Dart

# The rundown on Dart/Flutter Mobile Development

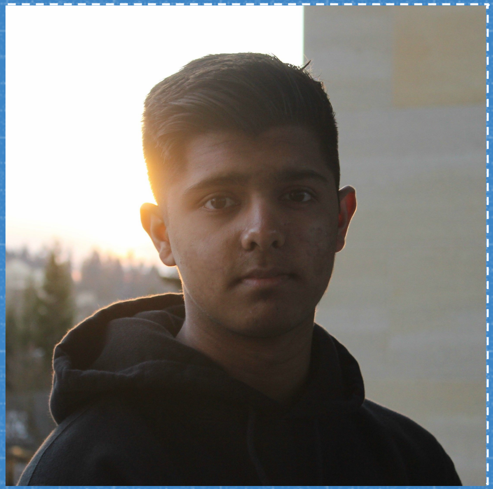
By Rohit Ganguly

# Introduction

## I’m Rohit!

I’m the President of Computer Science Club @Pitt!

I am an avid user of Flutter and I’m currently working on a Flutter plugin for content management.



## Why do I use Dart/Flutter?

- This summer, I interned at a startup doing mobile development in Dart/using Flutter.
- It’s new and open source! You can contribute to it!
- It’s straight up the best of its kind (as you’ll see)
- Excellent range of development tools and features that makes it a pleasure to work with

# 1

# What is Dart? Flutter?

I’m about to drop some knowledge

Dart is a language, Flutter is a framework/SDK.

# What is Dart? Why does Flutter use Dart?

- General purpose programming language made by Google
- Object Oriented, Class Defined language
  - Similar to C-style Syntax
    - Personally I think it’s a mix of Java/JavaScript.
  - Interfaces, Abstract  classes, etc.
- In this case, Dart is compiled AOT to native machine code.
  - More on this later
- Dart, relatively speaking, is **easy to learn.**
  - If you’ve taken CS 0401, you’re in a good spot.
- Flutter uses Dart due to their symbiotic relationship
  - Again, more on this later.

# What is Flutter?

- Flutter is a mobile SDK/Dart framework also made by Google.
- Flutter engine written in C++ (unfortunately not open source)
  - Language used to program is Dart
- Built of collections of Widgets
  - More on this later
- Also the framework for Google’s spooky Fuchsia OS project
  - Not too much information on this! We’ll be seeing it in the future, though.

2

# Modern Mobile Development

Why would I use Flutter??

# What are the methods of mobile development?

We have a few options:

- Native Code (OEM SDKs)
- WebViews
- Reactive Views
- Flutter

There’s also Xamarin, but we won’t cover that as it’s just as unique as Flutter (but requires a paid plan for commercial use).

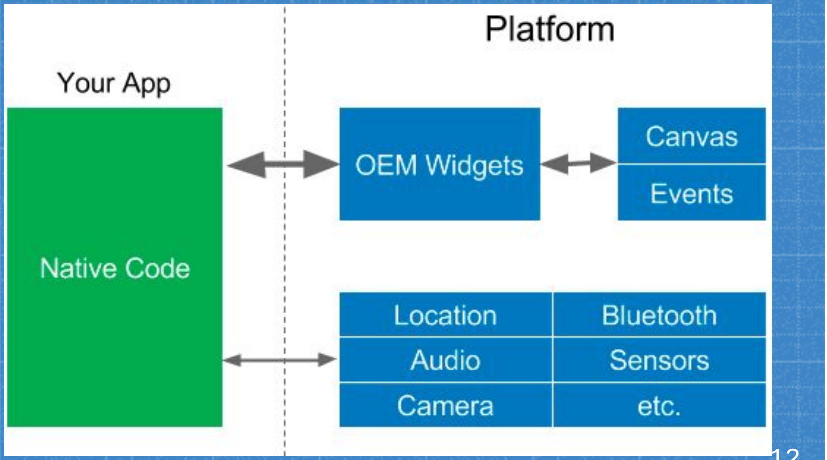
# Things to consider

- When assessing each option, we must look at multiple things.
  - Code to services communication
    - This is how the code connects to features like sensors, camera, etc
  - Code to display communication
    - This is for things like menus, buttons, etc. that are unique for each mobile operating system
    - Also just the method of accessing the display in general (how it’s being outputted)

“No cross platform solution to mobile development can replace native code.”

# Method 1: Native Code

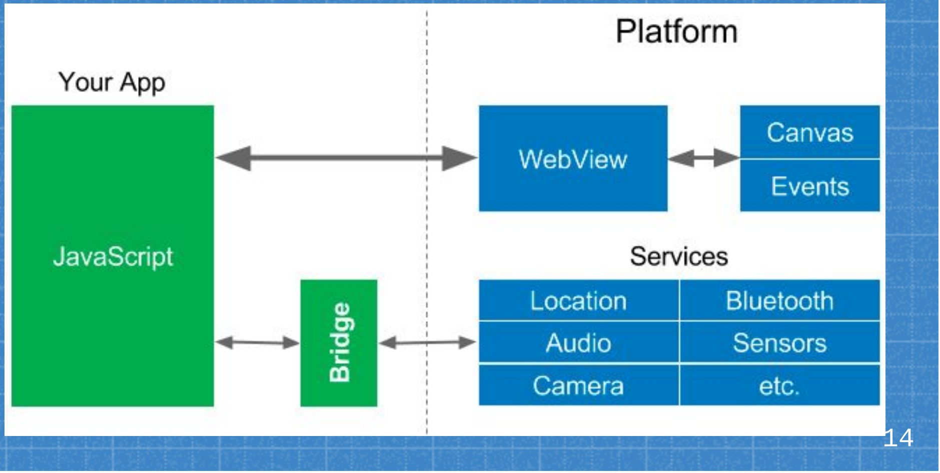
- **Two separate codebases** for iOS and Android (Objective-C/Swift vs Java/Kotlin)
- Your code talks directly to the platform for accessing widgets (which is good, as it is the most efficient form of access there is in all of mobile development)
- The cons?
  - Wildly different codebases for iOS/Android
  - Different teams/tools
  - Different design paradigms to follow for each



“This is fantastic. I want to write less code, though, this is a lot to write and maintain.”

