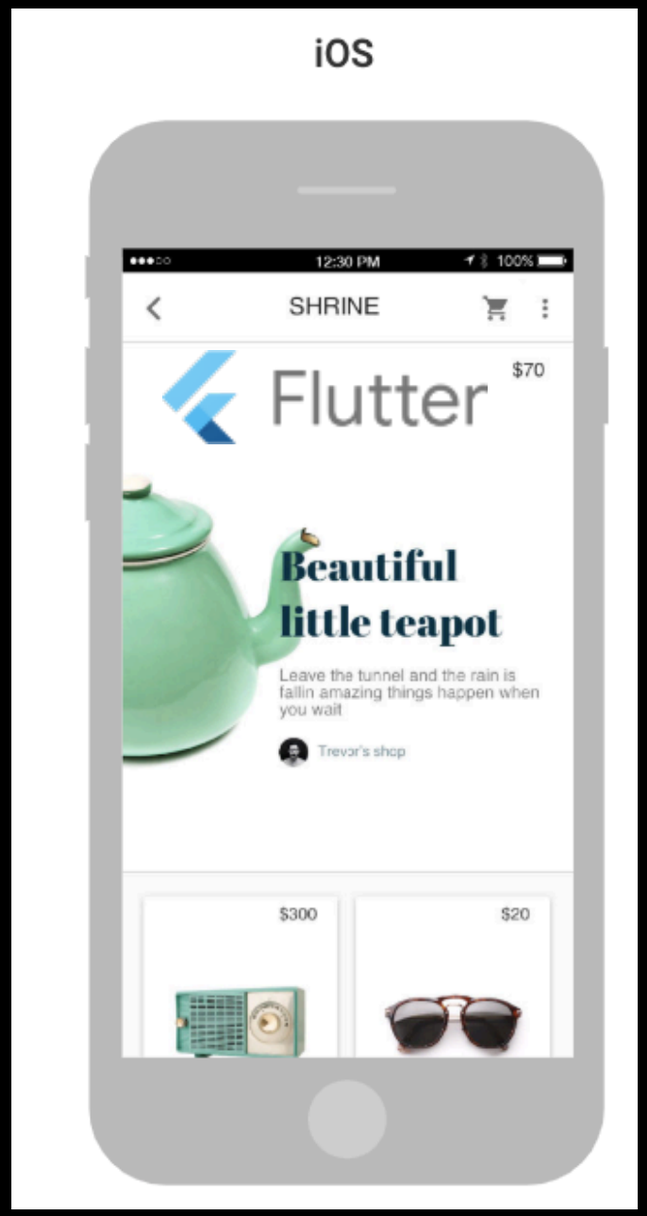
DANIEL GRAHAM

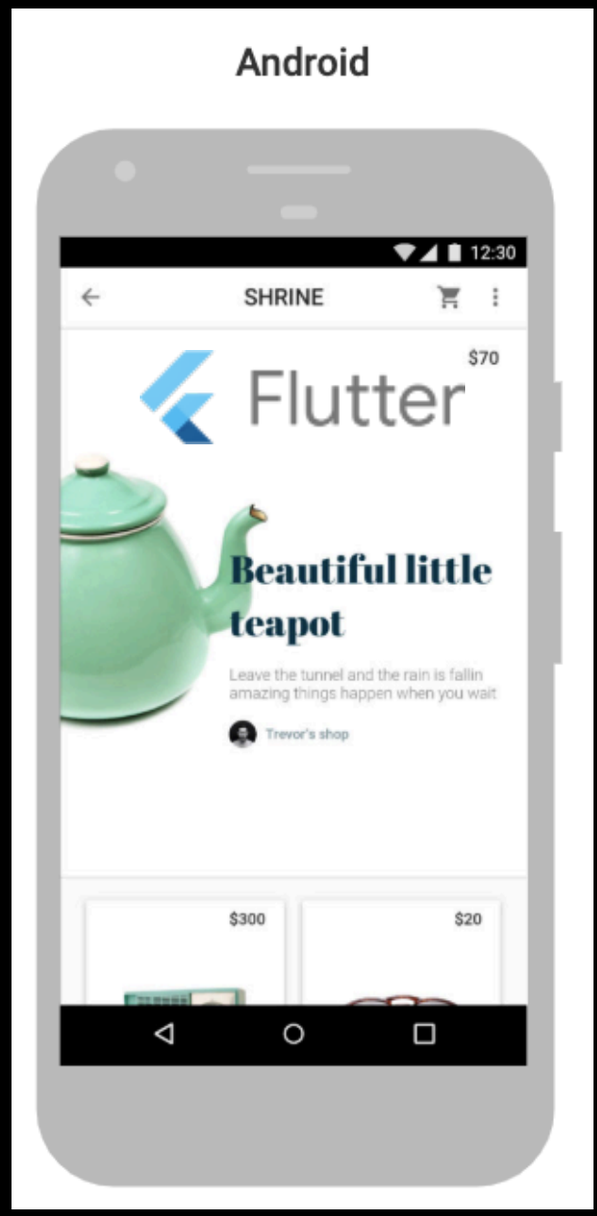
FLUTTER

INTRODUCTION

WHAT IS FLUTTER

- Is a software development kit that allows you to build application for for both android and IOS devices

