FAKULTAS
ILMU
KOMPUTER

Introduction to Dart Programming Language and Flutter Framework

Tim Dosen PBP

Dart Programming Language

Try Dart

Stepping stone:

https://dart.dev/tutorials/server/get-started

https://dartpad.dev/

And then explore!

NB: starts with main() method

About Dart

- Dart is a programming language that optimized and specialized around the needs of user interface creation.
- Basic example of the language:

```
// Define a function.
void printInteger(int aNumber) {
  print('The number is $aNumber.'); // Print to console.
}

// This is where the app starts executing.
void main() {
  var number = 42; // Declare and initialize a variable.
  printInteger(number); // Call a function.
}
```

More on Dart Syntax: https://dart.dev/samples

Important Concepts in Dart

- Everything you can place in a variable is an object, and every object is an instance of a class
- Although Dart is strongly typed, type annotations are optional because Dart can infer types.
  - Typed languages are the languages in which we define the type of data type and it will be known by machine at the compile-time or at runtime.
    1. Statically typed languages: the data type of a variable is known at the compile time which means the programmer has to specify the data type of a variable at the time of its declaration. Examples: C, C++, and Java.
    2. Dynamically typed language: In these languages, interpreters assign the data type to a variable at runtime depending on its value. We don't even need to specify the type of variable that a function is returning or accepting in these languages. Examples: JavaScript, Python, Ruby, and Perl.

https://dart.dev/guides/language/language-tour#important-concepts

Important Concepts in Dart

- If you enable null safety, variables can't contain null unless you say they can. You can make a variable nullable by putting a question mark (?) at the end of its type.
- When you want to explicitly say that any type is allowed, use the type Object? (if you've enabled null safety), Object, or — if you must defer type checking until runtime
- Dart supports generic types, like List<int> (a list of integers) or List<Object> (a list of objects of any type).

https://dart.dev/guides/language/language-tour#important-concepts

Important Concepts in Dart

- Dart supports top-level functions (such as main()), as well as functions tied to a class or object (static and instance methods, respectively). You can also create functions within functions (nested or local functions).
- Similarly, Dart supports top-level variables, as well as variables tied to a class or object (static and instance variables). Instance variables are sometimes known as fields or properties.
- Unlike Java, Dart doesn't have the keywords public, protected, and private. If an identifier starts with an underscore (_), it's private to its library.
- Identifiers can start with a letter or underscore (_), followed by any combination of those characters plus digits.
- Dart has both expressions (which have runtime values) and statements (which don't).
- Dart tools can report two kinds of problems: warnings and errors.

https://dart.dev/guides/language/language-tour#important-concepts

Please take your time to try run Fibonacci source code sample from:
https://dart.dev/samples

Using DartPad
https://dartpad.dev/

Flutter Framework

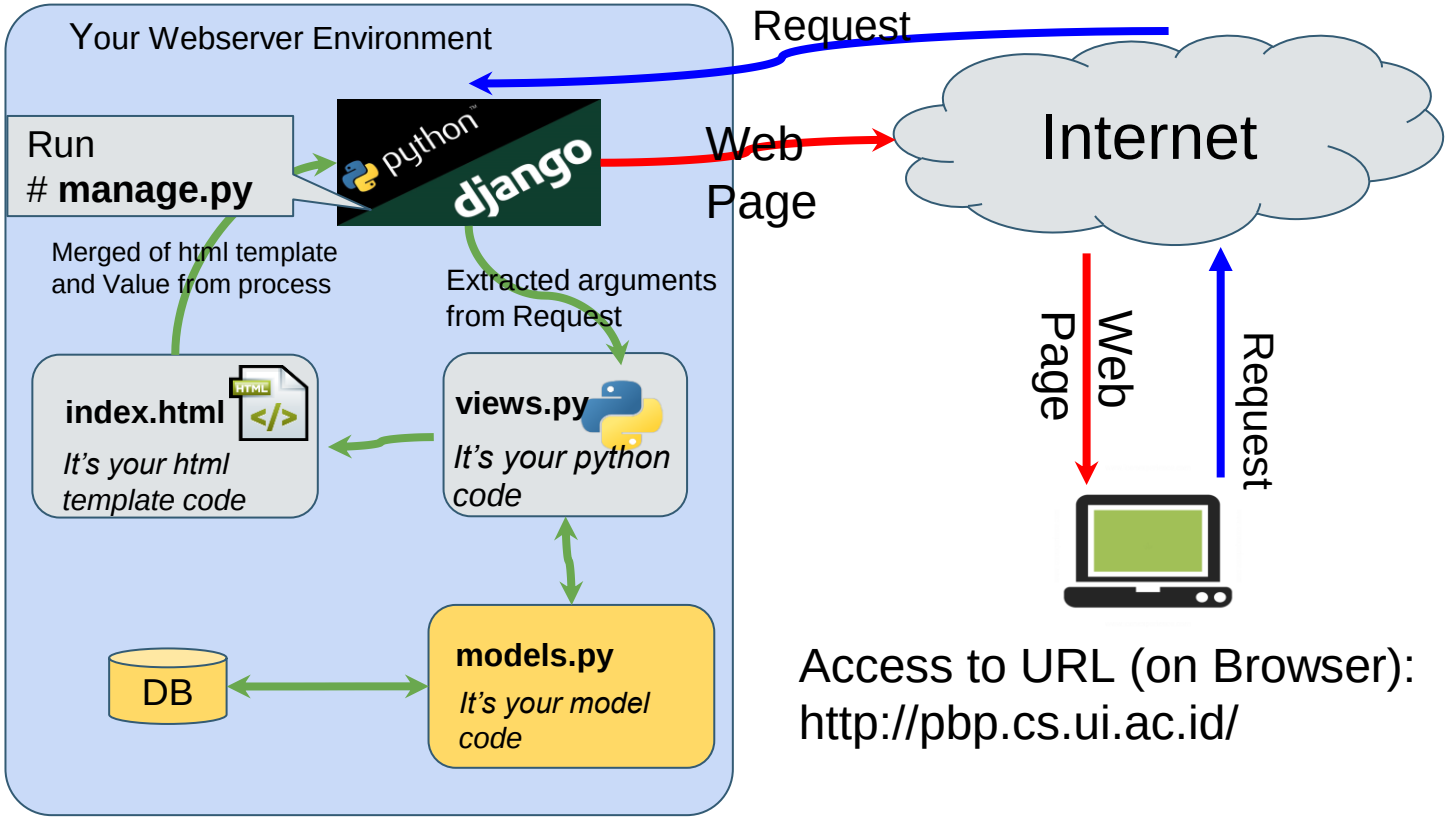
Installation & Test Drive

Installation
https://flutter.dev/docs/get-started/install

Test Drive
https://flutter.dev/docs/get-started/test-drive?tab=terminal#create-app

