Flutter

Flutter is Google’s mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.

<https://www.youtube.com/watch?v=fq4N0hgOWzU>

Developers are constantly looking to expand their knowledge and learn new technologies to help them build high-performing applications. App owners, on the other hand, seek to optimize time spent on development. That’s why Flutter has entered the mobile development world. It was announced by Google in February 2018 at Mobile World Congress. Google had previously introduced the alpha version at Google I/O in 2017. Since releasing this new framework, Google has continued to promote it at conferences as they want more people to use it.

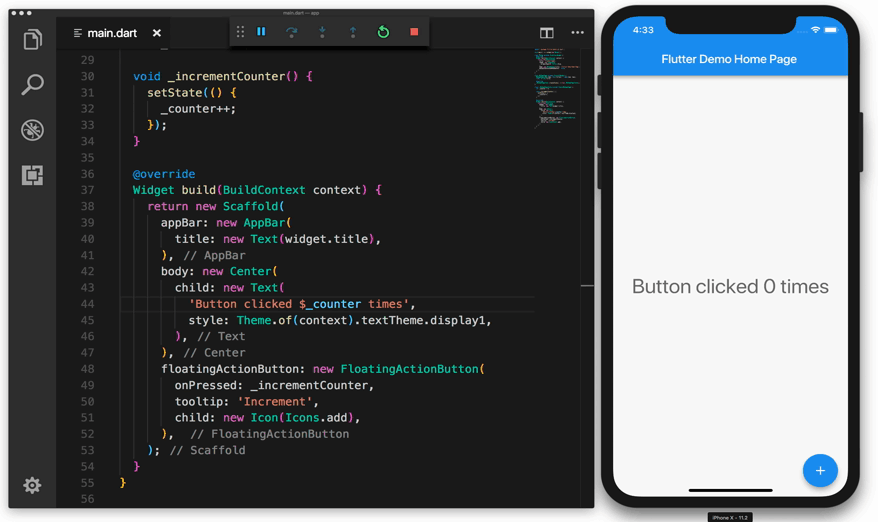
**Pros**

Flutter has a number of advantages that allow it to attract a large developer community and lots of app owners. Let’s take a look at benefits of Flutter framework to mobile app developers and business owners.

**Hot reload**

One great thing about Flutter is hot reload. Developers can see all the changes they’ve made to the code right away in the app. It takes just seconds to see changes, so teams can quickly add new features, fix bugs, and experiment. Hot reload enables better collaboration between developers and designers when they want to improve how the app looks and check effects immediately. There are some [limitations](https://flutter.io/hot-reload/#limitations) to hot reloading, though.

With native app development, you need to wait up to several minutes to see every change you make.



