

# Flutter and Dart



# DART

# Some considerations

- Dart is the language behind Flutter
- Flutter is a framework to develop Apps
- Both are evolving quickly
- Google is really behind them
  - Well documented
  - Good packages
  - Support for most HW/OS
- Community
  - Many contributions
  - Not all are “bullet proof”
  - Not all are “~well documented”



# Dart (programming language)

From Wikipedia, the free encyclopedia

*For the advertising application formerly called Google Dart, see DoubleClick for Publishers by Google.*

**Dart** is a client-optimized<sup>[7]</sup> programming language for apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, backend and web applications.<sup>[8]</sup>

Dart is an object-oriented, class defined, garbage-collected language<sup>[9]</sup> using a C-style syntax that transcompiles optionally into JavaScript. It supports interfaces, mixins, abstract classes, reified generics, static typing, and a sound type system.<sup>[10]</sup>

| <b>Contents</b> [hide]                             |  |
|--|--|
| <a href="#">1 History</a>                          |  |
| <a href="#">2 Usage</a>                            |  |
| <a href="#">3 Isolates</a>                         |  |
| <a href="#">4 Snapshots</a>                        |  |
| <a href="#">5 Native mobile apps</a>               |  |
| <a href="#">6 Compiling to JavaScript</a>          |  |
| <a href="#">7 Editors</a>                          |  |
| <a href="#">7.1 Chrome Dev Editor</a>              |  |
| <a href="#">7.2 DartPad</a>                        |  |
| <a href="#">8 SIMD</a>                             |  |
| <a href="#">9 Example</a>                          |  |
| <a href="#">10 Influences from other languages</a> |  |
| <a href="#">11 Criticism</a>                       |  |

|   |  |
|---|--|
|  | <b>Dart</b>  |
| <b>Paradigm</b>   | Multi-paradigm: scripting, object-oriented (class-based), imperative, reflective, functional, garbage-collected <sup>[1]</sup> |
| <b>Designed by</b>  | Lars Bak and Kasper Lund   |
| <b>Developer</b>  | Google   |
| <b>First appeared</b>   | October 10, 2011; 8 years ago <sup>[2]</sup>   |
| <b>Stable release</b>   | 2.7.0 / December 11, 2019; 53 days ago <sup>[3]</sup>  |
| <b>Preview release</b>  | 2.8.0-dev.0.0 / December 11, 2019; 53 days ago <sup>[3]</sup>  |
| <b>Typing discipline</b>  | 1.x: Optional, 2.x: Static <sup>[4]</sup>  |
| <b>Platform</b>   | Cross-platform   |
| <b>OS</b>   | Cross-platform   |
| <b>License</b>  | BSD  |



# Dart (programming language)

From Wikipedia, the free encyclopedia

*For the advertising application formerly called Google Dart, see DoubleClick for Publishers by Google.*

**Dart** is a client-optimized<sup>[7]</sup> programming language for apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, backend and web applications.<sup>[8]</sup>

Dart is an object-oriented, class defined, garbage-collected language<sup>[9]</sup> using a C-style syntax that transcompiles optionally into JavaScript. It supports interfaces, mixins, abstract classes, reified generics, static typing, and a sound type system.<sup>[10]</sup>

## Contents [hide]

- 1 History
- 2 Usage
- 3 Isolates
- 4 Snapshots
- 5 Native mobile apps
- 6 Compiling to JavaScript
- 7 Editors
  - 7.1 Chrome Dev Editor
  - 7.2 DartPad
- 8 SIMD
- 9 Example
- 10 Influences from other languages
- 11 Criticism

| Dart  |  |
|---|--|
|  | Multi-paradigm: scripting, object-oriented (class-based), imperative, reflective, functional, garbage-collected <sup>[1]</sup> |
| Paradigm  | Lars Bak and Kasper Lund   |
| Designed by   | Google   |
| Developer   | October 10, 2011; 8 years ago <sup>[2]</sup>   |
| First appeared  | 2.7.0 / December 11, 2019; 53 days ago <sup>[3]</sup>  |
| Stable release  | 2.8.0-dev.0.0 / December 11, 2019; 53 days ago <sup>[3]</sup>  |
| Preview release   | 2.8.0-dev.0.0 / December 11, 2019; 53 days ago <sup>[3]</sup>  |
| Typing discipline   | 1.x: Optional, 2.x: Static <sup>[4]</sup>  |
| Platform  | Cross-platform   |
| OS  | Cross-platform   |
| License   | BSD  |

# Dart: The Language behind Flutter and Fuchsia OS.



JAY TILLU

Follow

Oct 4, 2019 · 4 min read

Nice introductory posts,  
Not scientific but a good scope



Flutter 1.0 is officially announced on Dec 04, 2018. After that, the demand

Dart: The Language behind Flutter and Fuchsia OS.

<https://medium.com/@jaytillu/dart-the-language-behind-flutter-and-fuchsia-os-f48de133c97a>

oct2019

# Google Fuchsia

From Wikipedia, the free encyclopedia

**Fuchsia** is an open-source capability-based operating system currently being developed by Google. It first became known to the public when the project appeared on a self hosted form of git in August 2016 without any official announcement. The name means "Pink + Purple == Fuchsia (a new Operating System)",<sup>[2]</sup> which is a reference to Pink (Apple's first effort at an object-oriented, microkernel-based operating system), and Purple (the original iPhone's codename).<sup>[3]</sup> In contrast to prior Google-developed operating systems such as Chrome OS and Android, which are based on the Linux kernel, Fuchsia is based on a new kernel called Zircon.

## Contents [hide]

- 1 History
- 2 Overview
  - 2.1 Kernel
- 3 References
- 4 External links

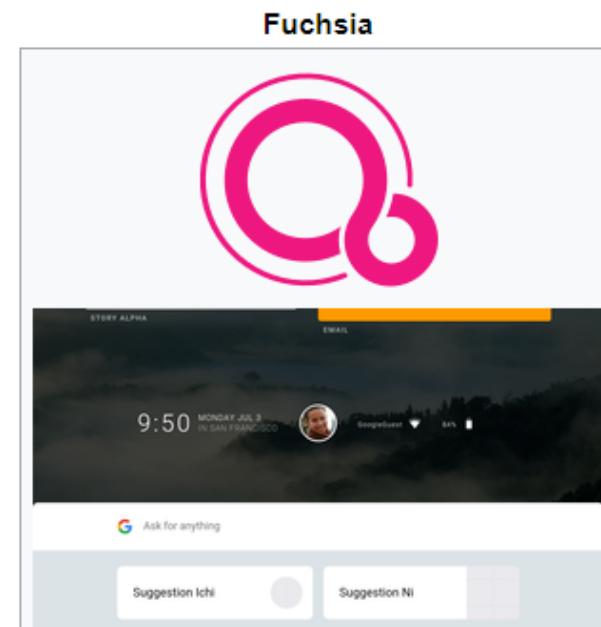
## History [ edit ]

In August 2016, media outlets reported on a mysterious codebase post published on GitHub, revealing that Google was developing a new operating system called "Fuchsia". No official announcement was made, but inspection of the code suggested its capability to run on universal devices, including "dash infotainment systems for cars, to embedded devices like traffic lights and digital watches, all the way up to smartphones, tablets and PCs". The code differs from Android and Chrome OS due to its being based on the Zircon kernel (formerly called Magenta)<sup>[4]</sup> rather than on the Linux kernel.<sup>[5][6][7][8][9]</sup>

In May 2017, Ars Technica wrote about Fuchsia's new user interface, an upgrade from its command-line interface at its first reveal in August, along with a developer writing that Fuchsia "isn't a toy thing, it's not a 20% Project, it's not a dumping ground of a dead

[https://en.wikipedia.org/wiki/Google\\_Fuchsia](https://en.wikipedia.org/wiki/Google_Fuchsia)

project's seemingly close ties to Android, with some speculating that Fuchsia might be



Screenshot of the Google Fuchsia GUI

|                        |   |
|------------------------|---|
| Developer              | Google  |
| Written in             | C, C++, Dart, Go, Rust, Python <sup>[1]</sup> |
| Working state          | Current                                       |
| Source model           | Open source                                   |
| Initial release        | August 15, 2016; 4 years ago                  |
| Repository             | fuchsia.googlesource.com                      |
| Available in           | English                                       |
| Platforms              | ARM64, x86-64                                 |
| Influenced by          | Android                                       |
| Default user interface | Ermine  |
| License                | BSD, MIT, Apache License 2.0                  |
| Official website       | fuchsia.dev                                   |

## Get started

### Overview

[Get Started with Fuchsia OS](#)

[Get source code](#)

[Configure and build Fuchsia](#)

[Explore Fuchsia](#)

[Set up a Fuchsia emulator](#)

[Set up and run FEMU](#)

Google is committed to advancing racial equity for Black communities.

Fuchsia > Get started

# Fuchsia getting started

## Contents

[Get Fuchsia source and build Fuchsia](#)

[Set up a Fuchsia device](#)

[Launch Fuchsia on an emulator](#)

[Install Fuchsia on hardware](#)

[Explore Fuchsia](#)

Welcome to Fuchsia! This guide has everything you need to get started.

**Note:** The Fuchsia source includes [Zircon](#), the core platform that underlies Fuchsia. Zircon is a full system kernel, and it is the primary focus of Fuchsia development. If you're new to Fuchsia, we recommend starting with [Getting started with Zircon](#).

## Get Fuchsia source and build Fuchsia

To download the Fuchsia source code and set up your build environment:

- [Get Fuchsia source code](#)
- [Build Fuchsia](#)

<https://fuchsia.dev/fuchsia-src/get-started>

[Set up a Fuchsia device](#)

[Overview](#)

[Hardware setup](#)

- ▶ [Set up Fuchsia hardware](#)

[Build and run](#)

- ▶ [Build](#)
- ▶ [Run](#)

[Write code](#)

- ▶ [Configure editors](#)
- ▶ [FIDL](#)
- ▶ [Modify the kernel](#)
- ▶ [Write a driver](#)
- ▶ [Develop components](#)
- ▶ [Inspect components](#)
- ▶ [Analyze problems with Triage](#)
- ▶ [Develop using the SDK](#)
- ▼ [Languages](#)
  - [Overview](#)
  - [Configure linters and formatter](#)
  - [Fuchsia endian policy](#)
  - ▶ [Support new languages](#)
  - ▶ [C/C++](#)
  - ▶ [Dart](#)
  - ▶ [Go](#)
  - ▶ [Python](#)
  - ▶ [Rust](#)
  - ▶ [Internationalization](#)



# Making Dart a Better Language for UI



Bob Nystrom

Follow

Mar 19, 2019 · 14 min read

[Twitter](#) [LinkedIn](#) [Facebook](#) [Bookmark](#) ...

```
3 Widget build(BuildContext context) {
4   return Row(
5     children: [
6       IconButton(icon: Icon(Icons.menu)),
7       if (!isAndroid) ...[
8         Expanded(child: title),
9         for (var icon in icons)
10        IconButton(icon: Icon(icon)),
11      ],
12    ],
13  );
}
```

On the [Dart](#) team, we are busy implementing a handful of language changes that I'm really excited about. They all relate to collection literals, the built-in syntax for creating lists, maps, and sets:

Making Dart a Better Language for UI

Bob Nystrom - Mar 19, 2019 · 14 min read

<https://medium.com/dartlang/making-dart-a-better-language-for-ui-f1ccaf9f546c>

3 var someSet = {1, 2, 3, 4};

# Making Dart a Better Language

C TTT

```
1 <p>I am an <strong>exciting</strong> paragraph!</p>
```

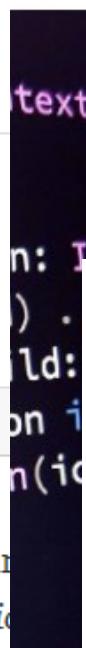
snippet03.html hosted with ❤ by GitHub



You had to write something like:

```
1 doc.beginTag("p");
2 doc.appendText("I am an ");
3 doc.beginTag("strong");
4 doc.appendText("exciting");
5 doc.endTag("strong");
6 doc.appendText("paragraph!");
7 doc.endTag("p");
```

snippet04.java hosted with ❤ by GitHub



## "UI as Code"

We called this initiative “UI as code”, since it’s about building your UI using code. But the ultimate goal is language features that are generally applicable to as many Dart programs, Flutter or not, as possible. (If you want a lot more background, here’s a [long motivation doc](#) I wrote.)

```
1 MaterialApp(
2   title: 'Welcome to Flutter',
3   home: Scaffold(
4     appBar: AppBar(title: Text('Welcome to Flutter')),
5     body: Column(
6       children: [
7         Text('Hello World'),
8         Text('This is Flutter'),
9         Text('Written in Dart')
10      ]
11    )
12  )
13)
```

snippet06.dart hosted with ❤ by GitHub

Fortunately, modern languages and APIs are designed to make this easier. Even though statements are imperative, *expressions* can be declarative. While the above code is gnarly, this one is about as readable as it gets.

a han

```
1 p("I am an ", strong("exciting"), " paragraph!")
```

snippet05.java hosted with ❤ by GitHub

MAKING DART A BETTER LANGUAGE FOR UI

Bob Nystrom - Mar 19, 2019 · 14 min read

<https://medium.com/dartlang/making-dart-a-better-language-for-ui-f1ccaf9f546c>

3 var someSet = {1, 2, 3, 4};

# Michael Thomsen

Home

Published in Dart · Aug 30

## Dart 2.18: Objective-C & Swift interop

Enhanced interoperability, platform-specific networking, improved type inference, and an important update on our null safety language roadmap — Dart 2.18 is available today. This release features a preview of...



Dart 9 min read



Published in Dart · May 11

## Dart 2.17: Productivity and integration

Language features. Productivity tools. Deeper & broader platform integration. — Today at Google I/O, we announced a new Dart SDK, version 2.17. This release builds on our core themes of leading...



Dart 8 min read



Published in Dart · Feb 3

## Dart 2.16: Improved tooling and platform handling

Supporting Flutter for Windows with package platform tagging and a new search experience for pub.dev — Today the Dart SDK version 2.16 is available. It has no new language features, but a bunch of bug fixes...



Dart 4 min read



Published in Dart · Dec 8, 2021

## Announcing Dart 2.15

Fast concurrency, constructor tear-offs, improved enums, and more — Today we're releasing version 2.15 of the Dart SDK, featuring fast concurrency with worker isolates, a new constructor tear-off language...



Dart 9 min read

<https://medium.com/@mit.mit>

Search



Michael Thomsen

5.3K Followers

Product Manager working on Dart and Flutter.  
Helping developers is my passion!

Follow



LAST SUMMER



# Michael Thomsen

Home About

Published in Dart · Feb 9

## Introducing Realm for Dart & Flutter

A case study in using Dart platform primitives to build a rich multi-platform library for Dart and Flutter apps. — We're excited to see MongoDB announce the general availability (GA) of Realm for Dart a...

Dart 4 min read



...

```
void main() {  
  final db = await RealmConfiguration.  
    schema([  
      SchemaObject('User')  
        ..addField('id', IntField().id).  
        ..addField('name', StringField()),  
      SchemaObject('Post')  
        ..addField('id', IntField().id).  
        ..addField('text', StringField()),  
      SchemaObject('Comment')  
        ..addField('id', IntField().id).  
        ..addField('text', StringField()),  
    ]).build();  
  
  final realm = await realmManager.getRealm();  
  realm.write(() {  
    realm.create('User', {  
      'name': 'Michael Thomsen',  
      'id': 1  
    });  
    realm.create('Post', {  
      'text': 'Hello world!',  
      'id': 1  
    });  
    realm.create('Comment', {  
      'text': 'Great post!',  
      'id': 1  
    });  
  });  
}
```

Published in Dart · Jan 25

## Introducing Dart 3 alpha

Preview the most productive, portable, and approachable version of Dart yet. Now available for early testing and experimentation. — In December, we gave a glimpse into the future with our first blog post...

Dart 8 min read



...



Published in Dart · Dec 8, 2022

## The road to Dart 3: A fully sound, null safe language

Preparing for the next major release, where Dart only supports sound null safety — Over the last four years, we've evolved Dart into a fast,...

Dart 6 min read



...



Published in Dart · Oct 6, 2022

## Partnering with GitHub on supply chain security for Dart packages

Starting today, GitHub supports Dart in its Advisory Database,



Michael Thomsen

6K Followers

Product Manager working on Dart and Flutter.  
Helping developers is my passion!

Follow



# What's new



## Contents

January 25, 2023: 2.19 + 3.0 alpha releases

August 30, 2022: 2.18 release

May 11, 2022: 2.17 release

February 3, 2022: 2.16 release

December 8, 2021: 2.15 release

September 8, 2021: 2.14 release

May 19, 2021: 2.13 release

March 3, 2021: 2.12 release

October 1, 2020: 2.10 release

This page describes what's new on the Dart website and blog. To see what's new in Flutter, visit the [Flutter what's new page](#).

For a list of Dart language changes in each Dart SDK, see the [language evolution page](#). To stay on top of announcements, including breaking changes, join the [Dart announcements Google group](#) and follow the [Dart blog](#).

## January 25, 2023: 2.19 + 3.0 alpha releases

This section lists notable changes made from August 31, 2022, through January 25, 2023. For details about the 2.19 + 3.0 alpha releases, see [Introducing Dart 3 alpha](#), and the [SDK changelog](#).

### Docs updated or added to dart.dev

In addition to bug fixes and incremental improvements, we made the following changes to this site:

- Introduced the [Fetch data from the internet](#) tutorial about using `package:http`.
- Added a page on [Automated publishing of packages to pub.dev](#).
- Included links to two new site translations in our [community resources](#) section:
  - [Korean version of this site \(한국어\)](#)
  - [Traditional Chinese version of this site \(正體中文版\)](#)
- Updated null safety content in preparation of Dart 3:
  - Changed the version constraints in the [migration guide](#) for Dart 3 compatibility.
  - Added Dart 3 full sound null safety overview to the [Sound null safety](#) page.
  - Emphasized Dart 3's incompatibility with [unsound null safety](#) in a note.
- Introduced the new [Learning Dart as a Swift developer](#) guide.
- Replaced an Effective Dart section with more general guidance on [booleans and equality operators](#).
- Documented [content-hashing](#) across the pub docs.
- Began effort to overhaul the [Zones](#) page by changing examples to use `runZonedGuarded` instead of `onError`.
- Updated content on libraries to cover new no-name declarations development:
  - Effective Dart: [Documentation](#), [Style](#), and [Usage](#)
  - New library directive section in [The language tour](#)
- Improved clarity surrounding Dart's single-threaded or multi-threaded status:
  - Removed the outdated [dart:io](#) page.
  - Added two sections to the FAQ:
    - [Is Dart single-threaded?](#)
    - [Is Dart single-threaded on the web?](#)
  - Expanded on [Dart's web concurrency capabilities](#).
- Rearranged and clarified [discussion](#) of default values for optional and positional parameters.
- Updated [Concurrency in Dart](#) to defunct.
- Documented specifying a file path w/<https://dart.dev/guides/whats-new>
- Rewrote [Learning Dart as a JavaScript developer](#).
- Added a brief overview of Dart DevTools to [dart run](#) page.
- Provided more clarity around [operator precedence and associativity](#) in the Language tour.
- Expanded Library tour section on [Building URLs](#) with URI http and factory constructor info.
- Accounted for [pub's transition to pub.dev](#) from pub.dartlang.org.
- Added documentation on [package screenshots](#).
- Improved the [explicit downcast](#) section of The Dart type system page.



Tutorials &  
codelabs



Language



Effective Dart



Core libraries



Packages



Development



Tools &  
techniques



Resources



FAQ

[Language evolution](#)

[Language specification](#)

Coming from ...



Books

Videos

Related sites



[API reference](#)

[Blog](#)

[DartPad \(online editor\)](#)

[Flutter](#)

[Package site](#)

# Dart language evolution



Contents

Changes in each release



Dart 2.19

Dart 2.18

Dart 2.17

Dart 2.16

Dart 2.15

Dart 2.14

Dart 2.13

Dart 2.12

Dart 2.10

Dart 2.9

Dart 2.8

Dart 2.7

Dart 2.6

Dart 2.5

Dart 2.4

Dart 2.3

Dart 2.2

Dart 2.1

Dart 2.0

Language versioning

Language version numbers

Per-library language version  
selection

This page lists notable changes and additions to the Dart programming language.

- To learn specific details about the most recent supported language version, check out the [language documentation](#) or the [language specification](#).
- For a full history of changes to the Dart SDK, see the [SDK changelog](#).

To use a language feature introduced after 2.0, set an [SDK constraint](#) no lower than the release when Dart first supported that feature.

**For example:** To use null safety, introduced in 2.12, set `2.12.0` as the lower constraint in the `pubspec.yaml` file.

```
environment:  
  sdk: '>=2.12.0 <3.0.0'
```

**Tip:** To review the features being discussed, investigated, and added to the Dart language, check out the [language funnel](#) tracker on the Dart language GitHub repo.