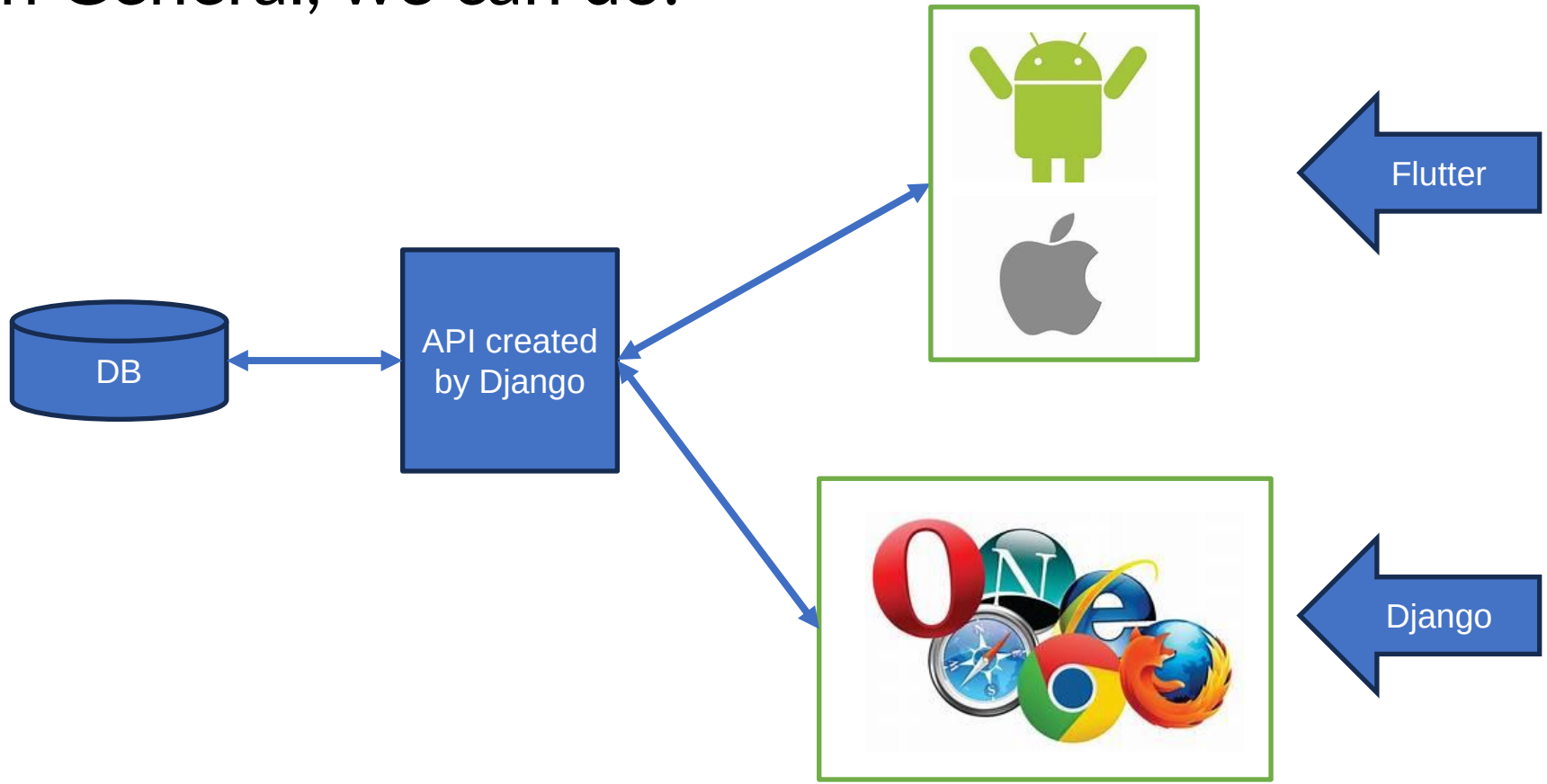
Mobile Platform Development Framework

Review: Django Architecture

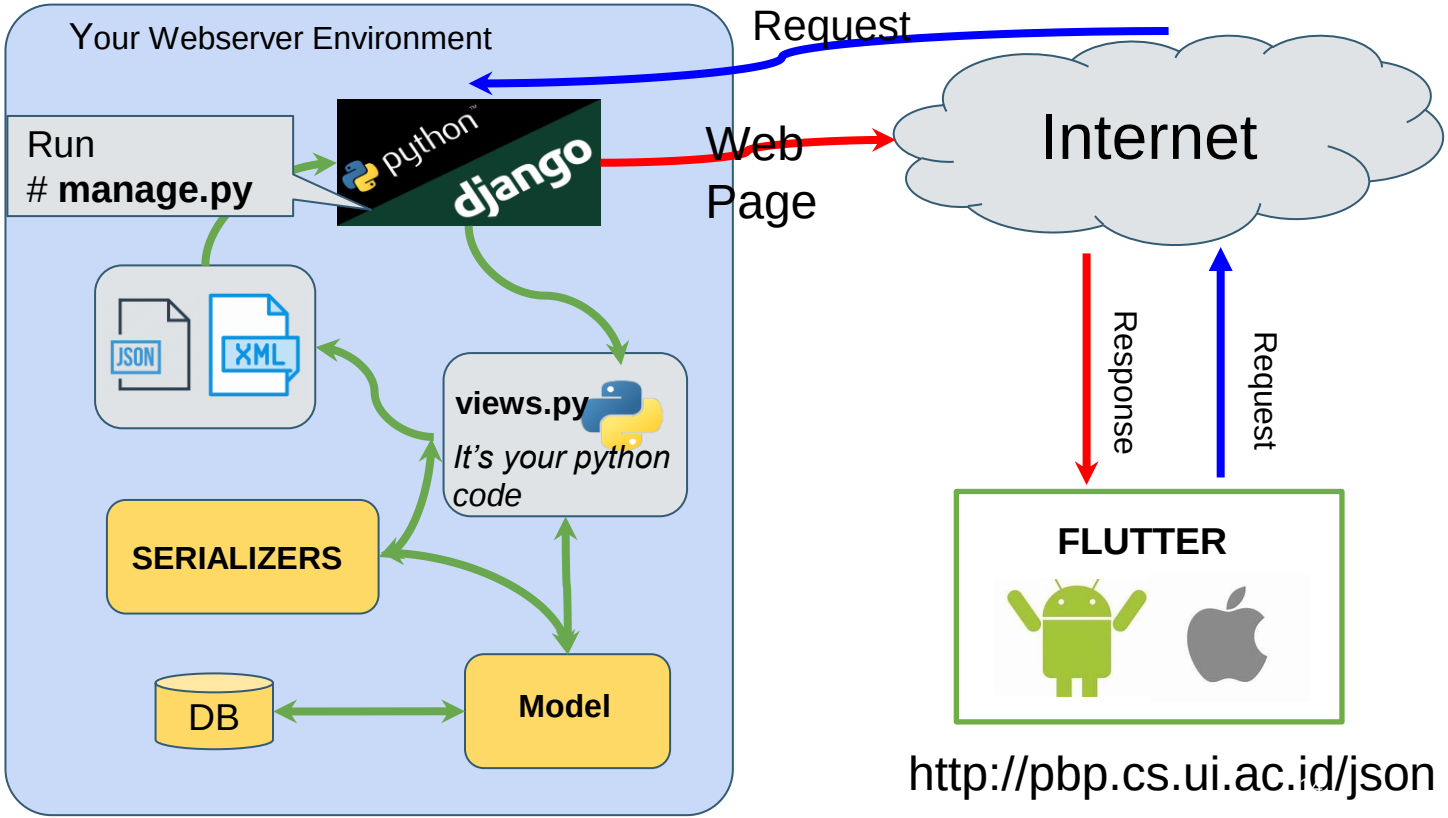


Access to URL (on Browser):
http://pbp.cs.ui.ac.id/

In General, we can do:



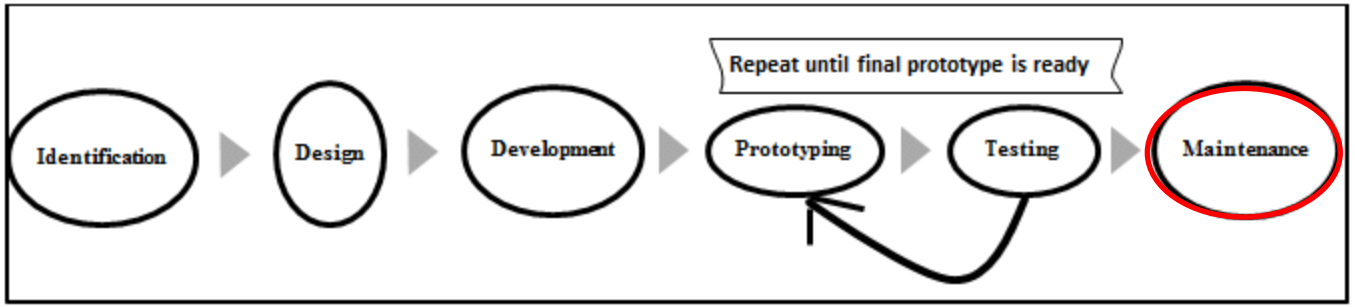
Django and Flutter Architecture



14

Mobile Platform Structure and Development Life Cycle

Mobile App Development Life Cycle (MADLC)



Web app diinstal di server.
