For Margin And Padding In Flutter, They Accept EdgeInsets-Values:

- EdgeInsets.all();
- EdgeInsets.symmetric(horizontal, vertical);
- EdgeInsets.only(values-here);
- EdgeInsets.fromLTRB(left, top, right, bottom);

****

For Making Perfect Circle Of Images Using Container We Can Use:

```
Container(
            width: 150.0,
            height: 100.0,
            clipBehavior: Clip.hardEdge,
            padding: EdgeInsets.all(24.0),
            decoration: BoxDecoration(
              shape: BoxShape.circle,
              color: Colors.white,
              image: DecorationImage(
                image: AssetImage('assets/cake.jpg'),
                fit: BoxFit.fill
              ),
            ),
          ),
```

But We Must set the clipBehavior (Any Value Except None) and decoration.

****

OR We Can Use ClipRRect-widget:

```
ClipRRect(
            borderRadius: BorderRadius.circular(75.0),
            child: Image(
              // color: Colors.transparent,
              image: AssetImage(
                "assets/cake.jpg",
              ),
              width: 150.0,
              height: 150.0,
              fit: BoxFit.fill,
            ),
          ),
```

****

To Set The Widget In Safe Area Of Mobile Shape We Can Use SafeArea-Widget.

****

Always when your widget does not listen to the constraints that you try to set up, first try to wrap it with `Align`

****

```
Container(
height: double.infinity,
width: double.infinity,
color: Colors.yellowAccent,
child: Text("Hi"),
),
```

****

```
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(title: Text('Container.foregroundDecoration')),
body: Container(
height: double.infinity,
width: double.infinity,
decoration: BoxDecoration(color: Colors.yellowAccent),
foregroundDecoration: BoxDecoration(
color: Colors.red.withOpacity(0.5),
),
child: Text("Hi"),
),
);
}
```

****

gradient

There are three types of gradients: `LinearGradient`, `RadialGradient` and `SweepGradient`

****

backgroundBlendMode

`backgroundBlendMode` is the most complex property of `BoxDecoration`. It's responsible for mixing together colors/gradients of `BoxDecoration` and whatever `BoxDecoration` is on top of.

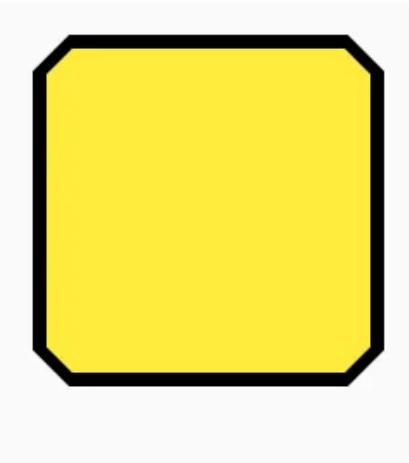
With `backgroundBlendMode` you can use a long list of algorithms specified in `BlendMode` enum.

****

`backgroundBlendMode` does not affect only the `Container` it's located in.

`backgroundBlendMode` changes the color of anything that is up the widget tree from the `Container`.
The following code has a parent `Container` that draws an `image` and child `Container` that uses `backgroundBlendMode`
****

```
Scaffold(
appBar: AppBar(title: Text('shape: BeveledRectangleBorder')),
body: Center(
child: Material(
shape: const BeveledRectangleBorder(
borderRadius: BorderRadius.all(Radius.circular(20)),
side: BorderSide(color: Colors.black, width: 4),
),
color: Colors.yellow,
child: Container(
height: 200,
width: 200,
),
),
),
);
```



****

```
Scaffold(
  appBar: AppBar(title: Text('SliverFillRemaining')),
  body: CustomScrollView(
    slivers: [
      SliverFillRemaining(
        hasScrollBody: false,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: const [
            FlutterLogo(size: 200),
            Text(
              'This is some longest text that should be centered'
              'together with the logo',
              textAlign: TextAlign.center,
            ),
          ],
        ),
      ),
    ],
  ),
);
```

****

Filling the remaining space

Apart from being useful for centering your content, `SliverFillRemaining` will fill the remaining viewport's free space. To do that this widget has to be placed in `CustomScrollView` and needs to be the last sliver

```
Scaffold(
appBar: AppBar(title: Text('SliverFillRemaining')),
body: CustomScrollView(
slivers: [
SliverList(
delegate: SliverChildListDelegate(const [
ListTile(title: Text('First item')),
ListTile(title: Text('Second item')),
ListTile(title: Text('Third item')),
ListTile(title: Text('Fourth item')),
]),
),
SliverFillRemaining(
hasScrollBody: false,
child: Container(
color: Colors.yellowAccent,
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: const [
FlutterLogo(size: 200),
Text(
'This is some longest text that should be centered'
'together with the logo',
textAlign: TextAlign.center,
),
],
),
),
),
],
),
);
```

****

For Column-Widget We Have: verticalDirection-property.

It Control How The Widgets Appear in Vertical Alignment:

- Up: it make them appear from bottom to top.
- Down (default): it make them appear from top to bottom.

****

When We Have Widget, And We Want To Make It Fill The Entire Space In Main Axis We Can use: ***Expanded-widget***.

****

In New Flutter Versions We don't have:

- FlatButton.
- RaisedButton.

****

If We Want To Set Global Settings For Theme, We Can Change The theme-attribute Of Material App. (ex: theme: ThemeData(

```
    primaryColor: Colors.red, // This Will Also Change The Color Of AppBar Background.

    accentColor: Colors.green, // This Will Also Change The Color Of FloatingActionButton-Widget.

    scaffoldBackgroundColor: Colors.green, // This Will Change The Background Color Of All Scaffolds In Our App.

    textTheme: TextTheme(

        body1: TextStyle(

            color: Colors.white, // This Will Change The Color Of All Text Widgets.


        ),

    ),
```

),).

****

We Can Use ThemeData.dark().copyWith(

primaryColor: Colors.red, // This Will Also Change The Color Of AppBar Background.

scaffoldBackgroundColor: Colors.green, // This Will Change The Background Color Of All Scaffolds In Our App.

);

This Will Copy All The ThemeData Of Dark Theme With The New Settings Of Colors.

The Dark Theme Has By Default White Color For Text-Widgets.

****

We Can Wrap Our Widgets With Theme-Widget To Override The Custom Theme That We Defined In Our MaterialApp.

But We Must Set The data-property To What We Want.

Ex: Theme(

data: ThemeData(...Here We Set The Attributes),

child: FloatingActionButton(...Data Here),

),

****

Always Remember To: Extract The Widgets, Constants, Styles, Data, And Anything (includes FontFamily, FontSize, Alignment, ...etc.).

****

When we extract values we should be attention what we extract: The Colors, The Styles, The Height, The Width, ...etc.

****

Note (To Remember): When Using Container-Widget Without Child, it will fit the size of parent, but When Set Child To It, it will resize depending on the size of child.

We Can Solve This By Many Ways:

- Set the Width-property to **double.infinity**.
- Wrap The Container With Column And Set crossAxisAlignment to CrossAxisAlignment.stretch.

****

We Must Set These Values Together:

```
crossAxisAlignment: CrossAxisAlignment.baseline,
textBaseline: TextBaseline.alphabetic,
```

****