FLUTTER RESUMEN DOCUMENTACIÓN

# Basic widgets

## APP BAR

A Material Design app bar.

An app bar consists of a toolbar and potentially other widgets, such as a [TabBar](https://api.flutter.dev/flutter/material/TabBar-class.html) and a [FlexibleSpaceBar](https://api.flutter.dev/flutter/material/FlexibleSpaceBar-class.html). App bars typically expose one or more common [actions](https://api.flutter.dev/flutter/material/AppBar/actions.html) with [IconButton](https://api.flutter.dev/flutter/material/IconButton-class.html)s which are optionally followed by a [PopupMenuButton](https://api.flutter.dev/flutter/material/PopupMenuButton-class.html) for less common operations (sometimes called the "overflow menu").

App bars are typically used in the [Scaffold.appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) property, which places the app bar as a fixed-height widget at the top of the screen. For a scrollable app bar, see [SliverAppBar](https://api.flutter.dev/flutter/material/SliverAppBar-class.html), which embeds an [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html) in a sliver for use in a [CustomScrollView](https://api.flutter.dev/flutter/widgets/CustomScrollView-class.html).

The AppBar displays the toolbar widgets, [leading](https://api.flutter.dev/flutter/material/AppBar/leading.html), [title](https://api.flutter.dev/flutter/material/AppBar/title.html), and [actions](https://api.flutter.dev/flutter/material/AppBar/actions.html), above the [bottom](https://api.flutter.dev/flutter/material/AppBar/bottom.html) (if any). The [bottom](https://api.flutter.dev/flutter/material/AppBar/bottom.html) is usually used for a [TabBar](https://api.flutter.dev/flutter/material/TabBar-class.html). If a [flexibleSpace](https://api.flutter.dev/flutter/material/AppBar/flexibleSpace.html) widget is specified then it is stacked behind the toolbar and the bottom widget. The following diagram shows where each of these slots appears in the toolbar when the writing language is left-to-right (e.g. English):

The [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html) insets its content based on the ambient [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html)'s padding, to avoid system UI intrusions. It's taken care of by [Scaffold](https://api.flutter.dev/flutter/material/Scaffold-class.html) when used in the [Scaffold.appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) property. When animating an [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html), unexpected [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html) changes (as is common in [Hero](https://api.flutter.dev/flutter/widgets/Hero-class.html) animations) may cause the content to suddenly jump. Wrap the [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html) in a [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html) widget, and adjust its padding such that the animation is smooth.

The leading widget is in the top left, the actions are in the top right,
the title is between them. The bottom is, naturally, at the bottom, and the
flexibleSpace is behind all of them.

If the [leading](https://api.flutter.dev/flutter/material/AppBar/leading.html) widget is omitted, but the [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html) is in a [Scaffold](https://api.flutter.dev/flutter/material/Scaffold-class.html) with a [Drawer](https://api.flutter.dev/flutter/material/Drawer-class.html), then a button will be inserted to open the drawer. Otherwise, if the nearest [Navigator](https://api.flutter.dev/flutter/widgets/Navigator-class.html) has any previous routes, a [BackButton](https://api.flutter.dev/flutter/material/BackButton-class.html) is inserted instead. This behavior can be turned off by setting the [automaticallyImplyLeading](https://api.flutter.dev/flutter/material/AppBar/automaticallyImplyLeading.html) to false. In that case a null leading widget will result in the middle/title widget stretching to start.

### Why don't my TextButton actions appear?

If the app bar's [actions](https://api.flutter.dev/flutter/material/AppBar/actions.html) contains [TextButton](https://api.flutter.dev/flutter/material/TextButton-class.html)s, they will not be visible if their foreground (text) color is the same as the app bar's background color.

In Material v2 (i.e., when [ThemeData.useMaterial3](https://api.flutter.dev/flutter/material/ThemeData/useMaterial3.html) is false), the default app bar [backgroundColor](https://api.flutter.dev/flutter/material/AppBar/backgroundColor.html) is the overall theme's [ColorScheme.primary](https://api.flutter.dev/flutter/material/ColorScheme/primary.html) if the overall theme's brightness is [Brightness.light](https://api.flutter.dev/flutter/dart-ui/Brightness.html). Unfortunately this is the same as the default [ButtonStyle.foregroundColor](https://api.flutter.dev/flutter/material/ButtonStyle/foregroundColor.html) for [TextButton](https://api.flutter.dev/flutter/material/TextButton-class.html) for light themes. In this case a preferable text button foreground color is [ColorScheme.onPrimary](https://api.flutter.dev/flutter/material/ColorScheme/onPrimary.html), a color that contrasts nicely with [ColorScheme.primary](https://api.flutter.dev/flutter/material/ColorScheme/primary.html). To remedy the problem, override [TextButton.style](https://api.flutter.dev/flutter/material/ButtonStyleButton/style.html):

### Constructors

[AppBar](https://api.flutter.dev/flutter/material/AppBar/AppBar.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? leading, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) automaticallyImplyLeading = true, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? title, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>? actions, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? flexibleSpace, [PreferredSizeWidget](https://api.flutter.dev/flutter/widgets/PreferredSizeWidget-class.html)? bottom, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? elevation, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? scrolledUnderElevation, [ScrollNotificationPredicate](https://api.flutter.dev/flutter/widgets/ScrollNotificationPredicate.html) notificationPredicate = defaultScrollNotificationPredicate, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? shadowColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? surfaceTintColor, [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)? shape, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? foregroundColor, [IconThemeData](https://api.flutter.dev/flutter/widgets/IconThemeData-class.html)? iconTheme, [IconThemeData](https://api.flutter.dev/flutter/widgets/IconThemeData-class.html)? actionsIconTheme, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) primary = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? centerTitle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeHeaderSemantics = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? titleSpacing, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) toolbarOpacity = 1.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) bottomOpacity = 1.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? toolbarHeight, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? leadingWidth, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? toolbarTextStyle, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? titleTextStyle, [SystemUiOverlayStyle](https://api.flutter.dev/flutter/services/SystemUiOverlayStyle-class.html)? systemOverlayStyle, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) forceMaterialTransparency = false, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)? clipBehavior})

Creates a Material Design app bar.

### Properties

[actions](https://api.flutter.dev/flutter/material/AppBar/actions.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>?

A list of Widgets to display in a row after the [title](https://api.flutter.dev/flutter/material/AppBar/title.html) widget.

final

[actionsIconTheme](https://api.flutter.dev/flutter/material/AppBar/actionsIconTheme.html) → [IconThemeData](https://api.flutter.dev/flutter/widgets/IconThemeData-class.html)?

The color, opacity, and size to use for the icons that appear in the app bar's [actions](https://api.flutter.dev/flutter/material/AppBar/actions.html).

final

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

Controls whether we should try to imply the leading widget if null.

final

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

The fill color to use for an app bar's [Material](https://api.flutter.dev/flutter/material/Material-class.html).

final

[bottom](https://api.flutter.dev/flutter/material/AppBar/bottom.html) → [PreferredSizeWidget](https://api.flutter.dev/flutter/widgets/PreferredSizeWidget-class.html)?

This widget appears across the bottom of the app bar.

final

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

How opaque the bottom part of the app bar is.

final

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

Whether the title should be centered.

final

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

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

final

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

The z-coordinate at which to place this app bar relative to its parent.

final

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

Whether the title should be wrapped with header [Semantics](https://api.flutter.dev/flutter/widgets/Semantics-class.html).

final

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

This widget is stacked behind the toolbar and the tab bar. Its height will be the same as the app bar's overall height.

final

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

Forces the AppBar's Material widget type to be [MaterialType.transparency](https://api.flutter.dev/flutter/material/MaterialType.html) (instead of Material's default type).

final

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

The default color for [Text](https://api.flutter.dev/flutter/widgets/Text-class.html) and [Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html)s within the app bar.

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

[iconTheme](https://api.flutter.dev/flutter/material/AppBar/iconTheme.html) → [IconThemeData](https://api.flutter.dev/flutter/widgets/IconThemeData-class.html)?

The color, opacity, and size to use for toolbar icons.

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

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

A widget to display before the toolbar's [title](https://api.flutter.dev/flutter/material/AppBar/title.html).

final

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

Defines the width of [leading](https://api.flutter.dev/flutter/material/AppBar/leading.html) widget.

final

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

A check that specifies which child's [ScrollNotification](https://api.flutter.dev/flutter/widgets/ScrollNotification-class.html)s should be listened to.

final

[preferredSize](https://api.flutter.dev/flutter/material/AppBar/preferredSize.html) → [Size](https://api.flutter.dev/flutter/dart-ui/Size-class.html)

A size whose height is the sum of [toolbarHeight](https://api.flutter.dev/flutter/material/AppBar/toolbarHeight.html) and the [bottom](https://api.flutter.dev/flutter/material/AppBar/bottom.html) widget's preferred height.

final

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

Whether this app bar is being displayed at the top of the screen.

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

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

The elevation that will be used if this app bar has something scrolled underneath it.

final

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

The color of the shadow below the app bar.

final

[shape](https://api.flutter.dev/flutter/material/AppBar/shape.html) → [ShapeBorder](https://api.flutter.dev/flutter/painting/ShapeBorder-class.html)?

The shape of the app bar's [Material](https://api.flutter.dev/flutter/material/Material-class.html) as well as its shadow.

final

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

The color of the surface tint overlay applied to the app bar's background color to indicate elevation.

final

[systemOverlayStyle](https://api.flutter.dev/flutter/material/AppBar/systemOverlayStyle.html) → [SystemUiOverlayStyle](https://api.flutter.dev/flutter/services/SystemUiOverlayStyle-class.html)?

Specifies the style to use for the system overlays (e.g. the status bar on Android or iOS, the system navigation bar on Android).

final

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

The primary widget displayed in the app bar.

final

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

The spacing around [title](https://api.flutter.dev/flutter/material/AppBar/title.html) content on the horizontal axis. This spacing is applied even if there is no [leading](https://api.flutter.dev/flutter/material/AppBar/leading.html) content or [actions](https://api.flutter.dev/flutter/material/AppBar/actions.html). If you want [title](https://api.flutter.dev/flutter/material/AppBar/title.html) to take all the space available, set this value to 0.0.

final

[titleTextStyle](https://api.flutter.dev/flutter/material/AppBar/titleTextStyle.html) → [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)?