Mobile app diinstal di handphone user.

https://www.researchgate.net/publication/276129115_Suitability_of_Existing_Software_Development_Life_Cycle_SDLC_in_Context_of_Mobile_Application_Development_Life_Cycle_MADLC

Development Environment

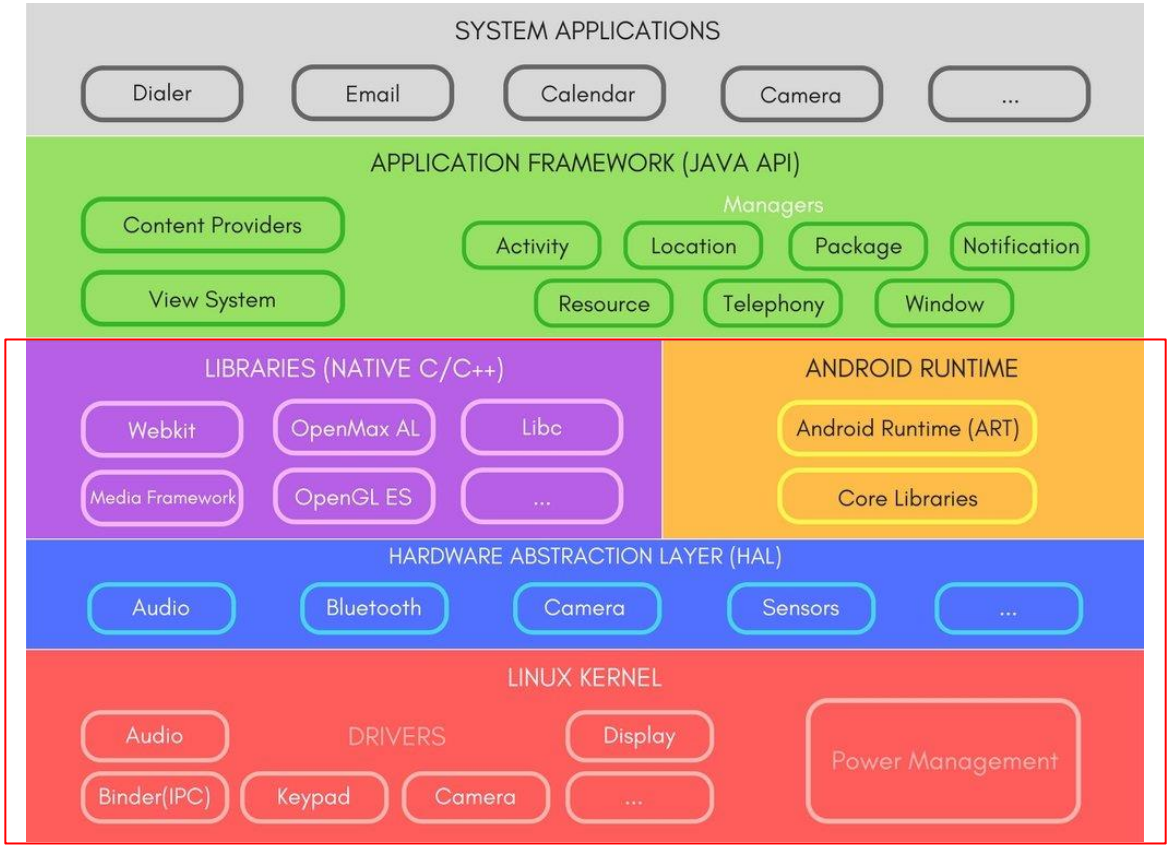
Development Environment

- Android’s integrated development environment: Android Studio
- Programming languages: Java, Kotlin, JavaScript, C++

