# Column class

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://docs.flutter.io/flutter/widgets/Expanded-class.html) widget.

The [Column](https://docs.flutter.io/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://docs.flutter.io/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://docs.flutter.io/flutter/widgets/ListView-class.html).

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

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

## Sample code

This example uses a [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) to arrange three widgets vertically, the last being made to fill all the remaining space.

**new** Column(

children: <Widget>[

**new** Text('Deliver features faster'),

**new** Text('Craft beautiful UIs'),

**new** Expanded(

child: **new** FittedBox(

fit: BoxFit.contain, *// otherwise the logo will be tiny*

child: **const** FlutterLogo(),

),

),

],

)

In the sample above, the text and the logo are centered on each line. In the following example, the [crossAxisAlignment](https://docs.flutter.io/flutter/widgets/Flex/crossAxisAlignment.html) is set to [CrossAxisAlignment.start](https://docs.flutter.io/flutter/rendering/CrossAxisAlignment-class.html), so that the children are left-aligned. The[mainAxisSize](https://docs.flutter.io/flutter/widgets/Flex/mainAxisSize.html) is set to [MainAxisSize.min](https://docs.flutter.io/flutter/rendering/MainAxisSize-class.html), so that the column shrinks to fit the children.

**new** Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.min,

children: <Widget>[

**new** Text('We move under cover and we move as one'),

**new** Text('Through the night, we have one shot to live another day'),

**new** Text('We cannot let a stray gunshot give us away'),

**new** Text('We will fight up close, seize the moment and stay in it'),

**new** Text('It’s either that or meet the business end of a bayonet'),

**new** Text('The code word is ‘Rochambeau,’ dig me?'),

**new** Text('Rochambeau!', style: DefaultTextStyle.of(context).style.apply(fontSizeFactor: 2.0)),

],

)

## Troubleshooting

### When the incoming vertical constraints are unbounded

When a [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) has one or more [Expanded](https://docs.flutter.io/flutter/widgets/Expanded-class.html) or [Flexible](https://docs.flutter.io/flutter/widgets/Flexible-class.html) children, and is placed in another [Column](https://docs.flutter.io/flutter/widgets/Column-class.html), or in a [ListView](https://docs.flutter.io/flutter/widgets/ListView-class.html), or in some other context that does not provide a maximum height constraint for the [Column](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Flexible-class.html) or [Expanded](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Column-class.html) is receiving unbounded vertical constraints.

One common reason for this to happen is that the [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) has been placed in another [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) (without using [Expanded](https://docs.flutter.io/flutter/widgets/Expanded-class.html) or [Flexible](https://docs.flutter.io/flutter/widgets/Flexible-class.html) around the inner nested [Column](https://docs.flutter.io/flutter/widgets/Column-class.html)). When a [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) lays out its non-flex children (those that have neither [Expanded](https://docs.flutter.io/flutter/widgets/Expanded-class.html) or [Flexible](https://docs.flutter.io/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://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Column-class.html) inside a [ListView](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Column-class.html)should have an [Expanded](https://docs.flutter.io/flutter/widgets/Expanded-class.html) or [Flexible](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Expanded-class.html) or [Flexible](https://docs.flutter.io/flutter/widgets/Flexible-class.html) widgets from around the inner children.

For more discussion about constraints, see [BoxConstraints](https://docs.flutter.io/flutter/rendering/BoxConstraints-class.html).

### The yellow and black striped banner

When the contents of a [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) exceed the amount of space available, the [Column](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Column-class.html) saying how much overflow was detected.

The usual solution is to use a [ListView](https://docs.flutter.io/flutter/widgets/ListView-class.html) rather than a [Column](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Column-class.html) is rendered by the framework. See [BoxConstraints](https://docs.flutter.io/flutter/rendering/BoxConstraints-class.html) for an introduction to box layout models.

