Introduction to Flutter

What is Flutter?

Flutter is Google's mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time.

flutter.io

What is Flutter?

Flutter is Google's mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time.

flutter.io

What is Flutter?

Flutter is Google's mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time.

flutter.io

What is Flutter?

Flutter is Google's mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time.

flutter.io

What is Flutter?

Flutter is Google's mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time.

flutter.io

Yet another one of these cross-platform frameworks🙄

Why is Flutter?

- Dart support JIT and AOP compilation
- No OEM widgets

  no fragmentation or compatibility issue.
- High-performance

  no bridge needed.

Why is Flutter?

Worst Programming Languages to Learn in 2018 Rankings

Ranked from Worst to Best Languages to Learn

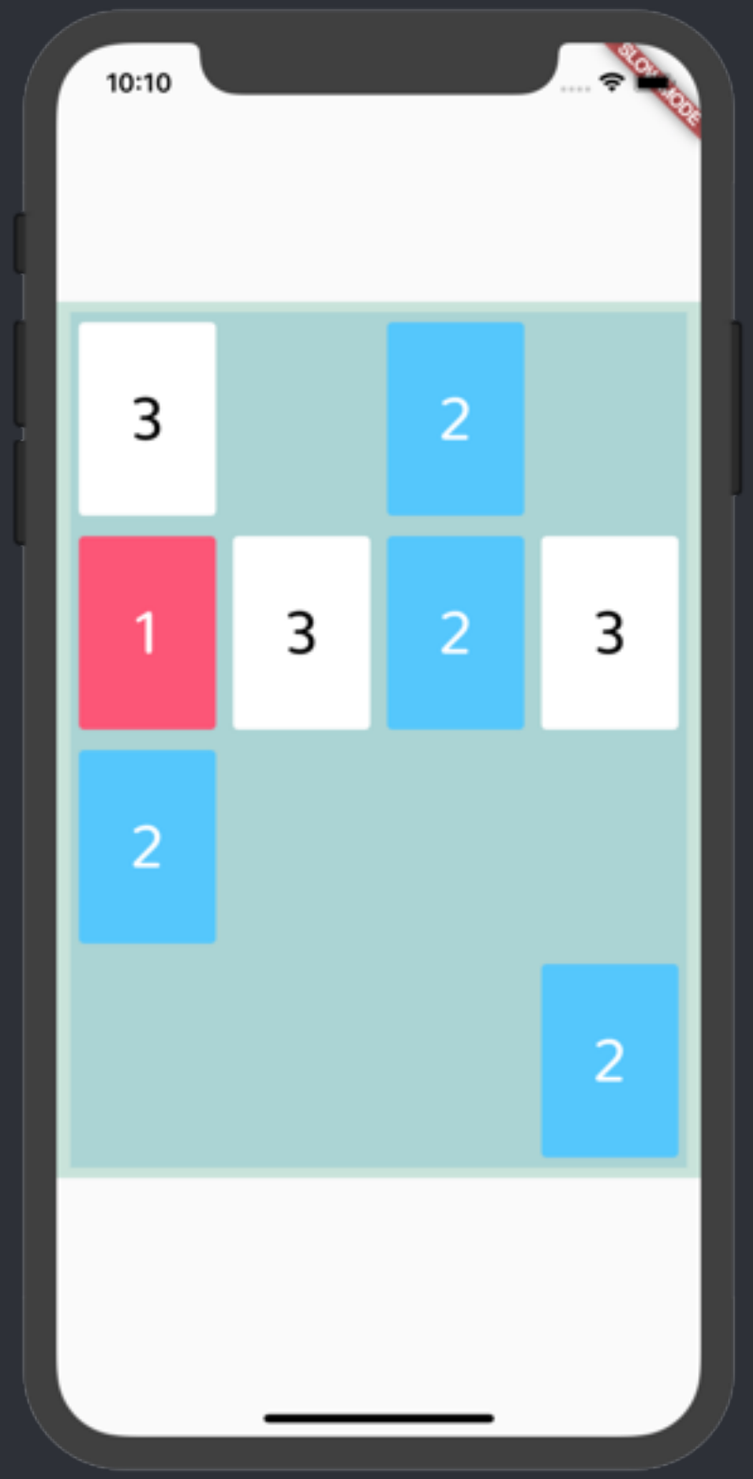
	Overall Rankings	Community Engagement	Job Market	Growth and Trends
1	Dart	Dart	Dart	Objective-C
2	Objective-C	CoffeeScript	Rust	CoffeeScript
3	CoffeeScript	Objective-C	Elm	Dart
4	Erlang	Lua	Lua	Perl
5	Lua	Elm	Erlang	Erlang
6	Clojure	Clojure	Clojure	Clojure
7	Perl	Elixir	Kotlin	Ruby
8	Elm	Erlang	Elixir	C#
9	Elixir	Kotlin	R	Lua
10	Haskell	Perl	Perl	C
11	Rust	Scala	Haskell	Haskell
12	Scala	TypeScript	CoffeeScript	Rust

Everything is Widget

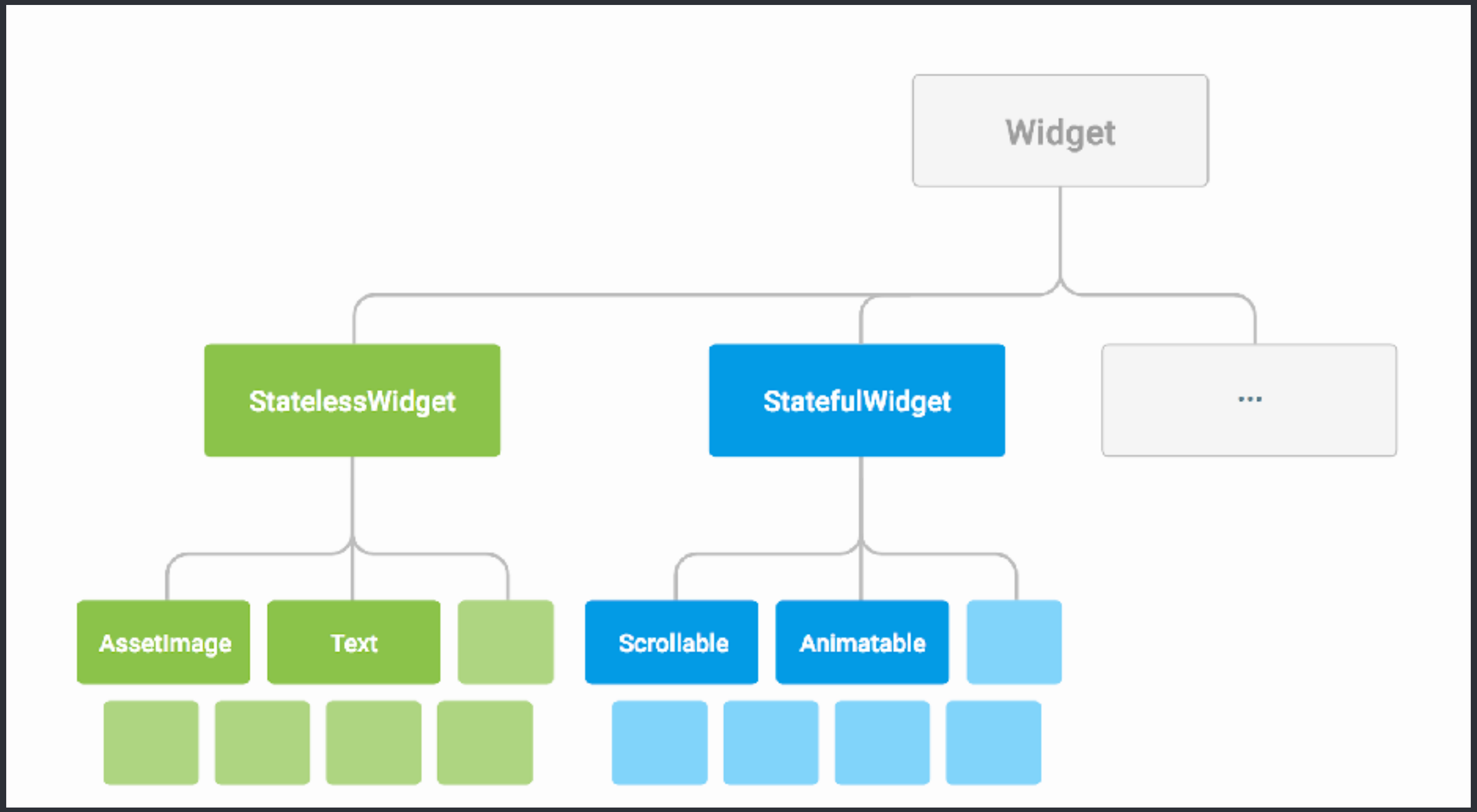
- a structural **element**
  (e.g. button, image)
- a stylistic **element**
  (e.g. styles, fonts)
- an aspect of layout
  (e.g. padding, center)
- even Navigation, Gesture, Animation