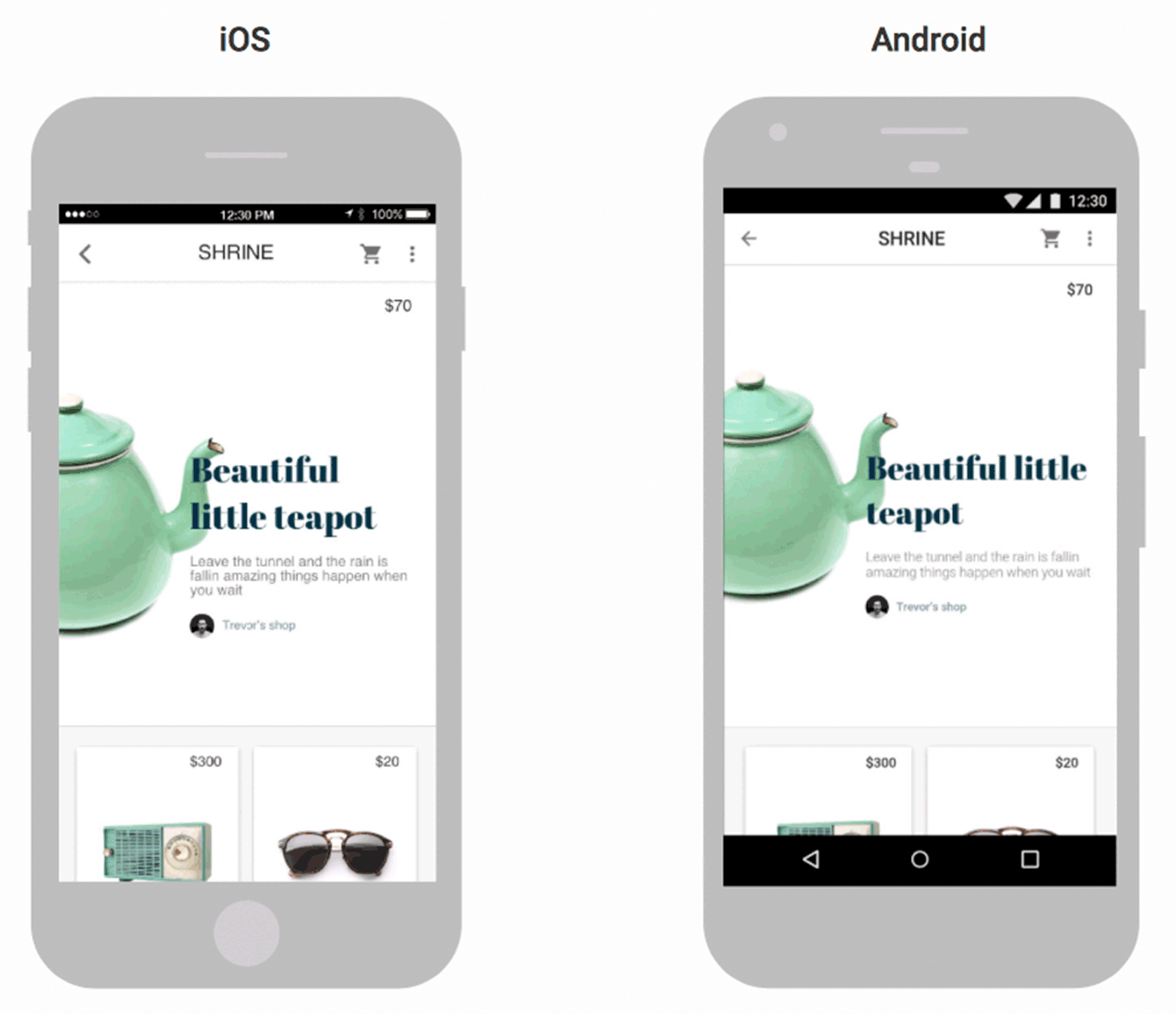
*Source: Flutter.io*

**Cross-platform development**

With cross-platform development, there’s no need to build separately for two platforms. You can build just once, have one codebase, and use that one codebase for two applications. Besides cross platform development, Flutter can give you a start on developing for the Fuchsia platform. Fuchsia is an experimental operating system in development at Google. Many think that Fuchsia is the future replacement for Android.

**Perfect for an MVP**

If you want to show your product to investors as soon as possible, you can build a Flutter mobile application that looks native on both Android and iOS. Thus, your investors will have a clear idea of what your [MVP](https://steelkiwi.com/blog/what-mvp-and-why-it-necessary/) looks like. Developing two separate apps would take much more time and money.



*Source:*[*Flutter.io*](https://flutter.io/technical-overview/)

**Less code**

The programming language of Flutter is Dart. Dart is object-oriented and strongly typed. Flutter resembles React Native, and its programming style is reactive and declarative. Flutter doesn’t need to use a JavaScript bridge, which improves app startup times and overall performance. Dart achieves this thanks to Ahead-of-Time, or AOT, compilation. Dart also makes use of Just-in-Time, or JIT, compilation. JIT compilation with Flutter improves the development workflow by allowing for a hot reload capability to refresh the UI during development without the need for an entirely new build.



*Source:*[*Steelkiwi.com*](https://steelkiwi.com/)

Flutter means less code. And where there’s less code, there are fewer bugs. With Dart, 100% of the code is usually shared between iOS and Android apps.

**Widgets**

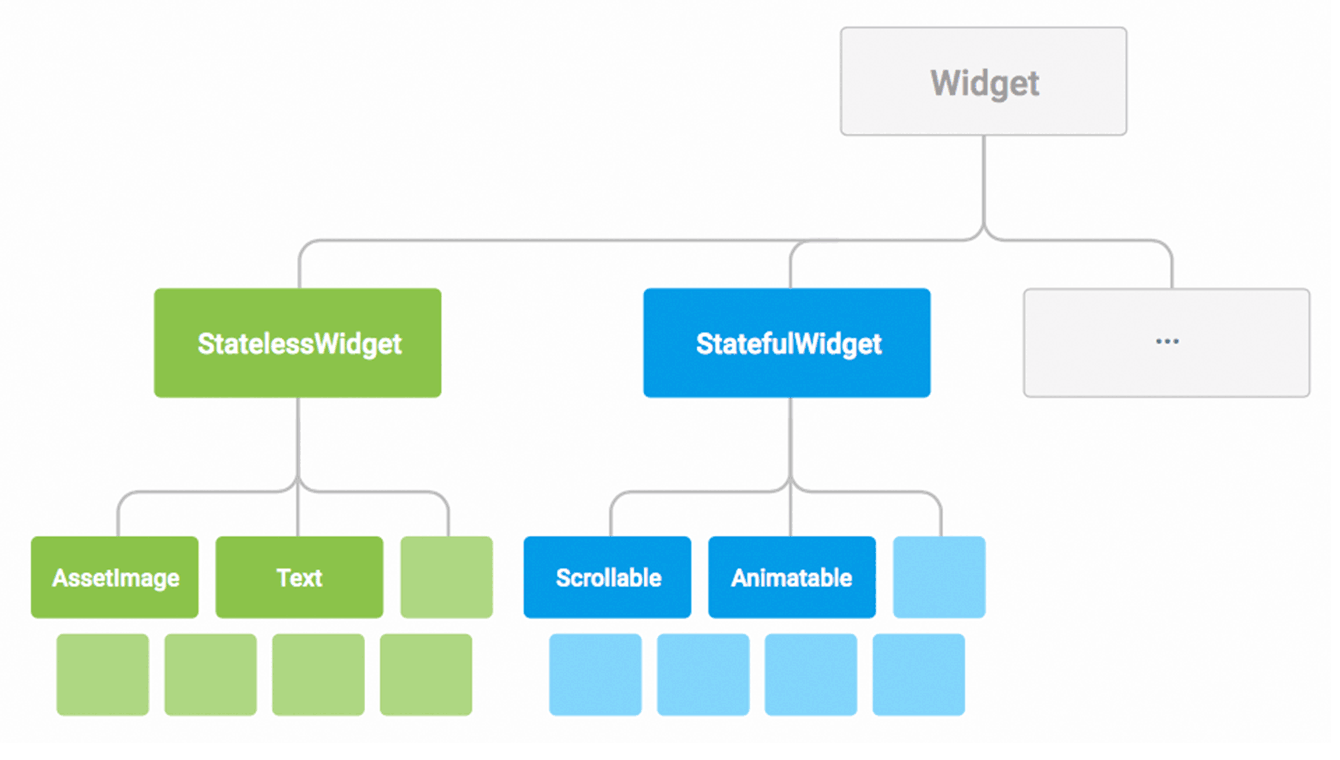
Widgets are necessary for an app’s view and interface. They must have a natural look and feel regardless of screen size. They also must be fast, extensible, and customizable. Flutter takes the everything’s a [widget](https://flutter.io/widgets/)approach. It has a rich set of widgets and extensive capabilities for creating complex custom widgets. In Flutter, widgets aren’t only used for views. They’re also used for entire screens and even for the app itself.

