CrossPlatform - Flutter de Google





Chengjie ZHANG - MPL6

Table de matière

[**A. Flutter- Google UI toolkit framework**](#_lgstkgsz514m) **3**

[Introduction and problematic](#_rgsvosgz4dt0) 3

[Advantages of Flutter](#_ibvqz6j55p9x) 3

[Architecture of Flutter](#_jsx9t6jgb3uy) 3

[**B. CrossPlateform**](#_7lal867ixcn1) **5**

[Tools for developing command-line apps and servers](#_sstda2usm2ja) 5

[Tools for developing web apps](#_1u26qfb5tlrg) 5

[Command-line tools](#_fxnwkxekjogn) 6

[**C. Fast development**](#_sxa8mqmien30) **7**

[**D. Expressive + Flexible UI**](#_5igwmz5wzdu1) **9**

[Everything’s a widget in flutter](#_5hv90qjbu8f) 9

[Existing UI theme: Material & Cupertino](#_oam26cfrr3bi) 10

[**E. Bibliography**](#_xfxmavl5bzfd) **11**

# 

# A. Flutter- Google UI toolkit framework

## Introduction and problematic

Flutter is an app SDK for building high-performance, high-fidelity apps for iOS, Android, and web from a single codebase.

The goal is to enable developers to deliver high-performance apps that feel natural on different platforms. We embrace differences in scrolling behaviors, typography, icons, and more.

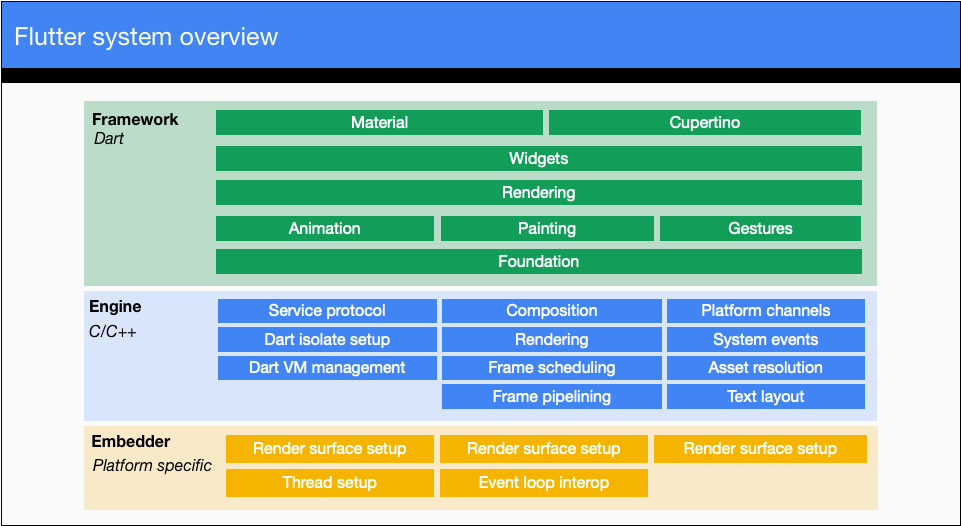
Flutter includes a modern react-style framework, a 2D rendering engine, ready-made widgets, and development tools. These components work together to help you design, build, test, and debug apps. Everything is organized around a few core principles.

## Advantages of Flutter

* Be highly productive
  + Develop for iOS and Android from a single codebase
  + Do more with less code, even on a single OS, with a modern, expressive language and a declarative approach
  + Prototype and iterate easily
    - Experiment by changing code and reloading as your app runs (with hot reload)
    - Fix crashes and continue debugging from where the app left off
* Create beautiful, highly-customized user experiences
  + Benefit from a rich set of Material Design and Cupertino (iOS-flavor) widgets built using Flutter’s own framework
  + Realize custom, beautiful, brand-driven designs, without the limitations of OEM widget sets

## Architecture of Flutter

The Flutter framework is organized into a series of layers (Framework, Engine and Embedder), with each layer building upon the previous layer.



The upper layers of the framework are used more frequently than the lower layers. For the complete set of libraries that make up the Flutter’s layered framework, see [API documentation](https://api.flutter.dev/).

The goal of this design is to help you do more with less code. For example, the Material layer is built by composing basic widgets from the widgets layer, and the widgets layer itself is built by orchestrating lower-level objects from the rendering layer.

