Table of Contents

[Table of Contents 1](#_Toc98230101)

[1 Mobile development with Flutter 3](#_Toc98230102)

[1.1 Differences between Native, Web and Hybrid-Apps 3](#_Toc98230103)

[1.1.1 Native apps 3](#_Toc98230104)

[1.1.2 Web apps 4](#_Toc98230105)

[1.1.3 Hybrid apps 5](#_Toc98230106)

[1.2 What is Flutter 6](#_Toc98230107)

[1.3 Main features of Flutter 7](#_Toc98230108)

[1.3.1 Single language for development 7](#_Toc98230109)

[1.3.2 Full support of Google 8](#_Toc98230110)

[1.3.3 Agile development 8](#_Toc98230111)

[1.3.4 Hot reload 8](#_Toc98230112)

[*1.3.5* Special cases 9](#_Toc98230113)

[1.4 Widgets 10](#_Toc98230114)

[1.4.1 Common Widgets 11](#_Toc98230115)

[1.4.2 Text 11](#_Toc98230116)

[1.4.3 Row 12](#_Toc98230117)

[1.4.4 Column 13](#_Toc98230118)

[1.5 Stateless vs Stateful 14](#_Toc98230119)

[1.6 Architecture Overview 15](#_Toc98230120)

[1.7 Dart 16](#_Toc98230121)

[1.7.1 Sound null safety 17](#_Toc98230122)

[1.7.2 Compiler technology 17](#_Toc98230123)

[1.8 Flutter versus React Native 18](#_Toc98230124)

[1.8.1 Main features of React Native 19](#_Toc98230125)

[1.8.2 Main disadvantages of React Native 20](#_Toc98230126)

[1.8.3 When to choose what 20](#_Toc98230127)

[1.9 Flutter versus native 22](#_Toc98230128)

[1.9.1 Main features of native 22](#_Toc98230129)

[1.9.2 Main disadvantages of native 23](#_Toc98230130)

[2 References 25](#_Toc98230131)

# Mobile development with Flutter

***Sebastian Stecher***

An app, which is the abbreviation of “application”, is a software which can be installed and run on every electronic device for instance a smartphone or a computer. Most of the apps fulfil a specific purpose. [[1]](#footnote-2)

For example, an alarm clock app is only designed for users who need an alarm at a certain time. This app can’t be used for anything else, such as using it as a calendar or day planner. [[2]](#footnote-3)

The goal of YourTable is to simplify and speed up the process of reserving a table. YourTable's mobile application, which allows users to reserve a table of their choice, was built using the Flutter framework.

## Differences between Native, Web and Hybrid-Apps

There are three main types of applications. These types are native, web and hybrid-apps.

### Native apps

Native apps are installed on a specific device and are developed for this specific operating system of the device. These apps are available on app stores like Google Play, Apple App Store, etc.[[3]](#footnote-4)

As said before native apps are made for a specific operating system such as Android OS or Apple iOS. This means an app which is made for Android OS it will not run/work on a device which operating system is Apple iOS. Because of this it is necessary to build an app for every operating system which means a company has to spend more money and more effort. [[4]](#footnote-5)

#### Advantages of native apps

* + Native apps are faster than web apps.
  + These apps can work without an internet connection.
  + Because native apps must be authorized by the App Store, they are safer and more secure than web apps.
  + Native apps have access to system/device resources like GPS and cameras.

#### Disadvantages of native apps

* + When compared to web apps, these apps are more expensive to create.
  + The expense of maintaining and updating native apps is higher.
  + Developing the app for different operating systems is costly and time taking.

### Web apps

The user can access web apps through a browser on any device which has such a browser. They don’t need to be downloaded but the user needs an internet connection to be able to access the web app.[[5]](#footnote-6)

These applications can be used on every device which has a web browser and they are not operating system specific unlike the mobile app. The program is written in HTML and CSS, with interactive elements written in JavaScript or a comparable language. [[6]](#footnote-7)

#### Advantages of web apps

* They function in-browser, so no download is required.
* Easy to keep up, as they need a standard codebase irrespective of the operating system.
* They can update automatically.
* Easier and faster to make than native mobile apps.