As Flutter’s documentation puts it, each widget is an immutable declaration of part of the user interface. Other frameworks separate views, view controllers, layouts, and other properties. Flutter, on the other hand, has a consistent, unified object model: the widget.

A widget can define:

* a structural element (like a button or menu);
* a stylistic element (like a font or color scheme);
* an aspect of the layout (like padding);
* and so on.

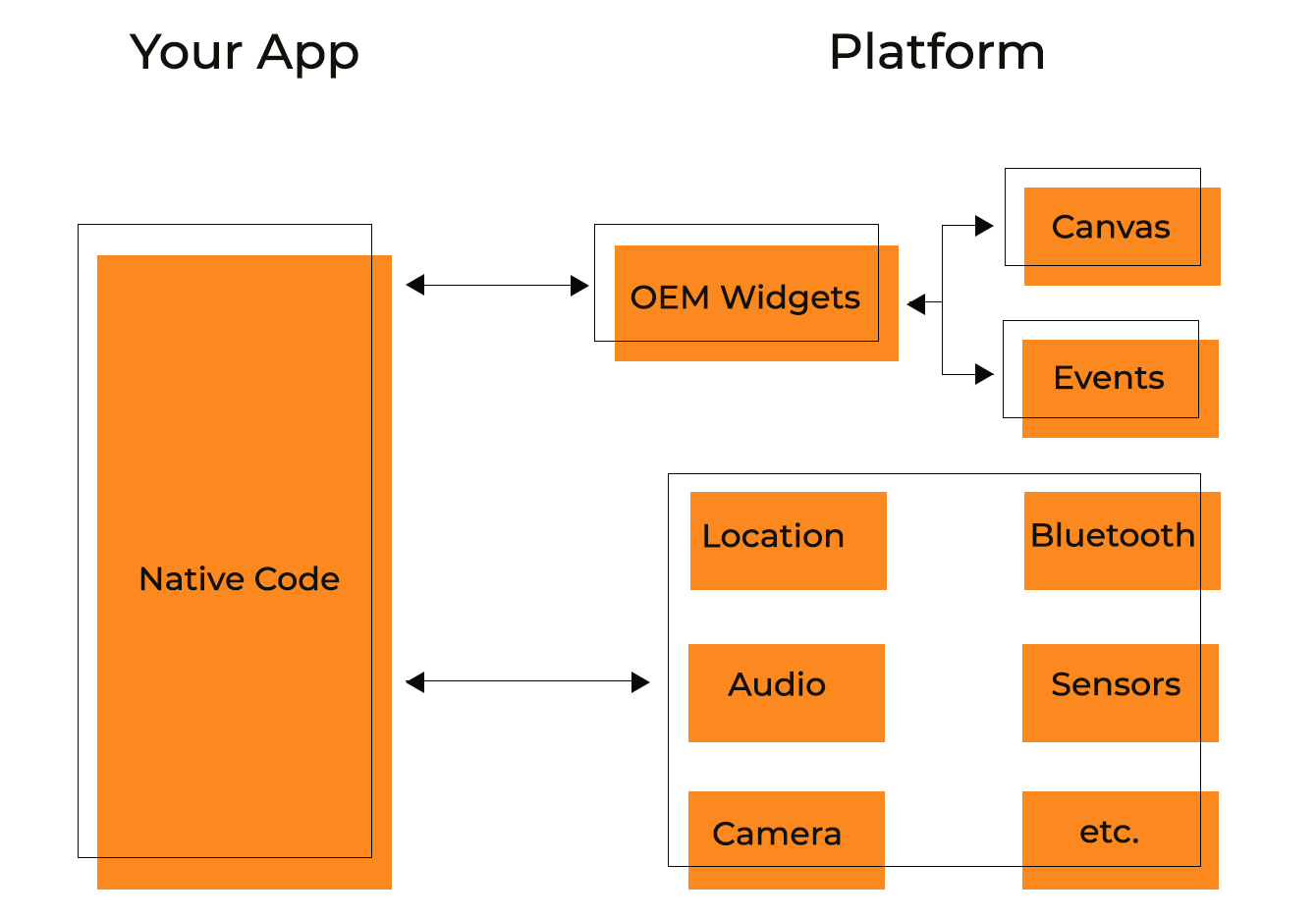
Widgets form a hierarchy based on their composition. Each widget nests inside of and inherits properties from its parent. There’s no separate application object. Instead, the root widget serves this role.



*Source:*[*Flutter.io*](https://flutter.io/technical-overview/)