The layers offer many options for building apps. Choose a customized approach to unlock the full expressive power of the framework, or use building blocks from the widgets layer, or mix and match. You can compose the ready-made widgets Flutter provides, or create your own custom widgets using the same tools and techniques that the Flutter team used to build the framework.

Nothing is hidden from you. You reap the productivity benefits of a high-level, unified widget concept, without sacrificing the ability to dive as deeply as you wish into the lower layers

# 

# B. CrossPlateform

write once, run everywhere

One single codebase for mobile, web and desktop

The Dart SDK has the libraries and command-line tools that you need to develop Dart web, command-line, and server apps. If just for developing only mobile apps, you don’t need the Dart SDK; just install flutter.

## Tools for developing command-line apps and servers

The following tools support developing or running command-line apps and servers:

* [dart](https://github.com/dart-lang/site-www/blob/master/tools/dart-vm) : The standalone Dart VM, which you can use to execute Dart code. IDEs that support Dart, and some of the pub commands, use this command behind-the-scenes to execute Dart scripts. Note that you must configure your IDE with the location of the dart binary.
* [dart2native](https://github.com/dart-lang/site-www/blob/master/tools/dart2native) : An ahead-of-time (AOT) compiler that compiles Dart code to native x64 machine code. The output is either a standalone executable (the default) or an AOT snapshot.
* [dartaotruntime](https://github.com/dart-lang/site-www/blob/master/tools/dartaotruntime) : A Dart runtime that you can use to run AOT snapshots.

## Tools for developing web apps

The following tools support developing web apps:

* [dart2js](https://github.com/dart-lang/site-www/blob/master/tools/dart2js) : The original Dart-to-JavaScript compiler, with tree shaking. IDEs and the webdev CLI use dart2js when building web apps for deployment.
* [dartdevc](https://github.com/dart-lang/site-www/blob/master/tools/dartdevc) : The Dart dev compiler, a modular Dart-to-JavaScript compiler. IDEs and the webdev CLI use dartdevc when running a development server.
* [webdev](https://github.com/dart-lang/site-www/blob/master/tools/webdev) : A command-line interface (CLI) for Dart web app development, including building and serving web apps.

## Command-line tools

The Dart SDK includes the following general-purpose tools:

* [dartanalyzer](https://github.com/dart-lang/site-www/blob/master/tools/dartanalyzer) : A static analyzer that evaluates and reports any errors or warnings in your code. The Dart plugin for your IDE should make use of Dart's analysis engine, but you can also run the analyzer from the command line.
* [dartdoc](https://github.com/dart-lang/site-www/blob/master/tools/dartdoc) : A documentation generator. For examples of dartdoc's output, see the API reference documentation published at [api.dart.dev](https://github.com/dart-lang/site-www/blob/master/src/tools/%7B%7Bsite.dart_api%7D%7D) and pub.dev (for example, the [path API reference](https://github.com/dart-lang/site-www/blob/master/src/tools/%7B%7Bsite.pub-api%7D%7D/path)).
* [dartfmt](https://github.com/dart-lang/site-www/blob/master/tools/dartfmt) : An opinionated code formatter that follows the recommendations of the [Dart style guide](https://github.com/dart-lang/site-www/blob/master/guides/language/effective-dart/style). IDEs that support Dart generally allow you to format the code within the IDE. Or you can run the formatter from the command line.
* [pub](https://github.com/dart-lang/site-www/blob/master/tools/pub/cmd) : A package manager that makes it easy for you to install, use, and share Dart libraries, command-line tools, and other assets. Some Dart technologies, such as Flutter, may not support all of the pub commands. IDEs that support Dart generally have special support for pub, but you can also use it from the command line.
* Some additional tools are available in [packages](https://github.com/dart-lang/site-www/blob/master/guides/packages). To install these tools, use the pub command, as described in each tool's installation instructions. Here are the general-purpose tools you might want to install:
* [build\_runner](https://github.com/dart-lang/site-www/blob/master/tools/build_runner) : A build package that's used behind-the-scenes by the webdev command.
* [dartfix](https://github.com/dart-lang/site-www/blob/master/src/tools/%7B%7Bsite.pub-pkg%7D%7D/dartfix) : A tool for migrating Dart source code and fixing common issues.