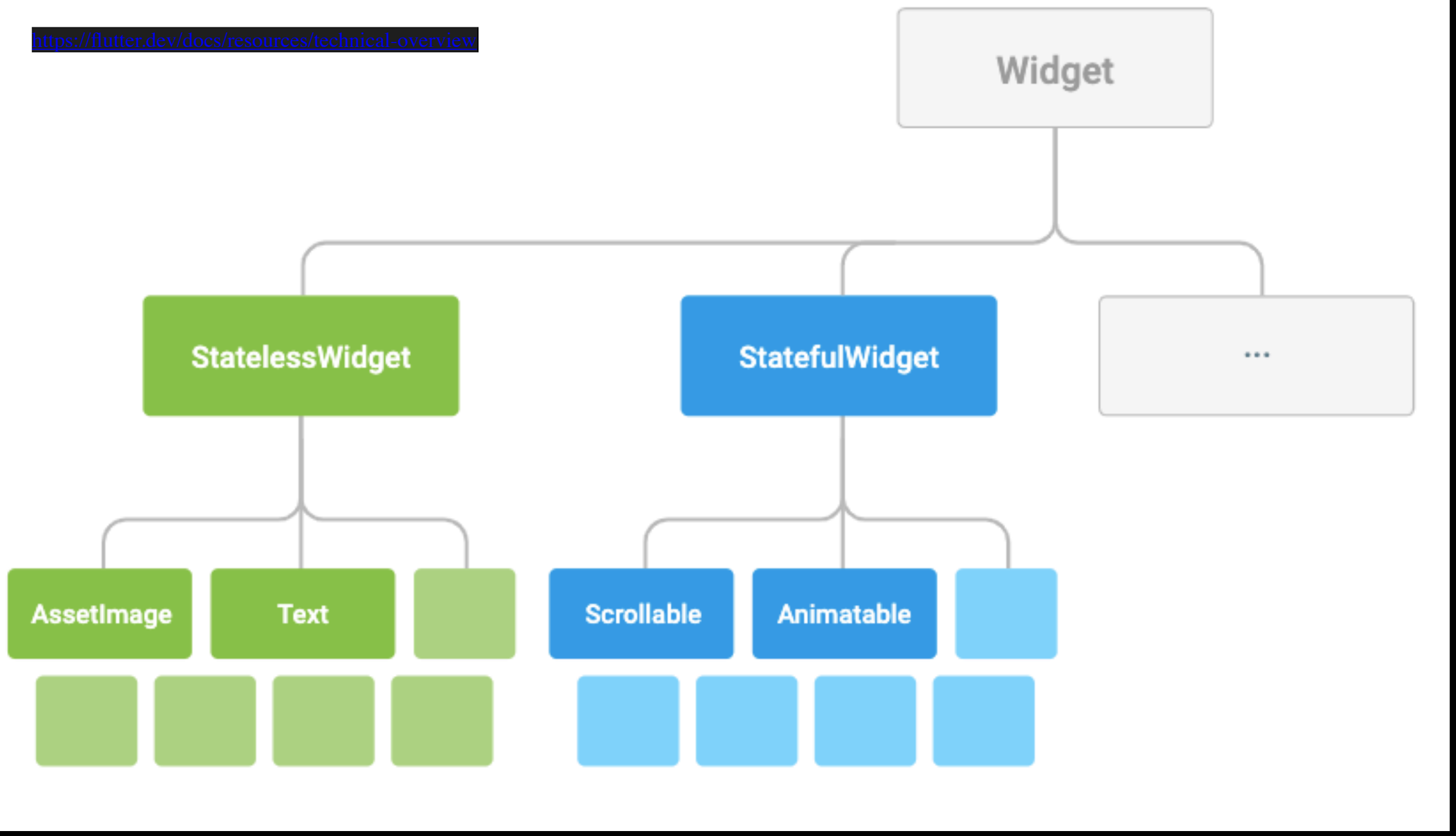




https://flutter.dev/docs/resources/technical-overview

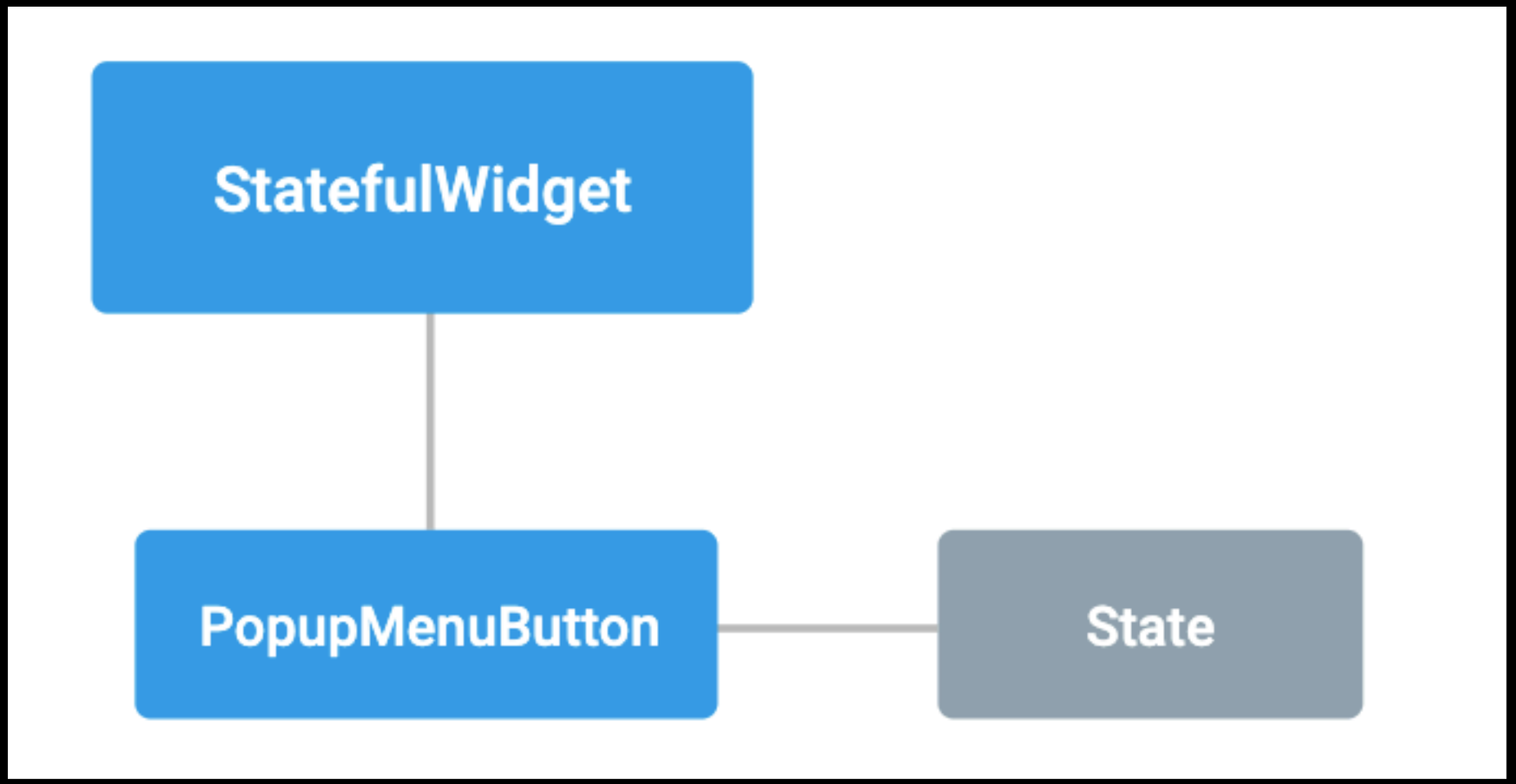
FLUTTER EVERYTHING IS A WIDGET

