How to create circular, rectangle, and rounded InkWell buttons in Flutter

Mike Vas Jul 13 · 3 min read

Source: https://flutterhq.com/blog/what-is-flutter-who-made-it-how-does-it-work-and-whats-in-its-future

The problem

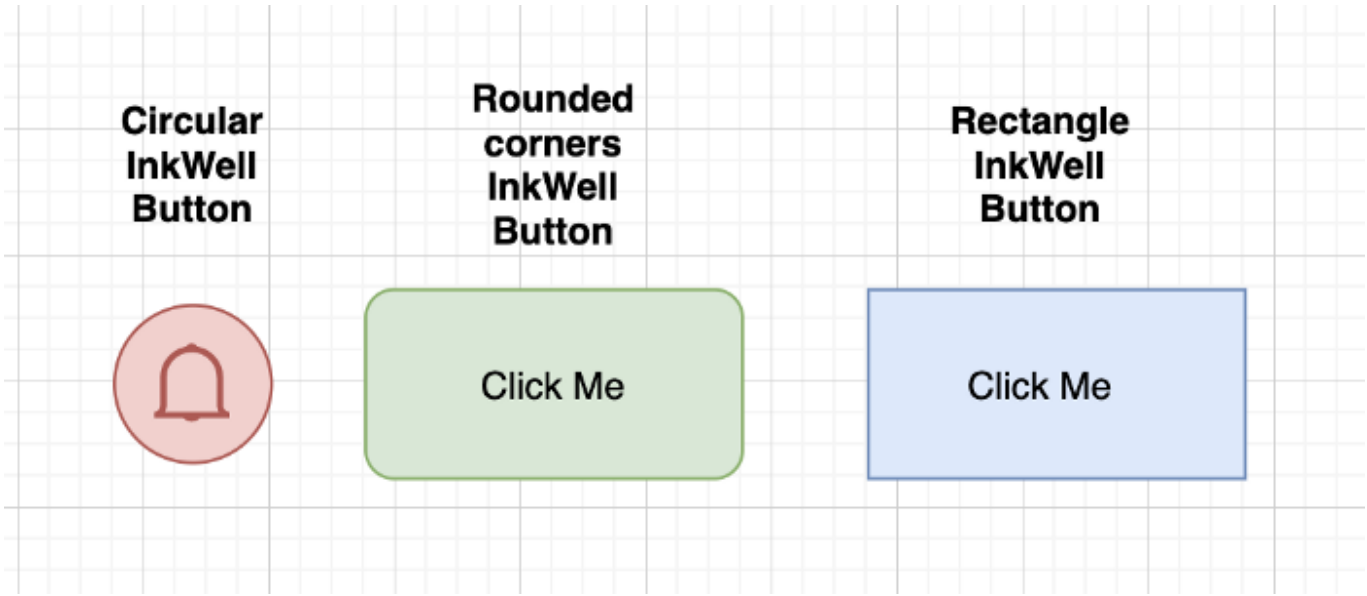


Image 2: Types of buttons

Creating differently shaped buttons is a problem because of the way how custom styling will be applied to them. For example, having a gradient background is really important for modern UI design. Once properly applied effects like splashing should work correctly, for example, not to spread like a square on the circular button, but rather to follow the circular structure.

The Solution

The solution is pretty nicely explained in the article Create gradient on button in Flutter by Aminullah Taj Muhammad. Please, read that article for more in-depth explanation, but in nutshell using *Ink* component would do the proper trick.

1. Having all proper effects on custom styling

How should we layout the components and their properties to achieve the wanted effect. Ink should be the parent component, it behaves similarly to Container component. Its child should be InkWell component. Components laid out in this way would allow having all proper effects like proper splashing or hovering the button, even with the gradient background.

```
Ink(
   width: 50,
   height: 50,
   decoration: BoxDecoration(
       gradient: LinearGradient(
           begin: Alignment.centerLeft,
           end: Alignment.centerRight,
           colors: [Colors.yellow, Colors.green],
       ),
   ), // LinearGradientBoxDecoration
   child: InkWell(
        onTap: () {},
        customBorder: _inkWellButtonModel.customBorder,
        child: Center(...),
        splashColor: Colors.red,
  ), // Red will correctly spread over gradient
),
```

2. Achieving correct shape

Once all button-related properties are correct, we can shape our button using *shape or borderRadius* property of BoxDecoration.

- Circular button:

```
BoxDecoration( shape: BoxShape.circle)
```

- Rectangle button:

```
BoxDecoration( shape: BoxShape.rectangle)
```

- Rounded button

```
BoxDecoration( borderRadius: 12)
```

3. Apply effects on the correct shape

In order not to have a rectangle spreading over a circular button proper ShapeBorder object should be created and applied.

