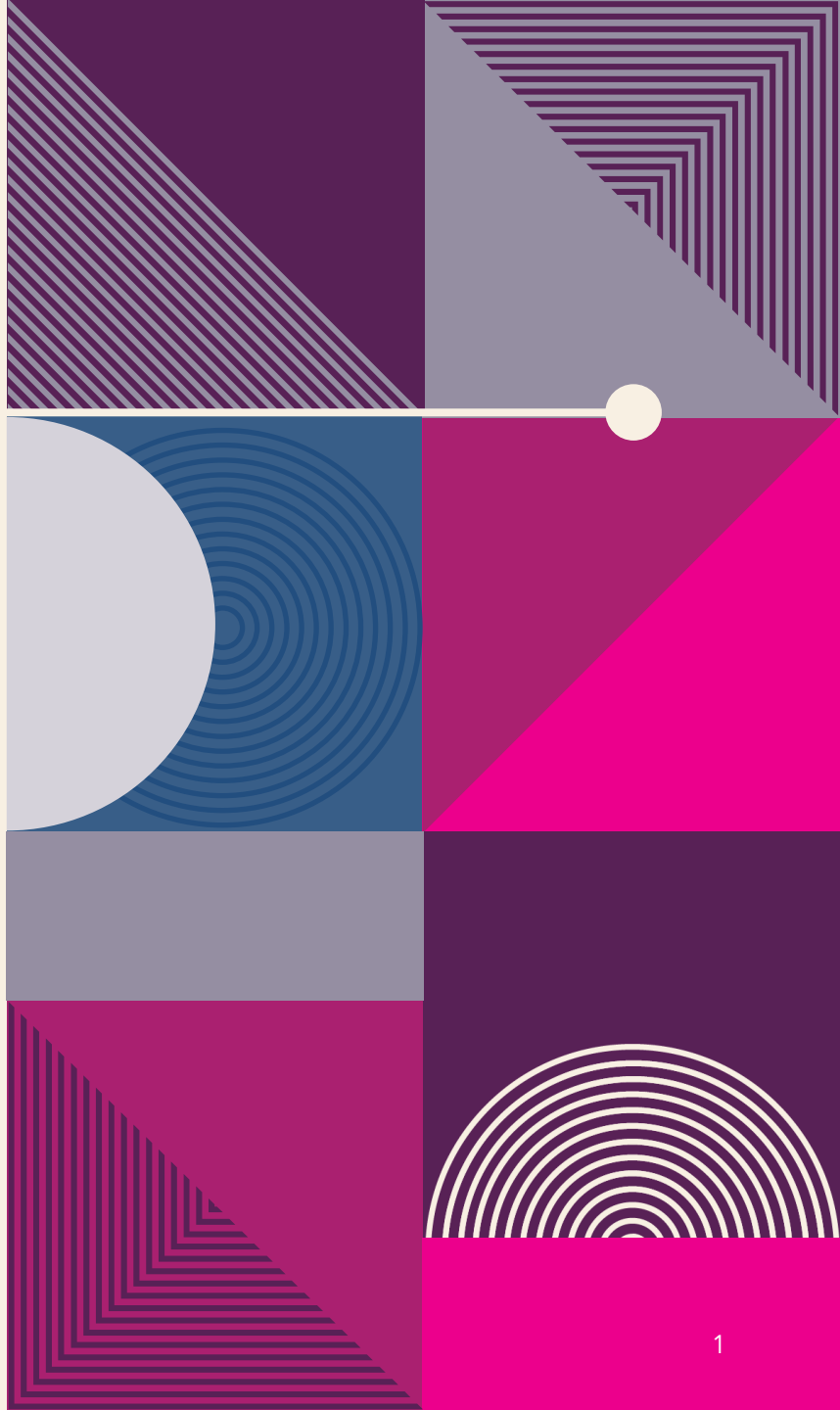
Using Common Widgets



USING BASIC WIDGETS

1. **Scaffold:** implements the basic Material Design **visual layout**, allowing you to add various widgets such as **AppBar**, **BottomAppBar**, **FloatingActionButton**, **Drawer**, **SnackBar**, **BottomSheet**.
2. **AppBar:** The AppBar widget usually contains the standard **title**, **toolbar**, **leading**, and **actions** properties (along with buttons).