https://flutter.dev/docs/resources/technical-overview

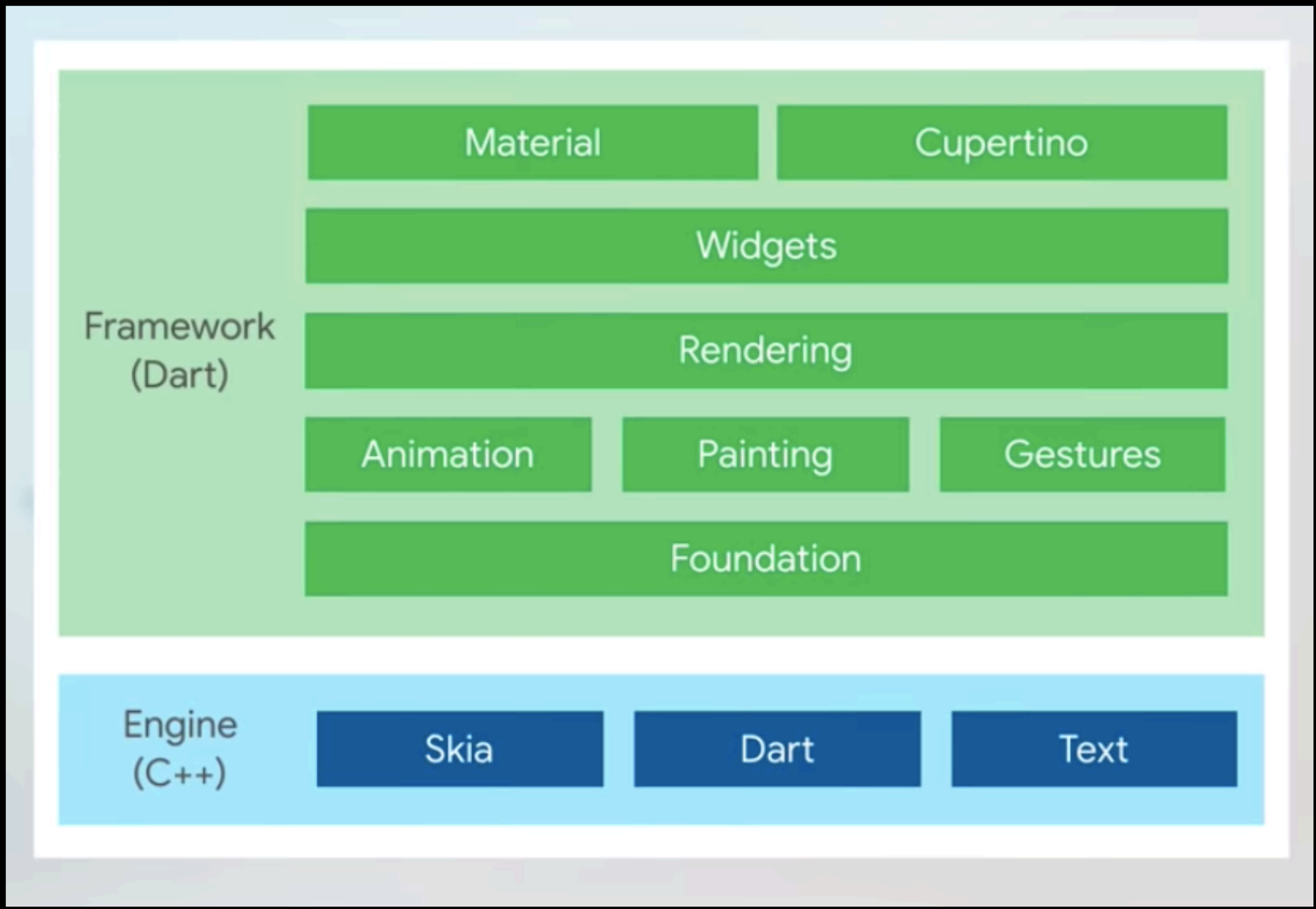


- New widgets can be built form other widgets.