- Apple's integrated development environment: Xcode
- Programming languages: Swift, C

Native Android Architecture

Source: https://www.studytonight.com/android/android-architecture



It's where Your Apps

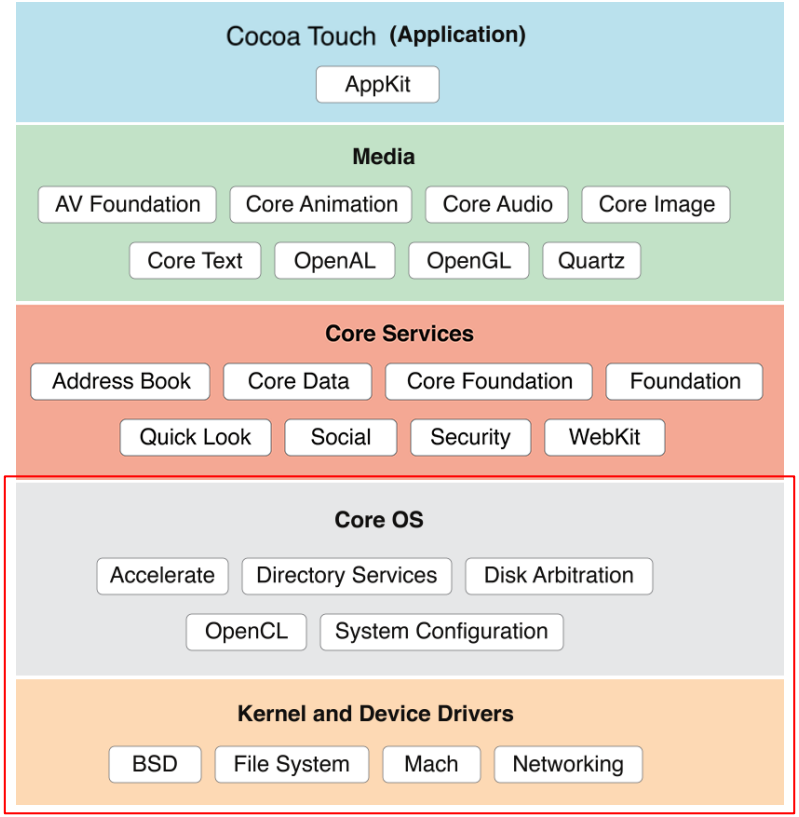
It's where Your Code uses Android Framework

Android OS & Library

Hardware Driver

Native iOS Architecture

Source: https://www.dotnettricks.com/learn/xamarin/understanding-xamarin-ios-build-native-ios-app



It's where Your Apps

It's where Your Code uses iOS Framework

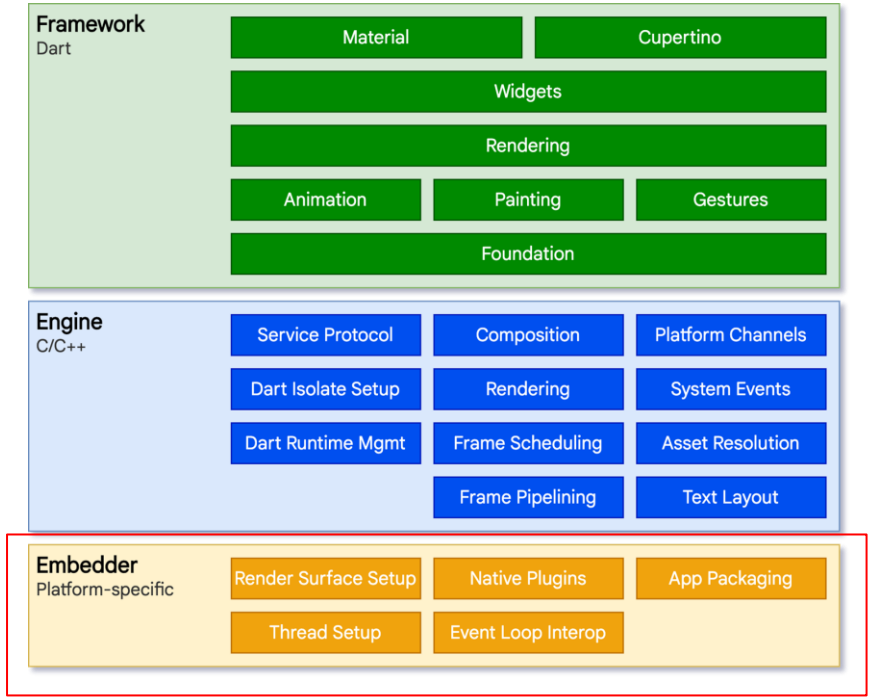
iOS OS & Library

Hardware Driver

About Flutter

Flutter is Google's UI toolkit for building natively compiled applications for mobile, web, desktop, and embedded devices from a single codebase.