#### Disadvantages of web apps

* They need an web connection.
* Web apps work slower than mobile apps.
* Web applications are tough to find since they aren't stored in a database like an app store.
* They have no security because web apps do nnot need to be approved by an app store.

### Hybrid apps

Mobile Hybrid-Apps are applications for smartphones and tablets that represent a combination of several application forms. A hybrid app combines the features of a native mobile app with a mobile web app. The primary differences between a hybrid and a native app could also be seen during the installation process. In a Hybrid-App, the whole program is not installed; instead, only a minimal framework is set up for it. When a user opens a hybrid app, it opens in a very browser-like interface, very similar to an internet app.[[7]](#footnote-8)

**Advantages of hybrid apps**

* They have a powerful platform independence.
* They are cheap in development.
* Hybrid apps do not need an internet connection.

**Disadvantages of hybrid apps**

* Hybrid apps are not nearly as good as native apps in performance.
* Problems with native functions from device just like the phone camera or GPS.
* Even if the user do not must install the complete application, the user must download a portion of it. This isn't the case with a web application.

The following table lists the characteristics of each application type. Each application type has its own strengths and weaknesses. Each product is different in itself. It is very important to consider all possibilities. Maybe each product was developed as a native app and worked well, but for the next product it is best to develop a web app.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Native apps | Web apps | Hybrid apps |
| Performance | + | - | - |
| Work without Internet connection | + | - | + |
| Approved by app store | + | - | + |
| Work without download | - | + | - |
| Fast to build and cheap | - | + | + |

**Figure 1: Comparison of native, web and hybrid applications**

## What is Flutter

Flutter was created by Google and is a free open-source mobile UI Framework. The first version was released in May 2017[[8]](#footnote-9). Flutter allows you to create a native mobile application and a web application with only one codebase. A disadvantage of native apps is that you need to develop for every operating system a different application. With only one codebase means that Flutter combines Apple iOS and Android OS. Because of this it is possible use one programming language and one codebase to create two different apps.[[9]](#footnote-10)

Flutter consists of two important parts. First of all the Software Development Kit (SDK). This is a collection of tools which helps during the developing of an application. The SDK includes tools to compile the code into native machine code. The second part is the Framework which is an UI Library based on widgets. It contains a collection of reusable UI elements like buttons, text inputs and so on. These elements can be personalized by the developer.[[10]](#footnote-11)

The programming language which is used by Flutter is Dart. Dart is a typed object programming language which can be compared to C# and Java. The language itself was created six years before the release of Flutter (October 2011).[[11]](#footnote-12)

## Main features of Flutter

### Single language for development

As mentioned before Flutter uses only one programming language which is Dart. Because of this the developer only need to learn one language to develop apps for Android OS and Apple iOS. This simplifies the work of developer enormous. For developer who are familiar with C# or Java adapting to Dart will be not a difficult task. [[12]](#footnote-13)

### Full support of Google

The documentation of Flutter[[13]](#footnote-14) is quite complete, very detailed and well-structured. Apart from the classic docs, it is possible to watch video lessons from the Google team. Google also has a Youtube channel called “Flutter”[[14]](#footnote-15) which features a series called “Flutter Widget of the Week”. In this series you get every Widget of Flutter explained in around one minute. Furthermore, the community of Flutter became the last years bigger and bigger and is still growing, which brings a series of solutions for the best use of the tool. [[15]](#footnote-16)

### Agile development

Flutter provides much agility to the developers. It supplies a widget system for development, because of that it is much easier to create elements who are interactive when building the software. [[16]](#footnote-17)

### Hot reload

With the agility of the development Flutter supply a feature called “hot reload”. This feature helps the developer to fast and easily experiment with the native application. Hot reload injects updated source code files into the running Dart Virtual Machine (VM). The Flutter framework then automatically rebuilds the widget tree after the VM updates classes with the new versions of fields and functions. This allows you to see the changes as fast as possible. [[17]](#footnote-18)

Beside the hot reload there are two other ways to run your application. One of them is called “hot restart”, this feature loads code changes into the VM and then restarts the Flutter app, losing the app state. The other way is the “full restart”. The full restart restarts the iOS, Android or web app. It recompiles the Java / Kotlin / ObjC / Swift code, because of this it takes longer than the other two methods. [[18]](#footnote-19)



**Figure 2: Shows Android Studio and where the Hot Reload button is placed**

### Special cases

There are also special cases where the hot reload not work as wanted. In this case a hot restart or a full restart is needed.

An application is killed:

* When an app is killed the hot reload can break. The app can break for example if it was in the background for too long.[[19]](#footnote-20)

Compilation errors:

* If a change in the code leads to a compilation error the hot reload generates an error message like: [[20]](#footnote-21)

Hot reload was rejected: '/Users/obiwan/Library/Developer/CoreSimulator/Devices/AC94F0FF-16F7-46C8-B4BF-218B73C547AC/data/Containers/Data/Application/4F72B076-42AD-44A4-A7CF-57D9F93E895E/tmp/ios\_testWIDYdS/ios\_test/lib/main.dart': warning: line 16 pos 38: unbalanced '{' opens here  
Widget build(BuildContext context) {  
^  
'/Users/obiwan/Library/Developer/CoreSimulator/Devices/AC94F0FF-16F7-46C8-B4BF-218B73C547AC/data/Containers/Data/Application/4F72B076-42AD-44A4-A7CF-57D9F93E895E/tmp/ios\_testWIDYdS/ios\_test/lib/main.dart': error: line 33 pos 5: unbalanced ')'  
);  
^

Changing fonts:

* Hot reload does not change fonts. In this case a hot restart is needed.[[21]](#footnote-22)

## Widgets

The inspiration of the Flutter widgets came from React (JavaScript library for building user interfaces). The idea behind the widgets is that the developer builds the user interface out of widgets. Widgets describe the appearance of their view based on their current setup and status. The widget rebuilds its description when the state changes, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to move from one state to the next. [[22]](#footnote-23)

### Common Widgets

Flutter provides a set of strong basic widgets that are very commonly used. Some will be explained in more detail in the coming lines.

### Text

“The Text widget displays a string as text with a single style.”[[23]](#footnote-24) Depending on the layout requirements, the string can span numerous lines or be presented on the same.

This example shows how the implementation of a Text widget works:

FittedBox(  
 fit: BoxFit.fitWidth,  
 child: Padding(  
 padding: EdgeInsets.only(top: 5.0, left: 10,bottom:10),  
 child: Text(  
 AuthService.*restaurants*[widget.restaurantIndex].restaurantName,  
 style: TextStyle(fontSize: 25),  
 ),  
 )  
),

Ein Bild, das Text, Essen, Sandwich, Zwischenmahlzeit enthält.

Automatisch generierte Beschreibung

**Figure 3: Shows a Text widget**

### Row

“A widget that displays its children in a horizontal array.”*[[24]](#footnote-25)* A Row widget is important if we want to have more widgets side by side. It is important to know that an error occurs when too much space is required.

This example shows how to implement a Row widget with two children:

Row(  
 children: [  
 Text("Kinderstuhl"),  
 Checkbox(  
 activeColor: secondColor,  
 value: childChair,  
 onChanged: (value){  
 setState(() {  
 childChair = value!;  
 });  
 }  
 ),  
 ],  
),



**Figure 4: Shows a Row widget**

Just a quick example of what it would look like if the line is longer than the space available:



**Figure 5: Shows an error because of too little space**

### Column

“A widget that displays its children in a vertical array.”[[25]](#footnote-26) The column is like the Row widget but in a vertical way. It is used to have widgets among each other. As with the Ro widget an error will occur when too much space is required.

This example shows how the implementation of a Column widget with three children works:

Column(  
 mainAxisAlignment: MainAxisAlignment.start,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Row(…),  
 Padding(…),  
 Padding(…),  
 ],  
),



**Figure 6: Shows a Column widget**

## Stateless vs Stateful

A widget in Flutter can either be stateless or stateful. Widgets who are reacting to an user activity are stateful. This means a stateless widget does not change. These widgets are useful when only the configuration information within the object itself and therefore the BuildContext during which the widget is build are important. For example, the Text widget is stateless because it does not change when a user interacts with it. Stateful widgets can change their properties during run-time.

