Introduction to widgets

## **Hello world**

The minimal Flutter app simply calls the [runApp()](https://api.flutter.dev/flutter/widgets/runApp.html) function with a widget:

*content\_copy*

import 'package:flutter/material.dart';

void main() {

runApp(

Center(

child: Text(

'Hello, world!',

textDirection: TextDirection.ltr,

),

),

);

}

## **Basic widgets**

Flutter comes with a suite of powerful basic widgets, of which the following are commonly used:

[**Text**](https://api.flutter.dev/flutter/widgets/Text-class.html)

[**Row**](https://api.flutter.dev/flutter/widgets/Row-class.html)**,**[**Column**](https://api.flutter.dev/flutter/widgets/Column-class.html)

[**Stack**](https://api.flutter.dev/flutter/widgets/Stack-class.html) **posionned , ( overflow prop )**

[**Container**](https://api.flutter.dev/flutter/widgets/Container-class.html)

[**Icon**](https://api.flutter.dev/flutter/widgets/Container-class.html)**,**

**TextField**

**Radio**

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

title: 'Flutter Tutorial',

home: TutorialHome(),

));

}

class TutorialHome extends StatelessWidget {

@override

Widget build(BuildContext context) {

// Scaffold is a layout for the major Material Components.

return Scaffold(

appBar: AppBar(

leading: IconButton(

icon: Icon(Icons.menu),

tooltip: 'Navigation menu',

onPressed: null,

),

title: Text('Example title'),

actions: <Widget>[

IconButton(

icon: Icon(Icons.search),

tooltip: 'Search',

onPressed: null,

),

],

),

// body is the majority of the screen.

body: Center(

child: Text('Hello, world!'),

),

floatingActionButton: FloatingActionButton(

tooltip: 'Add', // used by assistive technologies

child: Icon(Icons.add),

onPressed: null,

),

);

}

}

## **Handling gestures**

Most applications include some form of user interaction with the system. The first step in building an interactive application is to detect input gestures. See how that works by creating a simple button:

*content\_copy*

class MyButton extends StatelessWidget {

@override

Widget build(BuildContext context) {

return GestureDetector(

onTap: () {

print('MyButton was tapped!');

},

child: Container(

height: 36.0,

padding: const EdgeInsets.all(8.0),

margin: const EdgeInsets.symmetric(horizontal: 8.0),

decoration: BoxDecoration(

borderRadius: BorderRadius.circular(5.0),

color: Colors.lightGreen[500],

),

child: Center(

child: Text('Engage'),

),

),

);

}

}

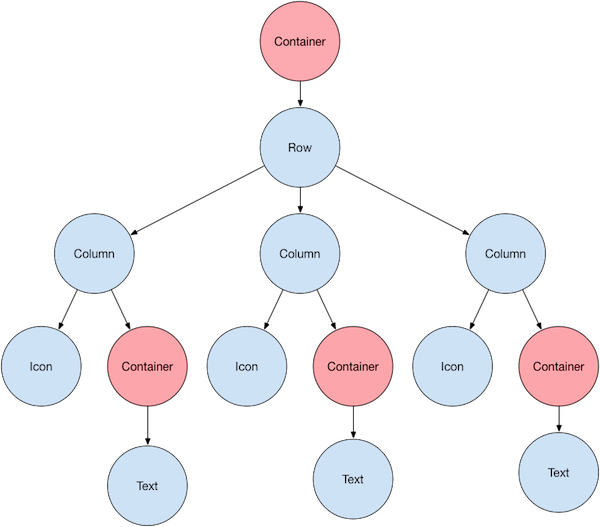
## **Responding to widget lifecycle events**

After calling [createState()](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html#createState) on the StatefulWidget, the framework inserts the new state object into the tree and then calls [initState()](https://api.flutter.dev/flutter/widgets/State-class.html#initState) on the state object. A subclass of [State](https://api.flutter.dev/flutter/widgets/State-class.html) can override initState to do work that needs to happen just once. For example, override initState to configure animations or to subscribe to platform services. Implementations of initState are required to start by calling super.initState.