The default text style for the AppBar's [title](https://api.flutter.dev/flutter/material/AppBar/title.html) widget.

final

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

Defines the height of the toolbar component of an [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html).

final

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

How opaque the toolbar part of the app bar is.

final

[toolbarTextStyle](https://api.flutter.dev/flutter/material/AppBar/toolbarTextStyle.html) → [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)?

The default text style for the AppBar's [leading](https://api.flutter.dev/flutter/material/AppBar/leading.html), and [actions](https://api.flutter.dev/flutter/material/AppBar/actions.html) widgets, but not its [title](https://api.flutter.dev/flutter/material/AppBar/title.html).

## COLUMN

A widget that displays its children in a vertical array.

To cause a child to expand to fill the available vertical space, wrap the child in an [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) widget.

The [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) widget does not scroll (and in general it is considered an error to have more children in a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) than will fit in the available room). If you have a line of widgets and want them to be able to scroll if there is insufficient room, consider using a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html).

For a horizontal variant, see [Row](https://api.flutter.dev/flutter/widgets/Row-class.html).

If you only have one child, then consider using [Align](https://api.flutter.dev/flutter/widgets/Align-class.html) or [Center](https://api.flutter.dev/flutter/widgets/Center-class.html) to position the child.

### When the incoming vertical constraints are unbounded

When a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) has one or more [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) or [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) children, and is placed in another [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), or in a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html), or in some other context that does not provide a maximum height constraint for the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), you will get an exception at runtime saying that there are children with non-zero flex but the vertical constraints are unbounded.

The problem, as described in the details that accompany that exception, is that using [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) or [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) means that the remaining space after laying out all the other children must be shared equally, but if the incoming vertical constraints are unbounded, there is infinite remaining space.

The key to solving this problem is usually to determine why the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) is receiving unbounded vertical constraints.

One common reason for this to happen is that the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) has been placed in another [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) (without using [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) or [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) around the inner nested [Column](https://api.flutter.dev/flutter/widgets/Column-class.html)). When a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) lays out its non-flex children (those that have neither [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) or [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) around them), it gives them unbounded constraints so that they can determine their own dimensions (passing unbounded constraints usually signals to the child that it should shrink-wrap its contents). The solution in this case is typically to just wrap the inner column in an [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) to indicate that it should take the remaining space of the outer column, rather than being allowed to take any amount of room it desires.

Another reason for this message to be displayed is nesting a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) inside a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) or other vertical scrollable. In that scenario, there really is infinite vertical space (the whole point of a vertical scrolling list is to allow infinite space vertically). In such scenarios, it is usually worth examining why the inner [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) should have an [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) or [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) child: what size should the inner children really be? The solution in this case is typically to remove the [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) or [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) widgets from around the inner children.

### The yellow and black striped banner

When the contents of a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) exceed the amount of space available, the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) overflows, and the contents are clipped. In debug mode, a yellow and black striped bar is rendered at the overflowing edge to indicate the problem, and a message is printed below the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) saying how much overflow was detected.

The usual solution is to use a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html) rather than a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html), to enable the contents to scroll when vertical space is limited.

### Layout algorithm

This section describes how a [*Column*](https://api.flutter.dev/flutter/widgets/Column-class.html) is rendered by the framework. See [*BoxConstraints*](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html) for an introduction to box layout models.

Layout for a [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) proceeds in six steps:

1. Layout each child with a null or zero flex factor (e.g., those that are not [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html)) with unbounded vertical constraints and the incoming horizontal constraints. If the [crossAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/crossAxisAlignment.html) is [CrossAxisAlignment.stretch](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html), instead use tight horizontal constraints that match the incoming max width.
2. Divide the remaining vertical space among the children with non-zero flex factors (e.g., those that are [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html)) according to their flex factor. For example, a child with a flex factor of 2.0 will receive twice the amount of vertical space as a child with a flex factor of 1.0.
3. Layout each of the remaining children with the same horizontal constraints as in step 1, but instead of using unbounded vertical constraints, use vertical constraints based on the amount of space allocated in step 2. Children with [Flexible.fit](https://api.flutter.dev/flutter/widgets/Flexible/fit.html) properties that are [FlexFit.tight](https://api.flutter.dev/flutter/rendering/FlexFit.html) are given tight constraints (i.e., forced to fill the allocated space), and children with [Flexible.fit](https://api.flutter.dev/flutter/widgets/Flexible/fit.html) properties that are [FlexFit.loose](https://api.flutter.dev/flutter/rendering/FlexFit.html) are given loose constraints (i.e., not forced to fill the allocated space).
4. The width of the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) is the maximum width of the children (which will always satisfy the incoming horizontal constraints).
5. The height of the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) is determined by the [mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) property. If the [mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) property is [MainAxisSize.max](https://api.flutter.dev/flutter/rendering/MainAxisSize.html), then the height of the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) is the max height of the incoming constraints. If the [mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) property is [MainAxisSize.min](https://api.flutter.dev/flutter/rendering/MainAxisSize.html), then the height of the [Column](https://api.flutter.dev/flutter/widgets/Column-class.html) is the sum of heights of the children (subject to the incoming constraints).
6. Determine the position for each child according to the [mainAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/mainAxisAlignment.html) and the [crossAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/crossAxisAlignment.html). For example, if the [mainAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/mainAxisAlignment.html) is [MainAxisAlignment.spaceBetween](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html), any vertical space that has not been allocated to children is divided evenly and placed between the children.

### Constructors

[Column](https://api.flutter.dev/flutter/widgets/Column/Column.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html) mainAxisAlignment = MainAxisAlignment.start, [MainAxisSize](https://api.flutter.dev/flutter/rendering/MainAxisSize.html) mainAxisSize = MainAxisSize.max, [CrossAxisAlignment](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html) crossAxisAlignment = CrossAxisAlignment.center, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html) verticalDirection = VerticalDirection.down, [TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)? textBaseline, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[]})

Creates a vertical array of children.

const

### Properties

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

The widgets below this widget in the tree.

finalinherited

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

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

finalinherited

[crossAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/crossAxisAlignment.html) → [CrossAxisAlignment](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html)

How the children should be placed along the cross axis.

finalinherited

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

The direction to use as the main axis.

finalinherited

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

The hash code for this object.

read-onlyinherited

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

Controls how one widget replaces another widget in the tree.

finalinherited

[mainAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/mainAxisAlignment.html) → [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html)

How the children should be placed along the main axis.

finalinherited

[mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) → [MainAxisSize](https://api.flutter.dev/flutter/rendering/MainAxisSize.html)

How much space should be occupied in the main axis.

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

[textBaseline](https://api.flutter.dev/flutter/widgets/Flex/textBaseline.html) → [TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)?

If aligning items according to their baseline, which baseline to use.

finalinherited

[textDirection](https://api.flutter.dev/flutter/widgets/Flex/textDirection.html) → [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)?

Determines the order to lay children out horizontally and how to interpret start and end in the horizontal direction.

finalinherited

[verticalDirection](https://api.flutter.dev/flutter/widgets/Flex/verticalDirection.html) → [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html)

Determines the order to lay children out vertically and how to interpret start and end in the vertical direction.

finalinherited

## CONTAINER

A convenience widget that combines common painting, positioning, and sizing widgets.

A container first surrounds the child with [padding](https://api.flutter.dev/flutter/widgets/Container/padding.html) (inflated by any borders present in the [decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html)) and then applies additional [constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html) to the padded extent (incorporating the width and height as constraints, if either is non-null). The container is then surrounded by additional empty space described from the [margin](https://api.flutter.dev/flutter/widgets/Container/margin.html).

During painting, the container first applies the given [transform](https://api.flutter.dev/flutter/widgets/Container/transform.html), then paints the [decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html) to fill the padded extent, then it paints the child, and finally paints the [foregroundDecoration](https://api.flutter.dev/flutter/widgets/Container/foregroundDecoration.html), also filling the padded extent.

Containers with no children try to be as big as possible unless the incoming constraints are unbounded, in which case they try to be as small as possible. Containers with children size themselves to their children. The width, height, and [constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html) arguments to the constructor override this.

By default, containers return false for all hit tests. If the [color](https://api.flutter.dev/flutter/widgets/Container/color.html) property is specified, the hit testing is handled by [ColoredBox](https://api.flutter.dev/flutter/widgets/ColoredBox-class.html), which always returns true. If the [decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html) or [foregroundDecoration](https://api.flutter.dev/flutter/widgets/Container/foregroundDecoration.html) properties are specified, hit testing is handled by [Decoration.hitTest](https://api.flutter.dev/flutter/painting/Decoration/hitTest.html).

### Layout behavior

See [*BoxConstraints*](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html) for an introduction to box layout models.

Since [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) combines a number of other widgets each with their own layout behavior, [Container](https://api.flutter.dev/flutter/widgets/Container-class.html)'s layout behavior is somewhat complicated.

Summary: [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) tries, in order: to honor [alignment](https://api.flutter.dev/flutter/widgets/Container/alignment.html), to size itself to the [child](https://api.flutter.dev/flutter/widgets/Container/child.html), to honor the width, height, and [constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html), to expand to fit the parent, to be as small as possible.

More specifically:

If the widget has no child, no height, no width, no [constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html), and the parent provides unbounded constraints, then [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) tries to size as small as possible.

If the widget has no child and no [alignment](https://api.flutter.dev/flutter/widgets/Container/alignment.html), but a height, width, or [constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html) are provided, then the [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) tries to be as small as possible given the combination of those constraints and the parent's constraints.

If the widget has no child, no height, no width, no [constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html), and no [alignment](https://api.flutter.dev/flutter/widgets/Container/alignment.html), but the parent provides bounded constraints, then [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) expands to fit the constraints provided by the parent.