**Stateless Widgets are**

* static widgets
* which have no state, after they rendered once they will not update themselves
* and do not depend on data change
* Example: Text and Icon

**Stateful Widgets are**

* dynamic widgets
* whose user action or data change can cause an update during runtime of the widget
* in an internal state
* Example: Checkbox and Radio Button

## Architecture Overview

Flutter reuses as much code as possible across different operating systems and provides applications with a direct interface to the underlying platform services. While the application is in the development phase, Flutter apps run on a virtual machine that provides hot-reload capability. Once the application is released, Flutter apps are compiled to the machine code.[[26]](#footnote-27)

*“Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has privileged access to the layer below, and every part of the framework level is designed to be optional and replaceable.“*[[27]](#footnote-28)

Generally, developers work with Flutter through the Flutter framework which is written in the programming language Dart. It includes the layout, a diverse selection of platforms, and numerous levels of foundational libraries. These layers are organized in the following order from bottom to top:[[28]](#footnote-29)

* Basic core classes, as well as building block services like animation, drawing, and gestures, provide regularly used abstractions over the underlying base.
* The rendering layer provides an abstraction for working with layout and constructing a renderable object tree.
* The widgets layer is a composition abstraction. In the widgets layer, each render object in the rendering layer has a matching class. Furthermore, the widgets layer lets developers create reusable class combinations. This is where the reactive programming model is first introduced.
* The Material and Cupertino libraries provide extensive sets of controls that implement the Material and iOS design languages using the widget layer's composition primitives.



**Figure 7: Flutter architecture overview**

## Dart

Dart is a client-optimized programming language for developing fast apps on any platform. Its purpose is to provide the most productive cross-platform programming language and a versatile runtime environment for app frameworks. [[29]](#footnote-30)

“Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many core developer tasks like formatting, analyzing, and testing code.”[[30]](#footnote-31)

Dart is type safe. Type safe means that a language guarantees that the variable’s value always is the same as the variable’s static type. Dart ensures this using a combination of static type checking and runtime checks. [[31]](#footnote-32)

The datatype dynamic is also provided by the Dart language. When a variable is dynamic it can change the datatype during its lifetime. All Dart objects are based on the type dynamic. The developer can also call a method on a dynamic variable and the compiler will not check if this method exists. Only when the program is executed it will throw an exception if the method is not existing. [[32]](#footnote-33)

### Sound null safety

Null safety means that no variable in the code can contain null except the developer explicit says they can. [[33]](#footnote-34)

In the following code, no variable can ever be null if null safety is enabled:

//Example of variables who can not be null  
String name = "I can not be null";  
var age = 19; // Used as int  
final isNull = false;

As mentioned earlier, a variable can be null only if the developer specifies so. This indication would look like the following code:

//Example of a variable which can be null using the "?" operator  
String? name = null;

### Compiler technology

Because of the compiler technology of Dart, it is possible to run the code in different ways. One way for native applications is a virtual machine with just-in-time compilation. On the other hand, there is a way called ahead-of-time compiler to produce the machine code. Because of the just-in-time compiler it is possible to use the hot reload function.

“When apps are ready to be deployed to production — whether you’re publishing to an app store or deploying to a production backend — the Dart AOT compiler enables ahead-of-time compilation to native ARM or x64 machine code. Your AOT-compiled app launches with consistent, short startup time.”[[34]](#footnote-35)



**Figure 8: Shows the Dart toolchain**

A Dart runtime is required to execute code, regardless of whatever platform you use or how you create it. The following key tasks are handled by this runtime:[[35]](#footnote-36)

* Unused memory - Dart does not tolerate unused memory. Because of this it uses a garbage collector which is used by a managing memory model to reclaim the unused memory.
* Type checks - Most of the type checks in Dart are static but there are still a few checks who are from the type dynamic. One of those dynamic checks are by type check and cast operators.
* Isolation - The primary isolation, as well as any additional isolates created by the app, is controlled by the Dart runtime.

## Flutter versus React Native

React native is as well as Flutter a cross-platform development framework. The framework is used for building mobile apps using the programming language JavaScript. Because it is cross-platform it uses the same codebase for devices with Android and iOS. React Native was created by Facebook in 2013[[36]](#footnote-37). It was first released in March 2015[[37]](#footnote-38). As mentioned above Flutter was released in May 2017 this means React is nearly two years older than Flutter. Because of that it has a bigger community than Flutter and Facebook had more time to stabilize the API and fixing issues.[[38]](#footnote-39)

### Main features of React Native

**Fast refresh**

What is called hot reload in Flutter is called fast refresh in React Native. This feature allows developer to test their code as fast as possible.[[39]](#footnote-40)

**Popular Language**

React Native uses a very popular programming language which many developers know. JavaScript is extremely more popular than Dart and this could be a reason why developers are more likely to use React Native than Flutter.[[40]](#footnote-41)

**Big community**

“React Native has a massive developer community. Not just that, but there are countless tutorials, libraries, and UI frameworks that make it easy to learn the technology – as well as quick and easy to develop with it.”[[41]](#footnote-42)

### Main disadvantages of React Native

**Less components**

React Native supports only basic components like a button. There are many other React Native components available in external repositories. They may be used in a project by a developer, but it will take more effort and time. This is not the case in Flutter. The Flutter framework provides much more widgets. For a developer using Flutter this saves time and it is possible to create many widgets out of the pre-made widgets from Flutter.[[42]](#footnote-43)

**Fragile UI**

Because React Native relies on native components, the developer can be certain that the app's components will be updated as soon as the OS UI changes. Very rarely this can break the UI of the application. Worse, if upgrades induce modifications to the Native Components API, they can become much more harmful. But this happens very rarely as well. In Flutter the UI is much more stable than in React Native. This is because the Flutter Framework recreates native components on their own.[[43]](#footnote-44)

**Big app size**

JavaScript code must be able to run in React Native applications (JavaScript Virtual Machine). Because Android misses this capability by default, apps must add a library that allows JavaScript code, resulting in apps that are significantly larger than their native Android counterparts. Although iOS apps do not have this issue, they are nonetheless larger than native apps.[[44]](#footnote-45)

### When to choose what

It can be said that after the mentioned features of both frameworks Flutter and React Native there is pretty much no necessary feature which the other framework does not have. Their main strength is the cross-platform nature, which both offer excellently. So, there is no one path that a development team theoretically has to take. Now the individuality of the development team comes into play. If the mobile app developers know Dart like a book, the way to go is Flutter rather than React Native. However, if the developers know JavaScript much better than Dart, React Native should be used.[[45]](#footnote-46)

“Remember that each application is different, so you must consider each one on its own merits. It’s always worth discussing your project with an experienced team of developers: people capable of considering the different approaches, with a varied enough skillset when it comes to cross-platform development – as if you seek the advice of programmers who know just one framework, they’ll likely steer you to use that tool.”[[46]](#footnote-47)

The following table shows the strengths of Flutter and React Native, and when it's probably better to choose one over the other. For example, your development team is not familiar with either of them, but they are really good at JavaScript. So, in this case, React Native is the better choice.

|  |  |  |
| --- | --- | --- |
|  | Flutter | React Native |
| Cross-platform | + | + |
| Using Dart | + | - |
| Using JavaScript | - | + |
| Hot reload/fast refresh | + | + |
| Low app size | - | - |
| Massive Community | - | + |
| Many pre-made widgets | + | - |

**Figure 9: Comparison of Flutter and React Native**

## Flutter versus native

Unlike than cross-platform frameworks, native application development is operating system specific. This means you develop either an application for Android or for iOS. Now that there are different applications for different operating systems, each OS has its own programming languages that are supported by the device's operating system. For Android, there is Kotlin and Java, as well as the Android Studio development environment, in which the application can be created. For iOS, there are the languages Swift and Objective-C. The development environment for creating iOS applications is called Xcode. [[47]](#footnote-48)

### Main features of native

**Perfect use of device resources**

Since native applications are written for one operating system, the use of device resources cannot be better. These resources are, for example, the device's camera or GPS. [[48]](#footnote-49)

**CPU usage**

The CPU consumption of native application is compared to Flutter applications about 50 percent less. [[49]](#footnote-50)

**Memory usage**

“As for memory usage, native apps show the most efficient memory consumption.”*[[50]](#footnote-51)* The memory consumption of applications created with Flutter is twice as high as the memory consumption of native applications.[[51]](#footnote-52)

### Main disadvantages of native

**Development time and cost**

Creating native applications takes more time because developers must create a separate application for each operating system. Also, the costs are higher because a company needs more developers to create two different applications.[[52]](#footnote-53)

**Maintaining the code**

Supporting a native application is more labor intensive in most cases. First, maintaining two code bases requires significant effort. Second, developers must create separate updates for each platform and identify issues along the way.[[53]](#footnote-54)

The following table shows the strengths of Flutter and native apps, and moreover, which one a development team should use. For example, if the team is small and doesn't have so much budget for development, Flutter is better than native development because it is much cheaper in development.

|  |  |  |
| --- | --- | --- |
|  | Flutter | Native |
| Cross-platform | + | - |
| Cheaper in development | + | - |
| Faster in development | + | - |
| Easier to maintain | + | - |
| Low app size | - | + |
| Less CPU usage | - | + |
| Perfect use of device resources | - | + |

**Figure 10: Comparison of Flutter and native applications**

# References

* Indeed Editorial Team: *What is an App? Types of Apps and Examples* -<https://www.indeed.com/career-advice/career-development/what-is-an-app> [accessed on 29 December 2021].
* Dheeraj Pandey: *Differences between Native Apps and Web Apps* - <https://www.geeksforgeeks.org/difference-between-native-apps-and-web-apps/> [accessed on 29 December 2021].
* Ionos: *Hybrid-App: Was unterscheidet sie von anderen App-Formaten* - <https://www.ionos.at/digitalguide/websites/web-entwicklung/hybrid-app-das-beste-aus-web-und-native-app/> Was ist eine Hybrid-App? [accessed on 28 February 2022].
* Gaël Thomas: *What is Flutter and Why You Should Learn it in 2020* - <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/> [accessed on 31 December 2020]
* LogAp: What is Flutter: main features and 6 advantages for mobile development: Flutter’s main features and advantages - <https://logap.com.br/en/blog/what-is-flutter/> [accessed on 30 December 2020]
* Flutter: *Hot reload* - <https://docs.flutter.dev/development/tools/hot-reload> [accessed on 2 January 2022]
* Flutter: Text class - <https://api.flutter.dev/flutter/widgets/Text-class.html> [accessed on 4 January 2022]
* Flutter: Row class - <https://api.flutter.dev/flutter/widgets/Row-class.html> [accessed on 4 January 2022]
* Flutter: Column class - <https://api.flutter.dev/flutter/widgets/Column-class.html> [accessed on 4 January 2022]
* Flutter: Flutter architectural overview - <https://docs.flutter.dev/resources/architectural-overview> [accessed on 26 February 2022]
* Dart: Dart overview - [https://dart.dev/overview#platform](https://dart.dev/overview" \l "platform) [accessed on 28 February 2022]
* Dart: The Dart type system - <https://dart.dev/guides/language/type-system> [accessed on 28 February 2022]
* Dart: Sound null safety - <https://dart.dev/null-safety> [accessed on 4 March 2022]
* Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 5 March 2022]
* Codemagic: *Dart vs JavaScript: detailed comparison* - <https://blog.codemagic.io/dart-vs-javascript/> [accessed on 5 March 2022]
* Surf: *Flutter vs Native: What is Best for Your Project?* - <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022]

1. Cmp. Indeed Editorial Team: *What is an App? Types of Apps and Examples: What is an App?* -<https://www.indeed.com/career-advice/career-development/what-is-an-app> [accessed on 29 December 2021]. [↑](#footnote-ref-2)
2. ibid [↑](#footnote-ref-3)
3. Cmp. Dheeraj Pandey: *Differences between Native Apps and Web Apps: Native Apps* - <https://www.geeksforgeeks.org/difference-between-native-apps-and-web-apps/> [accessed on 29 December 2021]. [↑](#footnote-ref-4)
4. ibid [↑](#footnote-ref-5)
5. Cmp. Dheeraj Pandey: *Differences between Native Apps and Web Apps: Web Apps* - <https://www.geeksforgeeks.org/difference-between-native-apps-and-web-apps/> [accessed on 29 December 2021]. [↑](#footnote-ref-6)
6. ibid [↑](#footnote-ref-7)
7. Cmp. Ionos: *Hybrid-App: Was unterscheidet sie von anderen App-Formaten?: Was ist eine Hybrid-App?* - <https://www.ionos.at/digitalguide/websites/web-entwicklung/hybrid-app-das-beste-aus-web-und-native-app/> Was ist eine Hybrid-App? accessed on 28 February 2022. [↑](#footnote-ref-8)
8. Gaël Thomas: *What is Flutter and Why You Should Learn it in 2020: What is Flutter?* - <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/> [accessed on 30 December 2021] [↑](#footnote-ref-9)
9. Cmp. Gaël Thomas: *What is Flutter and Why You Should Learn it in 2020: What is Flutter?* - <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/> [accessed on 30 December 2021] [↑](#footnote-ref-10)
10. Cmp. Gaël Thomas: *What is Flutter and Why You Should Learn it in 2020: What is Flutter?* - <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/> [accessed on 30 December 2021] [↑](#footnote-ref-11)
11. ibid [↑](#footnote-ref-12)
12. Cmp. LogAp: *What is Flutter: main features and 6 advantages for mobile development: Flutter’s main features and advantages: 2. Single language for development* - <https://logap.com.br/en/blog/what-is-flutter/> [accessed on 30 December 2021] [↑](#footnote-ref-13)
13. Flutter Documentation - <https://docs.flutter.dev/> [↑](#footnote-ref-14)
14. Youtube channel - <https://www.youtube.com/c/flutterdev> [↑](#footnote-ref-15)
15. Cmp. LogAp: *What is Flutter: main features and 6 advantages for mobile development: Flutter’s main features and advantages: 6. Google Support* - <https://logap.com.br/en/blog/what-is-flutter/> [accessed on 30 December 2021] [↑](#footnote-ref-16)
16. Cmp. LogAp: *What is Flutter: main features and 6 advantages for mobile development: Flutter’s main features and advantages: 1. Simplified and agile development* - <https://logap.com.br/en/blog/what-is-flutter/> [accessed on 30 December 2020] [↑](#footnote-ref-17)
17. Cmp. Flutter: *Hot reload* - <https://docs.flutter.dev/development/tools/hot-reload> [accessed on 2 January 2022] [↑](#footnote-ref-18)
18. Cmp. Flutter: *Hot reload: What is the difference between hot reload, hot restart, and full restart?* - <https://docs.flutter.dev/development/tools/hot-reload> [accessed on 2 January 2022] [↑](#footnote-ref-19)
19. Cmp. Flutter: *Hot reload: Special cases: An app is killed* - <https://docs.flutter.dev/development/tools/hot-reload> [accessed on 2 January 2022] [↑](#footnote-ref-20)
20. Cmp. Flutter: *Hot reload: Special cases: Compilation errors* - <https://docs.flutter.dev/development/tools/hot-reload> [accessed on 2 January 2022] [↑](#footnote-ref-21)
21. Cmp. Flutter: *Hot reload: Special cases: Changing fonts* - <https://docs.flutter.dev/development/tools/hot-reload> [accessed on 2 January 2022] [↑](#footnote-ref-22)
22. Cmp. Flutter: *Introduction to widgets* - <https://docs.flutter.dev/development/ui/widgets-intro> [accessed on 2 January 2022] [↑](#footnote-ref-23)
23. Flutter: *Text class* - <https://api.flutter.dev/flutter/widgets/Text-class.html> [accessed on 4 January 2022] [↑](#footnote-ref-24)
24. Flutter: *Row class* - <https://api.flutter.dev/flutter/widgets/Row-class.html> [accessed on 4 January 2022] [↑](#footnote-ref-25)
25. Flutter: *Column class* - <https://api.flutter.dev/flutter/widgets/Column-class.html> [accessed on 4 January 2022] [↑](#footnote-ref-26)
26. Cmp. Flutter: *Flutter architectural overview* - <https://docs.flutter.dev/resources/architectural-overview> [accessed on 26 February 2022] [↑](#footnote-ref-27)
27. Flutter: *Flutter architectural overview: Architectural layers* - <https://docs.flutter.dev/resources/architectural-overview> [accessed on 26 February 2022] [↑](#footnote-ref-28)