# Method 2: WebViews

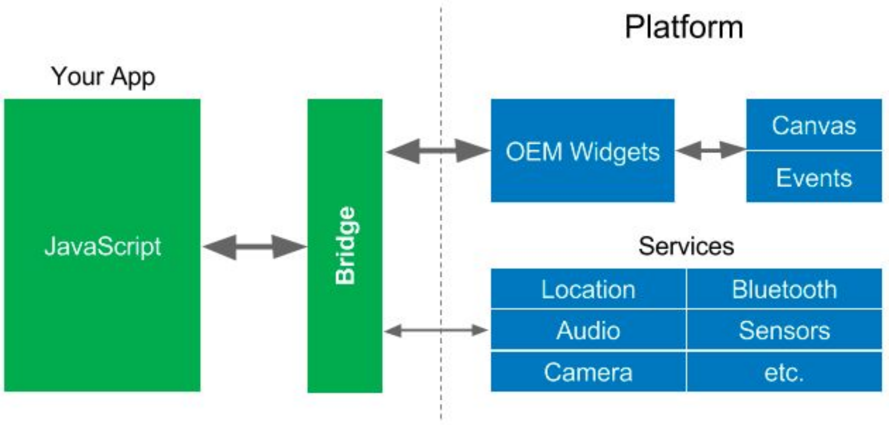
- A whole set of related frameworks (PhoneGap, Apache Cordova, Ionic, etc)
  - All used hardware’s WebView to simulate OEM widgets
- Code is HTML, CSS, JavaScript (*Web*View)
- Pro: easy to get started with
- Cons:
  - See the “Bridge”? Communication with JavaScript and app services is slow.
    - Not good for creating ‘fully-fledged’ apps.
  - You have to write JavaScript (/s)



“A valiant effort, but this isn’t ideal for creating *actual* apps that require a full mobile experience”

# Method 3: Reactive Views

- React Native is *good.*
  - *But,* it uses JavaScript
    - JavaScript -> Platform requires a bridge.
      - It didn’t last time for widgets since last time used the WebView and relied on HTML/CSS frontend to ‘simulate’ widgets.
  - Cons:
    - Widgets are used frequently (swiping, dragging, etc) and takes a lot of effort to access.



## Tal Kol, Founder of Orbs.com:

“Each realm by itself is blazingly fast. The performance bottleneck often occurs when we move from one realm to the other. In order to architect performant React Native apps, we must keep passes over the bridge to a minimum.”

***“We can do better”***

# Flutter is ***SPECIAL.***

- Flutter doesn’t have a JavaScript bridge.
  - Mic Drop
- It does, however have reactive-style views.
- How does it do both without the tradeoffs of React Native/WebView?

# The magic of Dart (part 1)

- Dart is compiled AOT into Native Code
  - Compiled Ahead of Time into Objective-C or Java
  - Compiled JIT for hot reload feature
- This allows the app to access OEM services, Canvas, and events without having to go through a bridge or a context switch.
- This ALSO improves app startup times.
- What is truly special about Flutter, however, is its Widget implementation.

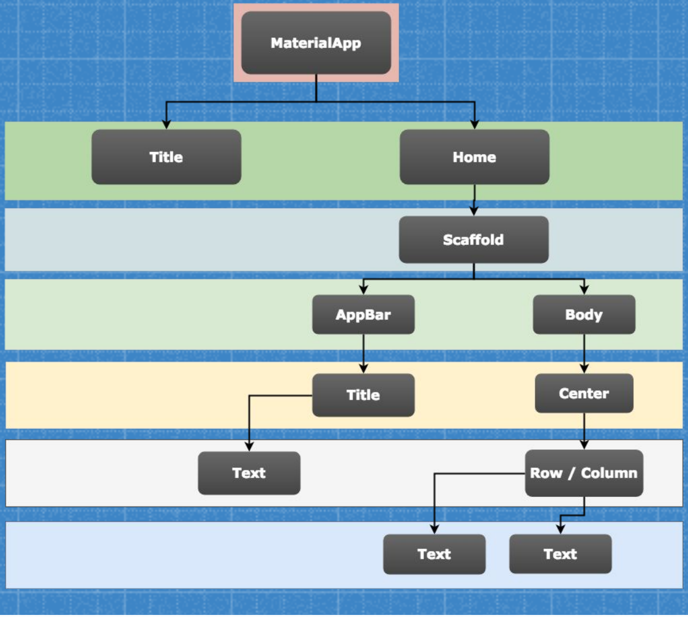
# What on earth is a Widget?

We’ll keep the definition as simple as possible.

- Widgets are a thing in Android (the little shortcut things that do part of a larger program)
  - This is NOT what we’re talking about.

- Widgets in this context are individual UI components (hamburger menus, buttons, text fields, etc) unique to an OS.

*“In Flutter,everything is a Widget”*



MaterialApp

Title

Home

Scaffold

AppBar

Body

Title

Center

Text

Row / Column

Text

Text

# Widgets are important!

- How can you tell the difference between an Android (Material Design) and an iOS (Cupertino) app?
- Widgets should perform as close to natively as possible
    - In React Native, everything’s a little slower thanks to that JS bridge, *but* the widgets look the same.
- Widgets **MUST** be customizable.
- Drum roll please...