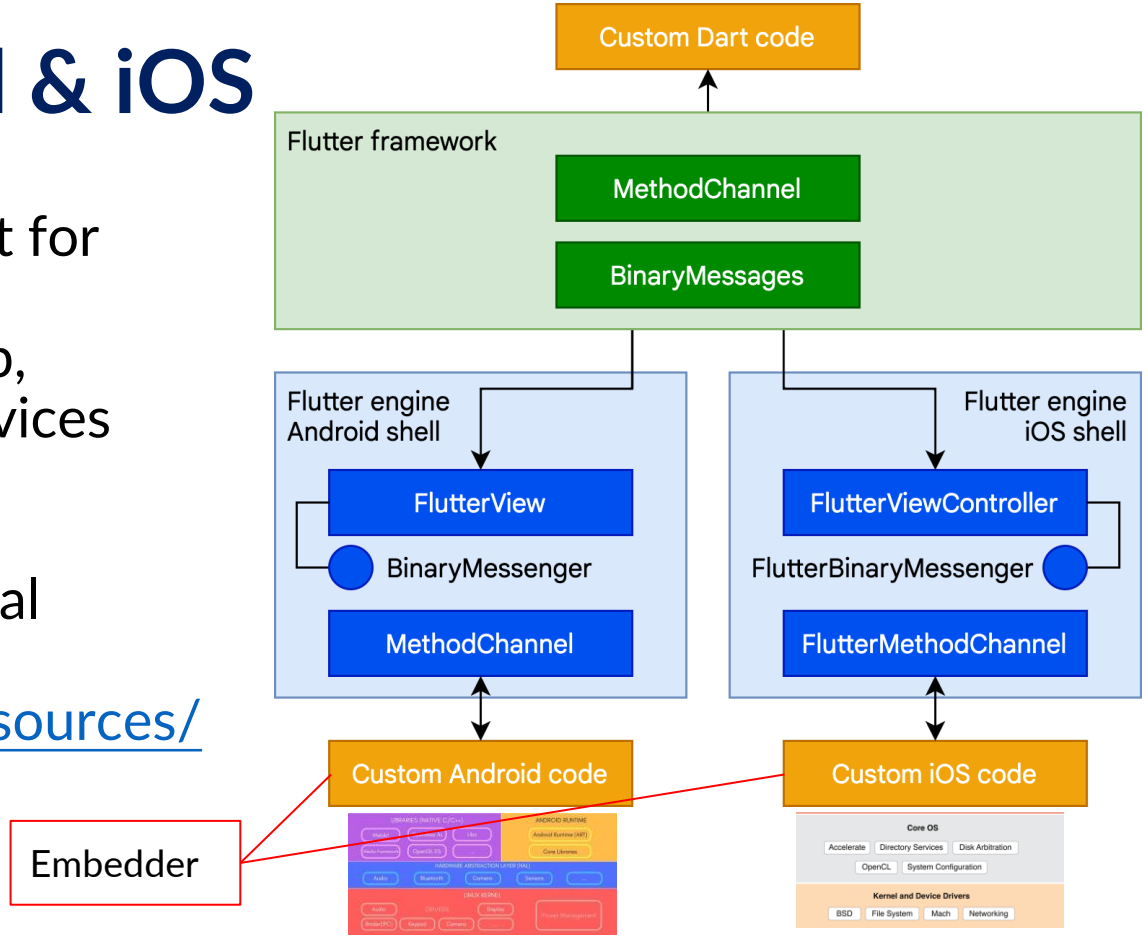
More on Flutter architectural overview: https://flutter.dev/docs/resources/architectural-overview



Flutter on Android & iOS

Flutter is Google's UI toolkit for building natively compiled applications for mobile, web, desktop, and embedded devices from a single codebase.

More on Flutter architectural overview: https://flutter.dev/docs/resources/architectural-overview



Why Flutter?

Design beautiful apps

Problems Flutter wants to solve

1. Long/more expensive development cycle
2. Multiple language to learn
3. Long build/compile time
4. Existing cross-platform solutions side-effects

Compared to previous existing framework, Flutter has:

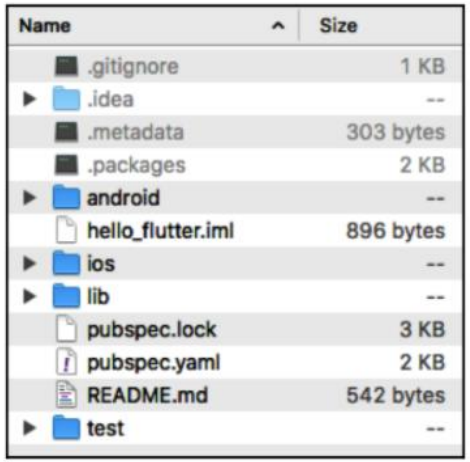
- High performance (owns the pixel, no extra layer, native)
- Full control of the user interface
- Dart language (declarative UI, syntax to layout, etc)
- Being back by Google
- Open Source
- Lots of developer resources and tooling (google codelabs, etc)

Project Structure

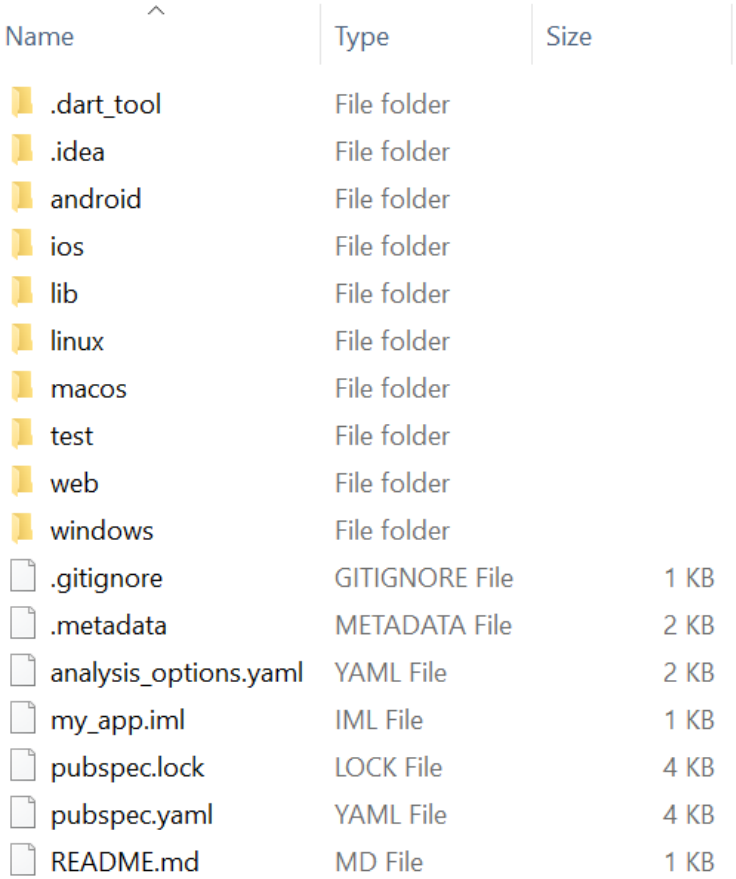
The “hello world”

```
flutter create <output_directory>
```

