://api.flutter.dev/flutter/widgets/Draggable/dragAnchorStrategy.html) → [DragAnchorStrategy](https://api.flutter.dev/flutter/widgets/DragAnchorStrategy.html)

A strategy that is used by this draggable to get the anchor offset when it is dragged.

final

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

The widget to show under the pointer when a drag is under way.

final

[feedbackOffset](https://api.flutter.dev/flutter/widgets/Draggable/feedbackOffset.html) → [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html)

The feedbackOffset can be used to set the hit test target point for the purposes of finding a drag target. It is especially useful if the feedback is transformed compared to the child.

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

[hitTestBehavior](https://api.flutter.dev/flutter/widgets/Draggable/hitTestBehavior.html) → [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html)

How to behave during hit test.

final

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

Whether the [feedback](https://api.flutter.dev/flutter/widgets/Draggable/feedback.html) widget is ignored during hit testing.

final

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

Whether the semantics of the [feedback](https://api.flutter.dev/flutter/widgets/Draggable/feedback.html) widget is ignored when building the semantics tree.

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

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

How many simultaneous drags to support.

final

[onDragCompleted](https://api.flutter.dev/flutter/widgets/Draggable/onDragCompleted.html) → [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)?

Called when the draggable is dropped and accepted by a [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html).

final

[onDragEnd](https://api.flutter.dev/flutter/widgets/Draggable/onDragEnd.html) → [DragEndCallback](https://api.flutter.dev/flutter/widgets/DragEndCallback.html)?

Called when the draggable is dropped.

final

[onDraggableCanceled](https://api.flutter.dev/flutter/widgets/Draggable/onDraggableCanceled.html) → [DraggableCanceledCallback](https://api.flutter.dev/flutter/widgets/DraggableCanceledCallback.html)?

Called when the draggable is dropped without being accepted by a [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html).

final

[onDragStarted](https://api.flutter.dev/flutter/widgets/Draggable/onDragStarted.html) → [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)?

Called when the draggable starts being dragged.

final

[onDragUpdate](https://api.flutter.dev/flutter/widgets/Draggable/onDragUpdate.html) → [DragUpdateCallback](https://api.flutter.dev/flutter/widgets/DragUpdateCallback.html)?

Called when the draggable is dragged.

final

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

Whether the feedback widget will be put on the root [Overlay](https://api.flutter.dev/flutter/widgets/Overlay-class.html).

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

## 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

## GESTURE DETECTOR

A widget that detects gestures.

Attempts to recognize gestures that correspond to its non-null callbacks.

If this widget has a child, it defers to that child for its sizing behavior. If it does not have a child, it grows to fit the parent instead.

By default a GestureDetector with an invisible child ignores touches; this behavior can be controlled with [behavior](https://api.flutter.dev/flutter/widgets/GestureDetector/behavior.html).

GestureDetector also listens for accessibility events and maps them to the callbacks. To ignore accessibility events, set [excludeFromSemantics](https://api.flutter.dev/flutter/widgets/GestureDetector/excludeFromSemantics.html) to true.

See [flutter.dev/gestures/](http://flutter.dev/gestures/) for additional information.

Material design applications typically react to touches with ink splash effects. The [InkWell](https://api.flutter.dev/flutter/material/InkWell-class.html) class implements this effect and can be used in place of a [GestureDetector](https://api.flutter.dev/flutter/widgets/GestureDetector-class.html) for handling taps.

### Troubleshooting

Why isn't my parent [GestureDetector.onTap](https://api.flutter.dev/flutter/widgets/GestureDetector/onTap.html) method called?

Given a parent [GestureDetector](https://api.flutter.dev/flutter/widgets/GestureDetector-class.html) with an onTap callback, and a child GestureDetector that also defines an onTap callback, when the inner GestureDetector is tapped, both GestureDetectors send a [GestureRecognizer](https://api.flutter.dev/flutter/gestures/GestureRecognizer-class.html) into the gesture arena. This is because the pointer coordinates are within the bounds of both GestureDetectors. The child GestureDetector wins in this scenario because it was the first to enter the arena, resolving as first come, first served. The child onTap is called, and the parent's is not as the gesture has been consumed. For more information on gesture disambiguation see: [Gesture disambiguation](https://docs.flutter.dev/development/ui/advanced/gestures#gesture-disambiguation).

Setting [GestureDetector.behavior](https://api.flutter.dev/flutter/widgets/GestureDetector/behavior.html) to [HitTestBehavior.opaque](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html) or [HitTestBehavior.translucent](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html) has no impact on parent-child relationships: both GestureDetectors send a GestureRecognizer into the gesture arena, only one wins.

Some callbacks (e.g. onTapDown) can fire before a recognizer wins the arena, and others (e.g. onTapCancel) fire even when it loses the arena. Therefore, the parent detector in the example above may call some of its callbacks even though it loses in the arena.

### Debugging

To see how large the hit test box of a [GestureDetector](https://api.flutter.dev/flutter/widgets/GestureDetector-class.html) is for debugging purposes, set [debugPaintPointersEnabled](https://api.flutter.dev/flutter/rendering/debugPaintPointersEnabled.html) to true.

See also:

* [Listener](https://api.flutter.dev/flutter/widgets/Listener-class.html), a widget for listening to lower-level raw pointer events.
* [MouseRegion](https://api.flutter.dev/flutter/widgets/MouseRegion-class.html), a widget that tracks the movement of mice, even when no button is pressed.
* [RawGestureDetector](https://api.flutter.dev/flutter/widgets/RawGestureDetector-class.html), a widget that is used to detect custom gestures.

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)
* GestureDetector

### Constructors

[GestureDetector](https://api.flutter.dev/flutter/widgets/GestureDetector/GestureDetector.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)? onTapDown, [GestureTapUpCallback](https://api.flutter.dev/flutter/gestures/GestureTapUpCallback.html)? onTapUp, [GestureTapCallback](https://api.flutter.dev/flutter/gestures/GestureTapCallback.html)? onTap, [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)? onTapCancel, [GestureTapCallback](https://api.flutter.dev/flutter/gestures/GestureTapCallback.html)? onSecondaryTap, [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)? onSecondaryTapDown, [GestureTapUpCallback](https://api.flutter.dev/flutter/gestures/GestureTapUpCallback.html)? onSecondaryTapUp, [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)? onSecondaryTapCancel, [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)? onTertiaryTapDown, [GestureTapUpCallback](https://api.flutter.dev/flutter/gestures/GestureTapUpCallback.html)? onTertiaryTapUp, [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)? onTertiaryTapCancel, [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)? onDoubleTapDown, [GestureTapCallback](https://api.flutter.dev/flutter/gestures/GestureTapCallback.html)? onDoubleTap, [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)? onDoubleTapCancel, [GestureLongPressDownCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressDownCallback.html)? onLongPressDown, [GestureLongPressCancelCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCancelCallback.html)? onLongPressCancel, [GestureLongPressCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCallback.html)? onLongPress, [GestureLongPressStartCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressStartCallback.html)? onLongPressStart, [GestureLongPressMoveUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressMoveUpdateCallback.html)? onLongPressMoveUpdate, [GestureLongPressUpCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressUpCallback.html)? onLongPressUp, [GestureLongPressEndCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressEndCallback.html)? onLongPressEnd, [GestureLongPressDownCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressDownCallback.html)? onSecondaryLongPressDown, [GestureLongPressCancelCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCancelCallback.html)? onSecondaryLongPressCancel, [GestureLongPressCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCallback.html)? onSecondaryLongPress, [GestureLongPressStartCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressStartCallback.html)? onSecondaryLongPressStart, [GestureLongPressMoveUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressMoveUpdateCallback.html)? onSecondaryLongPressMoveUpdate, [GestureLongPressUpCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressUpCallback.html)? onSecondaryLongPressUp, [GestureLongPressEndCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressEndCallback.html)? onSecondaryLongPressEnd, [GestureLongPressDownCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressDownCallback.html)? onTertiaryLongPressDown, [GestureLongPressCancelCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCancelCallback.html)? onTertiaryLongPressCancel, [GestureLongPressCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCallback.html)? onTertiaryLongPress, [GestureLongPressStartCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressStartCallback.html)? onTertiaryLongPressStart, [GestureLongPressMoveUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressMoveUpdateCallback.html)? onTertiaryLongPressMoveUpdate, [GestureLongPressUpCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressUpCallback.html)? onTertiaryLongPressUp, [GestureLongPressEndCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressEndCallback.html)? onTertiaryLongPressEnd, [GestureDragDownCallback](https://api.flutter.dev/flutter/gestures/GestureDragDownCallback.html)? onVerticalDragDown, [GestureDragStartCallback](https://api.flutter.dev/flutter/gestures/GestureDragStartCallback.html)? onVerticalDragStart, [GestureDragUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureDragUpdateCallback.html)? onVerticalDragUpdate, [GestureDragEndCallback](https://api.flutter.dev/flutter/gestures/GestureDragEndCallback.html)? onVerticalDragEnd, [GestureDragCancelCallback](https://api.flutter.dev/flutter/gestures/GestureDragCancelCallback.html)? onVerticalDragCancel, [GestureDragDownCallback](https://api.flutter.dev/flutter/gestures/GestureDragDownCallback.html)? onHorizontalDragDown, [GestureDragStartCallback](https://api.flutter.dev/flutter/gestures/GestureDragStartCallback.html)? onHorizontalDragStart, [GestureDragUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureDragUpdateCallback.html)? onHorizontalDragUpdate, [GestureDragEndCallback](https://api.flutter.dev/flutter/gestures/GestureDragEndCallback.html)? onHorizontalDragEnd, [GestureDragCancelCallback](https://api.flutter.dev/flutter/gestures/GestureDragCancelCallback.html)? onHorizontalDragCancel, [GestureForcePressStartCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressStartCallback.html)? onForcePressStart, [GestureForcePressPeakCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressPeakCallback.html)? onForcePressPeak, [GestureForcePressUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressUpdateCallback.html)? onForcePressUpdate, [GestureForcePressEndCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressEndCallback.html)? onForcePressEnd, [GestureDragDownCallback](https://api.flutter.dev/flutter/gestures/GestureDragDownCallback.html)? onPanDown, [GestureDragStartCallback](https://api.flutter.dev/flutter/gestures/GestureDragStartCallback.html)? onPanStart, [GestureDragUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureDragUpdateCallback.html)? onPanUpdate, [GestureDragEndCallback](https://api.flutter.dev/flutter/gestures/GestureDragEndCallback.html)? onPanEnd, [GestureDragCancelCallback](https://api.flutter.dev/flutter/gestures/GestureDragCancelCallback.html)? onPanCancel, [GestureScaleStartCallback](https://api.flutter.dev/flutter/gestures/GestureScaleStartCallback.html)? onScaleStart, [GestureScaleUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureScaleUpdateCallback.html)? onScaleUpdate, [GestureScaleEndCallback](https://api.flutter.dev/flutter/gestures/GestureScaleEndCallback.html)? onScaleEnd, [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html)? behavior, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) dragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) trackpadScrollCausesScale = false, [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html) trackpadScrollToScaleFactor = kDefaultTrackpadScrollToScaleFactor, [Set](https://api.flutter.dev/flutter/dart-core/Set-class.html)<[PointerDeviceKind](https://api.flutter.dev/flutter/dart-ui/PointerDeviceKind.html)>? supportedDevices})