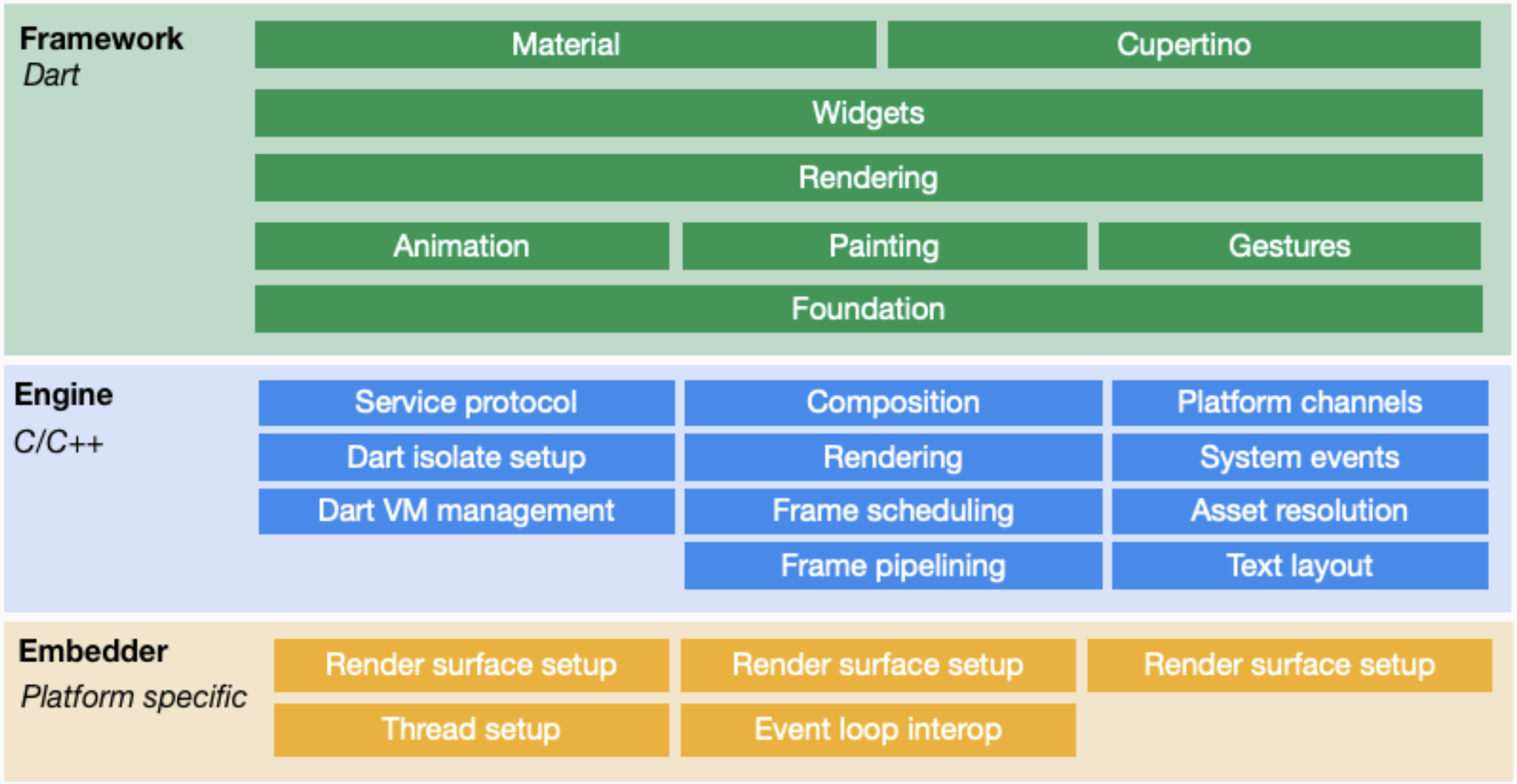
STATEFULL VS STATELESS WIDGETS



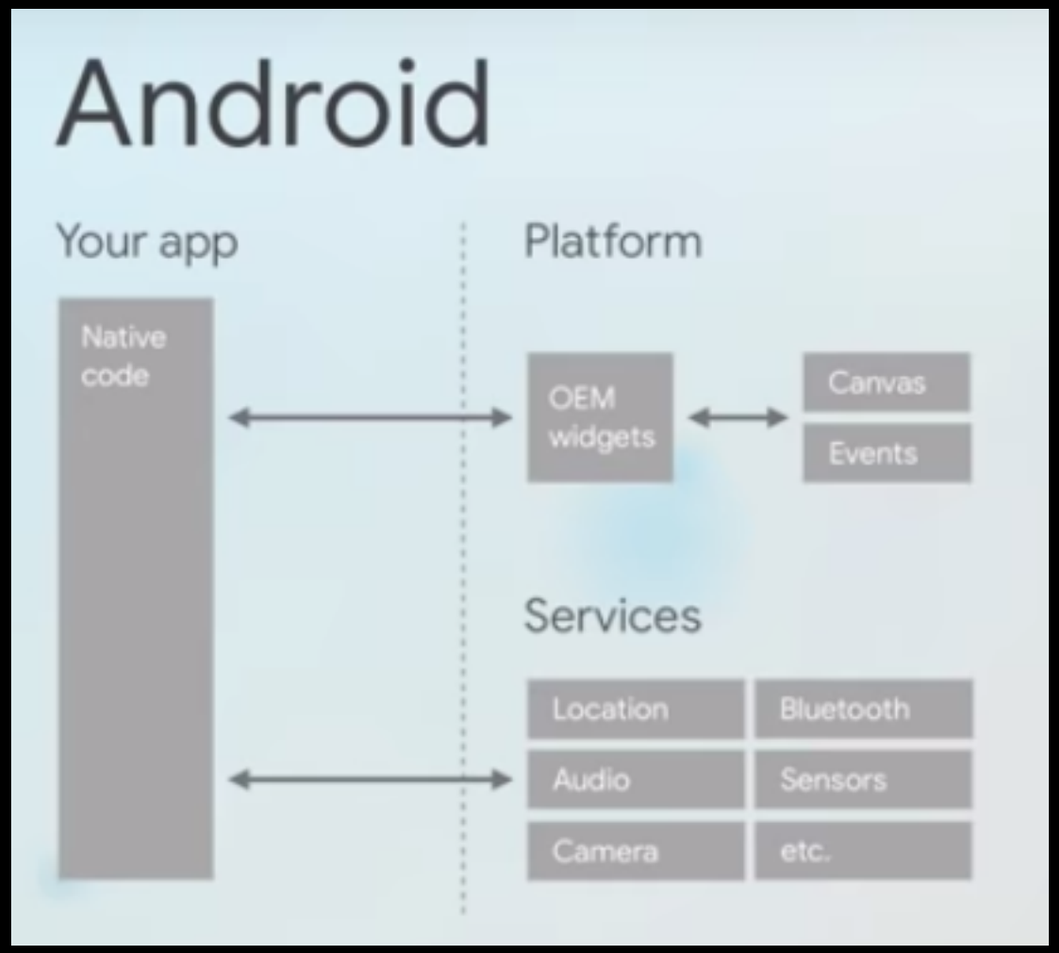
- UI = F(State)
- Change the state the UI will change to reflect the new state

