Flutter: Buttons

A new set of basic material button widgets and themes have been added to Flutter. The original classes have been deprecated and will eventually be removed. The overall goal is to make buttons more flexible, and easier to configure via constructor parameters or themes.

The FlatButton, RaisedButton and OutlineButton widgets have been replaced by TextButton, ElevatedButton, and OutlinedButton respectively. Each new button class has its own theme: TextButtonTheme, ElevatedButtonTheme, and OutlinedButtonTheme. The original ButtonTheme class is no longer used. The appearance of buttons is specified by a ButtonStyle object, instead of a large set of widget parameters and properties. This is roughly comparable to the way that the appearance of text is defined with a TextStyle object. The new button themes are also configured with a ButtonStyle object. A ButtonStyle is itself just a collection of visual properties. Many of these properties are defined with MaterialStateProperty, which means that their value can depend on the button's state.

TextButton

You can simply use TextButton at a place such as AppBar to close the screen, inside the dialog for closing it, etc. You'll want to avoid placing this button inside any scrollable list for obvious UI/UX reasons.

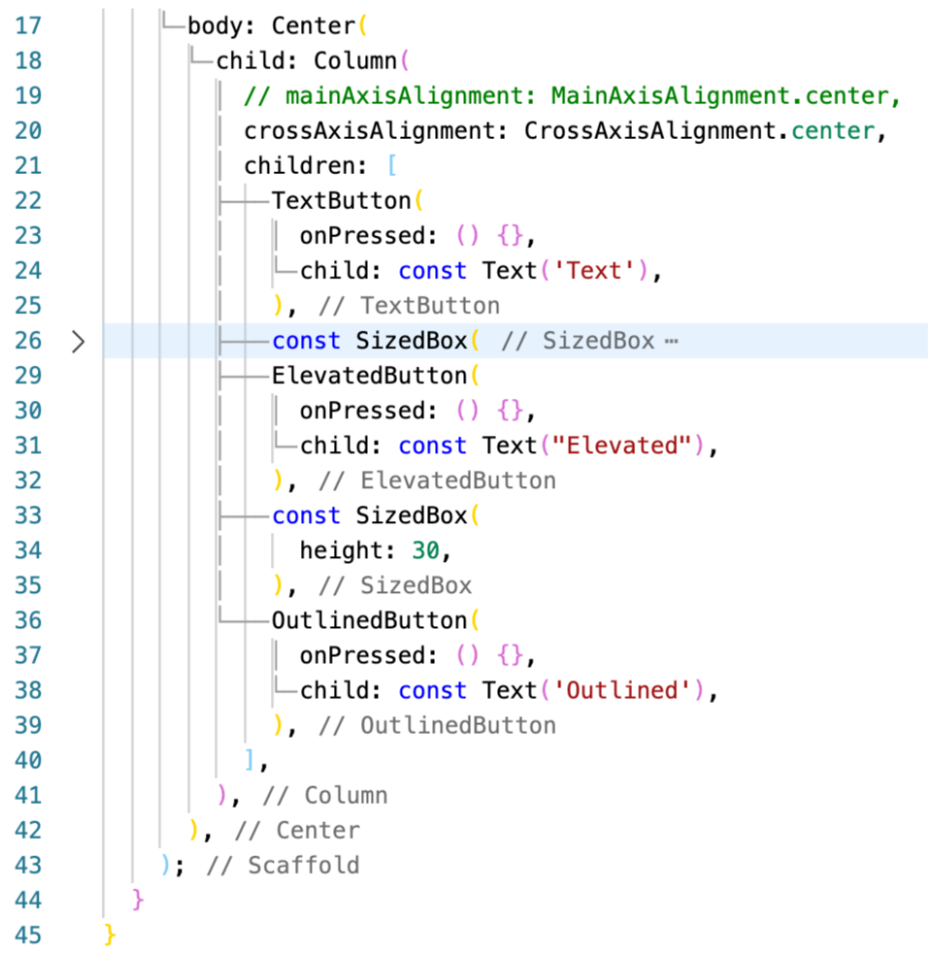
ElevatedButton

ElevatedButton is best suited in places where the app requires direct action from the user, such as confirming an order or booking a ticket. Since it's already elevated, you should avoid putting it over any other elevated widgets, such as cards.