- Circular button

```
ShapeBorder _customBorder = CircleBorder();
```

4. Final result

So, once everything combined for the circular button it should look like this:

```
Material(
    color: Colors.transparent,
    child: Ink(
        width: 50,
```

```
          height: 50,
          decoration: BoxDecoration(
              gradient: LinearGradient(
                  begin: Alignment.centerLeft,
                  end: Alignment.centerRight,
                  colors: [Colors.yellow, Colors.green],
              ),
              shape: BoxShape.circle,
          ), // LinearGradientBoxDecoration
          child: InkWell(
              onTap: () {},
              customBorder: CircleBorder(),
              child: Center(child: Text("Test"),),
              splashColor: Colors.red,
          ), // Red will correctly spread over gradient

    ),
    ),
```

If you would like to look at the whole example you can check out my, somewhat, generic InkWellButton:

mikevastech/flutter_bare_components

This repository should contain components found to be mostly used inside subracker_components project that can be…

github.com

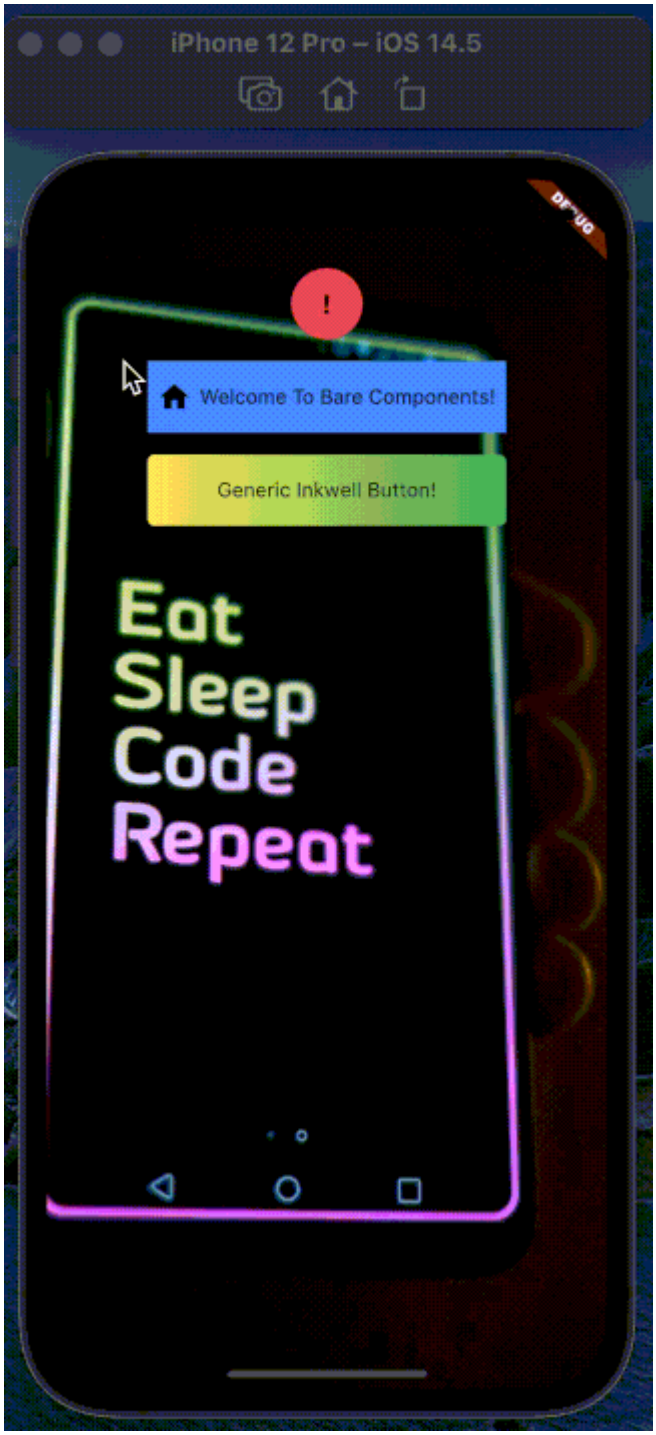
Or if you would like to see more generic components you can check out the whole repository:

mikevastech/flutter_bare_components

This repository should contain components found to be mostly used inside subracker_components project that can be…

github.com

Real-life example



Support

If you would like to support me, you could follow me on Instagram or Twitter, or even GitHub.