Creates a widget that detects gestures.

### Properties

[behavior](https://api.flutter.dev/flutter/widgets/GestureDetector/behavior.html) → [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html)?

How this gesture detector should behave during hit testing when deciding how the hit test propagates to children and whether to consider targets behind this one.

final

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

The widget below this widget in the tree.

final

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

Whether to exclude these gestures from the semantics tree. For example, the long-press gesture for showing a tooltip is excluded because the tooltip itself is included in the semantics tree directly and so having a gesture to show it would result in duplication of information.

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

[onDoubleTap](https://api.flutter.dev/flutter/widgets/GestureDetector/onDoubleTap.html) → [GestureTapCallback](https://api.flutter.dev/flutter/gestures/GestureTapCallback.html)?

The user has tapped the screen with a primary button at the same location twice in quick succession.

final

[onDoubleTapCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onDoubleTapCancel.html) → [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)?

The pointer that previously triggered [onDoubleTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onDoubleTapDown.html) will not end up causing a double tap.

final

[onDoubleTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onDoubleTapDown.html) → [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)?

A pointer that might cause a double tap has contacted the screen at a particular location.

final

[onForcePressEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onForcePressEnd.html) → [GestureForcePressEndCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressEndCallback.html)?

The pointer tracked by [onForcePressStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onForcePressStart.html) is no longer in contact with the screen.

final

[onForcePressPeak](https://api.flutter.dev/flutter/widgets/GestureDetector/onForcePressPeak.html) → [GestureForcePressPeakCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressPeakCallback.html)?

The pointer is in contact with the screen and has pressed with the maximum force. The amount of force is at least [ForcePressGestureRecognizer.peakPressure](https://api.flutter.dev/flutter/gestures/ForcePressGestureRecognizer/peakPressure.html).

final

[onForcePressStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onForcePressStart.html) → [GestureForcePressStartCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressStartCallback.html)?

The pointer is in contact with the screen and has pressed with sufficient force to initiate a force press. The amount of force is at least [ForcePressGestureRecognizer.startPressure](https://api.flutter.dev/flutter/gestures/ForcePressGestureRecognizer/startPressure.html).

final

[onForcePressUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onForcePressUpdate.html) → [GestureForcePressUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureForcePressUpdateCallback.html)?

A pointer is in contact with the screen, has previously passed the [ForcePressGestureRecognizer.startPressure](https://api.flutter.dev/flutter/gestures/ForcePressGestureRecognizer/startPressure.html) and is either moving on the plane of the screen, pressing the screen with varying forces or both simultaneously.

final

[onHorizontalDragCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onHorizontalDragCancel.html) → [GestureDragCancelCallback](https://api.flutter.dev/flutter/gestures/GestureDragCancelCallback.html)?

The pointer that previously triggered [onHorizontalDragDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onHorizontalDragDown.html) did not complete.

final

[onHorizontalDragDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onHorizontalDragDown.html) → [GestureDragDownCallback](https://api.flutter.dev/flutter/gestures/GestureDragDownCallback.html)?

A pointer has contacted the screen with a primary button and might begin to move horizontally.

final

[onHorizontalDragEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onHorizontalDragEnd.html) → [GestureDragEndCallback](https://api.flutter.dev/flutter/gestures/GestureDragEndCallback.html)?

A pointer that was previously in contact with the screen with a primary button and moving horizontally is no longer in contact with the screen and was moving at a specific velocity when it stopped contacting the screen.

final

[onHorizontalDragStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onHorizontalDragStart.html) → [GestureDragStartCallback](https://api.flutter.dev/flutter/gestures/GestureDragStartCallback.html)?

A pointer has contacted the screen with a primary button and has begun to move horizontally.

final

[onHorizontalDragUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onHorizontalDragUpdate.html) → [GestureDragUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureDragUpdateCallback.html)?

A pointer that is in contact with the screen with a primary button and moving horizontally has moved in the horizontal direction.

final

[onLongPress](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPress.html) → [GestureLongPressCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCallback.html)?

Called when a long press gesture with a primary button has been recognized.

final

[onLongPressCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPressCancel.html) → [GestureLongPressCancelCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCancelCallback.html)?

A pointer that previously triggered [onLongPressDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPressDown.html) will not end up causing a long-press.

final

[onLongPressDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPressDown.html) → [GestureLongPressDownCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressDownCallback.html)?

The pointer has contacted the screen with a primary button, which might be the start of a long-press.

final

[onLongPressEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPressEnd.html) → [GestureLongPressEndCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressEndCallback.html)?

A pointer that has triggered a long-press with a primary button has stopped contacting the screen.

final

[onLongPressMoveUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPressMoveUpdate.html) → [GestureLongPressMoveUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressMoveUpdateCallback.html)?

A pointer has been drag-moved after a long-press with a primary button.

final

[onLongPressStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPressStart.html) → [GestureLongPressStartCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressStartCallback.html)?

Called when a long press gesture with a primary button has been recognized.

final

[onLongPressUp](https://api.flutter.dev/flutter/widgets/GestureDetector/onLongPressUp.html) → [GestureLongPressUpCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressUpCallback.html)?

A pointer that has triggered a long-press with a primary button has stopped contacting the screen.

final

[onPanCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onPanCancel.html) → [GestureDragCancelCallback](https://api.flutter.dev/flutter/gestures/GestureDragCancelCallback.html)?

The pointer that previously triggered [onPanDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onPanDown.html) did not complete.

final

[onPanDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onPanDown.html) → [GestureDragDownCallback](https://api.flutter.dev/flutter/gestures/GestureDragDownCallback.html)?

A pointer has contacted the screen with a primary button and might begin to move.

final

[onPanEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onPanEnd.html) → [GestureDragEndCallback](https://api.flutter.dev/flutter/gestures/GestureDragEndCallback.html)?

A pointer that was previously in contact with the screen with a primary button and moving is no longer in contact with the screen and was moving at a specific velocity when it stopped contacting the screen.

final

[onPanStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onPanStart.html) → [GestureDragStartCallback](https://api.flutter.dev/flutter/gestures/GestureDragStartCallback.html)?

A pointer has contacted the screen with a primary button and has begun to move.

final

[onPanUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onPanUpdate.html) → [GestureDragUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureDragUpdateCallback.html)?

A pointer that is in contact with the screen with a primary button and moving has moved again.

final

[onScaleEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onScaleEnd.html) → [GestureScaleEndCallback](https://api.flutter.dev/flutter/gestures/GestureScaleEndCallback.html)?

The pointers are no longer in contact with the screen.

final

[onScaleStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onScaleStart.html) → [GestureScaleStartCallback](https://api.flutter.dev/flutter/gestures/GestureScaleStartCallback.html)?

The pointers in contact with the screen have established a focal point and initial scale of 1.0.

final

[onScaleUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onScaleUpdate.html) → [GestureScaleUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureScaleUpdateCallback.html)?

The pointers in contact with the screen have indicated a new focal point and/or scale.

final

[onSecondaryLongPress](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPress.html) → [GestureLongPressCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCallback.html)?