## Changes in each release

### Dart 2.19

*Released 25 January 2023*

Dart 2.19 introduced some precautions surrounding type inference. These include:

- More flow analysis flags for unreachable code cases.
- No longer delegate inaccessible private names to `noSuchMethod`.
- Top-level type inference throws on cyclic dependencies.

Dart 2.19 also introduced support for unnamed libraries. Library directives, used for appending library-level doc comments and annotations, can now be written without a name:

```
/// A really great test library.  
@TestOn('browser')  
library;
```

### Dart 2.18

*Released 30 August 2022 | Dart 2.18 announcement*



Develop

JIT + VM

Deploy

AOT + runtime

Native  
x64/ARM

dart2js

Web  
JavaScript

<https://dart.dev/platforms>

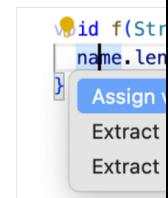


# Dart

A client-optimized language for fast apps on any platform

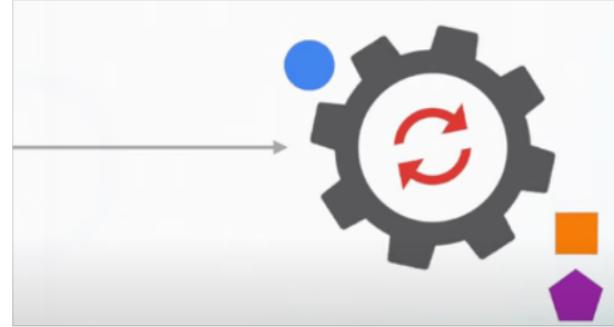
[ANNOUNCEMENTS](#)[ARCHIVE](#)[DART.DEV](#)

Follow



**Quick assists**  
Using localized reproductivity

B Brian Wilkerson  
Jul 13 · 4 min read



## Dart asynchronous programming: Streams

This article covers one of the fundamentals of reactive programming: streams, which are objects of type Stream.

Flutter Devrel Medium  
Apr 14 · 9 min read

## 18: Objective-C & Interop

Interoperability working, import and an import by language.

Thomsen  
9 min read

.dart ×

lib > dart

import 'da

import 'a

import 'b

.....

lication o

Vilkerson

6 min read



## Gradual null safety migration for large Dart projects

Dart null-safety migration is a 1-2 hour effort for a simple, small package but can be a months-long marathon for a large project. Ideally...

Polina C  
Mar 31 · 5 min read



Dart 2.16 is here, with improved tooling and platform handling. [Learn more](#)

Samples &  
tutorials

Language samples

Codelabs

List of Dart  
codelabs

Language  
cheatsheet

Iterable collections

[Asynchronous  
programming](#)

Null safety

Tutorials

Language

Core libraries

Packages

Development

# Asynchronous programming: futures, `async`, `await`

## Contents

[Why asynchronous code matters](#)

[Example: Incorrectly using an asynchronous function](#)

[What is a future?](#)

This codelab teaches you how to write asynchronous code using futures and the `async` and `await` keywords. Using embedded DartPad editors, you can test your knowledge by running example code and completing exercises.

To get the most out of this codelab, you should have the following:

- Knowledge of [basic Dart syntax](#).
- Some experience writing asynchronous code in another language.

This codelab covers <https://dart.dev/codelabs/async-await>

- How and when to use the `async` and `await` keywords.

Dart 2.16 is here, with improved tooling and platform handling. [Learn more](#)

Samples &  
tutorials

Language

Core libraries

Packages

Development

Futures, async, await

Streams

JSON

Interoperability

Google APIs

[Multi-platform apps](#)

Command-line &  
server apps

Web apps

# Multi-platform apps

We recommend the [Flutter framework](#) for developing multi-platform native apps for mobile (iOS & Android), desktop (Windows, Linux, and macOS), and the web.

Flutter is powered by the [Dart platform](#). The [Dart VM](#) provides an instant *hot reload* developer cycle. The Dart compilers – an [ahead-of-time \(AOT\) compiler for native code](#) and a [Dart-to-JavaScript compiler](#) for web code – create fast production code for any platform.



 [Get started](#)

<https://dart.dev/multiplatform-apps>

Dart 2.16 is here, with improved tooling and platform handling. [Learn more](#)

Samples &  
tutorials

Language

Core libraries

Packages

Development

Futures, async, await

Streams

JSON

Interoperability

C interop

JavaScript interop

Google APIs

Multi-platform apps

Command-line &

# C interop using dart:ffi



## Contents

[Examples](#)

[Walkthrough of hello\\_world](#)

[Files](#)

[Building and running](#)

[Using dart:ffi](#)

[Bundling and loading C libraries](#)

[Generating FFI bindings with package:ffigen](#)

Dart mobile, command-line, and server apps running on the [Dart Native platform](#) can use the `dart:ffi` library to call native C APIs, and to read, write, allocate, and deallocate native memory. *FFI* stands for *foreign function interface*.<sup>1</sup> Other terms for similar functionality include *native interface* and *language bindings*.

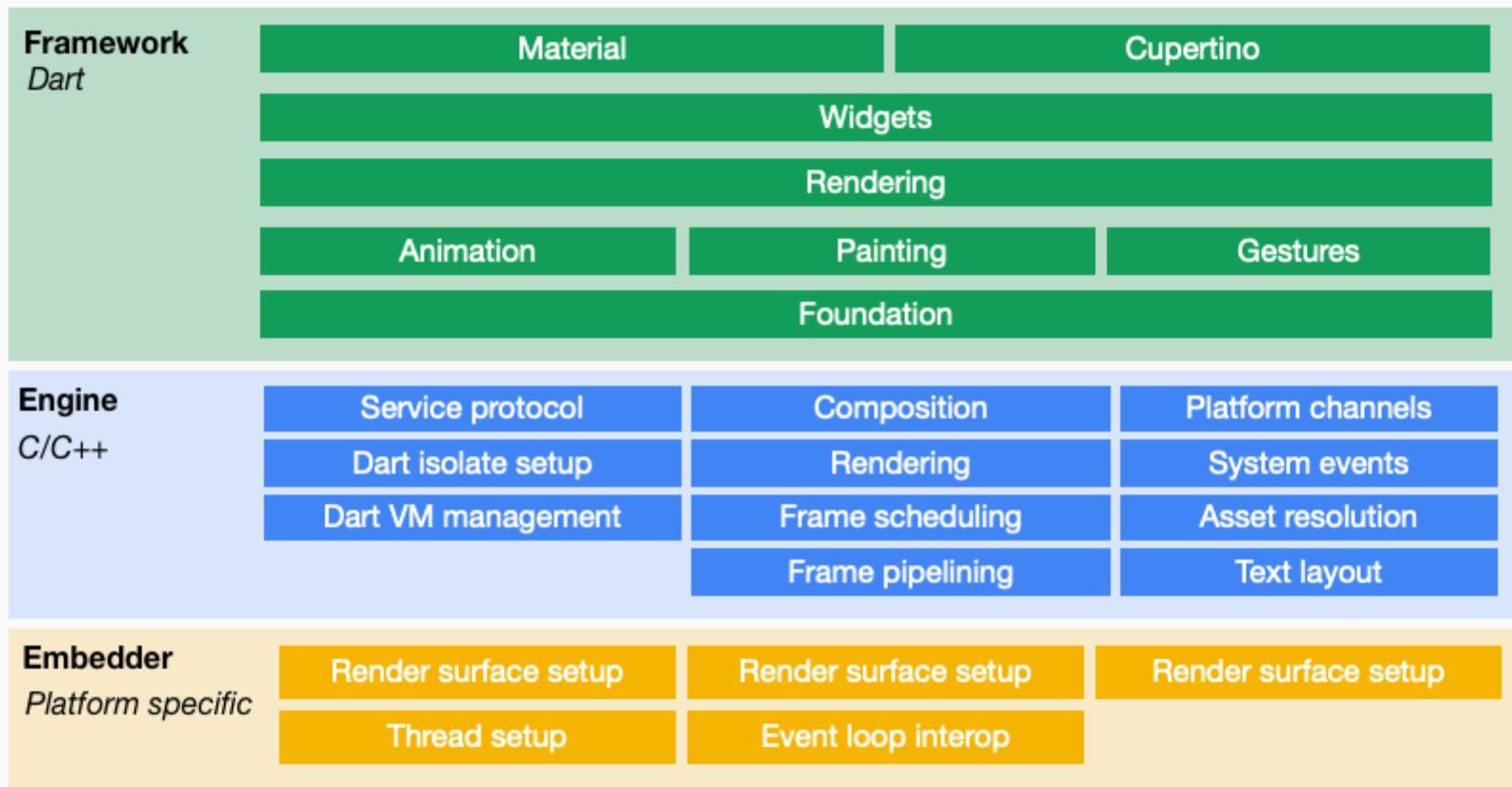
API documentation is available in the [dart:ffi API reference](#).<sup>2</sup>

## Examples

<https://dart.dev/guides/libraries/c-interop>

# How?

## Flutter system overview



Dart 2.7 is now available, with support for [extension methods](#) and more. [Read all about it.](#)

## Samples &amp; tutorials ▾

## Language ▾

## Effective Dart ▾

## Extension methods

## Specification

[Tour](#)

## Type system

## Core libraries ▾

## Packages ▾

## Development ▾

## Tools &amp; techniques ▾

## Resources ▾

## Related sites ▾

## API reference

## Blog

## DartPad (online editor)

## Flutter

## Package site

# A tour of the Dart language



## Contents

[A basic Dart program](#)[Important concepts](#)[Keywords](#)[Variables](#)[Default value](#)[Final and const](#)[Built-in types](#)[Numbers](#)[Strings](#)[Booleans](#)[Lists](#)[Sets](#)[Maps](#)[Runes and grapheme clusters](#)[Symbols](#)[Functions](#)[Optional parameters](#)[The main\(\) function](#)[Functions as first-class objects](#)[Anonymous functions](#)[Lexical scope](#)[Lexical closures](#)[Testing functions for equality](#)[Return values](#)[Operators](#)[Arithmetic operators](#)[Equality and relational operators](#)[Type test operators](#)[Assignment operators](#)[Logical operators](#)[Bitwise and shift operators](#)[Conditional](#)

This page shows you how to use each major Dart feature, from variables and operators to classes and libraries, with the assumption that you already know how to program in another language. For a briefer, less complete introduction to the language, see the [language samples page](#).

To learn more about Dart's core libraries, see the [library tour](#). Whenever you want more details about a language feature, consult the [Dart language specification](#).

**Note:** You can play with most of Dart's language features using DartPad ([learn more](#)). [Open DartPad](#)

This page uses embedded DartPads to display some of the examples. If you see empty boxes instead of DartPads, go to the [DartPad troubleshooting page](#).

## A basic Dart program

The following code uses many of Dart's most basic features:

```
// Define a function.
printInteger(int aNumber) {
    print('The number is $aNumber.');// Print to console.
}

// This is where the app starts executing.
main() {
    var number = 42; // Declare and initialize a variable.
    printInteger(number); // Call a function.
}
```

Here's what this program uses that applies to all (or almost all) Dart apps:

**// This is a comment.**

A single-line comment. Dart also supports multi-line and document comments. For details, see [Comments](#).

**int**

A type. Some of the other [built-in types](#) are `String`, `List`, and `bool`.

<https://dart.dev/guides/language/language-tour>

**print()**

A handy way to display output.

**'....' (or "....")**



Dart 2.7 is now available, with support for [extension methods](#) and more. [Read all about it.](#)

```
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(  
      title: Text(widget.title),  
    ),  
    body: Column(  
      children: <Widget>[  
        HeroImage(),  
        ...todaysDiscounts,  
        for (var d in items) ItemWidget(d),  
        ActionBar(),  
      ],  
    ),  
  );  
}
```

# Paint your UI to life

with Dart VM's  
instant **hot reload**

Build client-optimized language for fast apps on any platform  
<https://dart.dev/>

# Try Dart in your browser

The screenshot shows a DartPad interface with a dark theme. On the left, there's a code editor window titled "Dart" containing the following code:

```
1 main() {  
2   print('Hello, World!');  
3 }  
4
```

At the top center, there's a dropdown menu set to "Hello world". To the right of the code editor are three buttons: "Format", "Reset", and a blue "Run" button. Below the code editor, a vertical bar separates the code from the output area. The output area has a title "Console" and displays the text "Hello, World!". At the bottom right of the output area, it says "no issues".

Want more practice? [Try a codelab](#). Or [download Dart](#).

<https://dartpad.dev/>

[Samples & tutorials](#)[Language](#)[Core libraries](#)[Packages](#)[Development](#)[Tools & techniques](#)[Resources](#)[Related sites](#)[API reference](#)[Blog](#)[DartPad \(online editor\)](#)[Flutter](#)[Package site](#)

# Dart documentation

Welcome to the Dart documentation! Here are some of the most visited pages:

## Language samples

A brief, example-based introduction to the Dart language.

## Language tour

A more thorough (yet still example-based) introduction to the Dart language.

## Effective Dart

Best practices for building consistent, maintainable, efficient Dart code.

## Library tour

An example-based introduction to the major features in the Dart SDK's core libraries.

## Dart SDK

What's in the SDK, and how to install it.

## Futures, async, await

How to write asynchronous Dart code that uses futures and the `async` and `await` keywords.

<https://dart.dev/guides>



## Samples &amp; tutorials ▾

## Language ▾

## Effective Dart ▾

[Overview](#)[Style](#)[Documentation](#)[Usage](#)[Design](#)

## Extension methods

## Sound null safety ▾

## Evolution

## Specification

## Tour

## Type system

## Core libraries ▾

## Packages ▾

## Development ▾

## Tools &amp; techniques ▾

## Resources ▾

## Related sites ▾

[API reference](#)[Blog](#)[DartPad \(online editor\)](#)[Flutter](#)[Package site](#)

## Summary of all rules

## Style

## Identifiers

- DO name types using `UpperCamelCase`.
- DO name extensions using `UpperCamelCase`.
- DO name libraries, packages, directories, and source files using `lowercase_with_underscores`.
- DO name import prefixes using `lowercase_with_underscores`.
- DO name other identifiers using `lowerCamelCase`.
- PREFER using `lowerCamelCase` for constant names.
- DO capitalize acronyms and abbreviations longer than two letters like words.
- DON'T use a leading underscore for identifiers that aren't private.
- DON'T use prefix letters.

## Ordering

- DO place `"dart:"` imports before other imports.
- DO place `"package:"` imports before relative imports.
- DO specify exports in a separate section after all imports.
- DO sort sections alphabetically.

## Formatting

- DO format your code using `dartfmt`.
- CONSIDER changing your code to make it more formatter-friendly.
- AVOID lines longer than 80 characters.
- DO use curly braces for all flow control statements.

## Documentation

## Comments

- DO format comments like sentences.
- DON'T use block comments for documentation.

## Doc comments

- DO use `///` doc comments to document members and types.
- PREFER writing doc comments for public APIs.
- CONSIDER writing a library-level doc comment.
- CONSIDER writing doc comments for private APIs.
- DO start doc comments with a single-sentence summary.
- DO separate the first sentence of a doc comment into its own paragraph.
- AVOID redundancy with the surrounding context.
- PREFER starting function or method comments with third-person verbs.
- PREFER starting variable, getter, or setter comments with noun phrases.
- PREFER starting library or type comments with noun phrases.
- CONSIDER including code samples in doc comments.
- DO use square brackets in doc comments to refer to in-scope identifiers.
- DO use prose to explain parameters, return values, and exceptions.
- DO put doc comments before metadata annotations.

## Markdown

- AVOID using markdown excessively.
- AVOID using HTML for formatting.
- PREFER backtick fences for code blocks.

## Writing

- PREFER brevity.
- AVOID abbreviations and acronyms unless they are obvious.
- PREFER using `"this"` instead of `"the"` to refer to a member's instance.

## Usage

## Design

<https://dart.dev/guides/language/effective-dart>

- DO use strings in part of URLs.
- DON'T import libraries that are inside the `src` directory of
- DO use `libraryName`.
- AVOID abbreviations.

## Contents

[The guides](#)[How to read the guides](#)[Glossary](#)[Summary of all rules](#)[Style](#)[Documentation](#)[Usage](#)[Design](#)

Dart 2.7 is now available, with support for [extension methods](#) and more. [Read all about it.](#)

## Samples & tutorials ^

### [Language samples](#)

### Codelabs

[List of Dart codelabs](#)

[Language cheatsheet](#)

[Iterable collections](#)

[Asynchronous programming](#)

### Tutorials

### Language

### Core libraries

### Packages

### Development

### Tools & techniques

### Resources

### Related sites

### API reference

### Blog

### DartPad (online editor)

### Flutter

### Package site

# Language samples



## Contents

[Language tour](#)

[Library tour](#)

[Hello World](#)

[Variables](#)

[Control flow statements](#)

[Functions](#)

[Comments](#)

[Imports](#)

[Classes](#)

[Inheritance](#)

[Mixins](#)

[Interfaces and abstract classes](#)

[Async](#)

[Exceptions](#)

[Other topics](#)

## Language tour

A comprehensive tour, with examples, of the Dart language. Most of the [read more](#) links in this page point to the language tour.

## Library tour

An example-based introduction to the Dart core libraries. See how to use the built-in types, collections, dates and times, streams, and more.

## Hello World

Every app has a `main()` function. To display text on the console, you can use the top-level `print()` function:

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

## Variables

Even in type-safe Dart code, most variables don't need explicit types, thanks to type inference:

```
var name = 'Voyager I';  
var year = 1977;  
var antennaDiameter = 3.7;  
var flybyObjects = ['Jupiter', 'Saturn', 'Uranus', 'Neptune'];  
var image = {  
    'tags': ['saturn'],  
    'url': '//path/to/saturn.jpg'  
};
```

<https://dart.dev/samples>

including default values, the `final` and `const` keywords, and static types.

Dart 2.7 is now available, with support for [extension methods](#) and more. [Read all about it.](#)

## Samples & tutorials ^

### Language samples

### Codelabs

List of Dart codelabs

Language cheatsheet

Iterable collections

Asynchronous  
programming

### Tutorials

### Language

Core libraries

Packages

Development

Tools & techniques

Resources

Related sites

API reference

Blog

DartPad (online editor)

Flutter

Package site

# Dart cheatsheet codelab



The Dart language is designed to be easy to learn for coders coming from other languages, but it has a few unique features. This codelab – which is based on a [Dart language cheatsheet](#) written by and for Google engineers – walks you through the most important of these language features.

statements

The embedded editors in this codelab have partially completed code snippets. You can use these editors to test your knowledge by completing the code and clicking your work and res

## Iterable collections



**Note:** Th DartPad tr

This codelab teaches you how to use collections that implement the [Iterable](#) class – for example [List](#) and [Set](#). Iterables are basic building blocks for all sorts of Dart applications, and you're probably already using them, even without noticing. This codelab helps you make the most out of them.

Using the embedded DartPad editors, you can test your knowledge by running example code and completing exercises.

### String inte

To get the most out of this codelab, you should have basic knowledge of [Dart syntax](#).

### Resources

This codelab covers the following material:

- How to read elements of an Iterable.
- How to check if the elements of an Iterable satisfy a condition.
- How to filter the contents of an Iterable.
- How to map the contents of an Iterable to a different value.

### Related sites

### String

'\\${3 + 2}'  
'\\${"word"}.to  
'SmyObject'

Estimated time to complete this codelab: 60 minutes.

### Code example

The following fun  
stringify(2,

**Note:** This page uses embedded DartPads to display examples and exercises. If you see empty boxes instead of DartPads, go to the [DartPad troubleshooting page](#).

Dart

<https://dart.dev/samples>

you need collections?

Iterables are a kind of collection. A collection is an object that represents a group of objects, which are called *elements*.

