Flutter

Cross Platform Mobile App Development

What is Flutter

Main Goal: Building User Interfaces

1. Focused on writing reactive views, not the language

Dart programming language

1. AOT (Ahead of time) Compilation allows a static language program (or an intermediate one) to be compiled into native machine language
2. JIT (Just in Time) Allows a dynamic language to be interpreted when it is required (Example: Java > Bytecode > VM)

AOT	**JIT**
1. Slow for development 2. Fast for production	1. Fast for development 2. Slow for production

Dart programming language

1. Dart supports AOT and JIT
   a. This means a fast development
   b. And fast deployment
2. Dart can be compiled into Javascript
3. Dart can be used in server side
   a. Compiling to native code or…
   b. To Javascript and run via NodeJS

Dart programming language

1. Open Source (like Flutter)
2. ECMAScript Standard (ISO 16262)
3. Used inside and outside of Google
    a. Flutter, Fuchsia, Adwords, etc
    b. Libraries for Firebase, Redux, RxDart, internationalization, encryption, databases, routing, collections, and more (https://pub.dev/)

Flutter / Dart main features

1. Extremely fast hot reload
    a. App state is retained across reloads whenever possible, so the app can continue from where it left off
2. Preemptive scheduling, time slicing, and shared resources
    a. Dart is single threaded
    b. Threads are called Isolates
        i. Do not share memory
        ii. No lock required
        iii. Messaging passing through channels
        iv. Similar to Actors in Erlang or Web workers in Javascript
3. Allocation and Garbage collection
    a. Allocating many short lived objects
    b. Single pointer for allocated objects

Flutter / Dart main features

4. Unified Layout: No XML - Main language required

```
Center(child:
  Column(children: [
    Text('Hello, World!'),
    Icon(Icons.star, color: Colors.green),
  ])
)
```

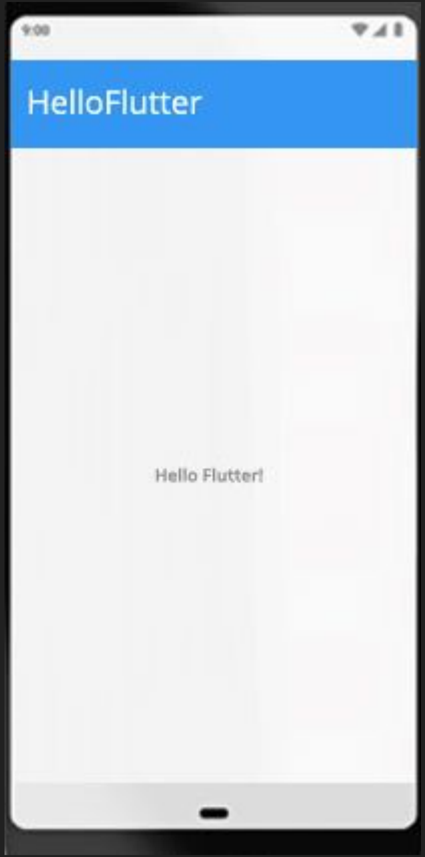
Why Flutter uses Dart?

1. Is AOT (Ahead Of Time) compiled to fast, predictable, native code
2. Can also be JIT (Just In Time) compiled for exceptionally fast development cycles and game-changing workflow
3. Dart makes it easier to create smooth animations and transitions that run at 60fps
4. Dart avoids preemptive scheduling and shared memory (and thus locks)
5. Dart allows Flutter to avoid the need for a separate declarative layout language like JSX or XML
6. Developers have found that Dart is particularly easy to learn because it has features that are familiar to users of both static and dynamic languages.

Understanding Flutter

1. Everything is a Widget
   a. They are Immutables
   b. You can only change its **state**

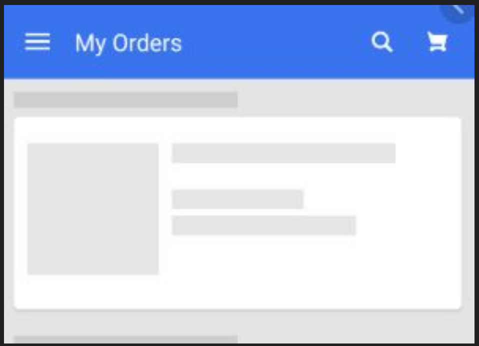
2. Flutter includes **MaterialComponents** and **Apple HIG**
   a. **List**
   b. **Text**
   c. **Layouts**



Understanding Flutter

StatelessWidget

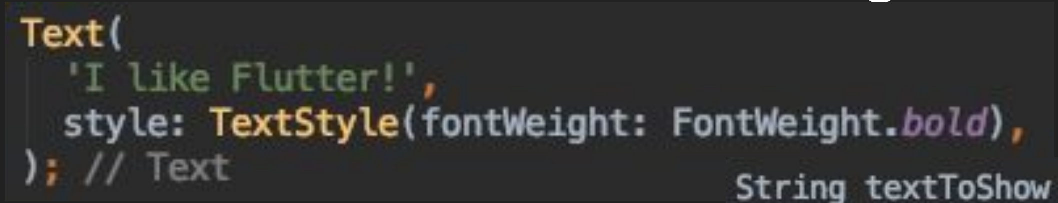
1. No information available on its state
2. Example: ImageView