Called when a long press gesture with a secondary button has been recognized.

final

[onSecondaryLongPressCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPressCancel.html) → [GestureLongPressCancelCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCancelCallback.html)?

A pointer that previously triggered [onSecondaryLongPressDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPressDown.html) will not end up causing a long-press.

final

[onSecondaryLongPressDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPressDown.html) → [GestureLongPressDownCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressDownCallback.html)?

The pointer has contacted the screen with a secondary button, which might be the start of a long-press.

final

[onSecondaryLongPressEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPressEnd.html) → [GestureLongPressEndCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressEndCallback.html)?

A pointer that has triggered a long-press with a secondary button has stopped contacting the screen.

final

[onSecondaryLongPressMoveUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPressMoveUpdate.html) → [GestureLongPressMoveUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressMoveUpdateCallback.html)?

A pointer has been drag-moved after a long press with a secondary button.

final

[onSecondaryLongPressStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPressStart.html) → [GestureLongPressStartCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressStartCallback.html)?

Called when a long press gesture with a secondary button has been recognized.

final

[onSecondaryLongPressUp](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryLongPressUp.html) → [GestureLongPressUpCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressUpCallback.html)?

A pointer that has triggered a long-press with a secondary button has stopped contacting the screen.

final

[onSecondaryTap](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryTap.html) → [GestureTapCallback](https://api.flutter.dev/flutter/gestures/GestureTapCallback.html)?

A tap with a secondary button has occurred.

final

[onSecondaryTapCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryTapCancel.html) → [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)?

The pointer that previously triggered [onSecondaryTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryTapDown.html) will not end up causing a tap.

final

[onSecondaryTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryTapDown.html) → [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)?

A pointer that might cause a tap with a secondary button has contacted the screen at a particular location.

final

[onSecondaryTapUp](https://api.flutter.dev/flutter/widgets/GestureDetector/onSecondaryTapUp.html) → [GestureTapUpCallback](https://api.flutter.dev/flutter/gestures/GestureTapUpCallback.html)?

A pointer that will trigger a tap with a secondary button has stopped contacting the screen at a particular location.

final

[onTap](https://api.flutter.dev/flutter/widgets/GestureDetector/onTap.html) → [GestureTapCallback](https://api.flutter.dev/flutter/gestures/GestureTapCallback.html)?

A tap with a primary button has occurred.

final

[onTapCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onTapCancel.html) → [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)?

The pointer that previously triggered [onTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onTapDown.html) will not end up causing a tap.

final

[onTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onTapDown.html) → [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)?

A pointer that might cause a tap with a primary button has contacted the screen at a particular location.

final

[onTapUp](https://api.flutter.dev/flutter/widgets/GestureDetector/onTapUp.html) → [GestureTapUpCallback](https://api.flutter.dev/flutter/gestures/GestureTapUpCallback.html)?

A pointer that will trigger a tap with a primary button has stopped contacting the screen at a particular location.

final

[onTertiaryLongPress](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPress.html) → [GestureLongPressCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCallback.html)?

Called when a long press gesture with a tertiary button has been recognized.

final

[onTertiaryLongPressCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPressCancel.html) → [GestureLongPressCancelCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressCancelCallback.html)?

A pointer that previously triggered [onTertiaryLongPressDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPressDown.html) will not end up causing a long-press.

final

[onTertiaryLongPressDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPressDown.html) → [GestureLongPressDownCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressDownCallback.html)?

The pointer has contacted the screen with a tertiary button, which might be the start of a long-press.

final

[onTertiaryLongPressEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPressEnd.html) → [GestureLongPressEndCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressEndCallback.html)?

A pointer that has triggered a long-press with a tertiary button has stopped contacting the screen.

final

[onTertiaryLongPressMoveUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPressMoveUpdate.html) → [GestureLongPressMoveUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressMoveUpdateCallback.html)?

A pointer has been drag-moved after a long press with a tertiary button.

final

[onTertiaryLongPressStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPressStart.html) → [GestureLongPressStartCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressStartCallback.html)?

Called when a long press gesture with a tertiary button has been recognized.

final

[onTertiaryLongPressUp](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryLongPressUp.html) → [GestureLongPressUpCallback](https://api.flutter.dev/flutter/gestures/GestureLongPressUpCallback.html)?

A pointer that has triggered a long-press with a tertiary button has stopped contacting the screen.

final

[onTertiaryTapCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryTapCancel.html) → [GestureTapCancelCallback](https://api.flutter.dev/flutter/gestures/GestureTapCancelCallback.html)?

The pointer that previously triggered [onTertiaryTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryTapDown.html) will not end up causing a tap.

final

[onTertiaryTapDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryTapDown.html) → [GestureTapDownCallback](https://api.flutter.dev/flutter/gestures/GestureTapDownCallback.html)?

A pointer that might cause a tap with a tertiary button has contacted the screen at a particular location.

final

[onTertiaryTapUp](https://api.flutter.dev/flutter/widgets/GestureDetector/onTertiaryTapUp.html) → [GestureTapUpCallback](https://api.flutter.dev/flutter/gestures/GestureTapUpCallback.html)?

A pointer that will trigger a tap with a tertiary button has stopped contacting the screen at a particular location.

final

[onVerticalDragCancel](https://api.flutter.dev/flutter/widgets/GestureDetector/onVerticalDragCancel.html) → [GestureDragCancelCallback](https://api.flutter.dev/flutter/gestures/GestureDragCancelCallback.html)?

The pointer that previously triggered [onVerticalDragDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onVerticalDragDown.html) did not complete.

final

[onVerticalDragDown](https://api.flutter.dev/flutter/widgets/GestureDetector/onVerticalDragDown.html) → [GestureDragDownCallback](https://api.flutter.dev/flutter/gestures/GestureDragDownCallback.html)?

A pointer has contacted the screen with a primary button and might begin to move vertically.

final

[onVerticalDragEnd](https://api.flutter.dev/flutter/widgets/GestureDetector/onVerticalDragEnd.html) → [GestureDragEndCallback](https://api.flutter.dev/flutter/gestures/GestureDragEndCallback.html)?

A pointer that was previously in contact with the screen with a primary button and moving vertically is no longer in contact with the screen and was moving at a specific velocity when it stopped contacting the screen.

final

[onVerticalDragStart](https://api.flutter.dev/flutter/widgets/GestureDetector/onVerticalDragStart.html) → [GestureDragStartCallback](https://api.flutter.dev/flutter/gestures/GestureDragStartCallback.html)?

A pointer has contacted the screen with a primary button and has begun to move vertically.

final

[onVerticalDragUpdate](https://api.flutter.dev/flutter/widgets/GestureDetector/onVerticalDragUpdate.html) → [GestureDragUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureDragUpdateCallback.html)?

A pointer that is in contact with the screen with a primary button and moving vertically has moved in the vertical 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

[supportedDevices](https://api.flutter.dev/flutter/widgets/GestureDetector/supportedDevices.html) → [Set](https://api.flutter.dev/flutter/dart-core/Set-class.html)<[PointerDeviceKind](https://api.flutter.dev/flutter/dart-ui/PointerDeviceKind.html)>?

The kind of devices that are allowed to be recognized.

final

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

Whether scrolling up/down on a trackpad should cause scaling instead of panning.

final

[trackpadScrollToScaleFactor](https://api.flutter.dev/flutter/widgets/GestureDetector/trackpadScrollToScaleFactor.html) → [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html)

A factor to control the direction and magnitude of scale when converting trackpad scrolling.

final

## IGNORE POINTER

A widget that is invisible during hit testing.

When [ignoring](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoring.html) is true, this widget (and its subtree) is invisible to hit testing. It still consumes space during layout and paints its child as usual. It just cannot be the target of located events, because it returns false from [RenderBox.hitTest](https://api.flutter.dev/flutter/rendering/RenderBox/hitTest.html).

### Semantics

Using this class may also affect how the semantics subtree underneath is collected.

If [ignoring](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoring.html) is true, pointer-related [SemanticsAction](https://api.flutter.dev/flutter/dart-ui/SemanticsAction-class.html)s are removed from the semantics subtree. Otherwise, the subtree remains untouched.

The usages of [ignoringSemantics](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoringSemantics.html) are deprecated and not recommended. This property was introduced to workaround the semantics behavior of the [IgnorePointer](https://api.flutter.dev/flutter/widgets/IgnorePointer-class.html) and its friends before v3.8.0-12.0.pre.

Before that version, entire semantics subtree is dropped if [ignoring](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoring.html) is true. Developers can only use [ignoringSemantics](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoringSemantics.html) to preserver the semantics subtrees.

After that version, with [ignoring](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoring.html) set to true, it only prevents semantics user actions in the semantics subtree but leaves the other [SemanticsProperties](https://api.flutter.dev/flutter/semantics/SemanticsProperties-class.html) intact. Therefore, the [ignoringSemantics](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoringSemantics.html) is no longer needed.

If [ignoringSemantics](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoringSemantics.html) is true, the semantics subtree is dropped. Therefore, the subtree will be invisible to assistive technologies.

If [ignoringSemantics](https://api.flutter.dev/flutter/widgets/IgnorePointer/ignoringSemantics.html) is false, the semantics subtree is collected as usual.