When a state object is no longer needed, the framework calls [dispose()](https://api.flutter.dev/flutter/widgets/State-class.html#dispose) on the state object. Override the dispose function to do cleanup work. For example, override dispose to cancel timers or to unsubscribe from platform services. Implementations of dispose typically end by calling super.dispose.

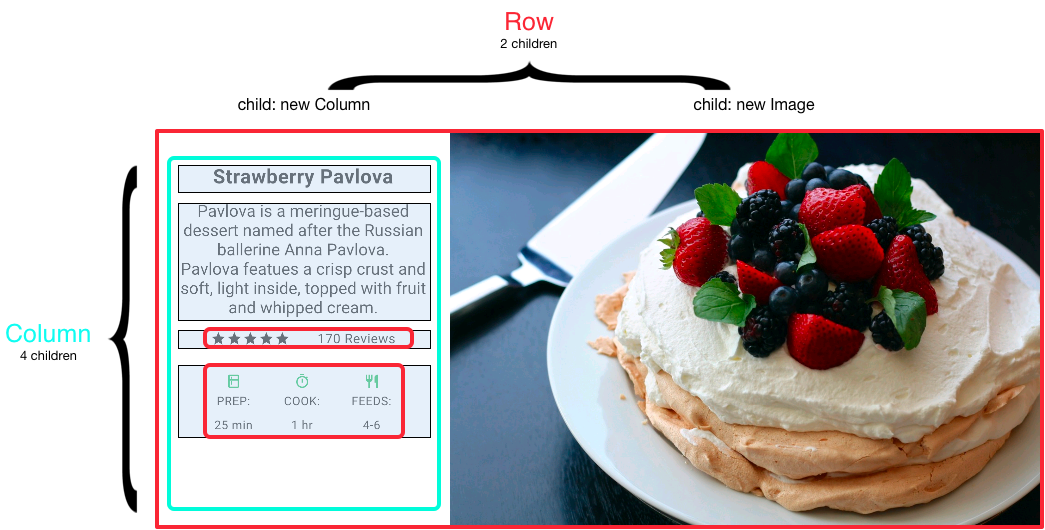
For more information, see [State](https://api.flutter.dev/flutter/widgets/State-class.html).

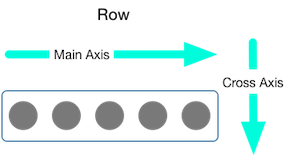
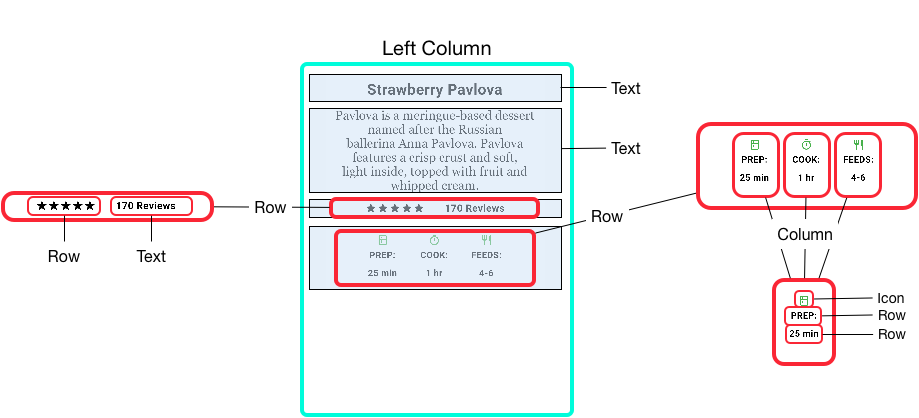
# Layouts in Flutter



#### **What's the point?**

* Row and Column are two of the most commonly used layout patterns.
* Row and Column each take a list of child widgets.
* A child widget can itself be a Row, Column, or other complex widget.
* You can specify how a Row or Column aligns its children, both vertically and horizontally.
* You can stretch or constrain specific child widgets.
* You can specify how child widgets use the Row’s or Column’s available space.





Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

Image.asset('images/pic1.jpg'),

Image.asset('images/pic2.jpg'),

Image.asset('images/pic3.jpg'),

],

);

### **Packing widgets**

By default, a row or column occupies as much space along its main axis as possible, but if you want to pack the children closely together, set its mainAxisSize to MainAxisSize.min. The following example uses this property to pack the star icons together.

*content\_copy*

Row(

mainAxisSize: MainAxisSize.min,

children: [

Icon(Icons.star, color: Colors.green[500]),

Icon(Icons.star, color: Colors.green[500]),

Icon(Icons.star, color: Colors.green[500]),

Icon(Icons.star, color: Colors.black),

Icon(Icons.star, color: Colors.black),

],

)

### **Standard widgets**

* [Container](https://flutter.dev/docs/development/ui/layout#container): Adds padding, margins, borders, background color, or other decorations to a widget.
* [GridView](https://flutter.dev/docs/development/ui/layout#gridview): Lays widgets out as a scrollable grid.
* [ListView](https://flutter.dev/docs/development/ui/layout#listview): Lays widgets out as a scrollable list.
* [Stack](https://flutter.dev/docs/development/ui/layout#stack): Overlaps a widget on top of another.

### **GridView**

Use [GridView](https://api.flutter.dev/flutter/widgets/GridView-class.html) to lay widgets out as a two-dimensional list. GridView provides two pre-fabricated lists, or you can build your own custom grid. When a GridView detects that its contents are too long to fit the render box, it automatically scrolls.

  Widget \_buildGrid() => GridView.extent(

    maxCrossAxisExtent: 100,

    padding: const EdgeInsets.all(4),

    mainAxisSpacing: 5,

    crossAxisSpacing: 4,

    children: \_buildGridTileList(30));

List<Container> \_buildGridTileList(int count) => List.generate(

    count, (i) => Container(child: Text('hello world'), color: Colors.blue,));

### **ListView**

[ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html), a column-like widget, automatically provides scrolling when its content is too long for its render box.

Widget \_buildList() => ListView(

      children: [

        \_tile('CineArts at the Empire', '85 W Portal Ave', Icons.theaters),

        \_tile('The Castro Theater', '429 Castro St', Icons.theaters),

        \_tile('Alamo Drafthouse Cinema', '2550 Mission St', Icons.theaters),

        \_tile('Roxie Theater', '3117 16th St', Icons.theaters),

        \_tile('United Artists Stonestown Twin', '501 Buckingham Way',

            Icons.theaters),

        \_tile('AMC Metreon 16', '135 4th St #3000', Icons.theaters),

        Divider(),

        \_tile('K\'s Kitchen', '757 Monterey Blvd', Icons.restaurant),

        \_tile('Emmy\'s Restaurant', '1923 Ocean Ave', Icons.restaurant),

        \_tile(

            'Chaiya Thai Restaurant', '272 Claremont Blvd', Icons.restaurant),

        \_tile('La Ciccia', '291 30th St', Icons.restaurant),

      ],

    );

    ListTile \_tile(String title, String subtitle, IconData icon) => ListTile(

      title: Text(title,

          style: TextStyle(

            fontWeight: FontWeight.w500,

            fontSize: 20,

          )),

      subtitle: Text(subtitle),

      leading: Icon(

        icon,

        color: Colors.blue[500],

      ),

    );

### **Stack**

Use [Stack](https://api.flutter.dev/flutter/widgets/Stack-class.html) to arrange widgets on top of a base widget—often an image. The widgets can completely or partially overlap the base widget.

Widget \_buildStack() => Stack(

alignment: const Alignment(0.6, 0.6),

children: [

CircleAvatar(

backgroundImage: AssetImage('images/pic.jpg'),

radius: 100,

),

Container(

decoration: BoxDecoration(

color: Colors.black45,

),

child: Text(

'Mia B',

style: TextStyle(

fontSize: 20,

fontWeight: FontWeight.bold,

color: Colors.white,

),

),

),

],

);

### **Card**

A [Card](https://api.flutter.dev/flutter/material/Card-class.html), from the [Material library](https://api.flutter.dev/flutter/material/material-library.html), contains related nuggets of information and can be composed from almost any widget, but is often used with [ListTile](https://api.flutter.dev/flutter/material/ListTile-class.html). Card has a single child, but its child can be a column, row, list, grid, or other widget that supports multiple children. By default, a Card shrinks its size to 0 by 0 pixels. You can use [SizedBox](https://api.flutter.dev/flutter/widgets/SizedBox-class.html) to constrain the size of a card.