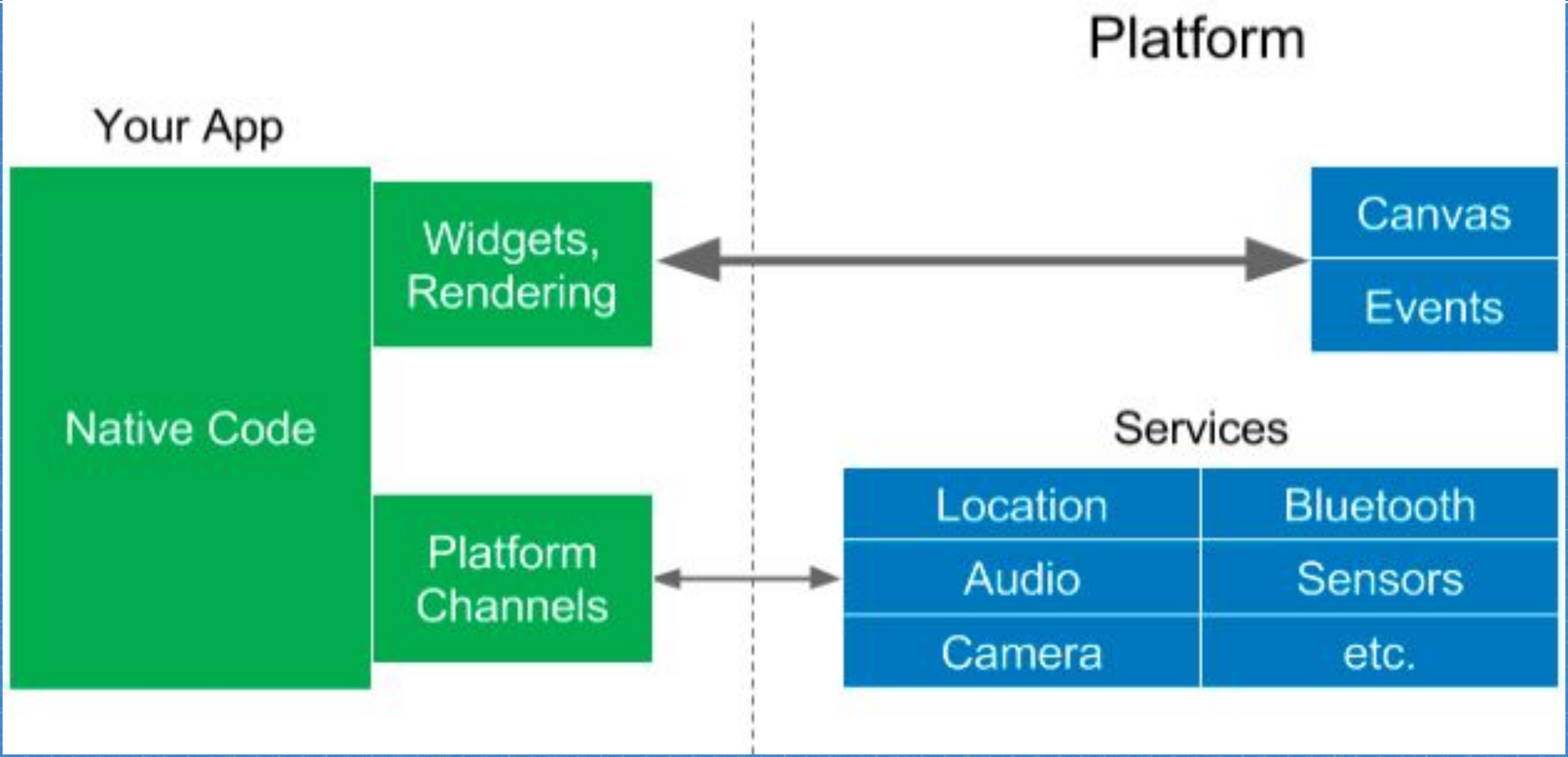
# Flutter Widget Handling

Flutter **does not** use OEM widgets *nor* WebView widget mimicking.

Instead, it uses its own (reactive) widgets designed to mimic the OS it is on (and follow design guidelines)

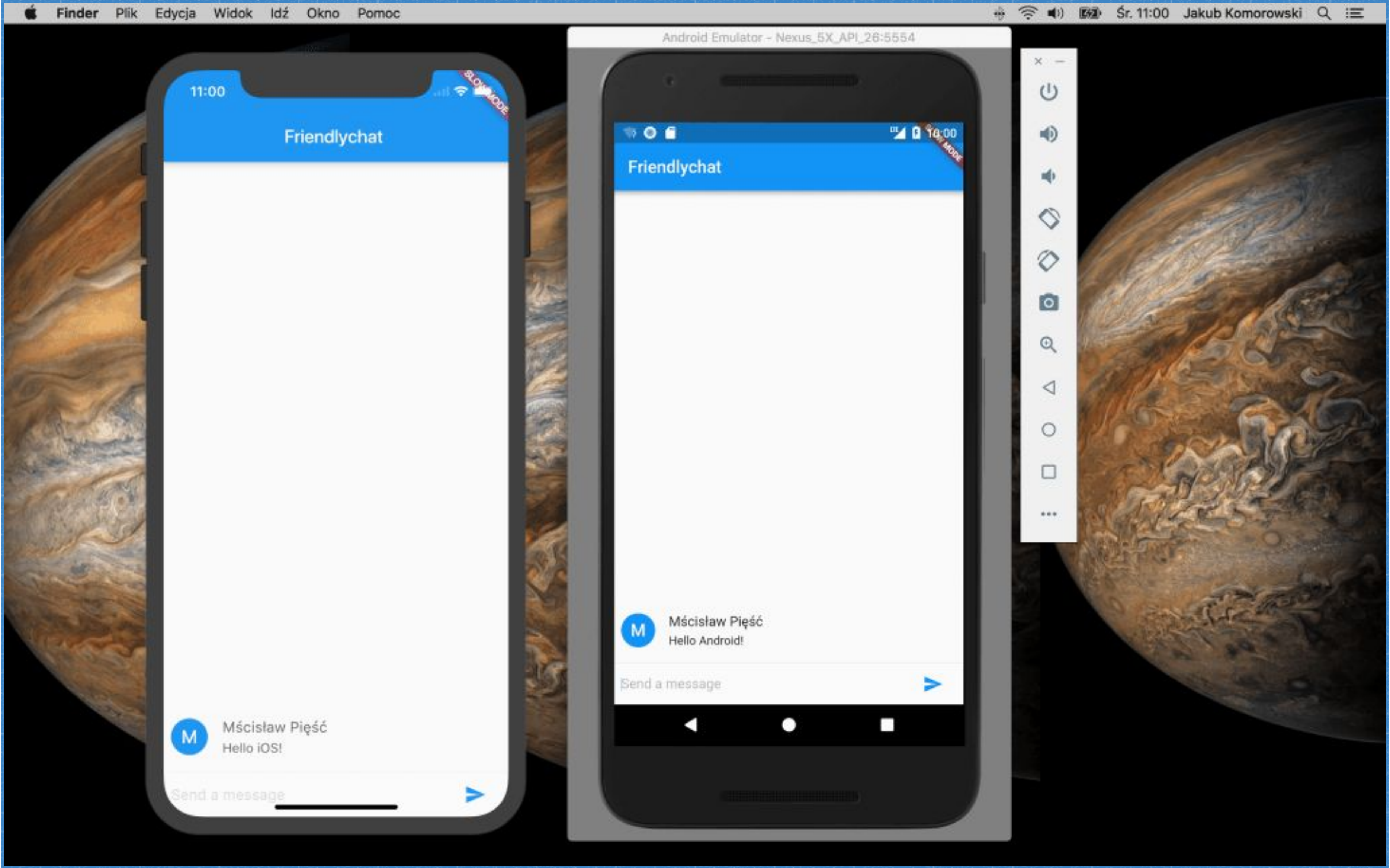
- Remember the compilation to native code? That’s is how it knows what to do.