- **title:** The title property is typically implemented with a **Text widget**.
- **leading** : displayed before the **title** property. **Usually** this is an **IconButton** or **BackButton**.
- **actions** : displayed to the **right** of the **title property**. It's a list of widgets aligned to the upper right of an **AppBar** widget usually with an **IconButton** or **PopupMenuButton**.
- **flexibleSpace** The flexibleSpace property is **stacked** behind the **Toolbar** or TabBar widget.

1. **SafeArea** : The SafeArea widget automatically adds sufficient padding to the child widget to avoid intrusions by the operating system.
2. **Container:** The Container widget is a commonly used widget that allows **customization of its child widget**. You can easily add properties such as **color, width, height, padding, margin, border, constraint, alignment, transform** .
3. **Text :** The Text widget is used to display a **string of characters**.

6. RichText: The RichText widget is a great way to display text **using multiple styles**.

7. Column: A Column widget displays its children **vertically**. It takes a children property containing an array of **List<Widget>,** meaning you can add multiple widgets. Each child widget can be embedded in an **Expanded** widget to fill the available space. CrossAxisAlignment, MainAxisAlignment, and MainAxisSize can be used to align and size how much space is occupied on the **main axis**.

8. Row: A Row widget displays its children **horizontally**. It takes a children property containing an array of **List<Widget>.** The same properties that the Column contains are applied to the Row widget.

9. Buttons : such as ElevatedButton, FloatingActionButton, TextButton, IconButton, PopupMenuButton, and ButtonBar.

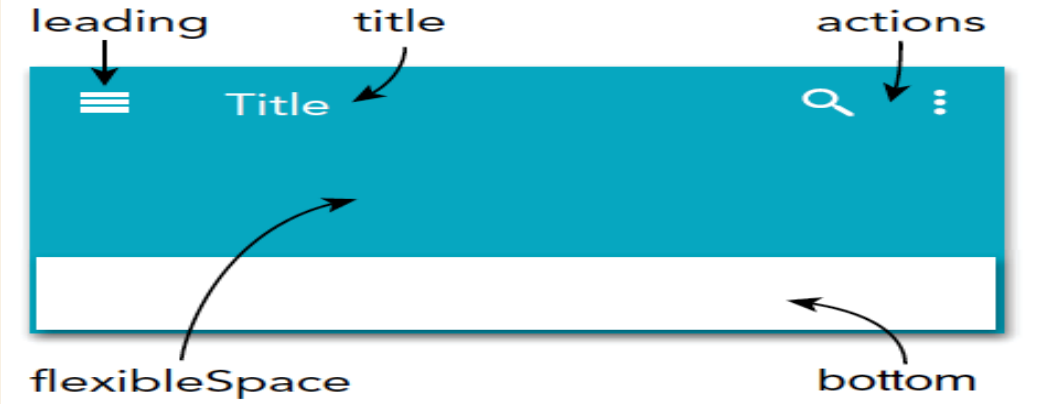
Adding AppBar Widgets

Create a new Flutter project called **ch6_basics**. You can follow the instructions from **LECTRUE 4**. For this project, you need to **create** the **pages folder** only.

1. Open the **main.dart** file. Change the **primarySwatch** property from **blue** to lightGreen.

```
primarySwatch: Colors.lightGreen,
```

2. Open the **home.dart** file. **Start** by customizing the **AppBar** widget properties.



3. Add to the **AppBar** a **leading IconButton**. If you **override** the leading property, it is usually an IconButton or BackButton.

```
leading: IconButton(
  icon: Icon(Icons.menu),
  onPressed: () {},
),
```

4. The **title** property is usually a **Text** widget, you have already added the Text widget to the title property; if not, add the Text widget with a value of 'Home'.

```
title: Text('Home'),
```