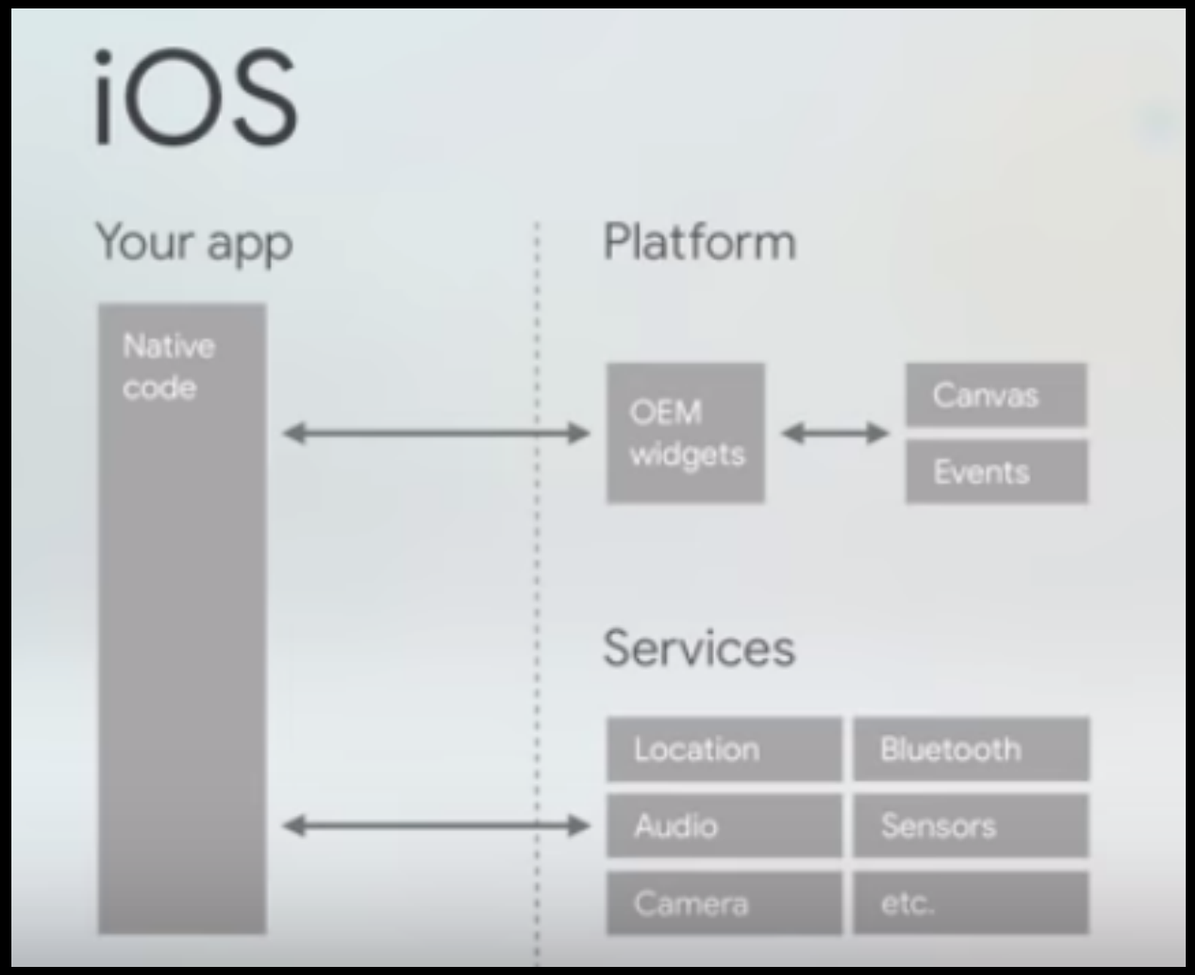

Framework
(Dart)
Material
Cupertino
Widgets
Rendering
Animation
Painting
Gestures
Foundation
Engine
(C++)
Skia
Dart
Text

Flutter system overview



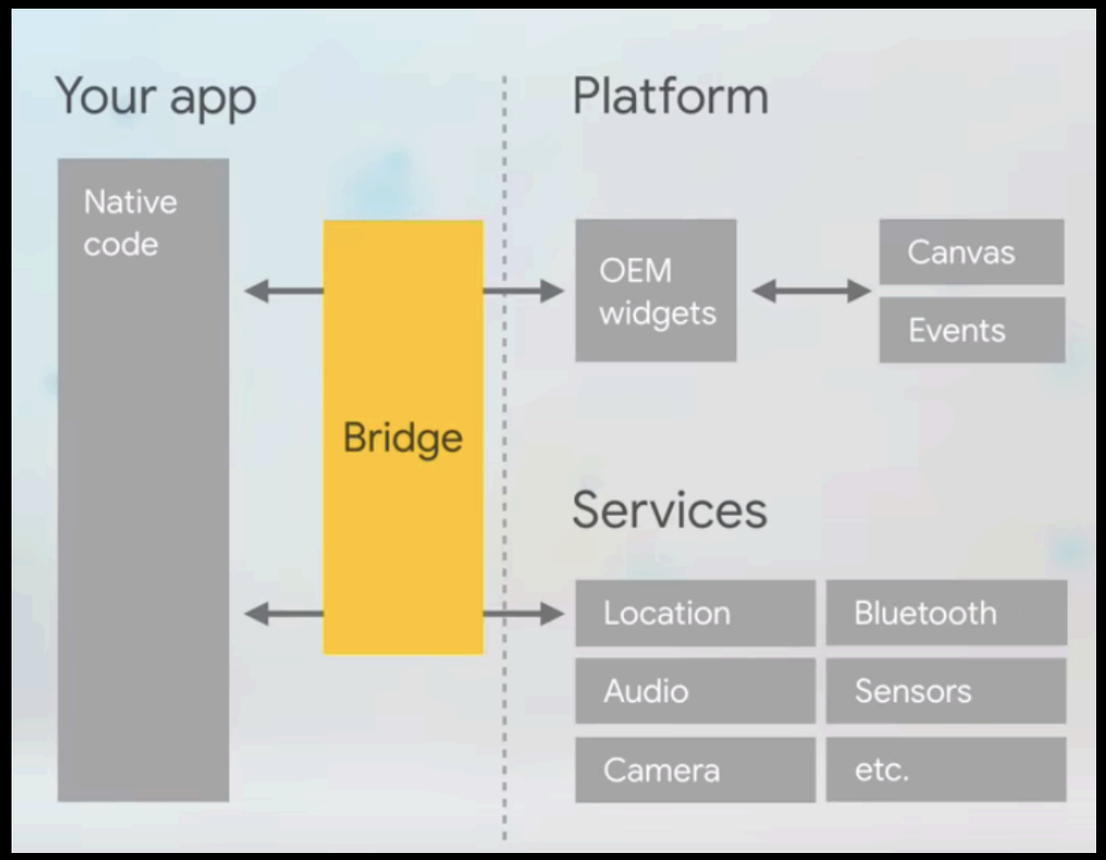
HOW DOES FLUTTER'S DIFFER FROM NATIVE AND REACT NATIVE APPROACH?

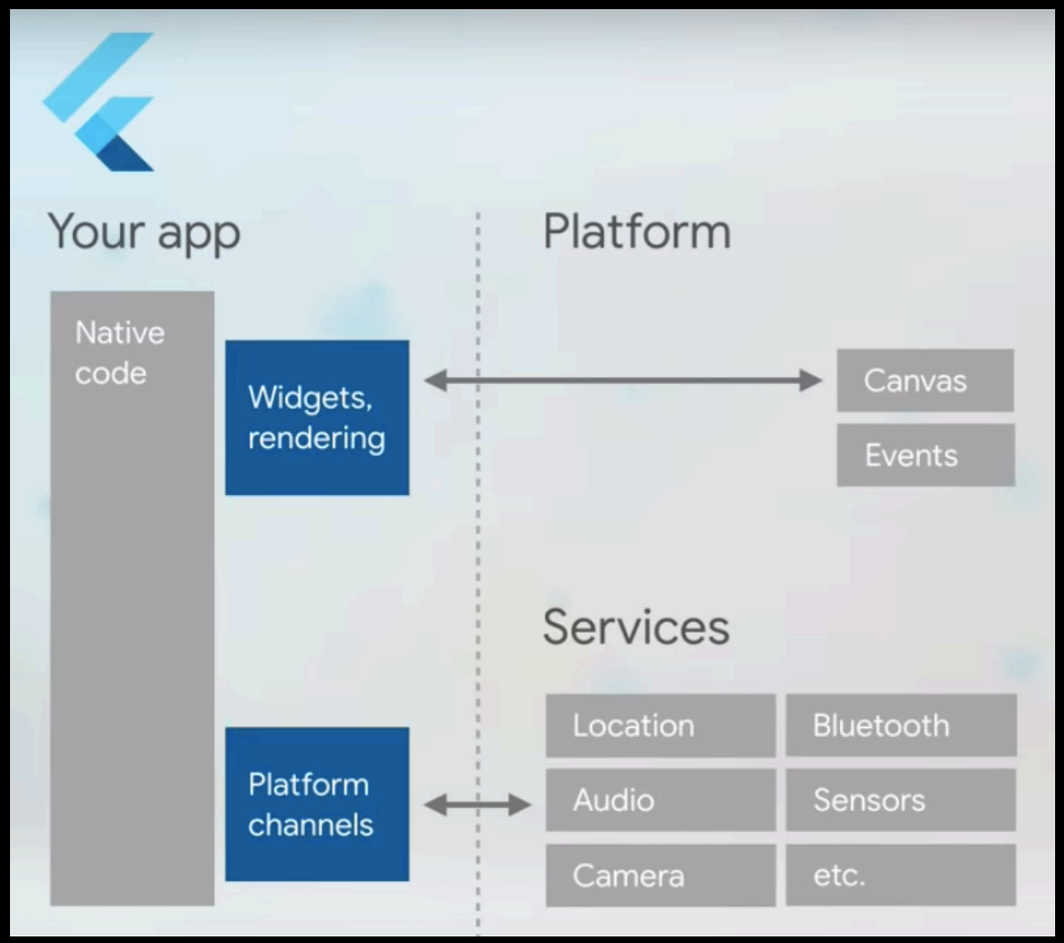




- Need to maintain two separate code bases.
- OEM Original Equipment Manufacturer Widgets