Let’s take a look at the diagram….



Let’s notice a few things:

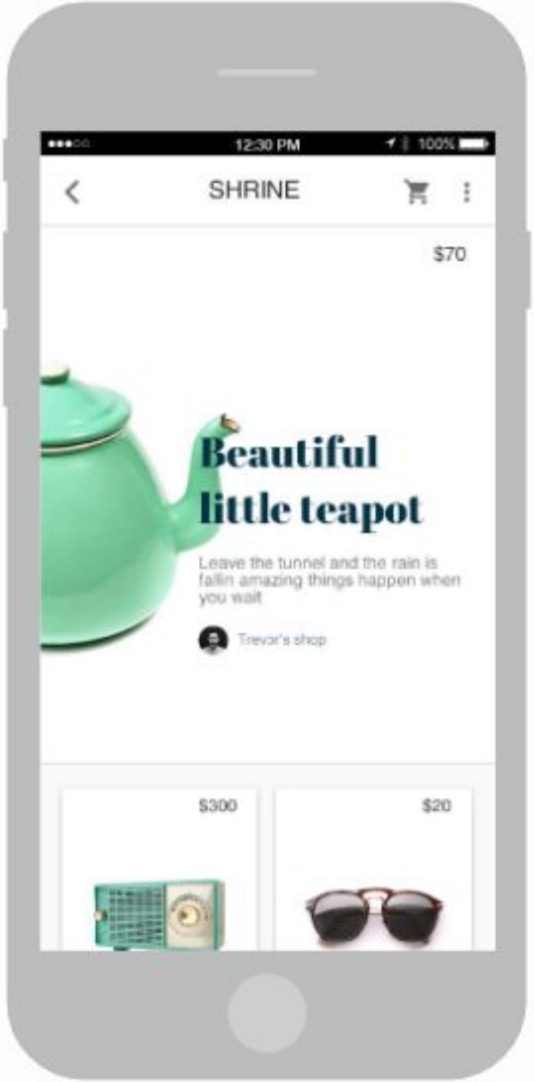
- The code handles a LOT of the things needed from a mobile app, legitimately all that it can.
- There’s an interface between the code and the platform, but this is *much* faster than a JS bridge.
- No JS bridge == much faster and smoother (closer to native experience)



Mścisław Pięść
Hello iOS!

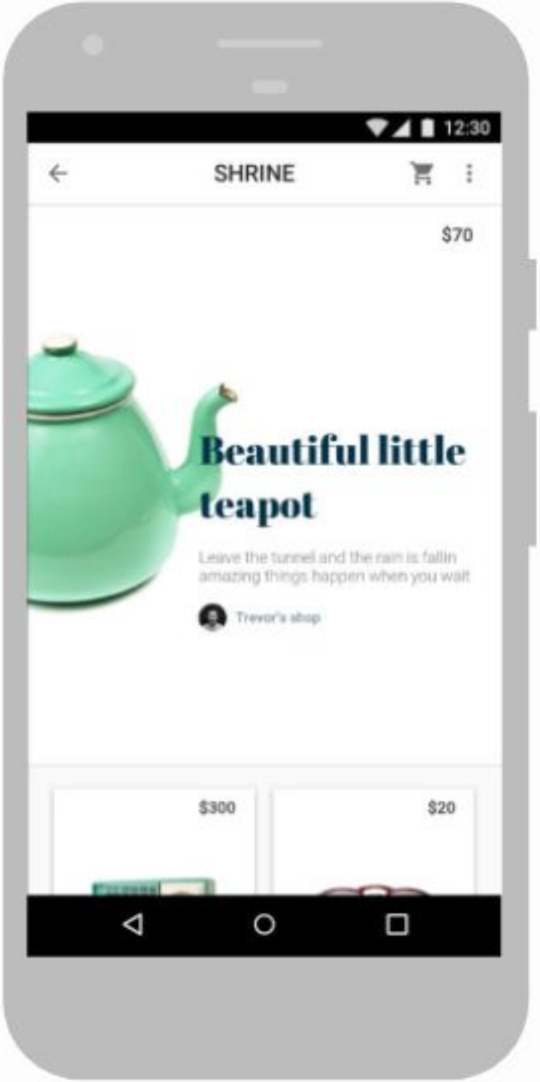
Mścisław Pięść
Hello Android!

iOS



Leave the tunnel and the rain is fallin amazing things happen when you wait

Android



Leave the tunnel and the rain is fallin amazing things happen when you wait

# 3

# Reactive Views

How do they work?