1. The **actions** property takes a **list of widgets**; add two **IconButton** widgets.

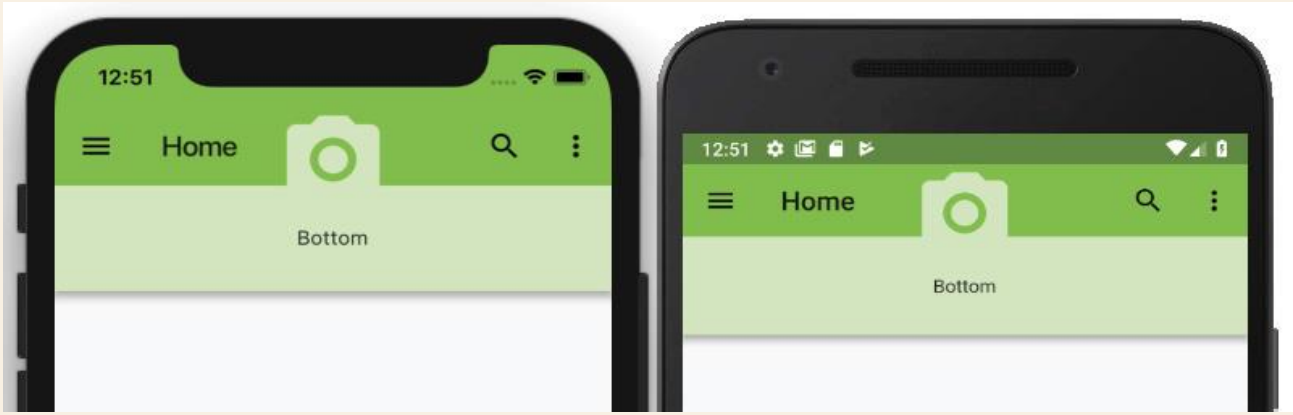
```
actions: <Widget>[
  IconButton(
    icon: Icon(Icons.search),
    onPressed: () {},
  ),
  IconButton(
    icon: Icon(Icons.more_vert),
    onPressed: () {},
  ),
],
```

1. Because you are using an **Icon** for the **flexibleSpace** property, let's **add** a **SafeArea** and an **Icon** as a **child**.

```
flexibleSpace: SafeArea(
  child: Icon(
    Icons.photo_camera,
    size: 75.0,
    color: Colors.white70,
  ),
),
```

1. Add a **PreferredSize** for the **bottom** property with a Container for a child.

```
bottom: PreferredSize(
  child: Container(
    color: Colors.lightGreen.shade100,
    height: 75.0,
    width: double.infinity,
    child: Center(
      child: Text('Bottom'),
    ),
  ),
  preferredSize: Size.fromHeight(75.0),
),
```



Adding a **SafeArea** to the **Body**

Continue modifying the **home.dart** file.

Add a **Padding** widget to the **body property** with a **SafeArea** as a **child**. add a **SingleChildScrollView** as a child of the **SafeArea**. The **SingleChildScrollView** allows the user to scroll and view hidden widgets;

```
body: Padding(
  padding: EdgeInsets.all(16.0),
  child: SafeArea(
    child: SingleChildScrollView(
      child: Column(
        children: <Widget>[
                    ],
                ),
            ),
      ),
),
```

Adding a Container

Continue modifying the **home.dart** file.

1. **In last step** for the **Column children**, add the call to the **ContainerWithBoxDecorationWidget()** widget class, which you will **create next**. Make sure the widget class uses the **const** keyword to take advantage of caching (**performance**).

```
body: Padding(
  padding: EdgeInsets.all(16.0),
  child: SafeArea(
    child: SingleChildScrollView(
      child: Column(
        children: <Widget>[
          const ContainerWithBoxDecorationWidget(),
],
                ),
            ),
     ),
),
```

1. Create the **ContainerWithBoxDecorationWidget() widget class after** class Home extends StatelessWidget {...}.

```
class ContainerWithBoxDecorationWidget extends StatelessWidget {
  const ContainerWithBoxDecorationWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: <Widget>[
        Container(),
              ],
       );
    }
}
```

2. Start **adding properties to the Container** by adding a **height** of **100.0 pixels**. Then go to the next line to add the **decoration** property, which accepts a **BoxDecoration** class. The BoxDecoration class provides different ways to **draw a box**.

```
Container(
  height: 100.0,
  decoration: BoxDecoration(),
),
```

3. Using the named constructor **BorderRadius.only()** allows you to control the sides to draw **round corners**.

```
decoration: BoxDecoration(
    borderRadius: BorderRadius.only(
      bottomLeft: Radius.circular(100.0),
      bottomRight: Radius.circular(10.0),
  ),
),
```

- The BoxDecoration also supports a gradient property.

```
gradient: LinearGradient(
  begin: Alignment.topCenter,
  end: Alignment.bottomCenter,
  colors: [
    Colors.white,
    Colors.lightGreen.shade500,
  ],
),
```

4. The boxShadow property is a great way to customize a shadow, and it takes a list of BoxShadows, called List<BoxShadow>.