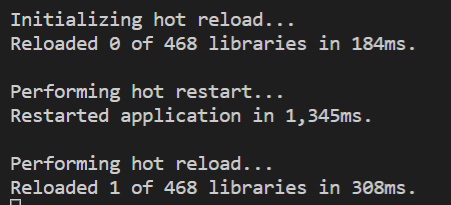
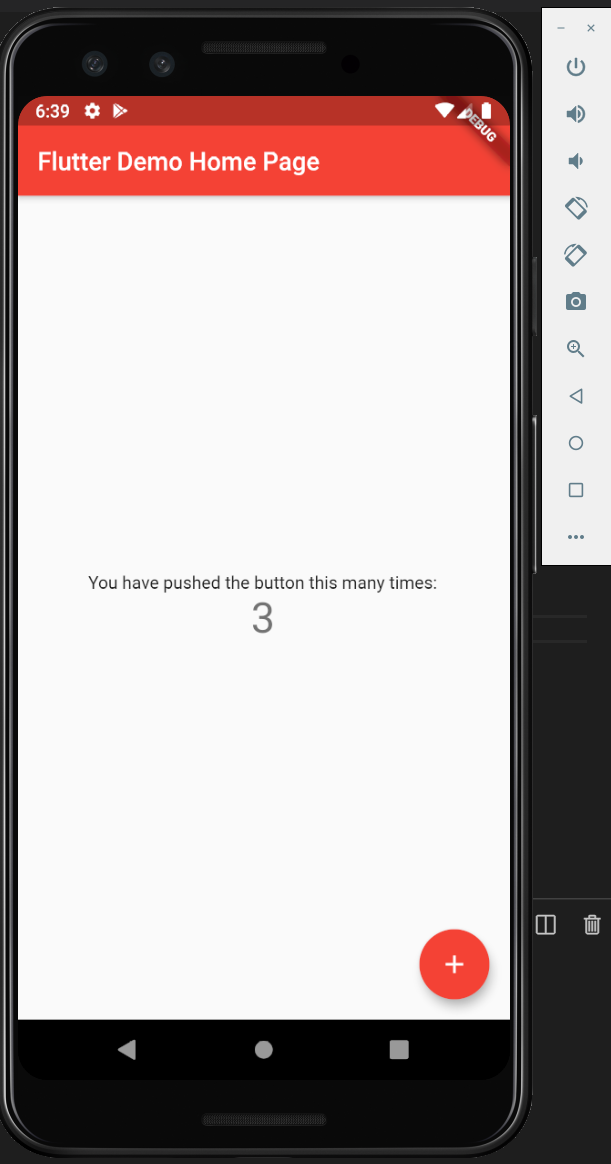
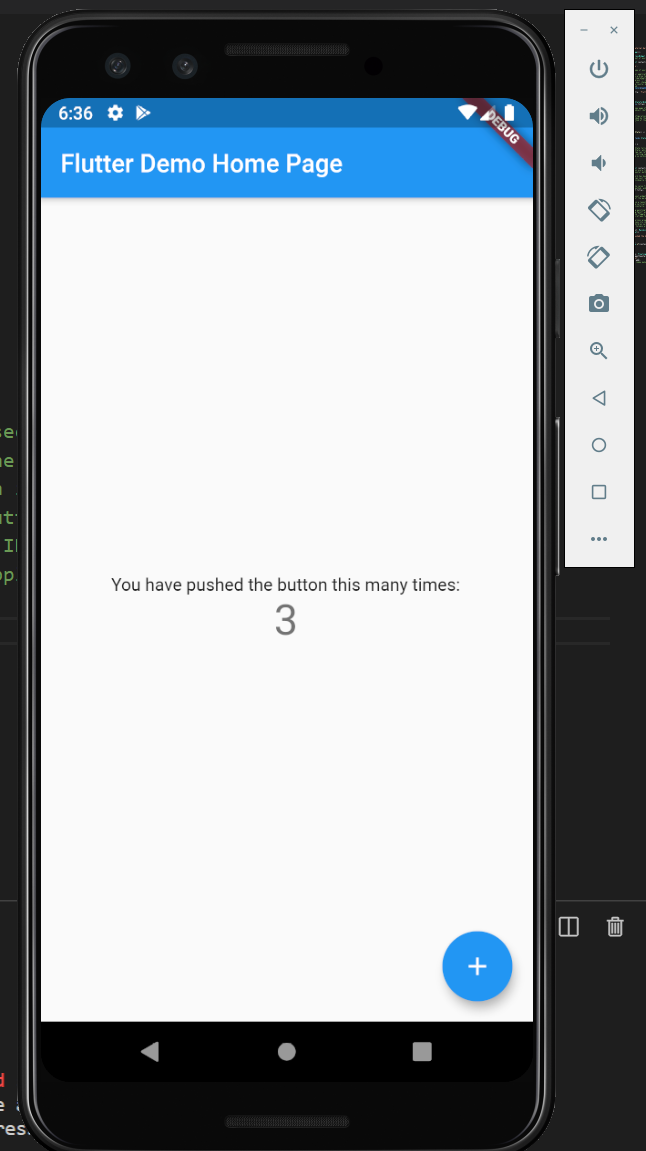
# 