See also:

* [AbsorbPointer](https://api.flutter.dev/flutter/widgets/AbsorbPointer-class.html), which also prevents its children from receiving pointer events but is itself visible to hit testing.
* [SliverIgnorePointer](https://api.flutter.dev/flutter/widgets/SliverIgnorePointer-class.html), the sliver version of this widget.

### Constructors

[IgnorePointer](https://api.flutter.dev/flutter/widgets/IgnorePointer/IgnorePointer.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) ignoring = true, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Use ExcludeSemantics or create a custom ignore pointer widget instead. ' 'This feature was deprecated after v3.8.0-12.0.pre.') [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? ignoringSemantics, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a widget that is invisible to hit testing.

const

### Properties

[child](https://api.flutter.dev/flutter/widgets/SingleChildRenderObjectWidget/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

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

Whether this widget is ignored during hit testing.

final

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

Whether the semantics of this widget is ignored when compiling the semantics subtree.

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

[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

## INTERACTIVE VIEWER

A widget that enables pan and zoom interactions with its child.

The user can transform the child by dragging to pan or pinching to zoom.

By default, InteractiveViewer clips its child using [Clip.hardEdge](https://api.flutter.dev/flutter/dart-ui/Clip.html). To prevent this behavior, consider setting [clipBehavior](https://api.flutter.dev/flutter/widgets/InteractiveViewer/clipBehavior.html) to [Clip.none](https://api.flutter.dev/flutter/dart-ui/Clip.html). When [clipBehavior](https://api.flutter.dev/flutter/widgets/InteractiveViewer/clipBehavior.html) is [Clip.none](https://api.flutter.dev/flutter/dart-ui/Clip.html), InteractiveViewer may draw outside of its original area of the screen, such as when a child is zoomed in and increases in size. However, it will not receive gestures outside of its original area. To prevent dead areas where InteractiveViewer does not receive gestures, don't set [clipBehavior](https://api.flutter.dev/flutter/widgets/InteractiveViewer/clipBehavior.html) or be sure that the InteractiveViewer widget is the size of the area that should be interactive.

See also:

* The [Flutter Gallery's transformations demo](https://github.com/flutter/gallery/blob/master/lib/demos/reference/transformations_demo.dart), which includes the use of InteractiveViewer.
* The [flutter-go demo](https://github.com/justinmc/flutter-go), which includes robust positioning of an InteractiveViewer child that works for all screen sizes and child sizes.
* The [Lazy Flutter Performance Session](https://www.youtube.com/watch?v=qax_nOpgz7E), which includes the use of an InteractiveViewer to performantly view subsets of a large set of widgets using the builder constructor.

### Constructors

[InteractiveViewer](https://api.flutter.dev/flutter/widgets/InteractiveViewer/InteractiveViewer.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Use panAxis instead. ' 'This feature was deprecated after v3.3.0-0.5.pre.') [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) alignPanAxis = false, [PanAxis](https://api.flutter.dev/flutter/widgets/PanAxis.html) panAxis = PanAxis.free, [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html) boundaryMargin = EdgeInsets.zero, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) constrained = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxScale = 2.5, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) minScale = 0.8, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) interactionEndFrictionCoefficient = \_kDrag, [GestureScaleEndCallback](https://api.flutter.dev/flutter/gestures/GestureScaleEndCallback.html)? onInteractionEnd, [GestureScaleStartCallback](https://api.flutter.dev/flutter/gestures/GestureScaleStartCallback.html)? onInteractionStart, [GestureScaleUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureScaleUpdateCallback.html)? onInteractionUpdate, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) panEnabled = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) scaleEnabled = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) scaleFactor = kDefaultMouseScrollToScaleFactor, [TransformationController](https://api.flutter.dev/flutter/widgets/TransformationController-class.html)? transformationController, [Alignment](https://api.flutter.dev/flutter/painting/Alignment-class.html)? alignment, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) trackpadScrollCausesScale = false, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child})

Create an InteractiveViewer.

[InteractiveViewer.builder](https://api.flutter.dev/flutter/widgets/InteractiveViewer/InteractiveViewer.builder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Use panAxis instead. ' 'This feature was deprecated after v3.3.0-0.5.pre.') [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) alignPanAxis = false, [PanAxis](https://api.flutter.dev/flutter/widgets/PanAxis.html) panAxis = PanAxis.free, [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html) boundaryMargin = EdgeInsets.zero, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) maxScale = 2.5, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) minScale = 0.8, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) interactionEndFrictionCoefficient = \_kDrag, [GestureScaleEndCallback](https://api.flutter.dev/flutter/gestures/GestureScaleEndCallback.html)? onInteractionEnd, [GestureScaleStartCallback](https://api.flutter.dev/flutter/gestures/GestureScaleStartCallback.html)? onInteractionStart, [GestureScaleUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureScaleUpdateCallback.html)? onInteractionUpdate, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) panEnabled = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) scaleEnabled = true, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) scaleFactor = 200.0, [TransformationController](https://api.flutter.dev/flutter/widgets/TransformationController-class.html)? transformationController, [Alignment](https://api.flutter.dev/flutter/painting/Alignment-class.html)? alignment, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) trackpadScrollCausesScale = false, required [InteractiveViewerWidgetBuilder](https://api.flutter.dev/flutter/widgets/InteractiveViewerWidgetBuilder.html) builder})

Creates an InteractiveViewer for a child that is created on demand.

### Properties

[alignment](https://api.flutter.dev/flutter/widgets/InteractiveViewer/alignment.html) → [Alignment](https://api.flutter.dev/flutter/painting/Alignment-class.html)?

The alignment of the child's origin, relative to the size of the box.

final

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

This property is deprecated, please use [panAxis](https://api.flutter.dev/flutter/widgets/InteractiveViewer/panAxis.html) instead.

final

[boundaryMargin](https://api.flutter.dev/flutter/widgets/InteractiveViewer/boundaryMargin.html) → [EdgeInsets](https://api.flutter.dev/flutter/painting/EdgeInsets-class.html)

A margin for the visible boundaries of the child.

final

[builder](https://api.flutter.dev/flutter/widgets/InteractiveViewer/builder.html) → [InteractiveViewerWidgetBuilder](https://api.flutter.dev/flutter/widgets/InteractiveViewerWidgetBuilder.html)?

Builds the child of this widget.

final

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

The child [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) that is transformed by InteractiveViewer.

final

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

If set to [Clip.none](https://api.flutter.dev/flutter/dart-ui/Clip.html), the child may extend beyond the size of the InteractiveViewer, but it will not receive gestures in these areas. Be sure that the InteractiveViewer is the desired size when using [Clip.none](https://api.flutter.dev/flutter/dart-ui/Clip.html).

final

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

Whether the normal size constraints at this point in the widget tree are applied to the child.

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

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

Changes the deceleration behavior after a gesture.

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

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

The maximum allowed scale.

final

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

The minimum allowed scale.

final

[onInteractionEnd](https://api.flutter.dev/flutter/widgets/InteractiveViewer/onInteractionEnd.html) → [GestureScaleEndCallback](https://api.flutter.dev/flutter/gestures/GestureScaleEndCallback.html)?

Called when the user ends a pan or scale gesture on the widget.

final

[onInteractionStart](https://api.flutter.dev/flutter/widgets/InteractiveViewer/onInteractionStart.html) → [GestureScaleStartCallback](https://api.flutter.dev/flutter/gestures/GestureScaleStartCallback.html)?

Called when the user begins a pan or scale gesture on the widget.

final

[onInteractionUpdate](https://api.flutter.dev/flutter/widgets/InteractiveViewer/onInteractionUpdate.html) → [GestureScaleUpdateCallback](https://api.flutter.dev/flutter/gestures/GestureScaleUpdateCallback.html)?

Called when the user updates a pan or scale gesture on the widget.

final

[panAxis](https://api.flutter.dev/flutter/widgets/InteractiveViewer/panAxis.html) → [PanAxis](https://api.flutter.dev/flutter/widgets/PanAxis.html)

When set to [PanAxis.aligned](https://api.flutter.dev/flutter/widgets/PanAxis.html), panning is only allowed in the horizontal axis or the vertical axis, diagonal panning is not allowed.

final

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

If false, the user will be prevented from panning.

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

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

If false, the user will be prevented from scaling.

final

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

Determines the amount of scale to be performed per pointer scroll.

final

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

Whether scrolling up/down on a trackpad should cause scaling instead of panning.

final

[transformationController](https://api.flutter.dev/flutter/widgets/InteractiveViewer/transformationController.html) → [TransformationController](https://api.flutter.dev/flutter/widgets/TransformationController-class.html)?

A [TransformationController](https://api.flutter.dev/flutter/widgets/TransformationController-class.html) for the transformation performed on the child.

final

## LONG PRESS DRAGGABLE

Makes its child draggable starting from long press.

See also:

* [Draggable](https://api.flutter.dev/flutter/widgets/Draggable-class.html), similar to the [LongPressDraggable](https://api.flutter.dev/flutter/widgets/LongPressDraggable-class.html) widget but happens immediately.
* [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html), a widget that receives data when a [Draggable](https://api.flutter.dev/flutter/widgets/Draggable-class.html) widget is dropped.

