Chapter 5:
Flutter UI building

Plan

1. **Exploring Flutter basic application.**
   - Stateless app.
   - Statefull app.
   - BuildContext.
2. **Graphical widgets.**
   - Layout widgets.
   - Display widgets.
   - Interactive widgets.
   - Scrolling widgets: ListView & GridView.

Plan

1. **Exploring Flutter basic application.**
   - Stateless app.
   - Statefull app.
   - BuildContext.
2. Graphical widgets.
   - Layout widgets.
   - Display widgets.
   - Interactive widgets.
   - Scrolling widgets: ListView & GridView.

Flutter basic app

- **General structure of a *Stateless app*:**

```
import 'package:flutter/material.dart';  // Set of Material design widgets.
void main() => runApp (MyApp()); // App's entry point
/* Root widget of the  apps */
class MyApp extends StatelessWidget {
  @override
  Widget build (BuildContext context) { // Describes the UI represented by this widget.
   return MaterialApp ( // Top-level Material Design widget, which wraps all Material widgets.
              title: ' My application ', //  Application title.
              theme: ThemeData(primarySwatch: Colors.blue),
              home: BasicPage () , // The interface displayed when the app starts.
              );
 } }

class BasicPage extends StatelessWidget {// The interface is stateless.
  @override
  Widget build (BuildContext context) {
    return Text( 'Hello World! '); //  Static text displayed on the screen.
}
```

Flutter basic app

- **General structure of a *Stateles***

build method has a BuildContext parameter that is provided by the framework and will become the BuildContext for the widget that is returned by the build method.

```
import 'package:flutter/material.dart';  // Set of Material desig
void main() => runApp (MyApp()); // App's entry point
/* Root widget of the  apps */
class MyApp extends StatelessWidget {
  @override
  Widget build (BuildContext context) { // Describes the UI represented by this widget.
   return MaterialApp ( // Top-level Material Design widget, which wraps all Material widgets.
              title: ' My application ', //  Application title.
              theme: ThemeData(primarySwatch: Colors.blue),
              home: BasicPage () , // The interface displayed when the app starts.
              );
 } }

class BasicPage extends StatelessWidget {// The interface is stateless.
  @override
  Widget build (BuildContext context) {
    return Text( 'Hello World! '); //  Static text displayed on the screen.
}
```

Flutter basic app

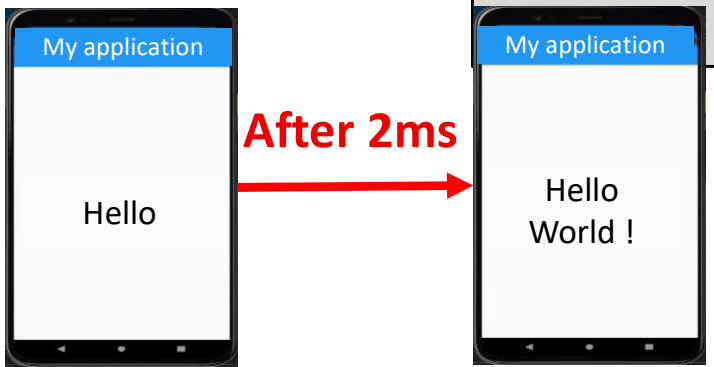
- **General structure of a *Stateless app*:**

```
import 'package:flutter/material.dart';  // Set of Material design widgets.
void main() => runApp (MyApp()); // App's entry point
/* Root widget of the  apps */
class MyApp extends StatelessWidget {
  @override
  Widget build (BuildContext context) { // Describes the UI represented by this widget.
   return MaterialApp ( // Top-level Material Design widget, which wraps all Material widgets.
              title: ' My application ', //  Application title.
              theme: ThemeData(primarySwatch: Colors.blue),
              home: BasicPage () , // The interface displayed when the app starts.
              );
 } }

class BasicPage extends StatelessWidget {// The interface is stateless.
  @override
  Widget build (BuildContext context) {
    return Text( 'Hello World! '); //  Static text displayed on the screen.
}
```

Flutter basic app

- **General structure of *Stateful app* :**
  - Creating a stateful widget by implementing two classes (**StatefulWidget** and a **State** subclasses).

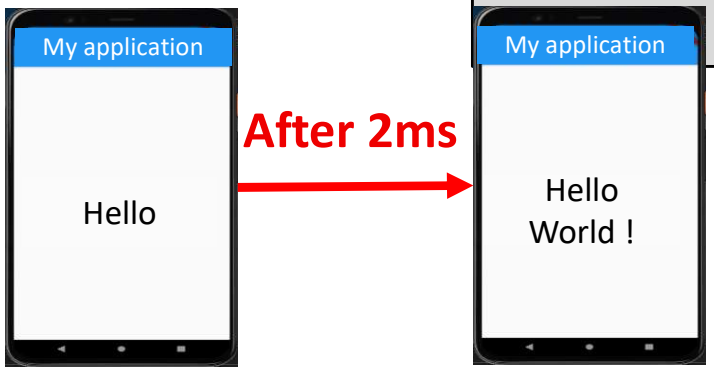
```
// A subclass defining a stateful widget.
class BasicPage extends StatefulWidget {

// Create the mutable state associated with the defined widget.
  @override
  BasicPageState createState () => BasicPageState ();
// The MyPageState class contains the state of MyPage.
class BasicPageState extends State < BasicPage > {
        String displayedText = "Hello"; // Variable that holds the
                                        text displayed on the screen.
```



Flutter basic app

- **General structure of *Stateful app* :**
  - Creating a stateful widget by implementing two classes (**StatefulWidget** and a **State** subclasses).

```
// A subclass defining a stateful widget.
class BasicPage extends StatefulWidget {

// Create the mutable state associated with the defined widget.
@override
OR
State < BasicPage > createState ( ) => BasicPageState ();
// The MyPageState class contains the state of MyPage.
class BasicPageState extends State < BasicPage > {
        String displayedText = "Hello"; // Variable that holds the
                                        text displayed on the screen.
```



Flutter basic app

- **General structure of *Stateful app* :** State methods.

```
@override
void initState() {// Method called once when the state is initialized.
        super.initState();
        // Example : change the text after 2 secondes
        Future.delayed(Duration(seconds: 2), () {
                setState(() {// setState notifies Flutter that the UI should be  rebuilt.
                        displayedText = "Hello World!"; });
        }); }
@override
void dispose() {//  Method called when the widget is removed from the widget tree.
        super.dispose(); }
 @override
 Widget build ( BuildContext context) {// Method that describes the UI represented by this
                                        widget.
        return Text(displayedText); } // Display the current text on the screen.
    } // End of MyPageState (the mutable state of the stateful widget).
} // End of MyPage widget (Stateful interface).
```

Flutter basic app

- **BuildContext:** represents the widget's position in the widget tree.
- Reflects relationships with parent and child widgets.
- Provides access to ancestor widgets and shared data (e.g., *Theme*, *MediaQuery*, *InheritedWidget*).
- Used for navigating between screens.
- Allows access to size and layout information of the target platform screen.

Flutter basic app

- **BuildContext:** use case examples.

```
  @override
  Widget build (BuildContext context) {
   final plat = Theme.of(context).platform ;  // returns Android for example.
   final color = Theme.of(context).primaryColor; //returns the primary color of the app theme.
   final size = MediaQuery.of(context).size ;  // returns the size of the screen eg (411,820)
   final orientation = MediaQuery.of(context).orientation; // returns portrait or landscape
                                                           according to the orientation of the device
   Navigator.pushNamed(context, '/OtherScreen'); //Navigate to the screen named '/OtherScree'.

  return ……
  }
}

   // body:  (orientation == Orientation.portrait): page1() ? page2()
```

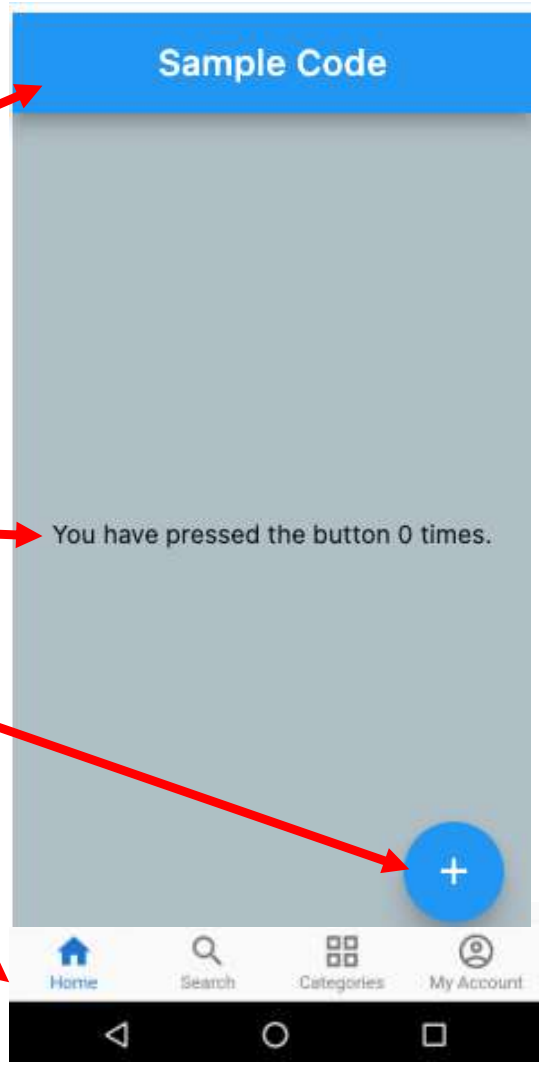
Plan

1. **Exploring Flutter basic application.**
   - Stateless app.
   - Statefull app.
   - BuildContext.
2. **Graphical widgets.**
   - Layout widgets.
   - Display widgets.
   - Interactive widgets.
   - Scrolling widgets: ListView & GridView.

Layout widgets

- **Scaffold,** a class representing the skeleton of the view :

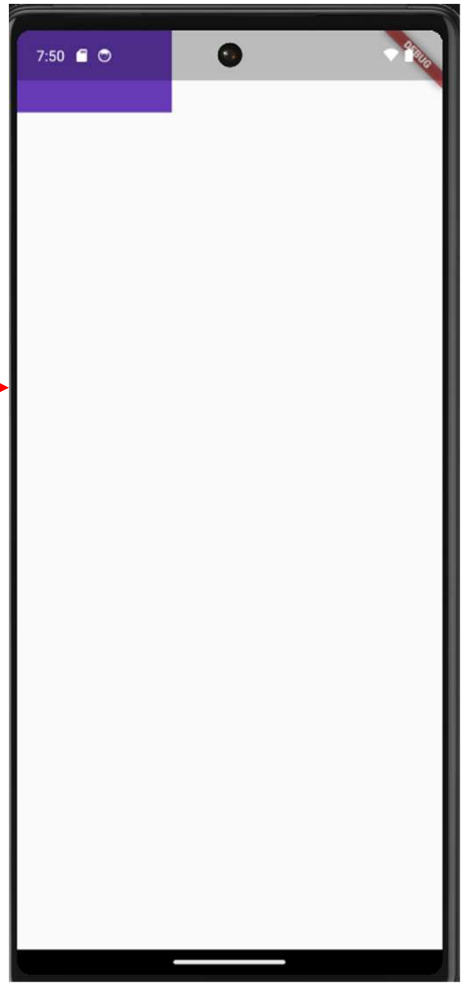
```
class BasicPage extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold (
              appBar: …. ,
              body : …. ,
              floatingActionButton : …. ,
              bottomNavigationBar : …. ,
               .
               .
               .
        );
    }
}
```



Layout widgets

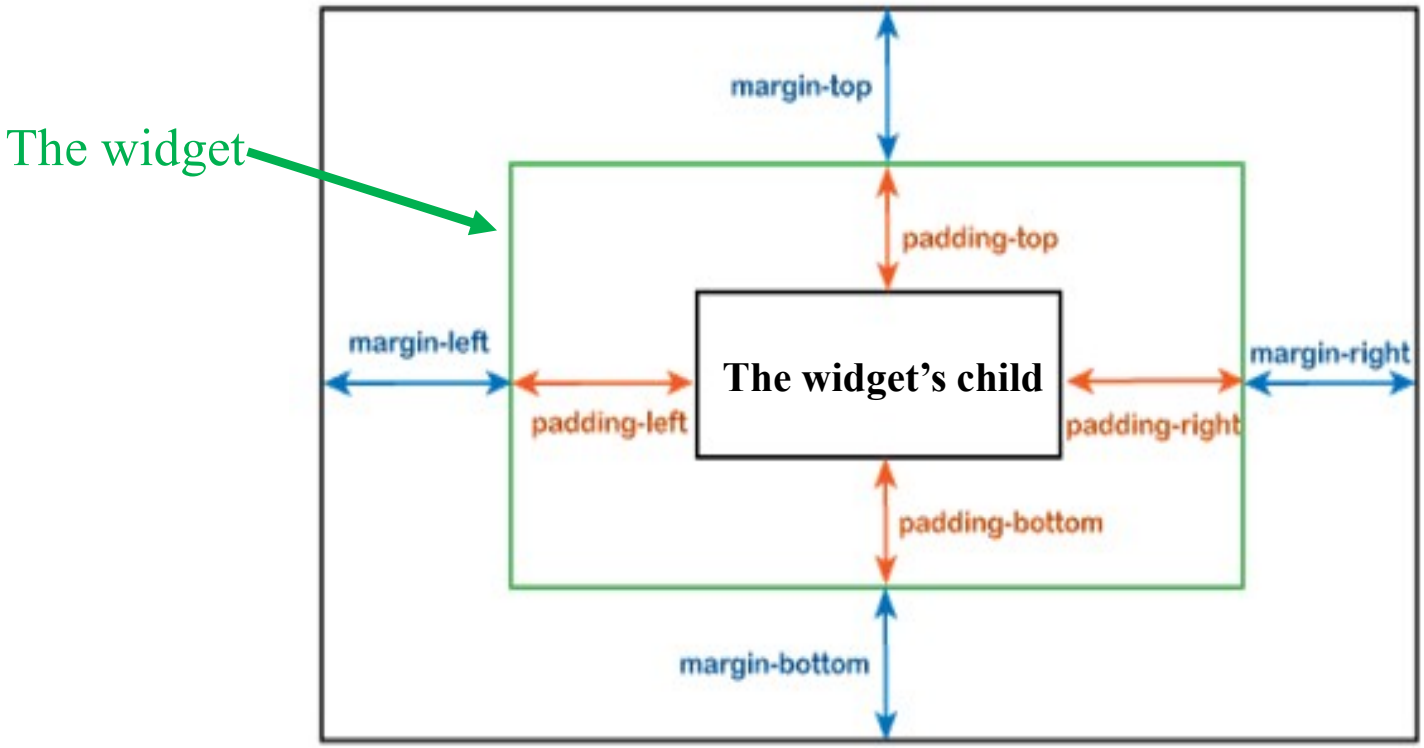
- **Container,** combines painting, positioning, and sizing features in a single convenient widget.

```
class BasicPage extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold (
                  body : Container (
                  height: 80 ,
                  width: 150 ,
                  color: Colors.deepPurple,
                  child: … ,
              ), // Container
          ); // Scaffold
      }
}
```



Layout widgets

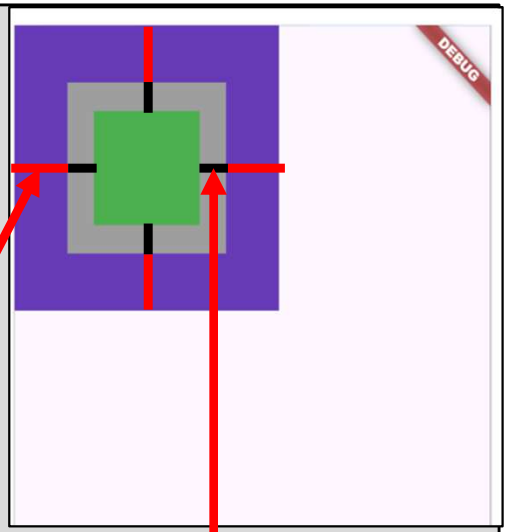
- **margin & padding,** two properties that control the spacing around a widget.
  - **margin** is the space outside a widget, creating distance between the widget and its parent.
  - **padding** is the space inside the widget, between its content and its boundary.