# C. Fast development

## Efficiency & better quality:

Just-in-Time (JiT) and Ahead-of-Time (AoT) compilation :

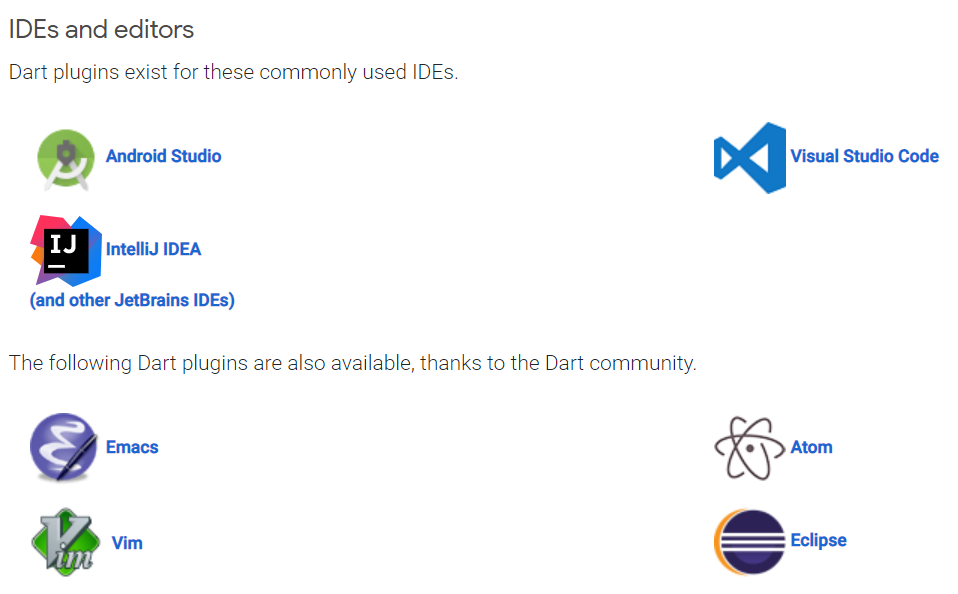
* JiT for dev Improve development efficiency: hot-reload without losing state