### Constructors

[LongPressDraggable](https://api.flutter.dev/flutter/widgets/LongPressDraggable/LongPressDraggable.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) feedback, T? data, [Axis](https://api.flutter.dev/flutter/painting/Axis.html)? axis, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? childWhenDragging, [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html) feedbackOffset = Offset.zero, [DragAnchorStrategy](https://api.flutter.dev/flutter/widgets/DragAnchorStrategy.html) dragAnchorStrategy = childDragAnchorStrategy, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? maxSimultaneousDrags, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onDragStarted, [DragUpdateCallback](https://api.flutter.dev/flutter/widgets/DragUpdateCallback.html)? onDragUpdate, [DraggableCanceledCallback](https://api.flutter.dev/flutter/widgets/DraggableCanceledCallback.html)? onDraggableCanceled, [DragEndCallback](https://api.flutter.dev/flutter/widgets/DragEndCallback.html)? onDragEnd, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onDragCompleted, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) hapticFeedbackOnStart = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) ignoringFeedbackSemantics = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) ignoringFeedbackPointer = true, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) delay = kLongPressTimeout, [AllowedButtonsFilter](https://api.flutter.dev/flutter/gestures/AllowedButtonsFilter.html)? allowedButtonsFilter})

Creates a widget that can be dragged starting from long press.

const

### Properties

[affinity](https://api.flutter.dev/flutter/widgets/Draggable/affinity.html) → [Axis](https://api.flutter.dev/flutter/painting/Axis.html)?

Controls how this widget competes with other gestures to initiate a drag.

finalinherited

[allowedButtonsFilter](https://api.flutter.dev/flutter/widgets/Draggable/allowedButtonsFilter.html) → [AllowedButtonsFilter](https://api.flutter.dev/flutter/gestures/AllowedButtonsFilter.html)?

Called when interaction starts. This limits the dragging behavior for custom clicks (such as scroll click). Its parameter comes from [PointerEvent.buttons](https://api.flutter.dev/flutter/gestures/PointerEvent/buttons.html).

finalinherited

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

The [Axis](https://api.flutter.dev/flutter/painting/Axis.html) to restrict this draggable's movement, if specified.

finalinherited

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

The widget below this widget in the tree.

finalinherited

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

The widget to display instead of [child](https://api.flutter.dev/flutter/widgets/Draggable/child.html) when one or more drags are under way.

finalinherited

[data](https://api.flutter.dev/flutter/widgets/Draggable/data.html) → T?

The data that will be dropped by this draggable.

finalinherited

[delay](https://api.flutter.dev/flutter/widgets/LongPressDraggable/delay.html) → [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)

The duration that a user has to press down before a long press is registered.

final

[dragAnchorStrategy](https://api.flutter.dev/flutter/widgets/Draggable/dragAnchorStrategy.html) → [DragAnchorStrategy](https://api.flutter.dev/flutter/widgets/DragAnchorStrategy.html)

A strategy that is used by this draggable to get the anchor offset when it is dragged.

finalinherited

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

The widget to show under the pointer when a drag is under way.

finalinherited

[feedbackOffset](https://api.flutter.dev/flutter/widgets/Draggable/feedbackOffset.html) → [Offset](https://api.flutter.dev/flutter/dart-ui/Offset-class.html)

The feedbackOffset can be used to set the hit test target point for the purposes of finding a drag target. It is especially useful if the feedback is transformed compared to the child.

finalinherited

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

Whether haptic feedback should be triggered 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

[hitTestBehavior](https://api.flutter.dev/flutter/widgets/Draggable/hitTestBehavior.html) → [HitTestBehavior](https://api.flutter.dev/flutter/rendering/HitTestBehavior.html)

How to behave during hit test.

finalinherited

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

Whether the [feedback](https://api.flutter.dev/flutter/widgets/Draggable/feedback.html) widget is ignored during hit testing.

finalinherited

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

Whether the semantics of the [feedback](https://api.flutter.dev/flutter/widgets/Draggable/feedback.html) widget is ignored when building the semantics tree.

finalinherited

[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

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

How many simultaneous drags to support.

finalinherited

[onDragCompleted](https://api.flutter.dev/flutter/widgets/Draggable/onDragCompleted.html) → [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)?

Called when the draggable is dropped and accepted by a [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html).

finalinherited

[onDragEnd](https://api.flutter.dev/flutter/widgets/Draggable/onDragEnd.html) → [DragEndCallback](https://api.flutter.dev/flutter/widgets/DragEndCallback.html)?

Called when the draggable is dropped.

finalinherited

[onDraggableCanceled](https://api.flutter.dev/flutter/widgets/Draggable/onDraggableCanceled.html) → [DraggableCanceledCallback](https://api.flutter.dev/flutter/widgets/DraggableCanceledCallback.html)?

Called when the draggable is dropped without being accepted by a [DragTarget](https://api.flutter.dev/flutter/widgets/DragTarget-class.html).

finalinherited

[onDragStarted](https://api.flutter.dev/flutter/widgets/Draggable/onDragStarted.html) → [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)?

Called when the draggable starts being dragged.

finalinherited

[onDragUpdate](https://api.flutter.dev/flutter/widgets/Draggable/onDragUpdate.html) → [DragUpdateCallback](https://api.flutter.dev/flutter/widgets/DragUpdateCallback.html)?

Called when the draggable is dragged.

finalinherited

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

Whether the feedback widget will be put on the root [Overlay](https://api.flutter.dev/flutter/widgets/Overlay-class.html).

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

## 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).

### 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

# ROUTING

## HERO

A widget that marks its child as being a candidate for [hero animations](https://flutter.dev/docs/development/ui/animations/hero-animations).

When a [PageRoute](https://api.flutter.dev/flutter/widgets/PageRoute-class.html) is pushed or popped with the [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html), the entire screen's content is replaced. An old route disappears and a new route appears. If there's a common visual feature on both routes then it can be helpful for orienting the user for the feature to physically move from one page to the other during the routes' transition. Such an animation is called a *hero animation*. The hero widgets "fly" in the Navigator's overlay during the transition and while they're in-flight they're, by default, not shown in their original locations in the old and new routes.

To label a widget as such a feature, wrap it in a [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) widget. When navigation happens, the [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) widgets on each route are identified by the [HeroController](https://api.flutter.dev/flutter/widgets/HeroController-class.html). For each pair of [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) widgets that have the same tag, a hero animation is triggered.

If a [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) is already in flight when navigation occurs, its flight animation will be redirected to its new destination. The widget shown in-flight during the transition is, by default, the destination route's [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html)'s child.

For a Hero animation to trigger, the Hero has to exist on the very first frame of the new page's animation.

### Discussion

Heroes and the [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)'s [Overlay](https://api.flutter.dev/flutter/widgets/Overlay-class.html) [Stack](https://api.flutter.dev/flutter/widgets/Stack-class.html) must be axis-aligned for all this to work. The top left and bottom right coordinates of each animated Hero will be converted to global coordinates and then from there converted to that [Stack](https://api.flutter.dev/flutter/widgets/Stack-class.html)'s coordinate space, and the entire Hero subtree will, for the duration of the animation, be lifted out of its original place, and positioned on that stack. If the [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) isn't axis aligned, this is going to fail in a rather ugly fashion. Don't rotate your heroes!

To make the animations look good, it's critical that the widget tree for the hero in both locations be essentially identical. The widget of the target is, by default, used to do the transition: when going from route A to route B, route B's hero's widget is placed over route A's hero's widget. Additionally, if the [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) subtree changes appearance based on an [InheritedWidget](https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html) (such as [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html) or [Theme](https://api.flutter.dev/flutter/material/Theme-class.html)), then the hero animation may have discontinuity at the start or the end of the animation because route A and route B provides different such [InheritedWidget](https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html)s. Consider providing a custom [flightShuttleBuilder](https://api.flutter.dev/flutter/widgets/Hero/flightShuttleBuilder.html) to ensure smooth transitions. The default [flightShuttleBuilder](https://api.flutter.dev/flutter/widgets/Hero/flightShuttleBuilder.html) interpolates [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html)'s paddings. If your [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) widget uses custom [InheritedWidget](https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html)s and displays a discontinuity in the animation, try to provide custom in-flight transition using [flightShuttleBuilder](https://api.flutter.dev/flutter/widgets/Hero/flightShuttleBuilder.html).

By default, both route A and route B's heroes are hidden while the transitioning widget is animating in-flight above the 2 routes. [placeholderBuilder](https://api.flutter.dev/flutter/widgets/Hero/placeholderBuilder.html) can be used to show a custom widget in their place instead once the transition has taken flight.

During the transition, the transition widget is animated to route B's hero's position, and then the widget is inserted into route B. When going back from B to A, route A's hero's widget is, by default, placed over where route B's hero's widget was, and then the animation goes the other way.

### Nested Navigators

If either or both routes contain nested [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)s, only [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html)es contained in the top-most routes (as defined by [Route.isCurrent](https://api.flutter.dev/flutter/widgets/Route/isCurrent.html)) of those nested [*Navigator*](https://api.flutter.dev/flutter/widgets/Navigator-class.html)s are considered for animation. Just like in the non-nested case the top-most routes containing these [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html)es in the nested [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)s have to be [PageRoute](https://api.flutter.dev/flutter/widgets/PageRoute-class.html)s.