Example 1: **flutter create hello_flutter**
The code for your app is in **lib/main.dart**

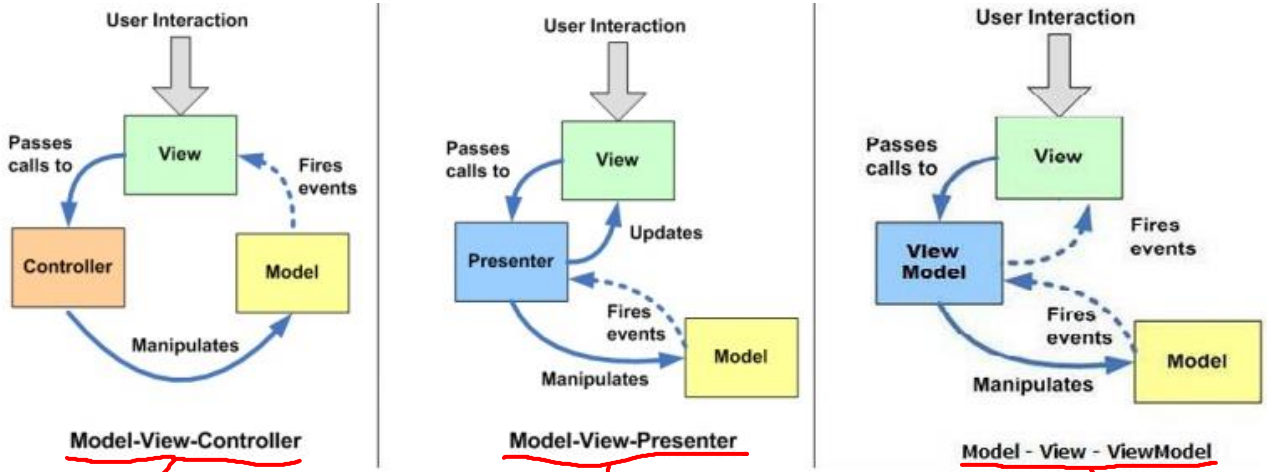


Example 2: **flutter create my_app**



Widgets (Stateless & Stateful)

MVC vs MVP vs MVVM Architecture



- In MVC, every action in the View correlates with a call to a Controller along with an action.
- The controller is responsible for handling the User Input and then updating the Model or the View.

- When the view notifies the presenter that the user has done something (for example, clicked a button), the presenter will then update the model and synchronize any changes between the Model and the View.
- One important thing to mention is that the Presenter doesn't communicate directly to the view. Instead, it communicates through an interface. This way, the presenter and the model can be tested in isolation.

- The term ViewModel means "Model of a View", and can be thought of as abstraction of the view, but it also provides a specialization of the Model that the View can use for data-binding.

What is Widget?

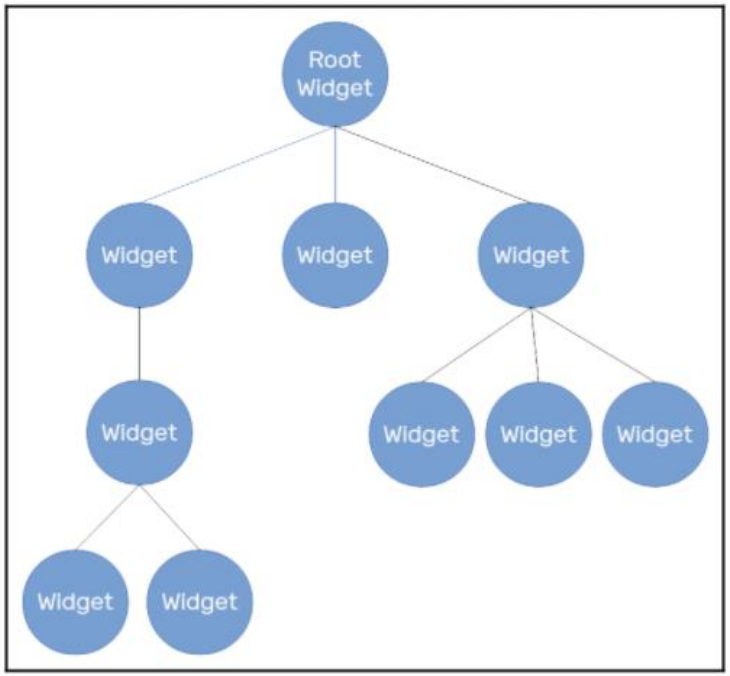
- It’s a reusable component
- Remember MTV?
- Flutter is component based => MVVM => Model, View, View-Model
- Widget is a component.

  - **Widgets** can be understood as the visual representation of parts of the application. And, also handle its own the behaviour.
  - Widgets are put together to compose the UI of an application.
  - Imagine it as a puzzle in which you define the pieces.

Everything is Widget!

- A visual/structural element that is a basic structural element, such as the Button or Text widgets
- A layout specific element that may define the position, margins, or padding, such as the Padding widget
- A style element that may help to colorize and theme a visual/structural element, such as the Theme widget
- An interaction element that helps to respond to interactions in different ways, such as the GestureDetector widget

Widget Tree



The widget tree is the logical representation of all the UIs widgets. It is computed during layout (measurements and structural info) and used during rendering (frame to screen) and hit testing (touch interactions), and this is the things Flutter does best.

Widgets are represented in the tree as nodes. It may have a state associated with it; every change to its state results in rebuilding the widget and the child involved.