OutlinedButton

In many ways, OutlinedButton is a mixture of TextButton and ElevatedButton. It's a TextButton if you remove the border and gives the impression of

ElevatedButton if you hover or press over it. This button is a medium-emphasis button meaning it can be used at the place where the action is important but not crucial.



```
      body: Center(
        child: Column(
          // mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            TextButton(
              onPressed: () {},
              child: const Text('Text'),
            ), // TextButton
            const SizedBox( // SizedBox ⋯
            ElevatedButton(
              onPressed: () {},
              child: const Text("Elevated"),
            ), // ElevatedButton
            const SizedBox(
              height: 30,
            ), // SizedBox
            OutlinedButton(
              onPressed: () {},
              child: const Text('Outlined'),
            ), // OutlinedButton
          ],
        ), // Column
      ), // Center
    ); // Scaffold
  }
}
```

<u>Properties used in buttons:</u>

onPressed: *This property is used for event handling. Code written that should be executed after the button is short pressed.*

child: *This property is used to put child on buttons. We can put widget like Text or icon inside the button.*

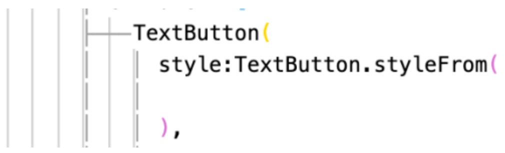
onLongPress: *Event-handling code when the button is long pressed.*

Primary: Colors.red→ *background in elevated button and foreground in outlined button.*

onPrimary:*foreground in elevated button and not used in outlined button.*

style: *This property is used to build your own button by specifying minimumsize, textstyle, primary(background color), onprimary(foreground color), padding, elevation, side, shape, onsurface.*

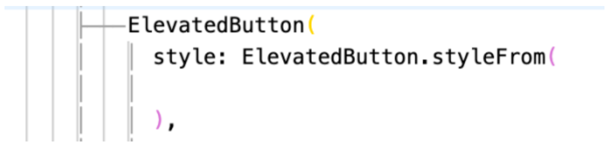
style in TextButton:



```
TextButton(
  style:TextButton.styleFrom(

  ),
```

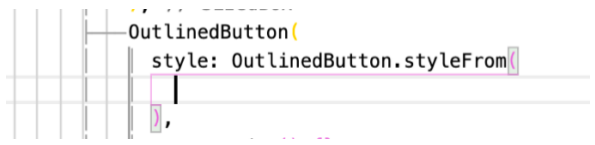
Style in ElevatedButton:



```
ElevatedButton(
  style: ElevatedButton.styleFrom(

  ),
```

Style in OutlinedButton:



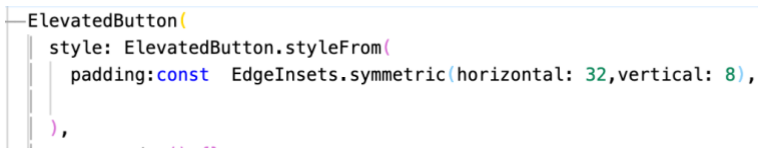
```
OutlinedButton(
  style: OutlinedButton.styleFrom(

  ),
```

Now lets learn about the attributes used in styleFrom().

<u>Padding in Buttton:</u>

Padding: EdgeInsets.symmetric(horizontal:32,vertical:8)



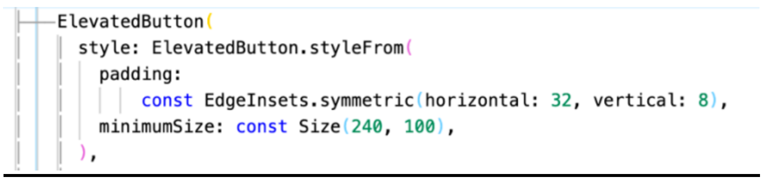
```
ElevatedButton(
  style: ElevatedButton.styleFrom(
    padding:const  EdgeInsets.symmetric(horizontal: 32,vertical: 8),

  ),
```

This will work same as in TextButton and OutlinedButton.

For giving specific size of Button:

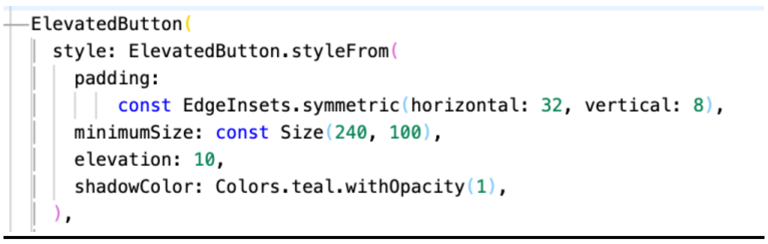
```
minimumSize: Size(240,100)
```



```
ElevatedButton(
  style: ElevatedButton.styleFrom(
    padding:
        const EdgeInsets.symmetric(horizontal: 32, vertical: 8),
    minimumSize: const Size(240, 100),
  ),
```

Adding Shadows to Buttons:

```
Elevation:8
```

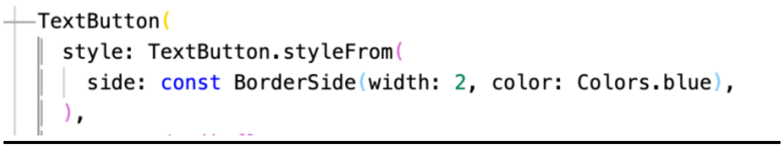


```
ElevatedButton(
  style: ElevatedButton.styleFrom(
    padding:
        const EdgeInsets.symmetric(horizontal: 32, vertical: 8),
    minimumSize: const Size(240, 100),
    elevation: 10,
    shadowColor: Colors.teal.withOpacity(1),
  ),
```

Here we have used shadowColor for color of shadow.