Everything is Widget

```
void main() => runApp(ThreesApp());

class ThreesApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Threes powered by Flutter',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        body: Center(
          child: Container(
            padding: EdgeInsets.all(8.0),
            color: Theme.Colors.lightGreen,
            child: GamePanel(),
          ),
        ),
      ),
    );
  }
}
```



Stateless & Stateful



Stateless & Stateful

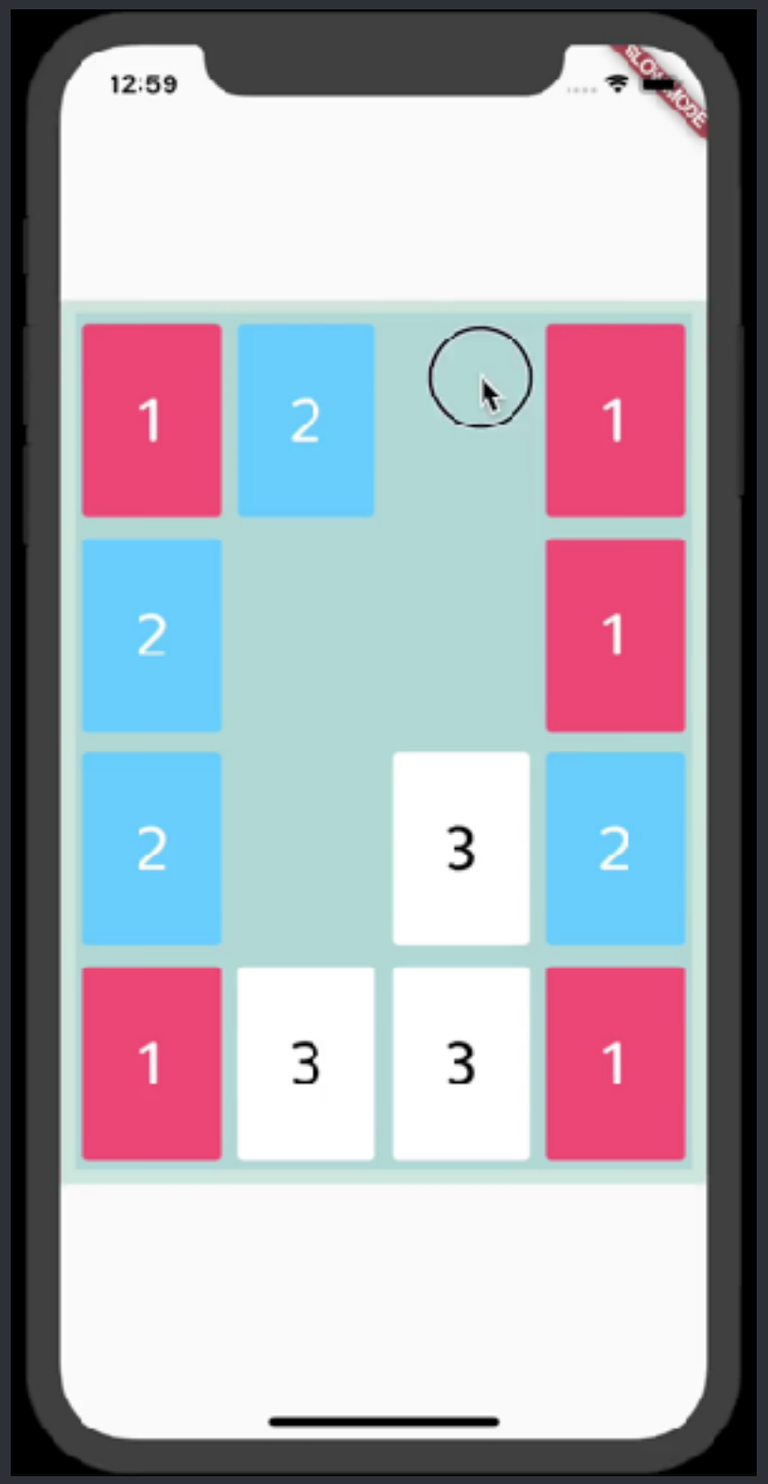
```
class HelloWidget extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return Text("Hello world!");
  }
}
```

Stateless & Stateful

```
class HelloWidget extends StatefulWidget {

  @override
  _HelloState createState() => _HelloState();
}

class _HelloState extends State< HelloWidget > {

  String _title = "Hello world!";

  @override
  Widget build(BuildContext context) {
    return Text(_title);
  }

  changeState() {
    setState(() {
      _title = "Hello everyone!";
    });
  }
}
```