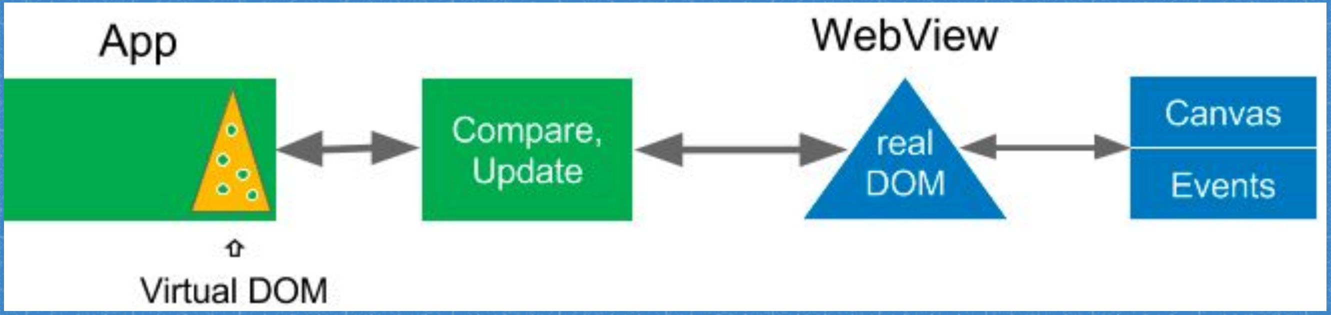
# DOM (Document Object Model)

The DOM is an API used by JavaScript to manipulate HTML documents. The Virtual DOM is an abstract version of this, created using objects in whatever programming language, in this case JavaScript.

# How’s this work in: Reactive WebViews?

The virtual DOM is immutable, and everytime an update is made, the virtual DOM is redrawn

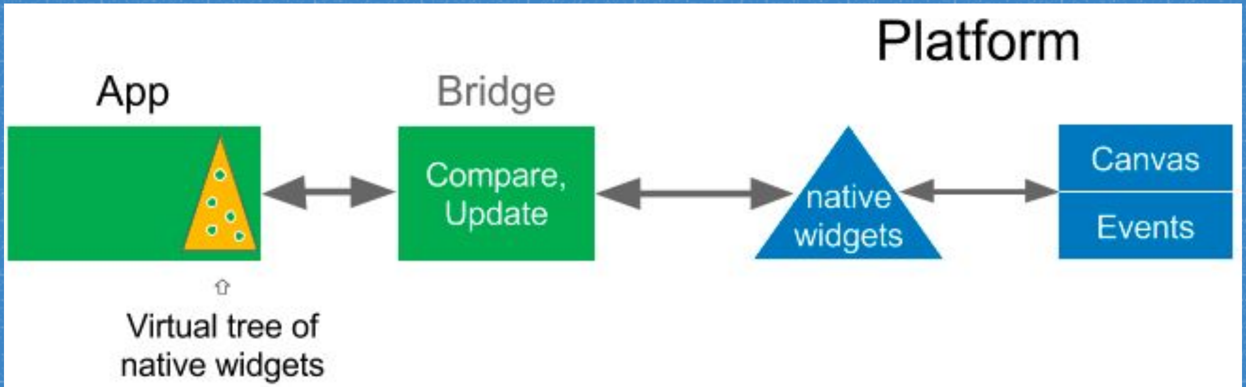
- This is because updating the HTML DOM is very resource expensive.
- After the redraw, the HTML DOM simply changes what’s different with the virtual DOM and itself.
  - Different from a full update



# How’s this work in: React Native?

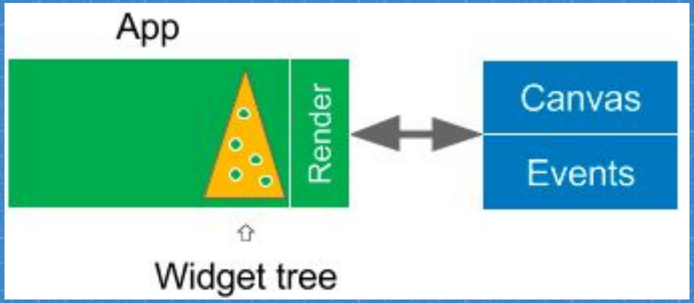
React Native is similar, but operates on the OEM widgets vs the HTML DOM of the WebView.

- React Native needs to communicate with native widgets, so it has a virtual tree that it operates whenever changing widgets and does the same comparison operation to prevent excess overhead.
- Once changes are made, it is rendered to the canvas.



# As you can guess, Flutter does this better.

- Flutter has no OEM widgets to access (it *is* the widgets it needs to access).
- When it renders widgets, it only renders widgets that need updating
  - Unused widgets are bit blitted
    - Won’t be getting into that, don’t worry.
    - Just know that this allows Flutter to render scrolling and all that stuff much easier/faster.



# Dart is special: part 2

- Dart’s garbage collector also is well-designed for mobile development, as it uses generational collection, which makes the dozens of widget operations needed for a mobile app cheap in terms of overhead.

- Dart also uses a ‘tree-shaking’ compiler, which only compiles code you need to use.
  - This allows you to use large widget libraries and not affect compile time.

4

# User Benefits of Flutter

Almost at the coding part.

Flutter is **extremely** dev-friendly.

- The concept is already dev-friendly
  - 1 codebase for both major OSes
- Hot Reload
  - Similar to HTML/CSS/JavaScript refreshing after saving code
- High Version Compatibility
- Very High Customizability
- Open Source