Border in Buttons:

```
Side:Borderside(width:2,color:Colors.blue)
```

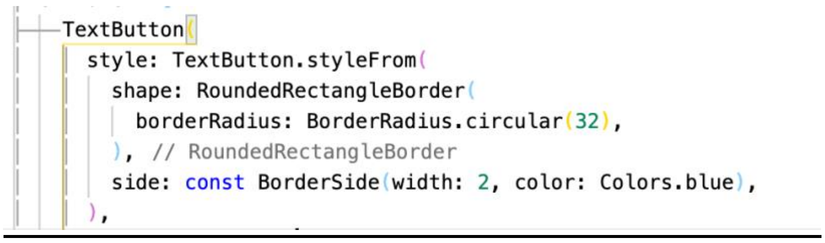


```
TextButton(
  style: TextButton.styleFrom(
    side: const BorderSide(width: 2, color: Colors.blue),
  ),
```

Changing Shape of Button:

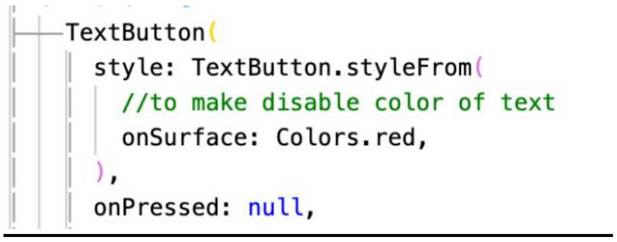
Shape:RoundRectangleBorder(

borderRadius:BorderRadius.circular(32))


```
TextButton(
  style: TextButton.styleFrom(
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(32),
    ), // RoundedRectangleBorder
    side: const BorderSide(width: 2, color: Colors.blue),
  ),
```

__Disable state of Buttons:__

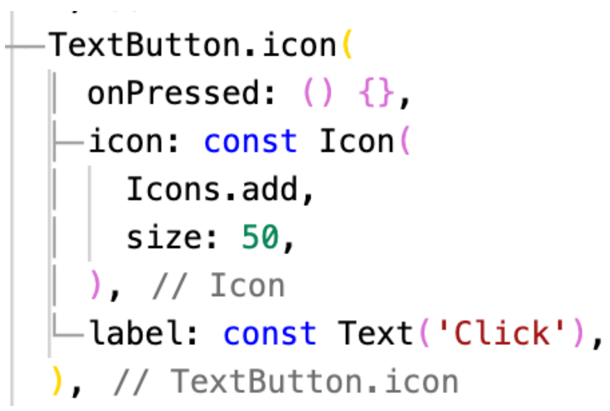
onPressed: null


```
TextButton(
  style: TextButton.styleFrom(
    //to make disable color of text
    onSurface: Colors.red,
  ),
  onPressed: null,
```

To create icon Button:

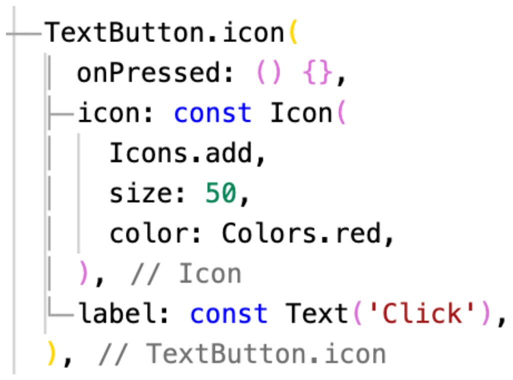
To create a cion button you can use:

1. TextButton.icon
2. ElevatedButton.icon
3. OutlinedButton.icon



```
TextButton.icon(
  onPressed: () {},
  icon: const Icon(
    Icons.add,
    size: 50,
  ), // Icon
  label: const Text('Click'),
), // TextButton.icon
```

You can use style properties here to change the style of Button. We can specify separate color for Icon by using color attributes of Icon widgets.



```
TextButton.icon(
  onPressed: () {},
  icon: const Icon(
    Icons.add,
    size: 50,
    color: Colors.red,
  ), // Icon
  label: const Text('Click'),
), // TextButton.icon
```