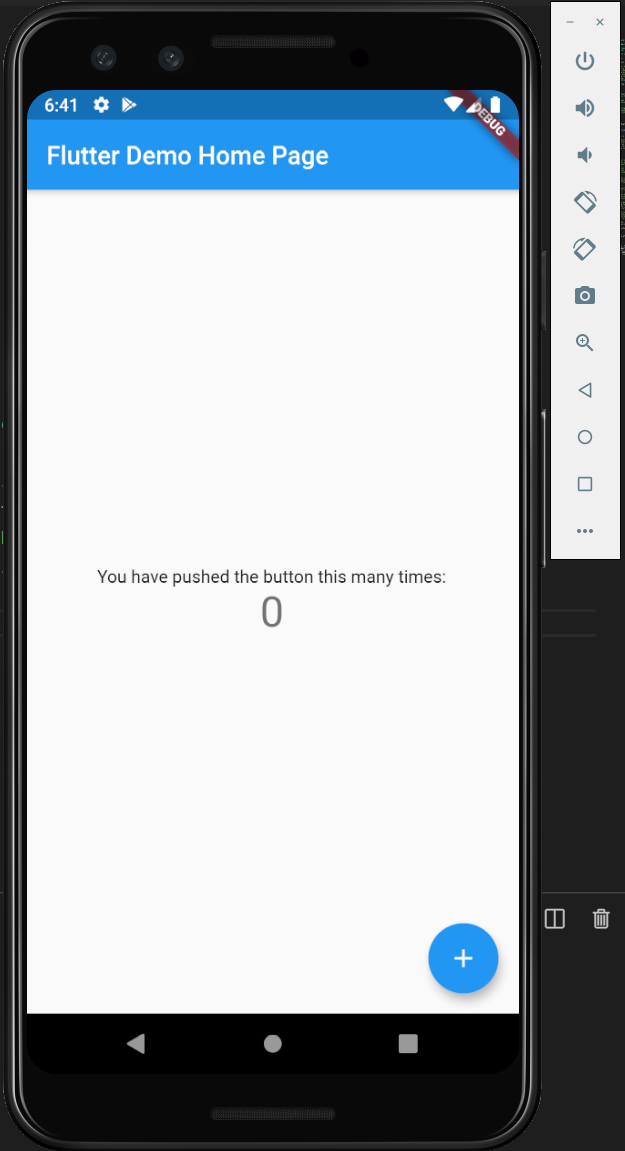
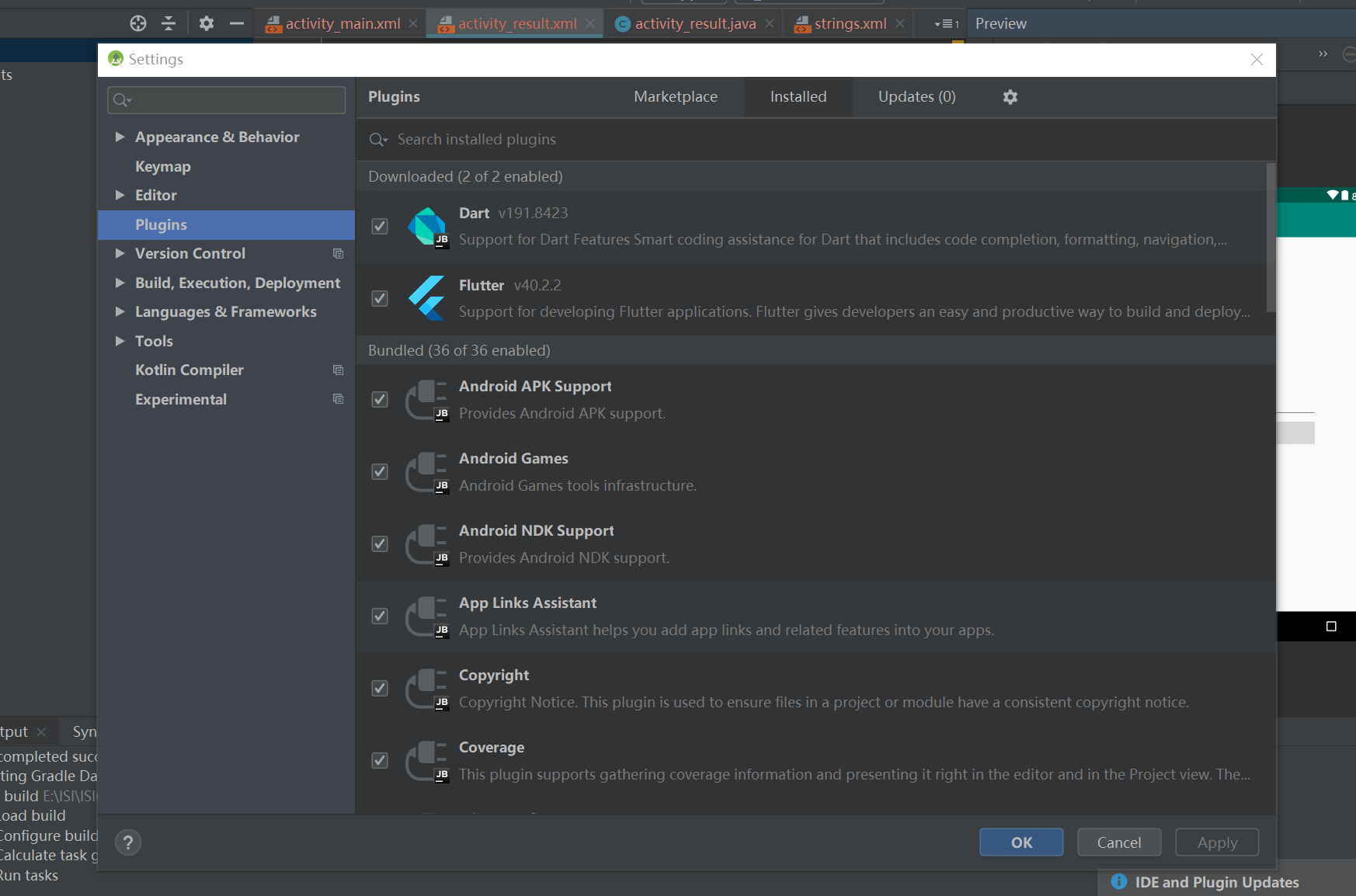
* AoT for release Improve execution efficiency: meilleure performance