Layout widgets

- **margin & padding,** two properties that control the spacing around a widget.

```
class BasicPage extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold (
              body : Container ( height: 200,
                                 width: 200,
                                 color: Colors.deepPurple,
                                 padding: EdgeInsets.all(40),
                                 child: Container( color: Colors.grey,
                                                   child: Container(
                                                   color: Colors.green,
                                                   margin: EdgeInsets.all(20), ),
                                                 ), // Container
                                 ), // Container
                        ); // Scaffold
}    }
```



EdgeInsets.only (left: 10, top: 20, right: 30, bottom: 40,)

Layout widgets

- **Align,** arranges a child widget inside its parent based on a given alignment.

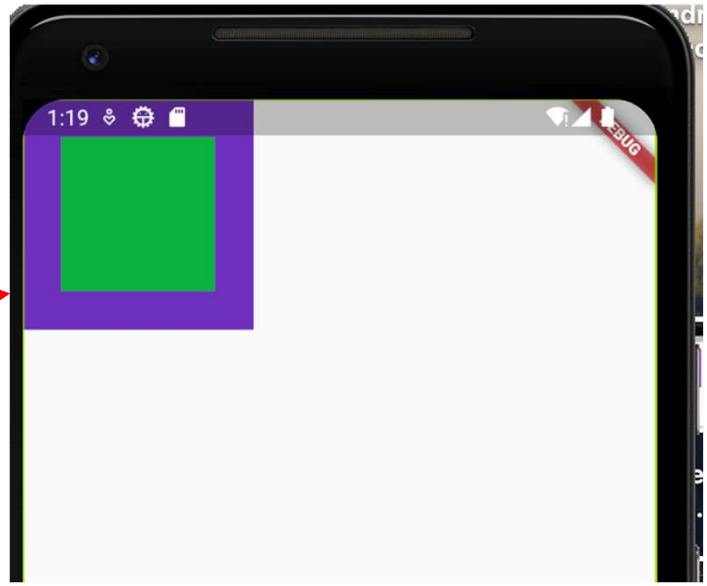
```
class BasicPage extends StatelessWidget {
@override
 Widget build (BuildContext context) {
     return Scaffold(
             body: Container(
                         width: 150,
                         height: 150,
                         color: Colors.green,
                         child: Align( alignment: Alignment.topRight,
                              child: Container(
                                                  width: 100,
                                                  height: 100,
                                                  color: Colors.deepPurple,
                                             ),// Container
                                    ), // Align
                               ), // Container
                          ); // Scaffold
             }
}
```

topRight / topCenter / topLeft / centerLeft / centrRight / bottomRight / bottomCenter / bottomLeft

Layout widgets

- **Center,** creates a widget that aligns its child widget to the center of the available space.

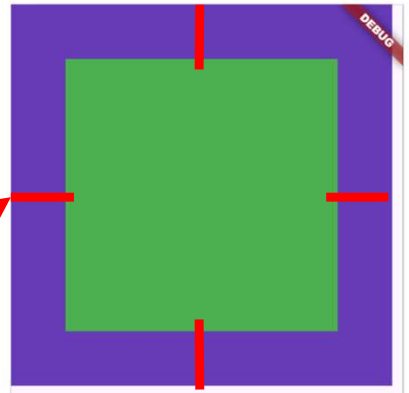
```
class BasicPage extends StatelessWidget {
@override
 Widget build (BuildContext context) {
    return Scaffold(
            body: Container(
            height: 150,
            width: 150,
            color: Colors.deepPurple,
            child: Center (
                        child: Container (
                                    width: 100,
                                    height: 100,
                                    color: Colors.green,
                        ), // Container
                    ), // Center
                ), // Container
          ); // Scaffold }}
```



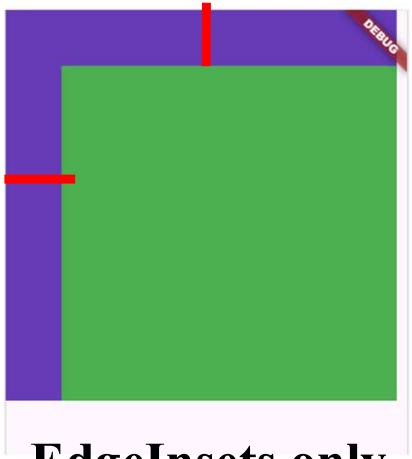
Layout widgets

- **Padding,** adds empty space inside its boundaries, between itself and its child.

```
class BasicPage extends StatelessWidget {
@override
 Widget build (BuildContext context) {
     return Scaffold(
             body: Container(
             height: 350,
             width: 350,
             color: Colors.deepPurple,
             child: Padding (
                                 padding: EdgeInsets.all(50),
                                 child: Container (
                                     color: Colors.green,
                                         ), // Container
                                 ), // Padding
                         ), // Container
             ); // Scaffold
}    }
```



EdgeInsets.all(50);

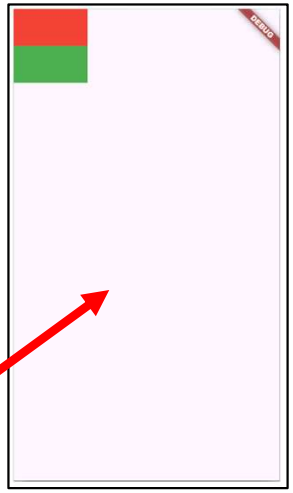


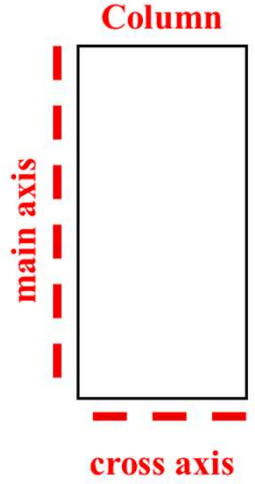
EdgeInsets.only
(top: 50, left: 50),

Layout widgets

- **Column,** is used to arrange multiple children in a vertical layout.

```
class BasicPage extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold(
            body: Column (
                          mainAxisAlignment : MainAxisAlignment.start,
                          children: [Container( height: 50,
                                                width: 100,
                                                color: Colors.red,),
                                     Container( height: 50,
                                                width: 100,
                                                color: Colors.green, ),
                                     …. //  Other children my be added here
                                    ], // children
                        ), // Column
            ); // Scaffold
  } }
```



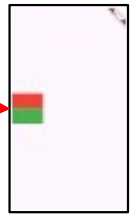


Layout widgets

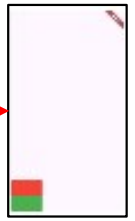
- **Column,** MainAxisAlignment property.

mainAxisAlignment: MainAxisAlignment.start,

mainAxisAlignment: MainAxisAlignment.center,

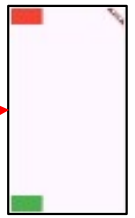


mainAxisAlignment: MainAxisAlignment.end,



mainAxisAlignment: MainAxisAlignment.spaceAround,

mainAxisAlignment: MainAxisAlignment.spaceBetween,

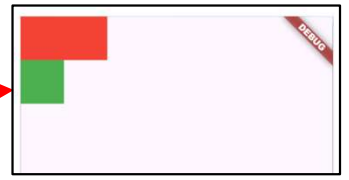


mainAxisAlignment: MainAxisAlignment.spaceEvenly,

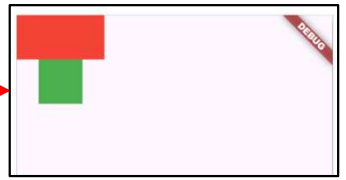
Layout widgets

- **Column,** with **MainAxisAlignment.start**, various **CrossAxisAlignment** options adjust how children align across the main axis.

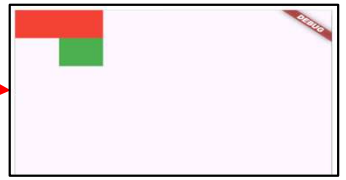
crossAxisAlignment: CrossAxisAlignment.start,



crossAxisAlignment: CrossAxisAlignment.center,



crossAxisAlignment: CrossAxisAlignment.end,

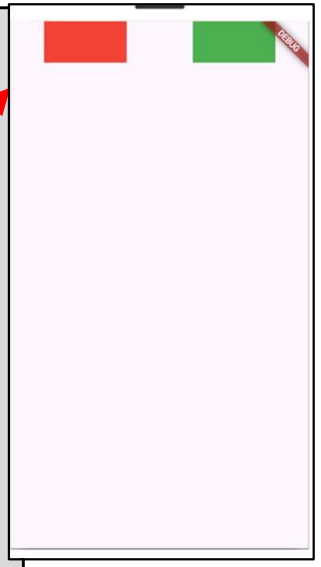


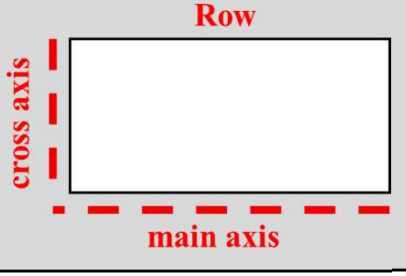
crossAxisAlignment: CrossAxisAlignment.stretch,

Layout widgets

- **Row,** a multi-child widget that displays its children in a horizontal array.

```
class BasicPage extends StatelessWidget {
 @override
 Widget build (BuildContext context) {
   return Scaffold(
          body: Row (mainAxisAlignment : MainAxisAlignment.spaceAround,
                        children: [Container(    height: 50,
                                                 width: 100,
                                                 color: Colors.red,),
                                   Container(    height: 50,
                                                 width: 100,
                                                 color: Colors.green, ),
                                  …. //  Other children my be added here
                              ], // children
                        ), // Row
          ); // Scaffold
    }   }
```



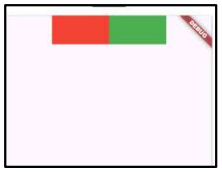


Layout widgets

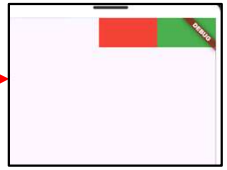
- **Row,** MainAxisAlignment property.

mainAxisAlignment: MainAxisAlignment.start,

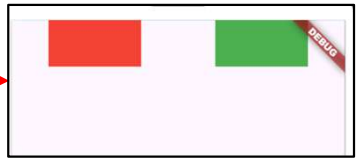
mainAxisAlignment: MainAxisAlignment.center,



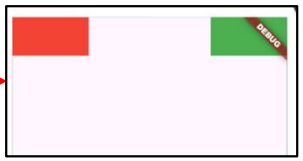
mainAxisAlignment: MainAxisAlignment.end,



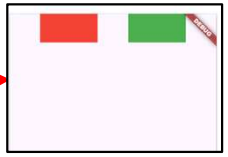
mainAxisAlignment: MainAxisAlignment.spaceAround,



mainAxisAlignment: MainAxisAlignment.spaceBetween,



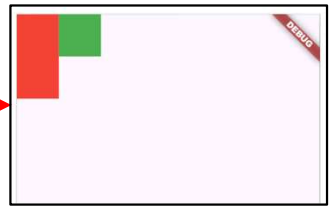
mainAxisAlignment: MainAxisAlignment.spaceEvenly,



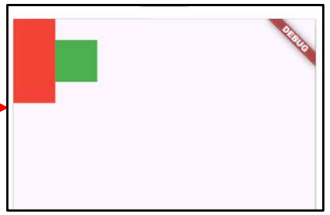
Layout widgets

- **Row,** with **MainAxisAlignment.start**, various **CrossAxisAlignment** options adjust how children align perpendicular to the main axis.

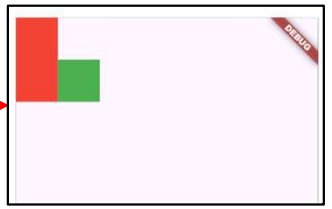
crossAxisAlignment: CrossAxisAlignment.start,



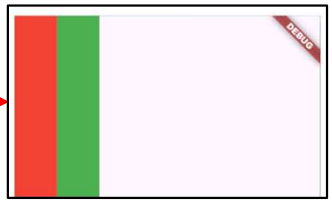
crossAxisAlignment: CrossAxisAlignment.center,



crossAxisAlignment: CrossAxisAlignment.end,



crossAxisAlignment: CrossAxisAlignment.stretch,



Layout widgets

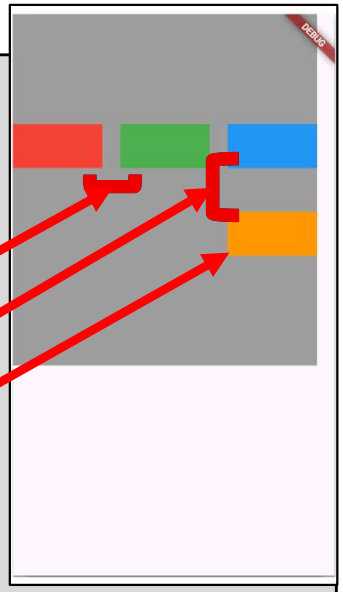
- **Wrap,** a multi-child widget that displays its children across multiple **rows** or **columns** **dynamically**, depending on the available space.

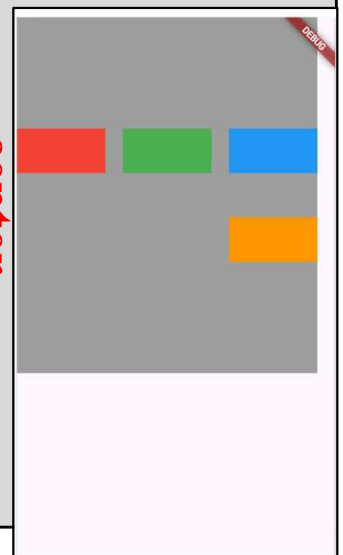
- It automatically wraps items to a new line or column if they can't fit within the current one.