### Constructors

[Hero](https://api.flutter.dev/flutter/widgets/Hero/Hero.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html) tag, [CreateRectTween](https://api.flutter.dev/flutter/widgets/CreateRectTween.html)? createRectTween, [HeroFlightShuttleBuilder](https://api.flutter.dev/flutter/widgets/HeroFlightShuttleBuilder.html)? flightShuttleBuilder, [HeroPlaceholderBuilder](https://api.flutter.dev/flutter/widgets/HeroPlaceholderBuilder.html)? placeholderBuilder, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) transitionOnUserGestures = false, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) child})

Create a hero.

const

### Properties

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

The widget subtree that will "fly" from one route to another during a [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) push or pop transition.

final

[createRectTween](https://api.flutter.dev/flutter/widgets/Hero/createRectTween.html) → [CreateRectTween](https://api.flutter.dev/flutter/widgets/CreateRectTween.html)?

Defines how the destination hero's bounds change as it flies from the starting route to the destination route.

final

[flightShuttleBuilder](https://api.flutter.dev/flutter/widgets/Hero/flightShuttleBuilder.html) → [HeroFlightShuttleBuilder](https://api.flutter.dev/flutter/widgets/HeroFlightShuttleBuilder.html)?

Optional override to supply a widget that's shown during the hero's flight.

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

[placeholderBuilder](https://api.flutter.dev/flutter/widgets/Hero/placeholderBuilder.html) → [HeroPlaceholderBuilder](https://api.flutter.dev/flutter/widgets/HeroPlaceholderBuilder.html)?

Placeholder widget left in place as the Hero's [child](https://api.flutter.dev/flutter/widgets/Hero/child.html) once the flight takes off.

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

[tag](https://api.flutter.dev/flutter/widgets/Hero/tag.html) → [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)

The identifier for this particular hero. If the tag of this hero matches the tag of a hero on a [PageRoute](https://api.flutter.dev/flutter/widgets/PageRoute-class.html) that we're navigating to or from, then a hero animation will be triggered.

final

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

Whether to perform the hero transition if the [PageRoute](https://api.flutter.dev/flutter/widgets/PageRoute-class.html) transition was triggered by a user gesture, such as a back swipe on iOS.

## NAVIGATOR

A widget that manages a set of child widgets with a stack discipline.

Many apps have a navigator near the top of their widget hierarchy in order to display their logical history using an [Overlay](https://api.flutter.dev/flutter/widgets/Overlay-class.html) with the most recently visited pages visually on top of the older pages. Using this pattern lets the navigator visually transition from one page to another by moving the widgets around in the overlay. Similarly, the navigator can be used to show a dialog by positioning the dialog widget above the current page.

### Using the Pages API

The [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) will convert its [Navigator.pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html) into a stack of [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)s if it is provided. A change in [Navigator.pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html) will trigger an update to the stack of [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)s. The [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) will update its routes to match the new configuration of its [Navigator.pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html). To use this API, one can create a [Page](https://api.flutter.dev/flutter/widgets/Page-class.html) subclass and defines a list of [Page](https://api.flutter.dev/flutter/widgets/Page-class.html)s for [Navigator.pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html). A [Navigator.onPopPage](https://api.flutter.dev/flutter/widgets/Navigator/onPopPage.html) callback is also required to properly clean up the input pages in case of a pop.

By Default, the [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) will use [DefaultTransitionDelegate](https://api.flutter.dev/flutter/widgets/DefaultTransitionDelegate-class.html) to decide how routes transition in or out of the screen. To customize it, define a [TransitionDelegate](https://api.flutter.dev/flutter/widgets/TransitionDelegate-class.html) subclass and provide it to the [Navigator.transitionDelegate](https://api.flutter.dev/flutter/widgets/Navigator/transitionDelegate.html).

For more information on using the pages API, see the [Router](https://api.flutter.dev/flutter/widgets/Router-class.html) widget.

### Using the Navigator API

Mobile apps typically reveal their contents via full-screen elements called "screens" or "pages". In Flutter these elements are called routes and they're managed by a [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) widget. The navigator manages a stack of [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) objects and provides two ways for managing the stack, the declarative API [Navigator.pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html) or imperative API [Navigator.push](https://api.flutter.dev/flutter/widgets/Navigator/push.html) and [Navigator.pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html).

When your user interface fits this paradigm of a stack, where the user should be able to navigate back to an earlier element in the stack, the use of routes and the Navigator is appropriate. On certain platforms, such as Android, the system UI will provide a back button (outside the bounds of your application) that will allow the user to navigate back to earlier routes in your application's stack. On platforms that don't have this build-in navigation mechanism, the use of an [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html) (typically used in the [Scaffold.appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) property) can automatically add a back button for user navigation.

### Displaying a full-screen route

Although you can create a navigator directly, it's most common to use the navigator created by the Router which itself is created and configured by a [WidgetsApp](https://api.flutter.dev/flutter/widgets/WidgetsApp-class.html) or a [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html) widget. You can refer to that navigator with [Navigator.of](https://api.flutter.dev/flutter/widgets/Navigator/of.html).

A [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html) is the simplest way to set things up. The [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html)'s home becomes the route at the bottom of the [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)'s stack. It is what you see when the app is launched.

void main() {

runApp(const MaterialApp(home: MyAppHome()));

}

To push a new route on the stack you can create an instance of [MaterialPageRoute](https://api.flutter.dev/flutter/material/MaterialPageRoute-class.html) with a builder function that creates whatever you want to appear on the screen. For example:

Navigator.push(context, MaterialPageRoute<void>(

builder: (BuildContext context) {

return Scaffold(

appBar: AppBar(title: const Text('My Page')),

body: Center(

child: TextButton(

child: const Text('POP'),

onPressed: () {

Navigator.pop(context);

},

),

),

);

},

));

The route defines its widget with a builder function instead of a child widget because it will be built and rebuilt in different contexts depending on when it's pushed and popped.

As you can see, the new route can be popped, revealing the app's home page, with the Navigator's pop method:

Navigator.pop(context);

It usually isn't necessary to provide a widget that pops the Navigator in a route with a [Scaffold](https://api.flutter.dev/flutter/material/Scaffold-class.html) because the Scaffold automatically adds a 'back' button to its AppBar. Pressing the back button causes [Navigator.pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html) to be called. On Android, pressing the system back button does the same thing.

### Using named navigator routes

Mobile apps often manage a large number of routes and it's often easiest to refer to them by name. Route names, by convention, use a path-like structure (for example, '/a/b/c'). The app's home page route is named '/' by default.

The [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html) can be created with a [Map<String, WidgetBuilder>](https://api.flutter.dev/flutter/dart-core/Map-class.html) which maps from a route's name to a builder function that will create it. The [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html) uses this map to create a value for its navigator's [onGenerateRoute](https://api.flutter.dev/flutter/widgets/Navigator/onGenerateRoute.html) callback.

void main() {

runApp(MaterialApp(

home: const MyAppHome(), // becomes the route named '/'

routes: <String, WidgetBuilder> {

'/a': (BuildContext context) => const MyPage(title: Text('page A')),

'/b': (BuildContext context) => const MyPage(title: Text('page B')),

'/c': (BuildContext context) => const MyPage(title: Text('page C')),

},

));

}

To show a route by name:

Navigator.pushNamed(context, '/b');

### Routes can return a value

When a route is pushed to ask the user for a value, the value can be returned via the [pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html) method's result parameter.

Methods that push a route return a [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html). The Future resolves when the route is popped and the [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)'s value is the [pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html) method's result parameter.

For example if we wanted to ask the user to press 'OK' to confirm an operation we could await the result of [Navigator.push](https://api.flutter.dev/flutter/widgets/Navigator/push.html):

bool? value = await Navigator.push(context, MaterialPageRoute<bool>(

builder: (BuildContext context) {

return Center(

child: GestureDetector(

child: const Text('OK'),

onTap: () { Navigator.pop(context, true); }

),

);

}

));

If the user presses 'OK' then value will be true. If the user backs out of the route, for example by pressing the Scaffold's back button, the value will be null.

When a route is used to return a value, the route's type parameter must match the type of [pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html)'s result. That's why we've used MaterialPageRoute<bool> instead of MaterialPageRoute<void> or just MaterialPageRoute. (If you prefer to not specify the types, though, that's fine too.)

### Popup routes

Routes don't have to obscure the entire screen. [PopupRoute](https://api.flutter.dev/flutter/widgets/PopupRoute-class.html)s cover the screen with a [ModalRoute.barrierColor](https://api.flutter.dev/flutter/widgets/ModalRoute/barrierColor.html) that can be only partially opaque to allow the current screen to show through. Popup routes are "modal" because they block input to the widgets below.

There are functions which create and show popup routes. For example: [showDialog](https://api.flutter.dev/flutter/material/showDialog.html), [showMenu](https://api.flutter.dev/flutter/material/showMenu.html), and [showModalBottomSheet](https://api.flutter.dev/flutter/material/showModalBottomSheet.html). These functions return their pushed route's Future as described above. Callers can await the returned value to take an action when the route is popped, or to discover the route's value.