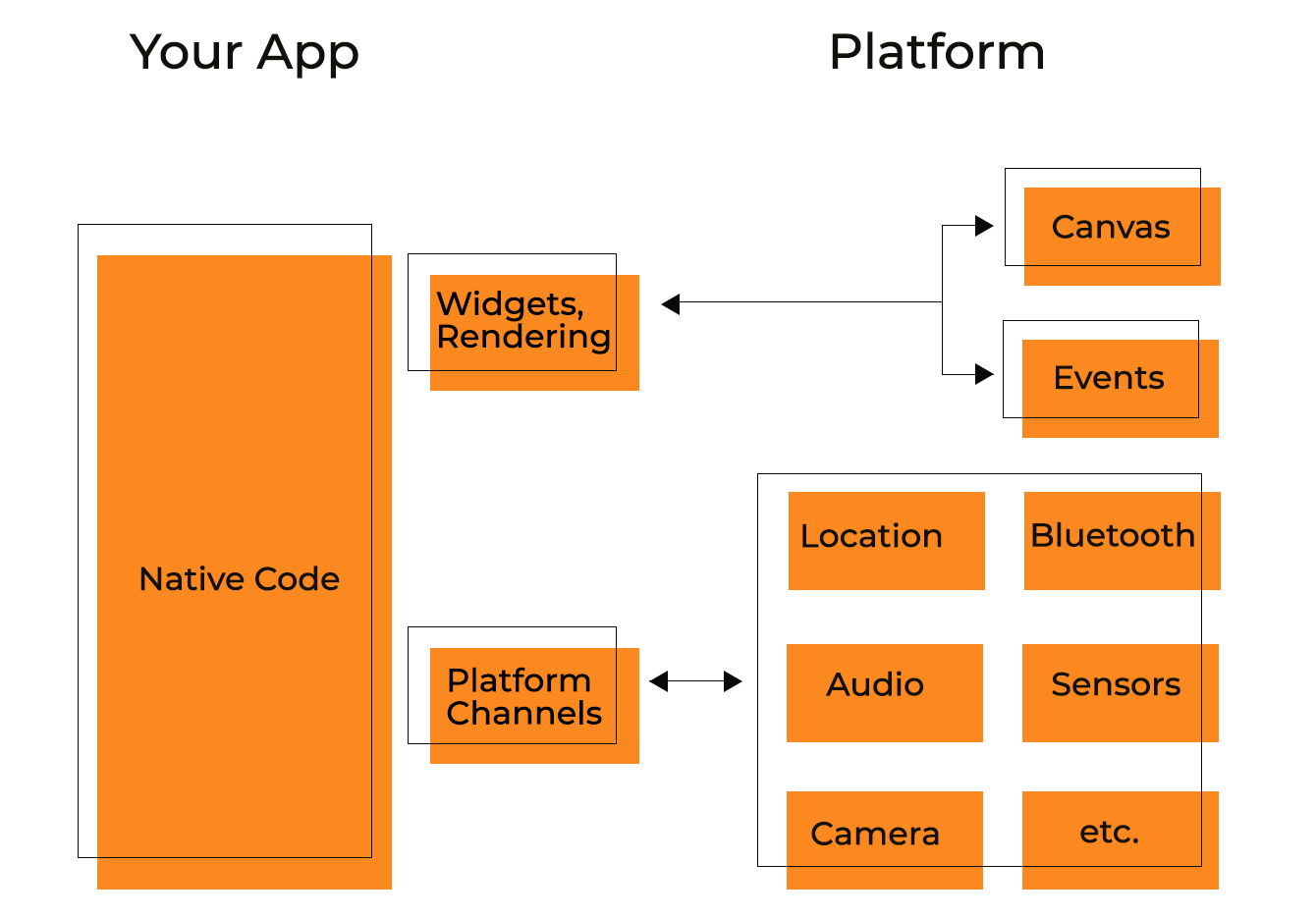
Flutter has a full set of widgets in Google’s [Material Design](https://flutter.io/widgets/material/) and in Apple’s style with the [Cupertino](https://flutter.io/widgets/cupertino/) pack. Widget rendering happens directly in the Skia engine without using Original Equipment Manufacturer widgets. So we get a smoother UI experience compared with other cross platform frameworks.

Here’s how native Android/iOS code interacts with other cross-platform frameworks like React Native:



*Source:*[*Hackernoon.com*](https://hackernoon.com/why-native-app-developers-should-take-a-serious-look-at-flutter-e97361a1c073)

And here’s how Flutter interacts with the platform:



*Source:*[*Hackernoon.com*](https://hackernoon.com/why-native-app-developers-should-take-a-serious-look-at-flutter-e97361a1c073)

All that the framework needs from the platform is a canvas in which to render the widgets so they can appear on the device screen and access to events (touches, timers, etc.) and services (location, camera, etc.).

**Access to native features**

Some platform-specific functions like camera and geolocation require access to native features. These functions need to be implemented by means of native languages, and Flutter gives the feel of developing on the native platform. Flutter allows you to reuse your existing Java, Swift, and Objective-C code to access native features and SDKs on iOS and Android.

**Official support**

Since Flutter is Google’s initiative, the framework has a big name behind it. There’s great support from the Material Design team and a community of over 1000 developers who are willing to contribute to the growth of this framework.

**Easy to set up**

Flutter is easy to set up and start coding on low-end machines. You don’t need to be worried that your operating system won’t be able to handle Flutter. Quite the contrary, Flutter doesn’t require a powerful machine.

Luigi Rosso, co-founder of 2Dimensions, says the following about Flutter app development:

“I love the friction-free iteration with Flutter. Hot Reload sets me in a feedback loop that keeps me focused and in tune with my work. One of my biggest productivity inhibitors are tools that run slower than the developer. Flutter finally resets that bar.”

**Cons**

The framework is relatively young, and it’s still growing and developing. The toolkit has its flaws. Let’s look at the them.

**Mobile only**

Flutter caters to mobile apps only and isn’t supported by web browsers. When you choose the technology stack you want to work with, this can influence your decision. If you want maximum device flexibility, then Flutter may not be the thing you’re looking for.

**Limited libraries**

Google supports Flutter and there are many helpful libraries. However, the framework is relatively new. You won’t find every functionality you’re looking for. Native apps, on the contrary, have everything in their native SDKs including Stripe, Twilio, and most streaming services such as WoWza and UStream. With Flutter, developers would have to build these libraries themselves, which is time-consuming.

**Continuous Integration**

There are many ready-made solutions for native Android and iOS apps that allow them to work with CI platforms like Travis and Jenkins. There are also ready-made Flutter solutions for most popular CI platforms like Travis or Circle. However, you need to set up the toolkit for these kinds of platforms. Travis can be used in [a number of repos](https://github.com/yjbanov/flutter_travis_sample). The framework also supports the [likes of Fastlane](https://flutter.io/fastlane-cd/) and [Nevercode](https://nevercode.io/blog/we-are-ready-for-flutter-build-test-and-distribute-flutter-apps-with-nevercode/). As for us, we’ve applied a complex decision using Gitlab CI with Fastlane for Flutter apps.

