# Row class

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

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

For a vertical variant, see [Column](https://docs.flutter.io/flutter/widgets/Column-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 divides the available space into three (horizontally), and places text centered in the first two cells and the Flutter logo centered in the third:

**new** Row(

children: <Widget>[

**new** Expanded(

child: **new** Text('Deliver features faster', textAlign: TextAlign.center),

),

**new** Expanded(

child: **new** Text('Craft beautiful UIs', textAlign: TextAlign.center),

),

**new** Expanded(

child: **new** FittedBox(

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

child: **const** FlutterLogo(),

),

),

],

)

## Troubleshooting

### 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://docs.flutter.io/flutter/widgets/Expanded-class.html) or [Flexible](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Expanded-class.html) and [Flexible](https://docs.flutter.io/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:

**new** Row(

children: <Widget>[

**const** FlutterLogo(),

**const** 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.'),

**const** Icon(Icons.sentiment\_very\_satisfied),

],

)

The row first asks its first child, the [FlutterLogo](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Expanded-class.html) widget, which tells the row that the child should be given the remaining room:

**new** Row(

children: <Widget>[

**const** FlutterLogo(),

**const** Expanded(

child: **const** 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.'),

),

**const** 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://docs.flutter.io/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://docs.flutter.io/flutter/material/FlutterLogo-class.html) and [Icon](https://docs.flutter.io/flutter/widgets/Icon-class.html) honor the ambient [IconTheme](https://docs.flutter.io/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.

## Layout algorithm

This section describes how a [Row](https://docs.flutter.io/flutter/widgets/Row-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 [Row](https://docs.flutter.io/flutter/widgets/Row-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 horizontal constraints and the incoming vertical 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 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://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 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://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 height of the [Row](https://docs.flutter.io/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://docs.flutter.io/flutter/widgets/Row-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 width of the [Row](https://docs.flutter.io/flutter/widgets/Row-class.html) is the max width 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 width of the [Row](https://docs.flutter.io/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://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 horizontal space that has not been allocated to children is divided evenly and placed between the children.

See also:

* [Column](https://docs.flutter.io/flutter/widgets/Column-class.html), for a vertical 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 by sized smaller (leaving some remaining room unused).
* 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)
* Row

## Constructors

[**Row**](https://docs.flutter.io/flutter/widgets/Row/Row.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 horizontal array of children. [[...]](https://docs.flutter.io/flutter/widgets/Row/Row.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*