There are also widgets which create popup routes, like [PopupMenuButton](https://api.flutter.dev/flutter/material/PopupMenuButton-class.html) and [DropdownButton](https://api.flutter.dev/flutter/material/DropdownButton-class.html). These widgets create internal subclasses of PopupRoute and use the Navigator's push and pop methods to show and dismiss them.

### Custom routes

You can create your own subclass of one of the widget library route classes like [PopupRoute](https://api.flutter.dev/flutter/widgets/PopupRoute-class.html), [ModalRoute](https://api.flutter.dev/flutter/widgets/ModalRoute-class.html), or [PageRoute](https://api.flutter.dev/flutter/widgets/PageRoute-class.html), to control the animated transition employed to show the route, the color and behavior of the route's modal barrier, and other aspects of the route.

The [PageRouteBuilder](https://api.flutter.dev/flutter/widgets/PageRouteBuilder-class.html) class makes it possible to define a custom route in terms of callbacks. Here's an example that rotates and fades its child when the route appears or disappears. This route does not obscure the entire screen because it specifies opaque: false, just as a popup route does.

Navigator.push(context, PageRouteBuilder<void>(

opaque: false,

pageBuilder: (BuildContext context, \_, \_\_) {

return const Center(child: Text('My PageRoute'));

},

transitionsBuilder: (\_\_\_, Animation<double> animation, \_\_\_\_, Widget child) {

return FadeTransition(

opacity: animation,

child: RotationTransition(

turns: Tween<double>(begin: 0.5, end: 1.0).animate(animation),

child: child,

),

);

}

));

The page route is built in two parts, the "page" and the "transitions". The page becomes a descendant of the child passed to the transitionsBuilder function. Typically the page is only built once, because it doesn't depend on its animation parameters (elided with \_ and \_\_ in this example). The transition is built on every frame for its duration.

(In this example, void is used as the return type for the route, because it does not return a value.)

### Nesting Navigators

An app can use more than one [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html). Nesting one [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) below another [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) can be used to create an "inner journey" such as tabbed navigation, user registration, store checkout, or other independent journeys that represent a subsection of your overall application.

#### Example

It is standard practice for iOS apps to use tabbed navigation where each tab maintains its own navigation history. Therefore, each tab has its own [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html), creating a kind of "parallel navigation."

In addition to the parallel navigation of the tabs, it is still possible to launch full-screen pages that completely cover the tabs. For example: an on-boarding flow, or an alert dialog. Therefore, there must exist a "root" [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) that sits above the tab navigation. As a result, each of the tab's [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)s are actually nested [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)s sitting below a single root [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html).

In practice, the nested [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)s for tabbed navigation sit in the [WidgetsApp](https://api.flutter.dev/flutter/widgets/WidgetsApp-class.html) and [CupertinoTabView](https://api.flutter.dev/flutter/cupertino/CupertinoTabView-class.html) widgets and do not need to be explicitly created or managed.

[Navigator.of](https://api.flutter.dev/flutter/widgets/Navigator/of.html) operates on the nearest ancestor [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) from the given [BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html). Be sure to provide a [BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) below the intended [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html), especially in large build methods where nested [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html)s are created. The [Builder](https://api.flutter.dev/flutter/widgets/Builder-class.html) widget can be used to access a [BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) at a desired location in the widget subtree.

### Finding the enclosing route

In the common case of a modal route, the enclosing route can be obtained from inside a build method using [ModalRoute.of](https://api.flutter.dev/flutter/widgets/ModalRoute/of.html). To determine if the enclosing route is the active route (e.g. so that controls can be dimmed when the route is not active), the [Route.isCurrent](https://api.flutter.dev/flutter/widgets/Route/isCurrent.html) property can be checked on the returned route.

### State Restoration

If provided with a [restorationScopeId](https://api.flutter.dev/flutter/widgets/Navigator/restorationScopeId.html) and when surrounded by a valid [RestorationScope](https://api.flutter.dev/flutter/widgets/RestorationScope-class.html) the [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) will restore its state by recreating the current history stack of [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)s during state restoration and by restoring the internal state of those [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)s. However, not all [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)s on the stack can be restored:

* [Page](https://api.flutter.dev/flutter/widgets/Page-class.html)-based routes restore their state if [Page.restorationId](https://api.flutter.dev/flutter/widgets/Page/restorationId.html) is provided.
* [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)s added with the classic imperative API ([push](https://api.flutter.dev/flutter/widgets/Navigator/push.html), [pushNamed](https://api.flutter.dev/flutter/widgets/Navigator/pushNamed.html), and friends) can never restore their state.
* A [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) added with the restorable imperative API ([restorablePush](https://api.flutter.dev/flutter/widgets/Navigator/restorablePush.html), [restorablePushNamed](https://api.flutter.dev/flutter/widgets/Navigator/restorablePushNamed.html), and all other imperative methods with "restorable" in their name) restores its state if all routes below it up to and including the first [Page](https://api.flutter.dev/flutter/widgets/Page-class.html)-based route below it are restored. If there is no [Page](https://api.flutter.dev/flutter/widgets/Page-class.html)-based route below it, it only restores its state if all routes below it restore theirs.

If a [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) is deemed restorable, the [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) will set its [Route.restorationScopeId](https://api.flutter.dev/flutter/widgets/Route/restorationScopeId.html) to a non-null value. Routes can use that ID to store and restore their own state. As an example, the [ModalRoute](https://api.flutter.dev/flutter/widgets/ModalRoute-class.html) will use this ID to create a [RestorationScope](https://api.flutter.dev/flutter/widgets/RestorationScope-class.html) for its content widgets.

### Constructors

[Navigator](https://api.flutter.dev/flutter/widgets/Navigator/Navigator.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Page](https://api.flutter.dev/flutter/widgets/Page-class.html)> pages = const <Page<dynamic>>[], [PopPageCallback](https://api.flutter.dev/flutter/widgets/PopPageCallback.html)? onPopPage, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? initialRoute, [RouteListFactory](https://api.flutter.dev/flutter/widgets/RouteListFactory.html) onGenerateInitialRoutes = Navigator.defaultGenerateInitialRoutes, [RouteFactory](https://api.flutter.dev/flutter/widgets/RouteFactory.html)? onGenerateRoute, [RouteFactory](https://api.flutter.dev/flutter/widgets/RouteFactory.html)? onUnknownRoute, [TransitionDelegate](https://api.flutter.dev/flutter/widgets/TransitionDelegate-class.html) transitionDelegate = const DefaultTransitionDelegate<dynamic>(), [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) reportsRouteUpdateToEngine = false, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.hardEdge, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[NavigatorObserver](https://api.flutter.dev/flutter/widgets/NavigatorObserver-class.html)> observers = const <NavigatorObserver>[], [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) requestFocus = true, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationScopeId, [TraversalEdgeBehavior](https://api.flutter.dev/flutter/widgets/TraversalEdgeBehavior.html) routeTraversalEdgeBehavior = kDefaultRouteTraversalEdgeBehavior})

Creates a widget that maintains a stack-based history of child widgets.

const

### Properties

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

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

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

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

The name of the first route to show.

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

[observers](https://api.flutter.dev/flutter/widgets/Navigator/observers.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[NavigatorObserver](https://api.flutter.dev/flutter/widgets/NavigatorObserver-class.html)>

A list of observers for this navigator.

final

[onGenerateInitialRoutes](https://api.flutter.dev/flutter/widgets/Navigator/onGenerateInitialRoutes.html) → [RouteListFactory](https://api.flutter.dev/flutter/widgets/RouteListFactory.html)

Called when the widget is created to generate the initial list of [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) objects if [initialRoute](https://api.flutter.dev/flutter/widgets/Navigator/initialRoute.html) is not null.

final

[onGenerateRoute](https://api.flutter.dev/flutter/widgets/Navigator/onGenerateRoute.html) → [RouteFactory](https://api.flutter.dev/flutter/widgets/RouteFactory.html)?

Called to generate a route for a given [RouteSettings](https://api.flutter.dev/flutter/widgets/RouteSettings-class.html).

final

[onPopPage](https://api.flutter.dev/flutter/widgets/Navigator/onPopPage.html) → [PopPageCallback](https://api.flutter.dev/flutter/widgets/PopPageCallback.html)?

Called when [pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html) is invoked but the current [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) corresponds to a [Page](https://api.flutter.dev/flutter/widgets/Page-class.html) found in the [pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html) list.

final

[onUnknownRoute](https://api.flutter.dev/flutter/widgets/Navigator/onUnknownRoute.html) → [RouteFactory](https://api.flutter.dev/flutter/widgets/RouteFactory.html)?

Called when [onGenerateRoute](https://api.flutter.dev/flutter/widgets/Navigator/onGenerateRoute.html) fails to generate a route.

final

[pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Page](https://api.flutter.dev/flutter/widgets/Page-class.html)>

The list of pages with which to populate the history.

final

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

Whether this navigator should report route update message back to the engine when the top-most route changes.

final

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

Whether or not the navigator and it's new topmost route should request focus when the new route is pushed onto the navigator.

final

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

Restoration ID to save and restore the state of the navigator, including its history.

final

[routeTraversalEdgeBehavior](https://api.flutter.dev/flutter/widgets/Navigator/routeTraversalEdgeBehavior.html) → [TraversalEdgeBehavior](https://api.flutter.dev/flutter/widgets/TraversalEdgeBehavior.html)

Controls the transfer of focus beyond the first and the last items of a focus scope that defines focus traversal of widgets within a route.