**Limited TV support**

If you’re wondering if apps built with the help of this framework can run on Android TV or Apple TV, then the answer is more no than yes. Flutter offers practically no support for Android TV and Apple TV.

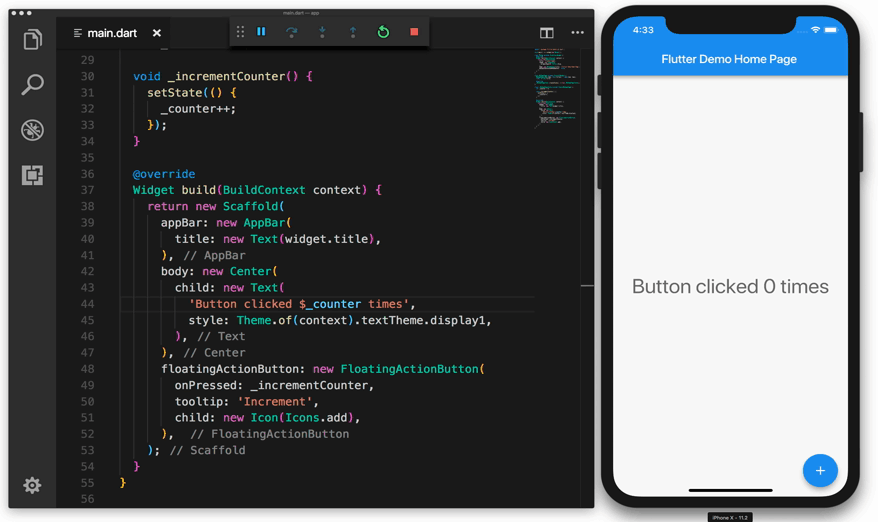
### Pros

Flutter has a number of advantages that allow it to attract a large developer community and lots of app owners. Let’s take a look at benefits of Flutter framework to mobile app developers and business owners.

### Hot reload

One great thing about Flutter is hot reload. Developers can see all the changes they’ve made to the code right away in the app. It takes just seconds to see changes, so teams can quickly add new features, fix bugs, and experiment. Hot reload enables better collaboration between developers and designers when they want to improve how the app looks and check effects immediately. There are some [limitations](https://flutter.io/hot-reload/#limitations) to hot reloading, though.

With native app development, you need to wait up to several minutes to see every change you make.



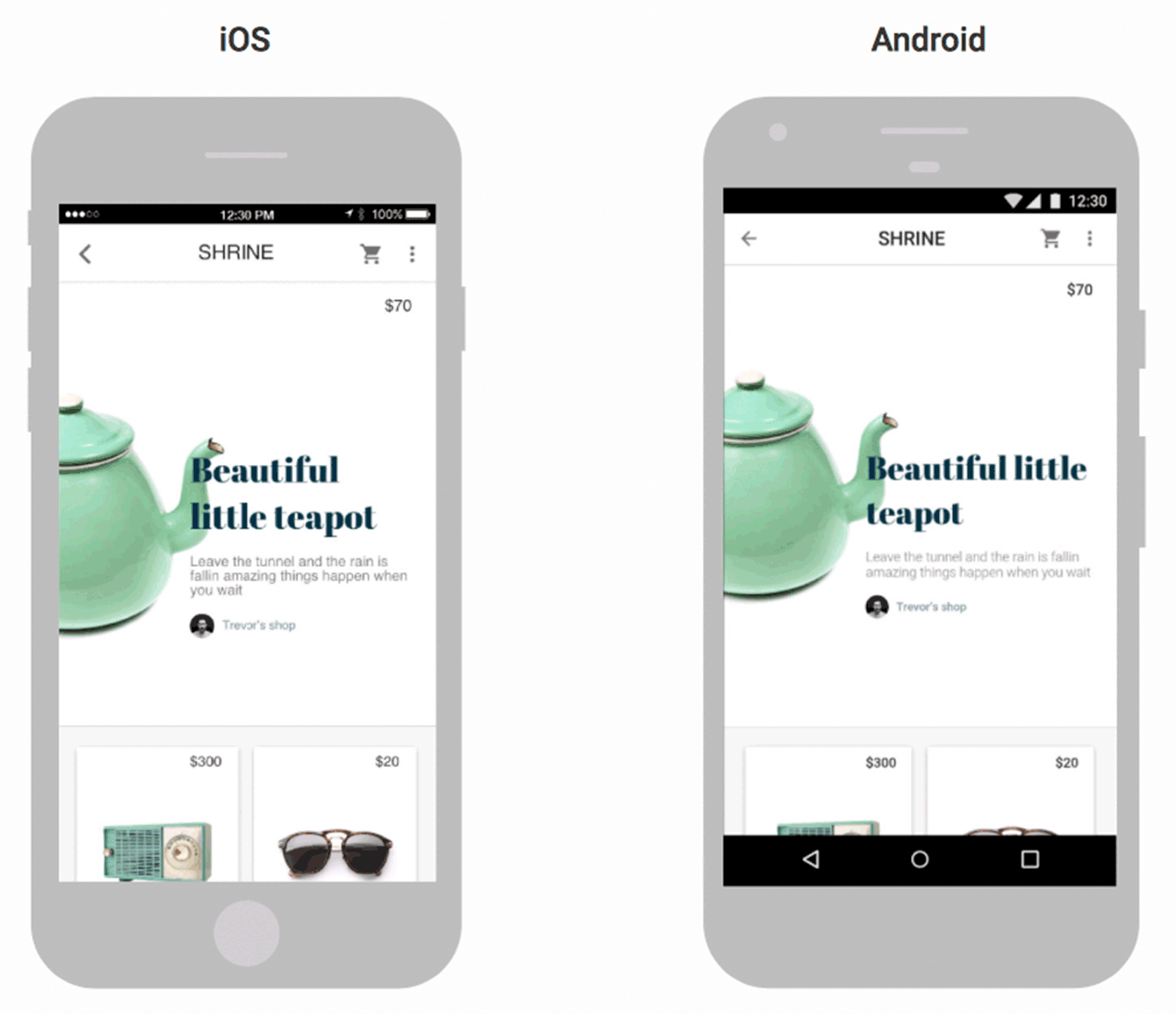
Source: Flutter.io

### Cross-platform development

With cross-platform development, there’s no need to build separately for two platforms. You can build just once, have one codebase, and use that one codebase for two applications. Besides cross platform development, Flutter can give you a start on developing for the Fuchsia platform. Fuchsia is an experimental operating system in development at Google. Many think that Fuchsia is the future replacement for Android.

### Perfect for an MVP

If you want to show your product to investors as soon as possible, you can build a Flutter mobile application that looks native on both Android and iOS. Thus, your investors will have a clear idea of what your [MVP](https://steelkiwi.com/blog/what-mvp-and-why-it-necessary/) looks like. Developing two separate apps would take much more time and money.



Source: [Flutter.io](https://flutter.io/technical-overview/)

### Less code

The programming language of Flutter is Dart. Dart is object-oriented and strongly typed. Flutter resembles React Native, and its programming style is reactive and declarative. Flutter doesn’t need to use a JavaScript bridge, which improves app startup times and overall performance. Dart achieves this thanks to Ahead-of-Time, or AOT, compilation. Dart also makes use of Just-in-Time, or JIT, compilation. JIT compilation with Flutter improves the development workflow by allowing for a hot reload capability to refresh the UI during development without the need for an entirely new build.



Source: [Steelkiwi.com](https://steelkiwi.com/)

Flutter means less code. And where there’s less code, there are fewer bugs. With Dart, 100% of the code is usually shared between iOS and Android apps.

### Widgets

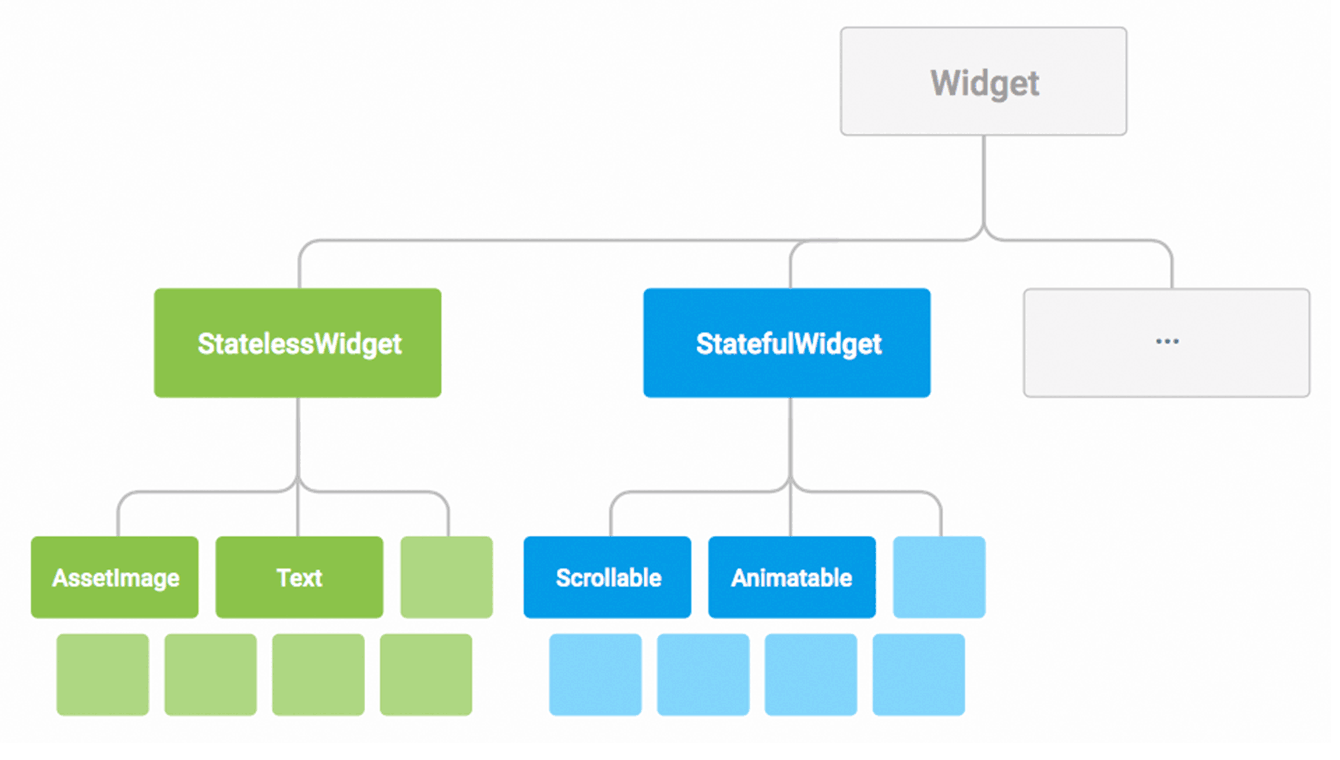
Widgets are necessary for an app’s view and interface. They must have a natural look and feel regardless of screen size. They also must be fast, extensible, and customizable. Flutter takes the everything’s a [widget](https://flutter.io/widgets/)approach. It has a rich set of widgets and extensive capabilities for creating complex custom widgets. In Flutter, widgets aren’t only used for views. They’re also used for entire screens and even for the app itself.

As Flutter’s documentation puts it, each widget is an immutable declaration of part of the user interface. Other frameworks separate views, view controllers, layouts, and other properties. Flutter, on the other hand, has a consistent, unified object model: the widget.