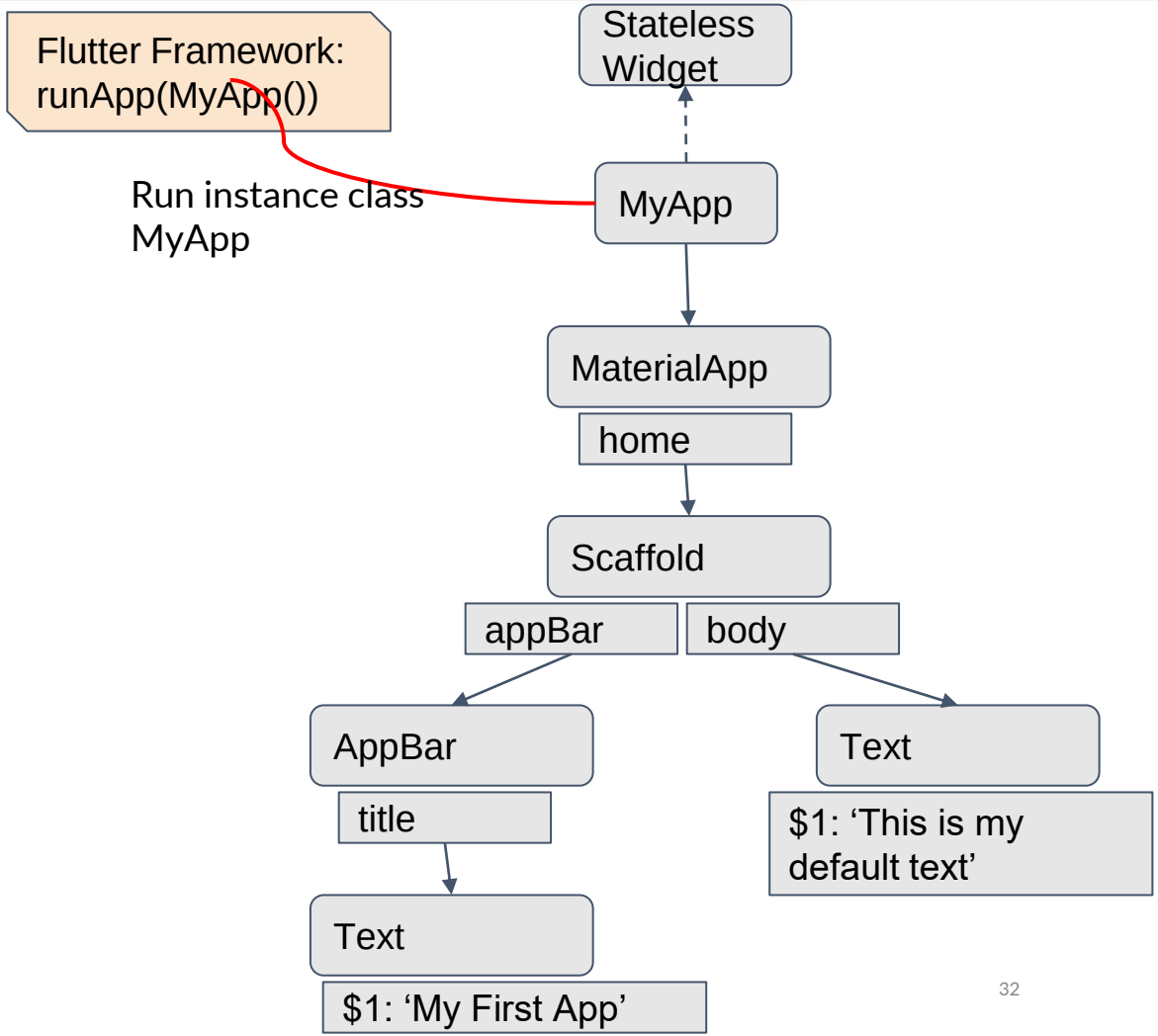
What is Widget?

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

// void main() {
//   runApp(MyApp());
// }

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('My First App'),
        ),
        body: Text('This is my default text!'),
      ),
    );
  }
}
```



Understanding “State”

In General

State is Data/ Information used by your App

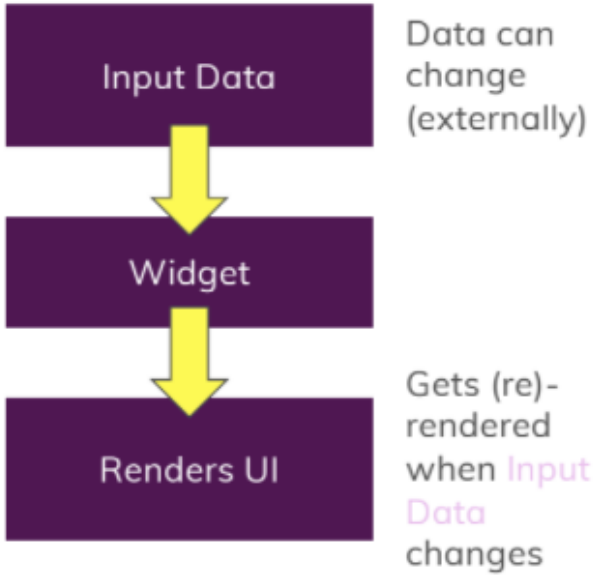
App State

Authenticated Users
Loaded Jobs
...

Widget State

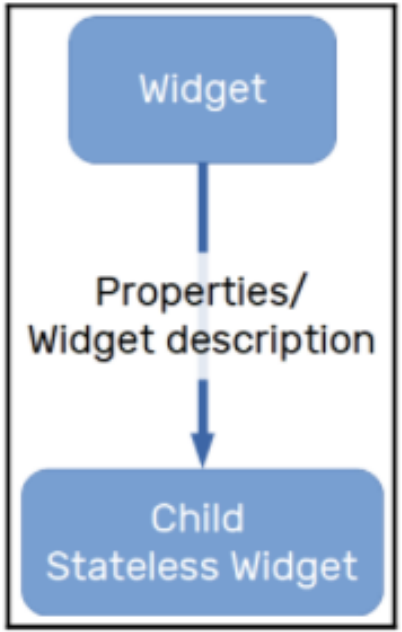
Current User Input
Is a Loading Spinner being shown?
...

Stateless Widget



- They do not have a state; they do not change by themselves through some internal action or behavior
- They are changed by external events on parent widgets in the widgets tree.
- Give control of how they are built to some parent widget in the tree.

Stateless Widget



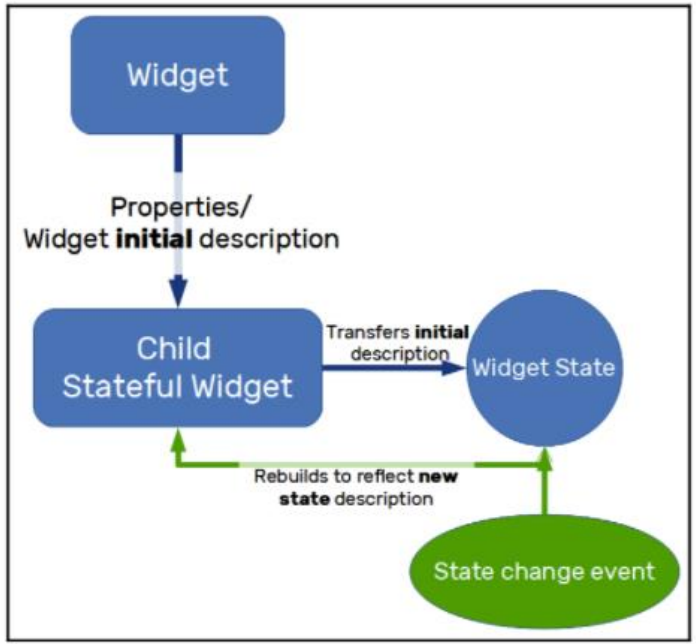
- The child widget will receive its description from the parent widget and will not change it by itself.
- Stateless widgets have only final properties defined during construction, and that's the only thing that needs to be built on the device screen.

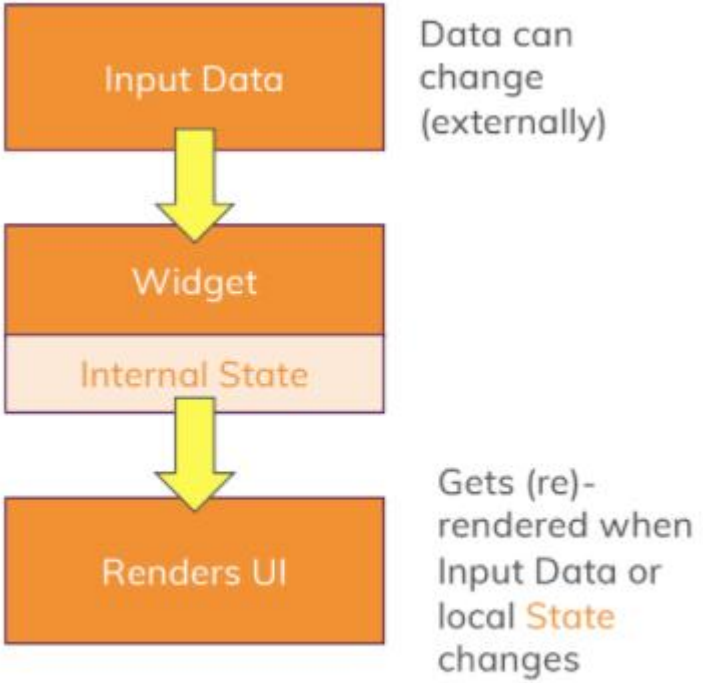
In Code