Layout widgets

- **Wrap,** an illustrative example :

```
class BasicPage extends StatelessWidget {
 @override
 Widget build (BuildContext context) {
   return Scaffold( body: Container(
                              height: 400, color: Colors.grey,
                              child: Wrap (
        spacing: 20.0, // Horizontal space between items.
        runSpacing: 50.0, // Vertical space between lines.
        alignment: WrapAlignment.end, // Aligns children at the end of each line horizontally.
        runAlignment:  WrapAlignment.center, // Aligns all lines vertically in the center of the Container.
        direction: Axis.horizontal, // Layout direction (horizontal)
        children: [ Container(color: Colors.red, width: 100, height: 50),
                    Container(color: Colors.green, width: 100, height: 50),
                    Container(color: Colors.blue, width: 100, height: 50),
                    Container(color: Colors.orange, width: 100, height: 50),
                  ], // children
        ), // Wrap  ), // Container
); // Scaffold  }    }
```





Layout widgets

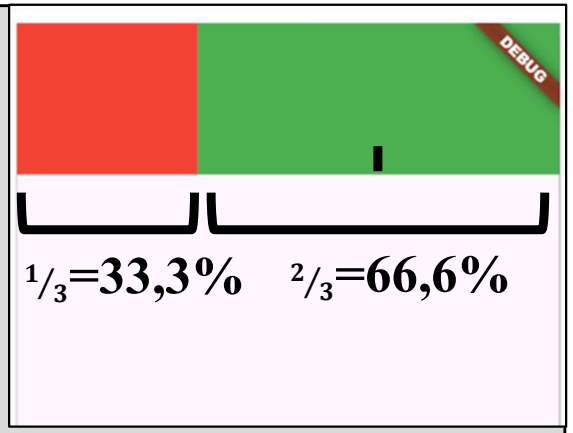
- **Expanded,** is used to make a child of a **Row** or a **Column** expand to fill the available space along the main axis (e.g., horizontally for a Row or vertically for a Column).

- If multiple children are expanded, the available space is divided among them according to the **flex factor**.

Layout widgets

- **Expanded,** an illustrative example :

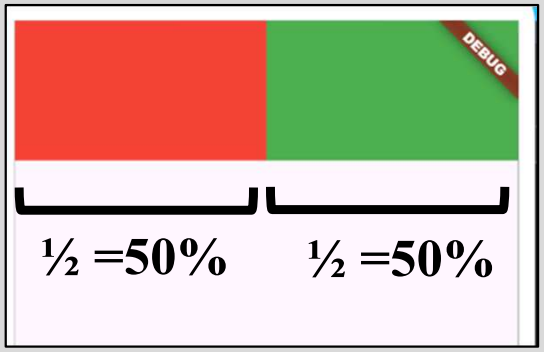
```
class BasicPage extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold(
              body: Row(
                        children: [ Expanded(
                                      flex:1,
                                      child: Container(color: Colors.red, height: 100),
                                    ), // Expanded
                                    Expanded(
                                      flex:2,
                                      child: Container(color: Colors.green, height: 100),
                                    ), // Expanded
                                  ],
                      ), // Row
                    ); // Scaffold
            }
}
```



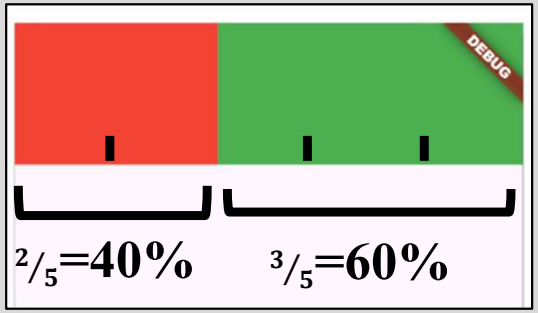
Layout widgets

- **Expanded,** an illustrative example :

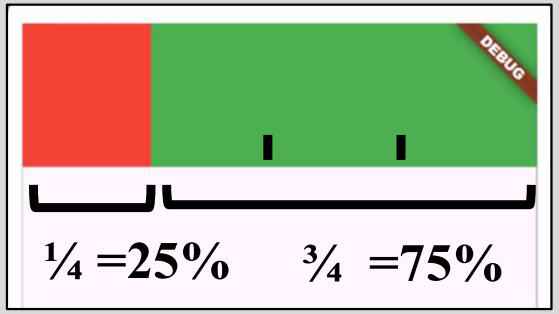
Expanded without Flex factor (flex=1)



Expanded with Flex factors 2 and 3



Expanded with Flex factors 1 and 3



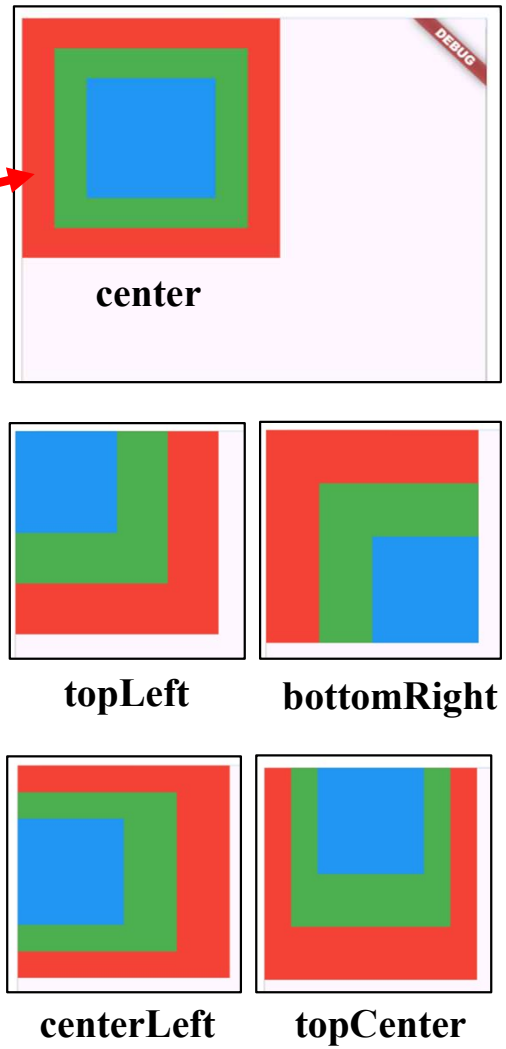
- The space of an Expanded child is equal to its flex value divided by the total of all flex values:

child space = child flex ÷ Σ(all flex factors).

Layout widgets

- **Stack,** is used to position widgets on top of each other.

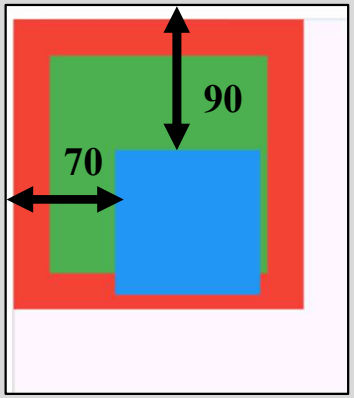
```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body:  Stack (
      alignment: Alignment.center, // topRight, topCenter, topLeft,
                                   //  bottomLeft, bottomCenter, bottomRight,
                                   // centerLeft or centerRight.
      children : [
         Container(color: Colors.red, width: 200, height: 200),
         Container(color: Colors.green, width: 150, height: 150),
         Container(color: Colors.blue, width: 100, height: 100),
       ], // children
      ), // Stack
   ); // Scaffold
  }
}
```



Layout widgets

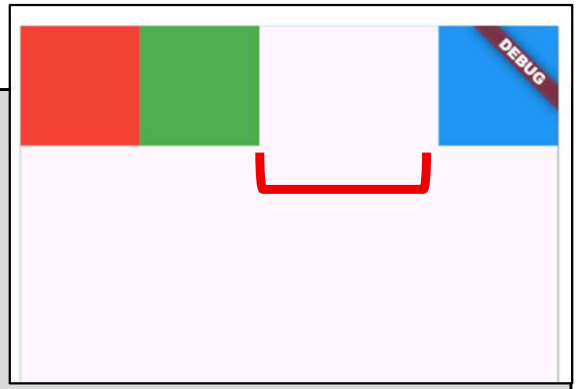
- **Positioned,** provides precise control for positioning a child widget within a **Stack**, allowing to define its location relative to the stack's boundaries.

```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body:  Stack (
     alignment: Alignment.center,
     children : [
        Container(color: Colors.red, width: 200, height: 200),
        Container(color: Colors.green, width: 150, height: 150),
        Positioned( left: 70,
                    top: 90,
                    child: Container(color: Colors.blue, width: 100, height: 100),
            ), // Positioned
       ], // children
      ), // Stack
   ); // Scaffold
 }   }
```



Layout widgets

- **Spacer,** creates a flexible block of space anywhere into a flexible widget:

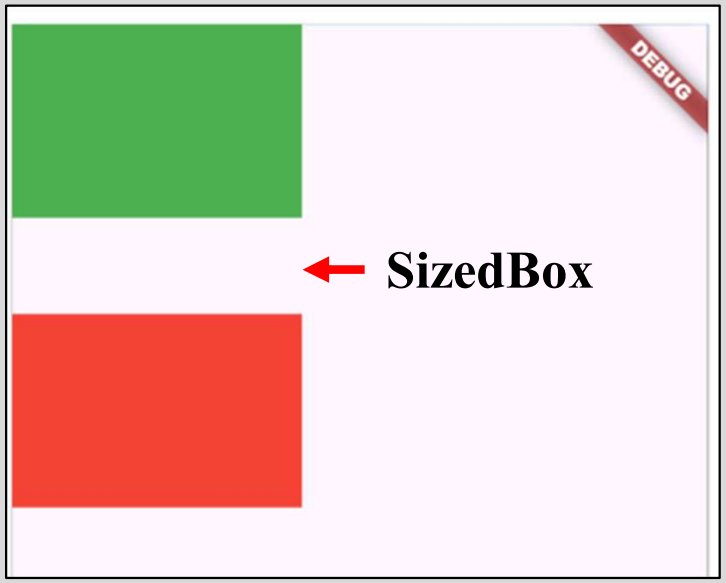



```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(Scaffold(
    body: Row(
      children: [ Container( width: 80, height: 80, color: Colors.red,),
                  Container (width: 80, height: 80, color: Colors.green, ),
                  Spacer(),
                  Container (width: 80, height: 80, color: Colors.blue, ),
                ], // children
      ), // Row
   ); // Scaffold
  }
}
```

Layout widgets

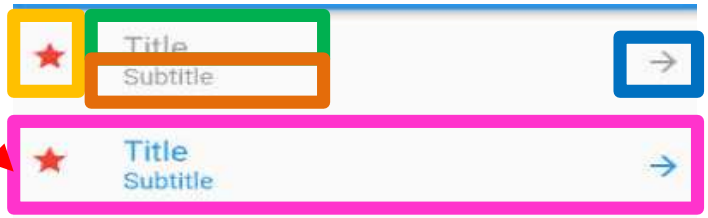
- **SizedBox,** defines a fixed size for a widget or to create spacing between widgets.

```
class BasicPage extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold ( body: Column (
            children: [ Container( width: 150, height: 100, color: Colors.green,),
                        SizedBox( height: 50, ),
                        Container( width: 150, height: 100, color: Colors.red,),
                      ],
            ), // Column
          ); // Scaffold
    }
}
```



Layout widgets

- **ListTile:** a flexible widget for creating well-structured and visually appealing layouts.



- ✓ A single fixed-height row (a convenient container).
- ✓ It contains three parts : **leading** widget (icon or avatar,…) , a **title**, a **subtitle**, **trailing** widget (checkbox, icon, radio,…).
- ✓ It supports **interaction** with the item (tapping, long press, …).

Layout widgets

- **ListTile,** an illustrative example :

```
class BasicPage extends StatelessWidget {
  @override
  Widget build (BuildContext context) {
    return Scaffold ( body:  ListTile (
                                    leading:  Icon(Icons.person, color: Colors.green),
                                    title: Text ('Med Med'),
                                    subtitle: Text('Software Engineer'),
                                    trailing: Icon(Icons.phone),
                                    onTap: ( ) { print ('You tapped on Med');  },
                                  ),
                        ); // Scaffold
                }
   }
```

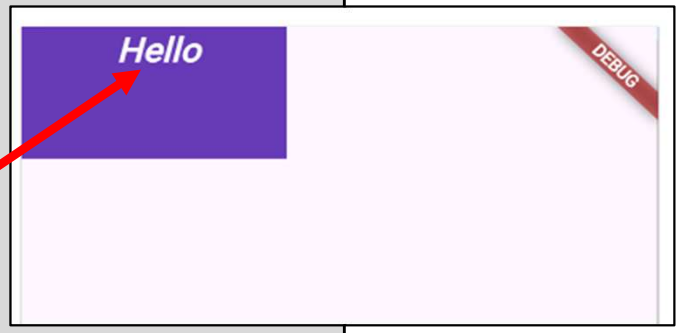
Plan

1. **Exploring Flutter basic application.**
   - Stateless app.
   - Statefull app.
   - BuildContext.
2. **Graphical widgets.**
   - Layout widgets.
   - Display widgets.
   - Interactive widgets.
   - Scrolling widgets: ListView & GridView.

Display widgets

- **Text,** displays a string with single style,

```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(Scaffold(
    body: Container (
                  height: 80 ,
                  width: 150 ,
                  color: Colors.deepPurple,
                  child: Text ( ' Hello',
                          textAlign: TextAlign.center,
                          style: TextStyle(  color: Colors.white,
                                             fontSize: 20,
                                             fontWeight: FontWeight.bold,
                                             fontStyle: FontStyle.italic,
                                          ),
                                ), // Text
                        ), // Container
      ); // Scaffold
} }
```



Display widgets

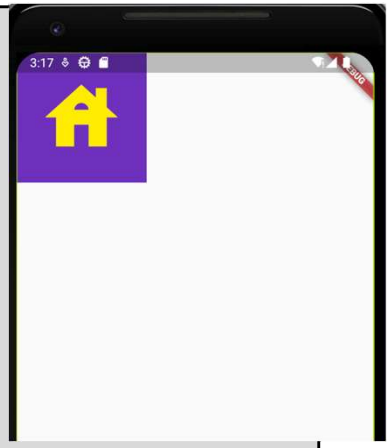
- **Text.rich,** displays text that uses multiple different styles described as a tree of **TextSpan** objects (one TextSpan = one subtree = one style).


```
child: Text.rich (
            TextSpan (
                text:"Hello",
                 style: TextStyle ( color: Colors.white, fontSize: 20,),
                 children: [ TextSpan(
                                    text: 'world',
                                    style:TextStyle( color: Colors.red, fontSize: 20,),
                                ), // Text Span
                                TextSpan(
                                    text: ' other style',
                                    style:TextStyle( color: Colors.green, fontSize: 20,),
                                ), // Text Span
                            ]
            ), // Text Span  ), // Text.rich), // Container ); // Scaffold
}}
```

Display widgets

- **Icon,** displays small graphical symbols to represent actions, objects, or concepts in the interface.