4

A collection can be empty, or it can contain many elements. Depending on the purpose, collections can have different structures and implementations.



```
void main() {  
  for (int i = 0; i < 5; i++) {  
    print('hello ${i + 1}');  
  }  
}
```

▶ RUN

Console

Documentation

ⓘ DartPad tip: You can play with the code in this page by copying it into a DartPad. ↗

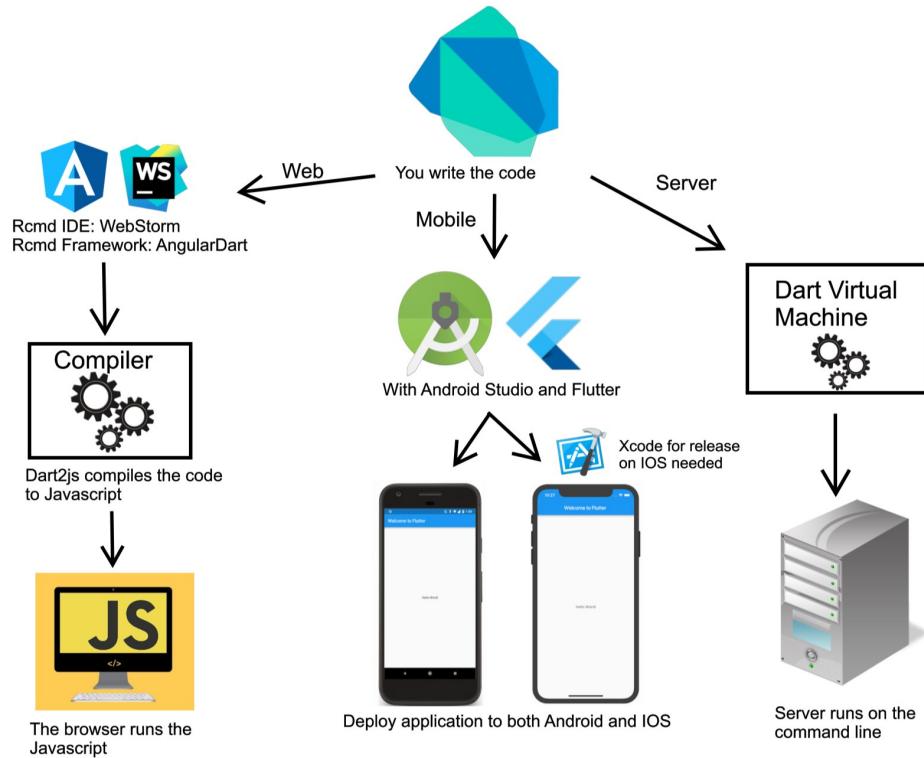
<https://dartpad.dev/>

# Disclaimer ...



- Will not be exhaustive on DART
- On a need to use / know basis
- similar to other mainstream languages – recognize flavours from java, python, javascript,...
  - types
    - Int, double, string...
  - operators
  - structures
    - List, map, set
  - Functions
  - Classes
  - ...
- Will spend time on relevant differences and mechanisms

# Flutter



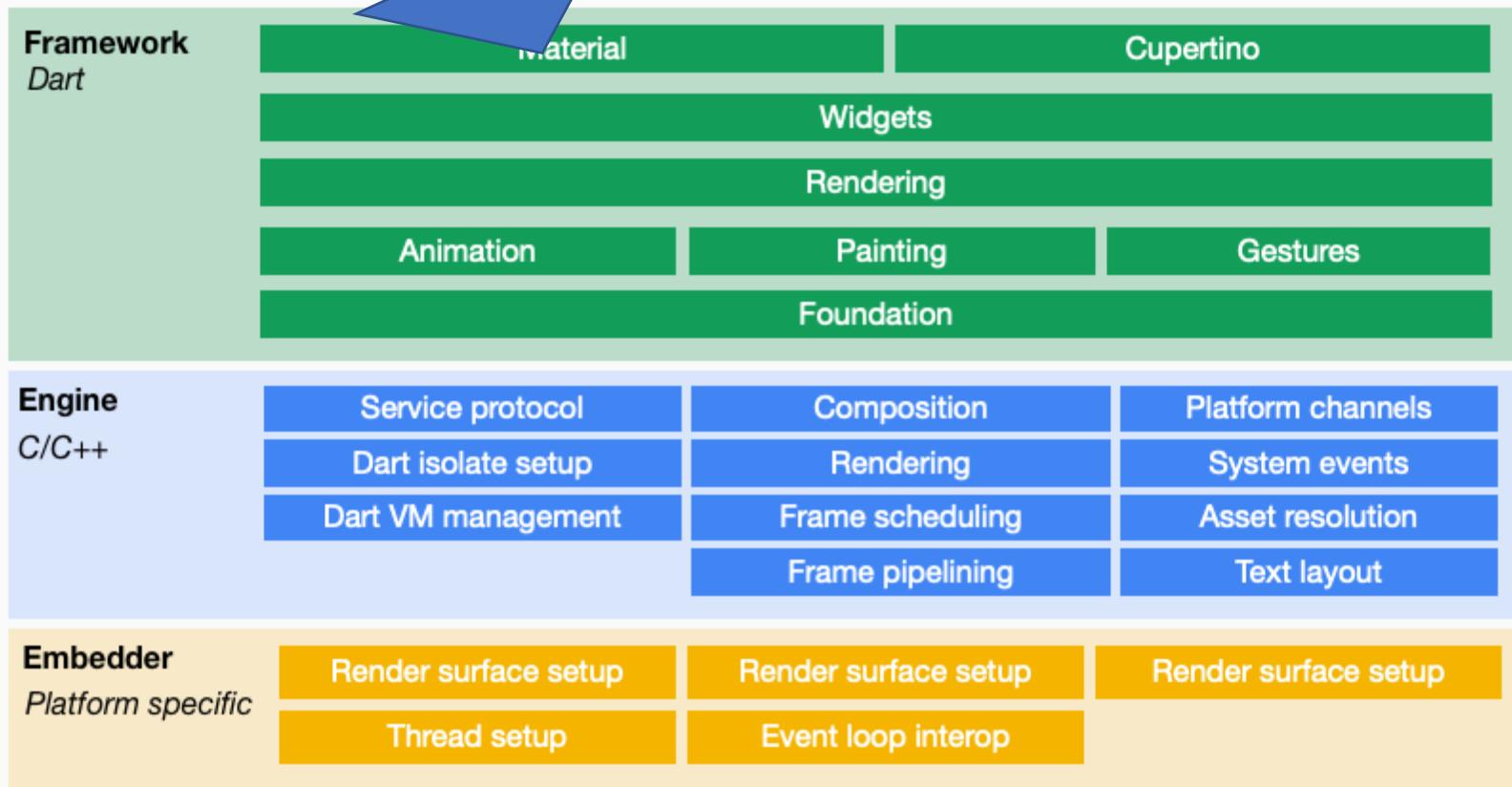
Flutter is built in C and C++ and provides a 2D rendering engine, a React-inspired functional-reactive framework, and a set of Material Design widgets. On Android, the C/C++ code is compiled with Android's NDK, while the most part of the framework and app code in Dart runs on the Dart VM, which generates JIT-compiled optimized native code. On iOS, on the other hand, Flutter's C/C++ code is compiled with LLVM, and Dart code is compiled ahead-of-time into native code.

<https://www.infoq.com/news/2015/12/flutter-dart-cross-platform/>

# How?

You program here

## Flutter system overview



# Chris Sells

Home

Published in **Flutter** · Feb 3

## What's New in Flutter 2.10

Windows stable, performance improvements, Material 3 updates and more! — I can't believe it's time again for a Flutter stable release! Hello and welcome to Flutter 2.10. It has been less than two months since ou...

Flutter 10 min read



...



...

Published in **Flutter** · Dec 9, 2021

## What's New in Flutter 2.8

Performance improvements, new Firebase features, desktop status, tooling updates and more! — It's that time of year in the northern hemisphere: the leaves are turning, the temperature is cooling and the...

Flutter 18 min read



...



Published in **Flutter** · Sep 8, 2021

## What's new in Flutter 2.5

Performance improvements, DevTools updates, new Material You support, a new app template, and more! — Hello and welcome to Flutter 2.5! This is a big release, with the 2nd highest stats in the history of...

Flutter 18 min read



...



Published in **Flutter** · May 19, 2021

## What's new in Flutter 2.2

The Flutter 2.2 release focuses on polish and optimization, including iOS performance improvements, Android deferred components, updated service worker for Flutter web and more! Today is the day we make...



<https://medium.com/@cseells> 18027

Search



Chris Sells

5K Followers

Chris Sells is a Google Product Manager on the Flutter development experience. He enjoys long walks on the beach and various technologies.

Following



### Following

Michael Thomsen

...

Supernova

...

Entronad

...

Mohit Mamoria

...

See all (11)



# Chris Sells

Home

Published in Flutter · Feb 3

## What's New in Flutter 2.10

Windows stable, performance improvements, Material 3 update more! — I can't believe it's time again for a Flutter stable release! and welcome to Flutter 2.10. It has been less than two months si

Flutter 10 min read

Published in Flutter · Dec 9, 2021

## What's New in Flutter 2.8

Performance improvements, new Firebase features, desktop styling updates and more! — It's that time of year in the northern hemisphere: the leaves are turning, the temperature is cooling a

Flutter 18 min read

Published in Flutter · Sep 8, 2021

## What's new in Flutter 2.5

Performance improvements, DevTools updates, new Material You support, a new app template, and more! — Hello and welcome to 2.5! This is a big release, with the 2nd highest stats in the history

Flutter 18 min read

Published in Flutter · May 19, 2021

## What's new in Flutter 2.2

The Flutter 2.2 release focuses on polish and optimization, including performance improvements, Android deferred components, updated service worker for Flutter web and more! Today is the day we ma

Flutter 22 min read

Search



Chris Sells

Search Bookmarks 5 J

# Flutter

Build beautiful native apps in record time

ANNOUNCEMENTS PERFORMANCE MATERIAL DESIGN WEB DEVELOPMENT FLUTTER.DEV

Flutter Following

## Wonderous: Explore the World with Flutter

A reference app that shows how to develop tailored UI using Flutter

Leigha Jarett Aug 30 · 3 min read

## Global Selection

Selectable row 1  
Selectable row 2  
Selectable row 3

## Studying developer's usage of IDEs for Flutter development

Findings from the Q2 2022 user survey.  
Written by JaYoung Lee and Ander Dobo.

JaYoung Lee Sep 12 · 5 min read

## Announcing Flutter 3.3 at Flutter Vikings

A new Flutter release, a beautiful new app, work on improving rendering performance, and a note of farewell

Tim Smoothy Aug 30 · 9 min read

## What's new

[Stay up to date](#) > What's new

This page contains current and previous announcements of what's new on the Flutter website and blog. For details about what's new in the Flutter releases see the [release notes](#) page.

To stay on top of Flutter announcements, including breaking changes, join the [flutter-announce](#) Google group.

For Dart, you can join the [Dart Announce](#) Google group, and review the [Dart changelog](#).

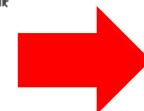
Jan 25, 2023, Flutter Forward edition: 3.7 release

Flutter 3.7 is live! This release contains many updates and improvements. This page lists the documentation changes, but you can also check out the [3.7 blog post](#) and the [3.7 release notes](#).

You might also check out [What's next for Flutter](#) and [Introducing Part 3 alpha](#).

## Docs updated or added since the 3.3 release

- You can now pass configuration information to the engine in the [initializeEngine](#) method. For more information, check out [Customizing web app initialization](#).
  - [Creating Flavors for Flutter](#) Learn how to create a flavor in Flutter (also known as a *build configuration* in iOS).
  - Internationalization support has been revamped and the [Internationalizing Flutter apps](#) page is updated.
  - The DevTools memory debugging tool has been completely overhauled and the corresponding page, [Using the memory view](#), is rewritten.
  - This release includes numerous improvements to Flutter's support for custom fragment shaders. For more information, see the new [Writing and using fragment shaders](#) page.
  - Some security tools falsely report security vulnerabilities in Flutter apps. The new [Security false positives](#) page lists the known false positives and why you can ignore them.
  - You can now invoke a platform channel from any isolate, including background isolates. For more information, check out [Writing custom platform-specific code](#) and the [Introducing isolate background channels](#) article on Medium.
  - We've updated our Swift documentation. New and updated pages include:
    - [Flutter for SwiftUI developers](#) - updated
    - [Add a Flutter screen to an iOS app](#) - updated for SwiftUI
    - [Flutter concurrency for Swift developers](#) - new
    - [Learning Dart as a Swift developer](#) on `dart dev` - new

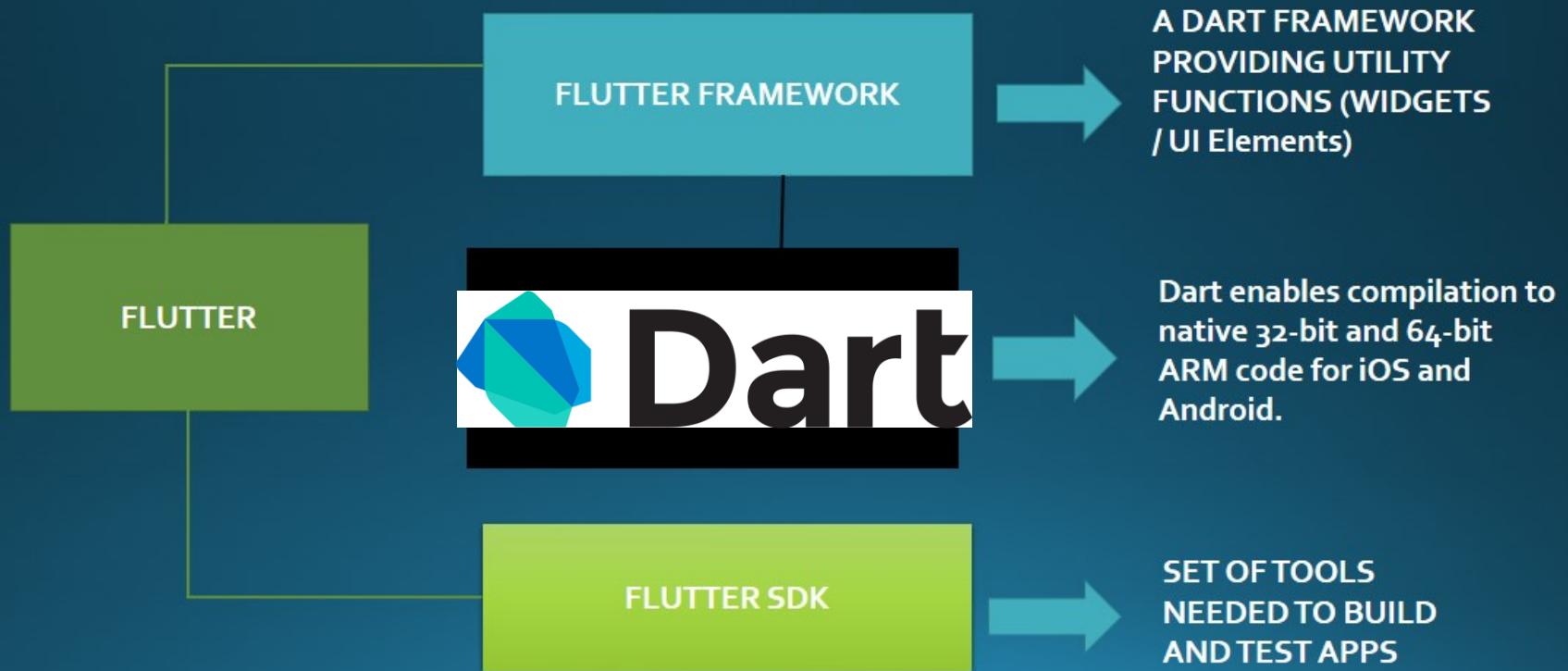


Contents

- Jan 25, 2023, Flutter Forward edition: 3.7 release
  - Aug 31, 2022, Flutter Vikings Edition: 3.3 release
  - May 11, 2022, Google I/O Edition: Flutter 3 release
  - Feb 3, 2022, Windows Support: 2.10 release
  - December 8, 2021: 2.8 release
  - September 8, 2021: 2.5 release
  - May 18, 2021, Google I/O Edition: 2.2 release
  - March 3, 2021, Flutter Engage Edition: 2.0 release
  - Oct 1, 2020: 1.22 release
  - Aug 5, 2020: 1.20 release
  - May 6, 2020, Work-From-Home Edition: 1.17 release
  - Dec 11, 2019, Flutter Interact Edition: 1.12 release
  - Sept 10, 2019: 1.9 release
  - July 9, 2019: 1.7 release
  - May 7, 2019, Google I/O Edition: 1.5 release
  - February 26, 2019: 1.2 release
  - November 5, 2018: new website

<https://docs.flutter.dev/release/whats-new>

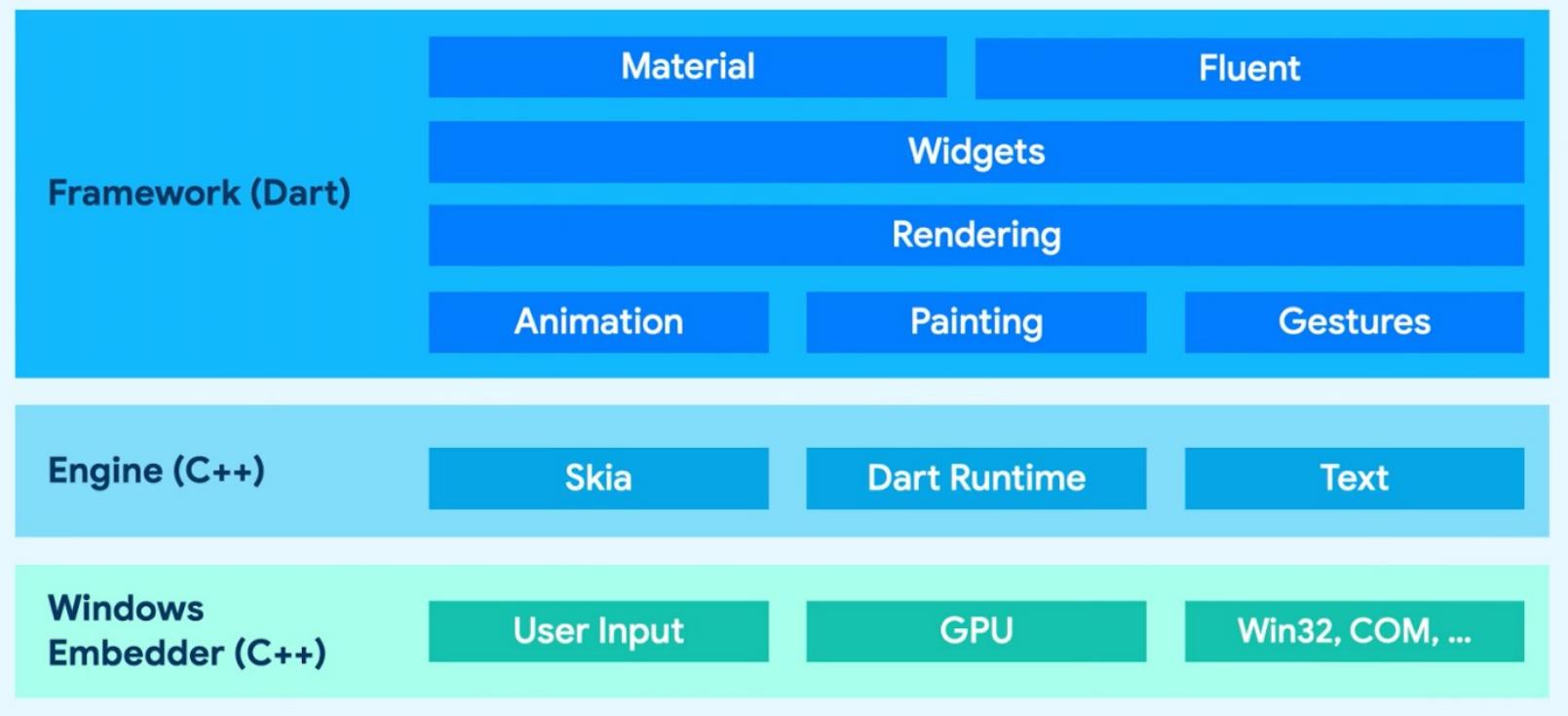
# Flutter VS Dart



<https://stackoverflow.com/questions/43854647/flutter-how-does-it-work-behind-the-scenes>

# Flutter in windows (fresh)

## Flutter Architecture



<https://medium.com/flutter/flutter-in-2022-strategy-and-roadmap-8c5eaf7c4275>

# Lessons Learned After Making the First 10 Commercial Apps in Flutter.



 LeanCode [Follow](#)  
Jul 28 · 10 min read ★

...

by Łukasz Kosman and Jakub Wojtczak

We are sharing the insights after making the first 10 commercial apps within the last 24 months during which we've spent some 17.193,00 hours on Flutter projects.

After reading this article you will learn:

- What are the reasons to choose Flutter? What impact Flutter has on budget and stabilization?
- Is Flutter ready for enterprise apps?
- How is Flutter performing in comparison to Xamarin?
- Which projects is Flutter suitable for?

<https://medium.com/swlh/lessons-learned-after-making-the-first-10-commercial-apps-in-flutter-f420808048cd>



# Lessons Learned After Making the First 10 Commercial Apps in Flutter.



90% of the code on average is shared between iOS and Android.

 LeanCode Follow  
Jul 28 · 10 min read ★

[Twitter](#) [LinkedIn](#) [Facebook](#) [Bookmark](#) ...

by Łukasz Kosman and Jakub Wojtczak

Cooperation with UX/UIs has never been so good.

We are sharing the insights after making the first 10 comm

some **Flutter is quicker.**

**Animations are easy and affordable.**

After reading this article you w

**Flutter apps are much lighter.**

- What are the reasons to choose Flutter? V budget and stabilization?
- Is Flutter ready for enterprise apps?
- How does it compare to native development? **DEVs are happy.**
- What projects is Flutter suitable for?

**Native code is accessible when needed.**

**Proof of Concept in Flutter is easy.**



# Why Flutter Is, in Fact, the Next Big Thing in App Development



It's an unstoppable train coming towards us, and you'd better hop on early



Erik van Baaren

Follow



Sep 23, 2020 · 5 min read



<https://medium.com/better-programming/why-flutter-is-in-fact-the-next-big-thing-in-app-development-8f514dd3a252>

# Why Flutter Is, in Fact, the Next Big Thing in App Development



It's an unstoppable train coming towards us, and you'd better hop on early



Erik van Baaren



## Cross-Platform Development Done Right

Sep 23, 2020 · 5 min read ★



...

Look and Feel

Fuchsia

Linux

Existing Platforms Are a Mess



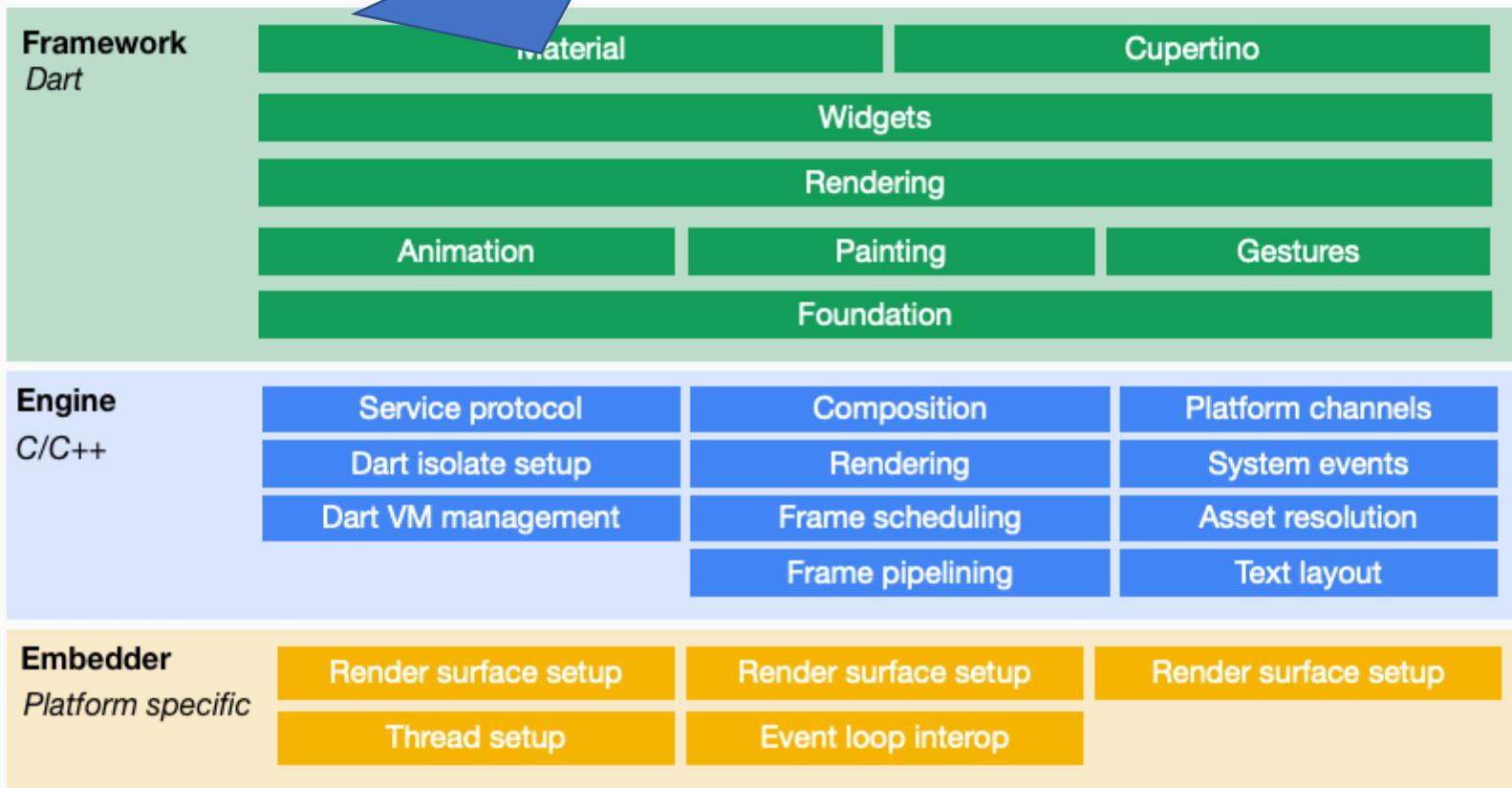
<https://medium.com/better-programming/why-flutter-is-in-fact-the-next-big-thing-in-app-development-8f514dd3a252>



# How?

You program here

## Flutter system overview





## DOCUMENTATION

Overview

**Guides**

Reference

Samples

Design &amp; Quality

- ▶ Android for Cars
- ▶ Android Things
- ▶ Chrome OS devices

App architecture

Introduction

Guide to app architecture

- ▶ Architecture Components
- ▶ App entry points
- ▶ App navigation
- ▶ Dependency injection
- App Startup

Core topics

- ▶ App compatibility
- ▶ Interact with other apps
- ▶ Intents and intent filters
- ▶ User interface
- ▶ App widgets
- ▶ Animations & transitions
- ▶ Images & graphics
- ▶ Audio
- ▶ Servi



For those curious

[Android Developers](#) > [Docs](#) > [Guides](#)Was this helpful?  

# Create and run a wearable app

On this page ▾

- [Set up your environment](#)
- [Create a Wear OS app](#)
- [Start a Wear OS project](#)
- [Launch the emulator and run your Wear OS app](#)
- [Pair a phone with the watch AVD](#)
- ...

Wear OS apps run directly on a watch, giving you access to hardware such as sensors and the GPU. Wearable apps are similar to other apps that use the Android SDK, but differ in design and functionality.

A Wear OS app should work independently of a phone app, allowing users the greatest flexibility in their choice of phones. For more information, see [Independent versus dependent Wear OS apps](#).



**Note:** You can test your app on an actual watch using USB, Wi-Fi, or

[Bluetooth as described in Debug a Wear OS app](#). [Debugging your app on](#)

<https://developer.android.com/training/wearables/get-started/creating>

with different screen shapes and sizes.





4



# Flutter: Building Wear OS app



Souvik Biswas

Follow



Jun 25, 2019 · 7 min read



OLD BUT AN EXAMPLE..

## Introduction

This article will help you to build a Flutter Wear OS (Android Wear) app from scratch.

<https://medium.com/flutter-community/flutter-building-wearos-app-fedf0f06d1b4>

inspiration from [Matt Sullivan](#) article "[Experimenting with Flutter on Wear OS](#)". The plugin, "[wear](#)", created by Matt helped me a lot in managing the screen size of the watch across different screens



# Flutter: Building Wear OS Apps



Souvik Biswas

Follow



Jun 25, 2019 · 7 min read

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search

Sign in Sign up

fluttercommunity / flutter\_wear\_plugin Public

Notifications

Star 77

Fork 24

Code

Issues 5

Pull requests 1

Actions

Projects

Wiki

Security

Insights



master

1 branch

2 tags

Go to file

Code

slightfoot Release v0.1.1

c9265bd on Oct 6, 2020 14 commits

|              |   |               |
|--------------|---|---------------|
| android      | ix Kotlin/Android compileOnly dep on com.google.andr... | 12 months ago |
| example      | Updated to AndroidX and Android embedding v2.           | 12 months ago |
| lib          | Updated to AndroidX and Android embedding v2.           | 12 months ago |
| .gitignore   | Updated to AndroidX and Android embedding v2.           | 12 months ago |
| .metadata    | Updated to AndroidX and Android embedding v2.           | 12 months ago |
| CHANGELOG.md | Release v0.1.1  | 12 months ago |
| LICENSE      | Updated LICENSE and README file                         | 4 years ago   |
| README.md    | ix Kotlin/Android compileOnly dep on com.google.andr... | 12 months ago |
| pubspec.yaml | Release v0.1.1  | 12 months ago |
| wear.iml     | Updated to AndroidX and Android embedding v2.           | 12 months ago |

README.md

## Flutter Wear Plugin

A plugin that offers Flutter support for Wear OS by Google (Android Wear).

To use this plugin you must set your `minSdkVersion` to `23`.

### About

A plugin that offers widgets for Wear OS by Google

[pub.dev/packages/wear](#)

[Readme](#)

[BSD-3-Clause License](#)

### Releases

2 tags

### Packages

No packages published

### Contributors 2

mjohnsullivan Matt Sullivan

slightfoot Simon Lightfoot

### Languages

Dart 65.2% Kotlin 34.8%

## Introduction

This article will help you to build from scratch.

<https://medium.com/flutter-community/flutter-building-wearos-app-fedf0f06d1b4>  
[https://github.com/fluttercommunity/flutter\\_wear\\_plugin](https://github.com/fluttercommunity/flutter_wear_plugin)

The plugin, `wear`, created by `matt` helped me a lot in managing the

screen size of the watch across different screens



For those curious

# Rpi 4 meets Flutter and Rust

#flutter #rust #raspberrypi #linux

At Charlie foxtrot we aim to stay relevant and keep up with new and exciting tech. So how do we do that? We experiment! This time we wanted to capitalize on our in-house competence in Flutter and Rust. Our Flutter expert, Jonathan, had read about Darts foreign function interface (ffi) capabilities and wanted to try it in conjunction with Rust. He asked me if I thought if we could build something that would run on a Raspberry pi (rpi). Without having a clue, I said "YES!" and got to work. The resulting code can be found [here](#).

After some debating, we landed in an idea where we would have a Flutter interface that let the user take a photo via the rpi camera module and then display the image in the app. This is a simple app but considering that Flutter don't have official support for the rpi, this could be quite the challenge. We also decided to build a small (read: very small) library in Rust to communicate with the camera.

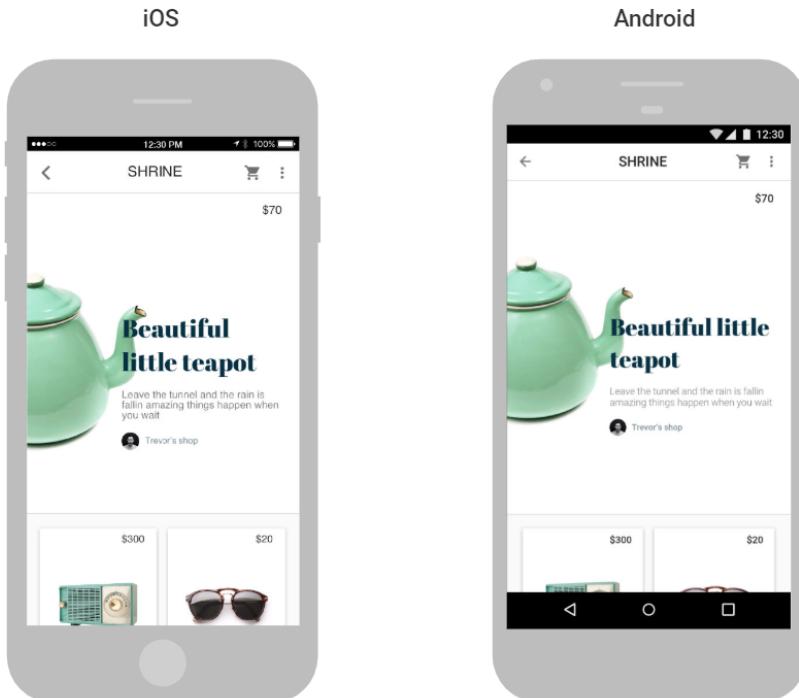
This project has quite a few requirements:

- [Rust](#)
  - [Flutter](#)
  - [flutter-pi](#)
  - [rpi 4 4gb ram](#)
  - [an rpi camera module](#)
  - a touch screen, we went with the [Hyperpixel 4.0](#).
- <https://dev.to/charliefoxtrot/rpi-4-meets-flutter-and-rust-23ma>
- [https://pub.dev/packages/rpi\\_gpio](https://pub.dev/packages/rpi_gpio)



# Flutter

- It allows you to develop apps which run on both iOS and Android. Flutter has built-in Material Design and Cupertino widgets which you can use to create beautiful and professional looking apps.



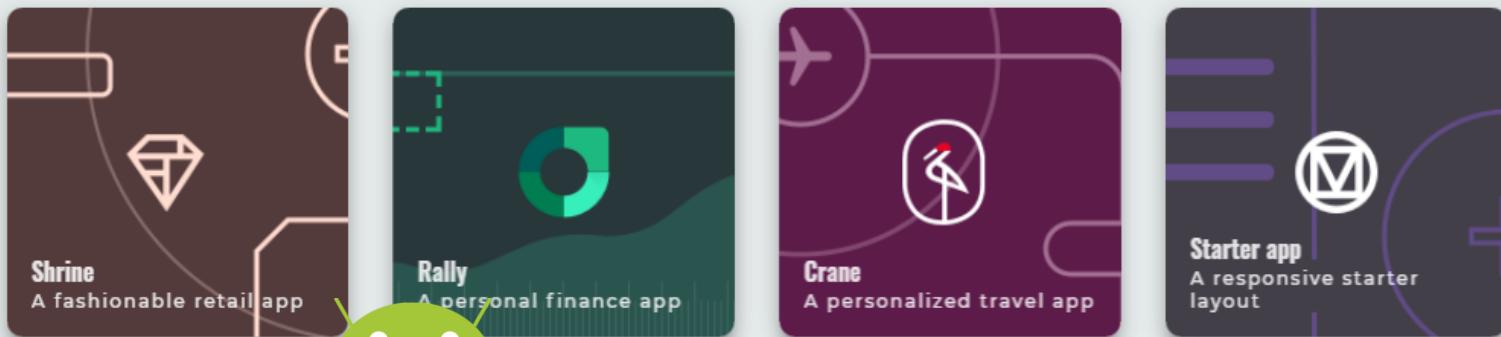
What is Flutter?

<https://flutter.dev/docs/resources/technical-overview>

Themes for both the [Android \(Material Design\)](#) and [iOS \(Cupertino\)](#) platforms.



## Gallery



## Categorie



| MATERIAL  | CUPERTINO   | REFERENCE STYLES & MEDIA  |
|---|---|---|
|  Banner<br>Displaying a banner within a list                     |  Activity Indicator<br>iOS-style activity indicators |  Colors<br>All of the predefined colors          |
|  Bottom app bar<br>Displays navigation and actions at the bottom |  Alerts<br>iOS-style alert dialogs                   |  Typography<br>All of the predefined text styles |

<https://flutter.github.io/samples/#/>

 Bottom navigation  
Bottom navigation with

 Buttons  
iOS-style buttons

# Try Flutter in your browser

Spinning Flutter ▾

Dart

Format Reset Run

```
1 import 'package:flutter/material.dart';
2
3 void main() async {
4   runApp(
5     MaterialApp(
6       debugShowCheckedModeBanner: false,
7       home: Scaffold(
8         body: MyApp(),
9       ),
10    ),
11  );
12 }
13
14 class MyApp extends StatefulWidget {
15   @override
16   _MyAppState createState() => _MyAppState();
17 }
18
19 class _MyAppState extends State<MyApp>
20   with SingleTickerProviderStateMixin {
21   AnimationController controller;
22   Animation<double> animation;
23 }
```

UI Output

JUST SCROLL DOWN ON ENTRY

Console

no issues

# My first Flutter application

# runtime

Flutter

New: You can now build Windows apps with Flutter! [Learn more...](#)

Get started

- 1. Install
- 2. Set up an editor
- 3. Test drive
- 4. Write your first app
- 5. Learn more

From another platform?

- Flutter for Android devs
- Flutter for iOS devs
- Flutter for React Native devs
- Flutter for web devs
- Flutter for Xamarin.Forms devs
- Introduction to declarative UI
- Dart language overview
- Building a web app

Samples & tutorials

Development

## Install

Get started > Install

Select the operating system on which you are installing Flutter:

 Windows    macOS    Linux    Chrome OS

**Important:** If you're in China, first read [Using Flutter in China](#).

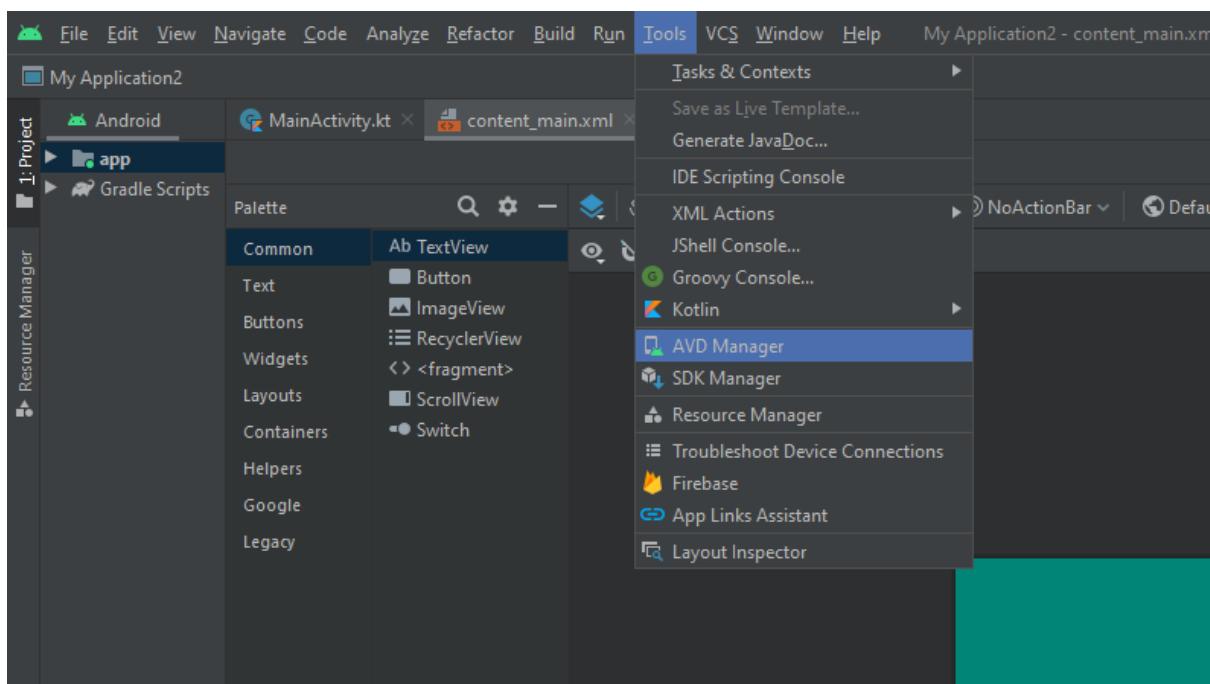
Set up an editor

<https://docs.flutter.dev/get-started/install>



# Android runtime

- Android SDK ( usually via Android Studio)
- Create AVD
- ...



# Create project ( CLI)

- > flutter create test1

```
PS C:\tmp\cm> flutter create test1
Creating project test1...
Running "flutter pub get" in test1...                                1,706ms
Wrote 96 files.

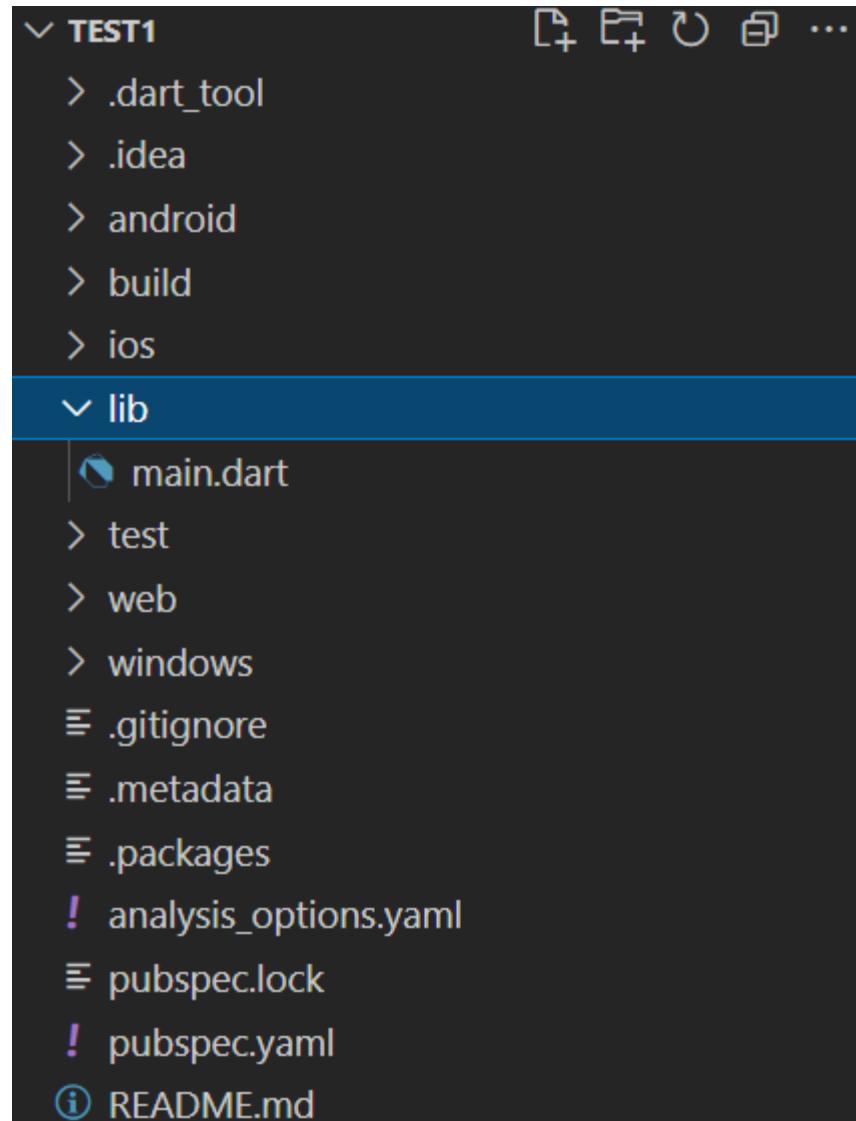
All done!
In order to run your application, type:

$ cd test1
$ flutter run

Your application code is in test1\lib\main.dart.
```

# Now you have this

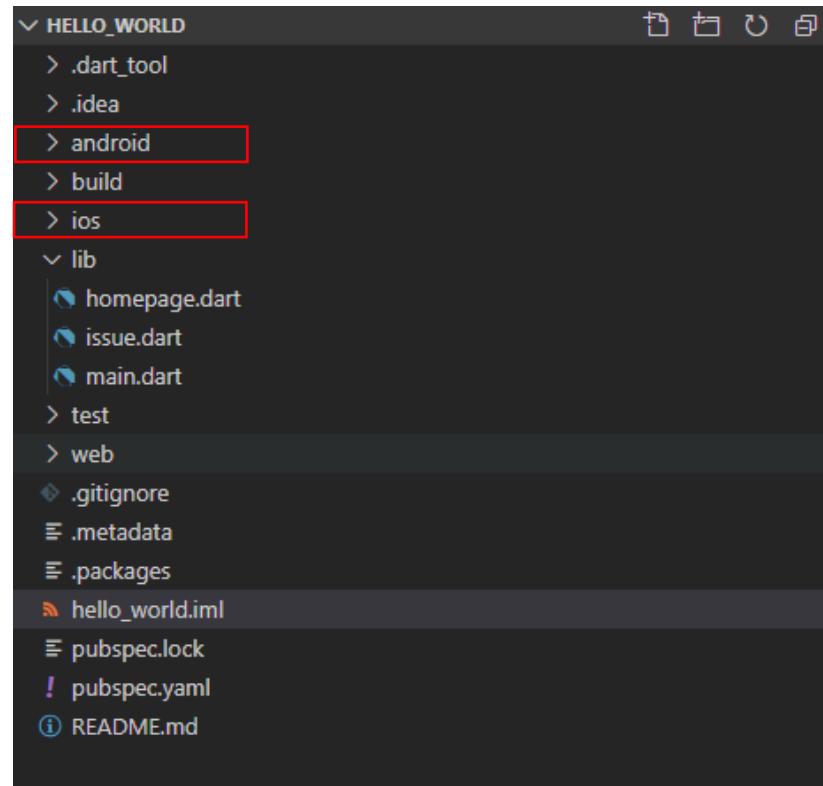
| Directory: C:\tmp\cm\test1 |                  |        |                       |
|----------------------------|------------------|--------|-----------------------|
| Mode                       | LastWriteTime    | Length | Name                  |
| d----                      | 09/03/2022 08:37 |        | .dart_tool            |
| d----                      | 09/03/2022 08:37 |        | .idea                 |
| d----                      | 09/03/2022 08:37 |        | android               |
| d----                      | 09/03/2022 08:37 |        | ios                   |
| d----                      | 09/03/2022 08:37 |        | lib                   |
| d----                      | 09/03/2022 08:37 |        | test                  |
| d----                      | 09/03/2022 08:37 |        | web                   |
| d----                      | 09/03/2022 08:37 |        | windows               |
| -a---                      | 09/03/2022 08:37 | 778    | .gitignore            |
| -a---                      | 09/03/2022 08:37 | 315    | .metadata             |
| -a---                      | 09/03/2022 08:37 | 2331   | .packages             |
| -a---                      | 09/03/2022 08:37 | 1482   | analysis_options.yaml |
| -a---                      | 09/03/2022 08:37 | 3794   | pubspec.lock          |
| -a---                      | 09/03/2022 08:37 | 3814   | pubspec.yaml          |
| -a---                      | 09/03/2022 08:37 | 564    | README.md             |
| -a---                      | 09/03/2022 08:37 | 913    | test1.iml             |



# After creating your first Flutter project

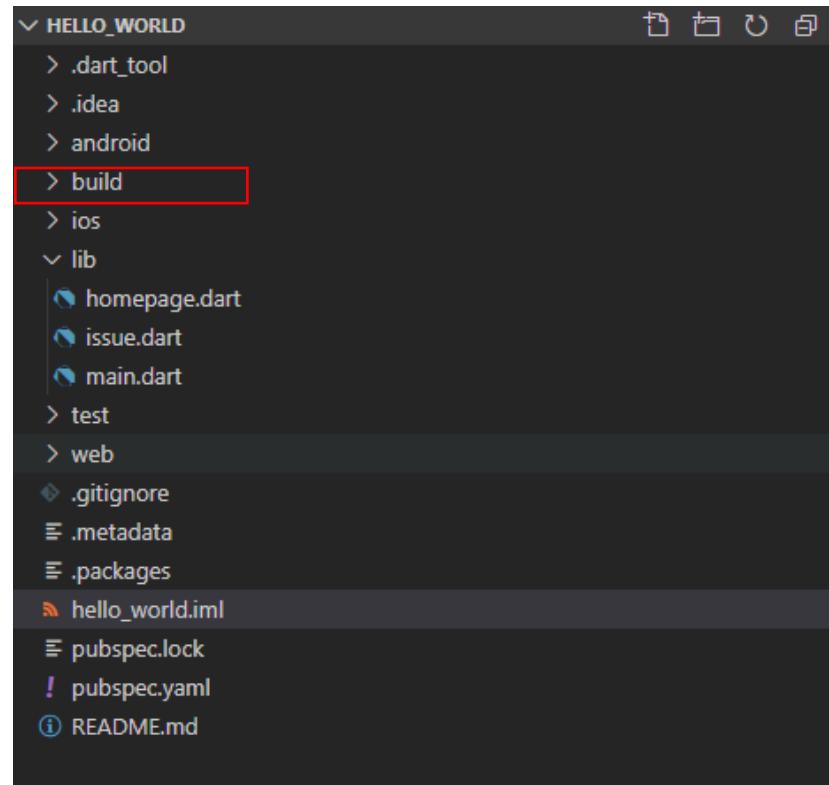
- **Android and iOS**

- a complete Android and iOS app respectively, with all their respective files. Whenever you want to implement any platform specific feature you can implement it by going in these directories.
- In most cases you do not need to edit anything in these folder directly.



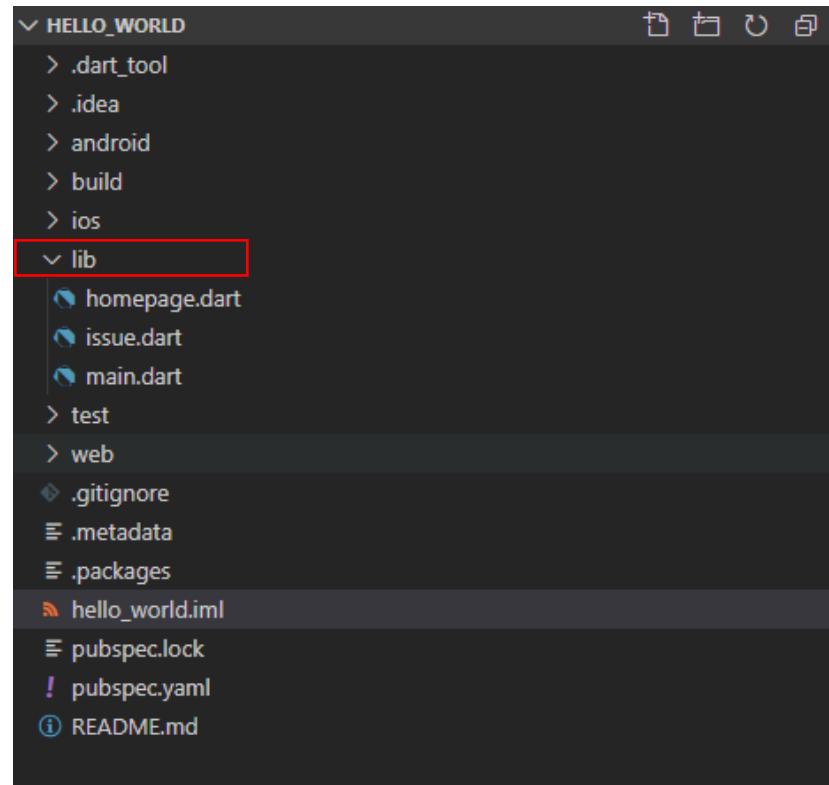
# The project structure

- build
  - compiled code of your Flutter application.
  - The content of this folder is automatically generated as part of the Flutter build process
  - you don't need to change anything manually inside this folder.



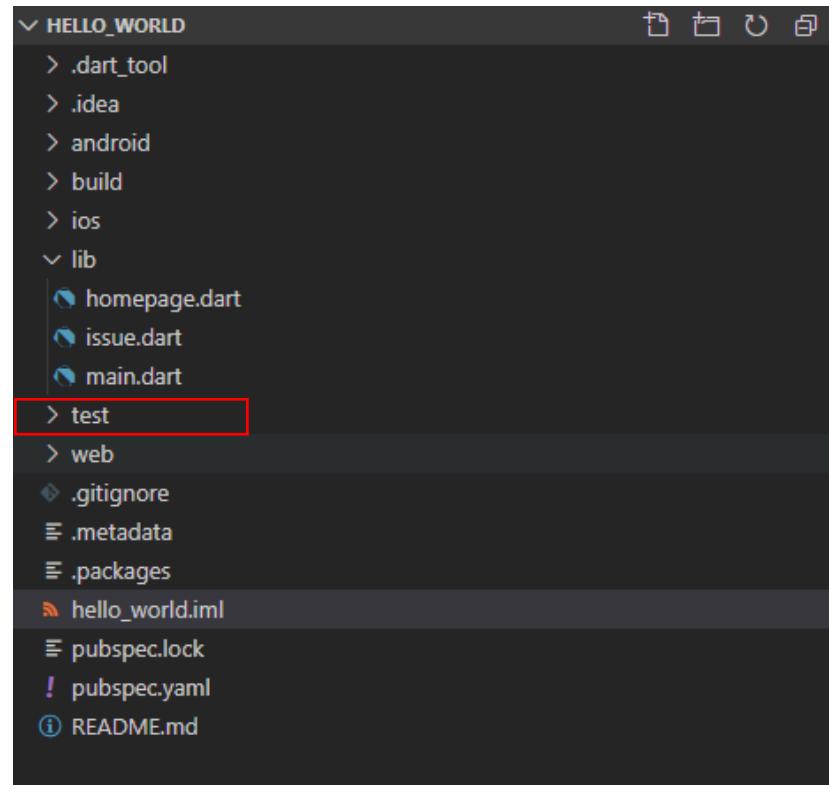
# The project structure

- **lib**
  - holds all your dart code used to run your app.
  - By default this folder is containing the file main.dart



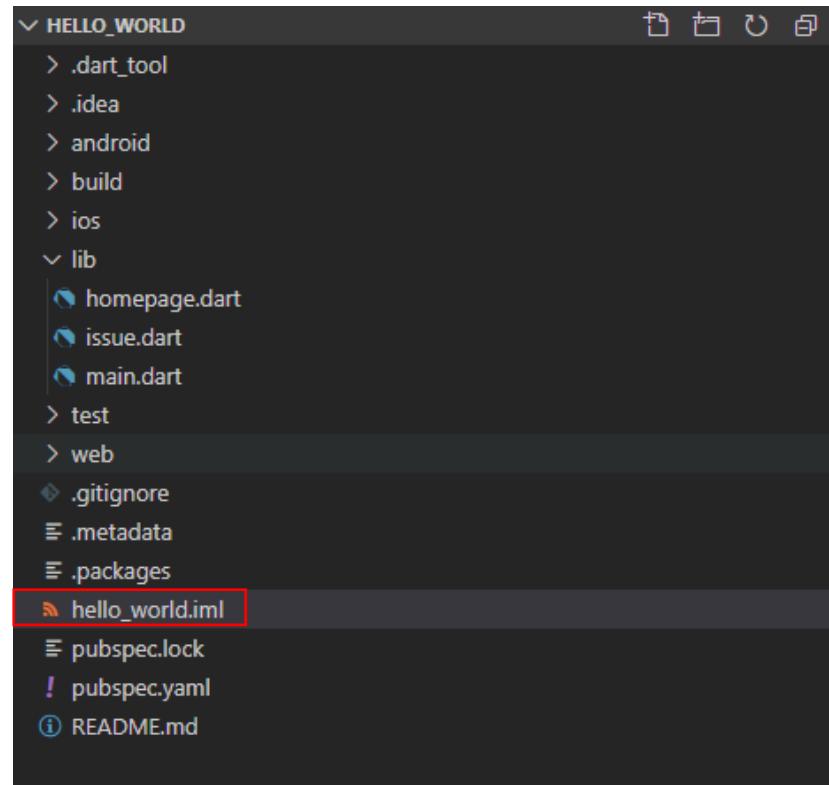
# The project structure

- test folder
  - is containing code which is written for the Flutter application in order to perform automated test when building.



# Files In Your Flutter Application

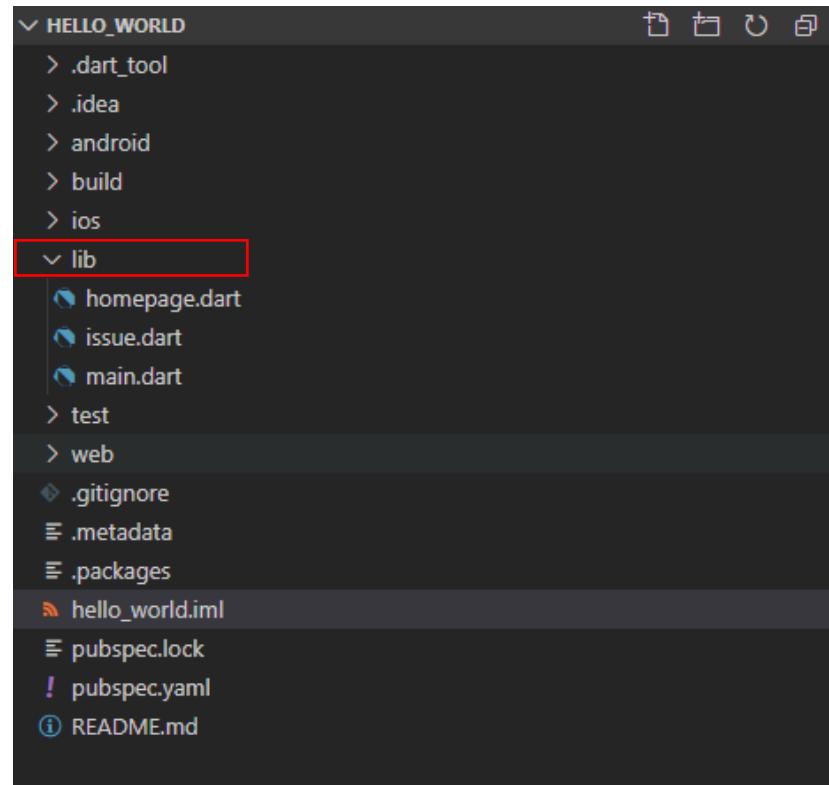
- *hello\_world.iml*
  - named according to the project's name and contains further settings of the Flutter project.
  - Usually not changed by you and the content is managed by the Flutter SDK in an automated way.



# Files In Your Flutter Application

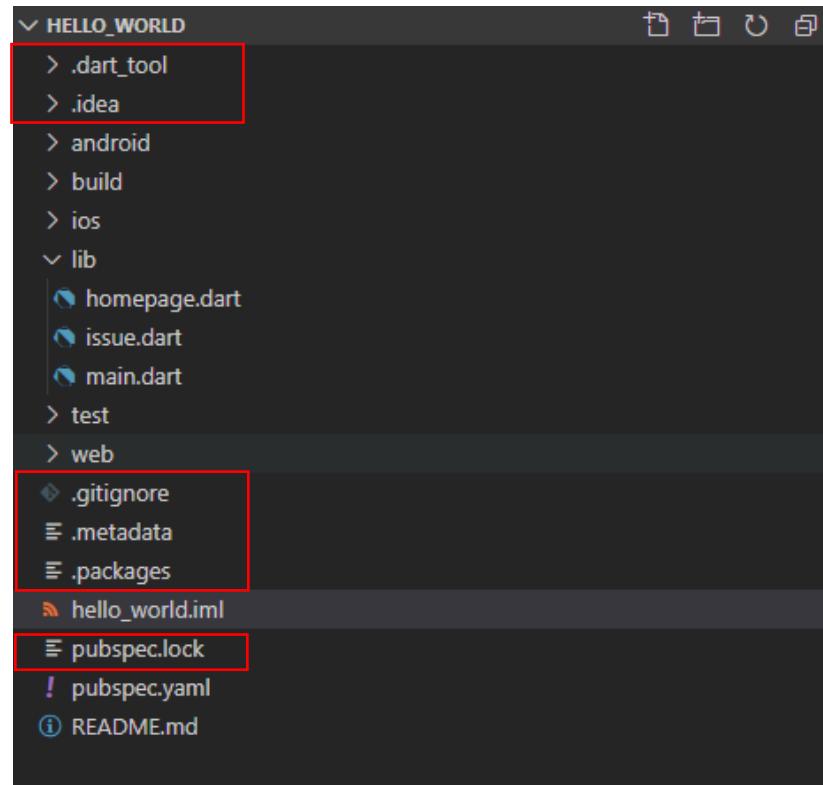
- *pubspec.yaml*

- *pubspec.yaml* is a special file. It contains your *app name, description, SDK version, dependencies*, and other important stuff.
- contains:
  - general project settings like name, description and version of the project
  - project dependencies
  - assets (e.g. images) which should be available in your Flutter application
  - E.g. when adding new dependencies (Flutter Packages) to the project this file will be used to add the name of the package in the dependencies section and then the Flutter SDK will take care of downloading and adding this package afterwards.



# The Files and Folder that we mainly don't touch

- In 99.99% of cases, we don't touch these files. As manually edit them can destroy the entire project (expect .gitignore, and README.md 😊).
- .idea directory holds all project specific settings. Settings are stored mainly in the form of XML files. We normally don't touch these folders.
- The .gitignore file is a text file that tells Git, which files or folders to ignore in a project.
- .metadata, .packages, pubspec.lock are generated by flutter itself. These files are used by the framework itself for internal usages. We normally don't touch them. As they perform internal tasks.
- README.md is a markdown file mainly used in version control like Git for representing information of other files.



# Run project (CLI)

- Select one of the available runtimes

```
PS C:\tmp\cm\test1> flutter run
Multiple devices found:
Windows (desktop) • windows • windows-x64      • Microsoft Windows [Version 10.0.19043.1526]
Chrome (web)       • chrome   • web-javascript • Google Chrome 98.0.4758.102
Edge (web)         • edge     • web-javascript • Microsoft Edge 99.0.1150.36
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (To quit, press "q/Q"): |
```

# Run project (CLI)

- Select one of the available runtimes

```
PS C:\tmp\cm\test1> flutter run
Multiple devices found:
Windows (desktop) • windows • windows-x64      • Microsoft Windows [Version 10.0.19043.1526]
Chrome (web)       • chrome   • web-javascript • Google Chrome 98.0.4758.102
Edge (web)         • edge     • web-javascript • Microsoft Edge 99.0.1150.36
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (To quit, press "q/Q"): 1
Launching lib\main.dart on Windows in debug mode...
Exception: Unable to find suitable Visual Studio toolchain. Please run `flutter doctor` for more details.
```

But you must have the toolchain available

Windows → visual studio toolchain

Android → Android SDK

iOS → Xcode toolchain

Web / browser – supported by default over javascript

# Debug using devtools

## DevTools

### Overview

Install from Android Studio & IntelliJ

Install from VS Code

Install from command line

Flutter inspector

Performance view

CPU Profiler view

Memory view

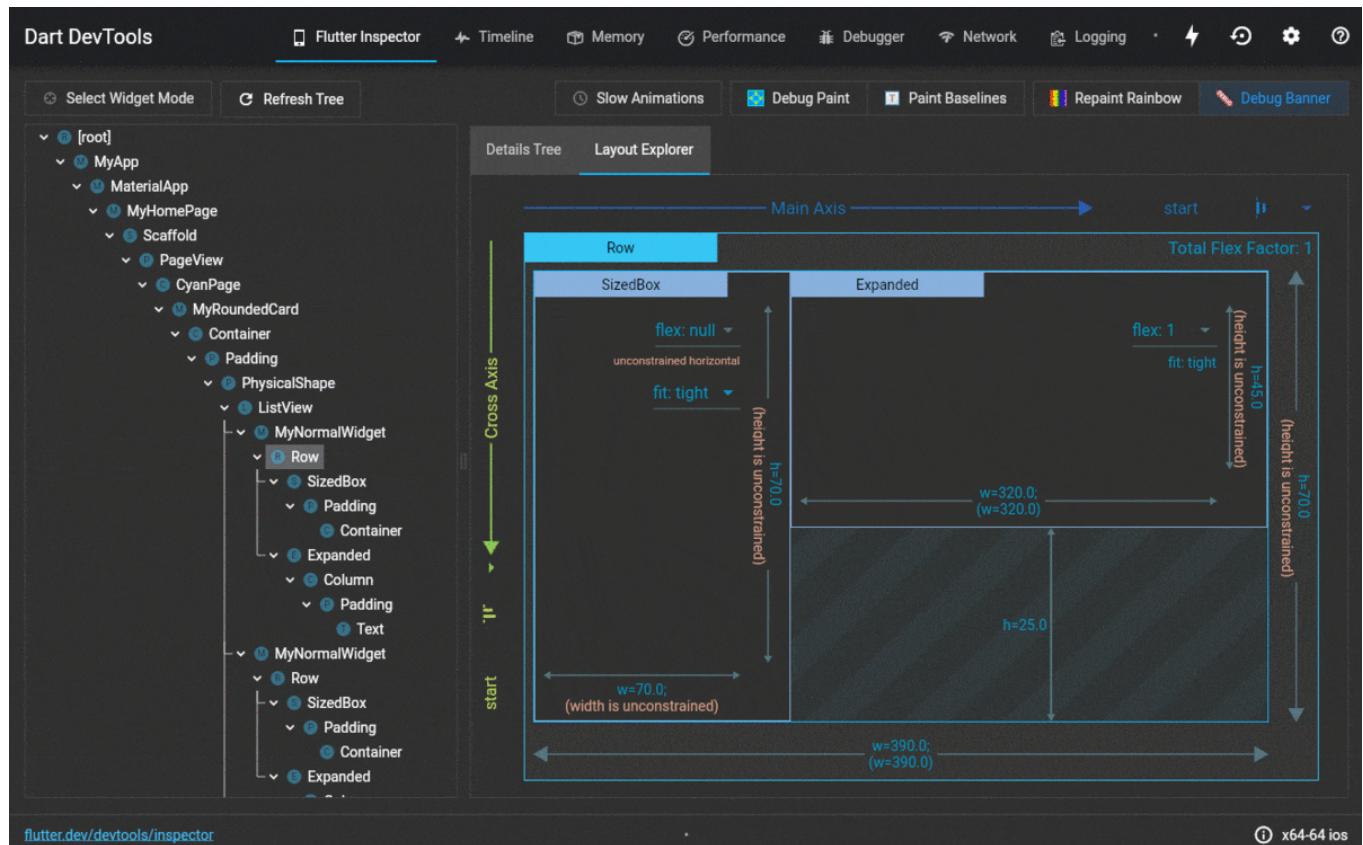
Network view

Debugger

Logging view

App size tool

Release notes



<https://docs.flutter.dev/development/tools/devtools/overview>

# App vs devtools ( in chrome, using CLI)

## Your flutter app

```
> Flutter run
```

```
Launching lib\main.dart on Chrome in debug mode...
```

```
lib\main.dart:1
```

```
This app is linked to the debug service:  
ws://127.0.0.1:64047/FwaoHyipZ5g=/ws
```

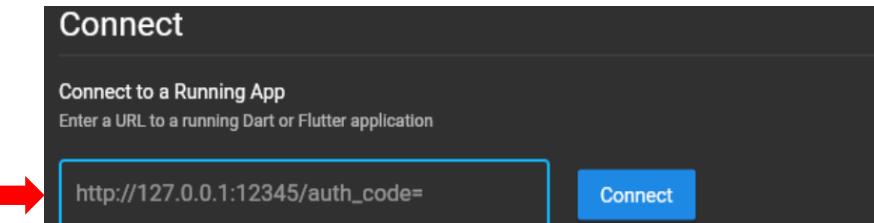
```
Debug service listening on
```

```
ws://127.0.0.1:64047/FwaoHyipZ5g=/ws
```

## devtools

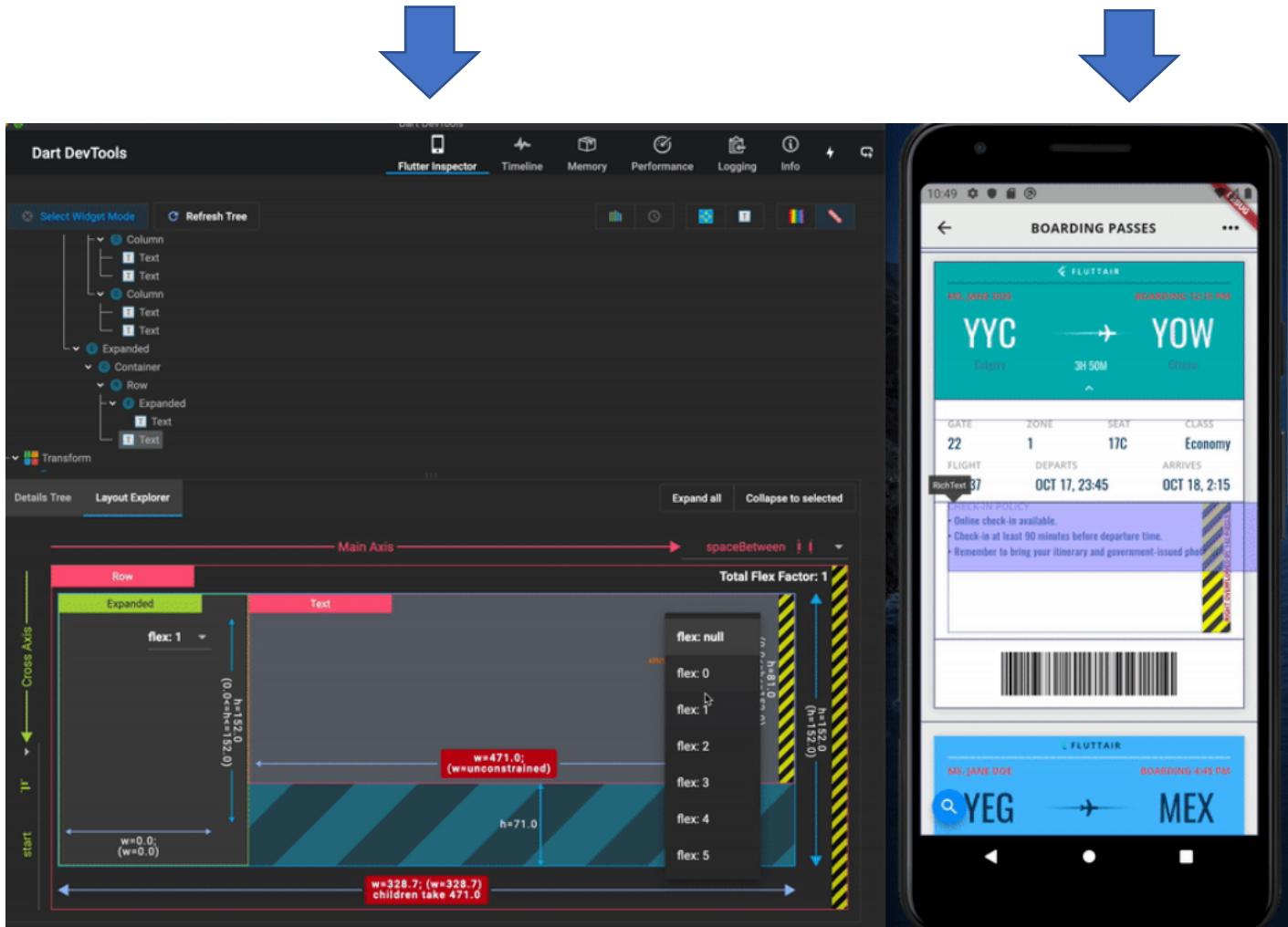
- Go to

<http://localhost:9100>



devtools

Emulator/chrome/phone

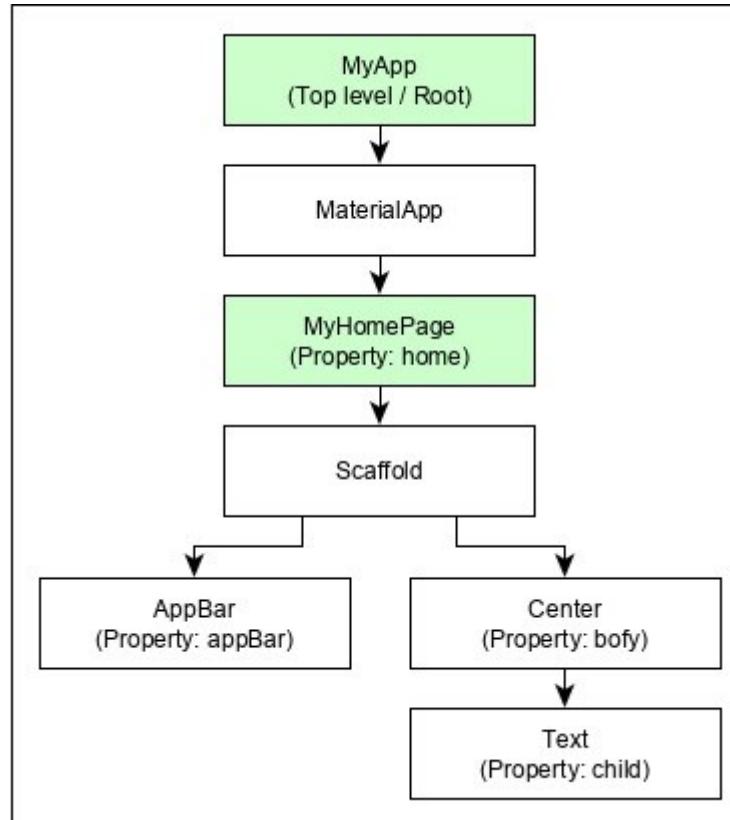


# Basic concepts

Quick overview

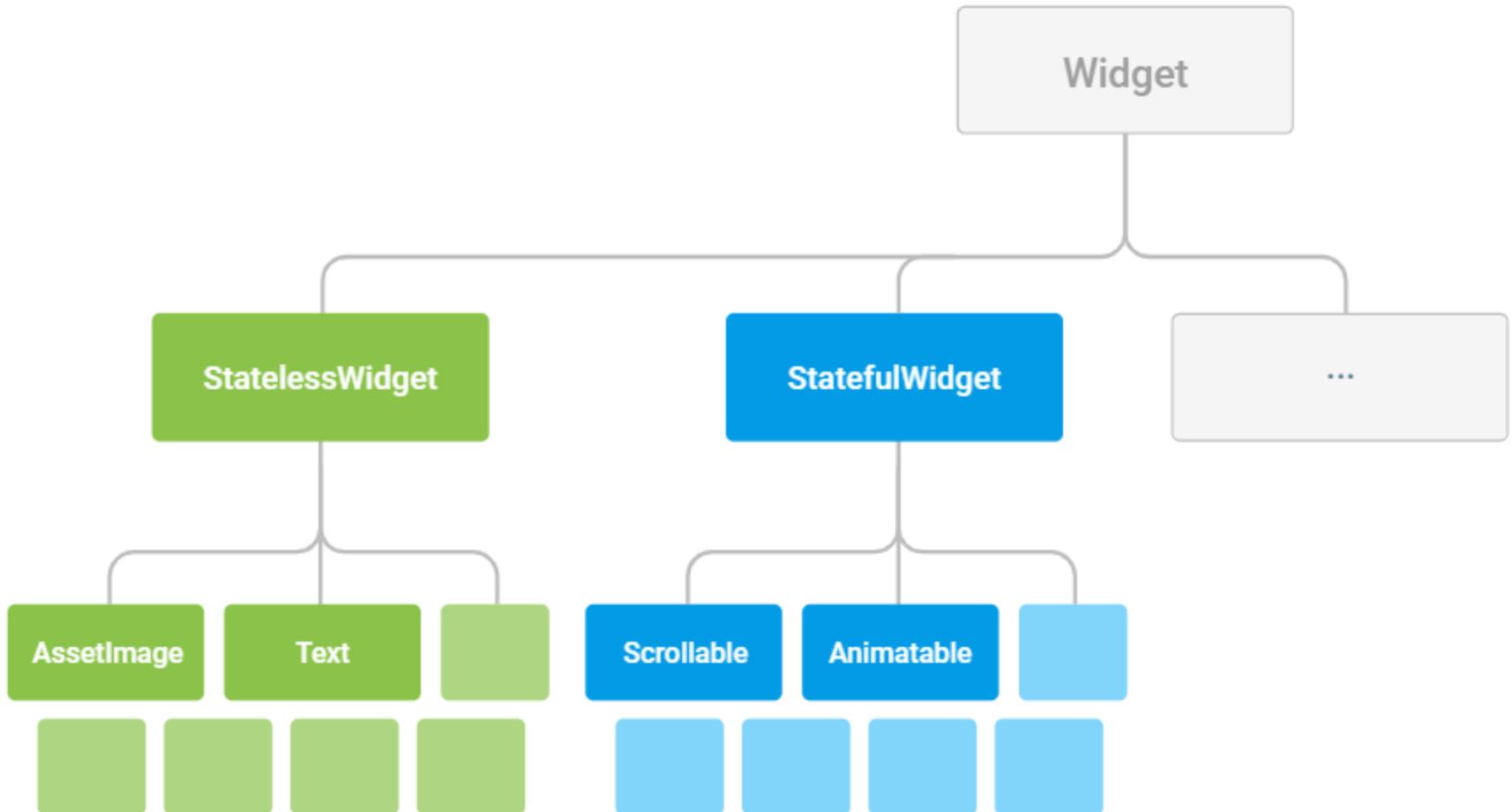


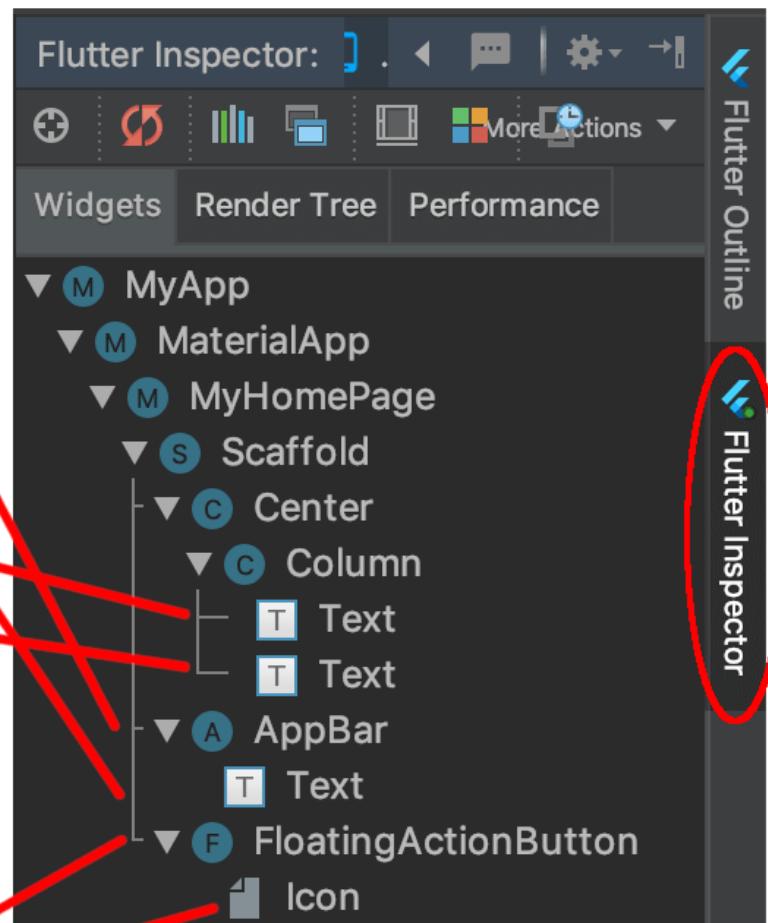
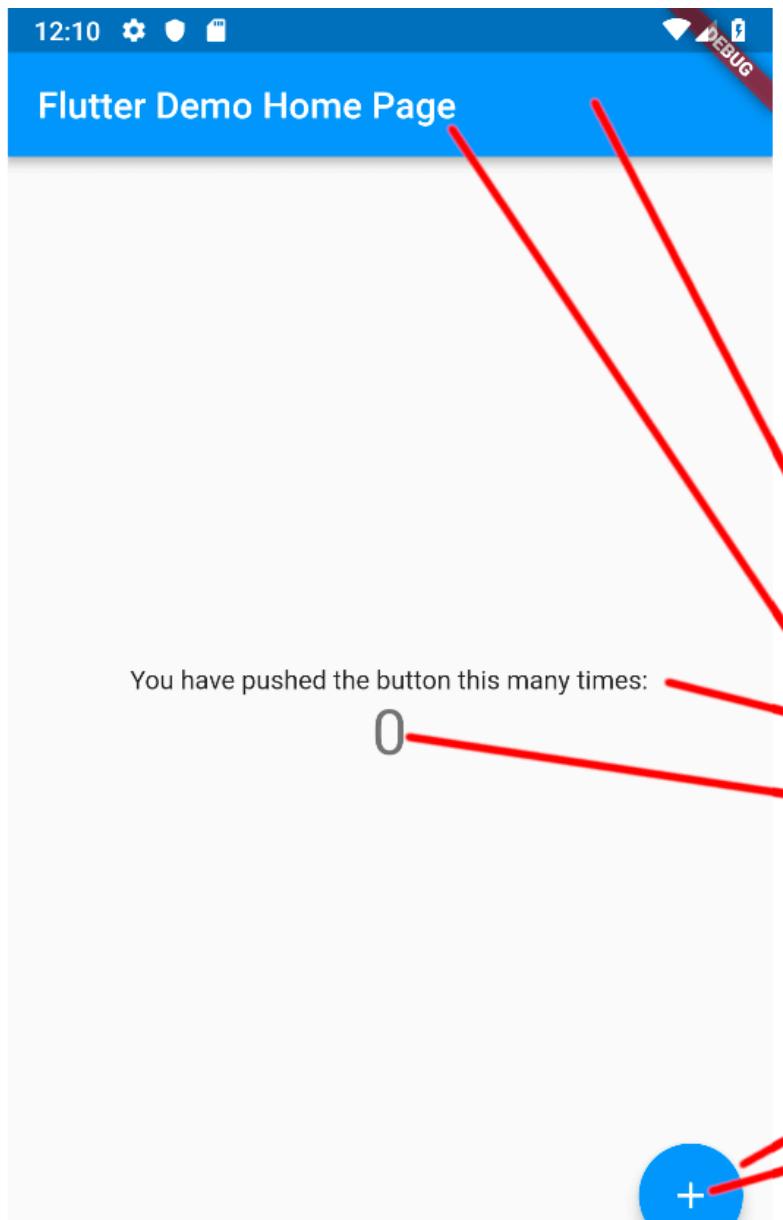
# Flutter application architecture



[https://www.tutorialspoint.com/flutter/flutter\\_architecture\\_application.htm](https://www.tutorialspoint.com/flutter/flutter_architecture_application.htm)

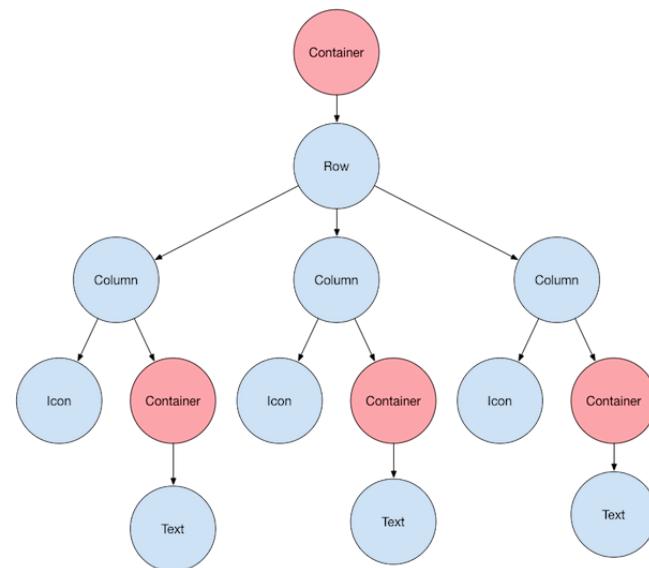
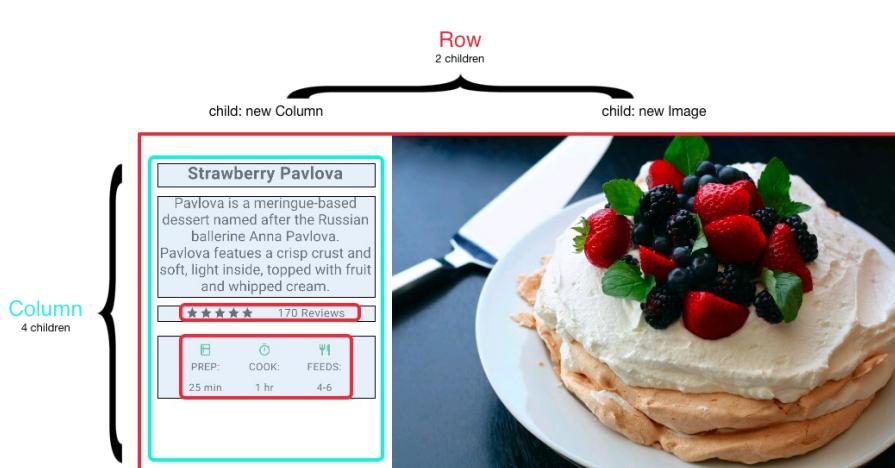
# Everything's a widget





# Building widgets

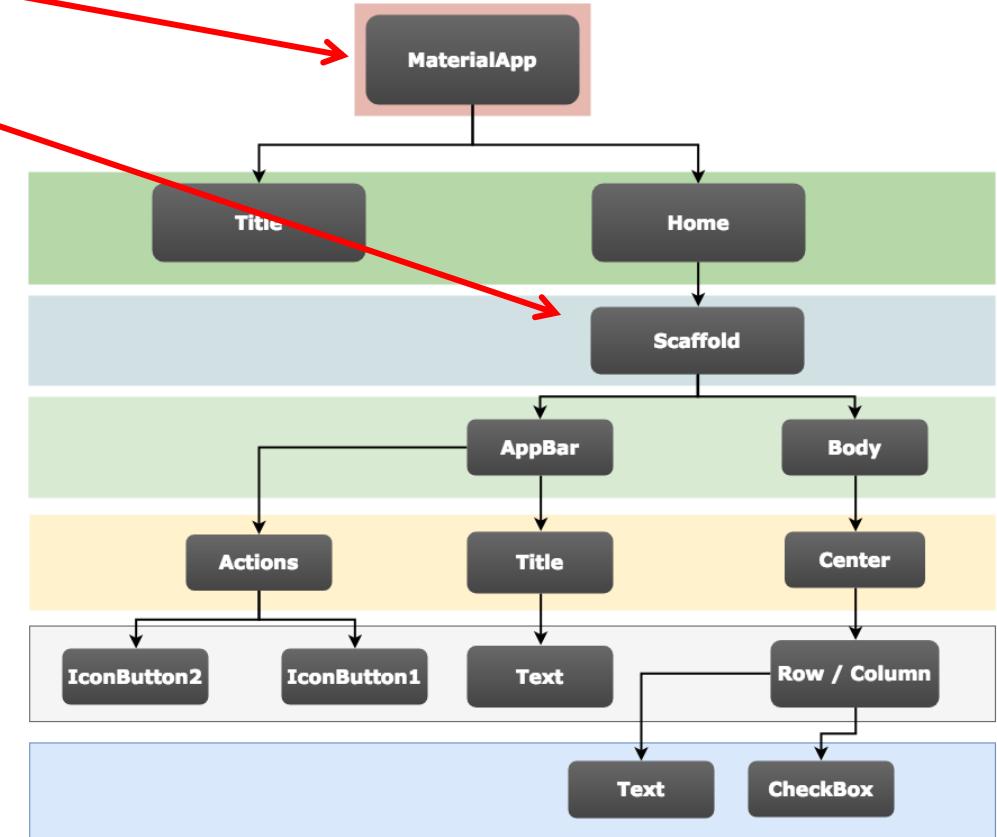
- You define the unique characteristics of a widget by implementing a `build()` function that returns a tree (or hierarchy) of widgets. This tree represents the widget's part of the user interface in more concrete terms.



```

class MyAppState extends State<MyApp> {
  bool checkBoxValue = false;
  String actionText = "Default";
  @override
  Widget build(BuildContext ctxt) {
    return new MaterialApp(
      title: "MySampleApplication",
      home: new Scaffold(
        appBar: new AppBar(
          title: new Text("Hello Flutter App"),
          actions: <Widget> [
            new IconButton (
              icon: new Icon(Icons.add_comment),
              onPressed: () {
                setState(() {
                  actionText = "New Text";
                });
              }
            ),
            new IconButton (
              icon: new Icon(Icons.remove),
              onPressed: () {
                setState(() {
                  actionText = "Default";
                });
              }
            ),
          ],
        ),
        body: new Center(
          child: new Column(
            children: <Widget>[
              widget.TextInput,
              new Text(actionText),
              new Checkbox(
                value: checkBoxValue,
                onChanged: (bool newValue){
                  setState(() {
                    checkBoxValue = newValue;
                  });
                }
              )
            ],
          ),
        ),
      );
    }
}

```



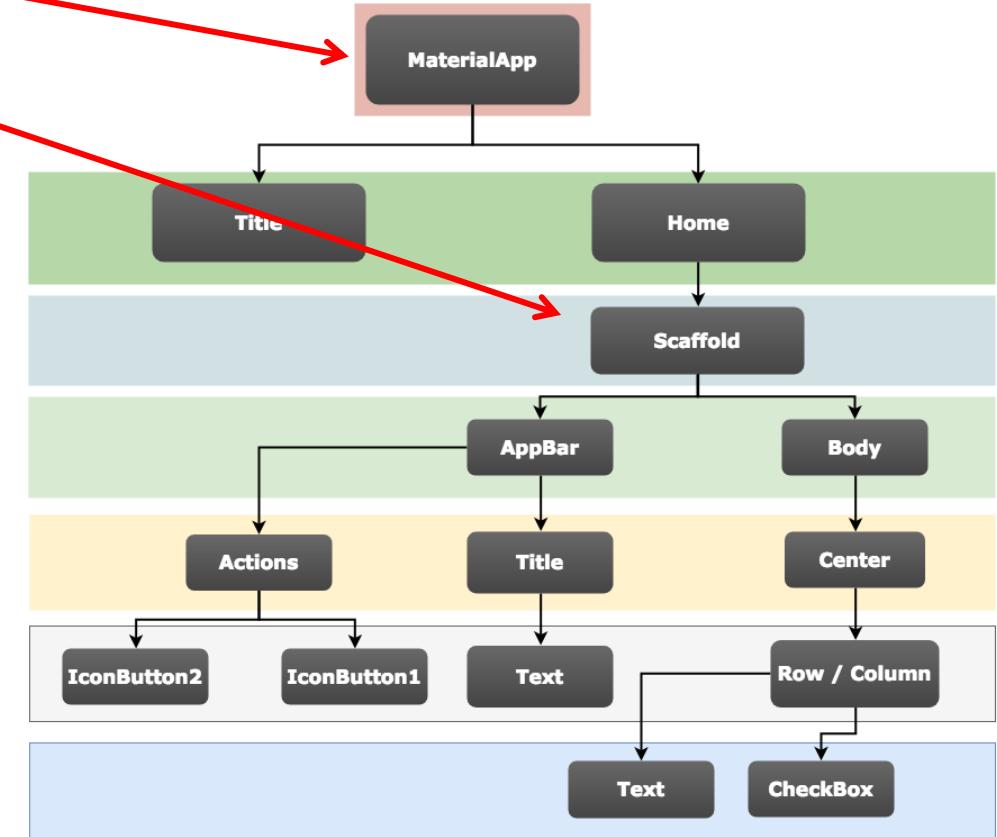
Flutter : From Zero To Comfortable

<https://proandroiddev.com/flutter-from-zero-to-comfortable-6b1d6b2d20e>

```

class MyAppState extends State<MyApp> {
  bool checkBoxValue = false;
  String actionText = "Default";
  @override
  Widget build(BuildContext ctxt) {
    return new MaterialApp(
      title: "MySampleApplication",
      home: new Scaffold(
        appBar: new AppBar(
          title: new Text("Hello Flutter App"),
          actions: <Widget> [
            new IconButton (
              icon: new Icon(Icons.add_comment),
              onPressed: () {
                setState(() {
                  actionText = "New Text";
                });
              }
            ),
            new IconButton (
              icon: new Icon(Icons.remove),
              onPressed: () {
                setState(() {
                  actionText = "Default";
                });
              }
            ),
          ],
        ),
        body: new Center(
          child: new Column(
            children: <Widget>[
              widget.TextInput,
              new Text(actionText),
              new Checkbox(
                value: checkBoxValue,
                onChanged: (bool newValue){
                  setState(() {
                    checkBoxValue = newValue;
                  });
                }
              )
            ],
          ),
        ),
      );
    }
}

```



Flutter : From Zero To Comfortable

<https://proandroiddev.com/flutter-from-zero-to-comfortable-6b1d6b2d20e>





# A lots of “{}()”....

```
// Copyright 2019 The Flutter team. All rights reserved.  
// Use of this source code is governed by a BSD-style  
license that can be  
// found in the LICENSE file.
```

```
import 'package:flutter/material.dart';  
  
import 'package:gallery/l10n/gallery_localizations.dart';  
  
enum ListDemoType {  
  oneLine,  
  twoLine,  
}  
  
class ListDemo extends StatelessWidget {  
  const ListDemo({Key key, this.type}) : super(key: key);  
  
  final ListDemoType type;  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        automaticallyImplyLeading: false,  
        title:  
        Text(GalleryLocalizations.of(context).demoListsTitle),  
      ),  
      body: Scrollbar(  
        child: ListView(  
          padding: EdgeInsets.symmetric(vertical: 8),  
          children: [  
            for (int index = 1; index < 21; index++)  
              ListTile(  
                leading: ExcludeSemantics(  
                  child: CircleAvatar(child:  
                    Text('$index'),  
                ),  
                title: Text(  
                  GalleryLocalizations.of(context).demoBottomSheetItem(index),  
                ),  
              ),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

## Lists

- 1 Item 1
- 2 Item 2
- 3 Item 3
- 4 Item 4
- 5 Item 5
- 6 Item 6
- 7 Item 7
- 8 Item 8
- 9 Item 9
- 10 Item 10
- 11 Item 11

# Types of Widget: UI visibility

## Visible (Output and Input)

- Text
- Button
- Image

## Invisible (Layout and Control)

- Column
  - vertical alignment.
- Row
  - horizontal alignment
- Center
  - center the child widget
- Padding
  - padding in specified directions.
- **Scaffold**
  - common material design elements like AppBar, Floating Action Buttons, Drawers, etc.
- **Stack**
  - mainly used for **overlapping** a widget
- (...)

# Types of Widget: state

## stateless

- don't store any state.
- don't store values that might change.
- an Icon is stateless;
  - you set the icon image when you create it and then it doesn't change any more.
- A Text widget is also stateless.
  - You might say, "But wait, you can change the text value." True, but if you want to change the text value, you just create a whole new widget with new text.
  - doesn't store a text property that can be changed.

## stateful widget

- can keep track of changes and update the UI based on those changes.
- it creates a State object that keeps track of the changes.
  - When the values in the State object change,
  - it creates a whole new widget with the updated values.
- a checkbox is a stateful widget

# Flutter resources?



Thanks for attending Flutter Interact 2019—you can still watch the session recordings!  
Flutter 1.12 is live! Check out the latest announcement and see what's new on the site.



Thanks for attending Flutter Interact 2019—you can still watch the session recordings!  
Flutter 1.12 is live! Check out the [latest announcement](#) and see what's new on the site.

Get started

Samples &amp; tutorials

Flutter Gallery [NEW] Sample apps on GitHub 

Cookbook

[Codelabs](#)

Tutorials

Development

▶ User interface

▶ Data &amp; backend

▶ Accessibility &amp; internationalization

▶ Platform integration

▶ Packages &amp; plugins

▶ Add Flutter to existing app

▶ Tools &amp; techniques

AndroidX migration

Testing &amp; debugging

Performance &amp; optimization

Deployment

<https://flutter.dev/docs/codelabs>

Re:

# Codelabs



## Contents

- [Basic Flutter layout concepts](#)
- [Write Your First Flutter App, part 1](#)
- [Write Your First Flutter App, part 2](#)
- [Building Beautiful UIs with Flutter](#)
- [Implicit Animations](#)
- [Adding Google Maps to a Flutter App](#)
- [Build a Photo Sharing App with Google Photos and Flutter](#)
- [Building a Cupertino app with Flutter](#)
- [Firebase for Flutter](#)
- [MDC 101 Flutter: Material Components \(MDC\) Basics](#)
- [MDC 102 Flutter: Material Structure and Layout](#)
- [MDC 103 Flutter: Material Theming with Color, Shape, Elevation, and Type](#)
- [MDC 104 Flutter: Material Advanced Components](#)
- [Other resources](#)

## Basic Flutter layout concepts

Use DartPad in a browser (no need to download Flutter or Dart!) to learn the basics of creating a Flutter layout.

## Write Your First Flutter App, part 1

! app that generates proposed names for a startup company. In part one, you'll use pairs of words at random and inserts them into an infinite scrolling list.

## Write Your First Flutter App, part 2

Create a simple mobile app that generates proposed names for a startup company. In part two, you'll



Thanks for attending Flutter Interact 2019—you can still watch the session recordings!  
Flutter 1.12 is live! Check out the [latest announcement](#) and see what's new on the site.

- Get started 
- Samples & tutorials 

  - Flutter Gallery [NEW] 
  - Sample apps on GitHub 

- Cookbook
- Codelabs** 
- Tutorials
- Development 

  - ▶ User interface
  - ▶ Data & backend
  - ▶ Accessibility & internationalization
  - ▶ Platform integration
  - ▶ Packages & plugins
  - ▶ Add Flutter to existing app
  - ▶ Tools & techniques
  - AndroidX migration

- Testing & debugging 
- Performance & optimization 
- Deployment 

# Codelabs



## Contents

- [Basic Flutter layout concepts](#)
- [Write Your First Flutter App, part 1](#)
- [Write Your First Flutter App, part 2](#)
- [Building Beautiful UIs with Flutter](#)
- [Implicit Animations](#)
- [Adding Google Maps to a Flutter App](#)
- [Build a Photo Sharing App with Google Photos and Flutter](#)
- [Building a Cupertino app with Flutter](#)
- [Firebase for Flutter](#)
- [MDC 101 Flutter: Material Components \(MDC\) Basics](#)
- [MDC 102 Flutter: Material Structure and Layout](#)
- [MDC 103 Flutter: Material Theming with Color, Shape, Elevation, and Type](#)
- [MDC 104 Flutter: Material Advanced Components](#)
- [Other resources](#)

## Basic Flutter layout concepts

Use DartPad in a browser (no need to download Flutter or Dart!) to learn the basics of creating a Flutter layout.

## Write Your First Flutter App, part 1

! app that generates proposed names for a startup company. In part one, you'll use pairs of words at random and inserts them into an infinite scrolling list.

<https://flutter.dev/docs/codelabs>

## Write Your First Flutter App, part 2

Create a simple mobile app that generates proposed names for a startup company. In part two, you'll

Thanks for attending Flutter Interact 2019—you can still watch the session recordings!  
Flutter 1.12 is live! Check out the [latest announcement](#) and see what's new on the site.

Get started

Samples &amp; tutorials

[Flutter Gallery \[NEW\]](#)[Sample apps on GitHub](#)[Cookbook](#)[Codelabs](#)[Tutorials](#)

Development

▶ User interface

▶ Data &amp; backend

▶ Accessibility &amp; internationalization

▶ Platform integration

▶ Packages &amp; plugins

▶ Add Flutter to existing app

▶ Tools &amp; techniques

[AndroidX migration](#)

Testing &amp; debugging

Performance &amp; optimization

Deployment

<https://flutter.dev/docs/codelabs>

Re:

# Codelabs



## Contents

- [Basic Flutter layout concepts](#)
- [Write Your First Flutter App, part 1](#)
- [Write Your First Flutter App, part 2](#)
- [Building Beautiful UIs with Flutter](#)
- [Implicit Animations](#)
- [Adding Google Maps to a Flutter App](#)
- [Build a Photo Sharing App with Google Photos and Flutter](#)
- [Building a Cupertino app with Flutter](#)
- [Firebase for Flutter](#)
- [MDC 101 Flutter: Material Components \(MDC\) Basics](#)
- [MDC 102 Flutter: Material Structure and Layout](#)
- [MDC 103 Flutter: Material Theming with Color, Shape, Elevation, and Type](#)
- [MDC 104 Flutter: Material Advanced Components](#)
- [Other resources](#)

## Basic Flutter layout concepts

Use DartPad in a browser (no need to download Flutter or Dart!) to learn the basics of creating a Flutter layout.

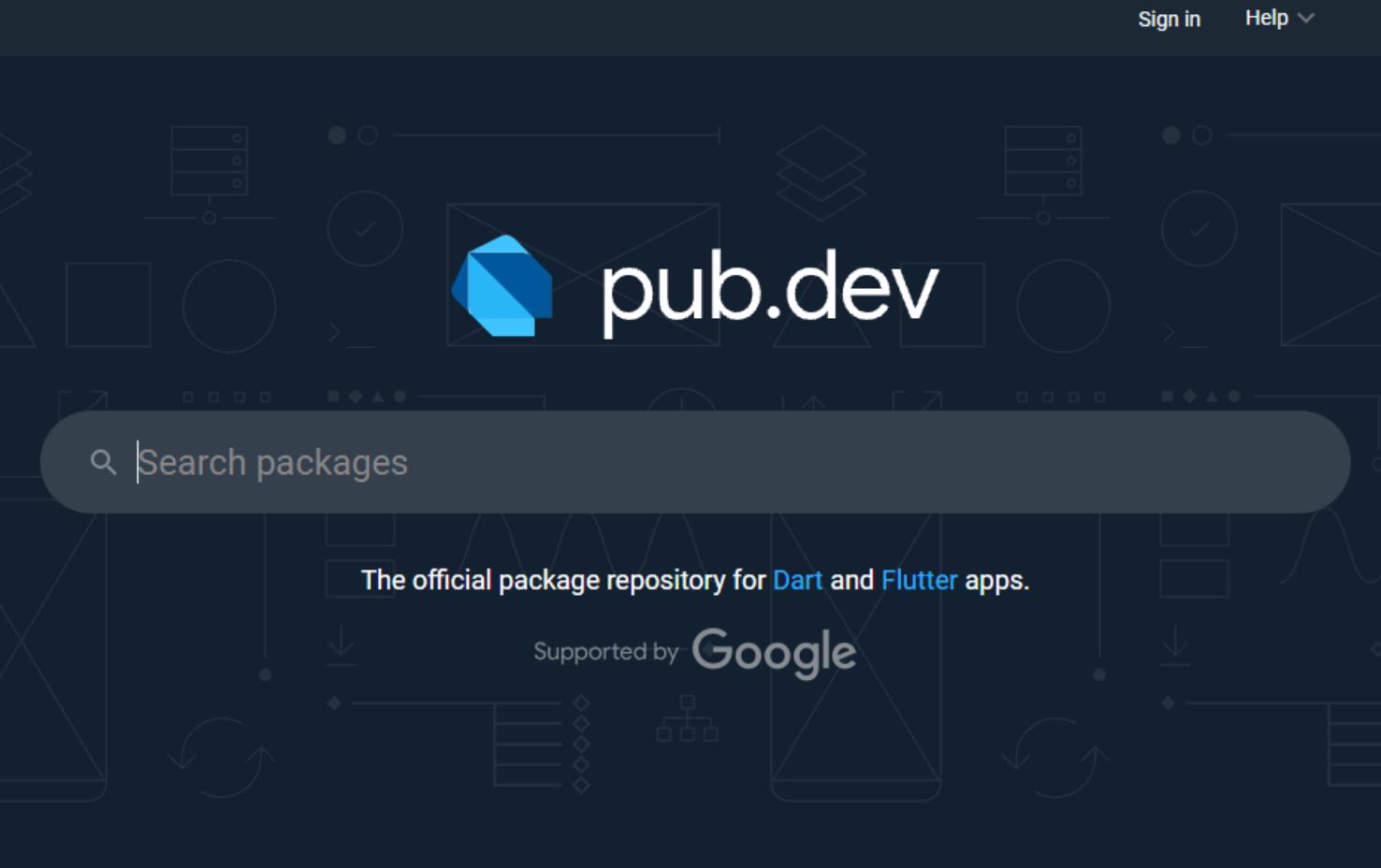
## Write Your First Flutter App, part 1

! app that generates proposed names for a startup company. In part one, you'll use pairs of words at random and inserts them into an infinite scrolling list.

## Write Your First Flutter App, part 2

Create a simple mobile app that generates proposed names for a startup company. In part two, you'll





## Flutter Favorites

Some of the packages that demonstrate the [highest levels of quality](#), selected by the Flutter Ecosystem Committee

[infinite\\_scroll\\_pagination](#)

Lazily load and display pages of

<https://pub.dev/>

[flutter\\_native\\_splash](#)

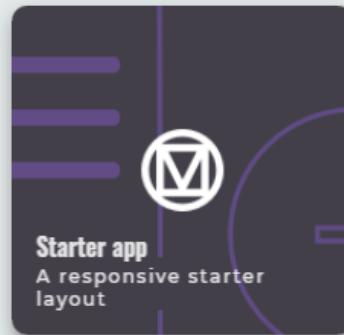
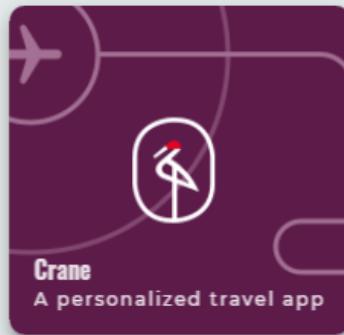
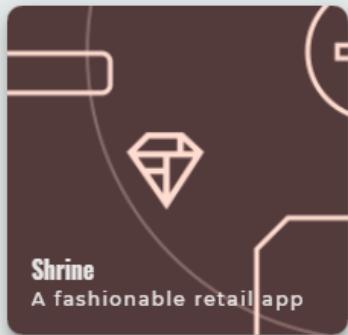
Generates native code to  
er's default  
ash screen with

[bottom\\_navy\\_bar](#)

A beautiful and animated  
bottom navigation. The  
navigation bar use your current



## Gallery



## Categories



**MATERIAL**

 **Banner**  
Displaying a banner within a list

 **Bottom app bar**  
Displays navigation and actions at the bottom



**CUPERTINO**

 **Activity Indicator**  
iOS-style activity indicators

 **Alerts**  
iOS-style alert dialogs



**REFERENCE STYLES & MEDIA**

 **Colors**  
All of the predefined colors

 **Typography**  
All of the predefined text styles

<https://flutter.github.io/samples/#/>

 **Bottom navigation**  
Bottom navigation with

 **Buttons**  
iOS-style buttons



# Options

One Line

Two Lines

## Lists

1 Item 1

2 Item 2

3 Item 3

4 Item 4

5 Item 5

6 Item 6

7 Item 7

8 Item 8

9 Item 9

10 Item 10

11 Item 11



COPY ALL