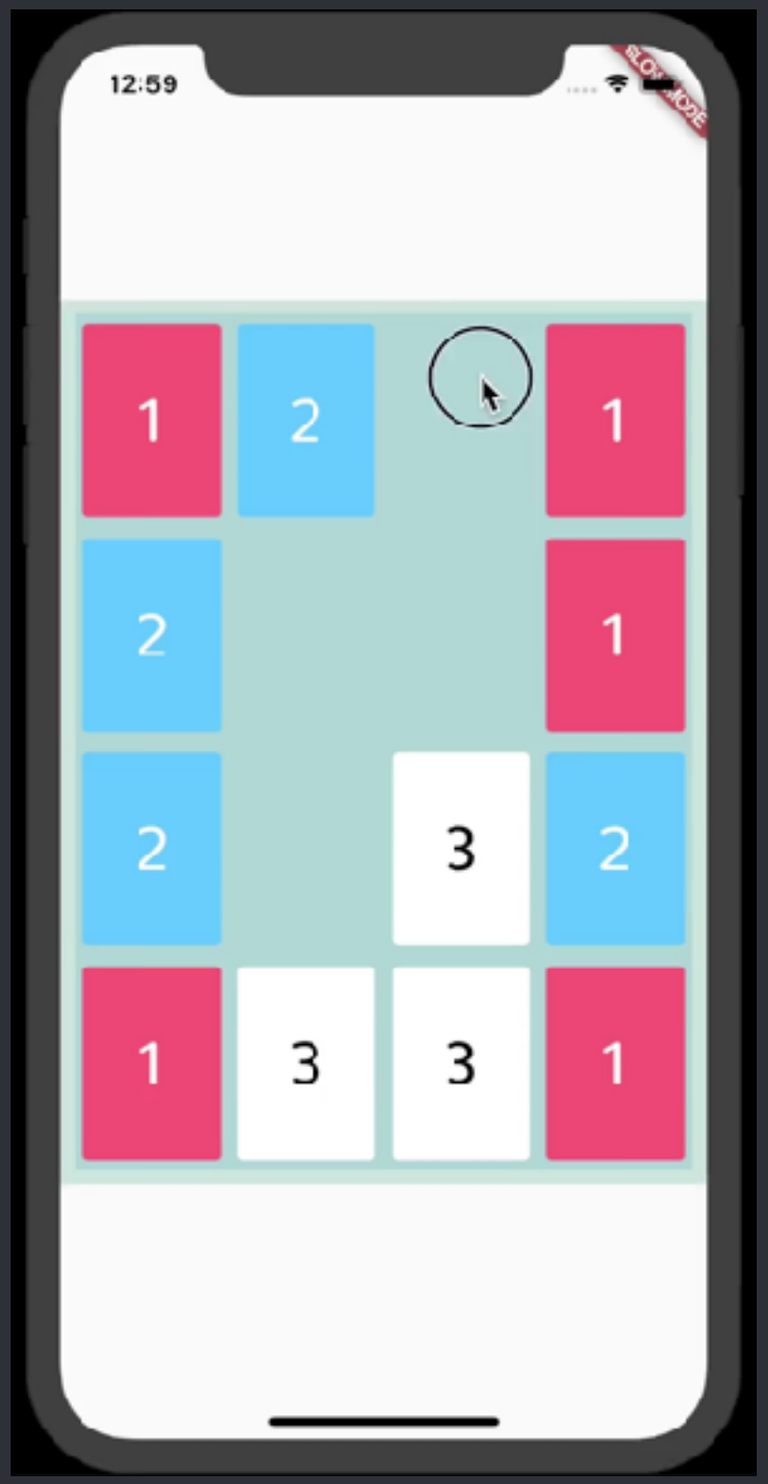
Everything is Widget

```
GestureDetector(
  behavior: HitTestBehavior.opaque,
  onHorizontalDragEnd: (DragEndDetails d) {
    if (d.primaryVelocity > 0) {
      dispatch(Direction.right);
    } else {
      dispatch(Direction.left);
    }
  },
  onVerticalDragEnd: (DragEndDetails d) {
    if (d.primaryVelocity > 0) {
      dispatch(Direction.down);
    } else {
      dispatch(Direction.up);
    }
  },
  child: ...,
)
```



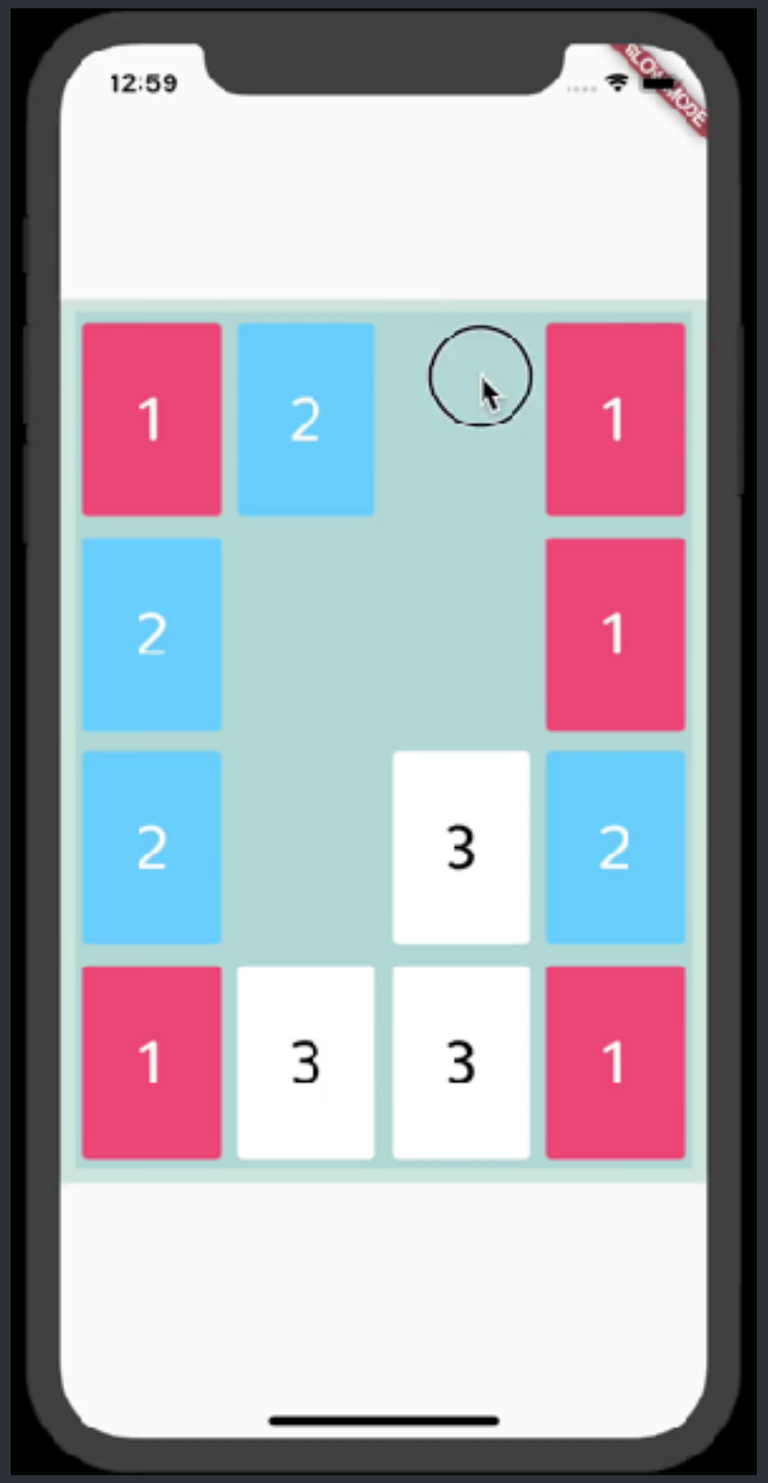
Everything is Widget

```
GestureDetector(
  behavior: HitTestBehavior.opaque,
  onHorizontalDragEnd: (DragEndDetails d) {
    if (d.primaryVelocity > 0) {
      dispatch(Direction.right);
    } else {
      dispatch(Direction.left);
    }
  },
  onVerticalDragEnd: (DragEndDetails d) {
    if (d.primaryVelocity > 0) {
      dispatch(Direction.down);
    } else {
      dispatch(Direction.up);
    }
  },
  child: ...,
)

class GestureDetector extends StatelessWidget {
  ...
}
```