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

1. Layout each child a null or zero flex factor (e.g., those that are not [Expanded](https://docs.flutter.io/flutter/widgets/Expanded-class.html)) with unbounded vertical constraints and the incoming horizontal constraints. If the [crossAxisAlignment](https://docs.flutter.io/flutter/widgets/Flex/crossAxisAlignment.html) is[CrossAxisAlignment.stretch](https://docs.flutter.io/flutter/rendering/CrossAxisAlignment-class.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://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Flexible/fit.html) properties that are [FlexFit.tight](https://docs.flutter.io/flutter/rendering/FlexFit-class.html) are given tight constraints (i.e., forced to fill the allocated space), and children with [Flexible.fit](https://docs.flutter.io/flutter/widgets/Flexible/fit.html) properties that are [FlexFit.loose](https://docs.flutter.io/flutter/rendering/FlexFit-class.html)are given loose constraints (i.e., not forced to fill the allocated space).
4. The width of the [Column](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Column-class.html) is determined by the [mainAxisSize](https://docs.flutter.io/flutter/widgets/Flex/mainAxisSize.html) property. If the [mainAxisSize](https://docs.flutter.io/flutter/widgets/Flex/mainAxisSize.html) property is [MainAxisSize.max](https://docs.flutter.io/flutter/rendering/MainAxisSize-class.html), then the height of the [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) is the max height of the incoming constraints. If the[mainAxisSize](https://docs.flutter.io/flutter/widgets/Flex/mainAxisSize.html) property is [MainAxisSize.min](https://docs.flutter.io/flutter/rendering/MainAxisSize-class.html), then the height of the [Column](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Flex/mainAxisAlignment.html) and the [crossAxisAlignment](https://docs.flutter.io/flutter/widgets/Flex/crossAxisAlignment.html). For example, if the [mainAxisAlignment](https://docs.flutter.io/flutter/widgets/Flex/mainAxisAlignment.html) is [MainAxisAlignment.spaceBetween](https://docs.flutter.io/flutter/rendering/MainAxisAlignment-class.html), any vertical space that has not been allocated to children is divided evenly and placed between the children.

See also:

* [Row](https://docs.flutter.io/flutter/widgets/Row-class.html), for a horizontal equivalent.
* [Flex](https://docs.flutter.io/flutter/widgets/Flex-class.html), if you don't know in advance if you want a horizontal or vertical arrangement.
* [Expanded](https://docs.flutter.io/flutter/widgets/Expanded-class.html), to indicate children that should take all the remaining room.
* [Flexible](https://docs.flutter.io/flutter/widgets/Flexible-class.html), to indicate children that should share the remaining room but that may size smaller (leaving some remaining room unused).
* [SingleChildScrollView](https://docs.flutter.io/flutter/widgets/SingleChildScrollView-class.html), whose documentation discusses some ways to use a [Column](https://docs.flutter.io/flutter/widgets/Column-class.html) inside a scrolling container.
* The [catalog of layout widgets](https://flutter.io/widgets/layout/).

Inheritance

* [Object](https://docs.flutter.io/flutter/dart-core/Object-class.html)
* [Diagnosticable](https://docs.flutter.io/flutter/foundation/Diagnosticable-class.html)
* [DiagnosticableTree](https://docs.flutter.io/flutter/foundation/DiagnosticableTree-class.html)
* [Widget](https://docs.flutter.io/flutter/widgets/Widget-class.html)
* [RenderObjectWidget](https://docs.flutter.io/flutter/widgets/RenderObjectWidget-class.html)
* [MultiChildRenderObjectWidget](https://docs.flutter.io/flutter/widgets/MultiChildRenderObjectWidget-class.html)
* [Flex](https://docs.flutter.io/flutter/widgets/Flex-class.html)
* Column

## Constructors

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

Creates a vertical array of children. [[...]](https://docs.flutter.io/flutter/widgets/Column/Column.html)

## Properties

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

The widgets below this widget in the tree. [[...]](https://docs.flutter.io/flutter/widgets/MultiChildRenderObjectWidget/children.html)

*final, inherited*

[*crossAxisAlignment*](https://docs.flutter.io/flutter/widgets/Flex/crossAxisAlignment.html) → [CrossAxisAlignment](https://docs.flutter.io/flutter/rendering/CrossAxisAlignment-class.html)

How the children should be placed along the cross axis. [[...]](https://docs.flutter.io/flutter/widgets/Flex/crossAxisAlignment.html)

*final, inherited*

[*direction*](https://docs.flutter.io/flutter/widgets/Flex/direction.html) → [Axis](https://docs.flutter.io/flutter/painting/Axis-class.html)

The direction to use as the main axis. [[...]](https://docs.flutter.io/flutter/widgets/Flex/direction.html)

*final, inherited*

[*hashCode*](https://docs.flutter.io/flutter/dart-core/Object/hashCode.html) → [int](https://docs.flutter.io/flutter/dart-core/int-class.html)

The hash code for this object. [[...]](https://docs.flutter.io/flutter/dart-core/Object/hashCode.html)