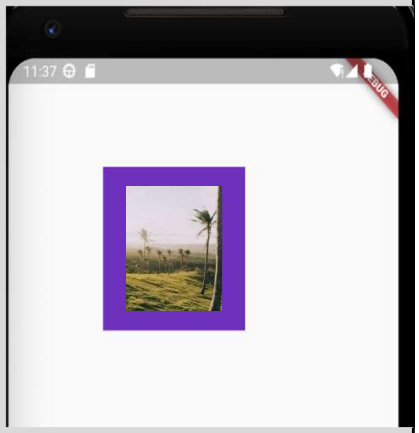
```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body: Container ( height: 150 , width: 150 ,
                      color: Colors.deepPurple,
                      child: Center (  child: Icon (
                                          Icons.house,
                                          size:100,
                                          color:Colors.yellow,
                                          ), // Icon
                                ), // Center
                      ), // Container
            ); // Scaffold
} }
```



Display widgets

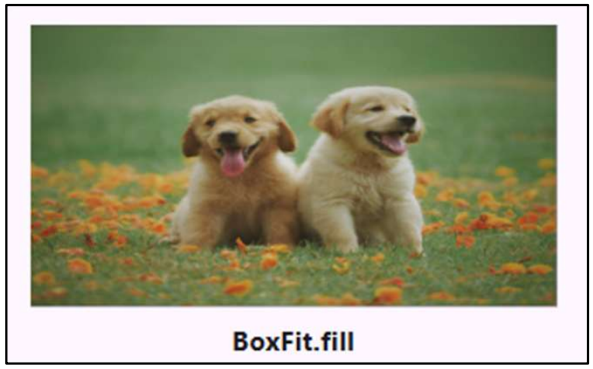
- **Image.network,** creates a widget that displays an image loaded from a network URL.

```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
                    body: Container ( height: 150 , width: 150 , color: Colors.deepPurple,
                          child : Center (
                                child: Image.network("https://www.pexels.com/photo/...",
                                                      fit: BoxFit.cover,
                                                      height: 100,
                                                      width: 100,
                                                ), // Image.network
                                ), // Center
                          ), // Container
                    ); // Scaffold
          }
}
```



Display widgets

- **Image.network,** fit property.



Stretches image to fill the box.

Fills box, may crop edges.

Fits image inside without cropping.

Matches box width.

Matches box height.

Keeps original size.

Display widgets

- **Image.asset,** creates a widget that displays an image loaded from the app's asset bundle. An asset is a static file (e.g. an image) included in the application.

Display widgets

- **Image.asset, steps to add an asset image :**

1. Create new folder in the project's root (e.g.myassets).
2. Copy the image in the folder.
3. Register the myssets folder in ***pubspec.yaml*** file and update it :

```
flutter:
    assets:
    - myassets/FirstImage.jpg
    - myassets/SecondImage.jpg
```

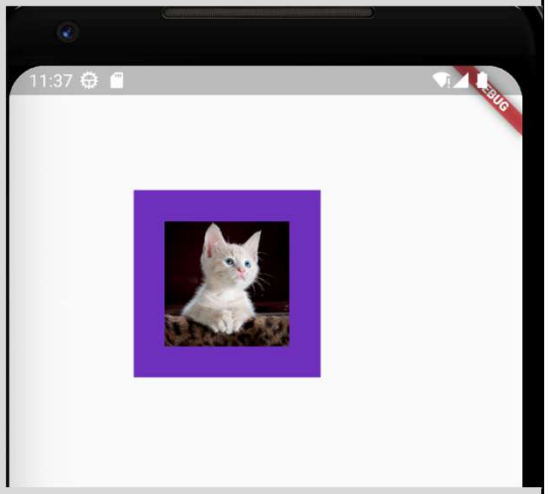
4. Insert the image code in the file, where you want to add the image:

```
Image.asset('myassets/SecondImage.jpg');
```

Display widgets

- **Image.asset,** an illustrative example :

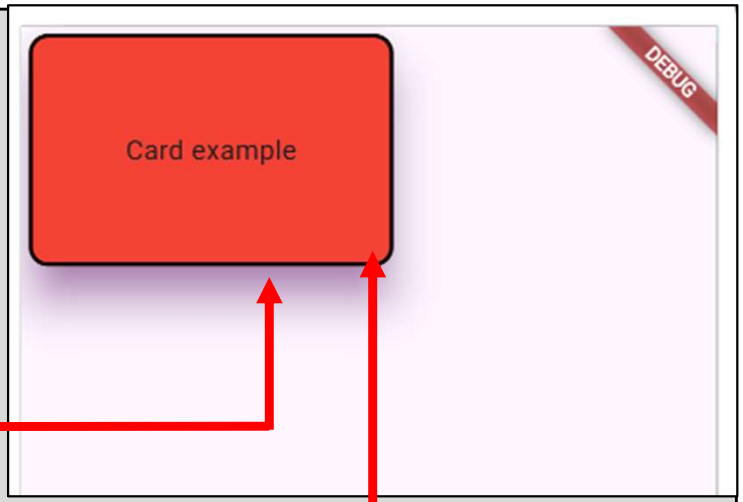
```
class BasicPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
   return Scaffold(Scaffold(
     body: Container(
            height: 150,
            width: 150,
            color: Colors.deepPurple,
            child: Image.asset(
                        "myassets/myCat.jpg",
                         fit: BoxFit.cover,
                         height: 100,
                         width: 100,
                       ), // Image.asset
                ), // Container
            ); // Scaffold
     }
}
```

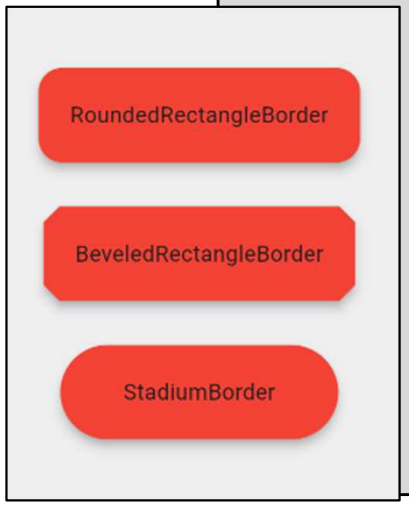


Display widgets

- **Card ,** creates a panel with rounded corners and a subtle shadow (elevation) to give a sense of depth.

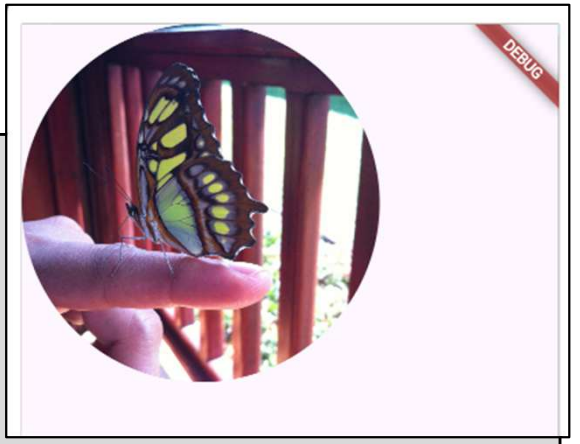
```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body: Card ( color: Colors.red,
                 shadowColor: Colors.purple,
                 elevation: 20,
                 child: Padding(
                     padding: EdgeInsets.all(50.0),
                     child: Text('Card example')),
                 shape: RoundedRectangleBorder(
                         side: BorderSide(color: Colors.black, width: 2),
                         borderRadius: BorderRadius.circular(10),
                          ), // RoundedRectangleBorder
                  ), // Padding  ), // Card
  ); // Scaffold   }   }
```





Display widgets

- **CircleAvatar,** circular image used to display user profile picture. It can display text like user's initials or just a background color.

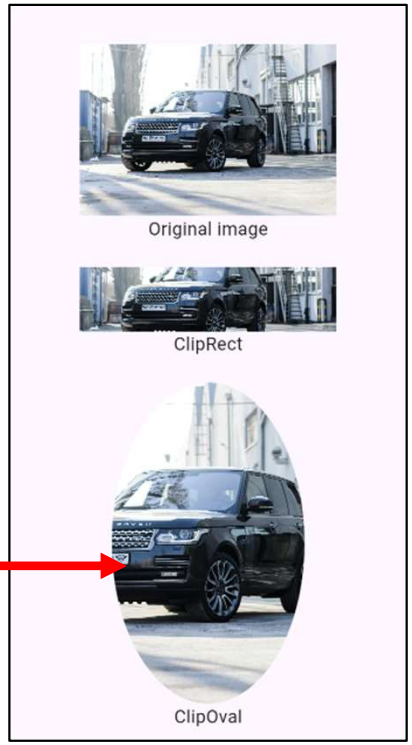



```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body: CircleAvatar(
            backgroundColor:Colors.blue, // Visible if no image
            radius: 50,
            foregroundImage: NetworkImage("https://www.pexels.com/photo/..."),
             ), // CircleAvatar
          ); // Scaffold
    }
}
```

Display widgets

- **ClipRect, ClipOval,** are used to clip their child widgets, showing only a portion of the widget in a specific shape.

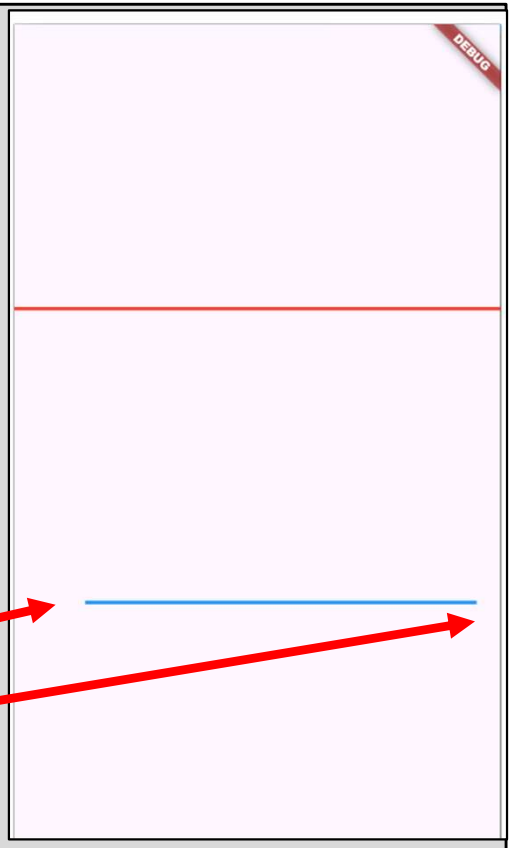
```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body: ClipOval(
                    child:Image.network (
                            ' https://www.pexels.com/photo/...',
                             width: 150,
                             height: 250,
                             fit: BoxFit.cover,
                    ), // Image.network
                ), // ClipOval
          ); // Scaffold
    }
}
```



Display widgets

- **Divider,** creates a separator. It is used to separate multiple child by the line.

```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body: Column(  children: [ Spacer(),
                               Divider ( color: Colors.red,
                                         thickness: 3, ),
                                Spacer(),
                                Divider( color: Colors.blue,
                                         thickness: 3,
                                         indent: 60,
                                         endIndent: 20, ),
                                ],
                        ), // Column
          ); // Scaffold
      }
}
```



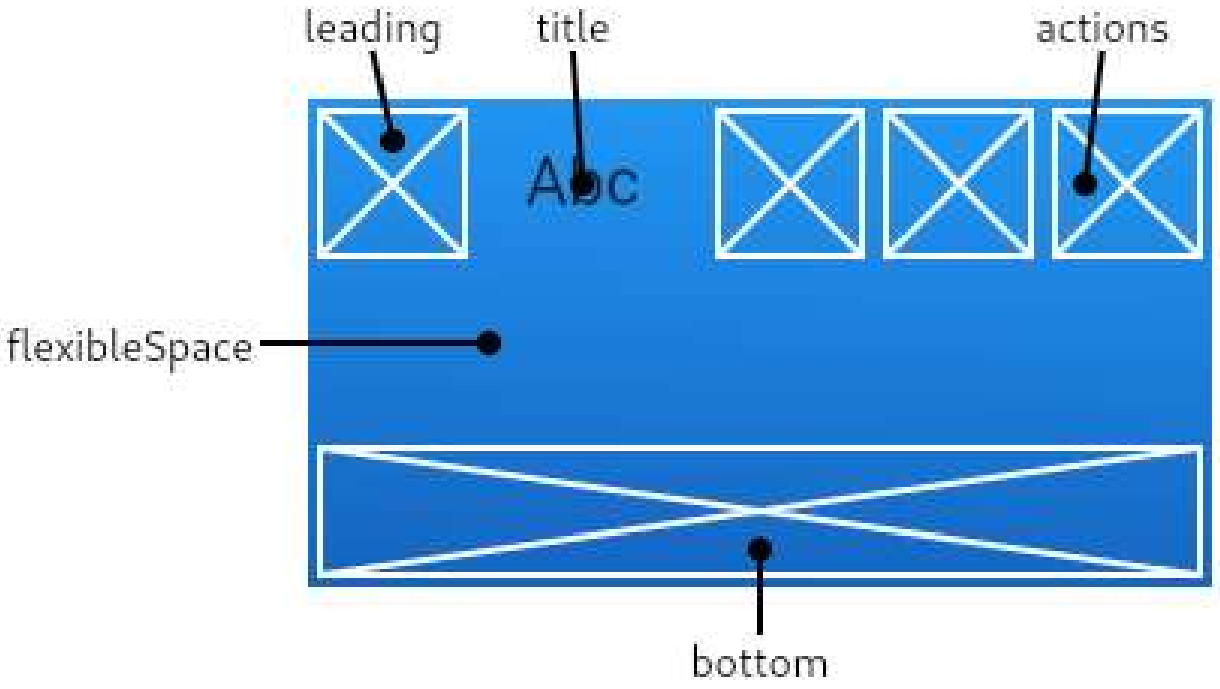
Display widgets

- **BoxDecoration,** is used with the **Container** widget to change its appearance (color, borders, rounded corners, shadows, background images).


```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold( body: Container(width: 200, height: 200,
                decoration: BoxDecoration (
                              color: Colors.blue,
                              border: Border.all(color: Colors.black,width: 3,),
                              image: DecorationImage( image: NetworkImage(" https://..."), ),
                              boxShadow: [ BoxShadow( color: Colors.red,
                                                      offset: Offset(15, 15),
                                                      blurRadius: 10,),
                                           ], // BoxShadow
                              borderRadius: BorderRadius.all(Radius.circular(15),),
                              //  OR shape: BoxShape.circle,
                            ), // BoxDecoration  ), //Container ); //Scaffold  } }
```

Display widgets

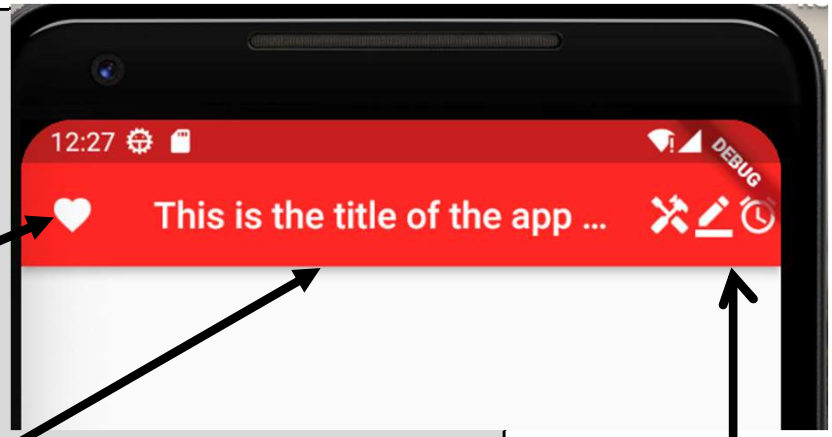
- **AppBar,** creates an app bar, usually assigned to the ***Scaffold.appBar*** property.



Display widgets

- **AppBar,** an illustrative example :

```
class BasicPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
   return Scaffold(
appBar: AppBar (
            leading: Icon(Icons.favorite),
            title: Text("This is the title of the app bar"),
            backgroundColor: Colors.red,
            actions: [
                  Icon(Icons.handyman),
                  Icon(Icons.border_color),
                  Icon(Icons.access_alarm)
            ],
        ), // AppBar
body: …… ……
            ); // Scaffold
      }
}
```



Display widgets

- **Chip,** is used to display compact information like a tag, category, or contact, often with an optional icon or delete action.

Display widgets

- **Chip,** an illustrative example :

```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold( body: Wrap ( children: [
      Chip ( avatar: CircleAvatar(  backgroundColor: Colors.white,  child: Text('F'), ),
             label: Text('Flutter'), backgroundColor: Colors.pink,
             deleteIcon: Icon(Icons.close, color: Colors.white),
             onDeleted: ( ) { … },   ), // Chip
      Chip ( avatar: CircleAvatar( backgroundColor: Colors.green,,child: Text('D'), ),
             label: Text('Dart'), backgroundColor: Colors.teal,
             deleteIcon: Icon(Icons.close, color: Colors.white),
             onDeleted: ( ) { … },  ), // Chip
      Chip ( avatar: Icon(Icons.star), // Icon
             label: Text('Firebase'), backgroundColor: Colors.teal,
             deleteIcon: Icon(Icons.close, color: Colors.white),
             onDeleted: ( ) { … }, ), // Chip
                        ],   ), // Wrap ); // Scaffold   }   }
```

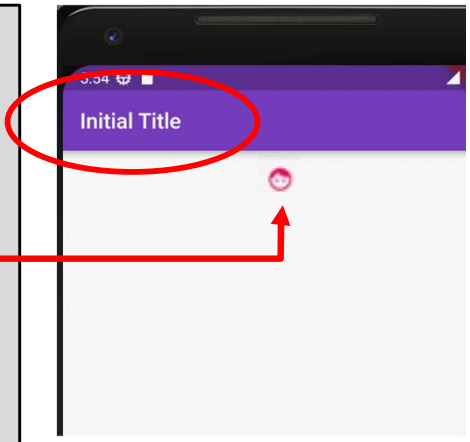
Plan

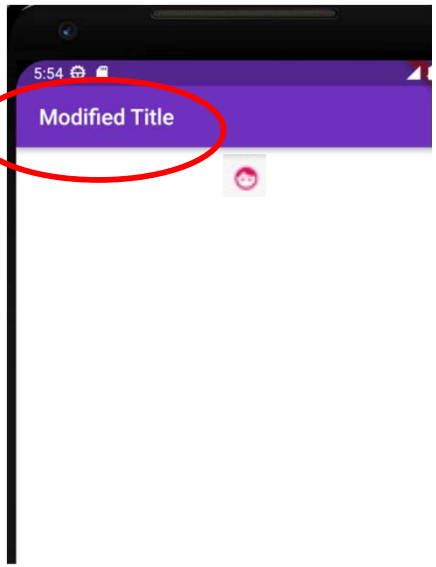
1. **Exploring Flutter basic application.**
   - Stateless app.
   - Statefull app.
   - BuildContext.
2. **Graphical widgets.**
   - Layout widgets.
   - Display widgets.
   - Interactive widgets.
   - Scrolling widgets: ListView & GridView.

Interactive widgets

- **TextButton,** a simple button without visible borders neither elevation. The label may be a Text or an Icon.

```
String title = "Initial Title";
body: Center(
        child: TextButton (
                child: Icon(Icons.face),
                style: TextButton.styleFrom(iconColor:Colors.pink),
                onPressed : updateAppBarText )
              ), // TextButton
        ), // Center
); // Scaffold

updateAppBarText(){
        setState(() {
        title = (title=="Initial Title")?" Modified Title": "Initial Title";
        });
} // updateAppBarText
```

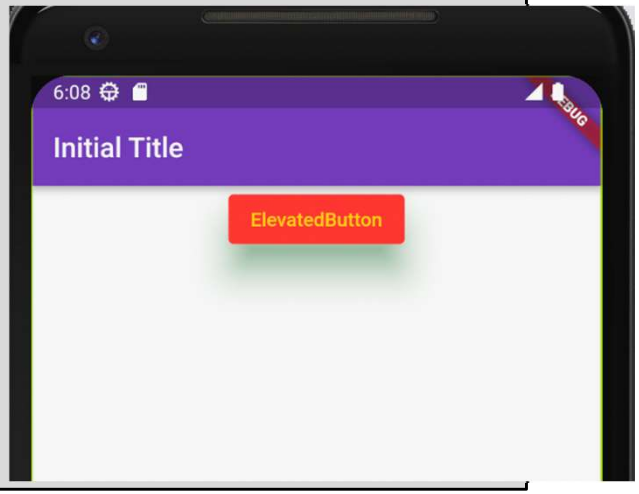




Interactive widgets

- **ElevatedButton,** is a button with a background color and elevation that increases when pressed, giving it a raised appearance.

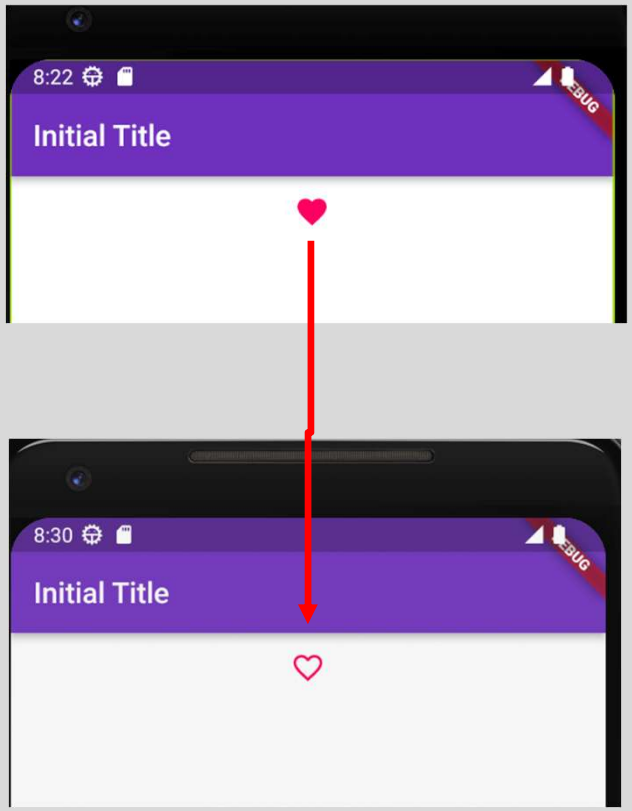
```
body: Center(
        child: ElevatedButton (
                child: Text("ElevatedButton",
                        style: TextStyle(color: Colors.amberAccent), ), // Text
                onPressed : null,
                onLongPressed : ( ) => print("Long press"),
                style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.red,
                        elevation: 20,
                        shadowColor: Colors.green, ),
            ) // ElevatedButton
        ), // Center
); // Scaffold
```



Interactive widgets

- **IconButton,** provides a graphical icon and responds to user taps.

```
Icon myIcon = Icons.favorite;

body: Center(
        child: IconButton (
                    icon: Icon(myIcon),
                    color: Colors.pink,
                    splashColor: Colors.pinkAccent,
                    onPressed : setIcon,
                ) // IconButton
      ), // Center
); // Scaffold

setIcon(){
        setState(() {
         myIcon = (myIcon ==Icons.favorite)? Icons.favorite_border:Icons.favorite;
        });
```



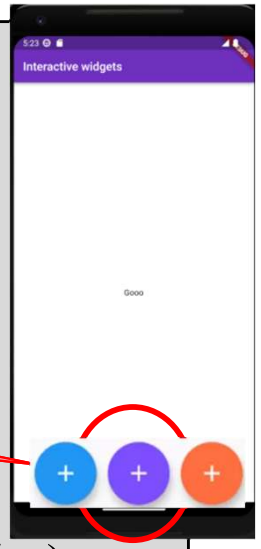
Interactive widgets

- **FloatingActionButton,** creates a circular button that floats above the main content of the screen.

```
Color myBackColor = Colors.white;
Color  MyTextColor = Colors.black;

@override
Widget build (BuildContext context) {
return Scaffold (
        backgroundColor: myBackColor ,
        appBar: AppBar (title: Text ("Interactive widgets"), ), //AppBar
        body: Center( child: Text("Gooo", style: TextStyle (color: MyTextColor),
                                //TextStyle ), // Text ), // Center
        floatingActionButton: FloatingActionButton(
                child: Icon( Icons.add_call,), //  Icon
                onPressed: myUpdateColors, //  Call myUpdateColors function
                ), // FloatingActionButton
        floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
      ); // Scaffold
} // build
```

To add multiple FloatingActionButton use **Wrap** widget:

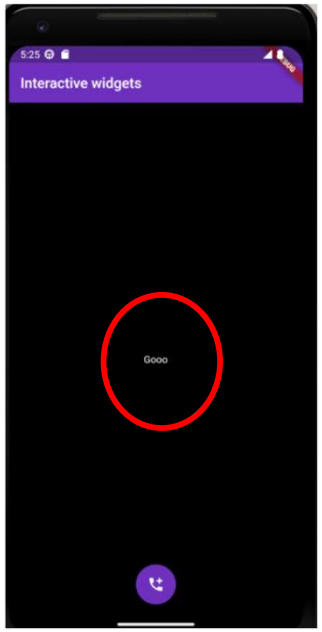
```
floatingActionButton:
        Wrap( children: <Widget>[
                FloatingActionButton(),
                FloatingActionButton(), …. ]
```

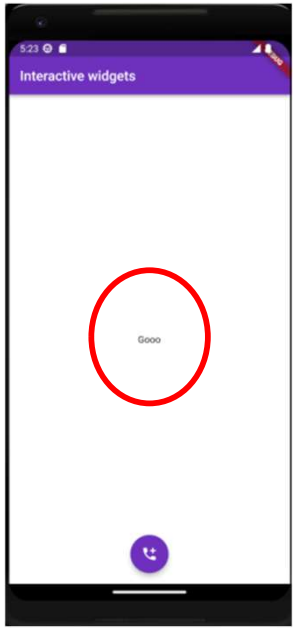


Interactive widgets

- **FloatingActionButton,** an illustrative example:

```
myUpdateColors() {
        // setState
        setState (() {
        myBackColor = (myBackColor == Colors.white) ? Colors.black : Colors.white;
        MyTextColor = (MyTextColor == Colors.black) ? Colors.white : Colors.black;}
        );} // setState
}
```

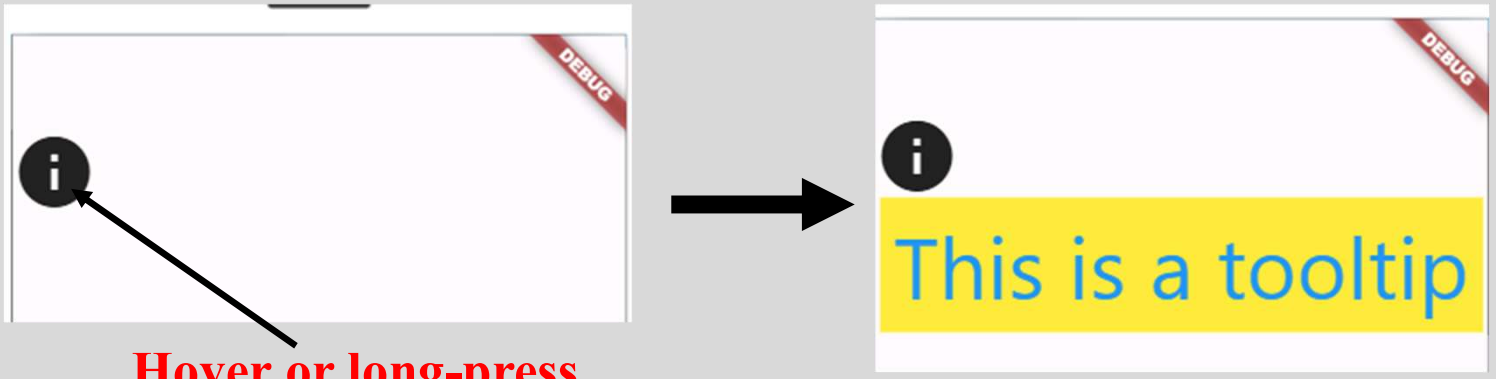




Interactive widgets

- **Tooltip,** is used to provide a small pop-up message when the user long-presses or hovers over a widget.

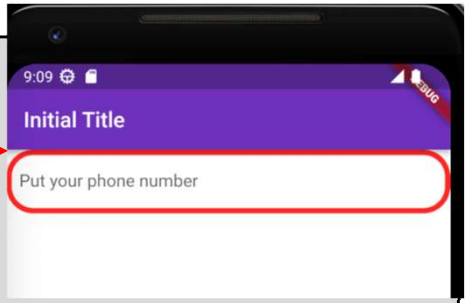
```
body: Tooltip(
        message: 'This is a tooltip',
        textStyle: TextStyle( color: Colors.blue, fontSize: 50,), // TextStyle
        decoration: BoxDecoration(color: Colors.yellow,), // BoxDecoration
        triggerMode: TooltipTriggerMode.longPress, //.tap
        child: Icon( Icons.info, size: 50.0,), // Icon
      ), // Tooltip
```

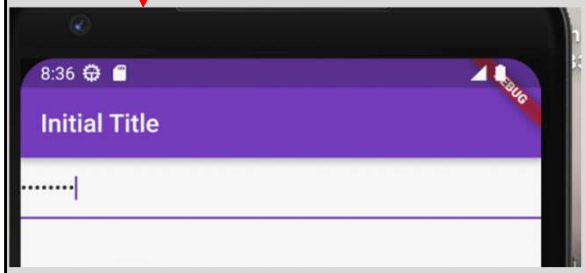


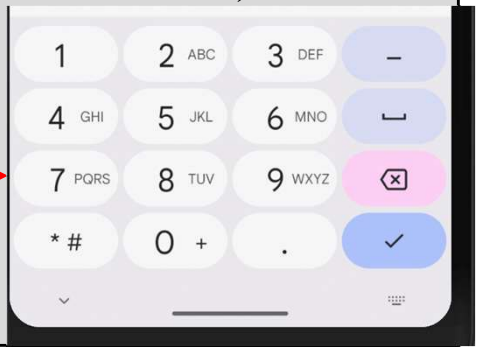
Interactive widgets

- **TextField,** allows users to type text into an app. It is used to build forms, send messages, create search field,…

```
body: TextField (
        obscureText : true, // Used to hide password
        decoration:InputDecoration(
                        hintText: "Put your phone number",
                        enabledBorder: OutlineInputBorder (// focusedBorder
                                                           // disabledBorder
                        borderRadius: BorderRadius.circular(25),
                        borderSide: BorderSide(color:Colors.red, width:5.0),
                                ), // OutlineInpuBorder
                ), // InpuDecoration
                keyboardType: TextInputType.phone,
), // TextField
```

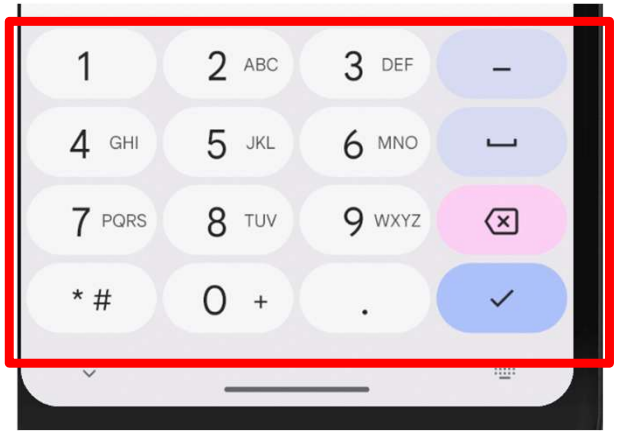




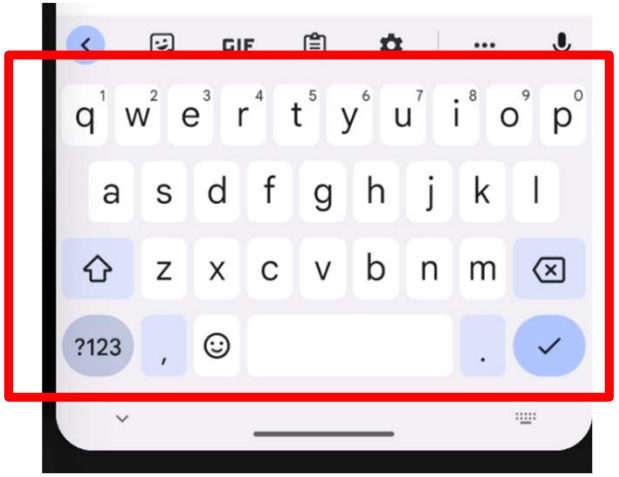


Interactive widgets

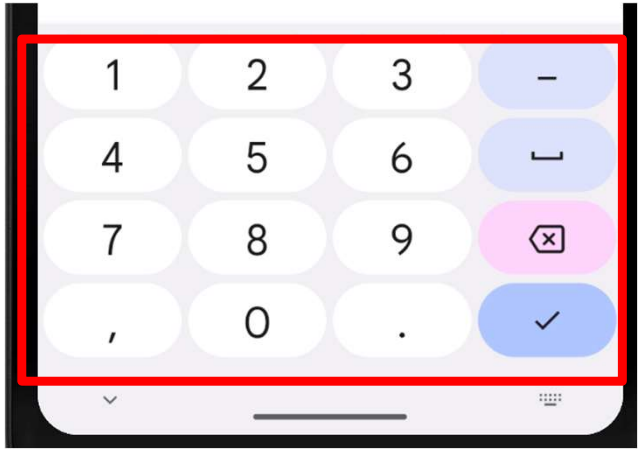
- **TextField,** an illustrative example.



TextInputType.phone



TextInputType.emailAdress

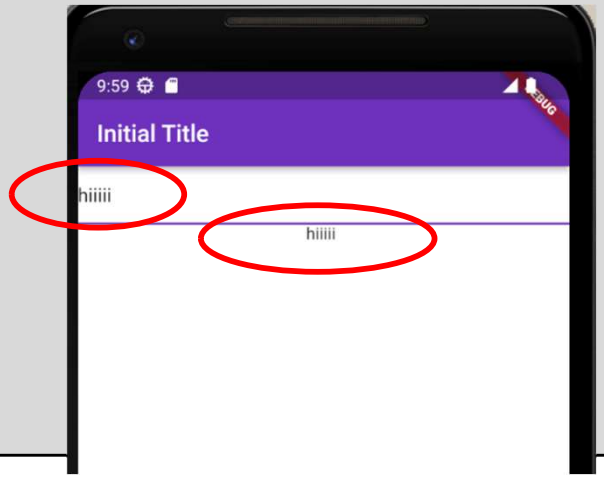


TextInputType.number

Interactive widgets

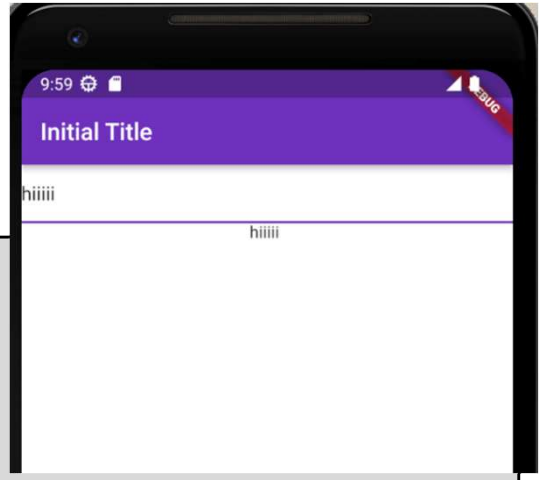
- **TextField controller,** handle changes to a textfield. An illustrative example:

```
late TextEditingController MyController ; //  late indicates that a non-nullable
                                              variable will be initialized later in the code.
body: Center(
    child: TextField (
          controller: MyController,
          decoration:InputDecoration( hintText: "Enter your name",), // InpuDecoration
          onChanged:  (newVal) {
                                    setState(( ) {});
                                   }, ),
          Text(MyController.text),
           ), // TextField
        ), // Center
); // Scaffold
```



Interactive widgets

- **TextField controller,** handle changes to a textfield. An illustrative example:

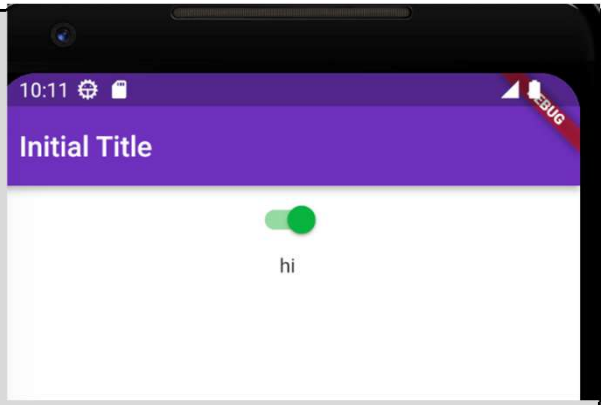


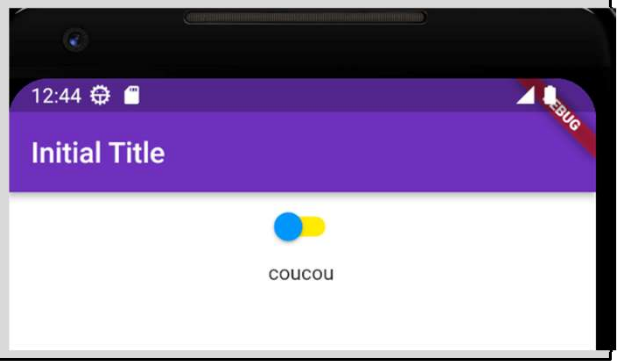

```
@override
        void initState() {
        super.initState();
        MyController = TextEditingController(); // Initialise the controller
        }

@override
        void dispose() {
        super.dispose();
        MyController.dispose(); //  remove the controller from the widget
                                    tree and release the allocated memory for it
        }
```

Interactive widgets

- **Switch,** is used to select between two options either ON (true) or OFF (false).

```
bool mySwitchVal =true;

body: Center(
    child: Switch (
            activeColor : Colors.green,
            inactiveTrackColor : Colors.yellow,
            inactiveThumbColor : Colors.blue,
            value : mySwitchVal,
            onChanged: ((newSwitchVal){ setState(( ) {
                                        mySwitchVal = newSwitchVal;
                                        },);
                        } ) ), // Switch

             Text((mySwitchVal )? "hi" : "coucou" ),
             ), // Center
); // Scaffold
```





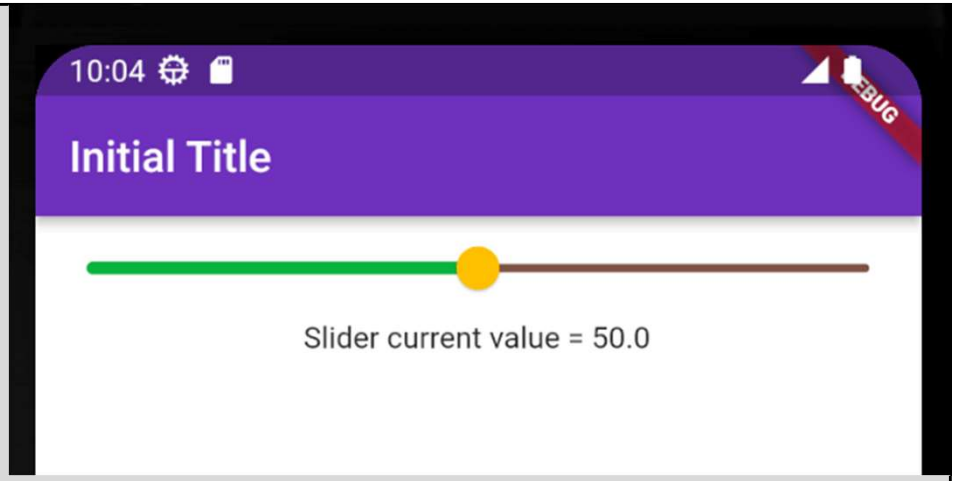
Interactive widgets

- **Slider,** consists of a seekbar (track) and a thumb (a control to slide). It allows the selection of a value from a defined range of values by moving the thumb in a horizontal direction.

Interactive widgets

- **Slider,** an illustrative example:

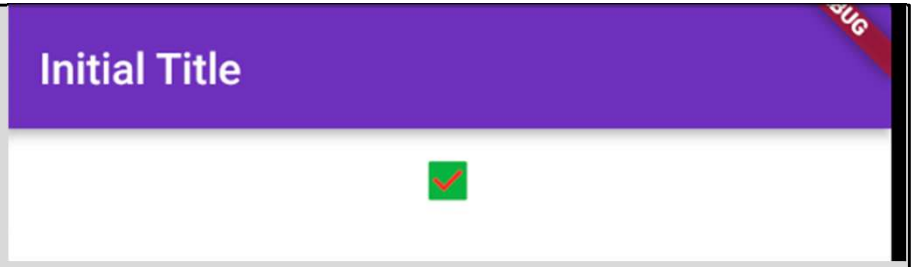
```
double sliderVal = 50;

body: Center(
    child: Slider (
            value: sliderVal,
            min: 0,
            max:100,
            thumbColor: Colors.amber,
            inactiveColor: Colors.brown,
            activeColor: Colors.green,
            onChanged: ((newval) {
                                setState(() {
                                sliderVal = newval; });
                            })
            ), // Slider
            Text("Slider current value = ${sliderVal}")
        ), // Center
); // Safford
```



Interactive widgets

- **Checkbox,** allows to toggle between two states: checked (on) and unchecked (off).

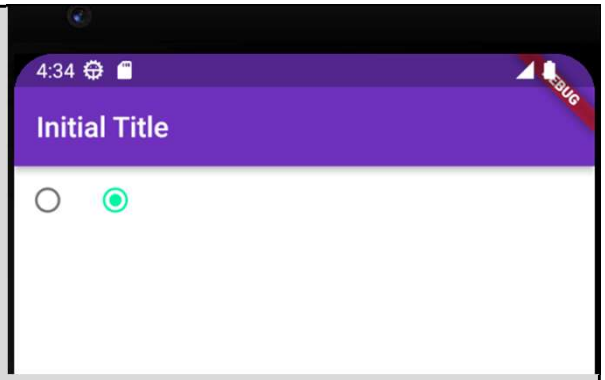
```
bool checkVal = false;

body: Center(
    child: Checkbox (
                        value: checkVal,
                        checkColor: Colors.red,
                        activeColor: Colors.green,
                        onChanged: ((newBool) =>
                                    setState(() => checkVal = newBool ?? false)),
            ), // Checkbox
        ), // Center
); // Safford
```



Interactive widgets

- **Radio,** allows to select one option from a set of mutually exclusive choices.

```
int ? selectedOption ; // nullable variable
 // int ? selectedOption = 1;  to give a default option
body: Center(
    child: Row( children: [
                    Radio (   activeColor: Colors.greenAccent,
                              value: 0,
                              groupValue: selectedOption,
                              onChanged: (newValue) { setState(() {
                                        selectedOption = newValue ;});
                                          }, ), // Radio 0
                    Radio (   activeColor: Colors.greenAccent,
                              value: 1,
                              groupValue: selectedOption,
                              onChanged: (newValue) { setState(() {
                                        selectedOption = newValue ;});
                                          }, ), // Radio 1
          ],), // Row ), // Center ); // Safford
```



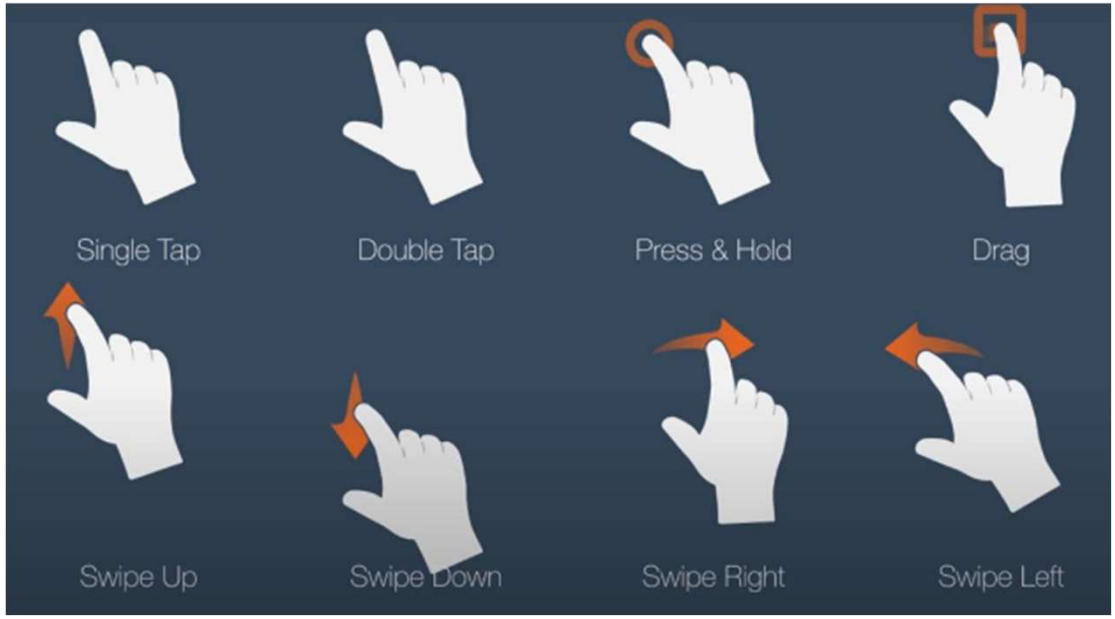
Interactive widgets

- **InkWell:** a rectangular area that responds to touch. It makes a complex widget clickable.


```
bool bougth = false;
InkWell(
        onTap: ( ) { setState(() {bougth = (! bougth ? true : false);});},
        child: Card(
                color: (bougth) ? Colors.amber : Colors.pink,
                child: Text("Cabbage"),
               ), // Card
), // InkWell
```

Interactive widgets

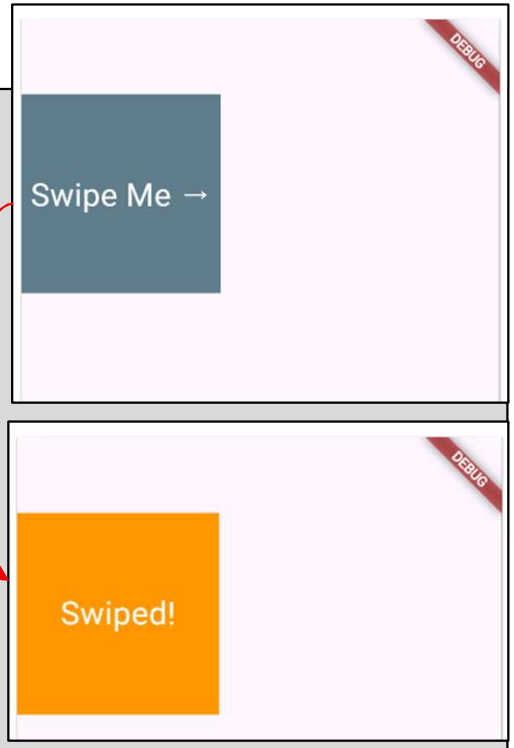
- **GestureDetector:** a non-visual widget that detects and responds to various user gestures such as ***tapping***, ***double tapping, long pressing, dragging,*** or ***scaling,*** allowing a widget to become interactive without changing its visual appearance.



Interactive widgets

- **GestureDetector:** an illustrative example :

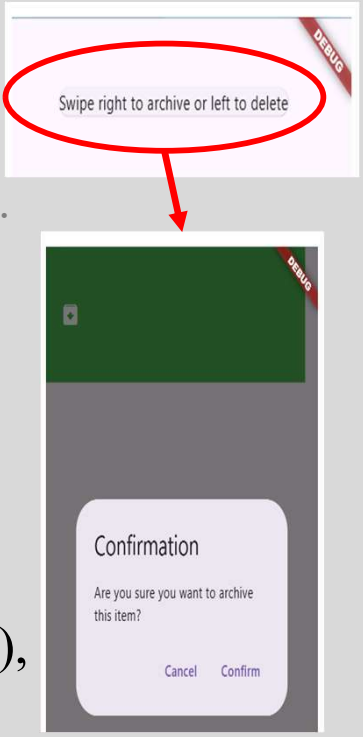
```
bool isSwiped = false;

child: GestureDetector(
        // Detection of a horizontal swipe gesture (swipe).
        onHorizontalDragEnd: (details) {
          setState(() {
            isSwiped = ! isSwiped;
          });
        },
        child: Container( height: 150, width: 150,
                          color: isSwiped ? Colors.orange : Colors.blueGrey,
                          alignment: Alignment.center,
                          child: Text( isSwiped ? "Swiped!" : "Swipe Me →",
                                       style: TextStyle(color: Colors.white, fontSize: 24),
                                     ), // Text
                        ), // Container
      ), // GestureDetector.
```



Interactive widgets

- **Dismissible:** a widget that can be dismissed by dragging it.

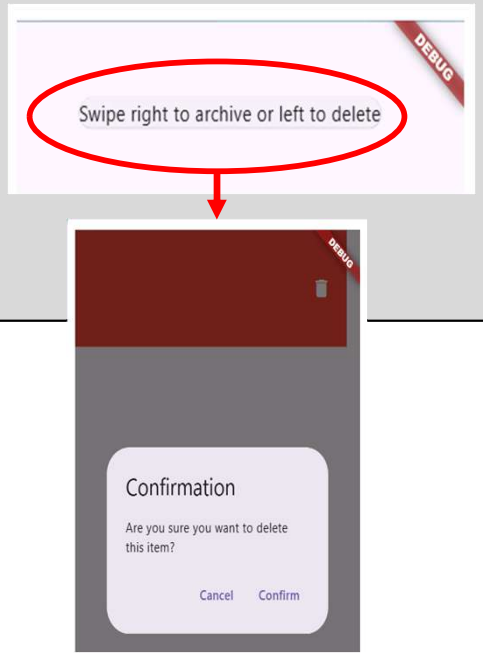
```
return Scaffold(
  body: Dismissible (
        key : ValueKey('unique_item'),
        direction : DismissDirection.horizontal, //DismissDirection.endToStart.
        background : Container (
                        color: Colors.green,
                        alignment: Alignment.centerLeft,
                        padding: EdgeInsets.only(left: 20),
                        child: Icon(Icons.archive, color: Colors.white),),
        onDismissed : (direction) {if (direction == …) … ekse …;}
        child : Card ( child: Text('Swipe right to archive or left to delete'),),
        confirmDismiss: (direction) async {
        String action = direction = = DismissDirection.startToEnd ? 'archive' : 'delete' ;
        return await showDialog <bool> ( context: context,
                builder: (context) => AlertDialog( title: Text('Confirmation'),
                    content: Text('Are you sure you want to $action this item?'),
                    actions: [ TextButton( onPressed: ( ) => {},  child: Text('Cancel'),),
                                TextButton( onPressed: ( ) => {}, child: Text('Confirm'),),
                            ], ), // AlertDialog ); // showDialog }, // confirmDismiss
```



Interactive widgets

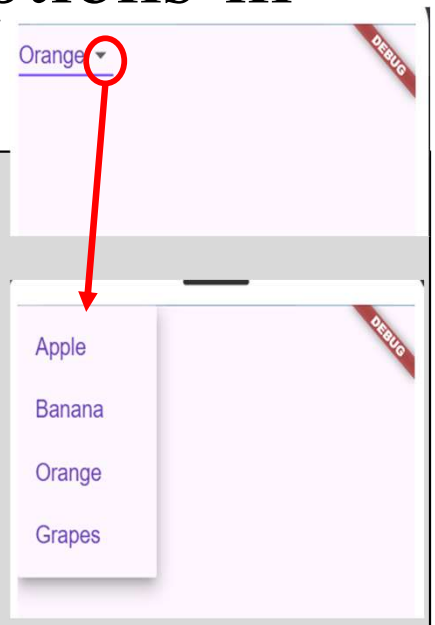
- **Dismissible:** ***secondaryBackground*** specify the widget that is stacked behind the child when dragged in the secondary (inverse) direction.

```
secondaryBackground: Container(
                                        color: Colors.red,
                                        alignment: Alignment.centerRight,
                                        padding: const EdgeInsets.only(right: 20),
                                        child: const Icon(Icons.delete, color: Colors.white),
                                      ),
                                ), // Dismissible
                        ); // Scaffold
                }
        }
```



Interactive widgets

- **DropdownButton:** displays a list of options in a drop-down menu.




```
String selectedValue = 'Apple';
final List<String> fruits = ['Apple', 'Banana', 'Orange', 'Grapes'];
   return Scaffold(
            body: DropdoxwnButton<String> (
            value: selectedValue,
            icon: Icon(Icons.arrow_drop_down),
            style: TextStyle(color: Colors.deepPurple, fontSize: 18),
            underline: Container( height: 2,color: Colors.deepPurpleAccent,),
            onChanged: (String? newValue) { setState(( ) {selectedValue = newValue !; });},
            items: fruits.map<DropdownMenuItem<String>>((String value) {
                        return DropdownMenuItem <String> (value: value, child: Text(value),);
      }).toList(),),
   ); // Scaffold
```

Plan

1. **Exploring Flutter basic application.**
   - Stateless app.
   - Statefull app.
   - BuildContext.
2. **Graphical widgets.**
   - Layout widgets.
   - Display widgets.
   - Interactive widgets.
   - Scrolling widgets: ListView & GridView.

Scrolling widgets

- **SingleChildScrollView,** allows a single child to be scrolled when there is no enough space to display it in a single view,

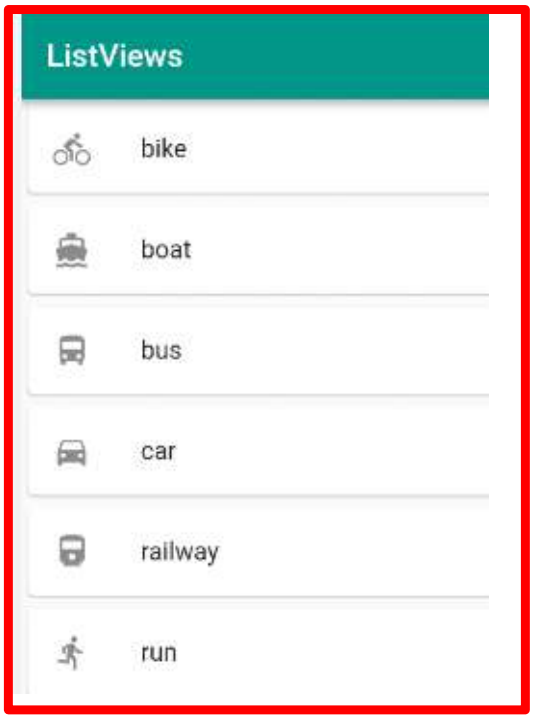
```
I/flutter (26328): ═╡ EXCEPTION CAUGHT BY RENDERING LIBRARY ╞═══════
I/flutter (26328): The following assertion was thrown during layout:
I/flutter (26328): A RenderFlex overflowed by 86 pixels on the right.
```

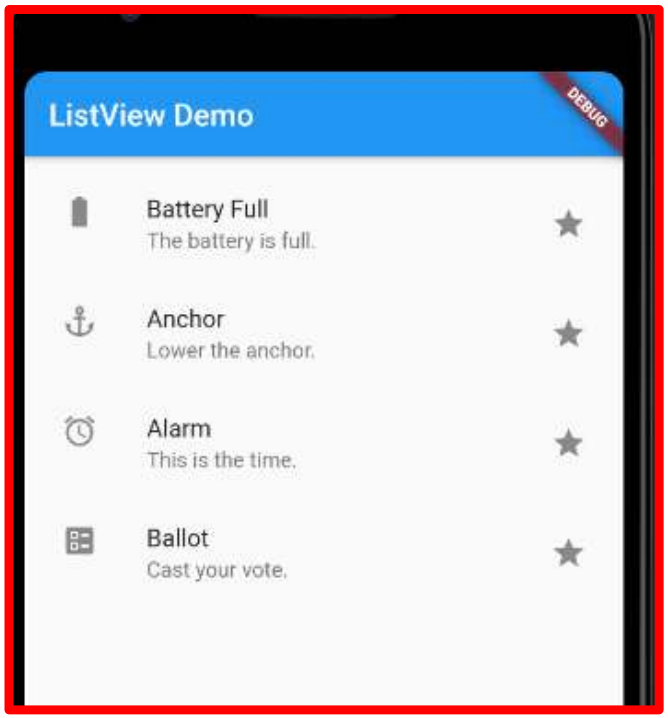
```
class BasicPage extends StatelessWidget {
  @override
 Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView( scrollDirection: Axis.horizontal,
                                 child:  Row ( children: [
                                         Image.network(" https//…", width: 1000,),
                                ), // Row ), //SingleChildScrollView,
          ); // Scaffold
  } }
```

Scrolling widgets

- **ListView:** a scrollable widget that displays its children sequentially, either vertically or horizontally, depending on the scroll direction.

- **Examples:**





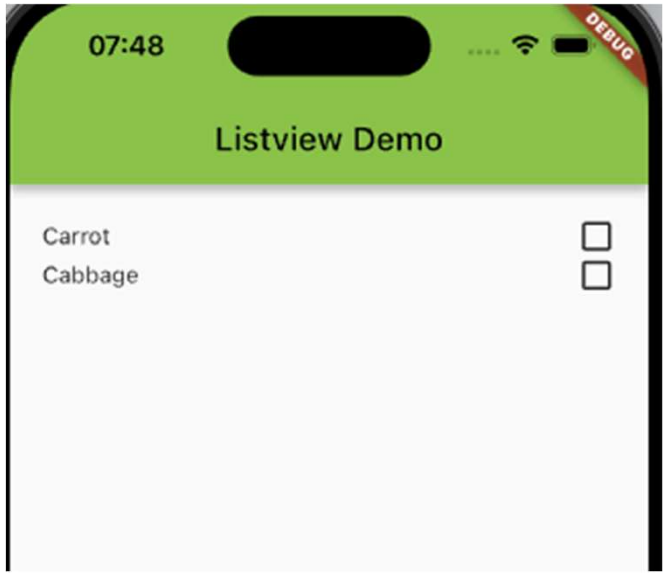
Scrolling widgets

- **Types of ListView:**
  - ListView.
  - ListView.builder.
  - ListView.separated.

Scrolling widgets

- **ListView:** is a simple widget that takes a list of widgets inside the children parameter and makes it scrollable.

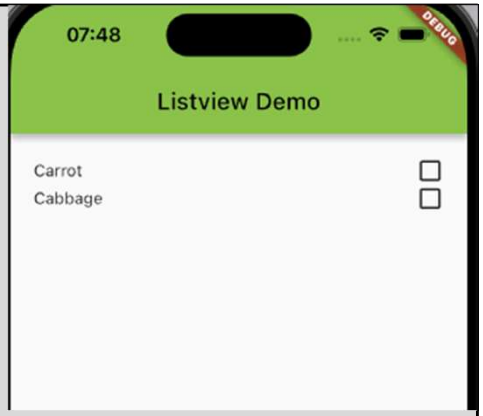
➢ Used for small list (list with few children).



Scrolling widgets

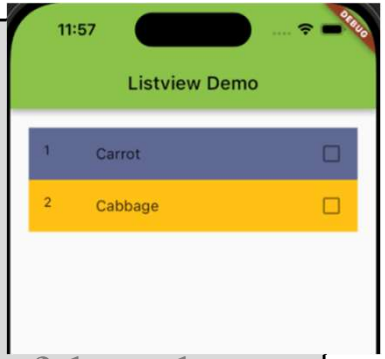
- **Implementation of ListView:**

```
//A ListView with two items

ListView(
        scrollDirection: Axis.vertical , // Axis.horizontal
        padding: EdgeInsets.all(20),
        children: <Widget>[
      Row( mainAxisAlignment: MainAxisAlignment.spaceBetween,
        mainAxisSize: MainAxisSize.max,
        children: [Text ("Carrot"),
                    Icon (Icons.check_box_outline_blank),],
        ), //Row
      Row( mainAxisAlignment: MainAxisAlignment.spaceBetween,
        mainAxisSize: MainAxisSize.max,
        children: [Text ("Cabbage"),
                    Icon (Icons.check_box_outline_blank),],
        ), //Row
    ], //children of ListView
), // ListView
```



Scrolling widgets

- **Using ListTile within a ListView :**

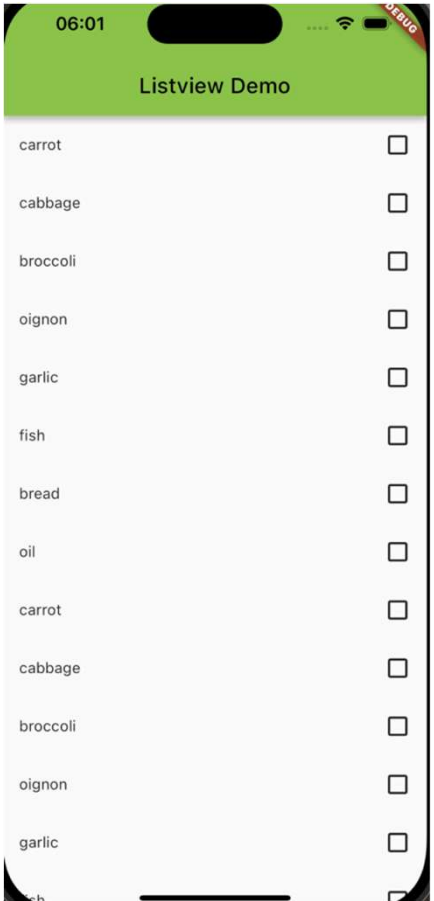



```
//A ListView with two items as a ListTile
ListView(
        scrollDirection: Axis.vertical , // Axis.horizontal
        padding: EdgeInsets.all(20),
        children: <Widget>[
    ListTile( // change the icon of trailing parameter according to the value of  bought
            title: Text ("Carrot"),
            leading: Text ("1"),
            tileColor: Colors.purple,
            trailing: Icon( (bought) ? Icons.check_box : Icons.check_box_outline_blank),
            onTap: () { setState((){ bought = (!bought?true:false); // change bought value
                });), //ListTile
     ListTile(
            title: Text ("Cabbage"),
             leading: Text ("2"),
            tileColor: Colors.amber,
            trailing: ……..    ), //ListTile
   ], //children of ListView ), // ListView
```

Scrolling widgets

- **ListView.builder:** allows the construction of a repeating list of widgets.

➢ It is useful for a large number of items.



Scrolling widgets

- **Implementation of ListView.builder:**

1. Create the Dart list:

```
List<String> articles = [
"carrot","cabbage","broccoli","oignon","garlic","fish","bread","oil","carrot","cabb
age","broccoli","oignon","garlic","fish","bread","oil"
];
```

Scrolling widgets

- **Implementation of ListView.builder:**

2. Create the widgets list:

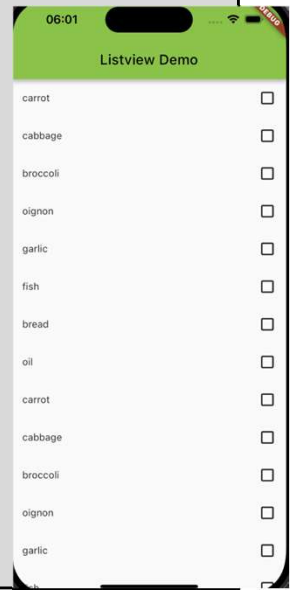
```
List <Widget> createWidgetItemsList() {

List <Widget> widgetItems = [ ]; //  Empty initial widgets list.
            articles.forEach((element) {
                        final widget = Padding( padding: EdgeInsets.all(15),
child: Row (mainAxisAlignment: MainAxisAlignment.spaceBetween,
                mainAxisSize: MainAxisSize.max,
                children: [
                            Text( element),
                            Icon(Icons.check_box_outline_blank),
                ], ), //Row); // Padding
widgetItems.add(widget ); // add the created widget to the list of widgets.
} // loop for
return widgetItems ;
} //createWidgetItemsList
```

Scrolling widgets

- **Implementation of ListView.builder:**

3. Construct the widgets list:

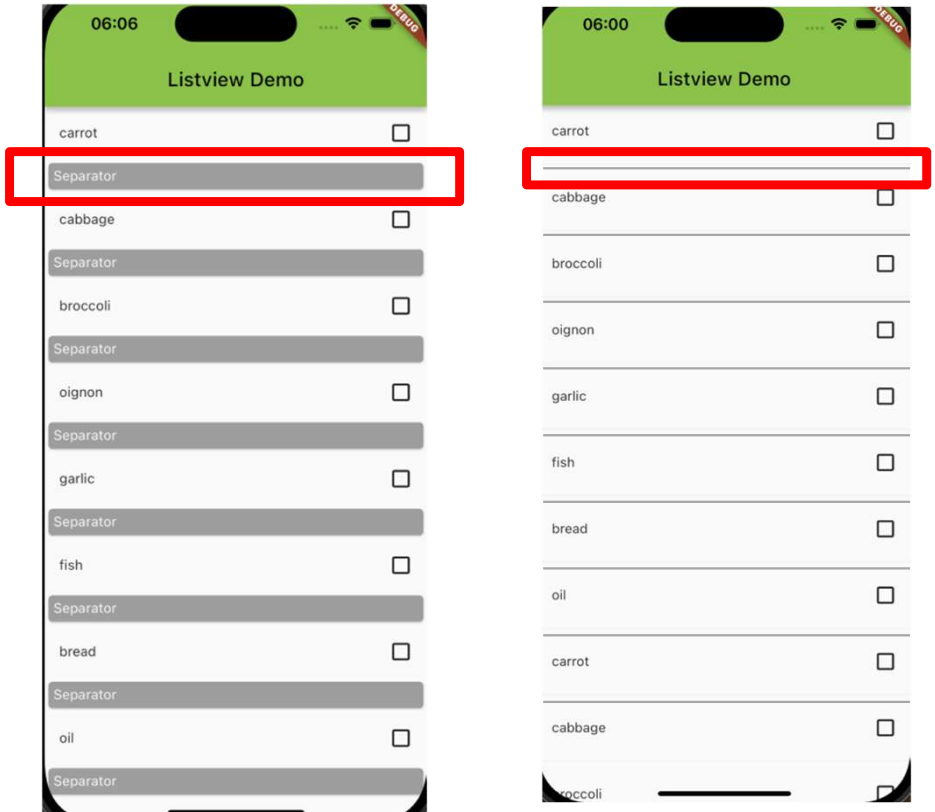
```
body: ListView.builder (
        itemCount: articles.length, // Number of repetitions
        itemBuilder: (context, index) {//construct the widget which will be
                                        generated 'itemCount' times.

                // get the list of widgets to build
                List<Widget> L= createWidgetItemsList();
                // Build the item widget in L at position index
                return L[ index];
        } // itemBuilder
) // ListView.builder
```



Scrolling widgets

- **ListView.separated:** allows the generation of a list of widget (the main list) with a separator widgets (the separator list).

➢ It is useful to distinguish visually between list items.



Scrolling widgets

- **Implementation ListView.separated:** construct the separated widgets list (**simple** separator).


```
body: ListView.separated(
        itemCount: articles.length, // Number of repetitions
        itemBuilder: (context, index) {//construct the widget which will be
                                        generated 'itemCount' times.
                // get the list of widgets to build
                List<Widget> L= createWidgetItemsList();
                // Build the item widget in L at position index
                return L[ index];
        } // itemBuilder
         separatorBuilder: (context, index) {//construct the  separator widget
                                              which will be generated for each item
                return  Divider() ; //A simple Divider used as separator
        } // separatorBuilder
) // ListView.separated
```

Scrolling widgets

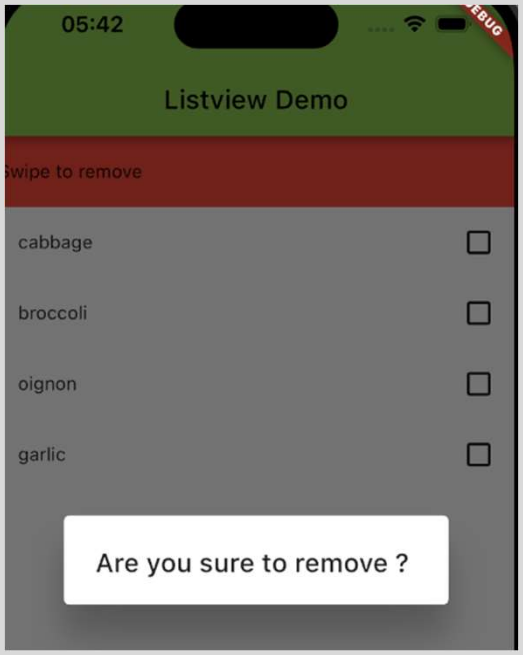
- **Implementation ListView.separated:** construct the separated widgets list (**custom** separator).


```
body: ListView.separated(
            itemCount: articles.length, // Number of repetitions
            itemBuilder: (context, index) { ……} // itemBuilder
            separatorBuilder: (context, index) { //construct the  separator widget
                                                  which will be generated for each item
//Use a customized separator
return Card(
            color: Colors.grey,
            child: Padding(
                        padding: const EdgeInsets.all(5.0),
                        child: Text( 'Separator ',style: TextStyle(color: Colors.white), ),
                    ), //Padding
                ); //Card
        } // separatorBuilder
) // ListView.separated
```

Scrolling widgets

- **Using Dismissible in a ListView:**

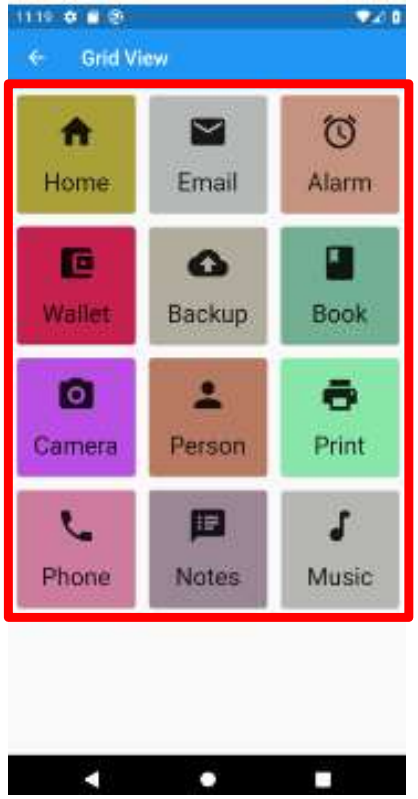
```
ListView.builder(
        itemCount: articles.length,
        itemBuilder: (context, index) {
        List<Widget> L = itemArticles();
            return Dismissible( key: ValueKey<String>(articles[index]),
                    child: L[index],
                    direction: DismissDirection.endToStart,
                    onDismissed: (direction) {
                            setState(() {articles.removeAt(index);});},
                    background: Container( color: Colors.orange, child:
Row( children: [Text("Swipe to remove")], ) //Row ), // Container
                    confirmDismiss: (direction) async {
                        return await showDialog(context: context,
                                                builder: (context) {
 return AlertDialog(title: Text('Are you sure to remove ? '),);});
                    }, // confirmDismiss
                ); // Dismissible
        } // itemBuilder
) //ListView.builder
```



Scrolling widgets

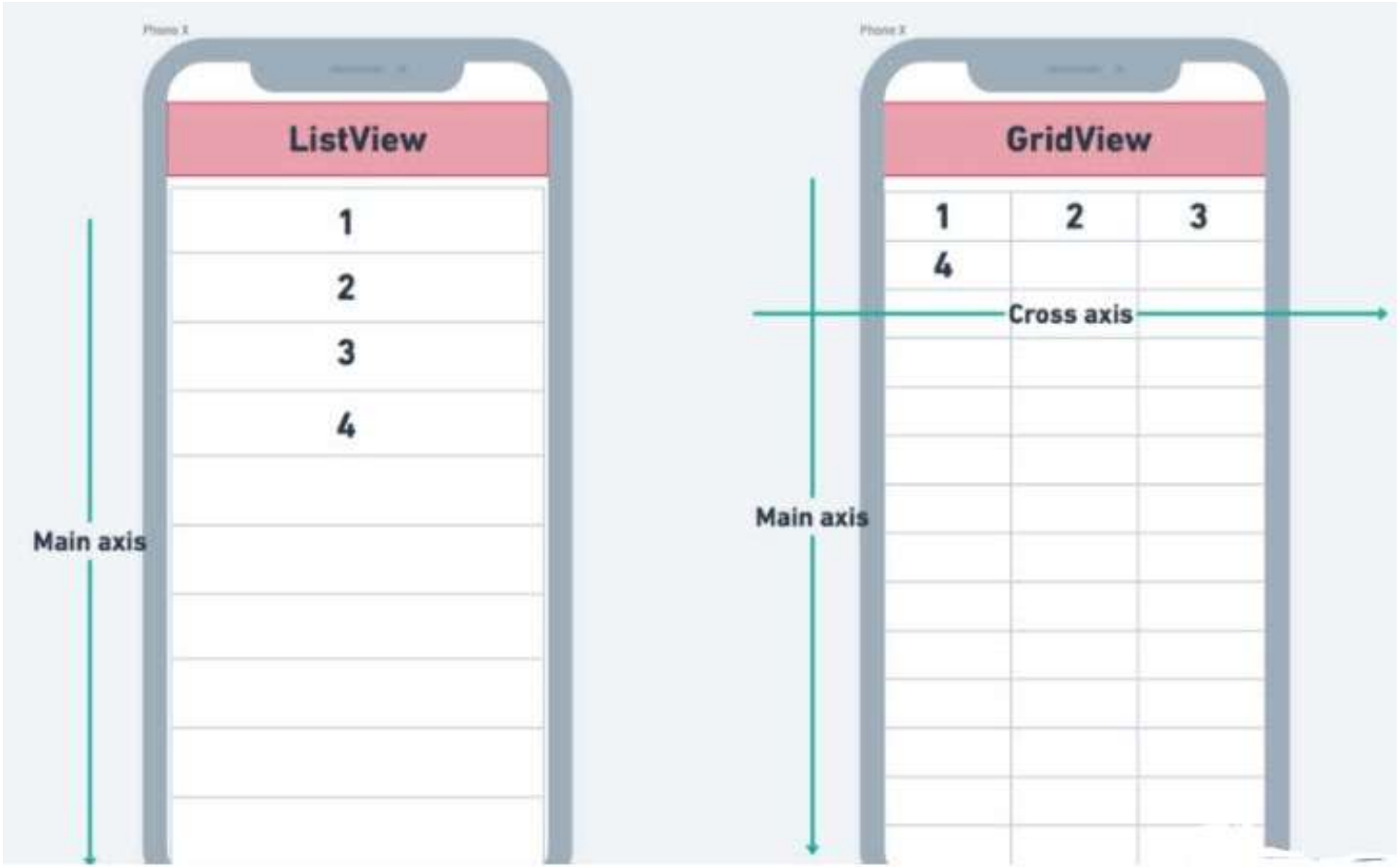
- **GridView:** a scrolling widget that displays a list of items as a 2D array (i.e. items are shown in a table format).

- **Examples:**



Scrolling widgets

- **GridView vs ListView:**



Scrolling widgets

- **Implementation of GridView:**

```
body: GridView (
        itemCount: articles.length,
        gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
                crossAxisCount: 3, // Number of items displayed horizontally
                mainAxisSpacing: 16, // Add space between  items on the main axis
                crossAxisSpacing: 16, // Add space between  items on the cross axis
                scrollDirection: Axis.vertical , // Axis.horizontal ),
        children: [
                Image.network('https://image=1'),
                Image.network('https://image=2'),
                Image.network('https://image=3'),
                Image.network('https://image=4'),
        ]
) // GridView
```

Scrolling widgets

- **GridView.builder:** creates a grid of widgets with a large (or infinite) number of children.


```
GridView.builder(
        scrollDirection: Axis.vertical, // Axis.horizontal
        itemCount: articles.length,
        gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
                crossAxisCount: 3, // Number of items displayed horizontally
                mainAxisSpacing: 16, // Add space between items on the main axis
                crossAxisSpacing: 16, // Add space between items on the cross axis
                ),
        itemBuilder: (BuildContext context, int index) {
                        List<Widget> L = itemArticles();
                        return L[index];
                },
), // GridView.builder
```