```
class MyApp extends statelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

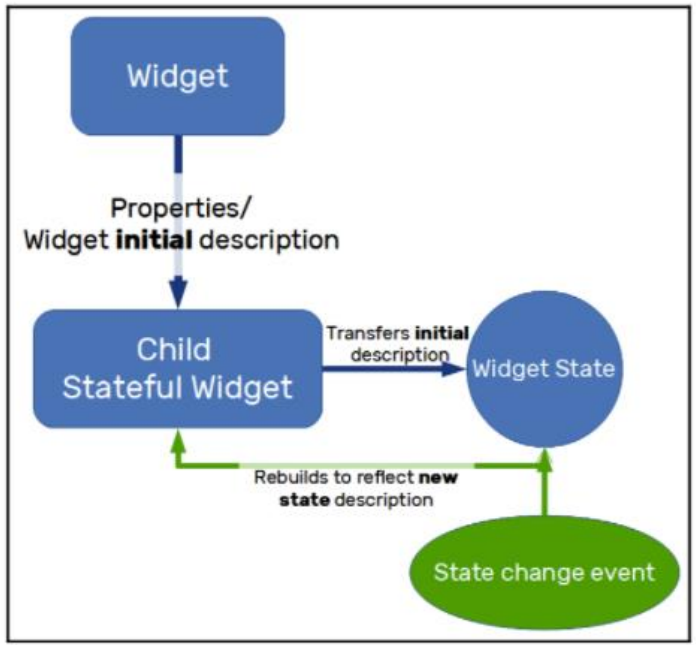
Another example: https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html

Stateful Widget





Stateful Widget



- Change their descriptions dynamically during their lifetimes.
- Immutable, but they have a company State class that represents the current state of the widget.

In Code

```
class MyHomePage extends statefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

By extending statefulWidget, MyHomePage must return a valid State object in its createState() method. In our example, it returns an instance of _MyHomePageState.

Another example: https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html

Flutter Hot Reload

Most Famous Feature

JIT => Just in time reload, you can instantly see the rendering result on the compilator without need to recompile the code

A JIT compilation is where the source code is loaded and compiled to native machine code by the Dart VM on the fly.

https://docs.flutter.dev/development/tools/hot-reload

What is the difference between hot reload, hot restart, and full restart?

- **Hot reload** loads code changes into the VM and re-builds the widget tree, preserving the app state; it doesn't rerun main() or initState(). (⌘\ in Intellij and Android Studio, ^F5 in VSCode)
- **Hot restart** loads code changes into the VM, and restarts the Flutter app, losing the app state. (⇧⌘\ in IntelliJ and Android Studio, ⇧⌘F5 in VSCode)
- **Full restart** restarts the iOS, Android, or web app. This takes longer because it also recompiles the Java / Kotlin / ObjC / Swift code. On the web, it also restarts the Dart Development Compiler. There is no specific keyboard shortcut for this; you need to stop and start the run configuration.

Flutter web currently supports hot restart but not hot reload.

https://docs.flutter.dev/development/tools/hot-reload

References

- https://www.freecodecamp.org/news/an-introduction-to-flutter-the-basics-9fe541fd39e2/
- http://livre21.com/LIVREF/F6/F006145.pdf
- https://www.studytonight.com/android/android-architecture
- https://www.dotnettricks.com/learn/xamarin/understanding-xamarin-ios-build-native-ios-app
- https://flutter.dev/docs/get-started/codelab
- https://flutter.dev/docs/get-started/test-drive?tab=terminal#create-app
- Official Flutter Docs: https://flutter.io/docs/
- Visual Studio Code: https://code.visualstudio.com/
- Visual Studio Code Flutter Extension: https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter
- Android Studio: https://developer.android.com/studio/
- https://arun-ts.blogspot.com/2014/10/
- https://learn.microsoft.com/en-us/archive/blogs/erwinvandervalk/the-difference-between-model-view-viewmodel-and-other-separated-presentation-patterns
- https://learn.microsoft.com/en-us/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps