ReDI School of Digital Integration

Mobile Development with Flutter

Introduction to Flutter and what makes it a valid option to develop mobile apps in 2020.

Luca

Salsabil

Your hosts tonight

Agenda

ReDI School of Digital Integration

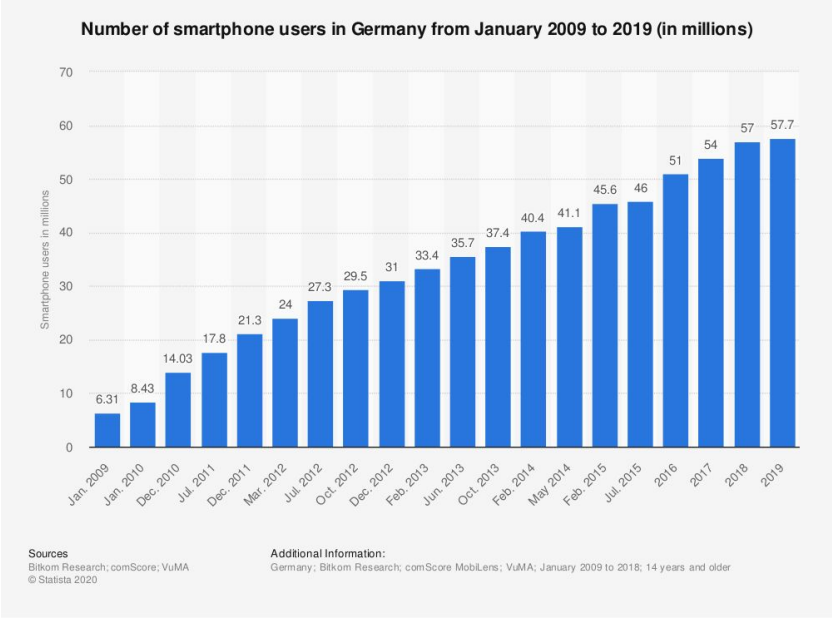
- Intro to mobile development
- Native vs cross-platform
- Introduction to Flutter
- Quiz & Demo
- Dart in a nutshell
- Quiz & Demo
- Closing remarks
- Q&A
- Goodbye

ReDI School of Digital Integration

Smartphones Facts

ReDI School of Digital Integration

Number of mobile app downloads worldwide in *2019* **204bn**
Number of mobile app downloads worldwide in *2016* **140.7bn**

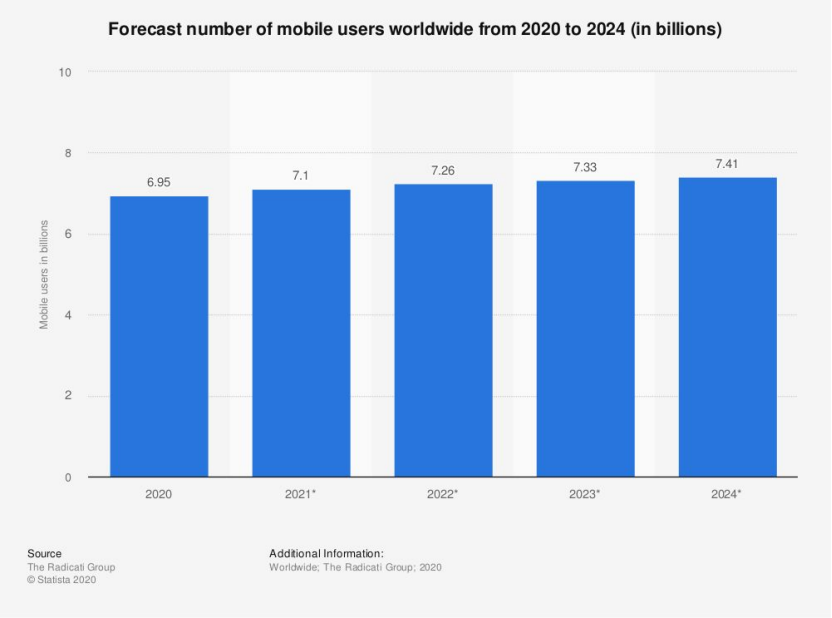


Why should we care about mobile development?