If the widget has an [alignment](https://api.flutter.dev/flutter/widgets/Container/alignment.html), and the parent provides unbounded constraints, then the [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) tries to size itself around the child.

If the widget has an [alignment](https://api.flutter.dev/flutter/widgets/Container/alignment.html), and the parent provides bounded constraints, then the [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) tries to expand to fit the parent, and then positions the child within itself as per the [alignment](https://api.flutter.dev/flutter/widgets/Container/alignment.html).

Otherwise, the widget has a [child](https://api.flutter.dev/flutter/widgets/Container/child.html) but no height, no width, no [constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html), and no [alignment](https://api.flutter.dev/flutter/widgets/Container/alignment.html), and the [Container](https://api.flutter.dev/flutter/widgets/Container-class.html) passes the constraints from the parent to the child and sizes itself to match the child.

The [margin](https://api.flutter.dev/flutter/widgets/Container/margin.html) and [padding](https://api.flutter.dev/flutter/widgets/Container/padding.html) properties also affect the layout, as described in the documentation for those properties. (Their effects merely augment the rules described above.) The [decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html) can implicitly increase the [padding](https://api.flutter.dev/flutter/widgets/Container/padding.html) (e.g. borders in a [BoxDecoration](https://api.flutter.dev/flutter/painting/BoxDecoration-class.html) contribute to the [padding](https://api.flutter.dev/flutter/widgets/Container/padding.html)); see [Decoration.padding](https://api.flutter.dev/flutter/painting/Decoration/padding.html).

### Constructors

[Container](https://api.flutter.dev/flutter/widgets/Container/Container.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? alignment, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? padding, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)? decoration, [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)? foregroundDecoration, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [BoxConstraints](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html)? constraints, [EdgeInsetsGeometry](https://api.flutter.dev/flutter/painting/EdgeInsetsGeometry-class.html)? margin, [Matrix4](https://api.flutter.dev/flutter/vector_math_64/Matrix4-class.html)? transform, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)? transformAlignment, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none})

Creates a widget that combines common painting, positioning, and sizing widgets.

### Properties

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

Align the [child](https://api.flutter.dev/flutter/widgets/Container/child.html) within the container.

final

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

The [child](https://api.flutter.dev/flutter/widgets/Container/child.html) contained by the container.

final

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

The clip behavior when [Container.decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html) is not null.

final

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

The color to paint behind the [child](https://api.flutter.dev/flutter/widgets/Container/child.html).

final

[constraints](https://api.flutter.dev/flutter/widgets/Container/constraints.html) → [BoxConstraints](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html)?

Additional constraints to apply to the child.

final

[decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html) → [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)?

The decoration to paint behind the [child](https://api.flutter.dev/flutter/widgets/Container/child.html).

final

[foregroundDecoration](https://api.flutter.dev/flutter/widgets/Container/foregroundDecoration.html) → [Decoration](https://api.flutter.dev/flutter/painting/Decoration-class.html)?

The decoration to paint in front of the [child](https://api.flutter.dev/flutter/widgets/Container/child.html).

final

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

The hash code for this object.

read-onlyinherited

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

Controls how one widget replaces another widget in the tree.

finalinherited

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

Empty space to surround the [decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html) and [child](https://api.flutter.dev/flutter/widgets/Container/child.html).

final

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

Empty space to inscribe inside the [decoration](https://api.flutter.dev/flutter/widgets/Container/decoration.html). The [child](https://api.flutter.dev/flutter/widgets/Container/child.html), if any, is placed inside this padding.

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

[transform](https://api.flutter.dev/flutter/widgets/Container/transform.html) → [Matrix4](https://api.flutter.dev/flutter/vector_math_64/Matrix4-class.html)?

The transformation matrix to apply before painting the container.

final

[transformAlignment](https://api.flutter.dev/flutter/widgets/Container/transformAlignment.html) → [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)?

The alignment of the origin, relative to the size of the container, if [transform](https://api.flutter.dev/flutter/widgets/Container/transform.html) is specified.

final

## ELEVATED BUTTON

A Material Design "elevated button".

Use elevated buttons to add dimension to otherwise mostly flat layouts, e.g. in long busy lists of content, or in wide spaces. Avoid using elevated buttons on already-elevated content such as dialogs or cards.

An elevated button is a label [child](https://api.flutter.dev/flutter/material/ButtonStyleButton/child.html) displayed on a [Material](https://api.flutter.dev/flutter/material/Material-class.html) widget whose [Material.elevation](https://api.flutter.dev/flutter/material/Material/elevation.html) increases when the button is pressed. The label's [Text](https://api.flutter.dev/flutter/widgets/Text-class.html) and [Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html) widgets are displayed in [style](https://api.flutter.dev/flutter/material/ButtonStyleButton/style.html)'s [ButtonStyle.foregroundColor](https://api.flutter.dev/flutter/material/ButtonStyle/foregroundColor.html) and the button's filled background is the [ButtonStyle.backgroundColor](https://api.flutter.dev/flutter/material/ButtonStyle/backgroundColor.html).

The elevated button's default style is defined by [defaultStyleOf](https://api.flutter.dev/flutter/material/ElevatedButton/defaultStyleOf.html). The style of this elevated button can be overridden with its [style](https://api.flutter.dev/flutter/material/ButtonStyleButton/style.html) parameter. The style of all elevated buttons in a subtree can be overridden with the [ElevatedButtonTheme](https://api.flutter.dev/flutter/material/ElevatedButtonTheme-class.html), and the style of all of the elevated buttons in an app can be overridden with the [Theme](https://api.flutter.dev/flutter/material/Theme-class.html)'s [ThemeData.elevatedButtonTheme](https://api.flutter.dev/flutter/material/ThemeData/elevatedButtonTheme.html) property.

The static [styleFrom](https://api.flutter.dev/flutter/material/ElevatedButton/styleFrom.html) method is a convenient way to create a elevated button [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html) from simple values.

If [onPressed](https://api.flutter.dev/flutter/material/ButtonStyleButton/onPressed.html) and [onLongPress](https://api.flutter.dev/flutter/material/ButtonStyleButton/onLongPress.html) callbacks are null, then the button will be disabled.

See also:

* [FilledButton](https://api.flutter.dev/flutter/material/FilledButton-class.html), a filled button that doesn't elevate when pressed.
* [FilledButton.tonal](https://api.flutter.dev/flutter/material/FilledButton/FilledButton.tonal.html), a filled button variant that uses a secondary fill color.
* [OutlinedButton](https://api.flutter.dev/flutter/material/OutlinedButton-class.html), a button with an outlined border and no fill color.
* [TextButton](https://api.flutter.dev/flutter/material/TextButton-class.html), a button with no outline or fill color.
* [material.io/design/components/buttons.html](https://material.io/design/components/buttons.html)
* [m3.material.io/components/buttons](https://m3.material.io/components/buttons)

### Constructors

[ElevatedButton](https://api.flutter.dev/flutter/material/ElevatedButton/ElevatedButton.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onPressed, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onLongPress, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onHover, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) autofocus = false, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html) clipBehavior = Clip.none, [MaterialStatesController](https://api.flutter.dev/flutter/material/MaterialStatesController-class.html)? statesController, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Create an ElevatedButton.

const

[ElevatedButton.icon](https://api.flutter.dev/flutter/material/ElevatedButton/ElevatedButton.icon.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onPressed, [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)? onLongPress, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onHover, [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>? onFocusChange, [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)? style, [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)? focusNode, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? autofocus, [Clip](https://api.flutter.dev/flutter/dart-ui/Clip.html)? clipBehavior, [MaterialStatesController](https://api.flutter.dev/flutter/material/MaterialStatesController-class.html)? statesController, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) icon, required [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html) label})

Create an elevated button from a pair of widgets that serve as the button's icon and label.

factory

### Properties

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

True if this widget will be selected as the initial focus when no other node in its scope is currently focused.

finalinherited

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

Typically the button's label.

finalinherited

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

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

finalinherited

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

Whether the button is enabled or disabled.

read-onlyinherited

[focusNode](https://api.flutter.dev/flutter/material/ButtonStyleButton/focusNode.html) → [FocusNode](https://api.flutter.dev/flutter/widgets/FocusNode-class.html)?

An optional focus node to use as the focus node for this widget.

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

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

Determine whether this subtree represents a button.

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

[onFocusChange](https://api.flutter.dev/flutter/material/ButtonStyleButton/onFocusChange.html) → [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>?

Handler called when the focus changes.

finalinherited

[onHover](https://api.flutter.dev/flutter/material/ButtonStyleButton/onHover.html) → [ValueChanged](https://api.flutter.dev/flutter/foundation/ValueChanged.html)<[bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)>?

Called when a pointer enters or exits the button response area.

finalinherited

[onLongPress](https://api.flutter.dev/flutter/material/ButtonStyleButton/onLongPress.html) → [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)?

Called when the button is long-pressed.

finalinherited

[onPressed](https://api.flutter.dev/flutter/material/ButtonStyleButton/onPressed.html) → [VoidCallback](https://api.flutter.dev/flutter/dart-ui/VoidCallback.html)?

Called when the button is tapped or otherwise activated.

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

[statesController](https://api.flutter.dev/flutter/material/ButtonStyleButton/statesController.html) → [MaterialStatesController](https://api.flutter.dev/flutter/material/MaterialStatesController-class.html)?

Represents the interactive "state" of this widget in terms of a set of [MaterialState](https://api.flutter.dev/flutter/material/MaterialState.html)s, like [MaterialState.pressed](https://api.flutter.dev/flutter/material/MaterialState.html) and [MaterialState.focused](https://api.flutter.dev/flutter/material/MaterialState.html).

finalinherited

[style](https://api.flutter.dev/flutter/material/ButtonStyleButton/style.html) → [ButtonStyle](https://api.flutter.dev/flutter/material/ButtonStyle-class.html)?