A widget can define:

* a structural element (like a button or menu);
* a stylistic element (like a font or color scheme);
* an aspect of the layout (like padding);
* and so on.

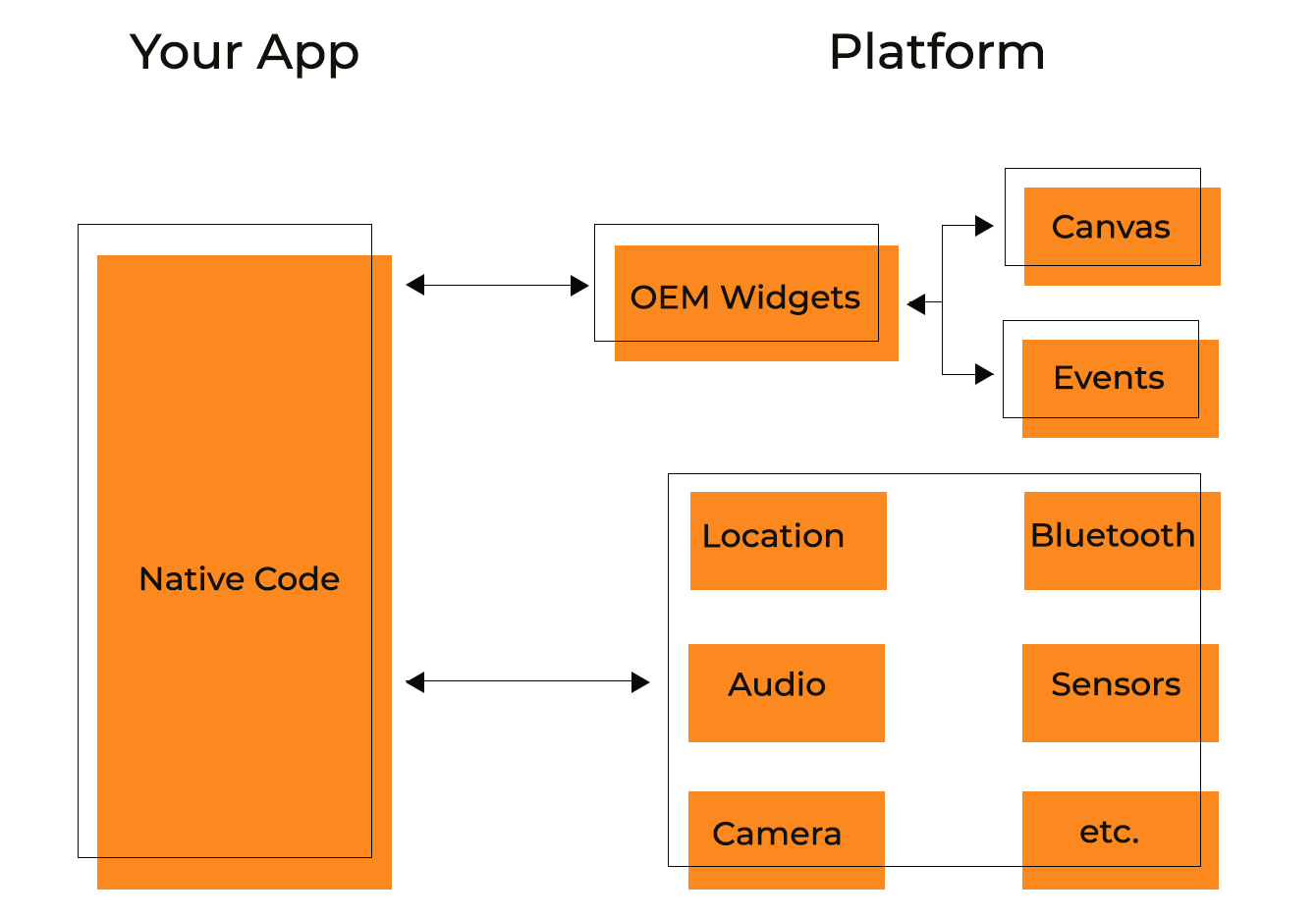
Widgets form a hierarchy based on their composition. Each widget nests inside of and inherits properties from its parent. There’s no separate application object. Instead, the root widget serves this role.



Source: [Flutter.io](https://flutter.io/technical-overview/)