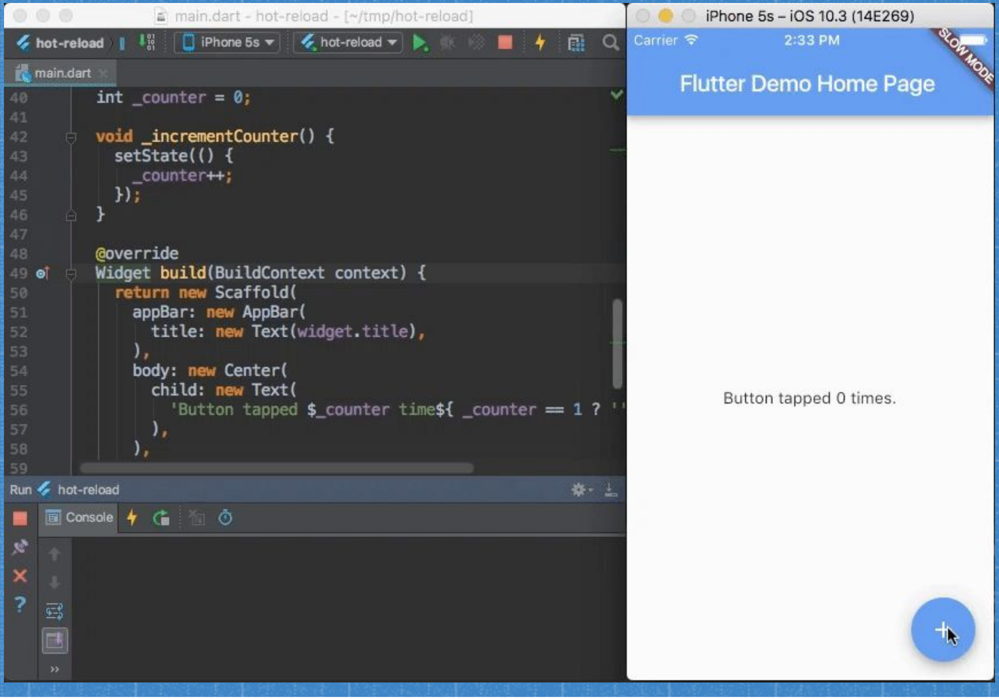
We just went over why Flutter is the best cross-platform, single codebase solution right now

# Hot Reload

## Easily one of the best dev tools available in mobile development

While running your app in a simulator/device and programming, you can save the code and it automatically ‘reloads’ the app (without re-compiling entirely)

- Saves a *ton* of time



```
int _counter = 0;

void _incrementCounter() {
  setState(() {
    _counter++;
  });
}

@override
Widget build(BuildContext context) {
  return new Scaffold(
    appBar: new AppBar(
      title: new Text(widget.title),
    ),
    body: new Center(
      child: new Text(
        'Button tapped $_counter time${ _counter == 1 ? ''
      ),
    ),
```

Flutter Demo Home Page

Button tapped 0 times.

# High Version Compatibility

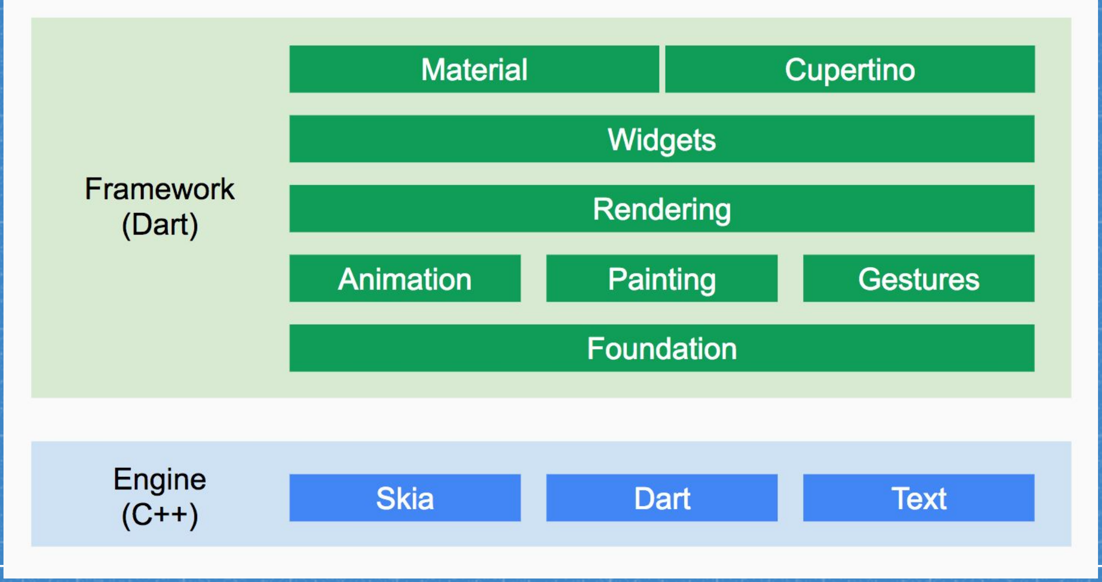
- Because Widgets are built-in to the language, there’s no compatibility packages needed to make sure your app works on all versions.
- Google works hard to keep all widget definitions up to date and design specs - nothing to worry about there
- Widgets will never change unless you specifically want them to
  - App will look as the developer intended
    - May not be the case in the event of an OS/design update by Android or iOS
  - Your brand design will remain intact

# Very High Customizability

- Dart has a *ton* of packages and plugins
  - Firebase access
  - Redux data store access
  - Specific plugins make life easier
    - Accelerometer plugin
    - Camera plugin
    - Maps plugin
    - Location, so much more
- Widget systems are easily customizable
  - You can make your app look completely different than anything else if you’d like, without much effort.
  - Themes allow your app to get its own identity
    - Redefines existing widgets (color, size, shading, etc.)