REACT NATIVE VS FLUTTER





Crossing the bridge can be slow so animations can sometimes be slow

- Full control over the rendering stack
- Reactive views with no bridge
- Great development experience with Hot Reload
- Fast, smooth, and predictable UI
- Deploy to multiple platforms from one codebase

VIDEO REFLECTLY

HTTPS://WWW.YOUTUBE.COM/WATCH?V=6ZPETBJJIPQ

FLUTTER FOR REACT NATIVE DEVELOPERS

JS/REACT

```
// JavaScript
function startHere() {
  // Can be used as entry point
}
```

No Predefined Entry Point

DART

```
// Dart
main() {
}
```

Has Entry Point
Main

FLUTTER FOR REACT NATIVE DEVELOPERS (TYPING AND DEFAULT

JS/REACT

```
console.log('Hello world!');
```

```
var name = 'JavaScript';
```

```
var name; // == undefined
```

1 & any non-null
Objects == true

DART

```
// Dart
print('Hello world!');
```

```
String name = 'dart';
// Explicitly typed as a string.

var otherName = 'Dart';
// Inferred string.

// Both are acceptable in Dart.
```

```
    var name; // == null
    int x; // == null
```

Only true == true

FUNCTIONS IN DART

REACT

```
function fn() {
    return true;
}
```

DART

```
fn() {
  return true;
}
// can also be written as
bool fn() {
  return true;
}
```

ASYNC PROGRAMMING

REACT

```
class Example {
  _getIPAddress() {
    const url = 'https://httpbin.org/ip';
    return fetch(url)
      .then(response => response.json())
      .then(responseJson => {
        const ip = responseJson.origin;
        return ip;
      });
  }
}

function main() {
  const example = new Example();
  example
    ._getIPAddress()
    .then(ip => console.log(ip))
    .catch(error => console.error(error));
}

main();
```

DART

```
import 'dart:convert';
import 'package:http/http.dart' as http;

class Example {
  Future<String> _getIPAddress() {
    final url = 'https://httpbin.org/ip';
    return http.get(url).then((response) {
      String ip
       = jsonDecode(response.body)['origin'];
      return ip;
    });
  }
}

main() {
  final example = new Example();
  example
      ._getIPAddress()
      .then((ip) => print(ip))
      .catchError((error) => print(error));
}
```

ASYNC PROGRAMMING

REACT

```
class Example {
    async function _getIPAddress() {
      const url = 'https://httpbin.org/ip';
      const response = await fetch(url);
      const json = await response.json();
      const data = await json.origin;
      return data;
    }
  }

  async function main() {
    const example = new Example();
    try {
      const ip = await example._getIPAddress();
      console.log(ip);
    } catch (error) {
      console.error(error);
    }
  }

  main();
```

DART

```
class Example {
      Future<String> _getIPAddress() async {
        final url = 'https://httpbin.org/ip';
        final response = await http.get(url);
        String ip =
         jsonDecode(response.body)['origin'];
        return ip;
      }
    }

    main() async {
      final example = new Example();
      try {
        final ip =
              await example._getIPAddress();
        print(ip);
      } catch (error) {
        print(error);
      }
    }
```

HELLO WORLD

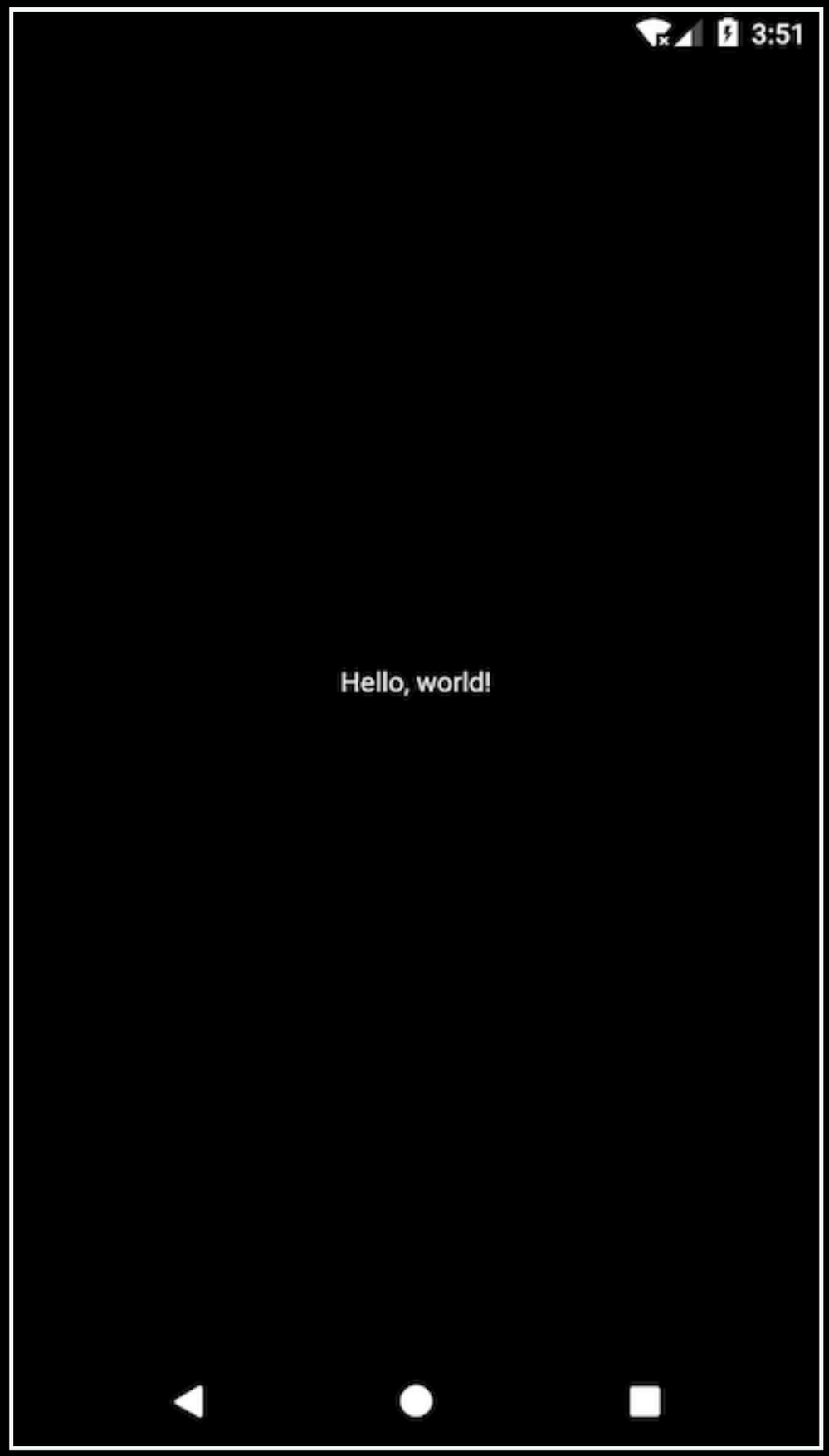
REACT

```
import React from 'react';
import { StyleSheet, Text, View }
from 'react-native';

export default class App
 extends React.Component {
  render() {
    return (
      <View style={styles.container}>
        <Text>Hello world!</Text>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center'
  }
});
```

