Mobile Applications

9. Introduction to cross-platform apps development

Boni García

boni.garcia@uc3m.es

Telematic Engineering Department
School of Engineering

2025/2026

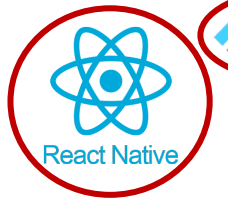
uc3m | Universidad Carlos III de Madrid

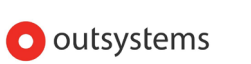
Table of contents

1. Introduction

- As we have seen, there are different ways to develop mobile apps:

1. Native: Platform-specific apps (Android or iOS)
2. Hybrid: Web app wrapped in a native container
3. Cross-platform: Single codebase compiled to native apps for multiple platforms
4. Progressive web apps (PWAs): Web apps installable on mobile devices with offline support
5. Low-code/no-code: Visual, drag-and-drop platforms for rapid app creation



This unit we study the basics of **React Native** and **Flutter**

Table of contents

2. React

- React is an open source front-end **JavaScript library** for building user interfaces (UIs) based on components created by Meta (formerly Facebook)
  - It is typically used to build Single-Page Application (**SPA**)
  - SPAs are web applications that interact with the user by dynamically rewriting the current web page with new data from the web server
    - Some popular SPAs are the web versions of Instagram, Facebook, Netflix, or Airbnb

React

The library for web and native user interfaces

https://react.dev/

2. React

- The key features of React are the following:
  - Component-based
    - React apps are built using reusable components (like Lego blocks)
    - Each component manages its own state and logic
  - JSX (JavaScript XML)
    - JavaScript extension that lets developers write HTML-like markup in JavaScript code
    - JSX allows us to write HTML elements in JavaScript and place them in the DOM (Document Object Model), i.e., the tree of objects in web pages
    - It is not mandatory, but recommended for easier development
  - Rich ecosystem
    - Works well with libraries and frameworks like Redux (state management), React Router (navigation), or Next.js (server-side rendering)

2. React

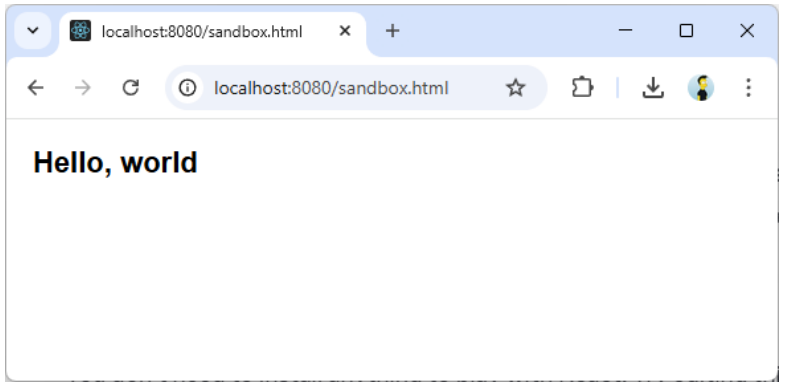
- There are different ways to create React apps:

1. Using a sandbox
   - A sandbox is an isolated, browser-based environment where we can write, run, and test React code without setting up a local project
     - It is a convenient way to quickly prototype without setting anything up locally
     - A popular sandbox is CodeSandbox
2. Setting up a local project
   - Using a build tool (convenient for basic projects)
     - The traditional tool is called Create React App, but it is deprecated nowadays
     - Modern options are: Vite, Parcel, or Rsbuild
   - Using a framework (convenient for complex projects)
     - Provide enhanced features (e.g., SEO, routing, optimizations, and others)
     - For example: Next.js, Remix, or Gatsby

2. React - Sandbox

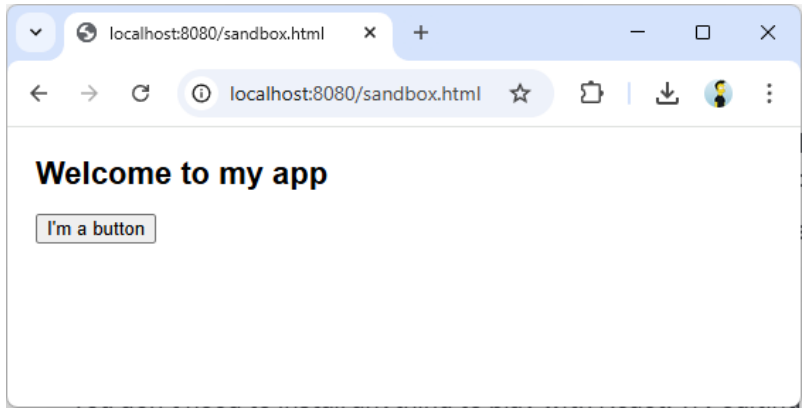
- There are basic sandboxes examples in the React doc using JSX, e.g.:

```
function Greeting({ name }) {
  return <h1>Hello, {name}</h1>;
}

let App = function App() {
  return <Greeting name="world" />
}
```




```
function MyButton() {
  return (
    <button>
      I'm a button
    </button>
  );
}

let App = function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton />
    </div>
  );
}
```

React **components** are JavaScript functions that return markup (Greeting and MyButton in these examples)



https://react.dev/learn

2. React - Local setup

- To set up a local React project, the requirements are:

1. Node.js
   - Node.js is an open source, cross-platform JavaScript runtime environment that enables the execution of JavaScript code outside a web browser
   - Node.js comes with NPM, its default package manager

https://nodejs.org/

```
> node --version
v24.14.0

> npm --version
11.1.1
```

After install Node.js, we can execute node and npm as command-line tools

2. Code editor (not mandatory, but highly recommended)
   - One of the most popular IDEs for React is Visual Studio Code
   - Other alternatives are: WebStorm, Sublime Text, or Atom

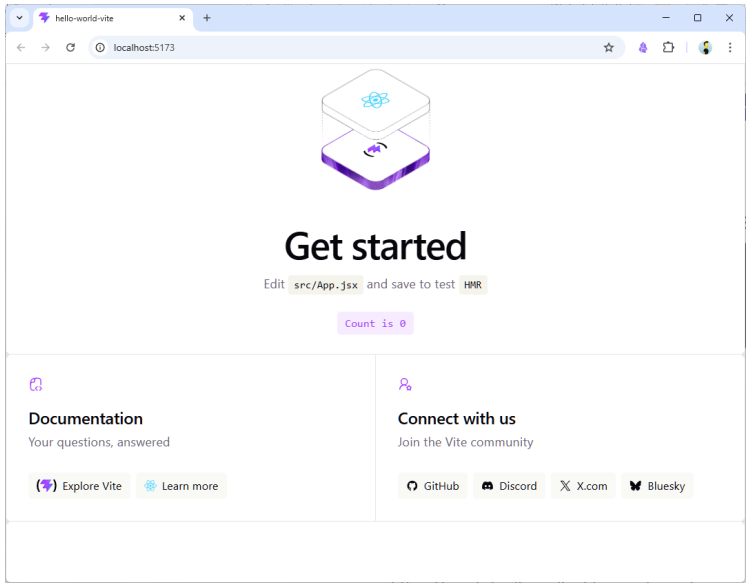
Visual Studio Code

https://code.visualstudio.com/

2. React - Local setup

- For instance, we can use Vite to set up a local React project:

```
> npm create vite@latest hello-world-vite -- --template react

 VITE v8.0.10  ready in 284 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```



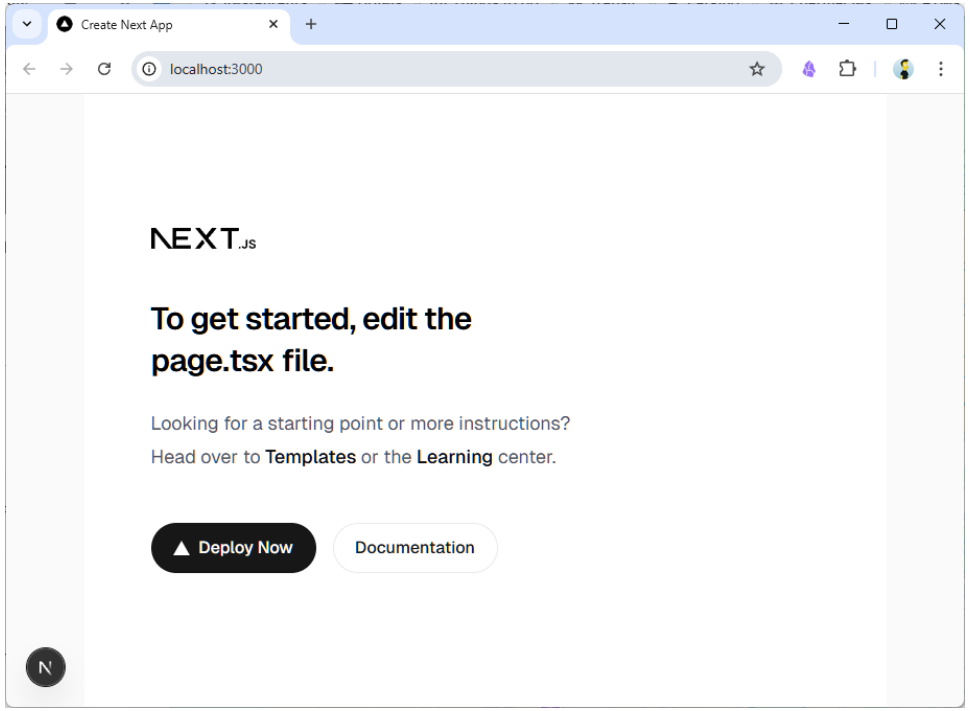
https://vite.dev/

https://react.dev/learn/build-a-react-app-from-scratch

2. React - Frameworks

- React frameworks support all the features required to deploy and scale an app in production
  - For example, Next.js:

```
> npx create-next-app@latest

√ What is your project named? ... hello-world-next
√ Would you like to use the recommended Next.js defaults? » Yes, use
recommended defaults

...

Success! Created my-app at C:\Users\boni\Documents\dev\mobile-apps\hello-
world-next

> cd hello-world-next

> npm run dev
```



npx is a CLI tool that comes with npm used to execute Node.js packages without installing them globally or locally

NEXT.JS
https://nextjs.org/

2. React - TypeScript

- **JavaScript** (JS) is a high-level, just-in-time compiled programming language
  - JavaScript first appeared in Netscape Navigator in 1995 and became widely available with Netscape Navigator 2 in 1996.
  - It is most well-known as the scripting language for web pages, being one of the core technologies of the Web, alongside HTML and CSS
  - It has **dynamic typing**, i.e., the type of a variable is known at runtime
- **TypeScript** (TS) is an open source programming language developed by Microsoft, first released in 2012
  - TypeScript is often referred to as a superset or extension of JavaScript
  - TypeScript extends JavaScript by adding **static typing**, i.e., the type of a variable is known at compile-time. This feature help developers build large-scale, robust applications more effectively
  - Unlike JavaScript, TypeScript can't be executed directly in browsers, it must be *transpiled* into JavaScript first

2. React - TypeScript

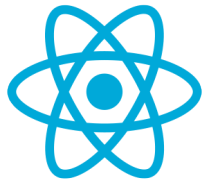
- The following table summarizes the advantages and disadvantages of using JavaScript and TypeScript in React development:

	JavaScript	TypeScript
Pros	• Faster development time: JavaScript is a dynamic and flexible language that allows developers to build applications quickly • Easier to learn: Gentler learning curve	• Code quality: TypeScript adds static typing to JavaScript, which helps catch bugs at compile-time • Developer experience: Features such as auto-completion and error checking
Cons	• No static typing: JavaScript is a dynamically typed, which can lead to bugs at runtime • Code can become complex: Without the benefit of static typing, it can be challenging to maintain code quality and readability as applications grow	• Higher learning curve: TypeScript introduces additional syntax and concepts • Slower initial development: Static typing can add overhead, especially for small projects.

Table of contents

3. React Native

- **React Native** is an open source cross-platform app framework for building mobile apps (Android and iOS)
  - Internally it uses the React library to manage the components. For that reason, we can use JavaScript or TypeScript to develop React Native apps
  - Like React, React Native is maintained by Meta (formerly Facebook) and a community of individual developers and companies



https://reactnative.dev/

3. React Native - Expo

- The recommended way to set up a local React Native project is using a framework like Expo
- **Expo** is a framework and platform for building and deploying React Native apps
- **Expo Go** is an open source mobile app for testing React Native apps on any Android or iOS device
  - It is available on both the Android Play Store and iOS App Store
    - Android Play Store (Android 12+)
    - iOS App Store (iOS 13+)

Expo

https://expo.dev/

https://expo.dev/client

3. React Native - Hello world

- We can create a basic React Native app with Expo using npx as follows:

```
> npx create-expo-app hello-world-react-native

✓ Downloaded and extracted project files.
✓ Installed JavaScript dependencies.

☑ Your project is ready!

To run your project, navigate to the directory and run one of the following npm
commands.

- cd hello-world-react-native
- npm run android
- npm run ios # you need to use macOS to build the iOS project - use the Expo app if
you need to do iOS development without a Mac
- npm run web
```

3. React Native - Hello world

```
> npm run android

Starting project at C:\Users\boni\Documents\dev\mobile-apps\hello-world-react-native
Starting Metro Bundler
Tunnel connected.
Tunnel ready.
› Opening exp://87vuv4m-anonymous-8081.exp.direct on Pixel_4
```

We can use this QR code to load the app in Expo Go. If it does not load, it is usually caused by a connectivity issue. In that case, we can try:

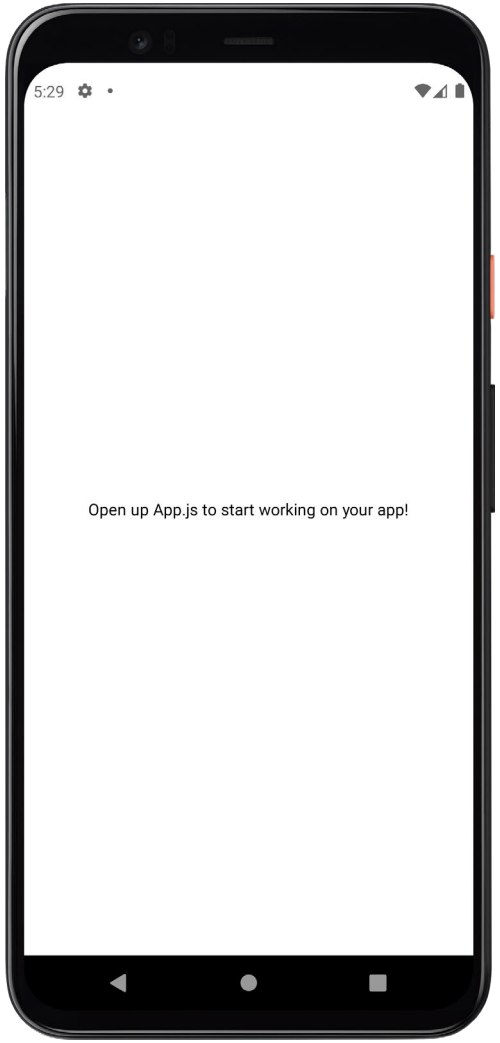
```
> npx expo start --android --tunnel
```

```
› Metro waiting on exp://87vuv4m-anonymous-8081.exp.direct
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Using Expo Go
› Press s │ switch to development build

› Press a │ open Android
› Press w │ open web

› Press j │ open debugger
› Press r │ reload app
› Press m │ toggle menu
› shift+m │ more tools
› Press o │ open project code in your editor

› Press ? │ show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
Android Bundled 508ms index.js (618 modules)
```

To run our app locally, we need a running AVD (e.g., executed with Android Studio)

3. React Native - Hello world


```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Open up App.js to start working on your app!</Text>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```



3. React Native - Core Components

- A **view** is the basic building block of the UI in both Android and iOS
  - It represents a rectangular area of the screen used to display text, images, inputs, etc.
  - With React Native, we describe views using JavaScript/TypeScript and React components
- At runtime, React Native renders the corresponding native Android and iOS views
- React Native provides ready-to-use views (called **core components**), e.g.:

React Native	Android	iOS	Web analog	Purpose
`<View>`	`ViewGroup`	`UIView`	`<div>`	Layout container
`<Text>`	`TextView`	`UITextView`	`<p>`	Text display
`<Image>`	`ImageView`	`UIImageView`	`<img>`	Image display
`<ScrollView>`	`ScrollView`	`UIScrollView`	`<div>`	Scrollable container
`<TextInput>`	`EditText`	`UITextField`	`<input>`	Text input

https://reactnative.dev/docs/intro-react-native-components

3. React Native - Core Components


```
import React from 'react';
import { StyleSheet, View, Text, Image, TextInput } from 'react-native';

const App = () => {
  return (
  <View style={styles.container}>
      <Text>Some text</Text>
      <Image source={require('./assets/react.png')} style={{width: 200, height: 200}} />
      <TextInput
        style={{
          width: 200,
          height: 40,
          borderColor: 'gray',
          borderWidth: 1,
        }}
        defaultValue="Type in me"
      />
    </View>
  );
};

export default App;

const styles = StyleSheet.create({
    container :{
       justifyContent: 'center', //Centered horizontally
       alignItems: 'center', //Centered vertically
       flex:1
    }
});
```

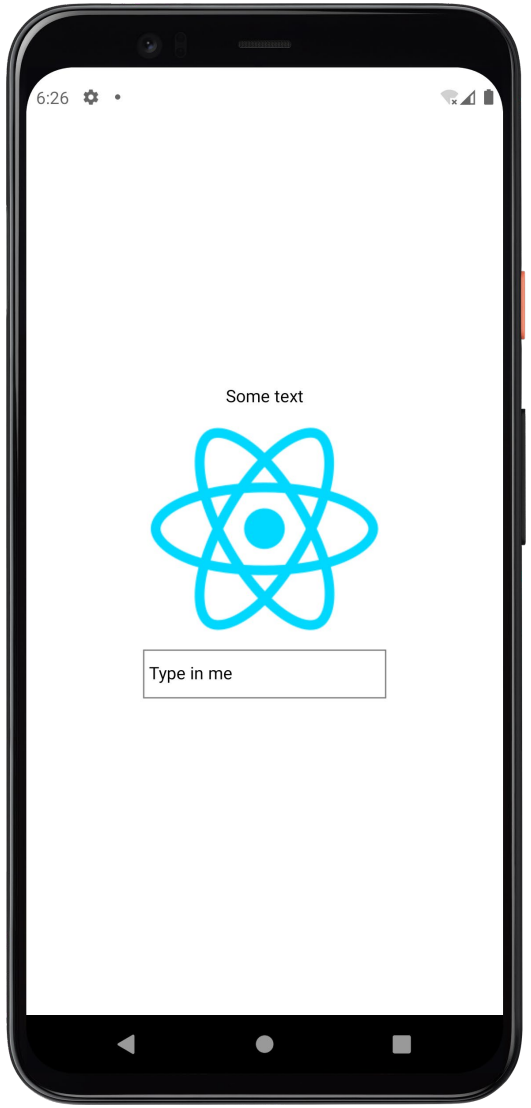



Table of contents

4. Flutter

- **Flutter** is an open source cross-platform framework created by Google used to develop applications for mobile (Android, iOS), web, and desktop from a single codebase
- Flutter applications are written in **Dart** language
  - Dart programming language developed by Google since 2011
  - Dart aims to help developers build UIs effectively
  - Dart is open source, object-oriented, and statically typed

https://flutter.dev/

https://dart.dev/

4. Flutter - Setup

- To install Flutter and Dart in our machine, we use the following tutorial:
  - https://docs.flutter.dev/get-started/install
- Once it is installed, we can execute the command-line tools `flutter` and `dart`:

```
> flutter --version
Flutter 3.41.8 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 02085feb3f (4 days ago) • 2026-04-24 13:54:45 -0700
Engine • hash 7a53c052bc4b472cf780b199087e1368e4a9aa8c (revision 59aa584fdf) (12 days ago)
• 2026-04-16 02:32:16.000Z
Tools • Dart 3.11.5 • DevTools 2.54.2

> dart --version
Dart SDK version: 3.11.5 (stable) (Wed Apr 15 00:36:32 2026 -0700) on "windows_x64"
```

- We can use the following command to verify our installation:

```
> flutter doctor -v
```

4. Flutter - Dart

- The key features of Dart are the following:
  - Compiled language
    - Can be compiled to native machine code (for mobile/desktop apps) or JavaScript (for web apps)
    - Uses Just-In-Time (JIT) compilation during development for hot reload (apply changes without restarting)
    - Uses Ahead-Of-Time (AOT) compilation for production for optimized performance
  - Object-oriented and supports for functional programming
    - Classes, inheritance, interfaces, mixins
    - Supports lambdas (anonymous functions), higher-order functions, and collections (like map, filter, reduce)
  - Static typing (types are checked at compile time)
  - Null safety (to prevent null reference exceptions)
  - Asynchronous programming with async/await
    - Built-in support for futures and streams for handling async operations

4. Flutter - Dart

- Hello world:

```dart
void main() {
  print('Hello, Dart World!');
}
```

```
> dart hello_world.dart
Hello, Dart World!
```

- Variables and data types:

```dart
void main() {
  // Variables (type inference with 'var')
  var name = "Alice"; // String

  int age = 25; // Explicit type
  double height = 5.9;
  bool isStudent = true;

  // Dynamic type (can change at runtime)
  dynamic dynamicVar = "Hello";
  dynamicVar = 42; // Now an int

  // Constants (compile-time)
  const PI = 3.14;
  final currentTime = DateTime.now(); // Runtime constant

  print("$name is $age years old"); // String interpolation
}
```

```
> dart variables_data_types.dart
Alice is 25 years old
```

https://dart.dev/language

4. Flutter - Dart

- Control flow and loops:

```dart
void main() {
  int score = 85;

  // If-else
  if (score >= 90) {
    print("A");
  } else if (score >= 80) {
    print("B"); // Output: B
  } else {
    print("C");
  }

  // For loop
  for (var i = 0; i < 3; i++) {
    print(i); // 0, 1, 2
  }

  // While loop
  int count = 0;
  while (count < 2) {
    print("Count: $count"); // Count: 0, Count: 1
    count++;
  }

  // Switch-case
  String grade = "B";
  switch (grade) {
    case "A":
      print("Excellent!");
      break;
    case "B":
      print("Good!"); // Output: Good!
      break;
    default:
      print("Unknown");
  }
}
```

```
> dart control_flow.dart
B
0
1
2
Count: 0
Count: 1
Good!
```

4. Flutter - Dart

- Functions:

```dart
// Basic function
void greet(String name) {
  print("Hello, $name!");
}

// Optional positional parameters
void sayHello(String name, [String? title]) {
  print("Hello, ${title ?? ''} $name");
}

// Named parameters (with defaults)
void describe({String name = "User", int age = 0}) {
  print("$name is $age years old");
}

// Main function
void main() {
  greet("Alice"); // Hello, Alice!

  describe(name: "Bob", age: 30); // Bob is 30 years old.

  sayHello("Alice"); // Hello, Alice
  sayHello("Alice", "Dr."); // Hello, Dr. Alice

  // Arrow function (short syntax)
  int add(int a, int b) => a + b;
  print(add(2, 3)); // 5
}
```

```
> dart functions.dart
Hello, Alice!
Bob is 30 years old.
Hello,  Alice
Hello, Dr. Alice
5
```

4. Flutter - Dart

- Collections (list, sets, maps):

```dart
void main() {
  // List (ordered, mutable)
  List<String> fruits = ["Apple", "Banana"];
  fruits.add("Cherry");
  print(fruits[1]); // Banana

  // Set (unique items)
  Set<int> numbers = {1, 2, 2, 3}; // {1, 2, 3}
  print(numbers.contains(2)); // true

  // Map (key-value pairs)
  Map<String, int> ages = {
    "Alice": 25,
    "Bob": 30,
  };
  print(ages["Bob"]); // 30
}
```

```
> dart collections.dart
Banana
true
30
```

4. Flutter - Dart

- Classes and objects:

```dart
// Class with constructor
class Person {
  String? name;
  int age;

  // Constructor (short syntax)
  Person(this.name, this.age);

  // Named constructor (multiple constructor with
  // descriptive name for different initialization scenarios)
  Person.guest() : name = "Guest", age = 0;

  // Method
  void introduce() {
    print("I'm $name, $age years old");
  }
}

void main() {
  var alice = Person("Alice", 25);
  alice.introduce();

  var guest = Person.guest();
  guest.introduce();
}
```

```
> dart oop.dart
I'm Alice, 25 years old
I'm Guest, 0 years old
```

4. Flutter - Dart

- Mixins:

```dart
// Mixins are a way of defining code that can
// be reused in multiple class hierarchies
mixin Swimming {
  void swim() => print("Swimming!");
}

mixin Flying {
  void fly() => print("Flying!");
}

// Apply mixins to a class
class Duck with Swimming, Flying {
  void quack() => print("Quack!");
}

void main() {
  var duck = Duck();
  duck.swim();  // Output: "Swimming!"
  duck.fly();   // Output: "Flying!"
  duck.quack(); // Output: "Quack!"
}
```

```
> dart mixins.dart
Swimming!
Flying!
Quack!
```

4. Flutter - Dart

- Async programming:

```dart
// Future represents a value (String) that will be available later
// async: Marks a function as asynchronous
Future<String> fetchUser() async {
  // await: Pauses execution until the Future completes (without blocking other code)
  await Future.delayed(Duration(seconds: 2)); // Simulate network request
  return "Alice";
}

// Using async/await
void getUser() async {
  String user = await fetchUser(); // Waits for fetchUser() to complete
  print("User: $user"); // Prints after 2 seconds
}

void main() {
  getUser(); // Starts the async operation
  print("Loading..."); // Runs immediately (non-blocking)
}
```

```
> dart async.dart
Loading...
User: Alice
```

4. Flutter - Dart

- Error handling:

```dart
void main() {
  try {
    // The operator ~/ divides two numbers and
    // returns the result as an integer while the
    // operator / returns the result as a double
    var result = 100 ~/ 0;
    print(result);
  } catch (e) {
    print("Error: $e");
  } finally {
    print("Done");
  }
}
```

```
> dart error_handling.dart
Error: IntegerDivisionByZeroException
Done
```

4. Flutter - Widgets

- Flutter provides a rich set of core **widgets** to build responsive UIs, such as:
- Basic (UI structure and layout)
  - Text, Container, Row & Column, Stack, Padding, Center, SizedBox, Expanded & Flexible
- Interactive
  - Buttons, TextField, Checkbox, Radio, Switch, Slider, GestureDetector, InkWell, ...
- Platform-specific
  - Material (Android): MaterialApp, Scaffold, AppBar, FloatingActionButton, Card, …
  - Cupertino (iOS): CupertinoApp, CupertinoNavigationBar, CupertinoButton, CupertinoPicker, ...
- Navigation and routing
  - Navigator, PageRoute, BottomNavigationBar, TabBar & TabBarView, ...
- State management
  - StatefulWidget, InheritedWidget, Provider, ValueNotifier, ChangeNotifier
- List and grids
  - ListView, GridView, ListTile, ListView.builder, …

https://docs.flutter.dev/ui/widgets

4. Flutter - Hello World

- To implement a “Hello World” app in Flutter, we can do the following:

1. Create a new Flutter project

```
> flutter create hello_world_flutter
```

2. We can check and edit our app (using Dart)

The official doc recommend to use Visual Studio Code for coding (although other IDEs can be used)

Visual Studio Code

https://code.visualstudio.com/

3. Run the app

```
> cd hello_world_flutter

> flutter run
```

4. Flutter - Hello World

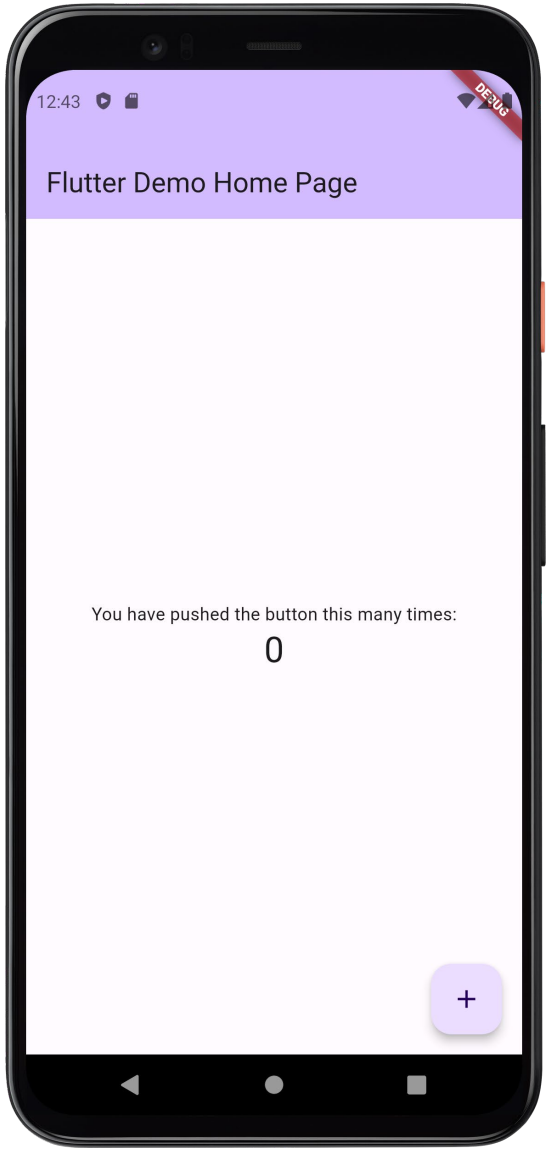
```dart
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}
```

```dart
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

4. Flutter - Hello World

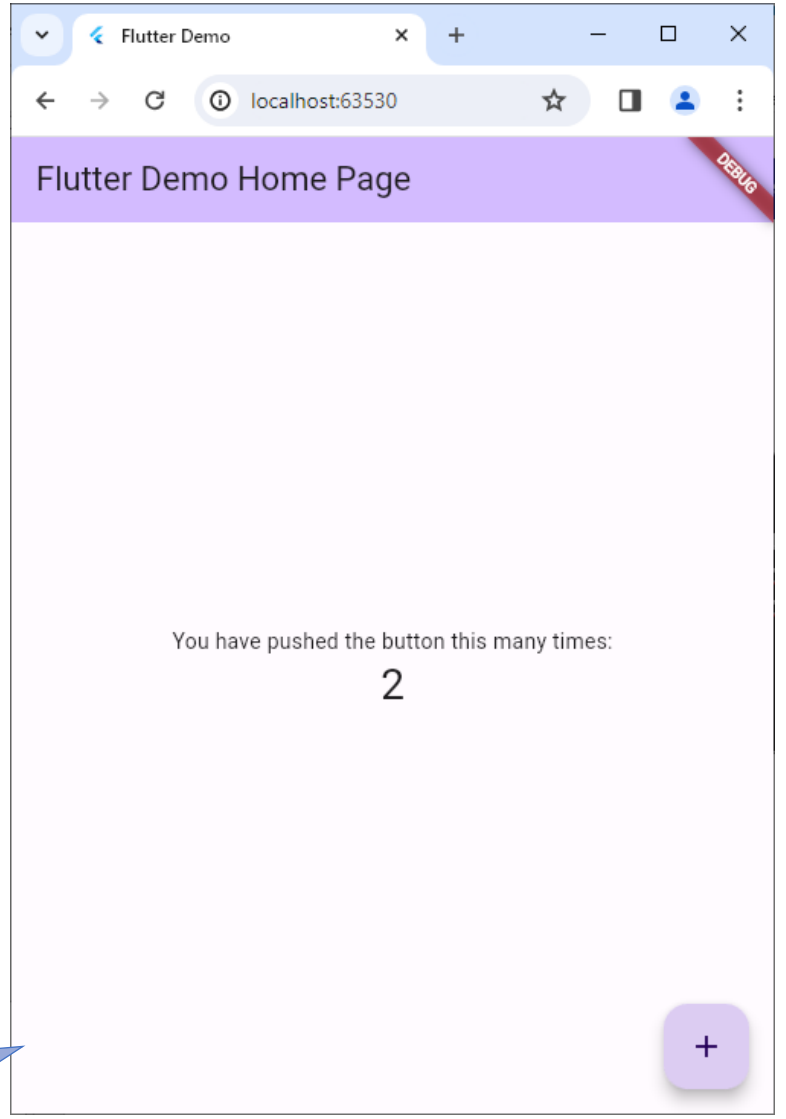
```
> flutter run
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
Running Gradle task 'assembleDebug'...                              5.0s
✓ Built build\app\outputs\flutter-apk\app-debug.apk
Installing build\app\outputs\flutter-apk\app-debug.apk...       1,527ms
D/FlutterJNI( 6924): Beginning load of flutter...
D/FlutterJNI( 6924): flutter (null) was loaded normally!
I/flutter ( 6924):
[IMPORTANT:flutter/shell/platform/android/android_context_gl_impeller.cc(104)] Using the
Impeller rendering backend (OpenGLES).
D/FlutterRenderer( 6924): Width is zero. 0,0
Syncing files to device sdk gphone64 x86 64...                      154ms

Flutter run key commands.
r Hot reload. 🔥🔥🔥
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

A Dart VM Service on sdk gphone64 x86 64 is available at:
http://127.0.0.1:36926/2ZYBWbuqadQ=/
```

If we have a running AVD, the same app will be deployed as an Android APK



4. Flutter - Hello World

```
> flutter run
Connected devices:
Windows (desktop) • windows • windows-x64    • Microsoft Windows [Version 10.0.26200.8246]
Chrome (web)      • chrome  • web-javascript • Google Chrome 147.0.7727.103
Edge (web)        • edge    • web-javascript • Microsoft Edge 147.0.3912.72
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome...              21.6s

Flutter run key commands.
r Hot reload. 🔥🔥🔥
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

Debug service listening on ws://127.0.0.1:30040/wuRhslCPuSo=/ws
A Dart VM Service on Chrome is available at: http://127.0.0.1:30040/wuRhslCPuSo=
The Flutter DevTools debugger and profiler on Chrome is available at:
http://127.0.0.1:30040/wuRhslCPuSo=/devtools/?uri=ws://127.0.0
Starting application from main method in: org-dartlang-app:/we
```



The same codebase can be deployed as a desktop or web app

4. Flutter - Examples

- There are plenty of sample apps maintained by the Flutter team:

https://github.com/flutter/samples

For example, this app contains a comprehensive set of Material 3 components

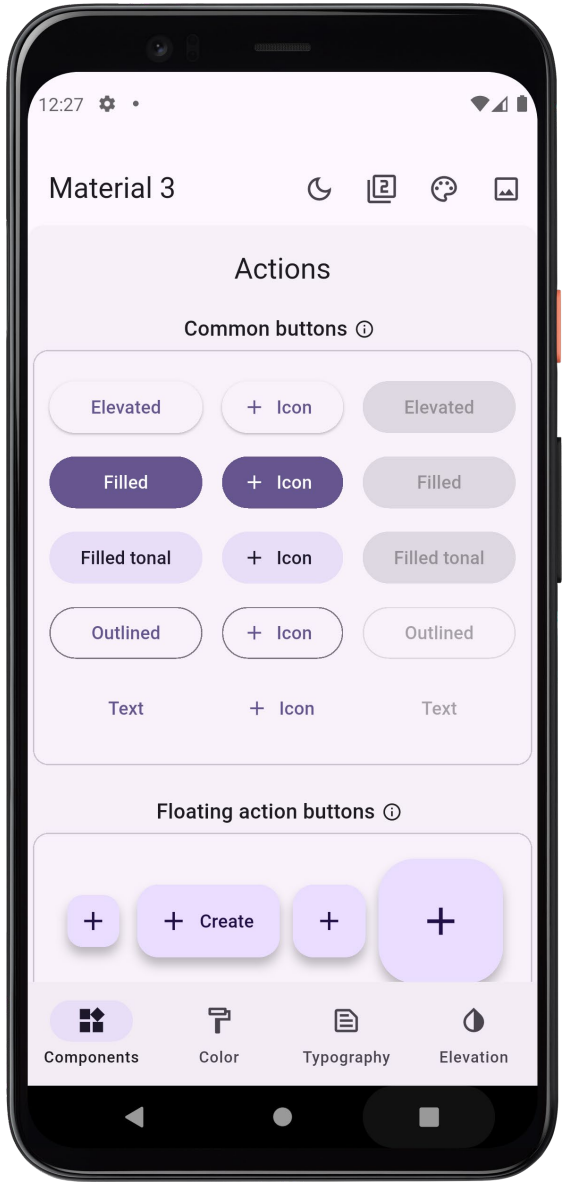


Table of contents

5. Takeaways

- React Native is an open-source cross-platform framework for building mobile apps for Android and iOS from a single codebase
  - It is based on React, so apps are built using JavaScript/TypeScript components
  - React Native renders native Android and iOS views
  - Expo Go simplifies testing apps on real devices
- Flutter is an open-source cross-platform framework created by Google for building apps for mobile, web, and desktop from a single codebase
  - Flutter apps are written in Dart
  - Flutter provides its own rich set of widgets to build UIs
  - The same codebase can target Android, iOS, web, and desktop platforms
- In practice:
  - Choose React Native if you already know React or JavaScript/TypeScript
  - Choose Flutter if you want a consistent UI across platforms and are comfortable learning Dart