## OPEN SOURCE

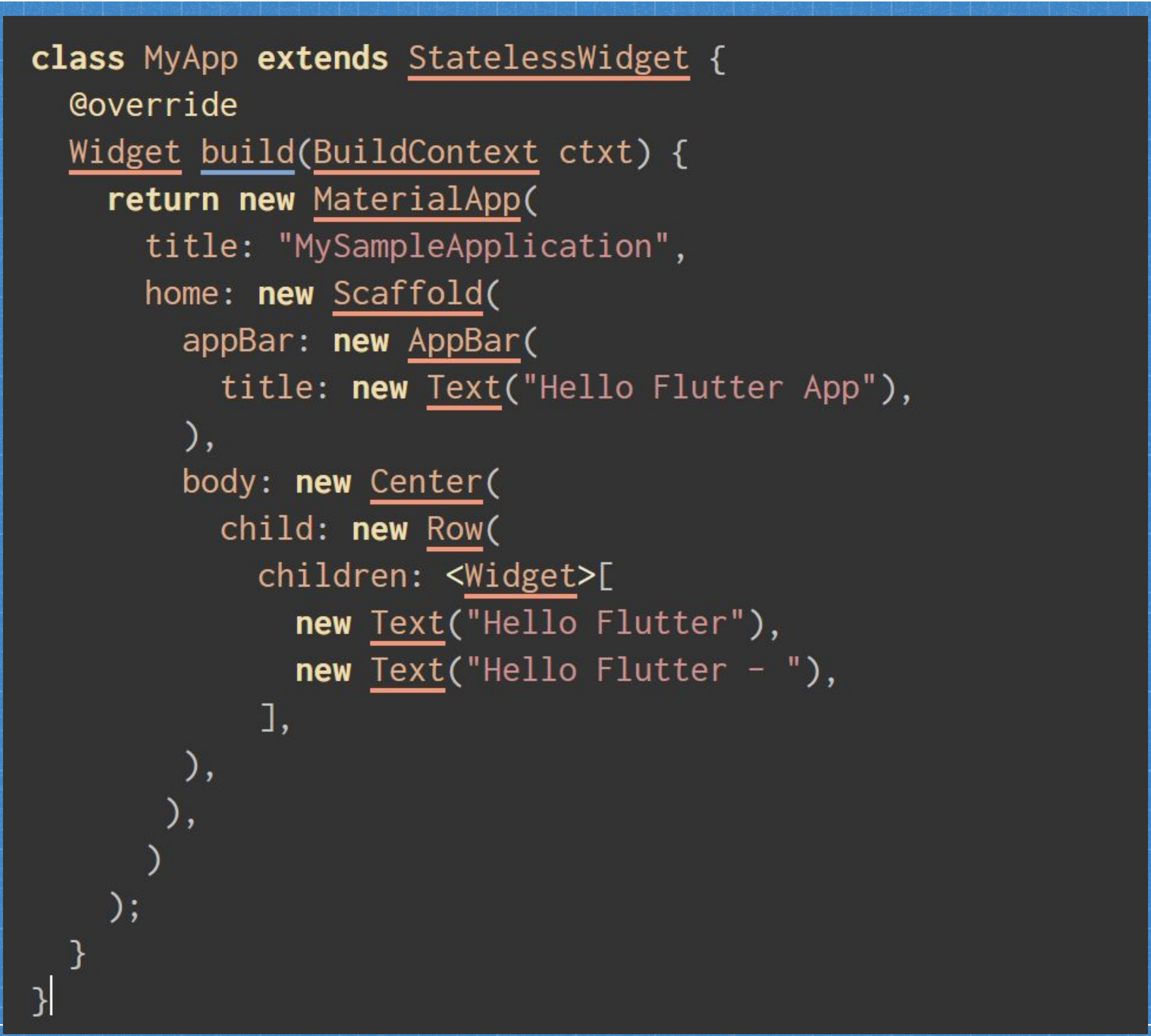
- Flutter is open source, meaning that you can change practically anything about it that you don’t like.
- Everything in green in the picture you can change to fit your needs.
- Make your own plugins/packages!