Customizes this button's appearance.

finalinherited

## FLUTTER LOGO

The Flutter logo, in widget form. This widget respects the [IconTheme](https://api.flutter.dev/flutter/widgets/IconTheme-class.html). For guidelines on using the Flutter logo, visit <https://flutter.dev/brand>.

### Constructors

[FlutterLogo](https://api.flutter.dev/flutter/material/FlutterLogo/FlutterLogo.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? size, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html) textColor = const Color(0xFF757575), [FlutterLogoStyle](https://api.flutter.dev/flutter/painting/FlutterLogoStyle.html) style = FlutterLogoStyle.markOnly, [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html) duration = const Duration(milliseconds: 750), [Curve](https://api.flutter.dev/flutter/animation/Curve-class.html) curve = Curves.fastOutSlowIn})

Creates a widget that paints the Flutter logo.

const

### Properties

[curve](https://api.flutter.dev/flutter/material/FlutterLogo/curve.html) → [Curve](https://api.flutter.dev/flutter/animation/Curve-class.html)

The curve for the logo animation if the [style](https://api.flutter.dev/flutter/material/FlutterLogo/style.html) or [textColor](https://api.flutter.dev/flutter/material/FlutterLogo/textColor.html) change.

final

[duration](https://api.flutter.dev/flutter/material/FlutterLogo/duration.html) → [Duration](https://api.flutter.dev/flutter/dart-core/Duration-class.html)

The length of time for the animation if the [style](https://api.flutter.dev/flutter/material/FlutterLogo/style.html) or [textColor](https://api.flutter.dev/flutter/material/FlutterLogo/textColor.html) properties are changed.

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

[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

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

The size of the logo in logical pixels.

final

[style](https://api.flutter.dev/flutter/material/FlutterLogo/style.html) → [FlutterLogoStyle](https://api.flutter.dev/flutter/painting/FlutterLogoStyle.html)

Whether and where to draw the "Flutter" text. By default, only the logo itself is drawn.

final

[textColor](https://api.flutter.dev/flutter/material/FlutterLogo/textColor.html) → [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)

The color used to paint the "Flutter" text on the logo, if [style](https://api.flutter.dev/flutter/material/FlutterLogo/style.html) is [FlutterLogoStyle.horizontal](https://api.flutter.dev/flutter/painting/FlutterLogoStyle.html) or [FlutterLogoStyle.stacked](https://api.flutter.dev/flutter/painting/FlutterLogoStyle.html).

final

## ICON

A graphical icon widget drawn with a glyph from a font described in an [IconData](https://api.flutter.dev/flutter/widgets/IconData-class.html) such as material's predefined [IconData](https://api.flutter.dev/flutter/widgets/IconData-class.html)s in [Icons](https://api.flutter.dev/flutter/material/Icons-class.html).

Icons are not interactive. For an interactive icon, consider material's [IconButton](https://api.flutter.dev/flutter/material/IconButton-class.html).

There must be an ambient [Directionality](https://api.flutter.dev/flutter/widgets/Directionality-class.html) widget when using [Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html). Typically this is introduced automatically by the [WidgetsApp](https://api.flutter.dev/flutter/widgets/WidgetsApp-class.html) or [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html).

This widget assumes that the rendered icon is squared. Non-squared icons may render incorrectly.

### Constructors

[Icon](https://api.flutter.dev/flutter/widgets/Icon/Icon.html)([IconData](https://api.flutter.dev/flutter/widgets/IconData-class.html)? icon, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? size, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? fill, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? weight, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? grade, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? opticalSize, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Shadow](https://api.flutter.dev/flutter/dart-ui/Shadow-class.html)>? shadows, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection})

Creates an icon.

const

### Properties

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

The color to use when drawing the icon.

final

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

The fill for drawing the icon.

final

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

The grade (granular stroke weight) for drawing the icon.

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

[icon](https://api.flutter.dev/flutter/widgets/Icon/icon.html) → [IconData](https://api.flutter.dev/flutter/widgets/IconData-class.html)?

The icon to display. The available icons are described in [Icons](https://api.flutter.dev/flutter/material/Icons-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

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

The optical size for drawing the icon.

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

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

Semantic label for the icon.

final

[shadows](https://api.flutter.dev/flutter/widgets/Icon/shadows.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Shadow](https://api.flutter.dev/flutter/dart-ui/Shadow-class.html)>?

A list of Shadows that will be painted underneath the icon.

final

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

The size of the icon in logical pixels.

final

[textDirection](https://api.flutter.dev/flutter/widgets/Icon/textDirection.html) → [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)?

The text direction to use for rendering the icon.

final

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

The stroke weight for drawing the icon.

final

## IMAGE

A widget that displays an image.

Several constructors are provided for the various ways that an image can be specified:

* [Image.new](https://api.flutter.dev/flutter/widgets/Image/Image.html), for obtaining an image from an [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html).
* [Image.asset](https://api.flutter.dev/flutter/widgets/Image/Image.asset.html), for obtaining an image from an [AssetBundle](https://api.flutter.dev/flutter/services/AssetBundle-class.html) using a key.
* [Image.network](https://api.flutter.dev/flutter/widgets/Image/Image.network.html), for obtaining an image from a URL.
* [Image.file](https://api.flutter.dev/flutter/widgets/Image/Image.file.html), for obtaining an image from a [File](https://api.flutter.dev/flutter/dart-io/File-class.html).
* [Image.memory](https://api.flutter.dev/flutter/widgets/Image/Image.memory.html), for obtaining an image from a [Uint8List](https://api.flutter.dev/flutter/dart-typed_data/Uint8List-class.html).

The following image formats are supported: JPEG, PNG, GIF, Animated GIF, WebP, Animated WebP, BMP, and WBMP. Additional formats may be supported by the underlying platform. Flutter will attempt to call platform API to decode unrecognized formats, and if the platform API supports decoding the image Flutter will be able to render it.

To automatically perform pixel-density-aware asset resolution, specify the image using an [AssetImage](https://api.flutter.dev/flutter/painting/AssetImage-class.html) and make sure that a [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html), [WidgetsApp](https://api.flutter.dev/flutter/widgets/WidgetsApp-class.html), or [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html) widget exists above the [Image](https://api.flutter.dev/flutter/widgets/Image-class.html) widget in the widget tree.

The image is painted using [paintImage](https://api.flutter.dev/flutter/painting/paintImage.html), which describes the meanings of the various fields on this class in more detail.

### Memory usage

The image is stored in memory in uncompressed form (so that it can be rendered). Large images will use a lot of memory: a 4K image (3840×2160) will use over 30MB of RAM (assuming 32 bits per pixel).

This problem is exacerbated by the images being cached in the [ImageCache](https://api.flutter.dev/flutter/painting/ImageCache-class.html), so large images can use memory for even longer than they are displayed.

The [Image.asset](https://api.flutter.dev/flutter/widgets/Image/Image.asset.html), [Image.network](https://api.flutter.dev/flutter/widgets/Image/Image.network.html), [Image.file](https://api.flutter.dev/flutter/widgets/Image/Image.file.html), and [Image.memory](https://api.flutter.dev/flutter/widgets/Image/Image.memory.html) constructors allow a custom decode size to be specified through cacheWidth and cacheHeight parameters. The engine will then decode and store the image at the specified size, instead of the image's natural size.

This can significantly reduce the memory usage. For example, a 4K image that will be rendered at only 384×216 pixels (one-tenth the horizontal and vertical dimensions) would only use 330KB if those dimensions are specified using the cacheWidth and cacheHeight parameters, a 100-fold reduction in memory usage.

### Web considerations

In the case where a network image is used on the Web platform, the cacheWidth and cacheHeight parameters are only supported when the application is running with the CanvasKit renderer. When the application is using the HTML renderer, the web engine delegates image decoding of network images to the Web, which does not support custom decode sizes.

### Constructors

[Image](https://api.flutter.dev/flutter/widgets/Image/Image.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, required [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)> image, [ImageFrameBuilder](https://api.flutter.dev/flutter/widgets/ImageFrameBuilder.html)? frameBuilder, [ImageLoadingBuilder](https://api.flutter.dev/flutter/widgets/ImageLoadingBuilder.html)? loadingBuilder, [ImageErrorWidgetBuilder](https://api.flutter.dev/flutter/widgets/ImageErrorWidgetBuilder.html)? errorBuilder, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? opacity, [BlendMode](https://api.flutter.dev/flutter/dart-ui/BlendMode.html)? colorBlendMode, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)? fit, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html) repeat = ImageRepeat.noRepeat, [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)? centerSlice, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) matchTextDirection = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) gaplessPlayback = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isAntiAlias = false, [FilterQuality](https://api.flutter.dev/flutter/dart-ui/FilterQuality.html) filterQuality = FilterQuality.low})

Creates a widget that displays an image.

const

[Image.asset](https://api.flutter.dev/flutter/widgets/Image/Image.asset.html)([String](https://api.flutter.dev/flutter/dart-core/String-class.html) name, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [AssetBundle](https://api.flutter.dev/flutter/services/AssetBundle-class.html)? bundle, [ImageFrameBuilder](https://api.flutter.dev/flutter/widgets/ImageFrameBuilder.html)? frameBuilder, [ImageErrorWidgetBuilder](https://api.flutter.dev/flutter/widgets/ImageErrorWidgetBuilder.html)? errorBuilder, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? scale, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? opacity, [BlendMode](https://api.flutter.dev/flutter/dart-ui/BlendMode.html)? colorBlendMode, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)? fit, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html) repeat = ImageRepeat.noRepeat, [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)? centerSlice, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) matchTextDirection = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) gaplessPlayback = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isAntiAlias = false, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? package, [FilterQuality](https://api.flutter.dev/flutter/dart-ui/FilterQuality.html) filterQuality = FilterQuality.low, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheWidth, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheHeight})

Creates a widget that displays an [ImageStream](https://api.flutter.dev/flutter/painting/ImageStream-class.html) obtained from an asset bundle. The key for the image is given by the name argument.