Source: statista.com

ReDI School of Digital Integration

Number of mobile app downloads worldwide in *2019* **204bn**
Number of mobile app downloads worldwide in *2016* **140.7bn**



Why should we care about mobile development?

Source: statista.com

Businesses need apps, good ones.

ReDI School of Digital Integration

What do we have at our disposal?

Native - Cross Platform

A *native app* is built using the programming language and tools for the specific device platform.

A *cross-platform app* is built with non-native tools and it has historically made use of web technologies to render WebViews

ReDI School of Digital Integration

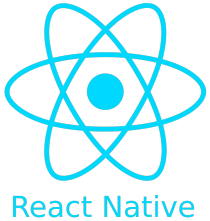
Native - Hybrid - Cross Platform

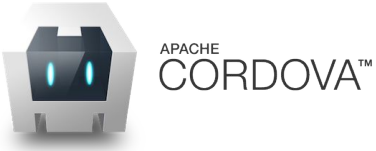
Kotlin
Java

Swift
Objective-C

ReDI School of Digital Integration

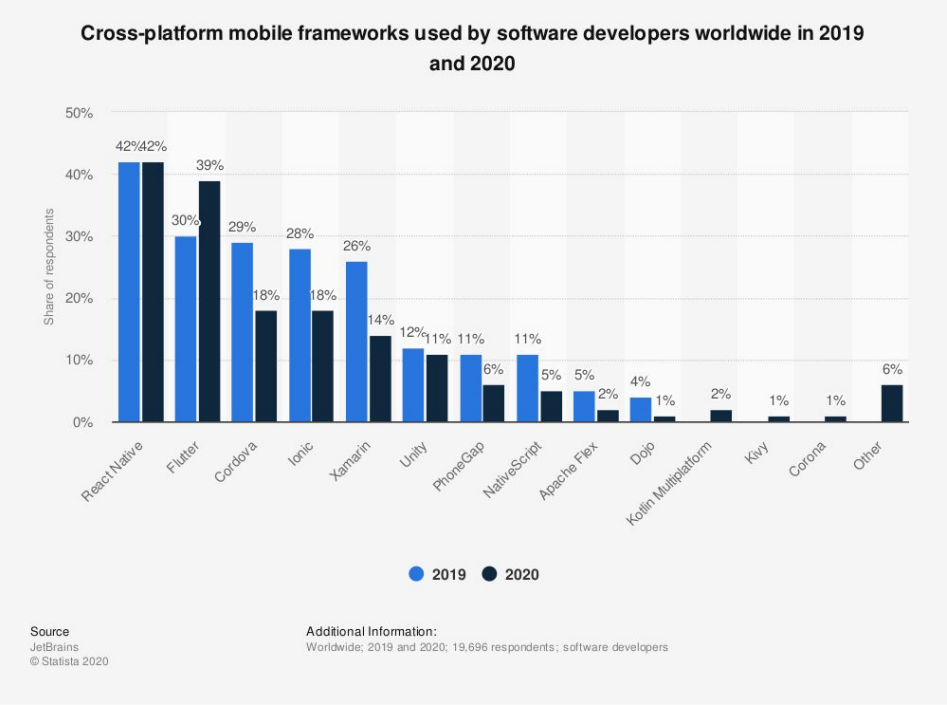
Native - Hybrid - Cross Platform





ReDI School of Digital Integration

Cross platform frameworks - distribution



Flutter is the definitely promising!

ReDI School of Digital Integration

What is Flutter?

Flutter - intro

- Free and Open Source UI toolkit
- Provides a SDK to develop apps for Android and iOS
- Good for developers, designers