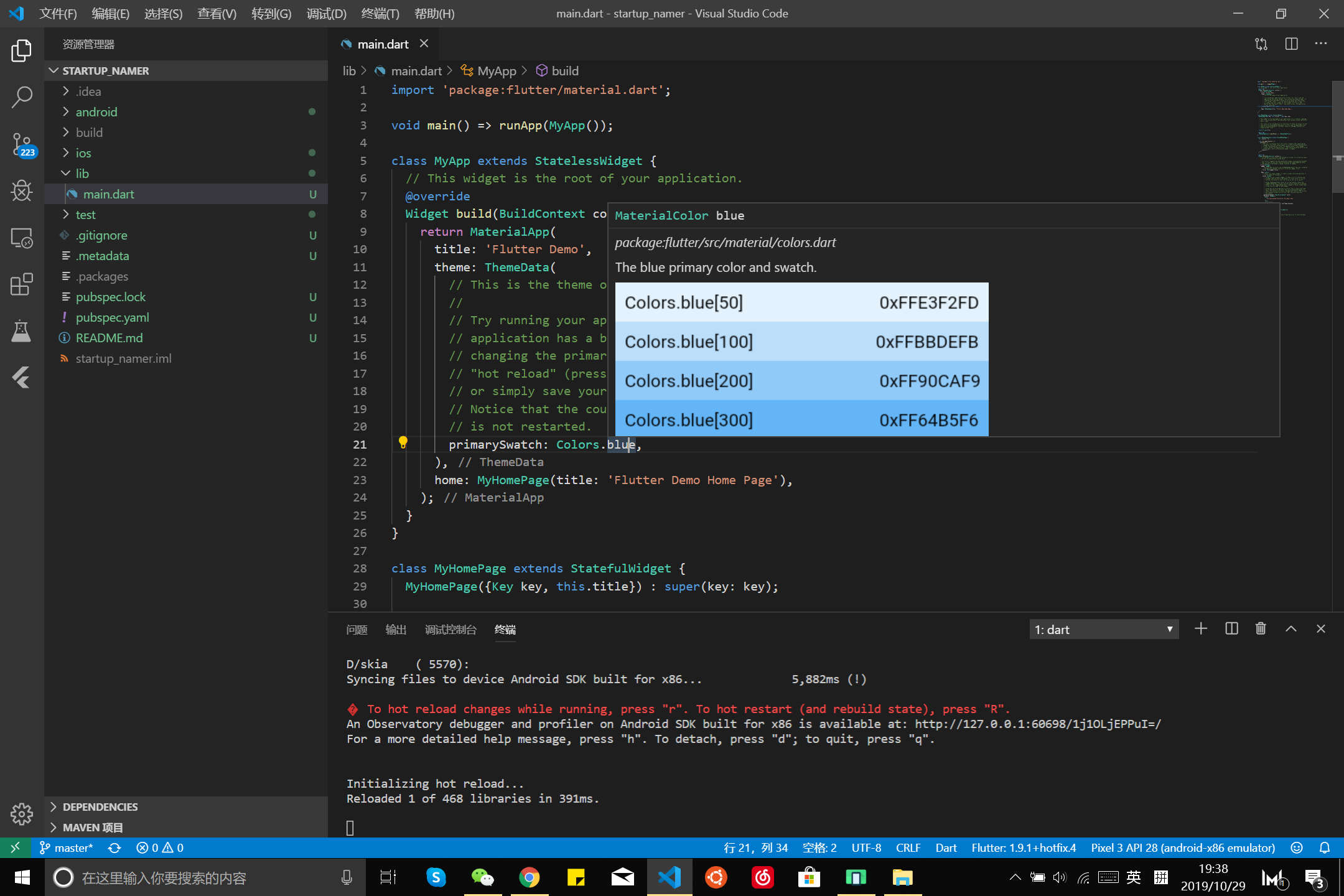
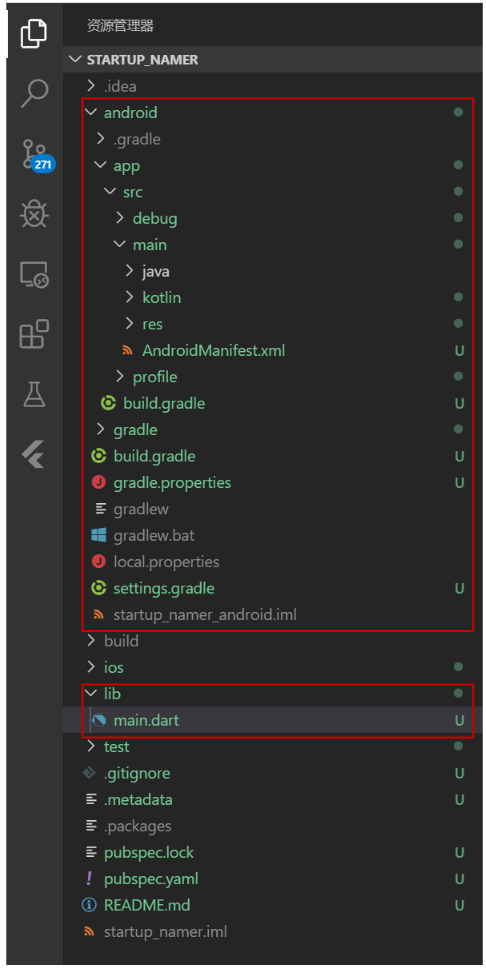
## Framework with IDE plugin

Instead of re-creating an IDE, Flutter chose to embed the popular IDEs as a plug-in. This makes it faster and easier for developers to get started.



For example, in Android Studio and VSCode





# D. Expressive + Flexible UI

## Everything’s a widget in flutter

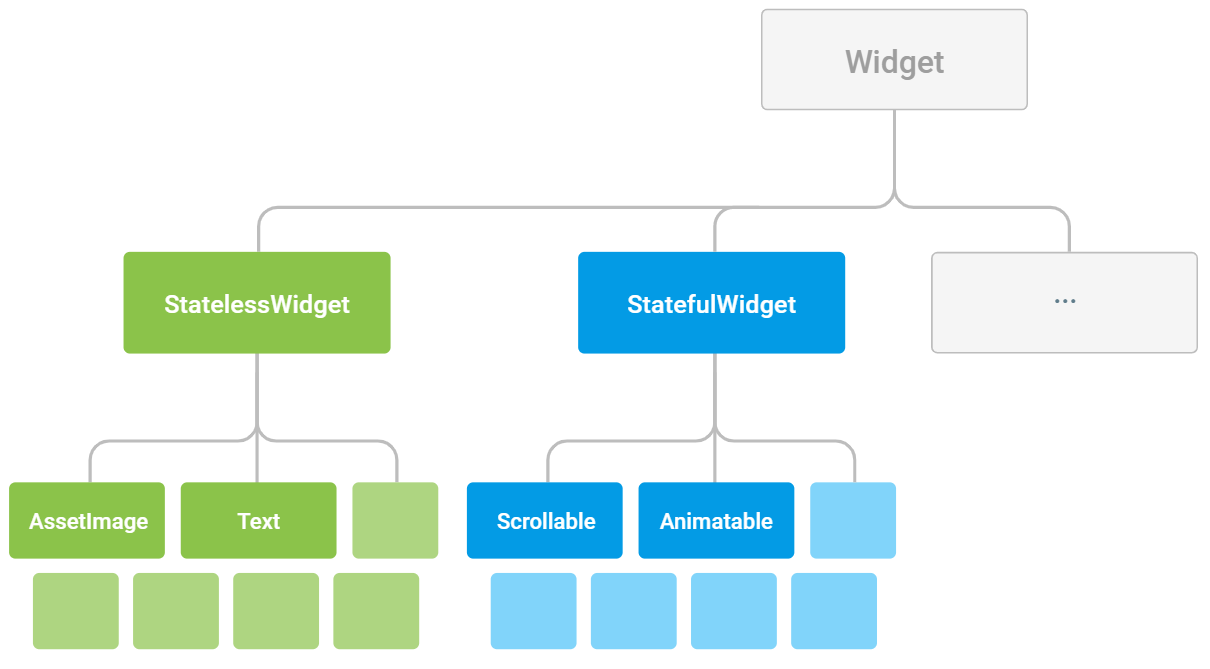
Widgets are the basic building blocks of a Flutter app’s user interface. Each widget is an immutable declaration of part of the user interface. Unlike other frameworks that separate views, view controllers, layouts, and other properties, Flutter has a consistent, unified object model: the widget.