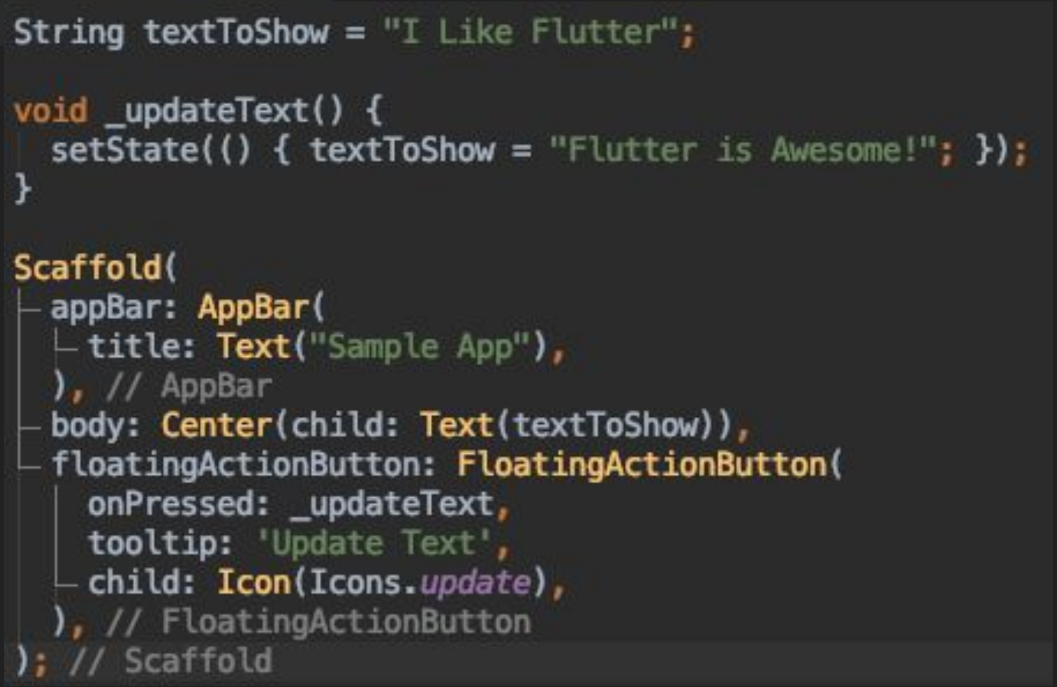
StatefulWidget

Changing its **state** is enough to update the widget

Stateless and Stateful widgets



```dart
Text(
  'I like Flutter!',
  style: TextStyle(fontWeight: FontWeight.bold),
); // Text
```

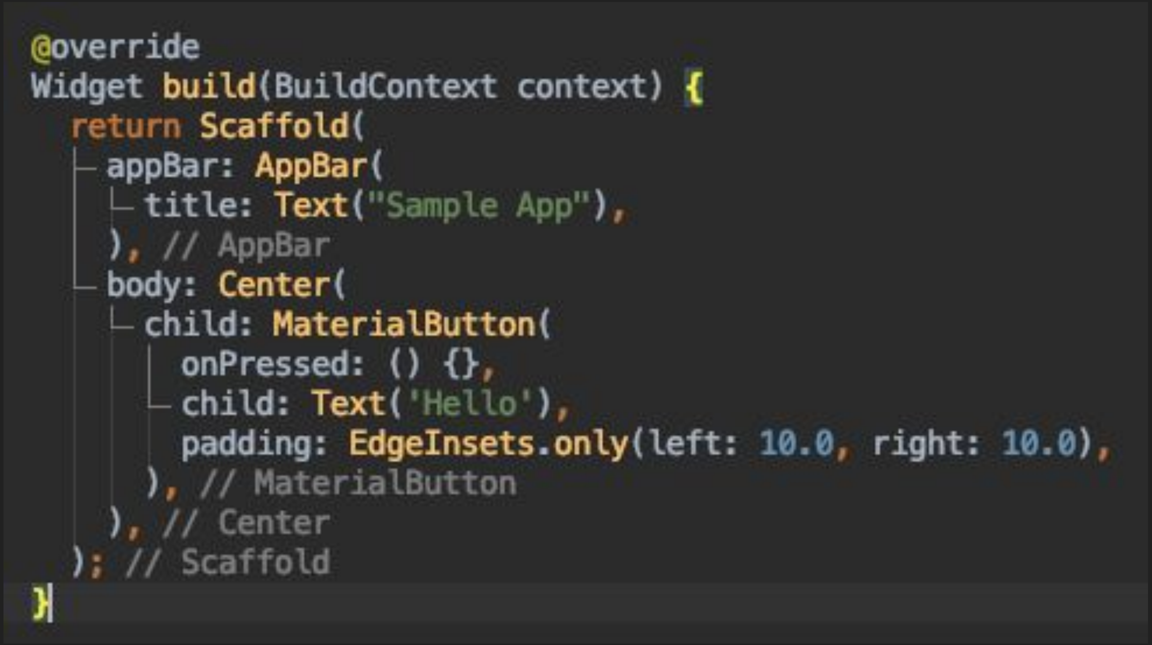


```dart
String textToShow = "I Like Flutter";

void _updateText() {
  setState(() { textToShow = "Flutter is Awesome!"; });
}

Scaffold(
  appBar: AppBar(
    title: Text("Sample App"),
  ), // AppBar
  body: Center(child: Text(textToShow)),
  floatingActionButton: FloatingActionButton(
    onPressed: _updateText,
    tooltip: 'Update Text',
    child: Icon(Icons.update),
  ), // FloatingActionButton
); // Scaffold
```

