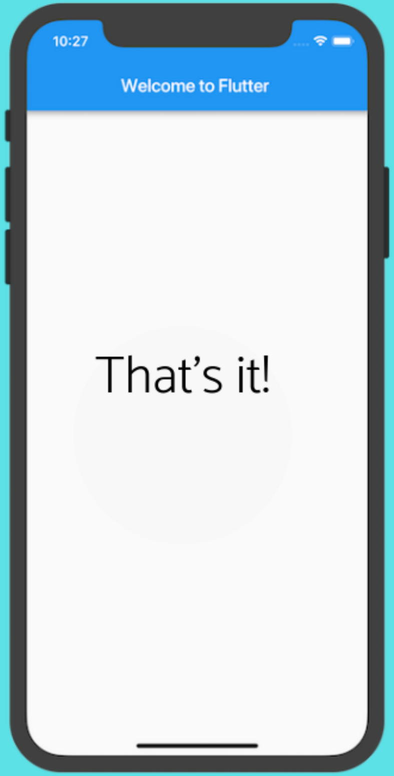
Flutter

Widgets

Flutter

Architecture of Flutter Application

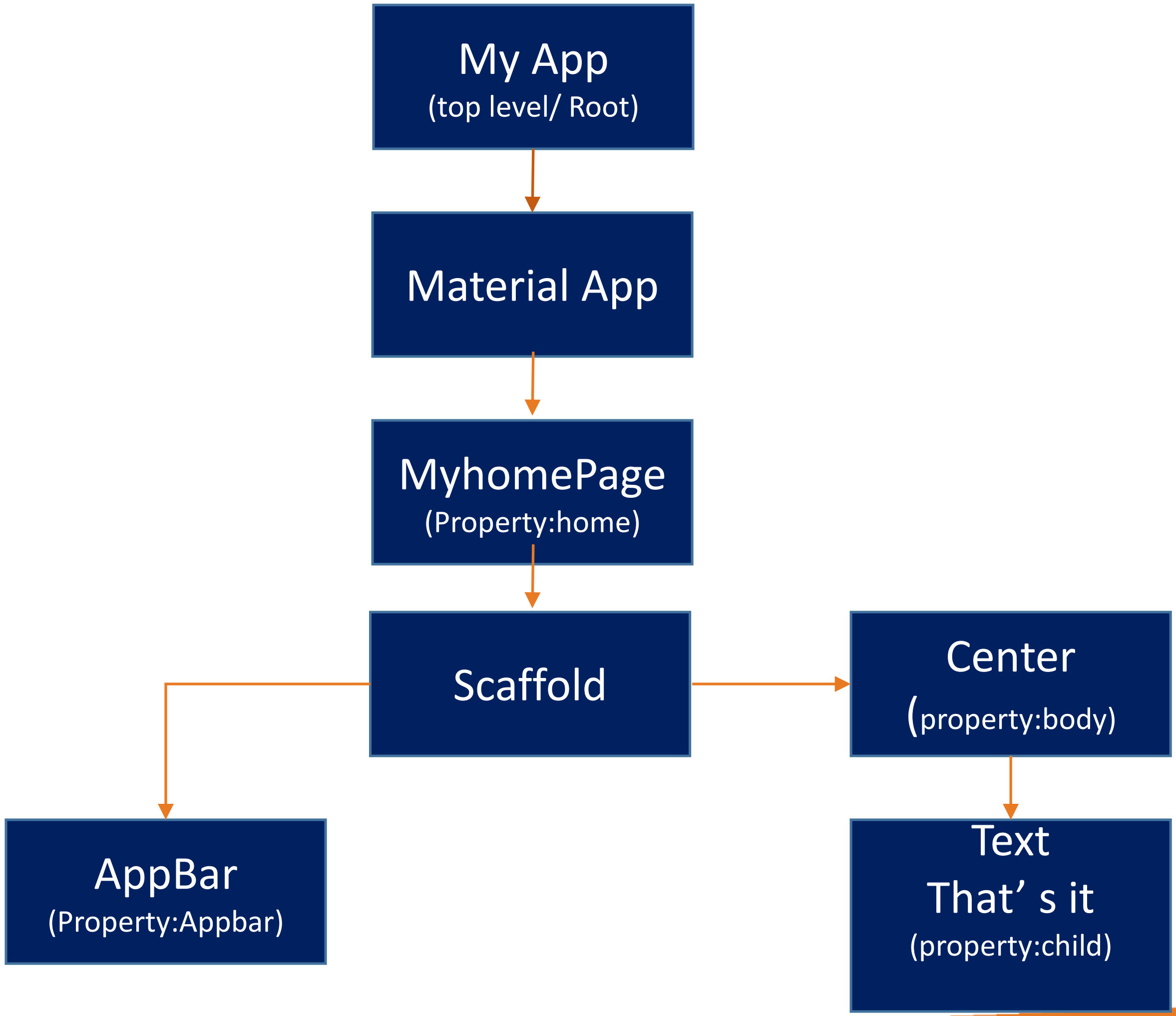


```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());


class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Welcome to Flutter'),
        ),
        body: Center(
          child: Text('That's it!'),
        ),
      ),
    );
  }
}
```

Widgets

- The core concept of the Flutter framework is In Flutter, Everything is a widget.
- Widgets are basically user interface components used to create the user interface of the application.
- In Flutter, the application is itself a widget. The application is the top- level widget and its UI is build using one or more children (widgets), which again build using its children widgets.
- This composability feature helps us to create a user interface of any complexity. For example, the widget hierarchy of the above example is as specified in the following diagram:

Widget Hierarchy



Widget Hierarchy in details

- **MyApp** is the user created widget and it is build using the Flutter native widget, MaterialApp.
- **MaterialApp** has a home property to specify the user interface of the home page, which is again a user created widget, MyHomePage.
- **MyHomePage** - is build using another flutter native widget, Scaffold.
- **Scaffold** has two properties – body and appBar.
- **body** is used to specify its main user interface and appBar is used to specify its header user interface.
- **Header UI** is build using flutter native widget, AppBar and Body UI is build using Center widget.
- The **Center** widget has a property, Child, which refers the actual content and it is build using Text widget.

Widgets..

In Flutter, widgets can be grouped into multiple categories based on their features, as listed below

- Platform specific widgets
- Layout widgets
- State maintenance widgets
- Platform independent / basic widgets

Platform specific widgets

- Android specific widgets are designed in accordance with Material design guideline by Android OS. Android specific widgets are called as **Material widgets**.
- iOS specific widgets are designed in accordance with Human Interface Guidelines by Apple and they are called as **Cupertino widgets**.

Platform specific widgets

- Scaffold , AppBar, BottomNavigationBar ,TabBar , TabBarView , ListTile , RaisedButton , FloatingActionButton , FlatButton, IconButton , DropdownButton, PopupMenuButton , ButtonBar, TextField, Checkbox ,Radio , Switch ,Slider , Date & Time Pickers , SimpleDialog , AlertDialog
- CupertinoButton , CupertinoPicker , CupertinoDatePicker , CupertinoTimerPicker , CupertinoNavigationBar, CupertinoTabBar , CupertinoTabScaffold , CupertinoTabView , CupertinoTextField , CupertinoDialog ,CupertinoDialogAction , CupertinoFullscreenDialogTransition , CupertinoPageScaffold , CupertinoPageTransition , CupertinoActionSheet, CupertinoActivityIndicator , CupertinoAlertDialog , CupertinoPopupSurface

Layout widgets

To compose multiple widgets into a single widget, Flutter provides large number of widgets with layout feature. For example, the child widget can be centered using Center widget.

Some of the popular layout widgets are as follows:

- Container: A rectangular box decorated using BoxDecoration widgets with background, border and shadow.
- Center: Center its child widget
- Row: Arrange its children in the horizontal direction.
- Column: Arrange its children in the vertical direction.
- Stack: Arrange one above the another.

State maintenance widgets

- The dynamic nature of the application is through interactive behavior of the widgets and the state changes during interaction

State maintenance widgets

The dynamic nature of the application is through interactive behavior of the widgets and the state changes during interaction

StatelessWidget

Only requires a single method build to be implemented in its derived class. The build method gets the context environment necessary to build the widgets through BuildContext parameter and returns the widget it builds

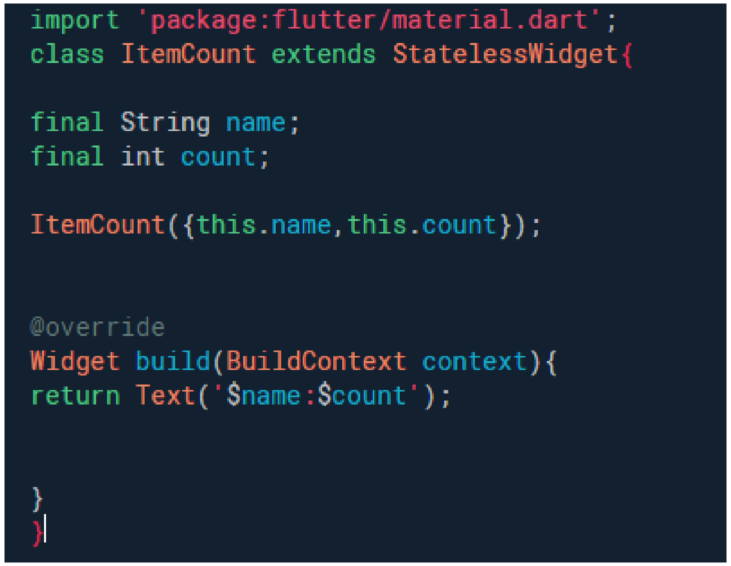
Icon, IconButton, and Text are examples of stateless widgets.

StatefullWidget

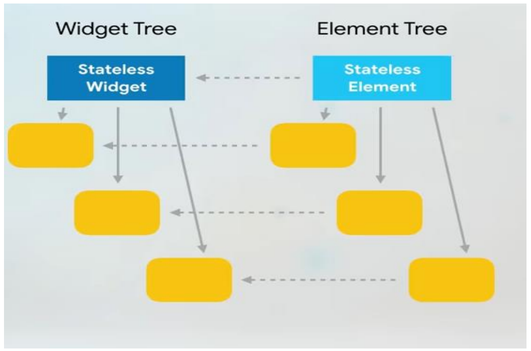
Stateful can be thought of as redering through user input that becomes redering according to the change of state. it can change its appearance in response to events triggered by user interactions or when it receives data

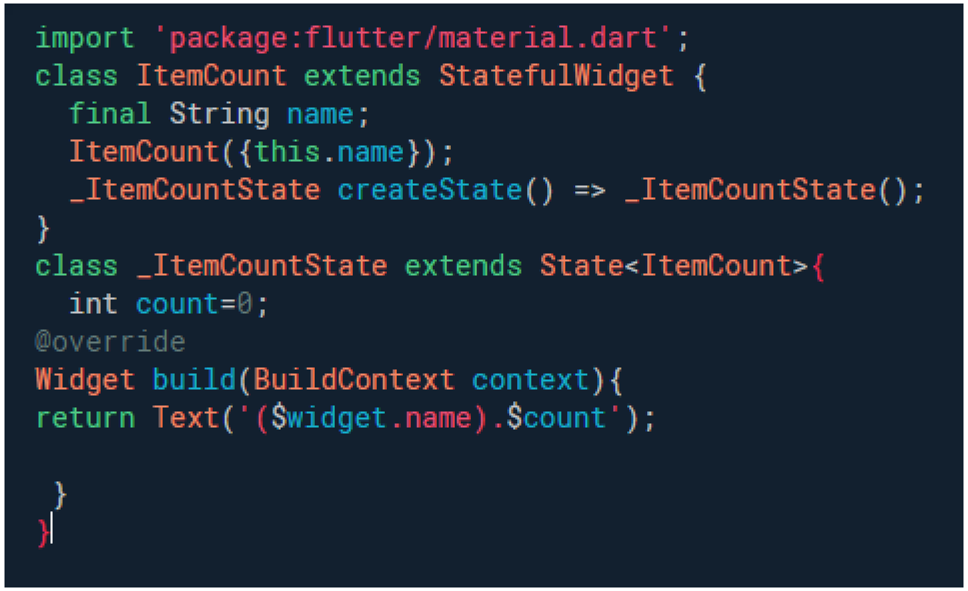
Checkbox, Radio, Slider, InkWell, Form, and TextField are examples of stateful widgets

States….?



```
import 'package:flutter/material.dart';
class ItemCount extends StatelessWidget{

final String name;
final int count;

ItemCount({this.name,this.count});


@override
Widget build(BuildContext context){
return Text('$name:$count');


}
}
```

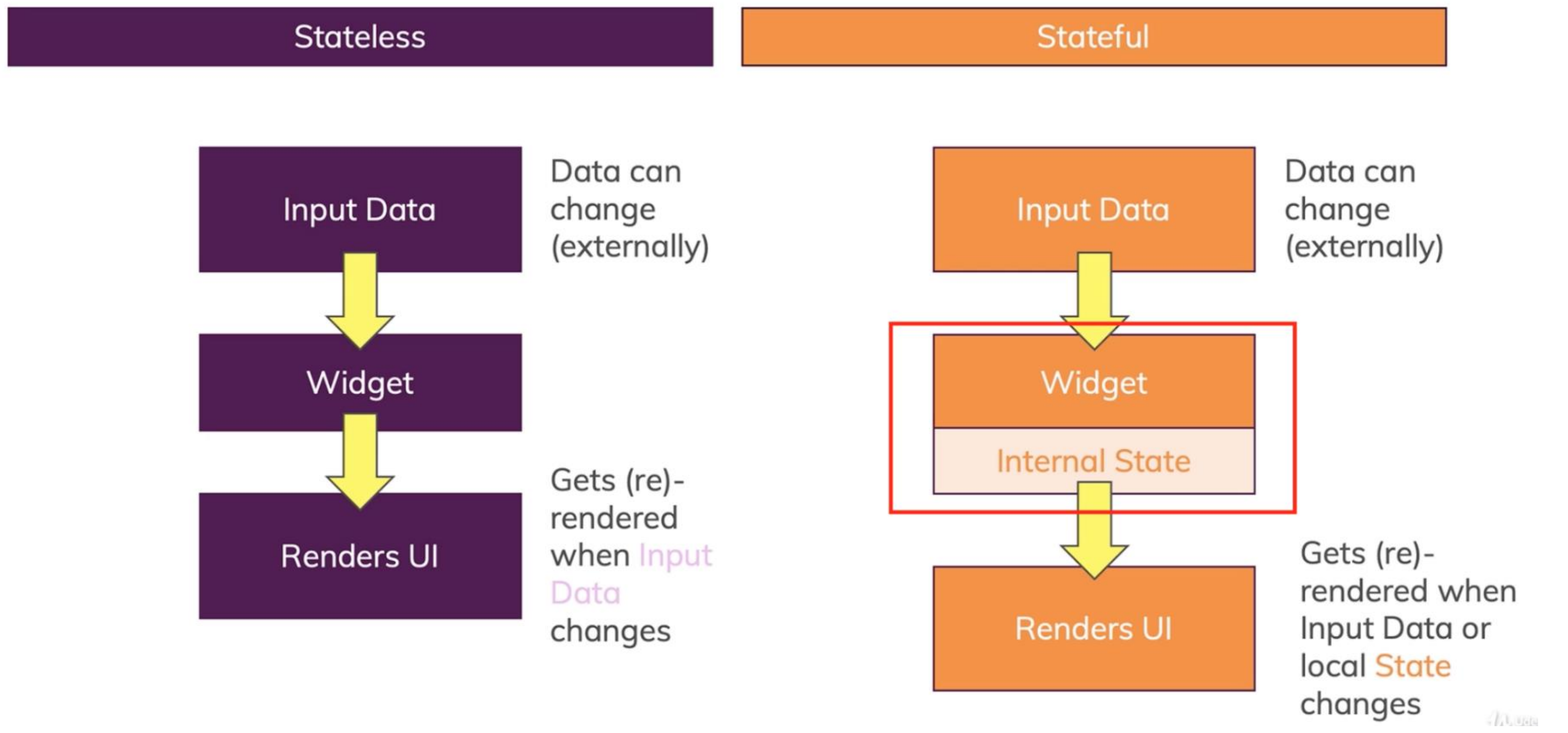




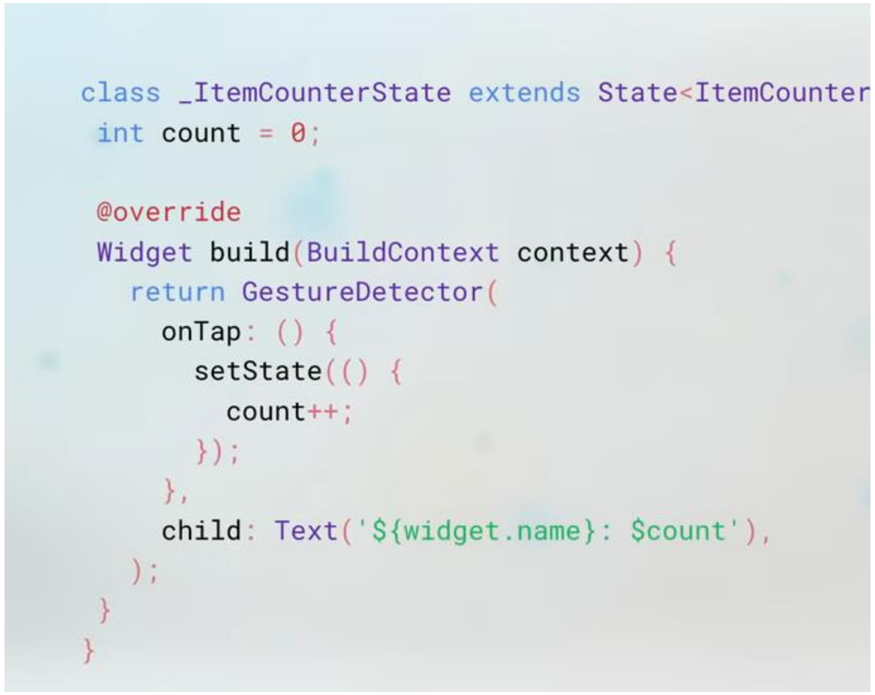
```
import 'package:flutter/material.dart';
class ItemCount extends StatefulWidget {
  final String name;
  ItemCount({this.name});
  _ItemCountState createState() => _ItemCountState();
}
class _ItemCountState extends State<ItemCount>{
  int count=0;
@override
Widget build(BuildContext context){
return Text('($widget.name).$count');

 }
}
```

Change the count ???

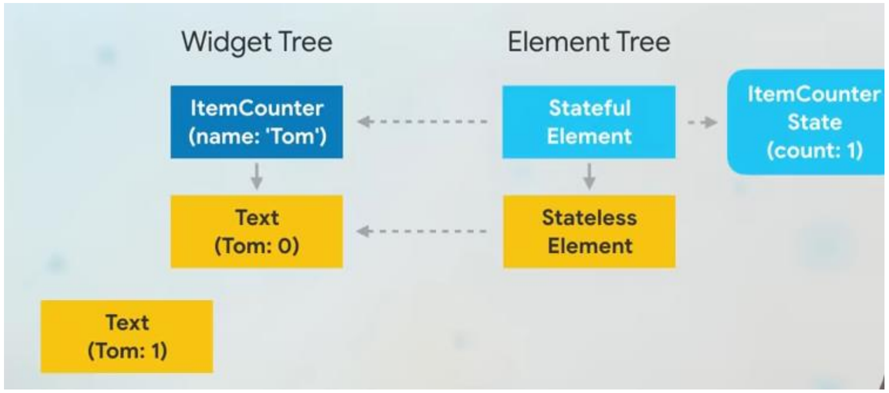
Stateless vs Stateful

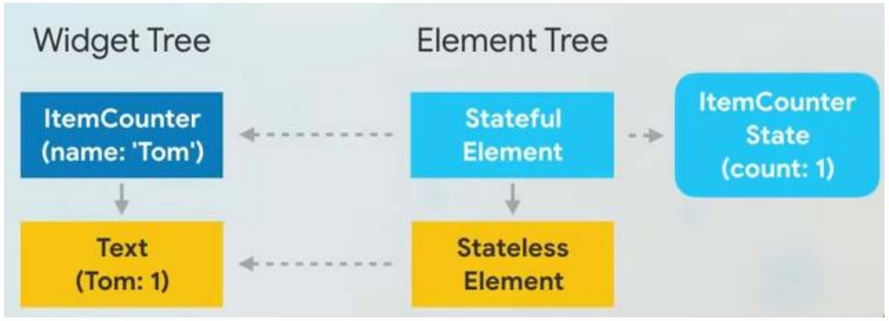

Stateless
Stateful
Input Data
Data can change (externally)
Widget
Renders UI
Gets (re)-rendered when Input Data changes
Input Data
Data can change (externally)
Widget
Internal State
Renders UI
Gets (re)-rendered when Input Data or local State changes

Statefull Widget



```
class _ItemCounterState extends State<ItemCounter
  int count = 0;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        setState(() {
          count++;
        });
      },
      child: Text('${widget.name}: $count'),
    );
  }
}
```

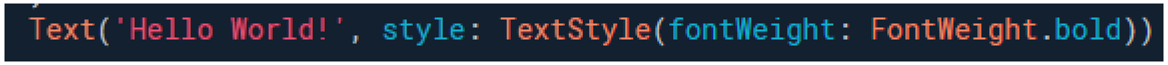




Platform independent / basic widgets

Text

Text widget is used to display a piece of string. The style of the string can be set by using **style property** and **TextStyle class**. The sample code for this purpose is as follows:



```
Text('Hello World!', style: TextStyle(fontWeight: FontWeight.bold))
```

- **maxLines, int:** Maximum number of lines to show
- **overflow, TextOverFlow:** Specify how visual overflow is handled using TextOverFlow class
- **style, TextStyle:** Specify the style of the string using TextStyle class
- **textAlign, TextAlign:** Alignment of the text like right, left, justify, etc., using TextAlign class
- **textDirection, TextDirection:** Direction of text to flow, either left-to-right or rightto-left

Platform independent / basic widgets

Image

Image widget provides different constructors to load images from multiple sources and they are as follows:

- Image - Generic image loader using ImageProvider
- Image.asset - Load image from flutter project's assets
- Image.file - Load image from system folder
- Image.memory - Load image from memory
- Image.Network - Load image from network

The easiest option to load and display an image in Flutter is by including the image as assets of the application and load it into the widget on demand.

Platform independent / basic widgets

- Create a folder, assets in the project folder and place the necessary images.
- Specify the assets in the pubspec.yaml as shown below:

```
flutter:
  assets:
    - assets/smiley.png
```

- Now, load and display the image in the application

```
Image.asset('assets/smiley.png')
```

The most important properties of the Image widget are as follows:

- image, ImageProvider: Actual image to load
- width, double - Width of the image
- height, double - Height of the image
- alignment, AlignmentGeometry - How to align the image within its bounds

Platform independent / basic widgets

Icon

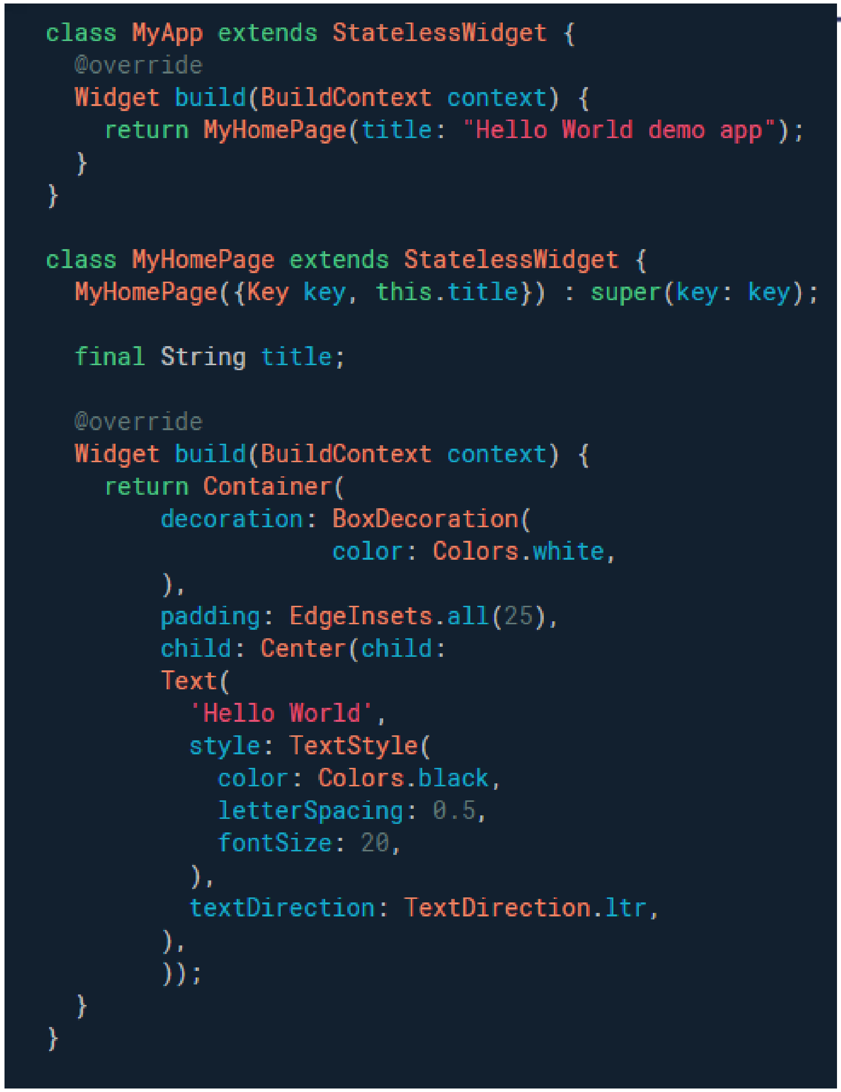
Icon widget is used to display a glyph from a font described in IconData class. The code to load a simple email icon is as follows:

Type of Layout Widgets

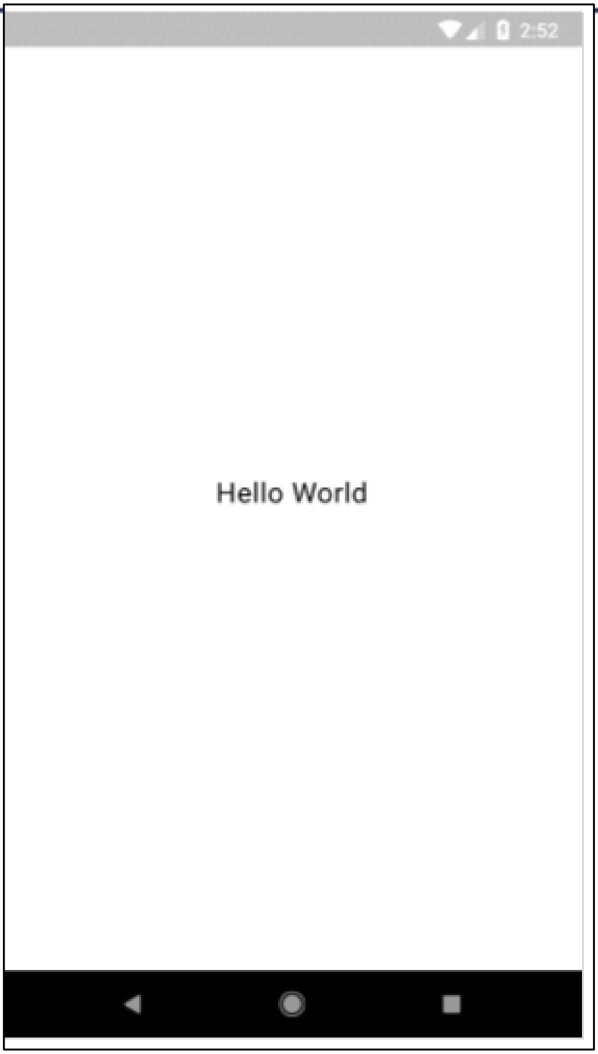
Widget supporting a single child

- Single Child Widgets In this category, widgets will have only one widget as its child and every widget will have a special layout functionality.
- For example, Center widget just centers it child widget with respect to its parent widget and Container widget provides complete flexibility to place it child at any given place inside it using different option like padding, decoration, etc.,
- Single child widgets are great options to create high quality widget having single functionality such as button, label, etc.,

Single Child Widget

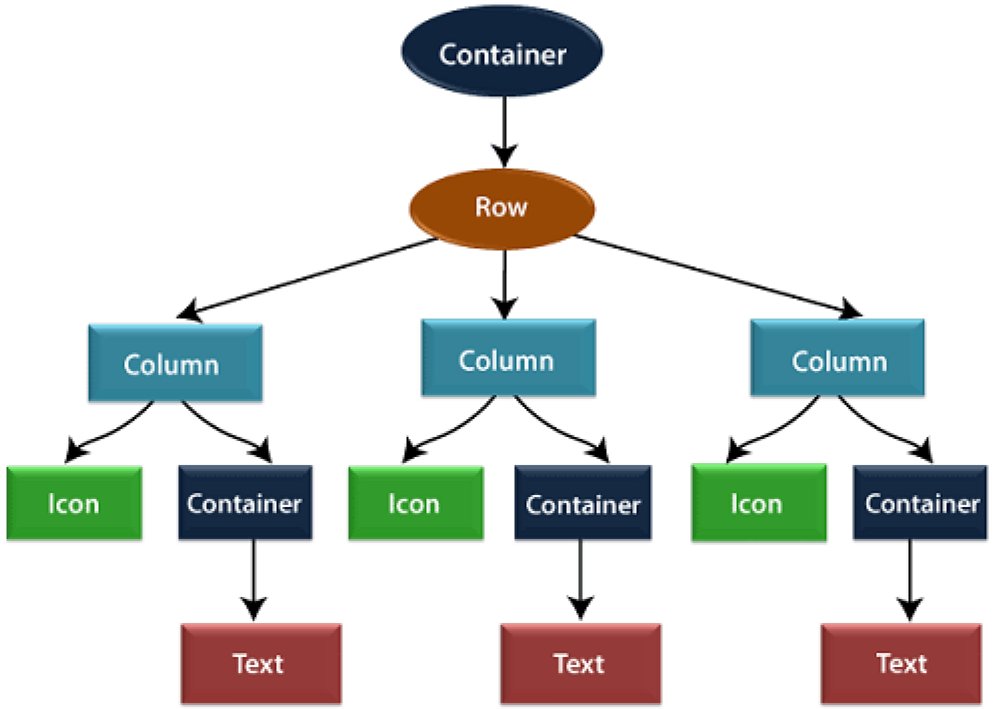


```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MyHomePage(title: "Hello World demo app");
  }
}

class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

  @override
  Widget build(BuildContext context) {
    return Container(
        decoration: BoxDecoration(
                    color: Colors.white,
        ),
        padding: EdgeInsets.all(25),
        child: Center(child:
        Text(
          'Hello World',
          style: TextStyle(
            color: Colors.black,
            letterSpacing: 0.5,
            fontSize: 20,
          ),
          textDirection: TextDirection.ltr,
        ),
        ));
  }
}
```



Multiple Child Widgets

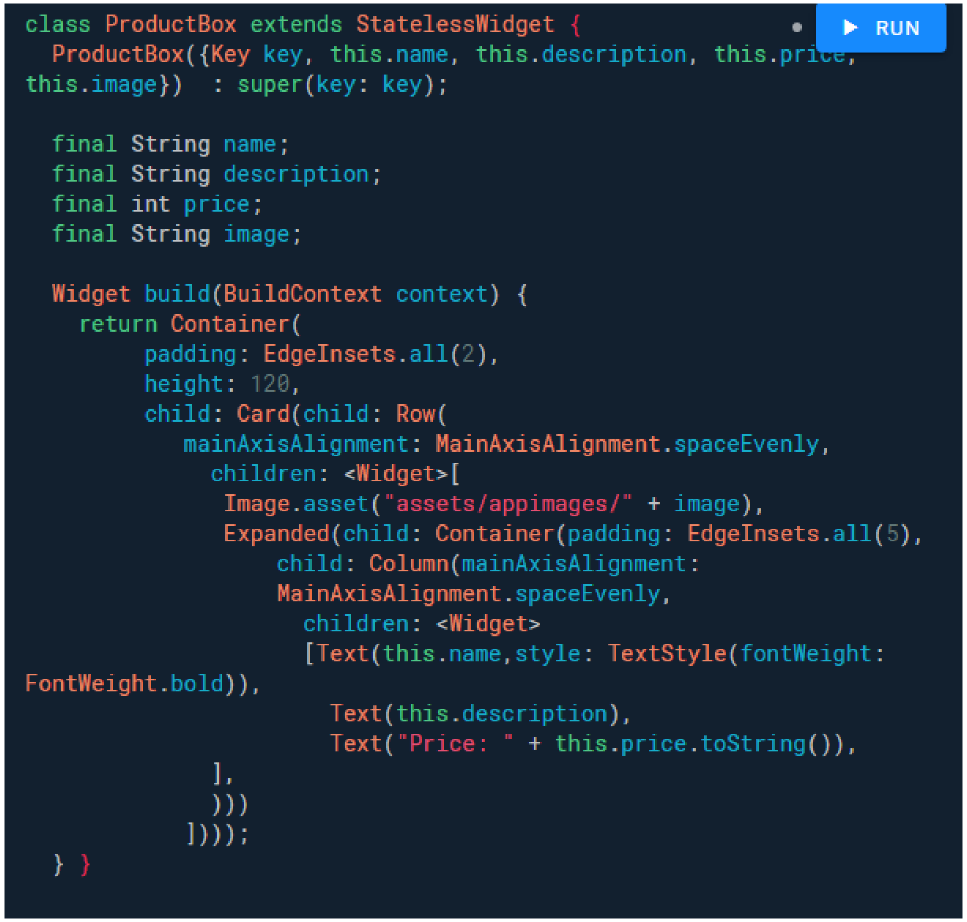
In this category, a given widget will have more than one child widgets and the layout of each widget is unique.



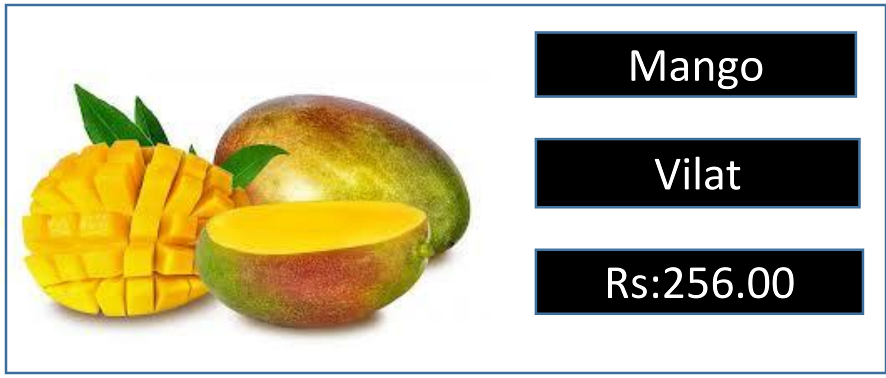
Multiple Child Widgets

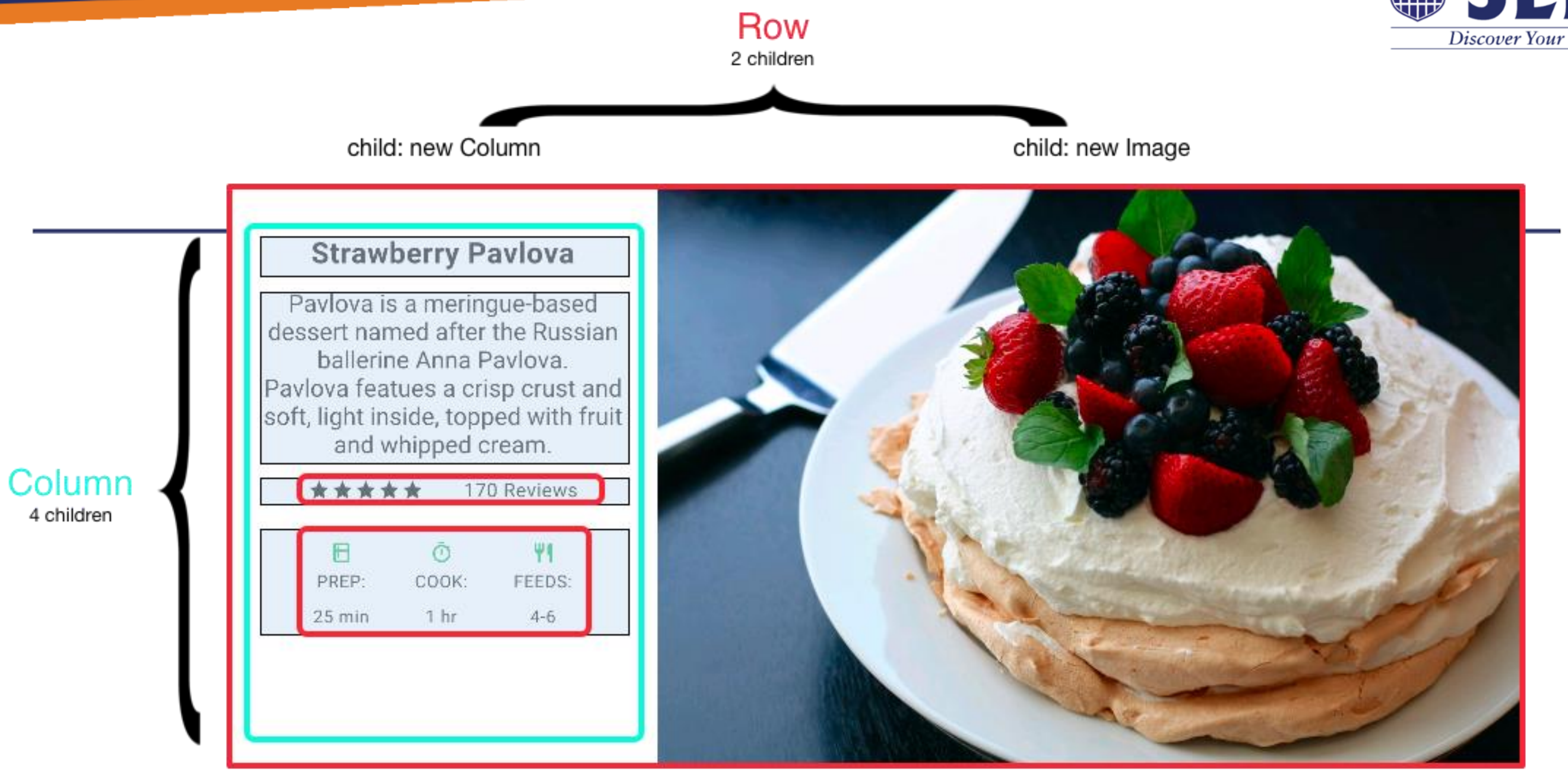
- Row - Allows to arrange its children in a horizontal manner.
- Column - Allows to arrange its children in a vertical manner.
- ListView - Allows to arrange its children as list.
- GridView - Allows to arrange its children as gallery.
- Expanded - Used to make the children of Row and Column widget to occupy the maximum possible area.

Multiple Child Widgets



```
class ProductBox extends StatelessWidget {
  ProductBox({Key key, this.name, this.description, this.price,
this.image})  : super(key: key);

  final String name;
  final String description;
  final int price;
  final String image;

  Widget build(BuildContext context) {
    return Container(
         padding: EdgeInsets.all(2),
         height: 120,
         child: Card(child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: <Widget>[
               Image.asset("assets/appimages/" + image),
               Expanded(child: Container(padding: EdgeInsets.all(5),
                   child: Column(mainAxisAlignment:
                   MainAxisAlignment.spaceEvenly,
                     children: <Widget>
                     [Text(this.name,style: TextStyle(fontWeight:
FontWeight.bold)),
                       Text(this.description),
                       Text("Price: " + this.price.toString()),
              ],
              )))
            ])));
  } }
```




Row
2 children
child: new Column
child: new Image
Column
4 children
Strawberry Pavlova
Pavlova is a meringue-based dessert named after the Russian ballerine Anna Pavlova. Pavlova featues a crisp crust and soft, light inside, topped with fruit and whipped cream.
★★★★★ 170 Reviews
PREP: 25 min
COOK: 1 hr
FEEDS: 4-6

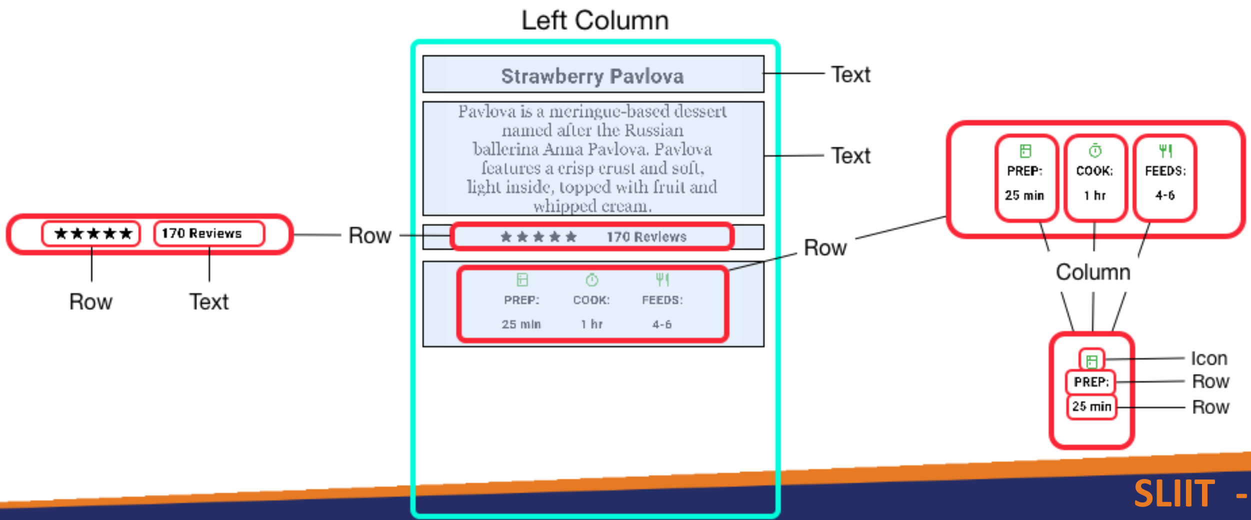
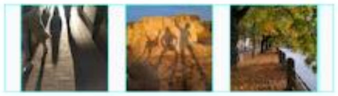

Left Column
Strawberry Pavlova
Text
Pavlova is a meringue-based dessert named after the Russian ballerina Anna Pavlova. Pavlova features a crisp crust and soft, light inside, topped with fruit and whipped cream.
Text
★★★★★ 170 Reviews
Row
Row
Text
Row
PREP: 25 min
COOK: 1 hr
FEEDS: 4-6
Column
Icon
Row
Row

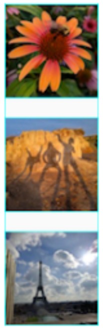
Image rows

```
Row (
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```

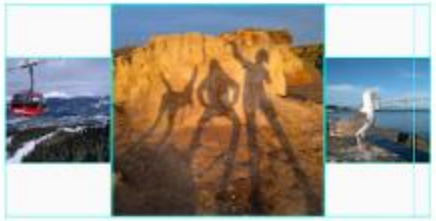


App source: row_column

```
Column (
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```



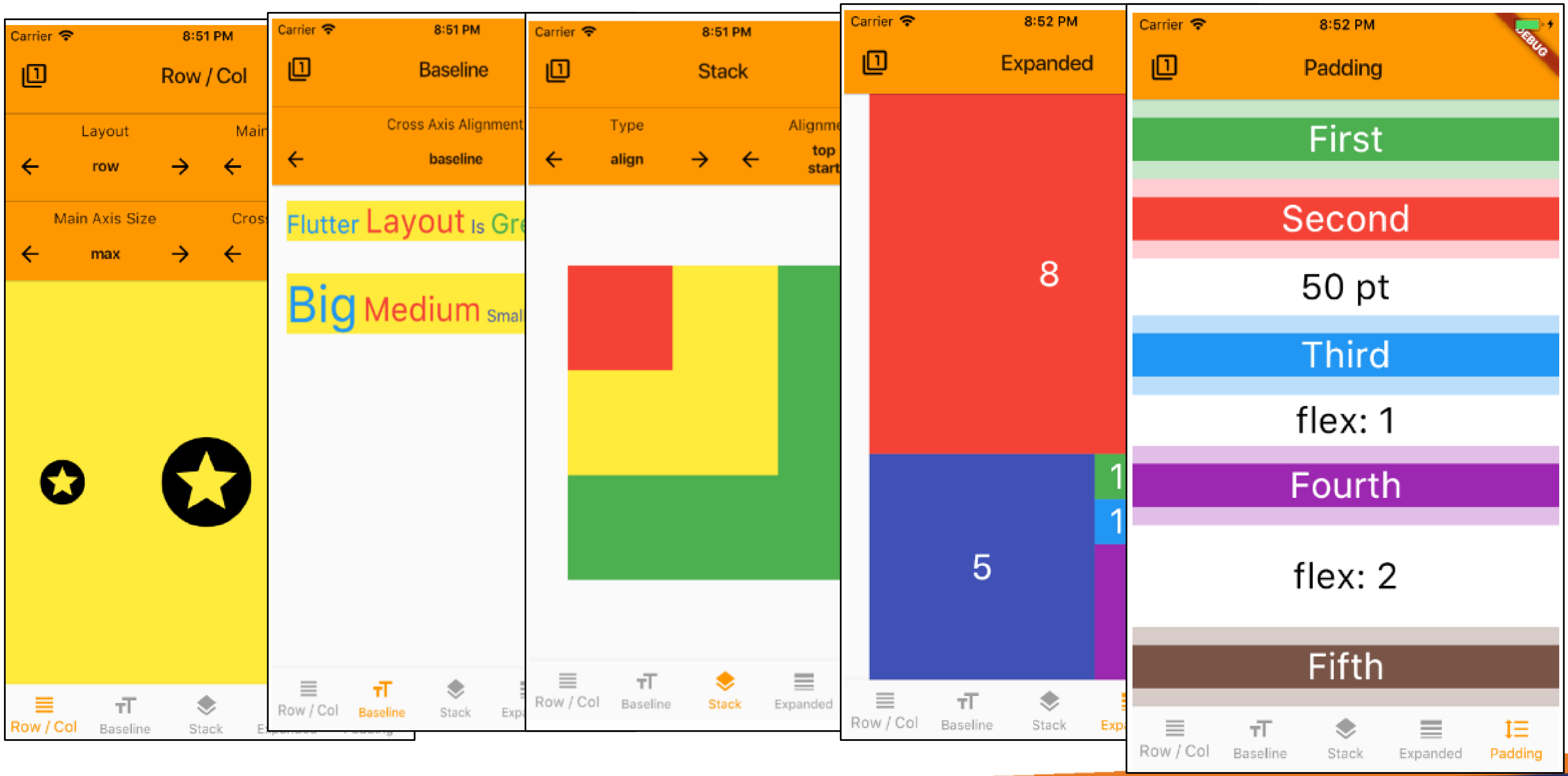
```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      flex: 2,
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```



App source: sizing

Layout widgets

- https://github.com/bizz84/layout-demo-flutter



Common layout widgets

standard widgets from the widgets library, and specialized widgets from the Material library. Any app can use the widgets library but only Material apps can use the Material Components library.

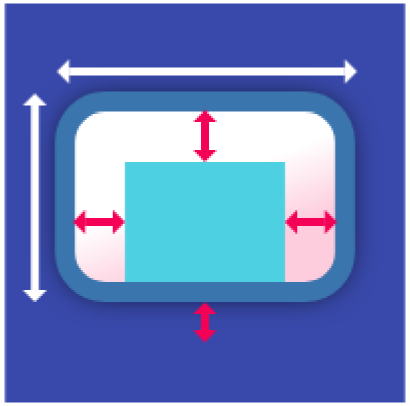
Standard widgets

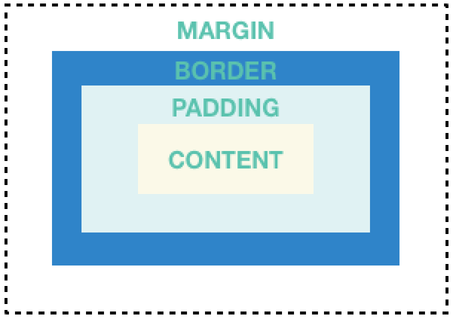
- Container: Adds padding, margins, borders, background color, or other decorations to a widget.
- GridView: Lays widgets out as a scrollable grid.
- ListView: Lays widgets out as a scrollable list.
- Stack: Overlaps a widget on top of another.

Material widgets

- Card: Organizes related info into a box with rounded corners and a drop shadow.
- ListTile: Organizes up to 3 lines of text, and optional leading and trailing icons, into a row.
-

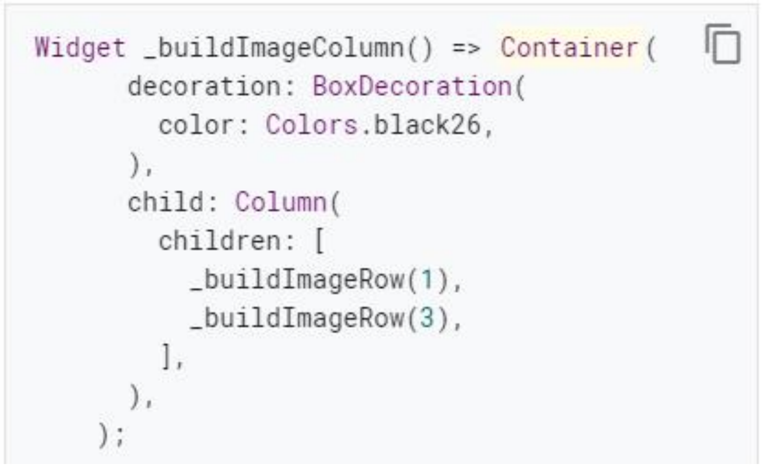
Container



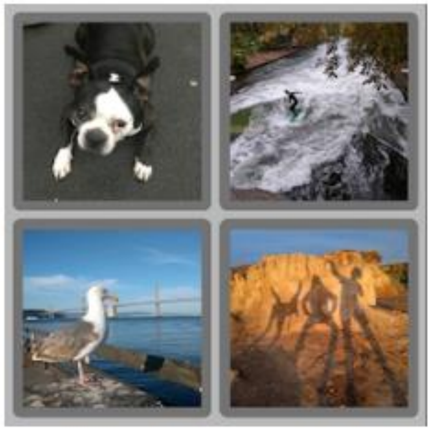


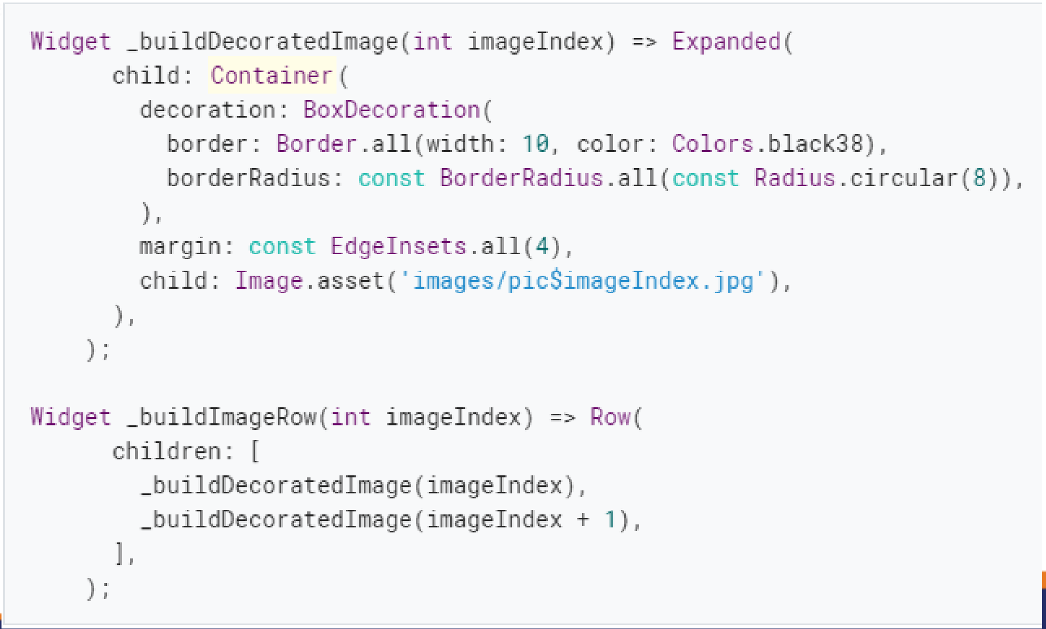
- A convenience widget that combines common painting, positioning, and sizing widgets.
- Containers with no children try to be as big as possible unless the incoming constraints are unbounded
- Containers with children size them self to children
- Container can have one child

Container



```
Widget _buildImageColumn() => Container(
      decoration: BoxDecoration(
        color: Colors.black26,
      ),
      child: Column(
        children: [
          _buildImageRow(1),
          _buildImageRow(3),
        ],
      ),
    );
```

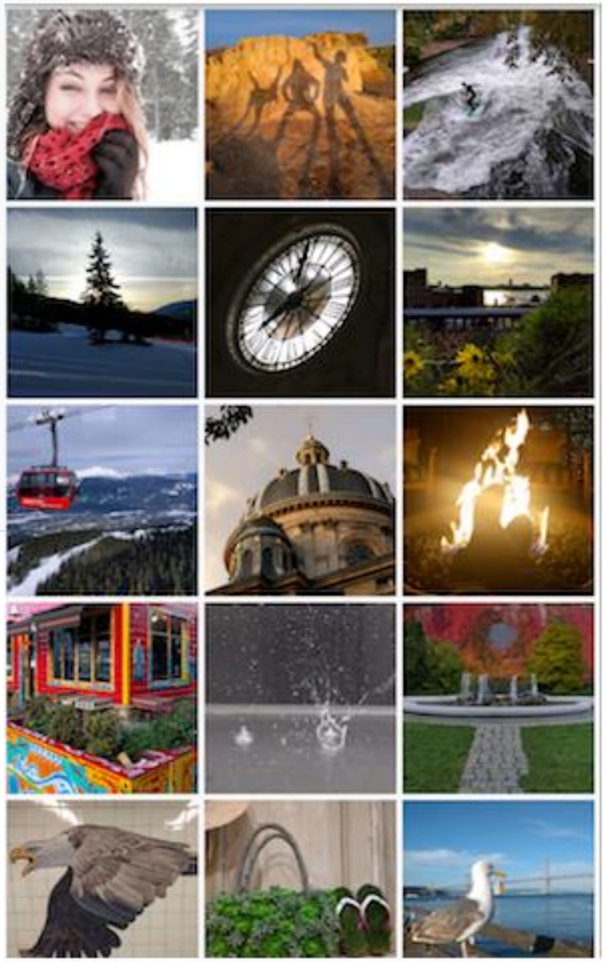




```
Widget _buildDecoratedImage(int imageIndex) => Expanded(
      child: Container(
        decoration: BoxDecoration(
          border: Border.all(width: 10, color: Colors.black38),
          borderRadius: const BorderRadius.all(const Radius.circular(8)),
        ),
        margin: const EdgeInsets.all(4),
        child: Image.asset('images/pic$imageIndex.jpg'),
      ),
    );

Widget _buildImageRow(int imageIndex) => Row(
      children: [
        _buildDecoratedImage(imageIndex),
        _buildDecoratedImage(imageIndex + 1),
      ],
    );
```

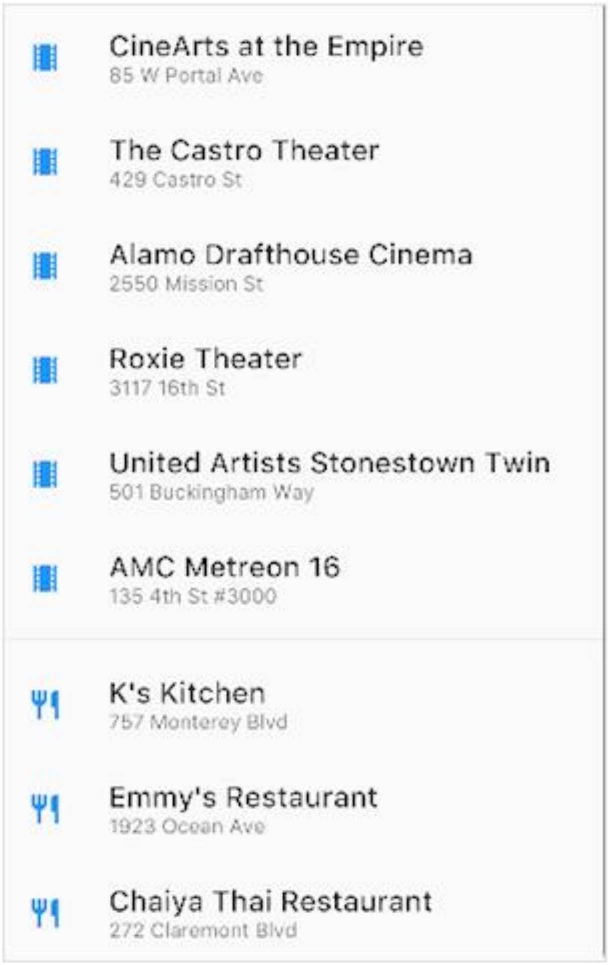
Grid View

```
Widget _buildGrid() => GridView.extent(
    maxCrossAxisExtent: 150,
    padding: const EdgeInsets.all(4),
    mainAxisSpacing: 4,
    crossAxisSpacing: 4,
    children: _buildGridTileList(30));

// The images are saved with names pic0.jpg, pic1.jpg...pic29.jpg.
// The List.generate() constructor allows an easy way to create
// a list when objects have a predictable naming pattern.
List<Container> _buildGridTileList(int count) => List.generate(
    count, (i) => Container(child: Image.asset('images/pic$i.jpg')));
```



ListView

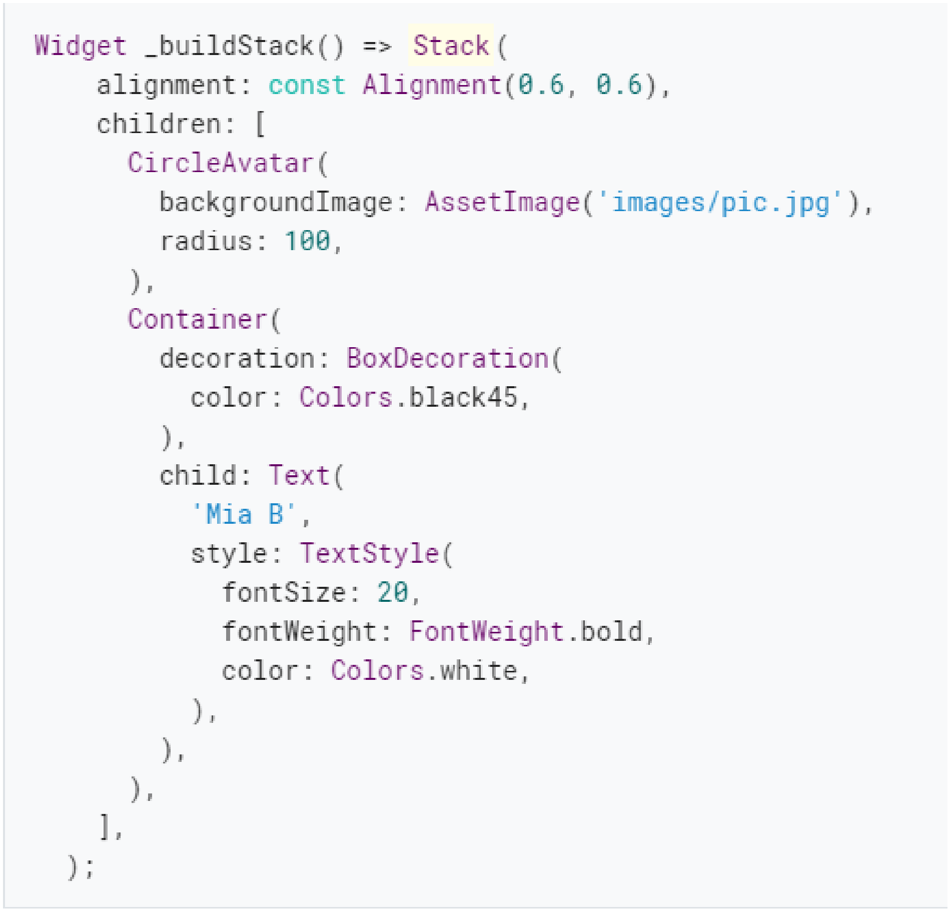
```
Widget _buildList() => ListView(
      children: [
        _tile('CineArts at the Empire', '85 W Portal Ave', Icons.theaters),
        _tile('The Castro Theater', '429 Castro St', Icons.theaters),
        _tile('Alamo Drafthouse Cinema', '2550 Mission St', Icons.theaters),
        _tile('Roxie Theater', '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('Kescaped_code#39;s Kitchen', '757 Monterey Blvd', Icons.restaurant),
        _tile('Emmyescaped_code#39;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],
      ),
    );
```


CineArts at the Empire
85 W Portal Ave
The Castro Theater
429 Castro St
Alamo Drafthouse Cinema
2550 Mission St
Roxie Theater
3117 16th St
United Artists Stonestown Twin
501 Buckingham Way
AMC Metreon 16
135 4th St #3000
K's Kitchen
757 Monterey Blvd
Emmy's Restaurant
1923 Ocean Ave
Chaiya Thai Restaurant
272 Claremont Blvd

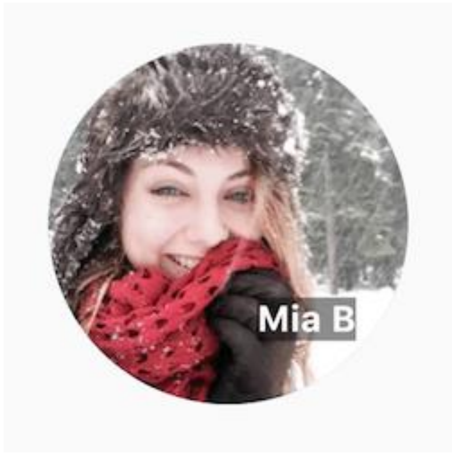
Stack

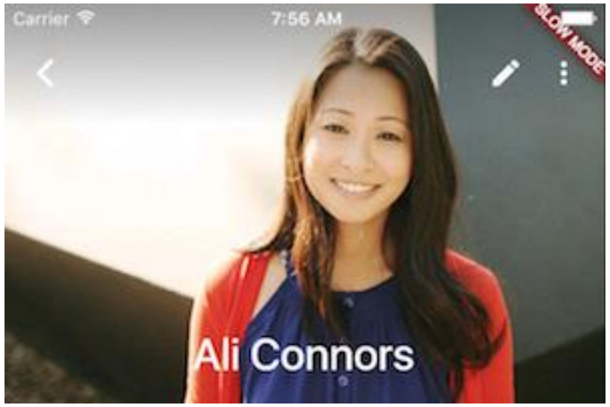
- Use for widgets that overlap another widget
- The first widget in the list of children is the base widget; subsequent children are overlaid on top of that base widget
- A Stack's content can't scroll
- You can choose to clip children that exceed the render box

Stack



```
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,
          ),
        ),
      ),
    ],
  );
```



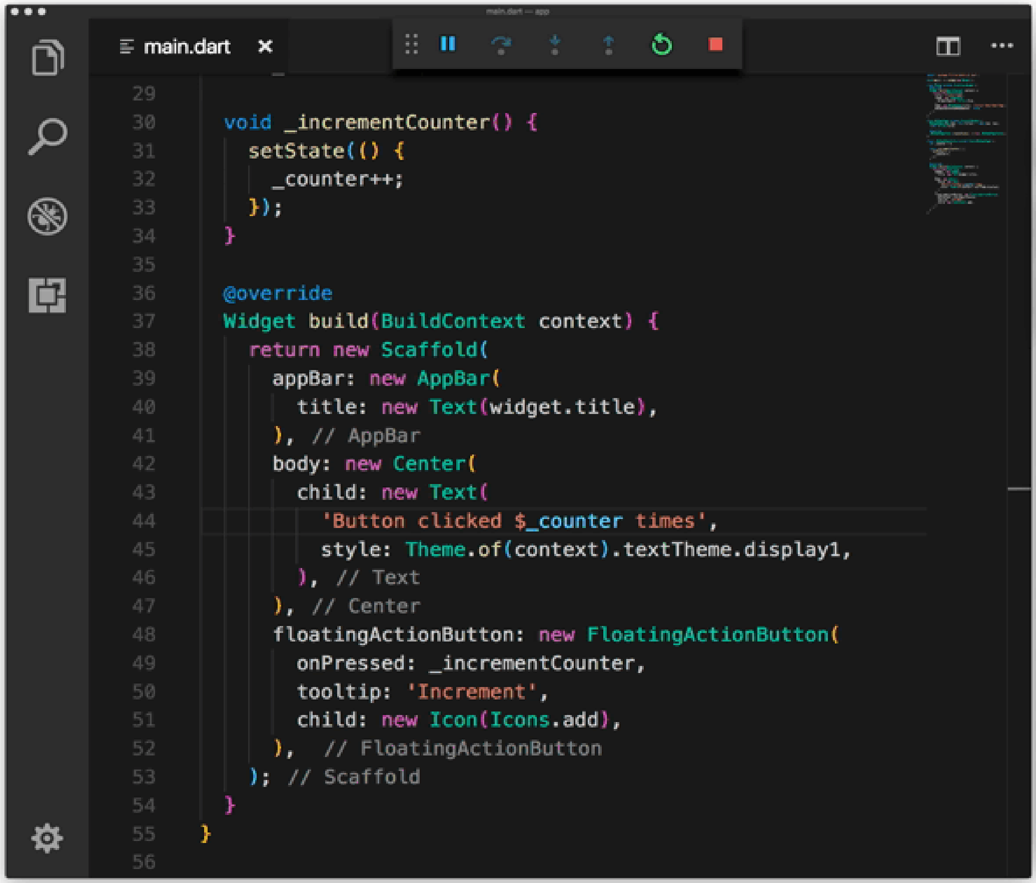


Hot Reload

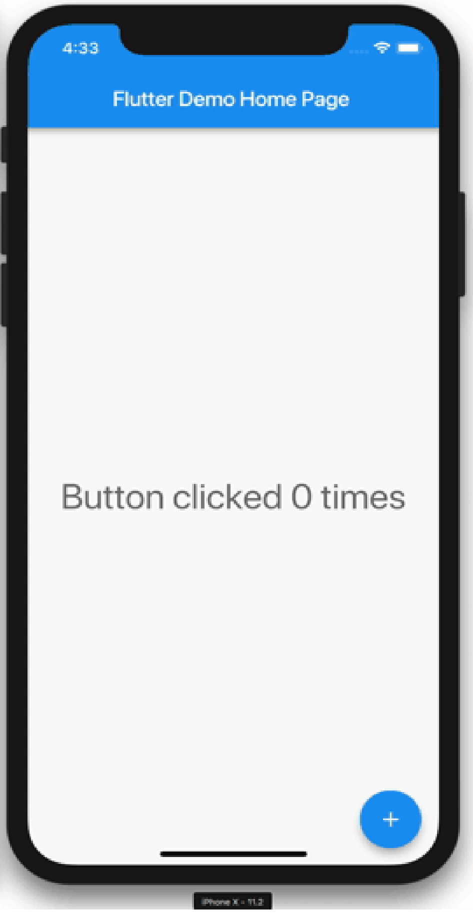
Hot reload

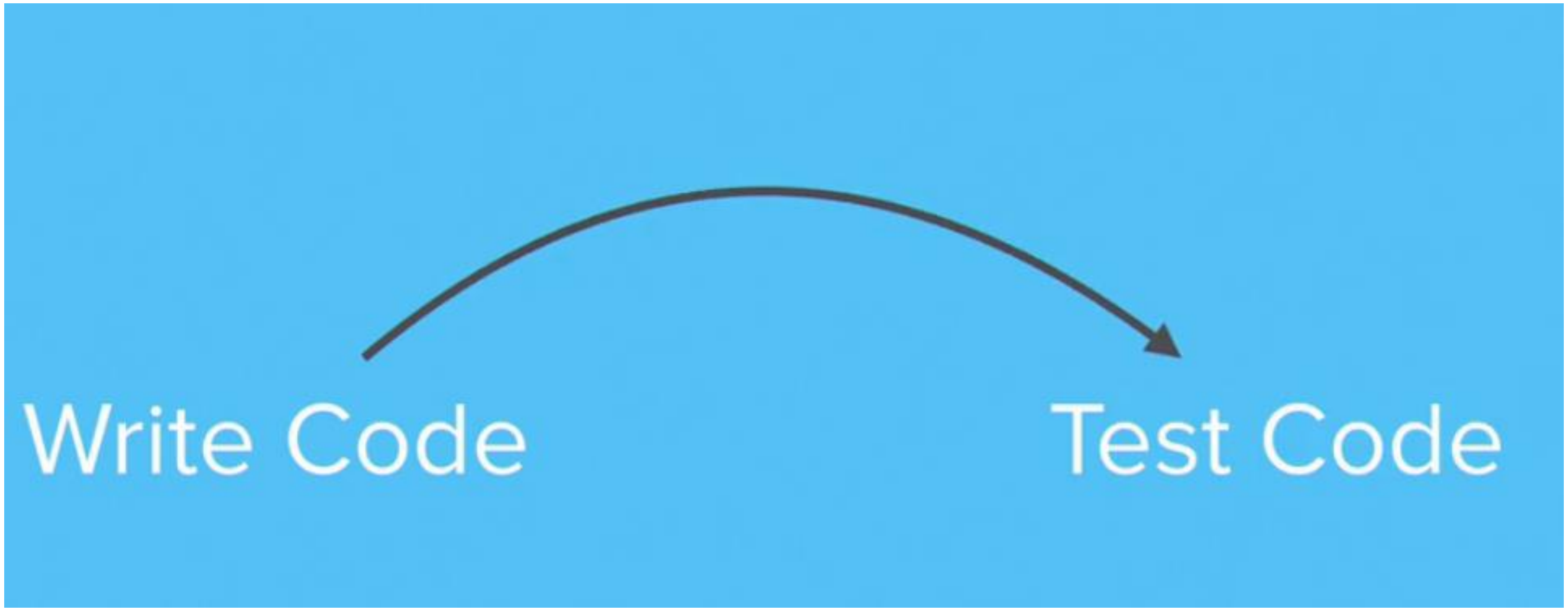
- Hot reload feature quickly compile the newly added code in our file and sent the code to Dart Virtual Machine.
- After done updating the Code Dart Virtual Machine update the app UI with widgets
- If you are using States in your application then Hot Reload preservers the States so they will not update on Hot Reload.
- Massively reduce the time from each development cycle.

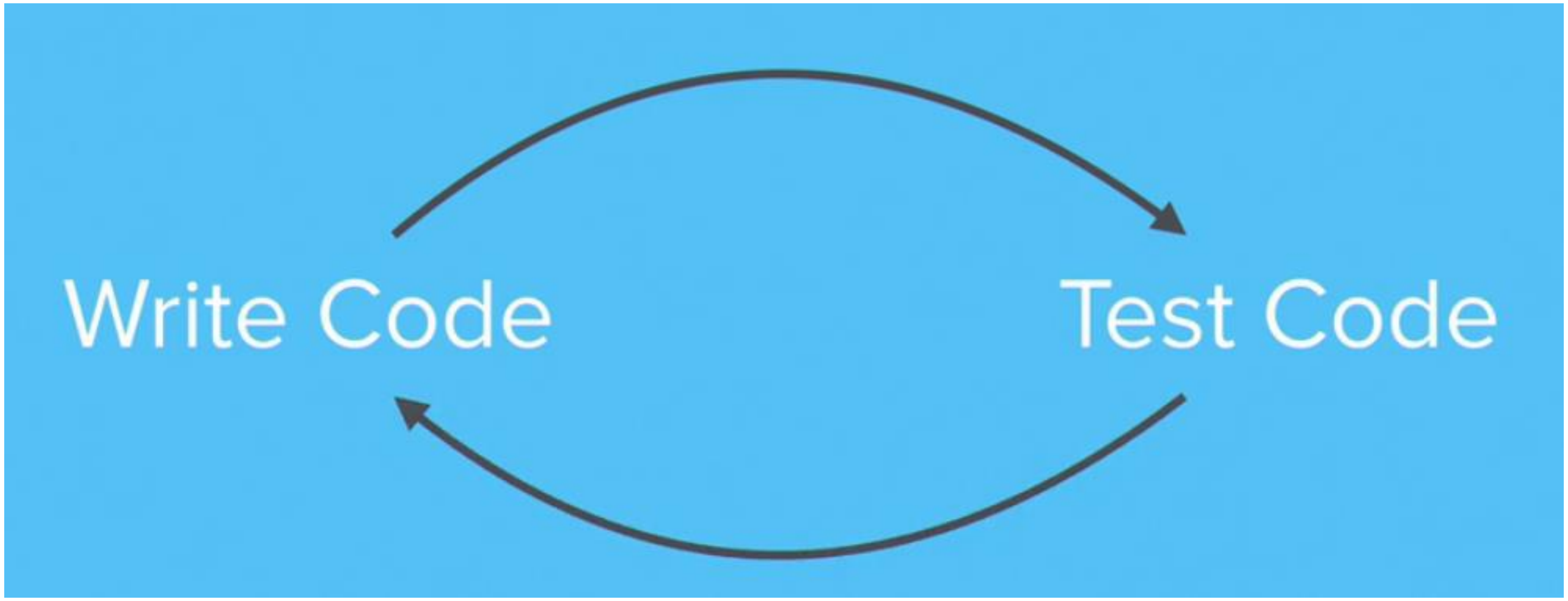
Hot reload



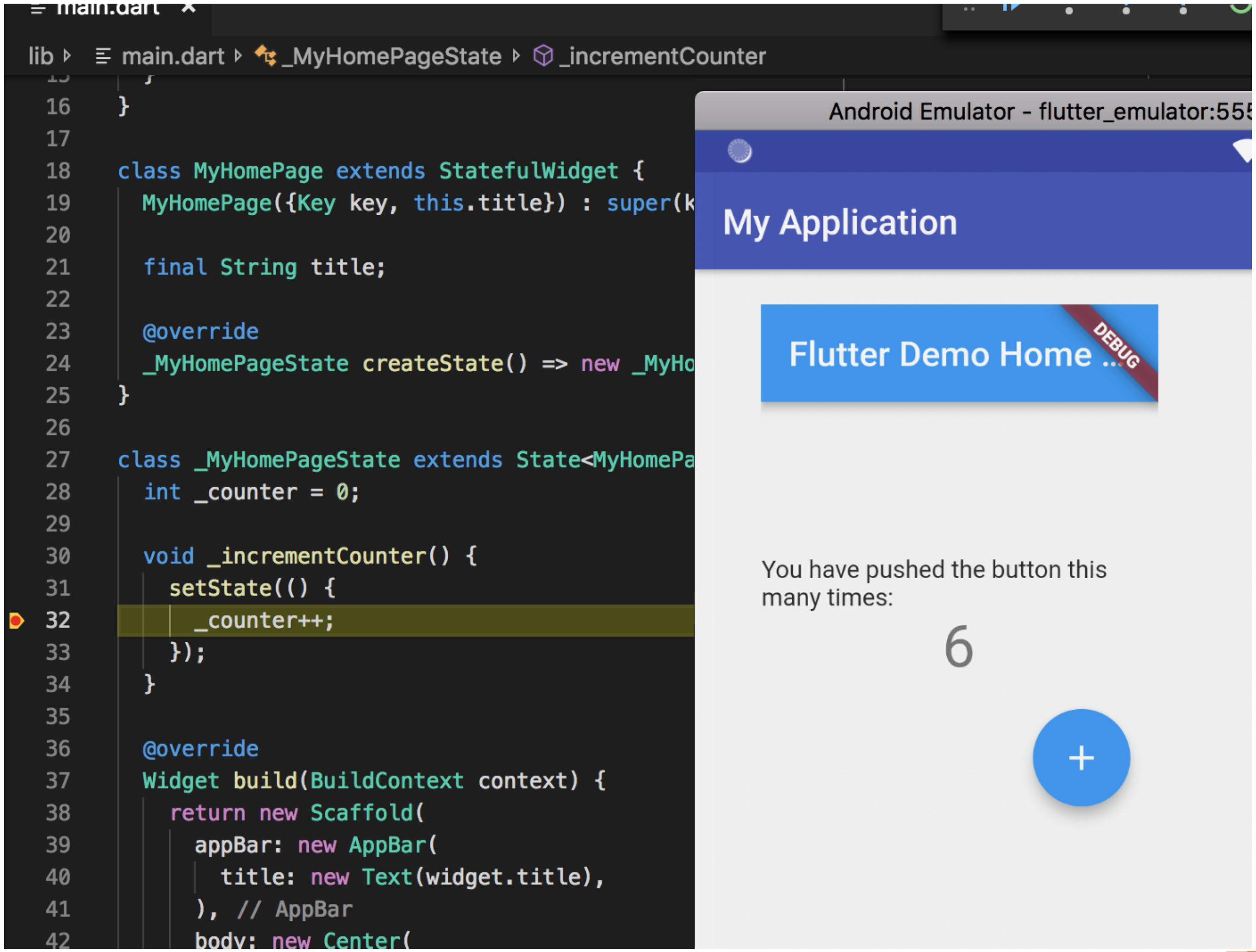
```
  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      appBar: new AppBar(
        title: new Text(widget.title),
      ), // AppBar
      body: new Center(
        child: new Text(
          'Button clicked $_counter times',
          style: Theme.of(context).textTheme.display1,
        ), // Text
      ), // Center
      floatingActionButton: new FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: new Icon(Icons.add),
      ),  // FloatingActionButton
    ); // Scaffold
  }
}
```




Write Code
Test Code


Write Code
Test Code

Hot Restart



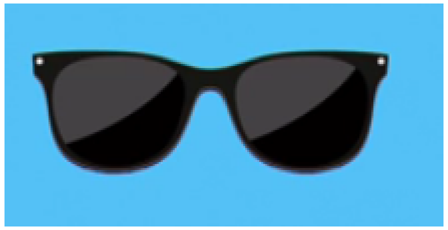
main.dart ×

lib › main.dart › _MyHomePageState › _incrementCounter

```
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(k

  final String title;

  @override
  _MyHomePageState createState() => new _MyHo
}

class _MyHomePageState extends State<MyHomePa
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      appBar: new AppBar(
        title: new Text(widget.title),
      ), // AppBar
      body: new Center(
```

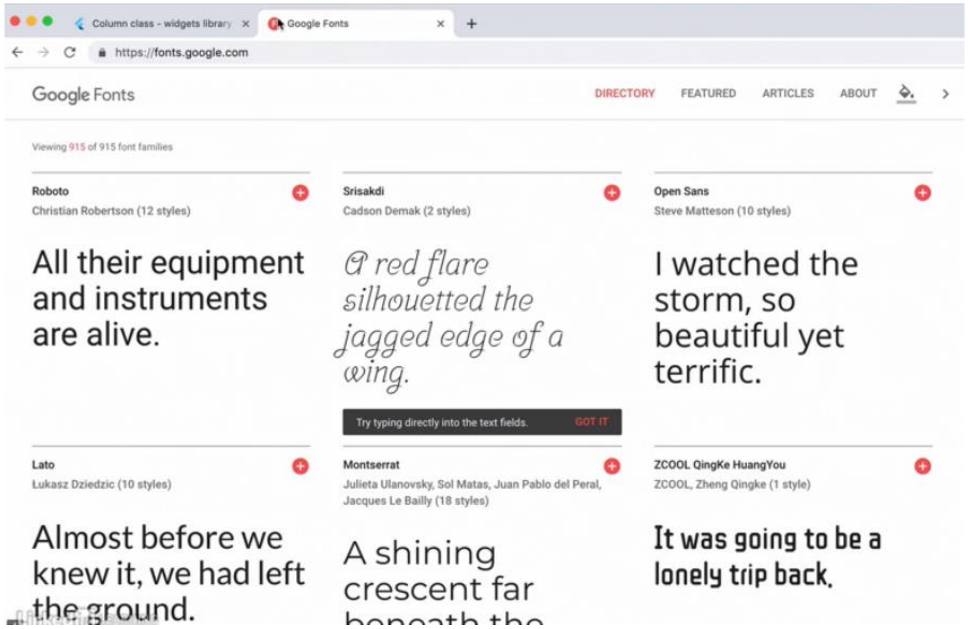
Android Emulator - flutter_emulator:555
My Application
Flutter Demo Home ...
DEBUG
You have pushed the button this many times:
6
+

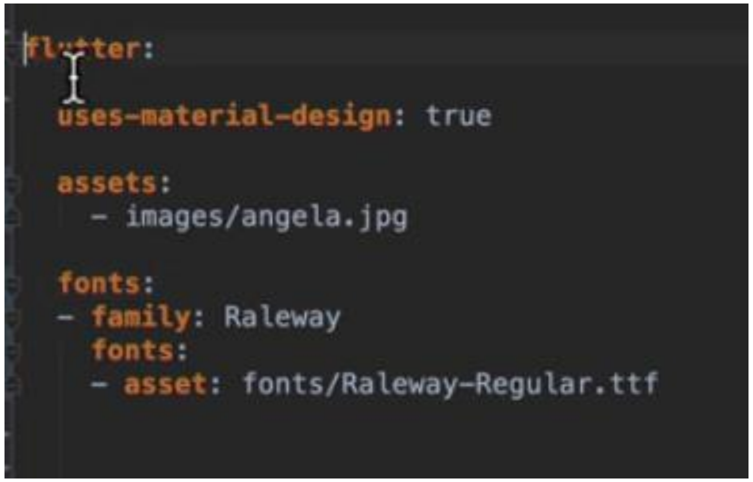
Hot Reload and Hot Restart ??



Size Doesn’t Matter

Custom Font ..

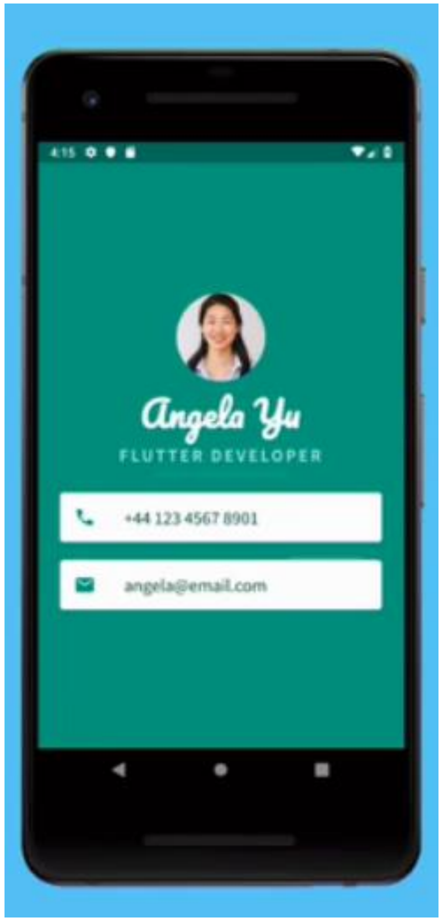

Column class - widgets library
Google Fonts
https://fonts.google.com
Google Fonts
DIRECTORY
FEATURED
ARTICLES
ABOUT
Viewing 915 of 915 font families
Roboto
Christian Robertson (12 styles)
All their equipment and instruments are alive.
Srisakdi
Cadson Demak (2 styles)
A red flare silhouetted the jagged edge of a wing.
Open Sans
Steve Matteson (10 styles)
I watched the storm, so beautiful yet terrific.
Try typing directly into the text fields.
GOT IT
Lato
Łukasz Dziedzic (10 styles)
Almost before we knew it, we had left the ground.
Montserrat
Julieta Ulanovsky, Sol Matas, Juan Pablo del Peral, Jacques Le Bailly (18 styles)
A shining crescent far
ZCOOL QingKe HuangYou
ZCOOL, Zheng Qingke (1 style)
It was going to be a lonely trip back.



```
flutter:

  uses-material-design: true

  assets:
    - images/angela.jpg

  fonts:
  - family: Raleway
    fonts:
    - asset: fonts/Raleway-Regular.ttf
```

- Download the font you want
- Create directory in your project “Fonts”
- Open pubspec.html
- **Indentation** is very important

Lets Create MiCard


Angela Yu
FLUTTER DEVELOPER
+44 123 4567 8901
angela@email.com