[Image.file](https://api.flutter.dev/flutter/widgets/Image/Image.file.html)([File](https://api.flutter.dev/flutter/dart-io/File-class.html) file, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) scale = 1.0, [ImageFrameBuilder](https://api.flutter.dev/flutter/widgets/ImageFrameBuilder.html)? frameBuilder, [ImageErrorWidgetBuilder](https://api.flutter.dev/flutter/widgets/ImageErrorWidgetBuilder.html)? errorBuilder, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? opacity, [BlendMode](https://api.flutter.dev/flutter/dart-ui/BlendMode.html)? colorBlendMode, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)? fit, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html) repeat = ImageRepeat.noRepeat, [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)? centerSlice, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) matchTextDirection = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) gaplessPlayback = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isAntiAlias = false, [FilterQuality](https://api.flutter.dev/flutter/dart-ui/FilterQuality.html) filterQuality = FilterQuality.low, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheWidth, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheHeight})

Creates a widget that displays an [ImageStream](https://api.flutter.dev/flutter/painting/ImageStream-class.html) obtained from a [File](https://api.flutter.dev/flutter/dart-io/File-class.html).

[Image.memory](https://api.flutter.dev/flutter/widgets/Image/Image.memory.html)([Uint8List](https://api.flutter.dev/flutter/dart-typed_data/Uint8List-class.html) bytes, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) scale = 1.0, [ImageFrameBuilder](https://api.flutter.dev/flutter/widgets/ImageFrameBuilder.html)? frameBuilder, [ImageErrorWidgetBuilder](https://api.flutter.dev/flutter/widgets/ImageErrorWidgetBuilder.html)? errorBuilder, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? opacity, [BlendMode](https://api.flutter.dev/flutter/dart-ui/BlendMode.html)? colorBlendMode, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)? fit, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html) repeat = ImageRepeat.noRepeat, [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)? centerSlice, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) matchTextDirection = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) gaplessPlayback = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isAntiAlias = false, [FilterQuality](https://api.flutter.dev/flutter/dart-ui/FilterQuality.html) filterQuality = FilterQuality.low, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheWidth, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheHeight})

Creates a widget that displays an [ImageStream](https://api.flutter.dev/flutter/painting/ImageStream-class.html) obtained from a [Uint8List](https://api.flutter.dev/flutter/dart-typed_data/Uint8List-class.html).

[Image.network](https://api.flutter.dev/flutter/widgets/Image/Image.network.html)([String](https://api.flutter.dev/flutter/dart-core/String-class.html) src, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) scale = 1.0, [ImageFrameBuilder](https://api.flutter.dev/flutter/widgets/ImageFrameBuilder.html)? frameBuilder, [ImageLoadingBuilder](https://api.flutter.dev/flutter/widgets/ImageLoadingBuilder.html)? loadingBuilder, [ImageErrorWidgetBuilder](https://api.flutter.dev/flutter/widgets/ImageErrorWidgetBuilder.html)? errorBuilder, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticLabel, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) excludeFromSemantics = false, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? width, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? height, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? color, [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>? opacity, [BlendMode](https://api.flutter.dev/flutter/dart-ui/BlendMode.html)? colorBlendMode, [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)? fit, [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html) alignment = Alignment.center, [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html) repeat = ImageRepeat.noRepeat, [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)? centerSlice, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) matchTextDirection = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) gaplessPlayback = false, [FilterQuality](https://api.flutter.dev/flutter/dart-ui/FilterQuality.html) filterQuality = FilterQuality.low, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) isAntiAlias = false, [Map](https://api.flutter.dev/flutter/dart-core/Map-class.html)<[String](https://api.flutter.dev/flutter/dart-core/String-class.html), [String](https://api.flutter.dev/flutter/dart-core/String-class.html)>? headers, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheWidth, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? cacheHeight})

Creates a widget that displays an [ImageStream](https://api.flutter.dev/flutter/painting/ImageStream-class.html) obtained from the network.

### Properties

[alignment](https://api.flutter.dev/flutter/widgets/Image/alignment.html) → [AlignmentGeometry](https://api.flutter.dev/flutter/painting/AlignmentGeometry-class.html)

How to align the image within its bounds.

final

[centerSlice](https://api.flutter.dev/flutter/widgets/Image/centerSlice.html) → [Rect](https://api.flutter.dev/flutter/dart-ui/Rect-class.html)?

The center slice for a nine-patch image.

final

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

If non-null, this color is blended with each image pixel using [colorBlendMode](https://api.flutter.dev/flutter/widgets/Image/colorBlendMode.html).

final

[colorBlendMode](https://api.flutter.dev/flutter/widgets/Image/colorBlendMode.html) → [BlendMode](https://api.flutter.dev/flutter/dart-ui/BlendMode.html)?

Used to combine [color](https://api.flutter.dev/flutter/widgets/Image/color.html) with this image.

final

[errorBuilder](https://api.flutter.dev/flutter/widgets/Image/errorBuilder.html) → [ImageErrorWidgetBuilder](https://api.flutter.dev/flutter/widgets/ImageErrorWidgetBuilder.html)?

A builder function that is called if an error occurs during image loading.

final

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

Whether to exclude this image from semantics.

final

[filterQuality](https://api.flutter.dev/flutter/widgets/Image/filterQuality.html) → [FilterQuality](https://api.flutter.dev/flutter/dart-ui/FilterQuality.html)

The rendering quality of the image.

final

[fit](https://api.flutter.dev/flutter/widgets/Image/fit.html) → [BoxFit](https://api.flutter.dev/flutter/painting/BoxFit.html)?

How to inscribe the image into the space allocated during layout.

final

[frameBuilder](https://api.flutter.dev/flutter/widgets/Image/frameBuilder.html) → [ImageFrameBuilder](https://api.flutter.dev/flutter/widgets/ImageFrameBuilder.html)?

A builder function responsible for creating the widget that represents this image.

final

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

Whether to continue showing the old image (true), or briefly show nothing (false), when the image provider changes. The default value is false.

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

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

If non-null, require the image to have this height (in logical pixels).

final

[image](https://api.flutter.dev/flutter/widgets/Image/image.html) → [ImageProvider](https://api.flutter.dev/flutter/painting/ImageProvider-class.html)<[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)>

The image to display.

final

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

Whether to paint the image with anti-aliasing.

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

[loadingBuilder](https://api.flutter.dev/flutter/widgets/Image/loadingBuilder.html) → [ImageLoadingBuilder](https://api.flutter.dev/flutter/widgets/ImageLoadingBuilder.html)?

A builder that specifies the widget to display to the user while an image is still loading.

final

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

Whether to paint the image in the direction of the [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html).

final

[opacity](https://api.flutter.dev/flutter/widgets/Image/opacity.html) → [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html)<[double](https://api.flutter.dev/flutter/dart-core/double-class.html)>?

If non-null, the value from the [Animation](https://api.flutter.dev/flutter/animation/Animation-class.html) is multiplied with the opacity of each image pixel before painting onto the canvas.

final

[repeat](https://api.flutter.dev/flutter/widgets/Image/repeat.html) → [ImageRepeat](https://api.flutter.dev/flutter/painting/ImageRepeat.html)

How to paint any portions of the layout bounds not covered by the image.

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

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

A Semantic description of the image.

final

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

If non-null, require the image to have this width (in logical pixels).

## PLACEHOLDER

A widget that draws a box that represents where other widgets will one day be added.

This widget is useful during development to indicate that the interface is not yet complete.

By default, the placeholder is sized to fit its container. If the placeholder is in an unbounded space, it will size itself according to the given [fallbackWidth](https://api.flutter.dev/flutter/widgets/Placeholder/fallbackWidth.html) and [fallbackHeight](https://api.flutter.dev/flutter/widgets/Placeholder/fallbackHeight.html).

### Constructors

[Placeholder](https://api.flutter.dev/flutter/widgets/Placeholder/Placeholder.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html) color = const Color(0xFF455A64), [double](https://api.flutter.dev/flutter/dart-core/double-class.html) strokeWidth = 2.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) fallbackWidth = 400.0, [double](https://api.flutter.dev/flutter/dart-core/double-class.html) fallbackHeight = 400.0, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? child})

Creates a widget which draws a box.

const

### Properties

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

The [child](https://api.flutter.dev/flutter/widgets/Placeholder/child.html) contained by the placeholder box.

final

[color](https://api.flutter.dev/flutter/widgets/Placeholder/color.html) → [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)

The color to draw the placeholder box.

final

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

The height to use when the placeholder is in a situation with an unbounded height.

final

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

The width to use when the placeholder is in a situation with an unbounded width.

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

[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

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

The width of the lines in the placeholder box.

final

## ROW

A widget that displays its children in a horizontal array.

To cause a child to expand to fill the available horizontal space, wrap the child in an [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) widget.

The [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) widget does not scroll (and in general it is considered an error to have more children in a [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) than will fit in the available room). If you have a line of widgets and want them to be able to scroll if there is insufficient room, consider using a [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html).

For a vertical variant, see [Column](https://api.flutter.dev/flutter/widgets/Column-class.html).

If you only have one child, then consider using [Align](https://api.flutter.dev/flutter/widgets/Align-class.html) or [Center](https://api.flutter.dev/flutter/widgets/Center-class.html) to position the child.

### Why does my row have a yellow and black warning stripe?

If the non-flexible contents of the row (those that are not wrapped in [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) or [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) widgets) are together wider than the row itself, then the row is said to have overflowed. When a row overflows, the row does not have any remaining space to share between its [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) and [Flexible](https://api.flutter.dev/flutter/widgets/Flexible-class.html) children. The row reports this by drawing a yellow and black striped warning box on the edge that is overflowing. If there is room on the outside of the row, the amount of overflow is printed in red lettering.

**Story time**

Suppose, for instance, that you had this code:

const Row(

children: <Widget>[

FlutterLogo(),

Text("Flutter's hot reload helps you quickly and easily experiment, build UIs, add features, and fix bug faster. Experience sub-second reload times, without losing state, on emulators, simulators, and hardware for iOS and Android."),

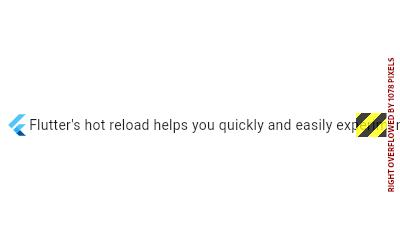
Icon(Icons.sentiment\_very\_satisfied),

],

)

The row first asks its first child, the [FlutterLogo](https://api.flutter.dev/flutter/material/FlutterLogo-class.html), to lay out, at whatever size the logo would like. The logo is friendly and happily decides to be 24 pixels to a side. This leaves lots of room for the next child. The row then asks that next child, the text, to lay out, at whatever size it thinks is best.

At this point, the text, not knowing how wide is too wide, says "Ok, I will be thiiiiiiiiiiiiiiiiiiiis wide.", and goes well beyond the space that the row has available, not wrapping. The row responds, "That's not fair, now I have no more room available for my other children!", and gets angry and sprouts a yellow and black strip.



The fix is to wrap the second child in an [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html) widget, which tells the row that the child should be given the remaining room:

const Row(

children: <Widget>[

FlutterLogo(),

Expanded(

child: Text("Flutter's hot reload helps you quickly and easily experiment, build UIs, add features, and fix bug faster. Experience sub-second reload times, without losing state, on emulators, simulators, and hardware for iOS and Android."),

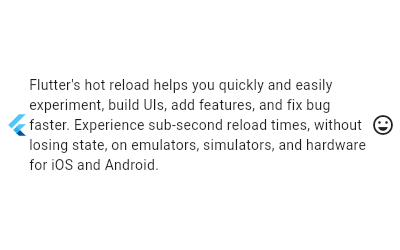
),

Icon(Icons.sentiment\_very\_satisfied),

],

)

Now, the row first asks the logo to lay out, and then asks the *icon* to lay out. The [Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html), like the logo, is happy to take on a reasonable size (also 24 pixels, not coincidentally, since both [FlutterLogo](https://api.flutter.dev/flutter/material/FlutterLogo-class.html) and [Icon](https://api.flutter.dev/flutter/widgets/Icon-class.html) honor the ambient [IconTheme](https://api.flutter.dev/flutter/widgets/IconTheme-class.html)). This leaves some room left over, and now the row tells the text exactly how wide to be: the exact width of the remaining space. The text, now happy to comply to a reasonable request, wraps the text within that width, and you end up with a paragraph split over several lines.



The [textDirection](https://api.flutter.dev/flutter/widgets/Flex/textDirection.html) property controls the direction that children are rendered in. [TextDirection.ltr](https://api.flutter.dev/flutter/dart-ui/TextDirection.html) is the default [textDirection](https://api.flutter.dev/flutter/widgets/Flex/textDirection.html) of [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) children, so the first child is rendered at the start of the [Row](https://api.flutter.dev/flutter/widgets/Row-class.html), to the left, with subsequent children following to the right. If you want to order children in the opposite direction (right to left), then [textDirection](https://api.flutter.dev/flutter/widgets/Flex/textDirection.html) can be set to [TextDirection.rtl](https://api.flutter.dev/flutter/dart-ui/TextDirection.html). This is shown in the example below

const Row(

textDirection: TextDirection.rtl,

children: <Widget>[

FlutterLogo(),

Expanded(

child: Text("Flutter's hot reload helps you quickly and easily experiment, build UIs, add features, and fix bug faster. Experience sub-second reload times, without losing state, on emulators, simulators, and hardware for iOS and Android."),

),

Icon(Icons.sentiment\_very\_satisfied),

],

)

### Layout algorithm

This section describes how a [*Row*](https://api.flutter.dev/flutter/widgets/Row-class.html) is rendered by the framework. See [*BoxConstraints*](https://api.flutter.dev/flutter/rendering/BoxConstraints-class.html) for an introduction to box layout models.

Layout for a [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) proceeds in six steps:

1. Layout each child with a null or zero flex factor (e.g., those that are not [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html)) with unbounded horizontal constraints and the incoming vertical constraints. If the [crossAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/crossAxisAlignment.html) is [CrossAxisAlignment.stretch](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html), instead use tight vertical constraints that match the incoming max height.
2. Divide the remaining horizontal space among the children with non-zero flex factors (e.g., those that are [Expanded](https://api.flutter.dev/flutter/widgets/Expanded-class.html)) according to their flex factor. For example, a child with a flex factor of 2.0 will receive twice the amount of horizontal space as a child with a flex factor of 1.0.
3. Layout each of the remaining children with the same vertical constraints as in step 1, but instead of using unbounded horizontal constraints, use horizontal constraints based on the amount of space allocated in step 2. Children with [Flexible.fit](https://api.flutter.dev/flutter/widgets/Flexible/fit.html) properties that are [FlexFit.tight](https://api.flutter.dev/flutter/rendering/FlexFit.html) are given tight constraints (i.e., forced to fill the allocated space), and children with [Flexible.fit](https://api.flutter.dev/flutter/widgets/Flexible/fit.html) properties that are [FlexFit.loose](https://api.flutter.dev/flutter/rendering/FlexFit.html) are given loose constraints (i.e., not forced to fill the allocated space).
4. The height of the [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) is the maximum height of the children (which will always satisfy the incoming vertical constraints).
5. The width of the [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) is determined by the [mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) property. If the [mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) property is [MainAxisSize.max](https://api.flutter.dev/flutter/rendering/MainAxisSize.html), then the width of the [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) is the max width of the incoming constraints. If the [mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) property is [MainAxisSize.min](https://api.flutter.dev/flutter/rendering/MainAxisSize.html), then the width of the [Row](https://api.flutter.dev/flutter/widgets/Row-class.html) is the sum of widths of the children (subject to the incoming constraints).
6. Determine the position for each child according to the [mainAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/mainAxisAlignment.html) and the [crossAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/crossAxisAlignment.html). For example, if the [mainAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/mainAxisAlignment.html) is [MainAxisAlignment.spaceBetween](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html), any horizontal space that has not been allocated to children is divided evenly and placed between the children.

### Constructors

[Row](https://api.flutter.dev/flutter/widgets/Row/Row.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html) mainAxisAlignment = MainAxisAlignment.start, [MainAxisSize](https://api.flutter.dev/flutter/rendering/MainAxisSize.html) mainAxisSize = MainAxisSize.max, [CrossAxisAlignment](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html) crossAxisAlignment = CrossAxisAlignment.center, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html) verticalDirection = VerticalDirection.down, [TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)? textBaseline, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)> children = const <Widget>[]})

Creates a horizontal array of children.

const

### Properties

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

The widgets below this widget in the tree.

finalinherited

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

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

finalinherited

[crossAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/crossAxisAlignment.html) → [CrossAxisAlignment](https://api.flutter.dev/flutter/rendering/CrossAxisAlignment.html)

How the children should be placed along the cross axis.

finalinherited

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

The direction to use as the main axis.

finalinherited

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

The hash code for this object.

read-onlyinherited

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

Controls how one widget replaces another widget in the tree.

finalinherited

[mainAxisAlignment](https://api.flutter.dev/flutter/widgets/Flex/mainAxisAlignment.html) → [MainAxisAlignment](https://api.flutter.dev/flutter/rendering/MainAxisAlignment.html)

How the children should be placed along the main axis.

finalinherited

[mainAxisSize](https://api.flutter.dev/flutter/widgets/Flex/mainAxisSize.html) → [MainAxisSize](https://api.flutter.dev/flutter/rendering/MainAxisSize.html)

How much space should be occupied in the main axis.

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

[textBaseline](https://api.flutter.dev/flutter/widgets/Flex/textBaseline.html) → [TextBaseline](https://api.flutter.dev/flutter/dart-ui/TextBaseline.html)?

If aligning items according to their baseline, which baseline to use.

finalinherited

[textDirection](https://api.flutter.dev/flutter/widgets/Flex/textDirection.html) → [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)?

Determines the order to lay children out horizontally and how to interpret start and end in the horizontal direction.

finalinherited

[verticalDirection](https://api.flutter.dev/flutter/widgets/Flex/verticalDirection.html) → [VerticalDirection](https://api.flutter.dev/flutter/painting/VerticalDirection.html)

Determines the order to lay children out vertically and how to interpret start and end in the vertical direction.

finalinherited

## SCAFFOLD

Implements the basic Material Design visual layout structure.

This class provides APIs for showing drawers and bottom sheets.

To display a persistent bottom sheet, obtain the [ScaffoldState](https://api.flutter.dev/flutter/material/ScaffoldState-class.html) for the current [BuildContext](https://api.flutter.dev/flutter/widgets/BuildContext-class.html) via [Scaffold.of](https://api.flutter.dev/flutter/material/Scaffold/of.html) and use the [ScaffoldState.showBottomSheet](https://api.flutter.dev/flutter/material/ScaffoldState/showBottomSheet.html) function.

Scaffold layout, the keyboard, and display "notches**"**

The scaffold will expand to fill the available space. That usually means that it will occupy its entire window or device screen. When the device's keyboard appears the Scaffold's ancestor [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html) widget's [MediaQueryData.viewInsets](https://api.flutter.dev/flutter/widgets/MediaQueryData/viewInsets.html) changes and the Scaffold will be rebuilt. By default the scaffold's [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) is resized to make room for the keyboard. To prevent the resize set [resizeToAvoidBottomInset](https://api.flutter.dev/flutter/material/Scaffold/resizeToAvoidBottomInset.html) to false. In either case the focused widget will be scrolled into view if it's within a scrollable container.

The [MediaQueryData.padding](https://api.flutter.dev/flutter/widgets/MediaQueryData/padding.html) value defines areas that might not be completely visible, like the display "notch" on the iPhone X. The scaffold's [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) is not inset by this padding value although an [appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) or [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) will typically cause the body to avoid the padding. The [SafeArea](https://api.flutter.dev/flutter/widgets/SafeArea-class.html) widget can be used within the scaffold's body to avoid areas like display notches.

### Floating action button with a draggable scrollable bottom sheet

If [Scaffold.bottomSheet](https://api.flutter.dev/flutter/material/Scaffold/bottomSheet.html) is a [DraggableScrollableSheet](https://api.flutter.dev/flutter/widgets/DraggableScrollableSheet-class.html), [Scaffold.floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) is set, and the bottom sheet is dragged to cover greater than 70% of the Scaffold's height, two things happen in parallel:

* Scaffold starts to show scrim (see [ScaffoldState.showBodyScrim](https://api.flutter.dev/flutter/material/ScaffoldState/showBodyScrim.html)), and
* [Scaffold.floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) is scaled down through an animation with a [Curves.easeIn](https://api.flutter.dev/flutter/animation/Curves/easeIn-constant.html), and disappears when the bottom sheet covers the entire Scaffold.

And as soon as the bottom sheet is dragged down to cover less than 70% of the [Scaffold](https://api.flutter.dev/flutter/material/Scaffold-class.html), the scrim disappears and [Scaffold.floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) animates back to its normal size.

**Troubleshooting**

### Nested Scaffolds

The Scaffold is designed to be a top level container for a [MaterialApp](https://api.flutter.dev/flutter/material/MaterialApp-class.html). This means that adding a Scaffold to each route on a Material app will provide the app with Material's basic visual layout structure.

It is typically not necessary to nest Scaffolds. For example, in a tabbed UI, where the [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) is a [TabBar](https://api.flutter.dev/flutter/material/TabBar-class.html) and the body is a [TabBarView](https://api.flutter.dev/flutter/material/TabBarView-class.html), you might be tempted to make each tab bar view a scaffold with a differently titled AppBar. Rather, it would be better to add a listener to the [TabController](https://api.flutter.dev/flutter/material/TabController-class.html) that updates the AppBar

Add a listener to the app's tab controller so that the [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html) title of the app's one and only scaffold is reset each time a new tab is selected.

[*link*](https://api.flutter.dev/flutter/)

TabController(vsync: tickerProvider, length: tabCount)..addListener(() {

if (!tabController.indexIsChanging) {

setState(() {

// Rebuild the enclosing scaffold with a new AppBar title

appBarTitle = 'Tab ${tabController.index}';

});

}

})

Although there are some use cases, like a presentation app that shows embedded flutter content, where nested scaffolds are appropriate, it's best to avoid nesting scaffolds.

See also:

* [AppBar](https://api.flutter.dev/flutter/material/AppBar-class.html), which is a horizontal bar typically shown at the top of an app using the [appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) property.
* [BottomAppBar](https://api.flutter.dev/flutter/material/BottomAppBar-class.html), which is a horizontal bar typically shown at the bottom of an app using the [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) property.
* [FloatingActionButton](https://api.flutter.dev/flutter/material/FloatingActionButton-class.html), which is a circular button typically shown in the bottom right corner of the app using the [floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) property.
* [Drawer](https://api.flutter.dev/flutter/material/Drawer-class.html), which is a vertical panel that is typically displayed to the left of the body (and often hidden on phones) using the [drawer](https://api.flutter.dev/flutter/material/Scaffold/drawer.html) property.
* [BottomNavigationBar](https://api.flutter.dev/flutter/material/BottomNavigationBar-class.html), which is a horizontal array of buttons typically shown along the bottom of the app using the [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) property.
* [BottomSheet](https://api.flutter.dev/flutter/material/BottomSheet-class.html), which is an overlay typically shown near the bottom of the app. A bottom sheet can either be persistent, in which case it is shown using the [ScaffoldState.showBottomSheet](https://api.flutter.dev/flutter/material/ScaffoldState/showBottomSheet.html) method, or modal, in which case it is shown using the [showModalBottomSheet](https://api.flutter.dev/flutter/material/showModalBottomSheet.html) function.
* [SnackBar](https://api.flutter.dev/flutter/material/SnackBar-class.html), which is a lightweight message with an optional action which briefly displays at the bottom of the screen. Use the [ScaffoldMessengerState.showSnackBar](https://api.flutter.dev/flutter/material/ScaffoldMessengerState/showSnackBar.html) method to show snack bars.
* [MaterialBanner](https://api.flutter.dev/flutter/material/MaterialBanner-class.html), which displays an important, succinct message, at the top of the screen, below the app bar. Use the [ScaffoldMessengerState.showMaterialBanner](https://api.flutter.dev/flutter/material/ScaffoldMessengerState/showMaterialBanner.html) method to show material banners.
* [ScaffoldState](https://api.flutter.dev/flutter/material/ScaffoldState-class.html), which is the state associated with this widget.
* [material.io/design/layout/responsive-layout-grid.html](https://material.io/design/layout/responsive-layout-grid.html)
* Cookbook: [Add a Drawer to a screen](https://flutter.dev/docs/cookbook/design/drawer)

### Constructors

[Scaffold](https://api.flutter.dev/flutter/material/Scaffold/Scaffold.html)({[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [PreferredSizeWidget](https://api.flutter.dev/flutter/widgets/PreferredSizeWidget-class.html)? appBar, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? body, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? floatingActionButton, [FloatingActionButtonLocation](https://api.flutter.dev/flutter/material/FloatingActionButtonLocation-class.html)? floatingActionButtonLocation, [FloatingActionButtonAnimator](https://api.flutter.dev/flutter/material/FloatingActionButtonAnimator-class.html)? floatingActionButtonAnimator, [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>? persistentFooterButtons, [AlignmentDirectional](https://api.flutter.dev/flutter/painting/AlignmentDirectional-class.html) persistentFooterAlignment = AlignmentDirectional.centerEnd, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? drawer, [DrawerCallback](https://api.flutter.dev/flutter/material/DrawerCallback.html)? onDrawerChanged, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? endDrawer, [DrawerCallback](https://api.flutter.dev/flutter/material/DrawerCallback.html)? onEndDrawerChanged, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? bottomNavigationBar, [Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)? bottomSheet, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? backgroundColor, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? resizeToAvoidBottomInset, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) primary = true, [DragStartBehavior](https://api.flutter.dev/flutter/gestures/DragStartBehavior.html) drawerDragStartBehavior = DragStartBehavior.start, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) extendBody = false, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) extendBodyBehindAppBar = false, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? drawerScrimColor, [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? drawerEdgeDragWidth, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) drawerEnableOpenDragGesture = true, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html) endDrawerEnableOpenDragGesture = true, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? restorationId})

Creates a visual scaffold for Material Design widgets.

const

### Properties

[appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) → [PreferredSizeWidget](https://api.flutter.dev/flutter/widgets/PreferredSizeWidget-class.html)?

An app bar to display at the top of the scaffold.

final

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

The color of the [Material](https://api.flutter.dev/flutter/material/Material-class.html) widget that underlies the entire Scaffold.

final

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

The primary content of the scaffold.

final

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

A bottom navigation bar to display at the bottom of the scaffold.

final

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

The persistent bottom sheet to display.

final

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

A panel displayed to the side of the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html), often hidden on mobile devices. Swipes in from either left-to-right ([TextDirection.ltr](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)) or right-to-left ([TextDirection.rtl](https://api.flutter.dev/flutter/dart-ui/TextDirection.html))

final

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

Determines the way that drag start behavior is handled.

final

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

The width of the area within which a horizontal swipe will open the drawer.

final

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

Determines if the [Scaffold.drawer](https://api.flutter.dev/flutter/material/Scaffold/drawer.html) can be opened with a drag gesture on mobile.

final

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

The color to use for the scrim that obscures primary content while a drawer is open.

final

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

A panel displayed to the side of the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html), often hidden on mobile devices. Swipes in from right-to-left ([TextDirection.ltr](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)) or left-to-right ([TextDirection.rtl](https://api.flutter.dev/flutter/dart-ui/TextDirection.html))

final

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

Determines if the [Scaffold.endDrawer](https://api.flutter.dev/flutter/material/Scaffold/endDrawer.html) can be opened with a gesture on mobile.

final

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

If true, and [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) or [persistentFooterButtons](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html) is specified, then the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) extends to the bottom of the Scaffold, instead of only extending to the top of the [bottomNavigationBar](https://api.flutter.dev/flutter/material/Scaffold/bottomNavigationBar.html) or the [persistentFooterButtons](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html).

final

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

If true, and an [appBar](https://api.flutter.dev/flutter/material/Scaffold/appBar.html) is specified, then the height of the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) is extended to include the height of the app bar and the top of the body is aligned with the top of the app bar.

final

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

A button displayed floating above [body](https://api.flutter.dev/flutter/material/Scaffold/body.html), in the bottom right corner.

final

[floatingActionButtonAnimator](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonAnimator.html) → [FloatingActionButtonAnimator](https://api.flutter.dev/flutter/material/FloatingActionButtonAnimator-class.html)?

Animator to move the [floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) to a new [floatingActionButtonLocation](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonLocation.html).

final

[floatingActionButtonLocation](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButtonLocation.html) → [FloatingActionButtonLocation](https://api.flutter.dev/flutter/material/FloatingActionButtonLocation-class.html)?

Responsible for determining where the [floatingActionButton](https://api.flutter.dev/flutter/material/Scaffold/floatingActionButton.html) should go.

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

[onDrawerChanged](https://api.flutter.dev/flutter/material/Scaffold/onDrawerChanged.html) → [DrawerCallback](https://api.flutter.dev/flutter/material/DrawerCallback.html)?

Optional callback that is called when the [Scaffold.drawer](https://api.flutter.dev/flutter/material/Scaffold/drawer.html) is opened or closed.

final

[onEndDrawerChanged](https://api.flutter.dev/flutter/material/Scaffold/onEndDrawerChanged.html) → [DrawerCallback](https://api.flutter.dev/flutter/material/DrawerCallback.html)?