*read-only, inherited*

[*key*](https://docs.flutter.io/flutter/widgets/Widget/key.html) → [Key](https://docs.flutter.io/flutter/foundation/Key-class.html)

Controls how one widget replaces another widget in the tree. [[...]](https://docs.flutter.io/flutter/widgets/Widget/key.html)

*final, inherited*

[*mainAxisAlignment*](https://docs.flutter.io/flutter/widgets/Flex/mainAxisAlignment.html) → [MainAxisAlignment](https://docs.flutter.io/flutter/rendering/MainAxisAlignment-class.html)

How the children should be placed along the main axis. [[...]](https://docs.flutter.io/flutter/widgets/Flex/mainAxisAlignment.html)

*final, inherited*

[*mainAxisSize*](https://docs.flutter.io/flutter/widgets/Flex/mainAxisSize.html) → [MainAxisSize](https://docs.flutter.io/flutter/rendering/MainAxisSize-class.html)

How much space should be occupied in the main axis. [[...]](https://docs.flutter.io/flutter/widgets/Flex/mainAxisSize.html)

*final, inherited*

[*runtimeType*](https://docs.flutter.io/flutter/dart-core/Object/runtimeType.html) → [Type](https://docs.flutter.io/flutter/dart-core/Type-class.html)

A representation of the runtime type of the object.

*read-only, inherited*

[*textBaseline*](https://docs.flutter.io/flutter/widgets/Flex/textBaseline.html) → [TextBaseline](https://docs.flutter.io/flutter/dart-ui/TextBaseline-class.html)

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

*final, inherited*

[*textDirection*](https://docs.flutter.io/flutter/widgets/Flex/textDirection.html) → [TextDirection](https://docs.flutter.io/flutter/dart-ui/TextDirection-class.html)

Determines the order to lay children out horizontally and how to interpret start and end in the horizontal direction. [[...]](https://docs.flutter.io/flutter/widgets/Flex/textDirection.html)

*final, inherited*

[*verticalDirection*](https://docs.flutter.io/flutter/widgets/Flex/verticalDirection.html) → [VerticalDirection](https://docs.flutter.io/flutter/painting/VerticalDirection-class.html)

Determines the order to lay children out vertically and how to interpret start and end in the vertical direction. [[...]](https://docs.flutter.io/flutter/widgets/Flex/verticalDirection.html)

*final, inherited*

## Methods

[*createElement*](https://docs.flutter.io/flutter/widgets/MultiChildRenderObjectWidget/createElement.html)( → [MultiChildRenderObjectElement](https://docs.flutter.io/flutter/widgets/MultiChildRenderObjectElement-class.html)

RenderObjectWidgets always inflate to a [RenderObjectElement](https://docs.flutter.io/flutter/widgets/RenderObjectElement-class.html) subclass.

*inherited*

[*createRenderObject*](https://docs.flutter.io/flutter/widgets/Flex/createRenderObject.html)([BuildContext](https://docs.flutter.io/flutter/widgets/BuildContext-class.html) context) → [RenderFlex](https://docs.flutter.io/flutter/rendering/RenderFlex-class.html)

Creates an instance of the [RenderObject](https://docs.flutter.io/flutter/rendering/RenderObject-class.html) class that this [RenderObjectWidget](https://docs.flutter.io/flutter/widgets/RenderObjectWidget-class.html) represents, using the configuration described by this[RenderObjectWidget](https://docs.flutter.io/flutter/widgets/RenderObjectWidget-class.html). [[...]](https://docs.flutter.io/flutter/widgets/Flex/createRenderObject.html)

*inherited*

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

Returns a list of [DiagnosticsNode](https://docs.flutter.io/flutter/foundation/DiagnosticsNode-class.html) objects describing this node's children. [[...]](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/debugDescribeChildren.html)

*@protected, inherited*

[*debugFillProperties*](https://docs.flutter.io/flutter/widgets/Flex/debugFillProperties.html)([DiagnosticPropertiesBuilder](https://docs.flutter.io/flutter/foundation/DiagnosticPropertiesBuilder-class.html) properties) → void

*inherited*

[*didUnmountRenderObject*](https://docs.flutter.io/flutter/widgets/RenderObjectWidget/didUnmountRenderObject.html)([RenderObject](https://docs.flutter.io/flutter/rendering/RenderObject-class.html) renderObject) → void

A render object previously associated with this widget has been removed from the tree. The given [RenderObject](https://docs.flutter.io/flutter/rendering/RenderObject-class.html) will be of the same type as returned by this object's [createRenderObject](https://docs.flutter.io/flutter/widgets/Flex/createRenderObject.html).