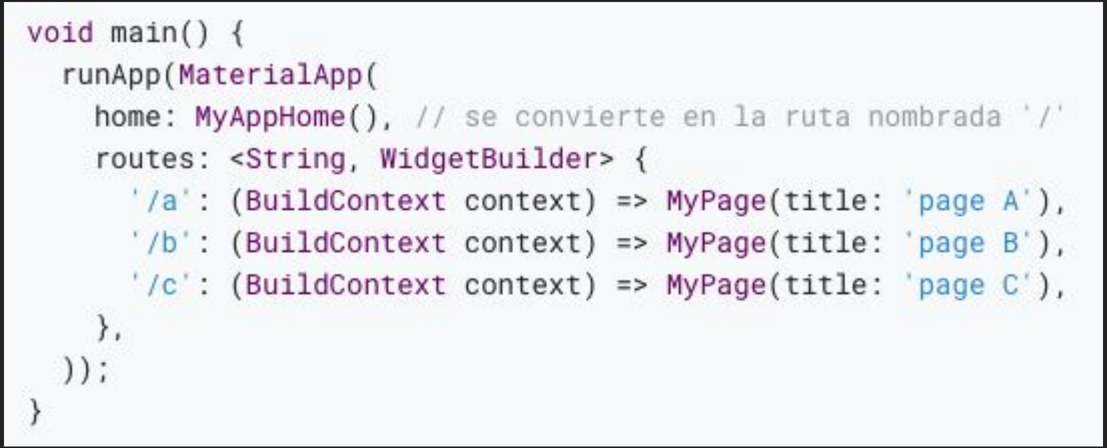
Create customized widgets



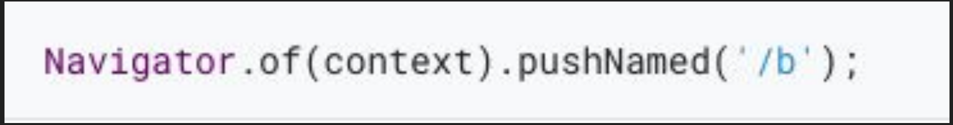
```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Sample App"),
    ), // AppBar
    body: Center(
      child: MaterialButton(
        onPressed: () {},
        child: Text('Hello'),
        padding: EdgeInsets.only(left: 10.0, right: 10.0),
      ), // MaterialButton
    ), // Center
  ); // Scaffold
}
```

“Screens” and “Intents”

1. Everything in Flutter happens in an unique Activity
    a. Each “Screen” is called a Route
    b. A Navigator handles the routes (do push() and pop() of Routes)



```
void main() {
  runApp(MaterialApp(
    home: MyAppHome(), // se convierte en la ruta nombrada '/'
    routes: <String, WidgetBuilder> {
      '/a': (BuildContext context) => MyPage(title: 'page A'),
      '/b': (BuildContext context) => MyPage(title: 'page B'),
      '/c': (BuildContext context) => MyPage(title: 'page C'),
    },
  ));
}
```



```
Navigator.of(context).pushNamed('/b');
```

Conclusions

1. Very fast way to create prototypes or materialize ideas in Android / iOS / Web without much effort
2. For full animations based applications, it seems one of the best choices (60fps)

> Running at 60 fps, user interfaces created with Flutter perform far better than those created with other cross-platform development frameworks.

> I can't stress enough how this is light years ahead of Android's Instant Run, or any similar solution. It just works, even on large non-trivial apps. And it is crazy fast. That's the power of Dart for you.
> In practice, that makes a visual editor interface redundant. I did not miss XCode's rather nice auto layout **at all**.

> Much easier to hire. We now look to take the best candidate regardless if they are from Web, iOS or Android.
> We have 3x the bandwidth now that all of our teams are consolidated on a single code base.
> Knowledge sharing is at an all time high

FIN

Gracias