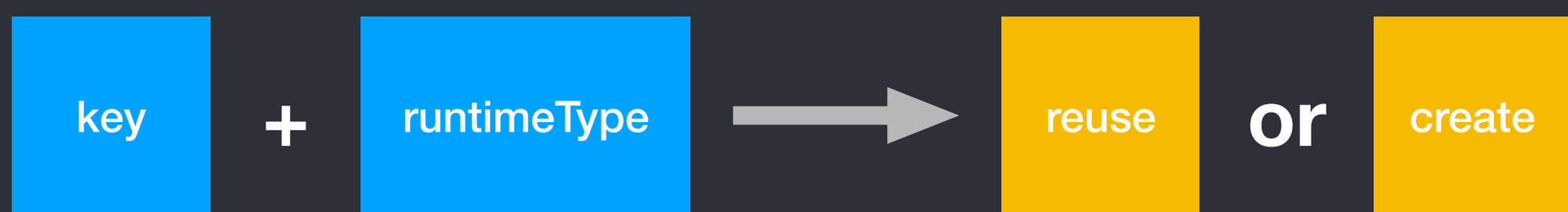
Everything is Widget

```
class _GamePanelState extends State<GamePanel>
    with TickerProviderStateMixin

AnimationController _controller = AnimationController(
  duration: Duration(milliseconds: 150),
  vsync: this,
);

Animation<Alignment> alignment = AlignmentTween(
  begin: Alignment(fromJPosition, fromIPosition),
  end: Alignment(destJPosition, destIPosition),
)
    .animate(CurvedAnimation(
  curve: Curves.easeOut,
  parent: _controller,
));

AlignTransition(
  alignment: alignment,
  child: ...,
);

_controller.forward();
```