Optional callback that is called when the [Scaffold.endDrawer](https://api.flutter.dev/flutter/material/Scaffold/endDrawer.html) is opened or closed.

final

[persistentFooterAlignment](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterAlignment.html) → [AlignmentDirectional](https://api.flutter.dev/flutter/painting/AlignmentDirectional-class.html)

The alignment of the [persistentFooterButtons](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html) inside the [OverflowBar](https://api.flutter.dev/flutter/widgets/OverflowBar-class.html).

final

[persistentFooterButtons](https://api.flutter.dev/flutter/material/Scaffold/persistentFooterButtons.html) → [List](https://api.flutter.dev/flutter/dart-core/List-class.html)<[Widget](https://api.flutter.dev/flutter/widgets/Widget-class.html)>?

A set of buttons that are displayed at the bottom of the scaffold.

final

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

Whether this scaffold is being displayed at the top of the screen.

final

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

If true the [body](https://api.flutter.dev/flutter/material/Scaffold/body.html) and the scaffold's floating widgets should size themselves to avoid the onscreen keyboard whose height is defined by the ambient [MediaQuery](https://api.flutter.dev/flutter/widgets/MediaQuery-class.html)'s [MediaQueryData.viewInsets](https://api.flutter.dev/flutter/widgets/MediaQueryData/viewInsets.html) bottom property.

final

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

Restoration ID to save and restore the state of the [Scaffold](https://api.flutter.dev/flutter/material/Scaffold-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

## TEXT

A run of text with a single style.

The [Text](https://api.flutter.dev/flutter/widgets/Text-class.html) widget displays a string of text with single style. The string might break across multiple lines or might all be displayed on the same line depending on the layout constraints.

The [style](https://api.flutter.dev/flutter/widgets/Text/style.html) argument is optional. When omitted, the text will use the style from the closest enclosing [DefaultTextStyle](https://api.flutter.dev/flutter/widgets/DefaultTextStyle-class.html). If the given style's [TextStyle.inherit](https://api.flutter.dev/flutter/painting/TextStyle/inherit.html) property is true (the default), the given style will be merged with the closest enclosing [DefaultTextStyle](https://api.flutter.dev/flutter/widgets/DefaultTextStyle-class.html). This merging behavior is useful, for example, to make the text bold while using the default font family and size

### interactivity

To make [Text](https://api.flutter.dev/flutter/widgets/Text-class.html) react to touch events, wrap it in a [GestureDetector](https://api.flutter.dev/flutter/widgets/GestureDetector-class.html) widget with a [GestureDetector.onTap](https://api.flutter.dev/flutter/widgets/GestureDetector/onTap.html) handler.

In a Material Design application, consider using a [TextButton](https://api.flutter.dev/flutter/material/TextButton-class.html) instead, or if that isn't appropriate, at least using an [InkWell](https://api.flutter.dev/flutter/material/InkWell-class.html) instead of [GestureDetector](https://api.flutter.dev/flutter/widgets/GestureDetector-class.html).

To make sections of the text interactive, use [RichText](https://api.flutter.dev/flutter/widgets/RichText-class.html) and specify a [TapGestureRecognizer](https://api.flutter.dev/flutter/gestures/TapGestureRecognizer-class.html) as the [TextSpan.recognizer](https://api.flutter.dev/flutter/painting/TextSpan/recognizer.html) of the relevant part of the text.

### Selection

[Text](https://api.flutter.dev/flutter/widgets/Text-class.html) is not selectable by default. To make a [Text](https://api.flutter.dev/flutter/widgets/Text-class.html) selectable, one can wrap a subtree with a [SelectionArea](https://api.flutter.dev/flutter/material/SelectionArea-class.html) widget. To exclude a part of a subtree under [SelectionArea](https://api.flutter.dev/flutter/material/SelectionArea-class.html) from selection, once can also wrap that part of the subtree with [SelectionContainer.disabled](https://api.flutter.dev/flutter/widgets/SelectionContainer/SelectionContainer.disabled.html).

### Constructors

[Text](https://api.flutter.dev/flutter/widgets/Text/Text.html)([String](https://api.flutter.dev/flutter/dart-core/String-class.html) data, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? style, [StrutStyle](https://api.flutter.dev/flutter/painting/StrutStyle-class.html)? strutStyle, [TextAlign](https://api.flutter.dev/flutter/dart-ui/TextAlign.html)? textAlign, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [Locale](https://api.flutter.dev/flutter/dart-ui/Locale-class.html)? locale, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? softWrap, [TextOverflow](https://api.flutter.dev/flutter/painting/TextOverflow.html)? overflow, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Use textScaler instead. ' 'Use of textScaleFactor was deprecated in preparation for the upcoming nonlinear text scaling support. ' 'This feature was deprecated after v3.12.0-2.0.pre.') [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? textScaleFactor, [TextScaler](https://api.flutter.dev/flutter/painting/TextScaler-class.html)? textScaler, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? maxLines, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsLabel, [TextWidthBasis](https://api.flutter.dev/flutter/painting/TextWidthBasis.html)? textWidthBasis, [TextHeightBehavior](https://api.flutter.dev/flutter/dart-ui/TextHeightBehavior-class.html)? textHeightBehavior, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectionColor})

Creates a text widget.

const

[Text.rich](https://api.flutter.dev/flutter/widgets/Text/Text.rich.html)([InlineSpan](https://api.flutter.dev/flutter/painting/InlineSpan-class.html) textSpan, {[Key](https://api.flutter.dev/flutter/foundation/Key-class.html)? key, [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)? style, [StrutStyle](https://api.flutter.dev/flutter/painting/StrutStyle-class.html)? strutStyle, [TextAlign](https://api.flutter.dev/flutter/dart-ui/TextAlign.html)? textAlign, [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)? textDirection, [Locale](https://api.flutter.dev/flutter/dart-ui/Locale-class.html)? locale, [bool](https://api.flutter.dev/flutter/dart-core/bool-class.html)? softWrap, [TextOverflow](https://api.flutter.dev/flutter/painting/TextOverflow.html)? overflow, @[Deprecated](https://api.flutter.dev/flutter/dart-core/Deprecated-class.html)('Use textScaler instead. ' 'Use of textScaleFactor was deprecated in preparation for the upcoming nonlinear text scaling support. ' 'This feature was deprecated after v3.12.0-2.0.pre.') [double](https://api.flutter.dev/flutter/dart-core/double-class.html)? textScaleFactor, [TextScaler](https://api.flutter.dev/flutter/painting/TextScaler-class.html)? textScaler, [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? maxLines, [String](https://api.flutter.dev/flutter/dart-core/String-class.html)? semanticsLabel, [TextWidthBasis](https://api.flutter.dev/flutter/painting/TextWidthBasis.html)? textWidthBasis, [TextHeightBehavior](https://api.flutter.dev/flutter/dart-ui/TextHeightBehavior-class.html)? textHeightBehavior, [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)? selectionColor})

Creates a text widget with a [InlineSpan](https://api.flutter.dev/flutter/painting/InlineSpan-class.html).

const

### Properties

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

The text to display.

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

[locale](https://api.flutter.dev/flutter/widgets/Text/locale.html) → [Locale](https://api.flutter.dev/flutter/dart-ui/Locale-class.html)?

Used to select a font when the same Unicode character can be rendered differently, depending on the locale.

final

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

An optional maximum number of lines for the text to span, wrapping if necessary. If the text exceeds the given number of lines, it will be truncated according to [overflow](https://api.flutter.dev/flutter/widgets/Text/overflow.html).

final

[overflow](https://api.flutter.dev/flutter/widgets/Text/overflow.html) → [TextOverflow](https://api.flutter.dev/flutter/painting/TextOverflow.html)?

How visual overflow should be handled.

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

[selectionColor](https://api.flutter.dev/flutter/widgets/Text/selectionColor.html) → [Color](https://api.flutter.dev/flutter/dart-ui/Color-class.html)?

The color to use when painting the selection.

final

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

An alternative semantics label for this text.

final

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

Whether the text should break at soft line breaks.

final

[strutStyle](https://api.flutter.dev/flutter/widgets/Text/strutStyle.html) → [StrutStyle](https://api.flutter.dev/flutter/painting/StrutStyle-class.html)?

The strut style to use. Strut style defines the strut, which sets minimum vertical layout metrics.

final

[style](https://api.flutter.dev/flutter/widgets/Text/style.html) → [TextStyle](https://api.flutter.dev/flutter/painting/TextStyle-class.html)?

If non-null, the style to use for this text.

final

[textAlign](https://api.flutter.dev/flutter/widgets/Text/textAlign.html) → [TextAlign](https://api.flutter.dev/flutter/dart-ui/TextAlign.html)?

How the text should be aligned horizontally.

final

[textDirection](https://api.flutter.dev/flutter/widgets/Text/textDirection.html) → [TextDirection](https://api.flutter.dev/flutter/dart-ui/TextDirection.html)?

The directionality of the text.

final

[textHeightBehavior](https://api.flutter.dev/flutter/widgets/Text/textHeightBehavior.html) → [TextHeightBehavior](https://api.flutter.dev/flutter/dart-ui/TextHeightBehavior-class.html)?

Defines how to apply [TextStyle.height](https://api.flutter.dev/flutter/painting/TextStyle/height.html) over and under text.

final

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

Deprecated. Will be removed in a future version of Flutter. Use [textScaler](https://api.flutter.dev/flutter/widgets/Text/textScaler.html) instead.

final

[textScaler](https://api.flutter.dev/flutter/widgets/Text/textScaler.html) → [TextScaler](https://api.flutter.dev/flutter/painting/TextScaler-class.html)?

The font scaling strategy to use when laying out and rendering the text.

final

[textSpan](https://api.flutter.dev/flutter/widgets/Text/textSpan.html) → [InlineSpan](https://api.flutter.dev/flutter/painting/InlineSpan-class.html)?

The text to display as a [InlineSpan](https://api.flutter.dev/flutter/painting/InlineSpan-class.html).

final

[textWidthBasis](https://api.flutter.dev/flutter/widgets/Text/textWidthBasis.html) → [TextWidthBasis](https://api.flutter.dev/flutter/painting/TextWidthBasis.html)?

Defines how to measure the width of the rendered text.

final