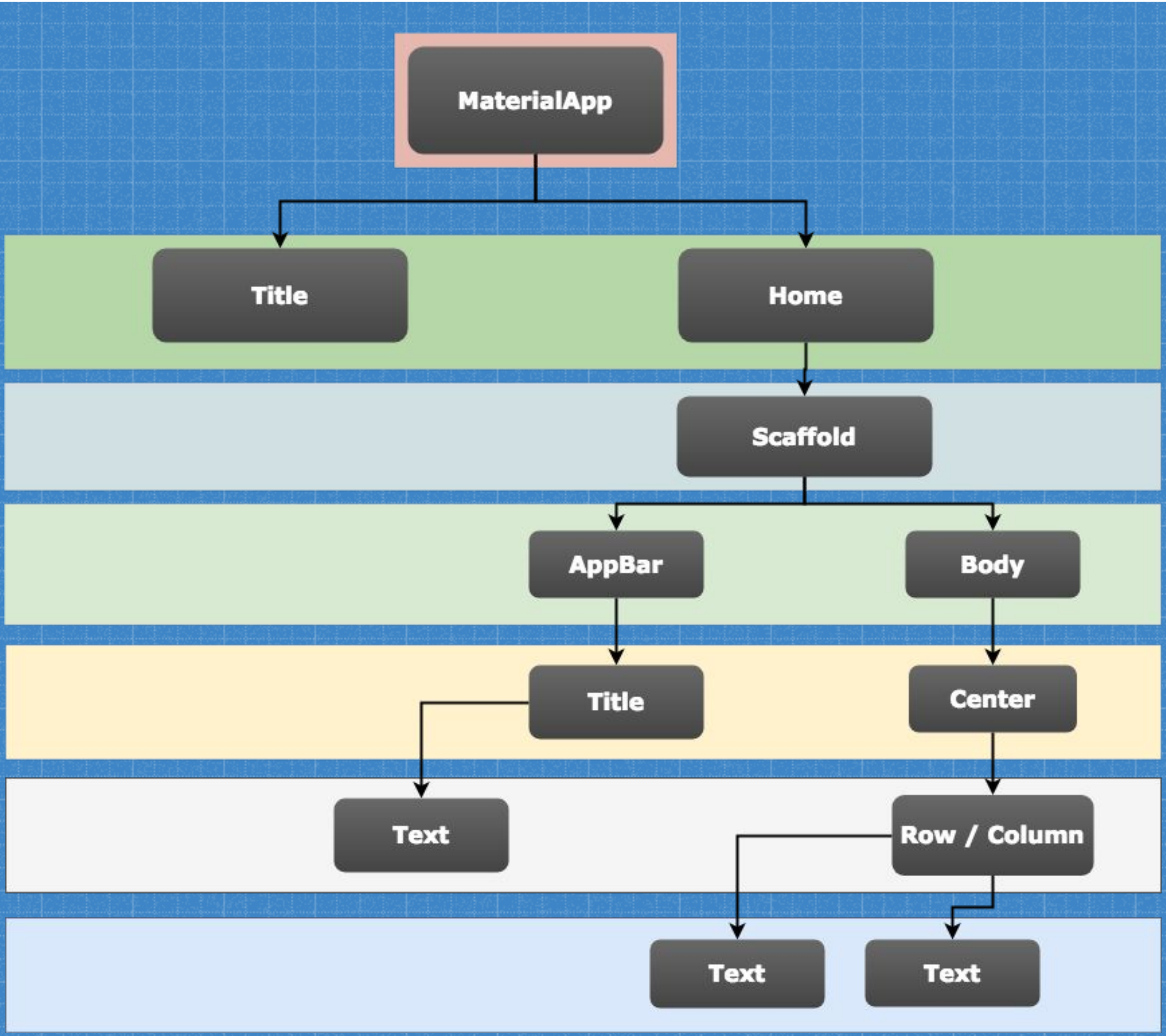
5

# Let’s take a look at some code!

Finally.



```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext ctxt) {
    return new MaterialApp(
      title: "MySampleApplication",
      home: new Scaffold(
        appBar: new AppBar(
          title: new Text("Hello Flutter App"),
        ),
        body: new Center(
          child: new Row(
            children: <Widget>[
              new Text("Hello Flutter"),
              new Text("Hello Flutter - "),
            ],
        ),
       ),
      )
    );
  }
}
```


MaterialApp
Title
Home
Scaffold
AppBar
Body
Title
Center
Text
Row / Column
Text
Text

Widget Tree

# Stateless vs Stateful Widgets

**Stateful**

- Dynamic
- Keeps track of state
- Conventions are to add underscores before starting them

**Stateless**

- Static data
- Immutable
- Stateless widgets can store stateful widgets

Let’s take a look at some more code and mess around with it!

# 6

# Recap

Almost done!

# Recap

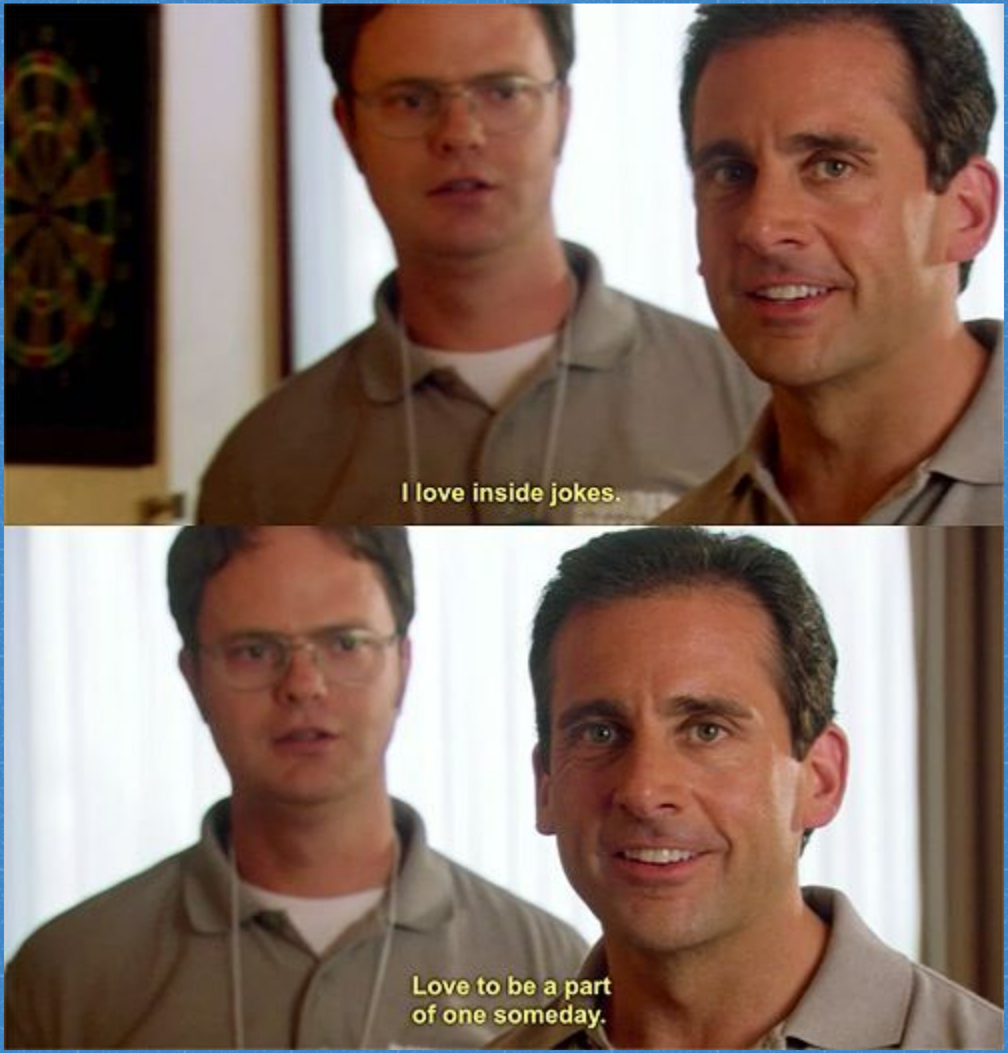
- Flutter - open source, flexible, fast, new!
- Widget-based architecture
- Very catering to developers
- Code similar to Java/JavaScript
- **Cross-Platform, Single Codebase**

## Suggestions

- If you’re looking for tutorials, check out https://flutter.io!
- If you want to work on a Flutter project, let me know!
- Dart/Flutter is taking off really fast
    - If you’re interested in mobile development, it’ll be a good skill to have
- Open source community is very active! Come join us :-)

# References

- Hackernoon
  - https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514
- Smashing Magazine
  - https://www.smashingmagazine.com/2018/06/google-flutter-mobile-development/
- Tal Kol’s Medium
  - https://medium.com/@talkol/performance-limitations-of-react-native-and-how-to-overcome-them-947630d7f440
- Flutter.io!
  - https://flutter.io



Be a part of something new and cutting edge!

# Thanks!

## ANY QUESTIONS?

You can find me at:

@rohit on Pitt CSC slack

rohitganguly.com

LinkedIn: Rohit Ganguly