Widget \_buildCard() => SizedBox(

    height: 210,

    child: Card(

      child: Column(

        children: [

          ListTile(

            title: Text('1625 Main Street',

                style: TextStyle(fontWeight: FontWeight.w500)),

            subtitle: Text('My City, CA 99984'),

            leading: Icon(

              Icons.restaurant\_menu,

              color: Colors.blue[500],

            ),

          ),

          Divider(),

          ListTile(

            title: Text('(408) 555-1212',

                style: TextStyle(fontWeight: FontWeight.w500)),

            leading: Icon(

              Icons.contact\_phone,

              color: Colors.blue[500],

            ),

          ),

          ListTile(

            title: Text('costa@example.com'),

            leading: Icon(

              Icons.contact\_mail,

              color: Colors.blue[500],

            ),

          ),

        ],

      ),

    ),

  );

# Adding interactivity to your Flutter app

#### **What you’ll learn**

* How to respond to taps.
* How to create a custom widget.
* The difference between stateless and stateful widgets.

### **Step 1: Decide which object manages the widget’s state**

### **tep 2: Subclass StatefulWidget**

### **Step 3: Subclass State**

void \_toggleFavorite() {

setState(() {

if (\_isFavorited) {

\_favoriteCount -= 1;

\_isFavorited = false;

} else {

\_favoriteCount += 1;

\_isFavorited = true;

}

});

}

# Adding assets and images

flutter:

assets:

- directory/

- directory/subdirectory/

# Navigation and routing

// Within the `FirstRoute` widget

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(builder: (context) => SecondRoute()),

);

}

## **Return to the first route using Navigator.pop()**

# Animations tutorial

# Fade a widget in and out

## **1. Create a box to fade in and out**

First, create something to fade in and out. For this example, draw a green box on screen.

*content\_copy*

Container(

width: 200.0,

height: 200.0,

color: Colors.green,

);

## **Define a StatefulWidget**

bool \_visible = true;

onPressed: () {

// Call setState. This tells Flutter to rebuild the

// UI with the changes.

setState(() {

\_visible = !\_visible;

});

},

## **4. Fade the box in and out**

AnimatedOpacity(

// If the widget is visible, animate to 0.0 (invisible).

// If the widget is hidden, animate to 1.0 (fully visible).

opacity: \_visible ? 1.0 : 0.0,

duration: Duration(milliseconds: 500),

// The green box must be a child of the AnimatedOpacity widget.

child: Container(

width: 200.0,

height: 200.0,

color: Colors.green,

),

);

# Hero animations

Hero(

                      tag: 'mytag',

                      child: Container(

                        height: 50,

                        width: 50,

                        color: Colors.cyan,

                      ),

                    ),

Hero(

                      tag: 'mytag',

                      child: Container(

                        height: 250,

                        width: 250,

                        color: Colors.cyan,

                      ),

        ),

Delay page transition

PageRouteBuilder(

                transitionDuration: Duration(seconds: 2),

                pageBuilder: (\_, \_\_, \_\_\_) => PhotoHero(),

              ),

# Using slivers to achieve fancy scrolling

    return Scaffold(

      body: NestedScrollView(

        headerSliverBuilder: (BuildContext context, bool innerBoxIsScrolled) {

          return <Widget>[

            SliverAppBar(

              expandedHeight: 200.0,

              floating: false,

              pinned: true,

              flexibleSpace: FlexibleSpaceBar(

                  centerTitle: true,

                  title: Text("Collapsing Toolbar",

                      style: TextStyle(

                        color: Colors.white,

                        fontSize: 16.0,

                      )),

                  background: Image.network(

                    "https://images.pexels.com/photos/396547/pexels-photo-396547.jpeg?auto=compress&cs=tinysrgb&h=350",

                    fit: BoxFit.cover,

                  )),

            ),

          ];

        },

        body: Center(

          child: Text("Sample Text"),

        ),

      ),

    );

  }