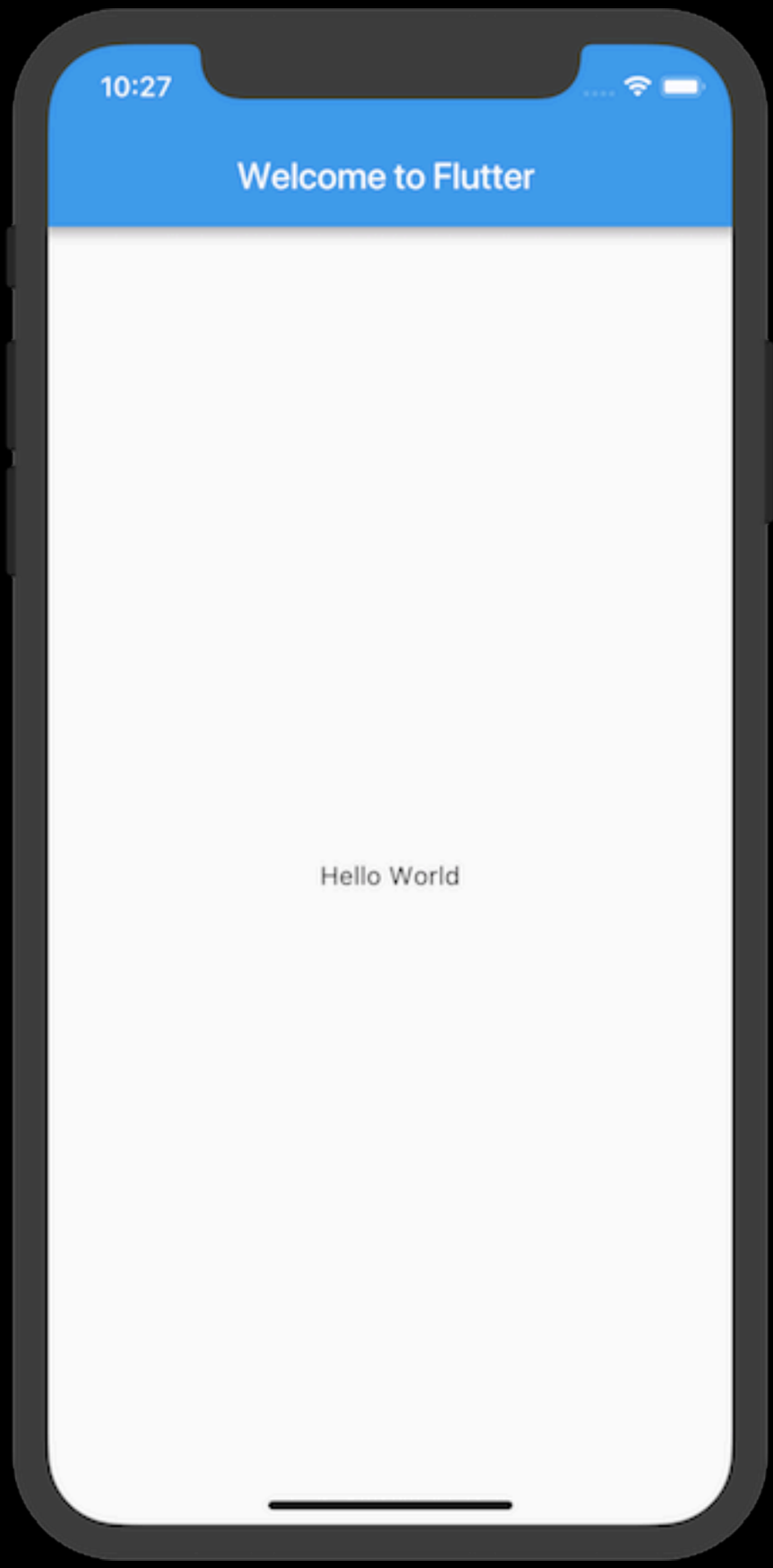
DART

```
import 'package:flutter/material.dart';

    void main() {
      runApp(
        Center(
          child: Text(
            'Hello, world!',
            textDirection:
                TextDirection.ltr,
          ),
        ),
      );
    }
```

```dart
import 'package:flutter/material.dart';

    void main() {
      runApp(
        Center(
          child: Text(
            'Hello, world!',
            textDirection:
                TextDirection.ltr,
          ),
        ),
      );
    }
```



```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Welcome to Flutter'),
        ),
        body: Center(
          child: Text('Hello world'),
        ),
      ),
    );
  }
}
```



CREATE REUSABLE COMPONENTS

REACT NATIVE

```
class CustomCard extends
 React.Component {
    render() {
      return (
        <View>
          <Text>
              Card {this.props.index}
          </Text>
          <Button
            title="Press"
            onPress={() =>

this.props.onPress(this.props.index)}
          />
        </View>
      );
    }
  }

  // Usage
  <CustomCard onPress={this.onPress}
               index={item.key} />
```