28. Cmp. Flutter: *Flutter architectural overview: Architectural layers* - <https://docs.flutter.dev/resources/architectural-overview> [accessed on 26 February 2022] [↑](#footnote-ref-29)
29. Cmp. Dart: *Dart overview* - <https://dart.dev/overview#platform> [accessed on 28 February 2022] [↑](#footnote-ref-30)
30. Dart: *Dart overview* - <https://dart.dev/overview> [accessed on 4 March 2022] [↑](#footnote-ref-31)
31. Cmp. Dart: *The Dart type system* - <https://dart.dev/guides/language/type-system> [accessed on 28 February 2022] [↑](#footnote-ref-32)
32. Cmp. Dart: *Dart overview: Dart: The language* - <https://dart.dev/overview> [accessed on 4 March 2022] [↑](#footnote-ref-33)
33. Cmp. Dart: Sound null safety - <https://dart.dev/null-safety> [accessed on 4 March 2022] [↑](#footnote-ref-34)
34. Dart: *Dart overview: Dart: The platforms* - <https://dart.dev/overview> [accessed on 4 March 2022] [↑](#footnote-ref-35)
35. Cmp. Dart: *Dart overview: Dart: The platforms: The Dart runtime* - <https://dart.dev/overview> [accessed on 4 March 2022] [↑](#footnote-ref-36)
36. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native: How mature is React Native?* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 5 March 2022] [↑](#footnote-ref-37)
37. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native: How mature is React Native?* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 5 March 2022] [↑](#footnote-ref-38)
38. Cmp. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 5 March 2022] [↑](#footnote-ref-39)
39. Cmp. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native: Pros and cons of React Native* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 5 March 2022] [↑](#footnote-ref-40)
40. Cmp. Codemagic: *Dart vs JavaScript: detailed comparison: Dart vs JavaScript Comparison: Dart vs JavaScript: Popularity* - <https://blog.codemagic.io/dart-vs-javascript/> [accessed on 5 March 2022] [↑](#footnote-ref-41)
41. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native: Pros and cons of React Native* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021#f9> [accessed on 5 March 2022] [↑](#footnote-ref-42)
42. Cmp. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native: Pros and cons of React Native* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 5 March 2022] [↑](#footnote-ref-43)
43. ibid [↑](#footnote-ref-44)
44. ibid [↑](#footnote-ref-45)
45. Cmp. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native: Summary* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 6 March 2022] [↑](#footnote-ref-46)
46. Bartosz Skuza, Damian Włodarczyk, and Agnieszka Mroczkowska: *Flutter vs. React Native – What to Choose in 2022?: Introduction to Flutter and React Native: Summary* - <https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021> [accessed on 6 March 2022] [↑](#footnote-ref-47)
47. Cmp. Surf: *Flutter vs Native: What is Best for Your Project?: What are native technologies?* - <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022] [↑](#footnote-ref-48)
48. Cmp. Surf: *Flutter vs Native: What is Best for Your Project?: Flutter vs native app: Speed of development*- <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022] [↑](#footnote-ref-49)
49. Cmp. Surf: *Flutter vs Native: What is Best for Your Project?: Flutter vs native app: Speed of development: App performance* - <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022] [↑](#footnote-ref-50)
50. Surf: *Flutter vs Native: What is Best for Your Project?: Flutter vs native app: Speed of development: App performance* - <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022] [↑](#footnote-ref-51)
51. Cmp. Surf: *Flutter vs Native: What is Best for Your Project?: Flutter vs native app: Speed of development: App performance* - <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022] [↑](#footnote-ref-52)
52. Cmp. Surf: *Flutter vs Native: What is Best for Your Project?: Flutter vs native app: Speed of development: App development cost* - <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022] [↑](#footnote-ref-53)
53. Cmp. Surf: *Flutter vs Native: What is Best for Your Project?: Flutter vs native app: Speed of development: Code Maintenance* - <https://surf.dev/flutter-vs-native/> [accessed on 6 March 2022] [↑](#footnote-ref-54)