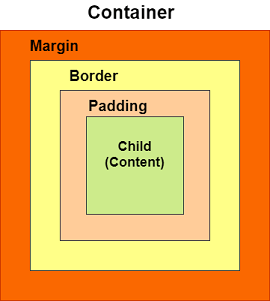
# Flutter Container

The container in Flutter is a **parent widget that can contain multiple child widgets** and manage them efficiently through width, height, padding, background color, etc. It is a widget that combines common painting, positioning, and sizing of the child widgets. It is also a class to store one or more widgets and position them on the screen according to our needs



### container class

Container({Key key,

           AlignmentGeometry alignment,

           EdgeInsetsGeometry padding,

           Color color,

**double** width,

**double** height,

           Decoration decoration,

           Decoration foregroundDecoration,

           BoxConstraints constraints,

           Widget child,

           Clip clipBehavior: Clip.none

});

## Properties of Container widget

**1. child:** This property is used to store the child widget of the container. Suppose we have taken a Text widget as its child widget that can be shown in the below example:

Container(

    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

**2. color:** This property is used to set the **background color of the text**. It also changes the background color of the entire container. See the below example:

Container(

    color: Colors.green,

    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

**3. height and width:** This property is used to set the container's height and width according to our needs. By default, the container always takes the space based on its child widget. See the below code:

Container(

    width: 200.0,

    height: 100.0,

    color: Colors.green,

    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

**4. margin:** This property is used to surround the e**mpty space around the container**. We can observe this by seeing white space around the container. Suppose we have used the **EdgeInsets.all(25)** that set the equal margin in all four directions, as shown in the below example:

Container(

    width: 200.0,

    height: 100.0,

    color: Colors.green,

    margin: EdgeInsets.all(20),

    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

**5. padding:** This property is used to **set the distance** between the border of the container (all four directions) and its child widget. We can observe this by seeing the space between the container and the child widget. Here, we have used an EdgeInsets.all(35) that set the space between text and all four container directions:

Container(

    width: 200.0,

    height: 100.0,

    color: Colors.green,

    padding: EdgeInsets.all(35),

    margin: EdgeInsets.all(20),

    child: Text("Hello! I am in the container widget", style: TextStyle(fontSize: 25)),

)

Flutter image program

1.After creating flutter application create a assets folder in main root and copy any image file into created new assets folder

2.open pubspec.yaml file to declare the image file

flutter:

  # The following line ensures that the Material Icons font is

  # included with your application, so that you can use the icons in

  # the material Icons class.

  uses-material-design: true

  assets:

    -  "assets/pe.jpg"

Stack

**Flutter – Stack Widget**

**Stack** widget is a built-in widget in flutter SDK which allows us to make a layer of widgets by putting them on top of each other

### **Constructor of Stack Class:**

Stack(

{Key key,

AlignmentGeometry alignment: AlignmentDirectional.topStart,

TextDirection textDirection,

StackFit fit: StackFit.loose,

Overflow overflow: Overflow.clip,

Clip clipBehavior: Clip.hardEdge,

List<Widget> children: const <Widget>[]}

)

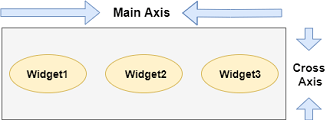
### **Properties of Stack Widget:**

* **alignment:**This property takes a parameter of Alignment Geometry, and controls how a child widget which is non-positioned or partially-positioned will be aligned in the *Stack*.
* **clipBehaviour:**This property decided whether the content will be clipped or not.
* **fit:**This property decided how the non-positioned children in the *Stack* will fill the space available to it.
* **overflow:**This property controls whether the overflow part of the content will be visible or not,
* **textDirection:**With this property, we can choose the text direction from right to left. or left to right.

# Flutter Row and Column

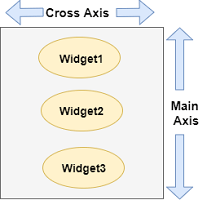
Row and column are the two essential widgets in [Flutter](https://www.javatpoint.com/flutter) that allows developers to **align children horizontally and vertically**

We can control how a row widget aligns its children based on our choice using the property **crossAxisAlignment** and **mainAxisAlignment**. The row's **cross-axis** will run **vertically**, and the **main axis** will run **horizontally**



We can align the row's children widget with the help of the following properties

* **start:** It will place the children from the starting of the main axis.
* **end:** It will place the children at the end of the main axis.
* **center:** It will place the children in the middle of the main axis.
* **spaceBetween:** It will place the free space between the children evenly.
* **spaceAround:** It will place the free space between the children evenly and half of that space before and after the first and last children widget.
* **spaceEvenly:** It will place the free space between the children evenly and before and after the first and last children widget.
* **This widget arranges its children in a vertical direction on the screen**
* We can also control how a column widget aligns its children using the property mainAxisAlignment and crossAxisAlignment. The column's **cross-axis**will run **horizontally**, and the **main axis** will run **vertically**.



Flutter Text

A Text is a widget in Flutter that allows us to **display a string of text with a single line in our application**.

**MaterialApp** widget that calls the home screen using the **MyTextPage()** class. This class contains the **scaffold** widget, which has **appBar** and **body** where we have used the **Text** widget to display the title and body, respectively.