*@protected, inherited*

[*getEffectiveTextDirection*](https://docs.flutter.io/flutter/widgets/Flex/getEffectiveTextDirection.html)([BuildContext](https://docs.flutter.io/flutter/widgets/BuildContext-class.html) context) → [TextDirection](https://docs.flutter.io/flutter/dart-ui/TextDirection-class.html)

The value to pass to [RenderFlex.textDirection](https://docs.flutter.io/flutter/rendering/RenderFlex/textDirection.html). [[...]](https://docs.flutter.io/flutter/widgets/Flex/getEffectiveTextDirection.html)

*@protected, inherited*

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

Invoked when a non-existent method or property is accessed. [[...]](https://docs.flutter.io/flutter/dart-core/Object/noSuchMethod.html)

*inherited*

[*toDiagnosticsNode*](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)([String](https://docs.flutter.io/flutter/dart-core/String-class.html) name, [DiagnosticsTreeStyle](https://docs.flutter.io/flutter/foundation/DiagnosticsTreeStyle-class.html) style }) → [DiagnosticsNode](https://docs.flutter.io/flutter/foundation/DiagnosticsNode-class.html)

Returns a debug representation of the object that is used by debugging tools and by [toStringDeep](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/toStringDeep.html). [[...]](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/toDiagnosticsNode.html)

*inherited*

[*toString*](https://docs.flutter.io/flutter/foundation/Diagnosticable/toString.html)([DiagnosticLevel](https://docs.flutter.io/flutter/foundation/DiagnosticLevel-class.html) minLevel: DiagnosticLevel.debug }) → [String](https://docs.flutter.io/flutter/dart-core/String-class.html)

Returns a string representation of this object.

*inherited*

[*toStringDeep*](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/toStringDeep.html)([String](https://docs.flutter.io/flutter/dart-core/String-class.html) prefixLineOne: '', [String](https://docs.flutter.io/flutter/dart-core/String-class.html) prefixOtherLines, [DiagnosticLevel](https://docs.flutter.io/flutter/foundation/DiagnosticLevel-class.html) minLevel: DiagnosticLevel.debug }) → [String](https://docs.flutter.io/flutter/dart-core/String-class.html)

Returns a string representation of this node and its descendants. [[...]](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/toStringDeep.html)

*inherited*

[*toStringShallow*](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/toStringShallow.html)([String](https://docs.flutter.io/flutter/dart-core/String-class.html) joiner: ', ', [DiagnosticLevel](https://docs.flutter.io/flutter/foundation/DiagnosticLevel-class.html) minLevel: DiagnosticLevel.debug }) → [String](https://docs.flutter.io/flutter/dart-core/String-class.html)

Returns a one-line detailed description of the object. [[...]](https://docs.flutter.io/flutter/foundation/DiagnosticableTree/toStringShallow.html)

*inherited*

[*toStringShort*](https://docs.flutter.io/flutter/widgets/Widget/toStringShort.html)( → [String](https://docs.flutter.io/flutter/dart-core/String-class.html)

A short, textual description of this widget.

*inherited*

[*updateRenderObject*](https://docs.flutter.io/flutter/widgets/Flex/updateRenderObject.html)([BuildContext](https://docs.flutter.io/flutter/widgets/BuildContext-class.html) context, [RenderFlex](https://docs.flutter.io/flutter/rendering/RenderFlex-class.html) renderObject) → void

Copies the configuration described by this [RenderObjectWidget](https://docs.flutter.io/flutter/widgets/RenderObjectWidget-class.html) to the given [RenderObject](https://docs.flutter.io/flutter/rendering/RenderObject-class.html), which will be of the same type as returned by this object's [createRenderObject](https://docs.flutter.io/flutter/widgets/Flex/createRenderObject.html). [[...]](https://docs.flutter.io/flutter/widgets/Flex/updateRenderObject.html)

*inherited*

## Operators

[*operator ==*](https://docs.flutter.io/flutter/dart-core/Object/operator_equals.html)(dynamic other) → [bool](https://docs.flutter.io/flutter/dart-core/bool-class.html)

The equality operator. [[...]](https://docs.flutter.io/flutter/dart-core/Object/operator_equals.html)

*inherited*