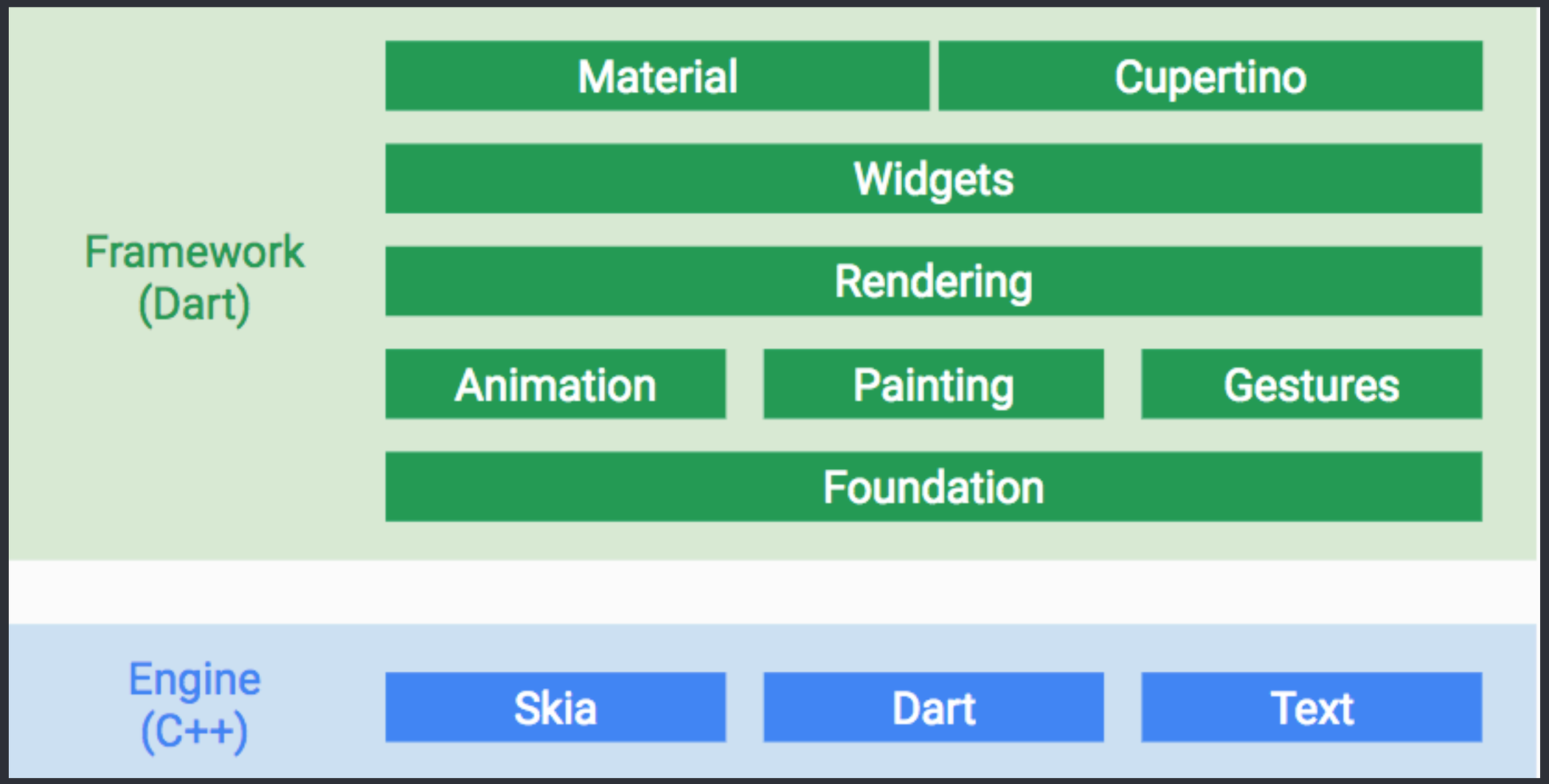


The magic of Key

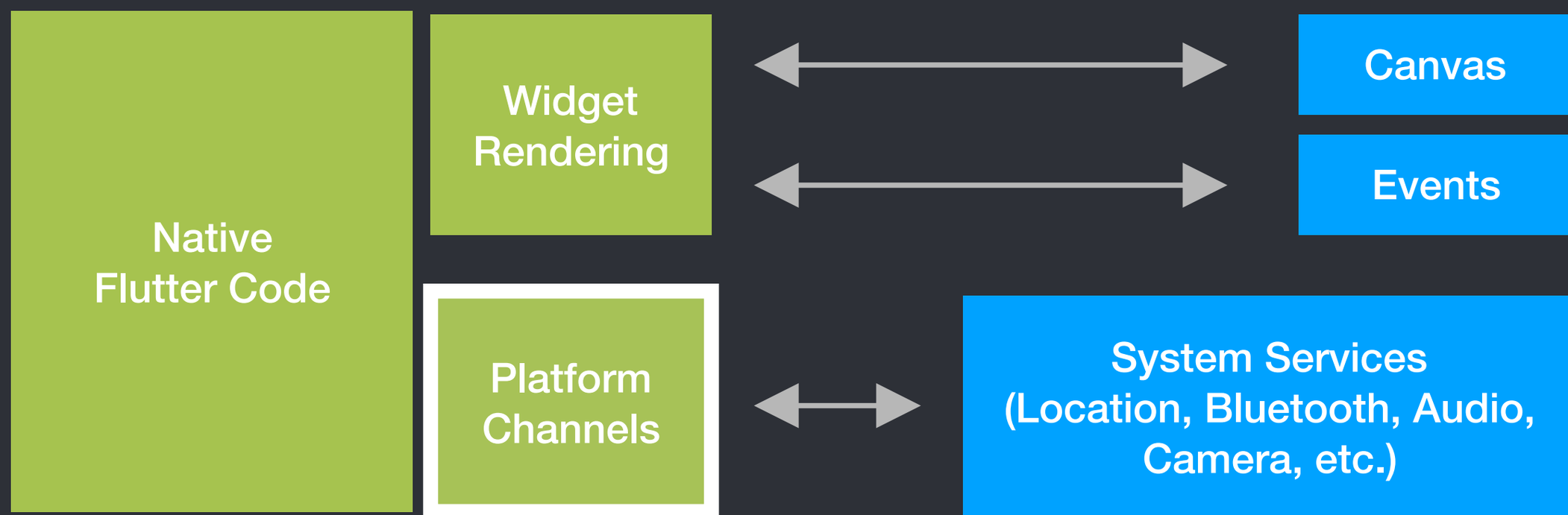
```
@immutable
abstract class Widget extends DiagnosticableTree {
  /// Initializes [key] for subclasses.
  const Widget({ this.key });

  final Key key;

  ...
}
```



Everything is Widget

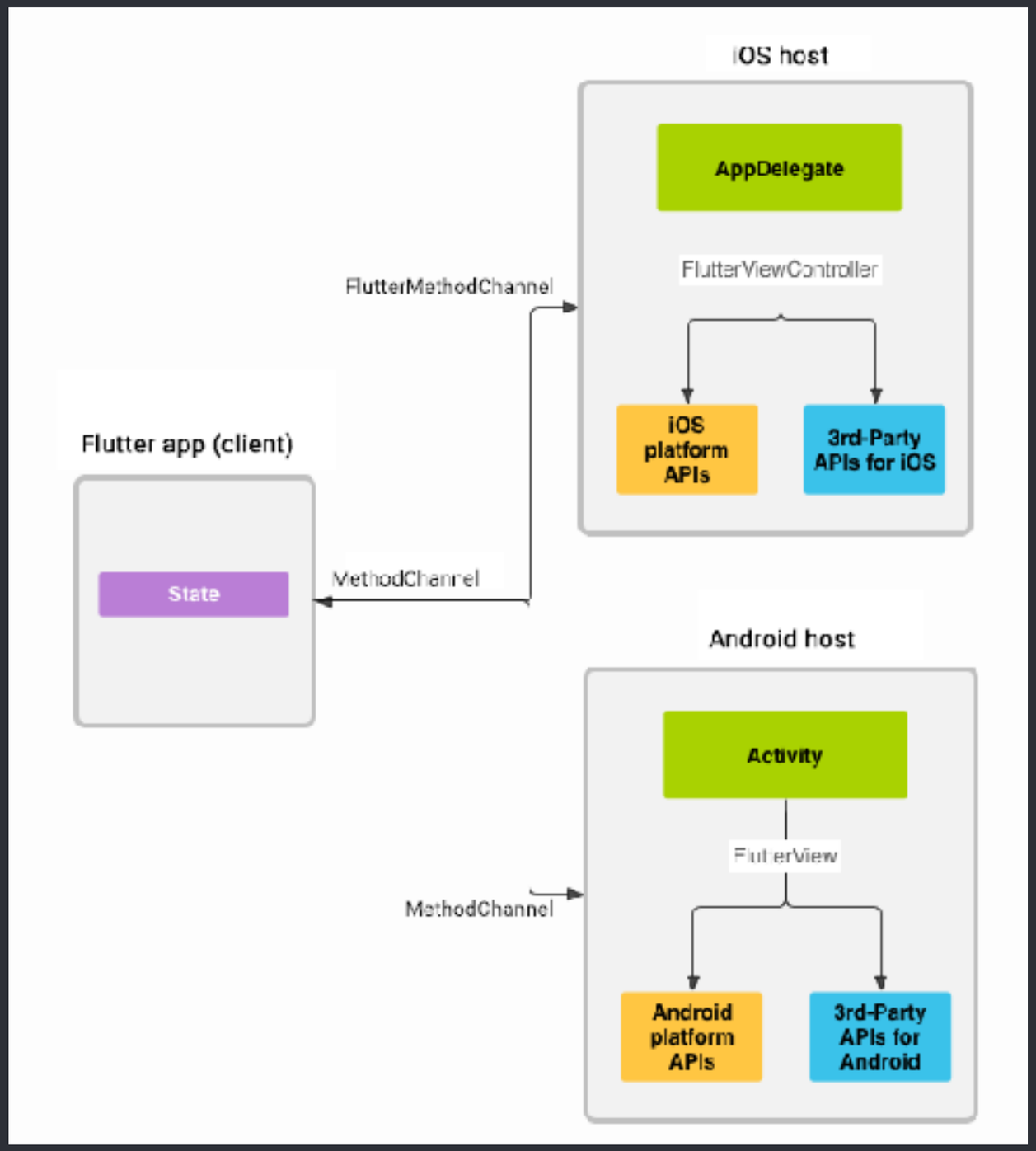


OS Integration



Base image from: https://speakerdeck.com/passsy/flutter-is-now-beta-an-introduction-mobile-dot-cologne-18

OS Integration



FlutterView
SurfaceView (Android)
UIView (iOS)

- **RPC-like**

  FlutterMethodChannel
- **Full Duplex**

  FlutterBasicMessageChannel

OS Integration

- **Dart**

```
static const methodChannel = const MethodChannel(_channelName);

static Future<Null> displayGameScore(int score) async {
  try {
    final Object result =
        await methodChannel.invokeMethod("displayGameScore", score);
  } on PlatformException catch (e) {
    print(e.toString());
  }
}
```

OS Integration

- **Swift**

```
let controller: FlutterViewController = ...

let threesChannel = FlutterMethodChannel(name: channelName,
                                         binaryMessenger: controller)

threesChannel.setMethodCallHandler({
    (call: FlutterMethodCall, result: FlutterResult) -> Void in
    ...
    result(...)
})
```

OS Integration

- **Dart**

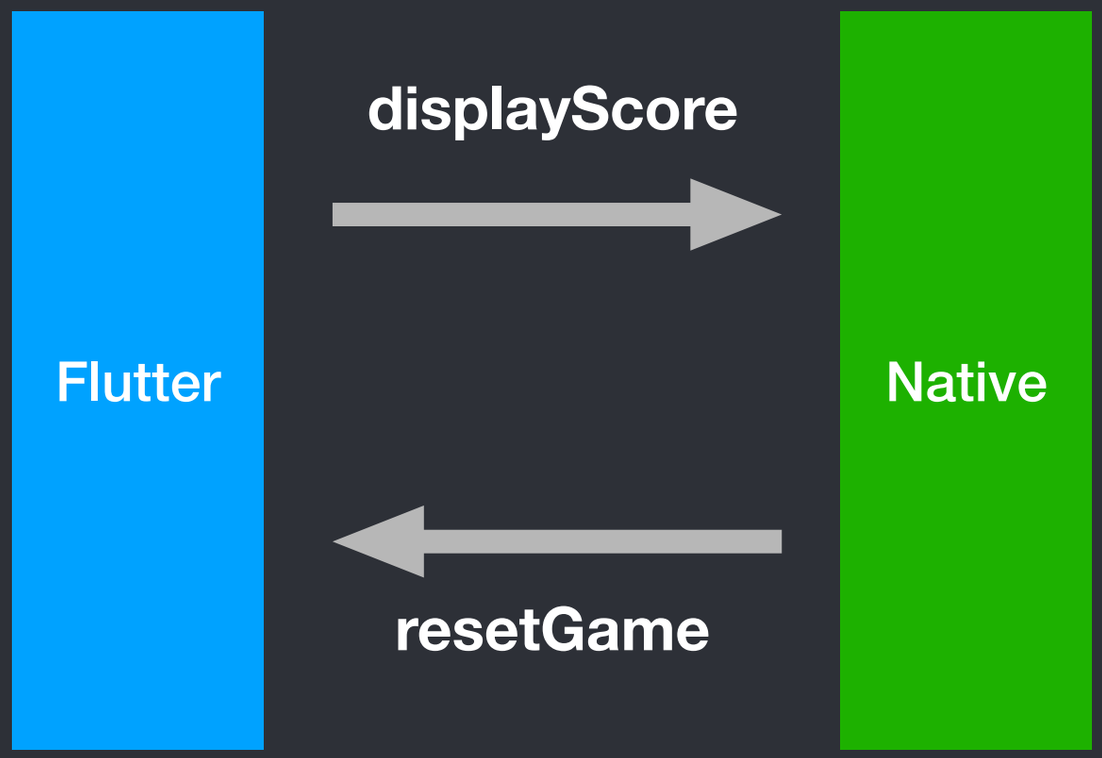
```
final BasicMessageChannel<String> messageChannel =
  const BasicMessageChannel<String>(_channelName, const StringCodec());

Future<String> _handlePlatformMessage(String message) async {
  ...
  return "";
}
```

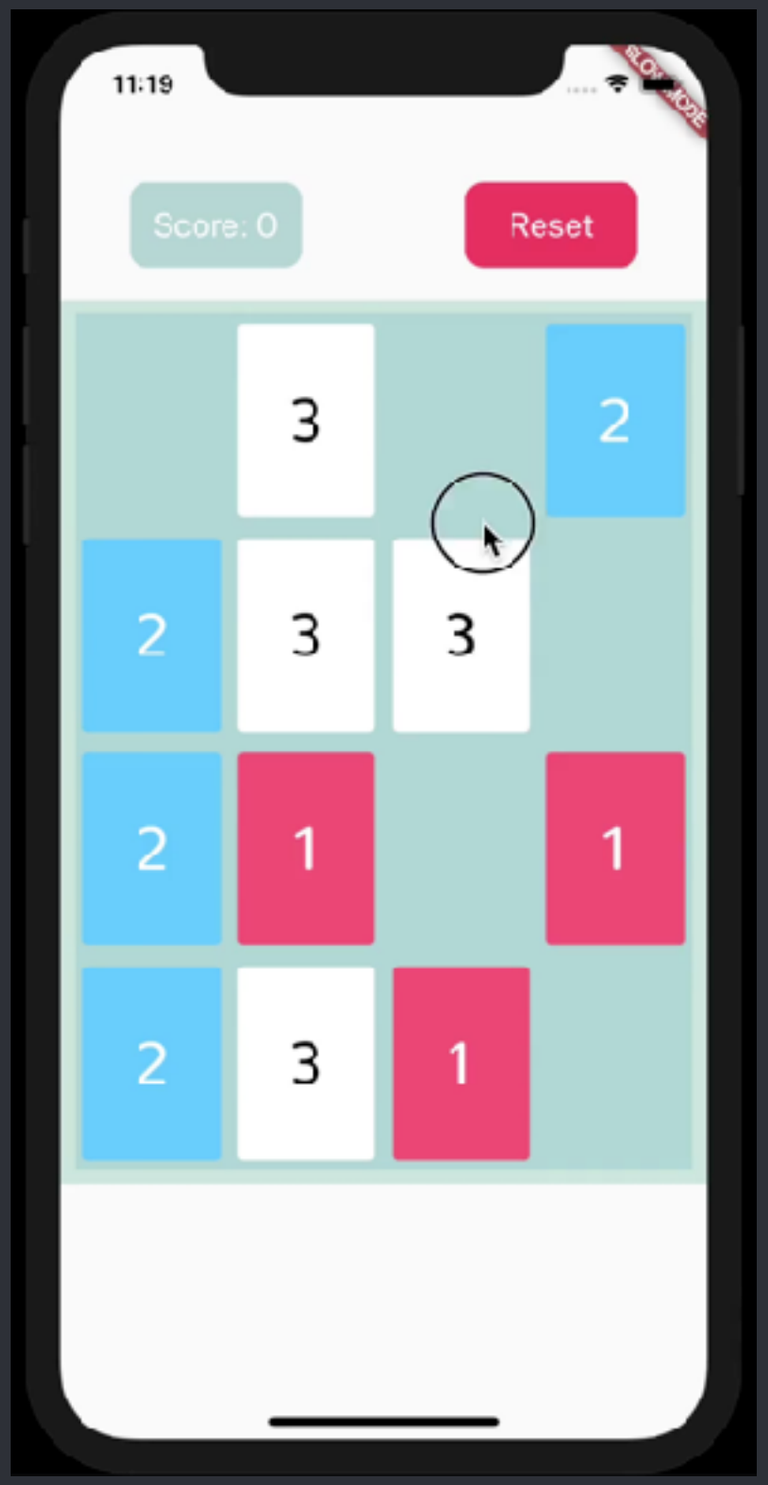
OS Integration

- **Swift**

```
let basicSenderChannel = FlutterBasicMessageChannel(name: channelName,
                                                    binaryMessenger: controller)

basicSenderChannel.sendMessage("resetThreesGame")
```

OS Integration





Demo & source
github.com/diov/threes-flutter

Drawbacks

- UI markup & layout system learning curve
- UI code can look very ugly with infinite nesting
- Lack of document
- Not a lot of “Best Practice” available
- Shortage of library
- Google’s product loyalty (2700+ issue 🤣)