A widget can define:

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

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

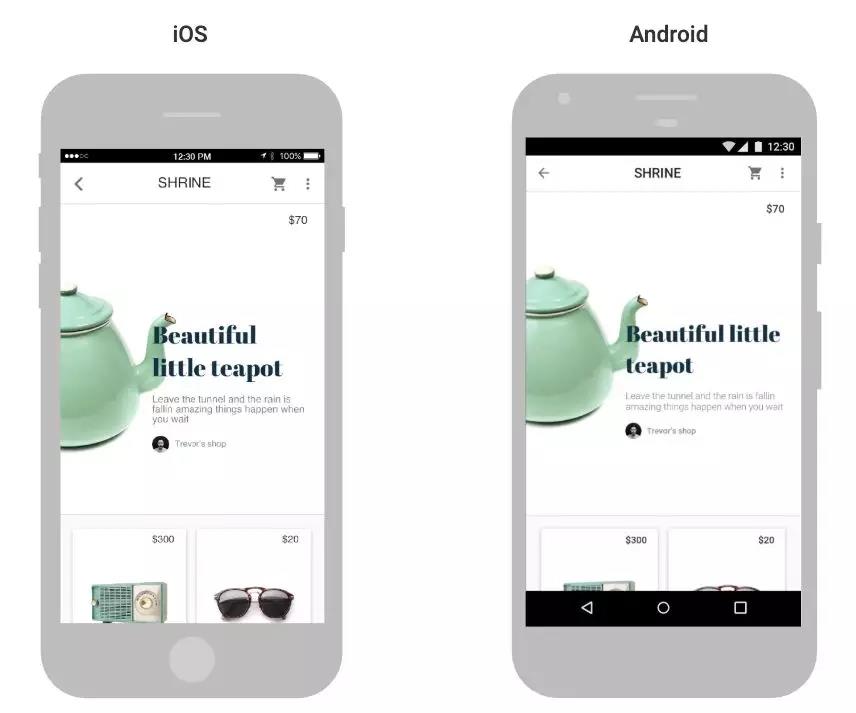
You can respond to events, like user interaction, by telling the framework to replace a widget in the hierarchy with another widget. The framework then compares the new and old widgets and efficiently updates the user interface.



## Existing UI theme: Material & Cupertino

Material Components widgets: <https://flutter.dev/docs/development/ui/widgets/material>

Cupertino (iOS-style) widgets: <https://flutter.dev/docs/development/ui/widgets/cupertino>



# 

# E. Bibliography

## Vidos

* Flutter introduction： <https://www.youtube.com/watch?v=fq4N0hgOWzU>
* What’s new in 2019： <https://www.youtube.com/watch?time_continue=177&v=5VbAwhBBHsg>
* Visual programation(Blockly+Flutter to create a list view) : <https://www.youtube.com/watch?v=JUeMqrBURtQ>
* Cupertino(IOS UI style): <https://www.youtube.com/watch?v=3PdUaidHc-E>

## Docs

* [](https://flutter.dev/docs) <https://flutter.dev/docs>
* Dart <https://dart.dev/guides>
*  <https://github.com/flutter/flutter/issues>
*  <https://stackoverflow.com/tags/flutter>

## Books

* + Beginning Flutter: A Hands On Guide to App Development

<https://www.wiley.com/en-us/Beginning+Flutter%3A+A+Hands+On+Guide+to+App+Development-p-9781119550822>

* + Flutter Succinctly

<https://www.syncfusion.com/ebooks/flutter-succinctly>