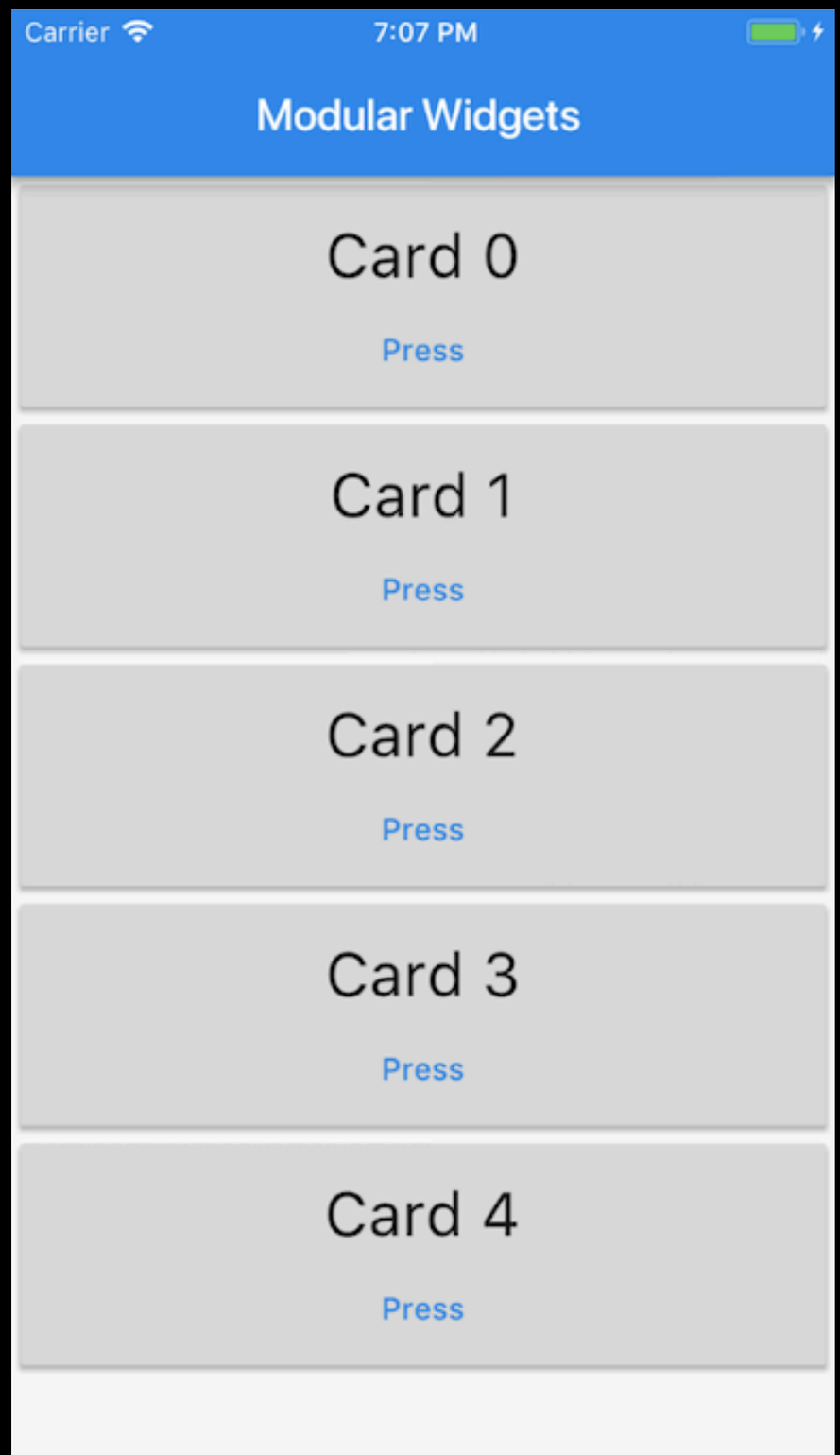
FLUTTER

```
class CustomCard extends StatelessWidget {
        CustomCard(@required this.index,
@required this.onPress);

        final index;
        final Function onPress;

        @override
        Widget build(BuildContext context) {
          return Card(
            child: Column(
              children: <Widget>[
                Text('Card $index'),
                FlatButton(
                  child: const Text('Press'),
                  onPressed: this.onPress,
                ),
              ],
            )
          );
        }
      }

      ...
        // Usage
        CustomCard(
          index: index,
          onPress: () {
            print('Card $index');
          },
        )
```

REACT NATIVE S

```
class CustomCard extends StatelessWidget {
        CustomCard(@required this.index,
@required this.onPress);

        final index;
        final Function onPress;

        @override
        Widget build(BuildContext context) {
          return Card(
            child: Column(
              children: <Widget>[
                Text('Card $index'),
                FlatButton(
                  child: const Text('Press'),
                  onPressed: this.onPress,
                ),
              ],
            )
          );
        }
      }
```

```
    ...
      // Usage
      CustomCard(
        index: index,
        onPress: () {
          print('Card $index');
        },
      )
```



REACT

```
    <View
  style={{
    flex: 1,
    flexDirection: 'column',
    justifyContent: 'space-between',
    alignItems: 'center'
  }} >
```

FLUTTER

```
Center(
  child: Column(
    children: <Widget>[
      Container(
        color: Colors.red,
        width: 100.0,
        height: 100.0,
      ),
      Container(
        color: Colors.blue,
        width: 100.0,
        height: 100.0,
      ),
      Container(
        color: Colors.green,
        width: 100.0,
        height: 100.0,
      ),
    ],
  ),
)
```

```
┬
└ projectname
  ┬
  ├ android      - Contains Android-specific files.
  ├ build        - Stores iOS and Android build files.
  ├ ios          - Contains iOS-specific files.
  ├ lib          - Contains externally accessible Dart source files.
    ┬
    └ src        - Contains additional source files.
    └ main.dart  - The Flutter entry point and the start of a new app.
                   This is generated automatically when you create a Flutter
                    project.
                   It's where you start writing your Dart code.
  ├ test         - Contains automated test files.
  └ pubspec.yaml - Contains the metadata for the Flutter app.
                   This is equivalent to the package.json file in React Native.
```

REACT NATIVE

```
<Image source={require('./my-icon.png')} />
```

FLUTTER

```
flutter:
  assets:
    - assets/my_icon.png
    - assets/background.png
```

```
image: AssetImage('assets/background.png'),
body: Image.network(
          'https://flutter.io/images/owl.jpg',
```

. Check your spaces when working in the YAML file because **white space matters**!

```
┬
└ projectname
  ┬
  ├ android      - Contains Android-specific files.
  ├ build        - Stores iOS and Android build files.
  ├ ios          - Contains iOS-specific files.
  ├ lib          - Contains externally accessible Dart source files.
    ┬
    └ src        - Contains additional source files.
    └ main.dart  - The Flutter entry point and the start of a new app.
                   This is generated automatically when you create a Flutter
                    project.
                   It's where you start writing your Dart code.
  ├ test         - Contains automated test files.
  └ pubspec.yaml - Contains the metadata for the Flutter app.
                   This is equivalent to the package.json file in React Native.
```

```yaml
flutter:
  assets:
    - assets/my_icon.png
    - assets/background.png
```

```yaml
dependencies:
  flutter:
    sdk: flutter
  google_sign_in: ^3.0.3
```