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

[transitionDelegate](https://api.flutter.dev/flutter/widgets/Navigator/transitionDelegate.html) → [TransitionDelegate](https://api.flutter.dev/flutter/widgets/TransitionDelegate-class.html)

The delegate used for deciding how routes transition in or off the screen during the [pages](https://api.flutter.dev/flutter/widgets/Navigator/pages.html) updates.

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/Navigator/createState.html)() → [NavigatorState](https://api.flutter.dev/flutter/widgets/NavigatorState-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

### Static Methods

[canPop](https://api.flutter.dev/flutter/widgets/Navigator/canPop.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context) → [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)

Whether the navigator that most tightly encloses the given context can be popped.

[defaultGenerateInitialRoutes](https://api.flutter.dev/flutter/widgets/Navigator/defaultGenerateInitialRoutes.html)([NavigatorState](https://api.flutter.dev/flutter/widgets/NavigatorState-class.html) navigator, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) initialRouteName) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Route](https://api.flutter.dev/flutter/widgets/Route-class.html)>

Turn a route name into a set of Route objects.

[maybeOf](https://api.flutter.dev/flutter/widgets/Navigator/maybeOf.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) rootNavigator = false}) → [NavigatorState](https://api.flutter.dev/flutter/widgets/NavigatorState-class.html)?

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

[maybePop](https://api.flutter.dev/flutter/widgets/Navigator/maybePop.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [T? result]) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>

Consults the current route's [Route.popDisposition](https://api.flutter.dev/flutter/widgets/Route/popDisposition.html) getter or [Route.willPop](https://api.flutter.dev/flutter/widgets/Route/willPop.html) method, and acts accordingly, potentially popping the route as a result; returns whether the pop request should be considered handled.

[of](https://api.flutter.dev/flutter/widgets/Navigator/of.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) rootNavigator = false}) → [NavigatorState](https://api.flutter.dev/flutter/widgets/NavigatorState-class.html)

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

[pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [T? result]) → void

Pop the top-most route off the navigator that most tightly encloses the given context.

[popAndPushNamed](https://api.flutter.dev/flutter/widgets/Navigator/popAndPushNamed.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?, TO extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) routeName, {TO? result, [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?>

Pop the current route off the navigator that most tightly encloses the given context and push a named route in its place.

[popUntil](https://api.flutter.dev/flutter/widgets/Navigator/popUntil.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [RoutePredicate](https://api.flutter.dev/flutter/widgets/RoutePredicate.html) predicate) → void

Calls [pop](https://api.flutter.dev/flutter/widgets/Navigator/pop.html) repeatedly on the navigator that most tightly encloses the given context until the predicate returns true.

[push](https://api.flutter.dev/flutter/widgets/Navigator/push.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)<T> route) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?>

Push the given route onto the navigator that most tightly encloses the given context.

[pushAndRemoveUntil](https://api.flutter.dev/flutter/widgets/Navigator/pushAndRemoveUntil.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)<T> newRoute, [RoutePredicate](https://api.flutter.dev/flutter/widgets/RoutePredicate.html) predicate) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?>

Push the given route onto the navigator that most tightly encloses the given context, and then remove all the previous routes until the predicate returns true.

[pushNamed](https://api.flutter.dev/flutter/widgets/Navigator/pushNamed.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) routeName, {[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?>

Push a named route onto the navigator that most tightly encloses the given context.

[pushNamedAndRemoveUntil](https://api.flutter.dev/flutter/widgets/Navigator/pushNamedAndRemoveUntil.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) newRouteName, [RoutePredicate](https://api.flutter.dev/flutter/widgets/RoutePredicate.html) predicate, {[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?>

Push the route with the given name onto the navigator that most tightly encloses the given context, and then remove all the previous routes until the predicate returns true.

[pushReplacement](https://api.flutter.dev/flutter/widgets/Navigator/pushReplacement.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?, TO extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)<T> newRoute, {TO? result}) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?>

Replace the current route of the navigator that most tightly encloses the given context by pushing the given route and then disposing the previous route once the new route has finished animating in.

[pushReplacementNamed](https://api.flutter.dev/flutter/widgets/Navigator/pushReplacementNamed.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?, TO extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) routeName, {TO? result, [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [Future](https://api.flutter.dev/flutter/dart-async/Future-class.html)<T?>

Replace the current route of the navigator that most tightly encloses the given context by pushing the route named routeName and then disposing the previous route once the new route has finished animating in.

[removeRoute](https://api.flutter.dev/flutter/widgets/Navigator/removeRoute.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) route) → void

Immediately remove route from the navigator that most tightly encloses the given context, and [Route.dispose](https://api.flutter.dev/flutter/widgets/Route/dispose.html) it.

[removeRouteBelow](https://api.flutter.dev/flutter/widgets/Navigator/removeRouteBelow.html)([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) anchorRoute) → void

Immediately remove a route from the navigator that most tightly encloses the given context, and [Route.dispose](https://api.flutter.dev/flutter/widgets/Route/dispose.html) it. The route to be removed is the one below the given anchorRoute.

[replace](https://api.flutter.dev/flutter/widgets/Navigator/replace.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {required [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) oldRoute, required [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)<T> newRoute}) → void

Replaces a route on the navigator that most tightly encloses the given context with a new route.

[replaceRouteBelow](https://api.flutter.dev/flutter/widgets/Navigator/replaceRouteBelow.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {required [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) anchorRoute, required [Route](https://api.flutter.dev/flutter/widgets/Route-class.html)<T> newRoute}) → void

Replaces a route on the navigator that most tightly encloses the given context with a new route. The route to be replaced is the one below the given anchorRoute.

[restorablePopAndPushNamed](https://api.flutter.dev/flutter/widgets/Navigator/restorablePopAndPushNamed.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?, TO extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) routeName, {TO? result, [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Pop the current route off the navigator that most tightly encloses the given context and push a named route in its place.

[restorablePush](https://api.flutter.dev/flutter/widgets/Navigator/restorablePush.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [RestorableRouteBuilder](https://api.flutter.dev/flutter/widgets/RestorableRouteBuilder.html)<T> routeBuilder, {[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Push a new route onto the navigator that most tightly encloses the given context.

[restorablePushAndRemoveUntil](https://api.flutter.dev/flutter/widgets/Navigator/restorablePushAndRemoveUntil.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [RestorableRouteBuilder](https://api.flutter.dev/flutter/widgets/RestorableRouteBuilder.html)<T> newRouteBuilder, [RoutePredicate](https://api.flutter.dev/flutter/widgets/RoutePredicate.html) predicate, {[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Push a new route onto the navigator that most tightly encloses the given context, and then remove all the previous routes until the predicate returns true.

[restorablePushNamed](https://api.flutter.dev/flutter/widgets/Navigator/restorablePushNamed.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) routeName, {[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Push a named route onto the navigator that most tightly encloses the given context.

[restorablePushNamedAndRemoveUntil](https://api.flutter.dev/flutter/widgets/Navigator/restorablePushNamedAndRemoveUntil.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) newRouteName, [RoutePredicate](https://api.flutter.dev/flutter/widgets/RoutePredicate.html) predicate, {[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Push the route with the given name onto the navigator that most tightly encloses the given context, and then remove all the previous routes until the predicate returns true.

[restorablePushReplacement](https://api.flutter.dev/flutter/widgets/Navigator/restorablePushReplacement.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?, TO extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [RestorableRouteBuilder](https://api.flutter.dev/flutter/widgets/RestorableRouteBuilder.html)<T> routeBuilder, {TO? result, [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Replace the current route of the navigator that most tightly encloses the given context by pushing a new route and then disposing the previous route once the new route has finished animating in.

[restorablePushReplacementNamed](https://api.flutter.dev/flutter/widgets/Navigator/restorablePushReplacementNamed.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?, TO extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, [String](https://api.flutter.dev/flutter/dart-core/String-class.html) routeName, {TO? result, [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Replace the current route of the navigator that most tightly encloses the given context by pushing the route named routeName and then disposing the previous route once the new route has finished animating in.

[restorableReplace](https://api.flutter.dev/flutter/widgets/Navigator/restorableReplace.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {required [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) oldRoute, required [RestorableRouteBuilder](https://api.flutter.dev/flutter/widgets/RestorableRouteBuilder.html)<T> newRouteBuilder, [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Replaces a route on the navigator that most tightly encloses the given context with a new route.

[restorableReplaceRouteBelow](https://api.flutter.dev/flutter/widgets/Navigator/restorableReplaceRouteBelow.html)<T extends [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)?>([BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) context, {required [Route](https://api.flutter.dev/flutter/widgets/Route-class.html) anchorRoute, required [RestorableRouteBuilder](https://api.flutter.dev/flutter/widgets/RestorableRouteBuilder.html)<T> newRouteBuilder, [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? arguments}) → [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

Replaces a route on the navigator that most tightly encloses the given context with a new route. The route to be replaced is the one below the given anchorRoute.

### Constants

[defaultRouteName](https://api.flutter.dev/flutter/widgets/Navigator/defaultRouteName-constant.html) → const [String](https://api.flutter.dev/flutter/dart-core/String-class.html)

The name for the default route of the application.

'/'