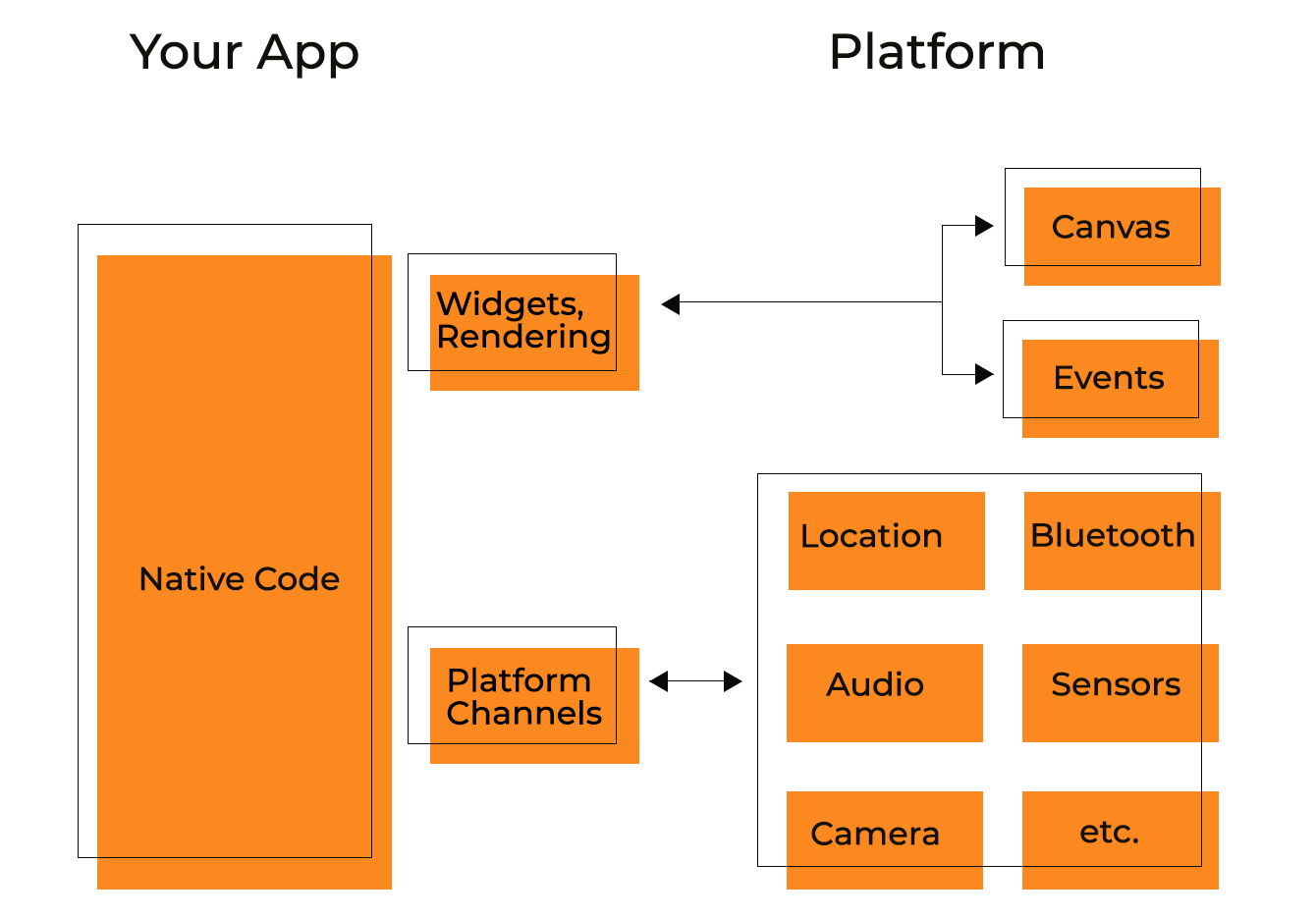
Flutter has a full set of widgets in Google’s [Material Design](https://flutter.io/widgets/material/) and in Apple’s style with the [Cupertino](https://flutter.io/widgets/cupertino/) pack. Widget rendering happens directly in the Skia engine without using Original Equipment Manufacturer widgets. So we get a smoother UI experience compared with other cross platform frameworks.

Here’s how native Android/iOS code interacts with other cross-platform frameworks like React Native:



Source: [Hackernoon.com](https://hackernoon.com/why-native-app-developers-should-take-a-serious-look-at-flutter-e97361a1c073)

And here’s how Flutter interacts with the platform:



Source: [Hackernoon.com](https://hackernoon.com/why-native-app-developers-should-take-a-serious-look-at-flutter-e97361a1c073)

All that the framework needs from the platform is a canvas in which to render the widgets so they can appear on the device screen and access to events (touches, timers, etc.) and services (location, camera, etc.).

### Access to native features

Some platform-specific functions like camera and geolocation require access to native features. These functions need to be implemented by means of native languages, and Flutter gives the feel of developing on the native platform. Flutter allows you to reuse your existing Java, Swift, and Objective-C code to access native features and SDKs on iOS and Android.

### Official support

Since Flutter is Google’s initiative, the framework has a big name behind it. There’s great support from the Material Design team and a community of over 1000 developers who are willing to contribute to the growth of this framework.

### Easy to set up

Flutter is easy to set up and start coding on low-end machines. You don’t need to be worried that your operating system won’t be able to handle Flutter. Quite the contrary, Flutter doesn’t require a powerful machine.

Luigi Rosso, co-founder of 2Dimensions, says the following about Flutter app development:

“I love the friction-free iteration with Flutter. Hot Reload sets me in a feedback loop that keeps me focused and in tune with my work. One of my biggest productivity inhibitors are tools that run slower than the developer. Flutter finally resets that bar.”

### Cons

The framework is relatively young, and it’s still growing and developing. The toolkit has its flaws. Let’s look at the them.

### Mobile only

Flutter caters to mobile apps only and isn’t supported by web browsers. When you choose the technology stack you want to work with, this can influence your decision. If you want maximum device flexibility, then Flutter may not be the thing you’re looking for.

### Limited libraries

Google supports Flutter and there are many helpful libraries. However, the framework is relatively new. You won’t find every functionality you’re looking for. Native apps, on the contrary, have everything in their native SDKs including Stripe, Twilio, and most streaming services such as WoWza and UStream. With Flutter, developers would have to build these libraries themselves, which is time-consuming.

### Continuous Integration

There are many ready-made solutions for native Android and iOS apps that allow them to work with CI platforms like Travis and Jenkins. There are also ready-made Flutter solutions for most popular CI platforms like Travis or Circle. However, you need to set up the toolkit for these kinds of platforms. Travis can be used in [a number of repos](https://github.com/yjbanov/flutter_travis_sample). The framework also supports the [likes of Fastlane](https://flutter.io/fastlane-cd/) and [Nevercode](https://nevercode.io/blog/we-are-ready-for-flutter-build-test-and-distribute-flutter-apps-with-nevercode/). As for us, we’ve applied a complex decision using Gitlab CI with Fastlane for Flutter apps.

### Limited TV support

If you’re wondering if apps built with the help of this framework can run on Android TV or Apple TV, then the answer is more no than yes. Flutter offers practically no support for Android TV and Apple TV.

<https://hackernoon.com/flutter-pros-and-cons-for-seamless-cross-platform-development-c81bde5a4083>