- For the **BoxShadow**, set the **color**, **blurRadius**, and **offset** properties.

```
boxShadow: [
  BoxShadow(
    color: Colors.grey,
    blurRadius: 10.0,
    offset: Offset(0.0, 10.0),
  ),
],
```

5. Add a **Center** widget as a child of the **Container**, and add to the Center widget child a **RichText** widget.

- By using the **RichText** widget and combining different **TextSpan** objects

```
child: Center(
  child: RichText(
    text: TextSpan(
      text: 'Flutter World',
      style: TextStyle(
        fontSize: 24.0,
        color: Colors.deepPurple,
        decoration: TextDecoration.underline,
        decorationColor: Colors.deepPurpleAccent,
        decorationStyle: TextDecorationStyle.dotted,
        fontStyle: FontStyle.italic,
        fontWeight: FontWeight.normal,
      ),
      children: <TextSpan>[
        TextSpan(
          text: ' for',
        ),
        TextSpan(
          text: ' Mobile',
          style: TextStyle(
              color: Colors.deepOrange,
              fontStyle: FontStyle.normal,
              fontWeight: FontWeight.bold),
        ),
      ],
    ),   ), ),
```

Full ContainerWithBoxDecorationWidget() widget class source code:

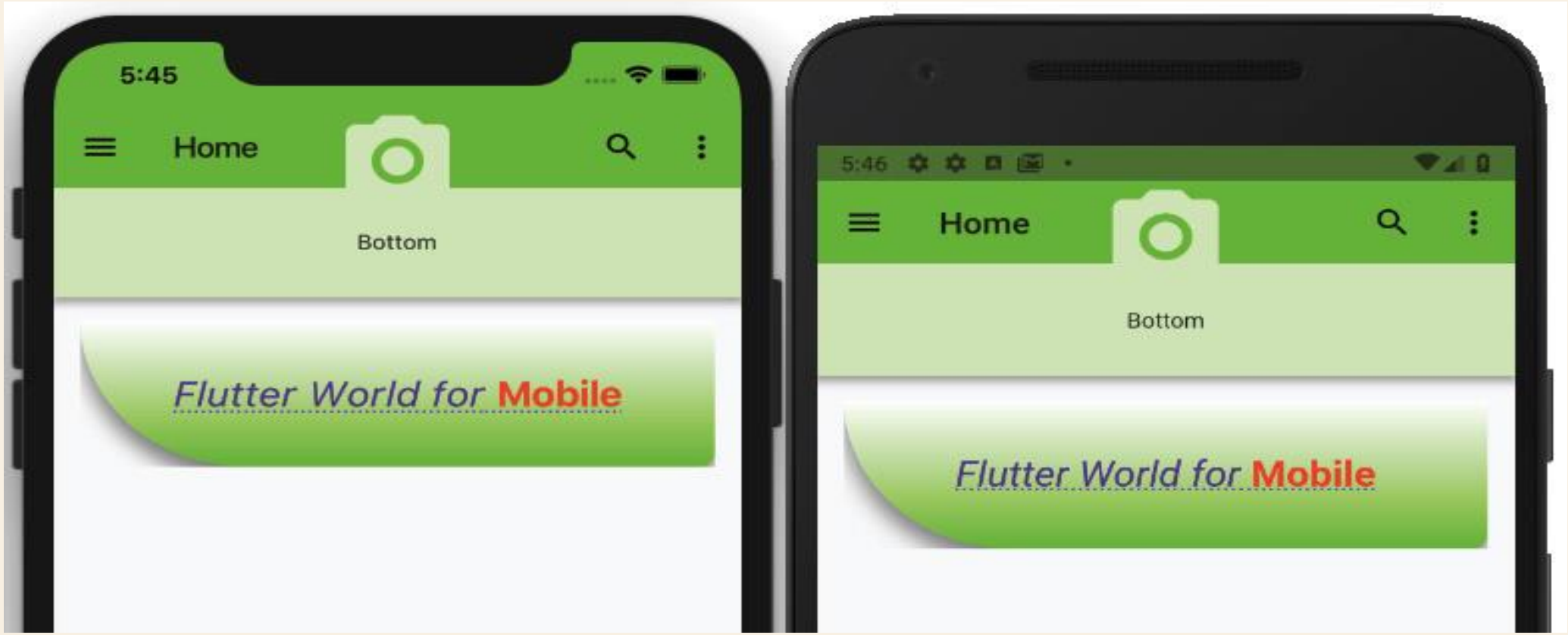
```
class ContainerWithBoxDecorationWidget extends StatelessWidget {
  const ContainerWithBoxDecorationWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: <Widget>[
        Container(
          height: 100.0,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.only(
              bottomLeft: Radius.circular(100.0),
              bottomRight: Radius.circular(10.0),
            ),
```

```
gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [
              Colors.white,
              Colors.lightGreen.shade500,
            ],
          ),
          boxShadow: [
            BoxShadow(
              color: Colors.grey,
              blurRadius: 10.0,
              offset: Offset(0.0, 10.0),
            ),
          ],
        ),
```

```
child: Center(
            child: RichText(
              text: TextSpan(
                text: 'Flutter World',
                style: TextStyle(
                  fontSize: 24.0,
                  color: Colors.deepPurple,
                  decoration: TextDecoration.underline,
                  decorationColor: Colors.deepPurpleAccent,
                  decorationStyle: TextDecorationStyle.dotted,
                  fontStyle: FontStyle.italic,
                  fontWeight: FontWeight.normal,
                ),
```

```
 children: <TextSpan>[
                    TextSpan(
                      text: ' for',
                    ),
                    TextSpan(
                      text: ' Mobile',
                      style: TextStyle(
                          color: Colors.deepOrange,
                          fontStyle: FontStyle.normal,
                          fontWeight: FontWeight.bold),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ],
      );
    }
}
```