Cross-platform Solutions





Cross-platform Solutions

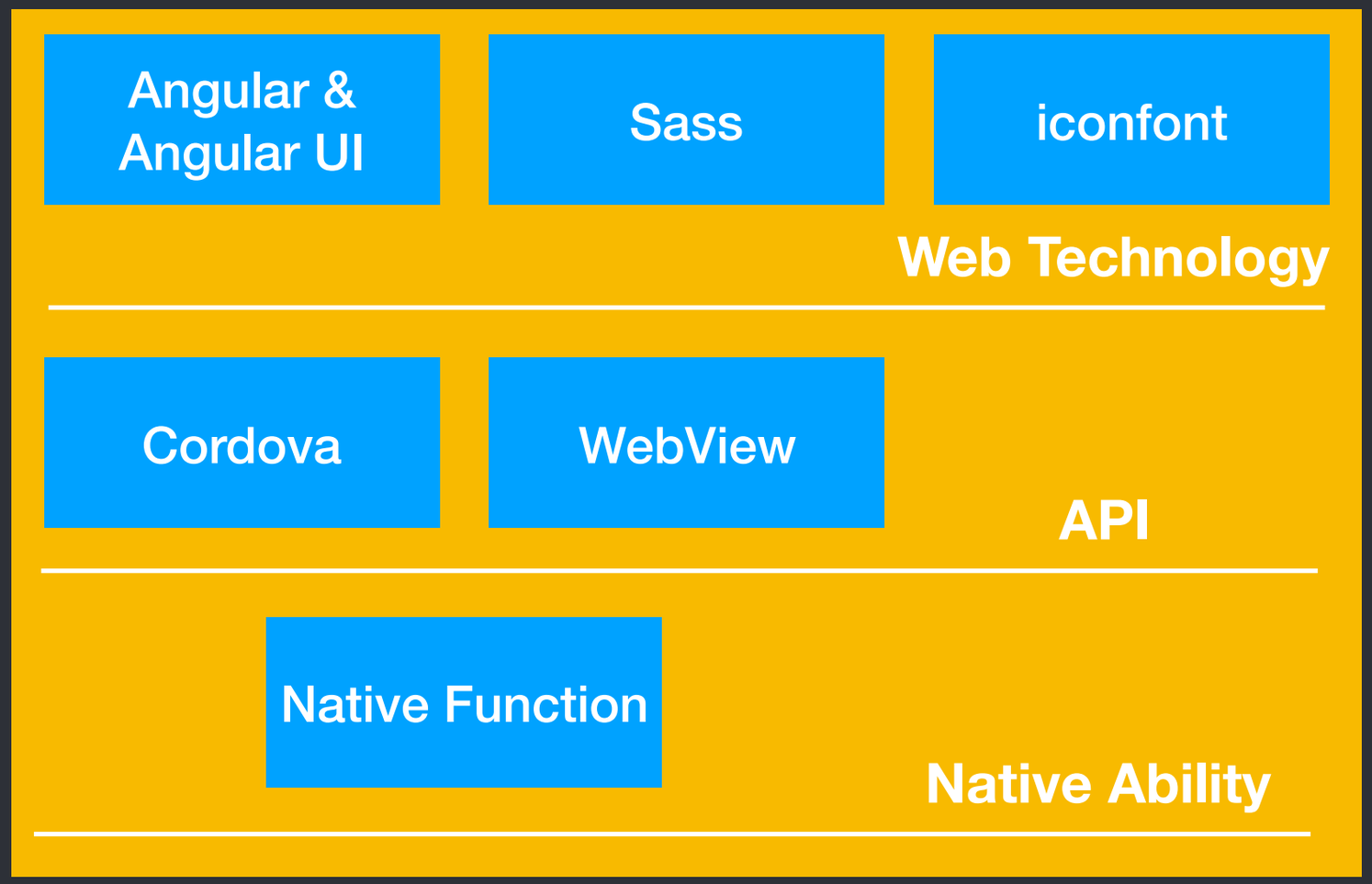
- Ionic2
- Cordova/PhoneGap/Titanium
- React Native
- Flutter
- Progressive Web Apps (PWA)
- Xamarin
- Kotlin Native
- J2ObjC/Doppl

Cross-platform Solutions

- Hybrid App (Ionic、Cordova)
- React Native
- Flutter

Hybrid App

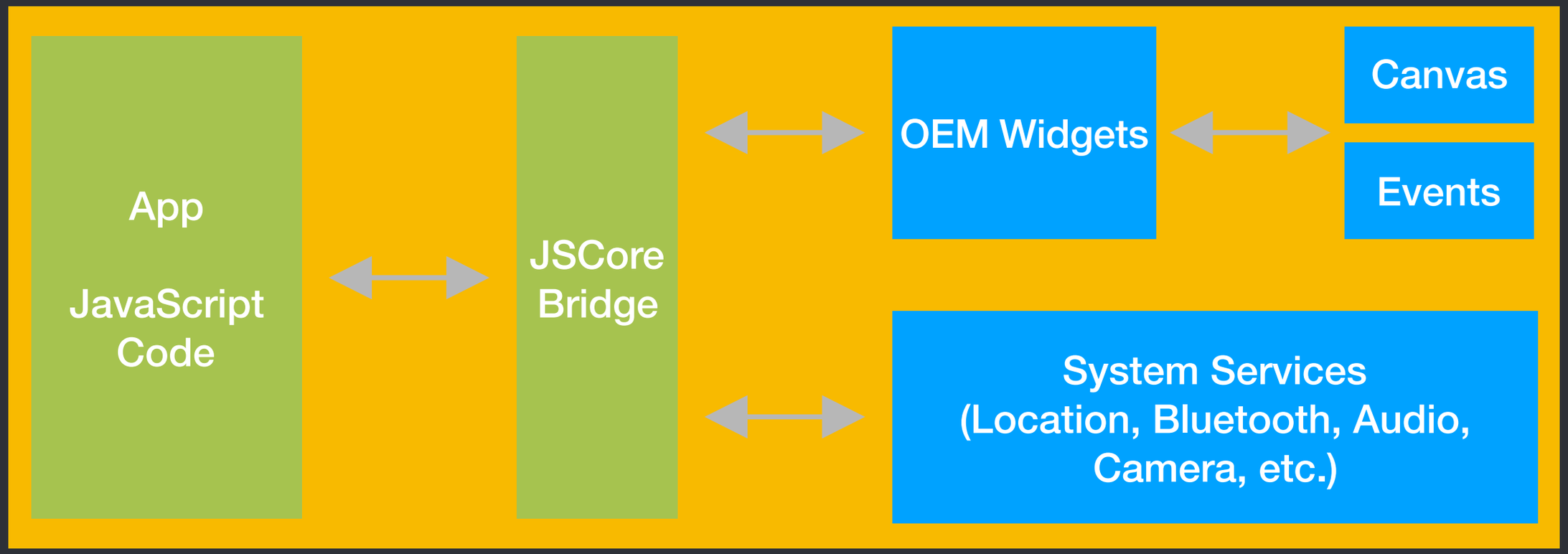
- ionic



Hybrid App

- Depend on WebView
- Write Once, Run Anywhere

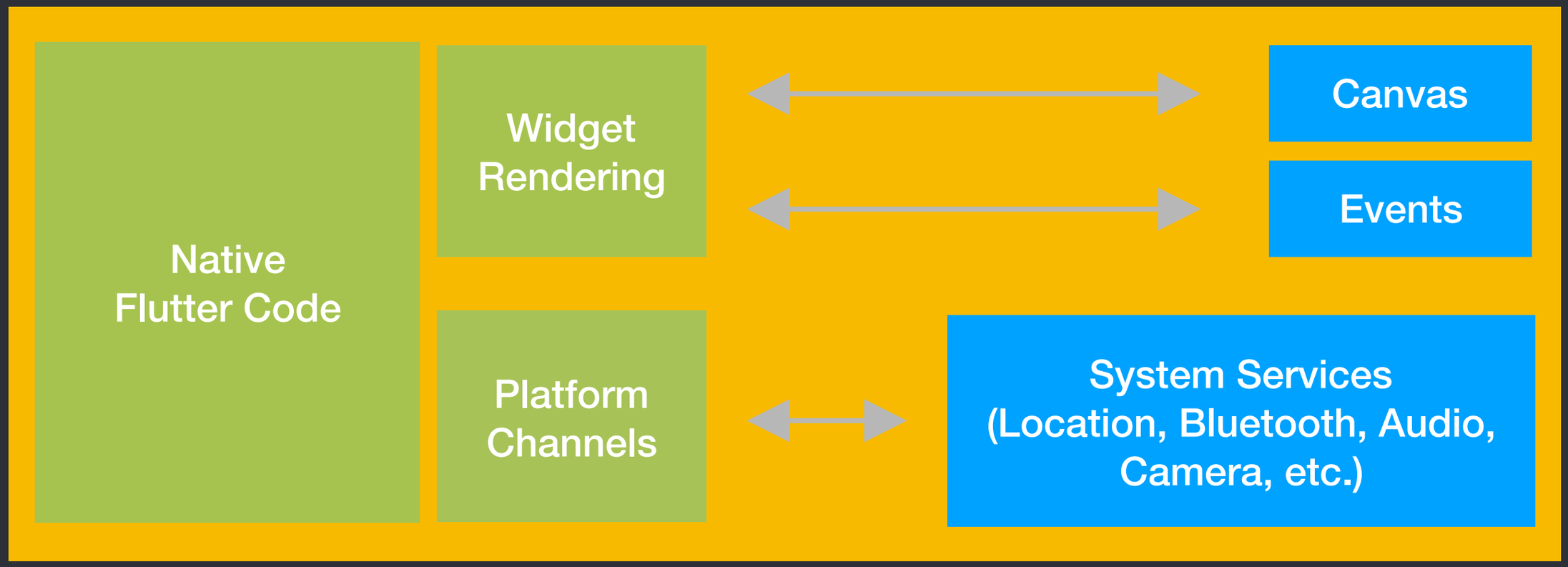
React Native



React Native

- Render with platform view, communicate with JavaScriptCore
- Learn Once, Write Anywhere

Flutter

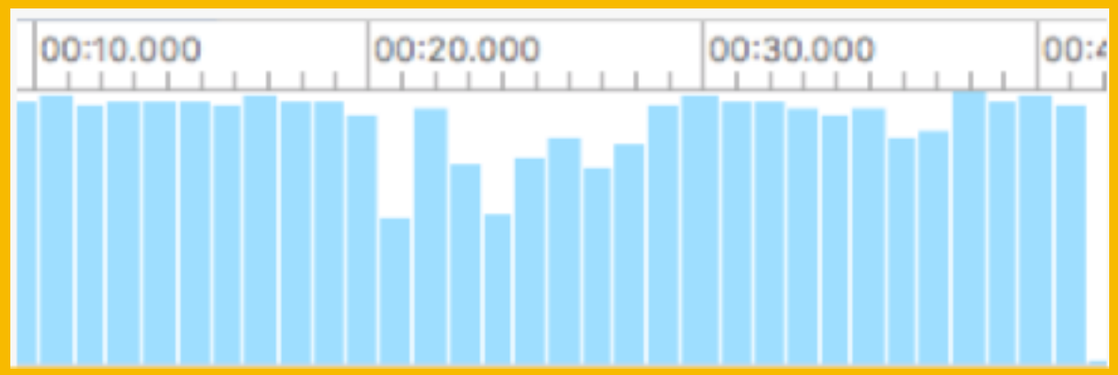

Native
Flutter Code
Widget
Rendering
Platform
Channels
Canvas
Events
System Services
(Location, Bluetooth, Audio,
Camera, etc.)

Flutter

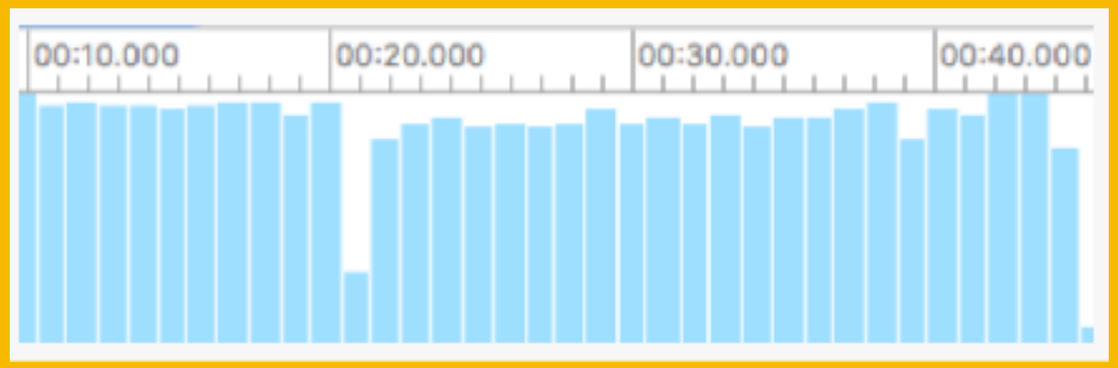
- Render with Canvas
- Develop with JIT, Run in AOT

Flutter VS RN

React Native



Flutter



- Performance (ListView FPS in 30 seconds)

Summary	Flutter	React Native	Hybrid
The costs of learning (language & framework)	4/8	3/8	2/4
Programming experience	6	8	9
Community support (StackOverFlow)	1,329	28,070	54,393
Hot reload	No	Yes	Yes
Invasive	8	10	5
Performance	8	6	4

PWA

- Web App Manifest
- Service Worker
- Push Notification

PWA

Safari 11.1

Safari 11.1 ships with iOS 11.3 and macOS 10.13.4. It is also available for macOS 10.12.6 and 10.11.6.

Highlights of Safari 11.1

- **Service Workers.** Implement background scripts for offline web applications and faster web pages.
- **Payment Request.** Provide a consistent user payment experience in Safari using a standards-based API.
- **Security Improvements.** Improved protection against memory corruption and code execution attacks.
- **Web Inspector Updates.** New designs for the Network Tab and the Styles sidebar in the Elements Tab.

Web APIs

- **New in Safari 11.1—Service Workers**
  - Added support for background scripts that can proxy network requests.
  - Added debugging for Service Workers to the Web Inspector.