Flutter - something more

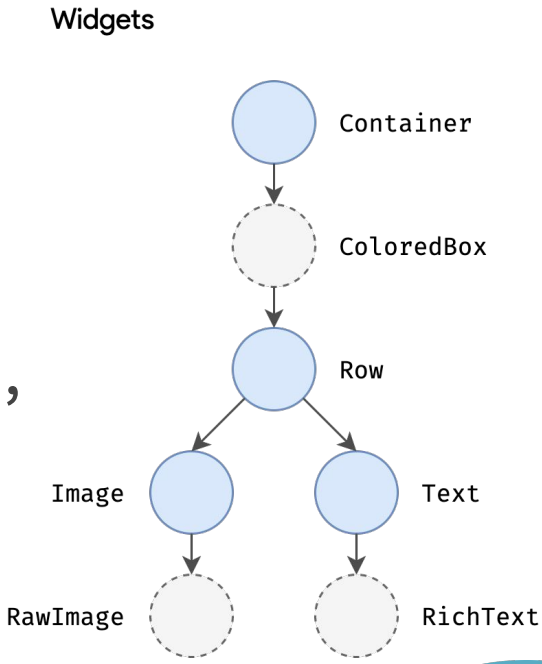
- Full control over the rendering stack
- Reactive views with no bridge
- Great development experience
- Fast, smooth and predictable UI
- Deploy on multiple platforms with one codebase
- Great community

ReDI School of Digital Integration

Flutter - widgets

- In Flutter, everything is a widget!
  - UI elements (Text, Image, etc.)
  - Layout constraints (Padding, Column, Alignment, etc.)

- You can compose widgets to create your UI
  - Flutter offers a rich set of default widgets (here)



Source: flutter.doc

Demo!

ReDI School of Digital Integration

Dart in a nutshell

Dart - intro

Dart

- Flutter code is written in Dart
- Dart is a client-optimized language for fast apps on any platform
- Can be both JIT compiled and AOT compiled
- Type safe: it uses a combination of static type checking and runtime checks

Source: flutter.doc

ReDI School of Digital Integration

Dart - a bit more

Typing system (primitive types, var vs dynamic)

```
var name = 'ReDI';
dynamic name = 'ReDI';
String name = 'ReDI';
final String nickname = 'ReDI';
```

Source: flutter.doc

ReDI School of Digital Integration

Dart - types

The Dart language has also support for the following types:

Lists

```
var list = [1, 2, 3];
list.add(5);
```

Maps

```
var gifts = {  'first': 'Backend',
               'second': 'Data Science',
               'fifth': 'Cloud Computing' };
```

ReDI School of Digital Integration

Control flow statements

```
if (year >= 2001) {
  print('21st century');                    // if else condition
} else if (year >= 1901) {
  print('20th century');
}

for (var object in flybyObjects) {          // foreach loop
  print(object);
}

for (int month = 1; month <= 12; month++) {   // for loop
  print(month);
}
                                            // while loop
while (year < 2016) {
  year += 1;
}
```

Functions

Define a simple addition function:

```
int sumUp(int a, int b, int c) {
  return a + b + c;
}
// ...
  int total = sumUp(1, 2, 3);
```

ReDI School of Digital Integration

Class

```
class MyClass {
  int property = 0;

  int get property => property;

  set property(int value) {
    if (value >= 0) {
      property = value;
    }
  }
}
```

Awaits / Async / Future

Synchronous operation:

A synchronous operation blocks other operations from executing until it completes.

Asynchronous operation:

Once initiated, an asynchronous operation allows other operations to execute before it completes.

ReDI School of Digital Integration

Awaits / Async / Future

The async and await keywords provide a declarative way to define asynchronous functions and use their results.

A future represents the result of an asynchronous operation, and can have three states:

- Uncompleted,
- Completed with a value,
- Completed with an error.

ReDI School of Digital Integration

Synchronous vs Asynchronous

```
String createOrderMessage() {
  var order = fetchUserOrder();
  return 'Your order is: $order';
}

Future<String> fetchUserOrder() =>
    // Imagine that this function is
    // more complex and slow.
    Future.delayed(
      Duration(seconds: 2),
      () => 'Large Latte',
    );

void main() {
  print('Fetching user order...');
  print(createOrderMessage());
}

//Fetching user order...
//Your order is: Instance of _Future<String>
```

```
Future<String> createOrderMessage() async {
  var order = await fetchUserOrder();
  return 'Your order is: $order';
}

Future<String> fetchUserOrder() =>
    // Imagine that this function is
    // more complex and slow.
    Future.delayed(
      Duration(seconds: 2),
      () => 'Large Latte',
    );

Future<void> main() async {
  print('Fetching user order...');
  print(await createOrderMessage());
}
//Fetching user order...
//Your order is: Large Latte
```

Demo!

Who's using Flutter?

...And many more!

You can see more here.

Source: flutter.doc

Questions?

ReDI School of Digital Integration

THANK YOU!