5:45
Home
Bottom
Flutter World for Mobile
5:46
Home
Bottom
Flutter World for Mobile

Adding **Column**, **Row**, and **Nesting the Row and Column** together as Widget Classes

1. **Add** the **widget class** names **ColumnWidget**(), **RowWidget**(), and **ColumnAndRowNestingWidget**() to the **Column** children widget list. The **Column widget** is located in the **body property**. **Add** a **Divider()** widget between each widget class name. Make sure each widget class uses the **const** keyword.

```
body: Padding(
  padding: EdgeInsets.all(16.0),
  child: SafeArea(
    child: SingleChildScrollView(
      child: Column(
        children: <Widget>[
          //ContainerWithBoxDecorationWidget
          const ContainerWithBoxDecorationWidget(),
          Divider(),
          //ColumnWidget,
          const ColumnWidget(),
          Divider(),
          //RowWidget,
          const RowWidget(),
          Divider(),
          //ColumnAndRowNestingWidget,
          const ColumnAndRowNestingWidget(),
        ],
      ),
    ),  ), ),
```

2. Create the ColumnWidget() widget class after the ContainerWithBoxDecorationWidget() widget class.

```
class ColumnWidget extends StatelessWidget {
  const ColumnWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment:CrossAxisAlignment.center,
      mainAxisAlignment:MainAxisAlignment.spaceEvenly,
      mainAxisSize: MainAxisSize.max,
      children: <Widget>[
        Text('Column 1'),
        Divider(),
        Text('Column 2'),
        Divider(),
        Text('Column 3'),
      ],
    );
  }
}
```

3. Create the **RowWidget**() widget **class** after the **ColumnWidget**() widget **class**.

```
class RowWidget extends StatelessWidget {
  const RowWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Row(
      crossAxisAlignment: CrossAxisAlignment.start,
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      mainAxisSize: MainAxisSize.max,
      children: <Widget>[
        Row(
          children: <Widget>[
            Text('Row 1'),
            Padding(padding: EdgeInsets.all(16.0),),
            Text('Row 2'),
            Padding(padding: EdgeInsets.all(16.0),),
            Text('Row 3'),
          ],
        ),
      ],
    );
  }
}
```

4. Create the ColumnAndRowNestingWidget() widget class after the RowWidget() widget class.

```
class ColumnAndRowNestingWidget extends StatelessWidget {
  const ColumnAndRowNestingWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      mainAxisSize: MainAxisSize.max,
      children: <Widget>[
        Text('Columns and Row Nesting 1',),
        Text('Columns and Row Nesting 2',),
        Text('Columns and Row Nesting 3',),
        Padding(padding: EdgeInsets.all(16.0),),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            Text('Row Nesting 1'),
            Text('Row Nesting 2'),
            Text('Row Nesting 3'),
          ],
        ),
      ],
    );
  }
}
```