```
// Copyright 2019 The Flutter team. All rights reserved.  
// Use of this source code is governed by a BSD-style  
license that can be  
// found in the LICENSE file.  
  
import 'package:flutter/material.dart';  
  
import 'package:gallery/l10n/gallery_localizations.dart';  
  
enum ListDemoType {  
  oneLine,  
  twoLine,  
}  
  
class ListDemo extends StatelessWidget {  
  const ListDemo({Key key, this.type}) : super(key: key);  
  
  final ListDemoType type;  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        automaticallyImplyLeading: false,  
        title:  
        Text(GalleryLocalizations.of(context).demoListsTitle),  
      ),  
      body: Scrollbar(  
        child: ListView(  
          padding: EdgeInsets.symmetric(vertical: 8),  
          children: [  
            for (int index = 1; index < 21; index++)  
              ListTile(  
                leading: ExcludeSemantics(  
                  child: CircleAvatar(child:  
                    Text('$index'),  
                ),  
                title: Text(  
                  GalleryLocalizations.of(context).demoBottomSheetItem(index  
                ),  
              ),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

## Lists

- 1 Item 1
- 2 Item 2
- 3 Item 3
- 4 Item 4
- 5 Item 5
- 6 Item 6
- 7 Item 7
- 8 Item 8
- 9 Item 9
- 10 Item 10
- 11 Item 11



COPY ALL

```
// Copyright 2019 The Flutter team. All rights reserved.  
// Use of this source code is governed by a BSD-style  
license that can be  
// found in the LICENSE file.  
  
import 'package:flutter/material.dart';  
  
import 'package:gallery/l10n/gallery_localizations.dart';  
  
enum ListDemoType {  
  oneLine,  
  twoLine,  
}  
  
class ListDemo extends StatelessWidget {  
  const ListDemo({Key key, this.type}) : super(key: key);  
  
  final ListDemoType type;  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        automaticallyImplyLeading: false,  
        title:  
        Text(GalleryLocalizations.of(context).demoListsTitle),  
      ),  
      body: Scrollbar(  
        child: ListView(  
          padding: EdgeInsets.symmetric(vertical: 8),  
          children: [  
            for (int index = 1; index < 21; index++)  
              ListTile(  
                leading: ExcludeSemantics(  
                  child: CircleAvatar(child:  
                    Text('$index'),  
                ),  
                title: Text(  
                  GalleryLocalizations.of(context).demoBottomSheetItem(index  
                ),  
              ),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

## Lists

- 1 Item 1
- 2 Item 2
- 3 Item 3
- 4 Item 4
- 5 Item 5
- 6 Item 6
- 7 Item 7
- 8 Item 8
- 9 Item 9
- 10 Item 10
- 11 Item 11

COPY ALL

Lists

Search API Docs

Flutter &gt; material &gt; ListTile class

## CLASSES

AboutDialog

AboutListTile

AbsorbPointer

Accumulator

Action

ActionChip

ActionDispatcher

Actions

ActivateAction

AlertDialog

Align

Alignment

AlignmentDirectional

AlignmentGeometry

AlignmentGeometryTween

AlignmentTween

AlignTransition

AlwaysScrollableScrollView

AlwaysStoppedAnimation

AndroidView

Animatable

AnimateWith

# ListTile class



## CONSTRUCTORS

ListTile

## PROPERTIES

contentPadding

dense

enabled

isThreeLine

leading

onLongPress

onTap

selected

subtitle

title

trailing

hashCode

key

runtimeType

## METHODS



A list tile contains one to three lines of text optionally flanked by icons or other widgets, such as check boxes. The icons (or other widgets) for the tile are defined with the `leading` and `trailing` parameters. The first line of text is not optional and is specified with `title`. The value of

<https://api.flutter.dev/flutter/material/ListTile-class.html>

Default `TextStyles` that wrap the `title` and `subtitle` widget are reduced.

COPY ALL

Lists

Search API Docs

Flutter &gt; material &gt; ListTile class

## CLASSES

AboutDialog

AboutListTile

AbsorbPointer

Accumulator

Action

ActionChip

ActionDispatcher

Actions

ActivateAction

AlertDialog

Align

Alignment

AlignmentDirectional

AlignmentGeometry

AlignmentGeometryTween

AlignmentTween

AlignTransition

AlwaysScrollableScrollView

AlwaysStoppedAnimation

AndroidView

Animatable

AnimateWith

# ListTile class



## CONSTRUCTORS

ListTile

## PROPERTIES

contentPadding

dense

enabled

isThreeLine

leading

onLongPress

onTap

selected

subtitle

title

trailing

hashCode

key

runtimeType

## METHODS



A list tile contains one to three lines of text optionally flanked by icons or other widgets, such as check boxes. The icons (or other widgets) for the tile are defined with the `leading` and `trailing` parameters. The first line of text is not optional and is specified with `title`. The value of

<https://api.flutter.dev/flutter/material/ListTile-class.html>

Default text styles that wrap the `title` and `subtitle` widget are reduced.

87  
debugDescribeChildr...

Thanks for attending Flutter Interact 2019—you can still watch the session recordings!  
Flutter 1.12 is live! Check out the [latest announcement](#) and see what's new on the site.

Get started

Samples &amp; tutorials

Flutter Gallery [NEW] Sample apps on GitHub 

Cookbook

**Codelabs**

Tutorials

Development

▶ User interface

▶ Data &amp; backend

▶ Accessibility &amp; internationalization

▶ Platform integration

▶ Packages &amp; plugins

▶ Add Flutter to existing app

▶ Tools &amp; techniques

AndroidX migration

Testing &amp; debugging

Performance &amp; optimization

Deployment

<https://flutter.dev/docs/codelabs>

Re:

# Codelabs



## Contents

- [Basic Flutter layout concepts](#)
- [Write Your First Flutter App, part 1](#)
- [Write Your First Flutter App, part 2](#)
- [Building Beautiful UIs with Flutter](#)
- [Implicit Animations](#)
- [Adding Google Maps to a Flutter App](#)
- [Build a Photo Sharing App with Google Photos and Flutter](#)
- [Building a Cupertino app with Flutter](#)
- [Firebase for Flutter](#)
- [MDC 101 Flutter: Material Components \(MDC\) Basics](#)
- [MDC 102 Flutter: Material Structure and Layout](#)
- [MDC 103 Flutter: Material Theming with Color, Shape, Elevation, and Type](#)
- [MDC 104 Flutter: Material Advanced Components](#)
- [Other resources](#)

## Basic Flutter layout concepts

Use DartPad in a browser (no need to download Flutter or Dart!) to learn the basics of creating a Flutter layout.

## Write Your First Flutter App, part 1

! app that generates proposed names for a startup company. In part one, you'll use pairs of words at random and inserts them into an infinite scrolling list.

## Write Your First Flutter App, part 2

Create a simple mobile app that generates proposed names for a startup company. In part two, you'll



Thanks for attending Flutter Interact 2019—you can still watch the session recordings!  
Flutter 1.12 is live! Check out the latest announcement and see what's new on the site.

Get started

Samples & tutorials

Flutter Gallery [NEW]

Sample apps on GitHub

[Cookbook](#)

Codelabs

Tutorials

Development

User interface

Data & backend

Accessibility & internationalization

# Cookbook

Docs > Cookbook

## Contents

- Animation
- Design
- Forms
- Gestures
- Images
- Lists
- Maintenance

This cookbook contains recipes that demonstrate how to solve common problems while writing Flutter apps. Each recipe is self-contained and can be used as a reference to help you build up an application.

User interface

AndroidX migration

Testing & debugging

Performance & optimization

Deployment

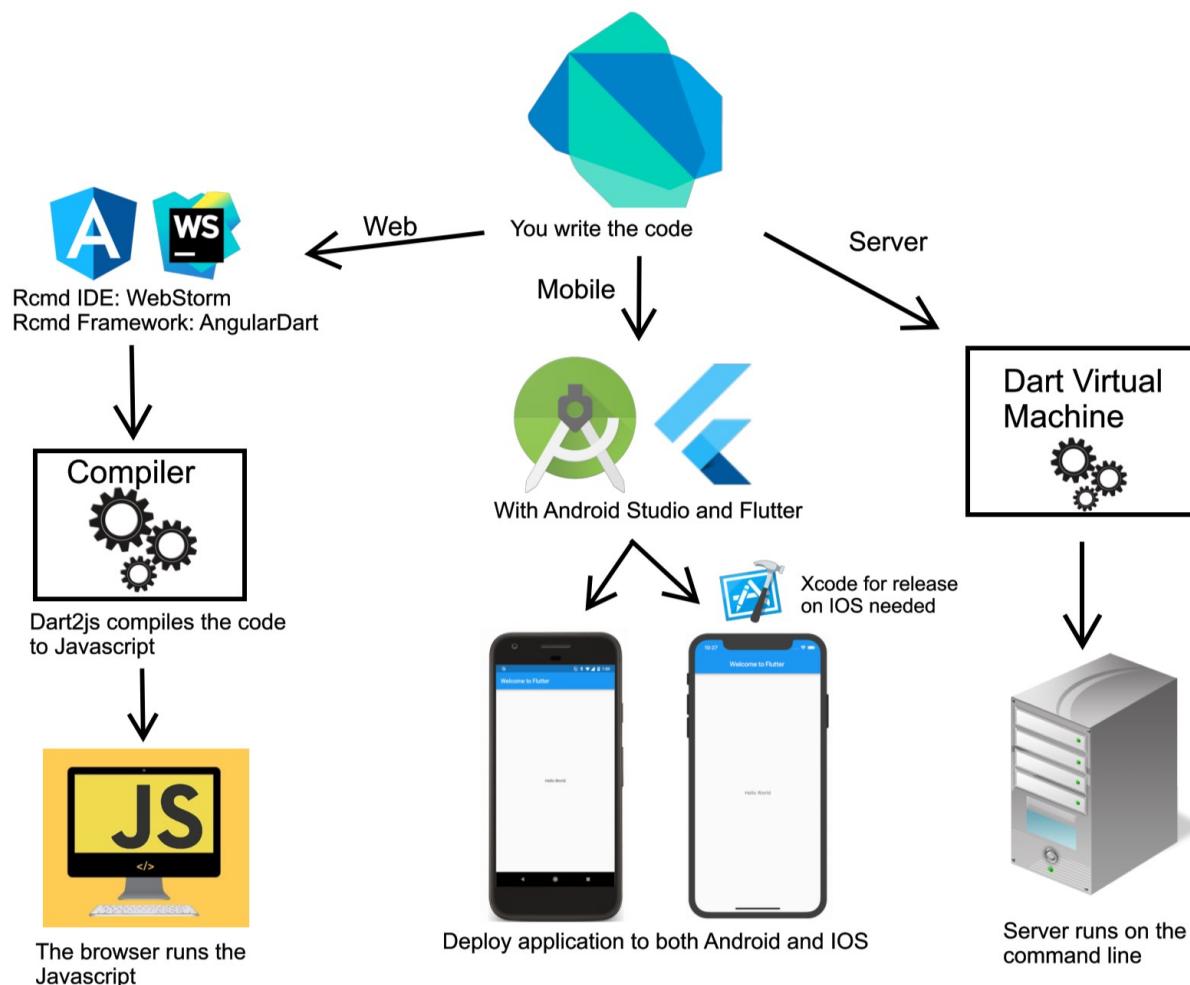
## Tools & techniques

- Integration
- Unit
- Widget

This cookbook contains recipes that demonstrate how to solve common problems while writing Flutter apps. Each recipe is self-contained and can be used as a reference to help you build up an application.

<https://flutter.dev/docs/cookbook>

# Flutter setup



## Android Studio

DOWNLOAD

WHAT'S NEW

USER GUIDE

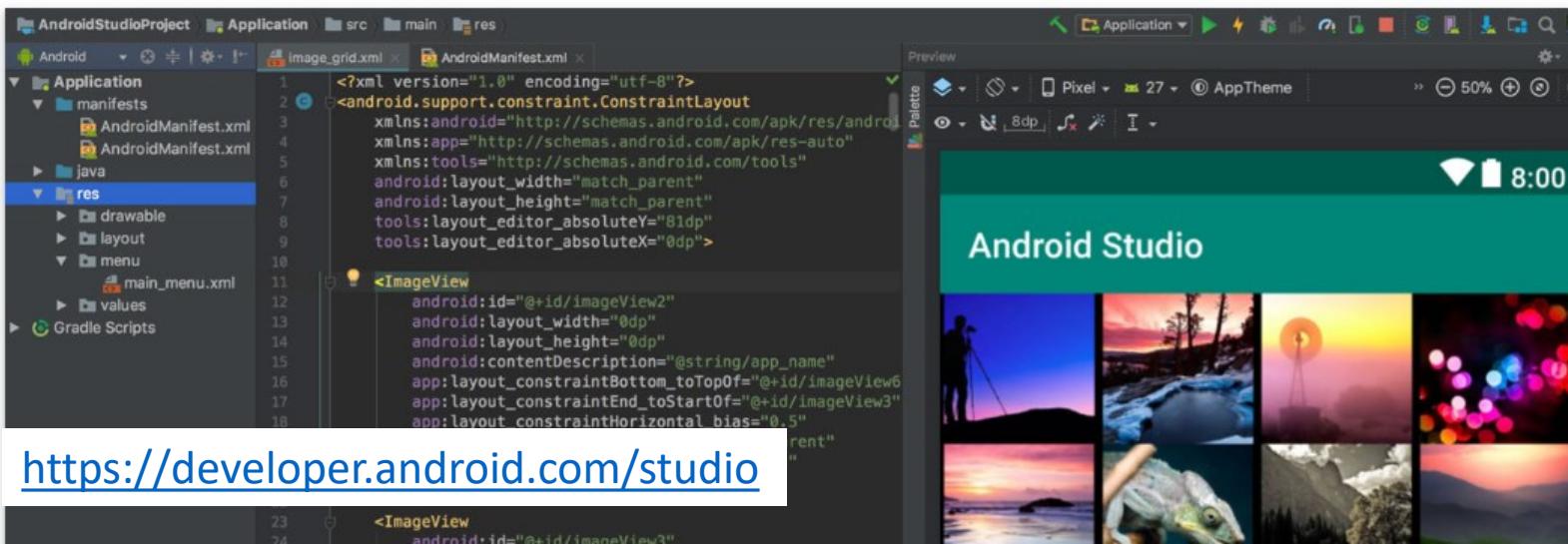
PREVIEW

# android studio

Android Studio provides the fastest tools for building apps on every type of Android device.

[DOWNLOAD ANDROID STUDIO](#)

3.5.3 for Windows 64-bit (718 MB)

[DOWNLOAD OPTIONS](#)[RELEASE NOTES](#)<https://developer.android.com/studio>

EXTENSIONS: MARKETPLACE

... JS

App.js JS

index.js JS

serviceWorker.js X

?

[ ]

...

# VS Code

Python 2019.6.24221 ↗ 54.9M ★ 4.5

Linting, Debugging (multi-threaded, ...)

Microsoft

Install

GitLens — Git sup... 9.8.5 ↗ 23.1M ★ 5

Supercharge the Git capabilities built...

Eric Amedio

Install

C/C++ 0.24.0 ↗ 23M ★ 3.5

C/C++ IntelliSense, debugging, and ...

Microsoft

Install

ESLint 1.9.0 ↗ 21.9M ★ 4.5

Integrates ESLint JavaScript into VS ...

Dirk Baeumer

Install

Debugger for Ch... 4.11.6 ↗ 20.6M ★ 4

Debug your JavaScript code in the C...

Microsoft

Install

Language Supp... 0.47.0 ↗ 18.7M ★ 4.5

Java Linting, Intellisense, formatting, ...

Red Hat

Install

vscode-icons 8.8.0 ↗ 17.2M ★ 5

Icons for Visual Studio Code

VSCode Icons Team

Install

Vetur 0.21.1 ↗ 17M ★ 4.5

Vue Coding for VS Code

Pineapple

Install

C# 12.1.0 ↗ 15.6M ★ 4

C# for Visual Studio Code (powered by ...)

Install



src &gt; JS serviceWorker.js &gt; register &gt; window.addEventListener('load') callback

checkValidServiceWorker(swUrl, config);

// Add some additional logging to localhost, p...

// service worker/PWA documentation.

navigator.serviceWorker.ready.then(() =&gt; {

} product

productSub

removeSiteSpecificTrackingException

removeWebWideTrackingException

requestMediaKeySystemAccess

sendBeacon

serviceWorker (property) Navigator.serviceWorke...

storage

storeSiteSpecificTrackingException

storeWebWideTrackingException

userAgent

vendor

}

function registerValidSW(swUrl, config) {

navigator.serviceWorker

.register(swUrl)

.then(registration =&gt; {

TERMINAL

...

1: node

+ [ ] ^ x

You can now view **create-react-app** in the browser.

Local: http://localhost:3000/

On Your Network: http://10.211.55.3:3000/

Note that the development build is not optimized.

Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript





EXTENSIONS: MARKETPLACE



JS App.js

JS index.js

JS serviceWorker.js ×



VS Code

Python 2019.6.24221 ⚡ 54.9M ★ 4.5  
Linting, Debugging (multi-threaded, ...)

Microsoft Install

GitLens — Git sup... 9.8.5 ⚡ 23.1M ★ 5  
Supercharge the Git capabilities built-in to VS Code.

Eric Amadio

C/C++ 0.24.0 ⚡  
C/C++ IntelliSense, debugging, refactoring, and more.

Microsoft

ESLint 1.9.0 ⚡ 21K  
Integrates ESLint JavaScript linter into VS Code.

Dirk Baeumer

Debugger for Ch... 4.11.6 ⚡  
Debug your JavaScript code in browser.

Microsoft

Language Supp... 0.47.0 ⚡ 18K  
Java Linting, Intellisense, formatters, and more.

Red Hat

vscode-icons 8.8.0 ⚡  
Icons for Visual Studio Code.

VSCodium 0.24.0 ⚡  
Vetur 0.24.0 ⚡  
Vue 3 Coding for VS Code

Pineapple 0.1.0 ⚡  
C# 12.1.0 ⚡  
C# for VS Code (powered by Roslyn)



```
src > JS serviceWorker.js > register > window.addEventListener('load') callback
39   |           checkValidServiceWorker(swUrl, config);
40   |
41   |           // Add some additional logging to localhost, p...
42   |           // service worker/PWA documentation.
43   |           navigator.serviceWorker.ready.then(() => {
44   |               product
```

EXTENSIONS: INSTALLED



@installed flutter

**Flutter** 3.8.0

Flutter support and debugger for Visual Studio Code.

Dart Code

1M ★ 5

**Flutter Stateful Widget Generator** 0.0.2

Generates a stateful widget into a new file.

8K

**Flutter Tree** 1.0.0

Extension for Flutter to build basic widget tree.

Marcelo Velasquez

18K ★ 5

**Flutter Widget Snippets** 2.0.0

A set of helpful widget snippets for day to day Flutter development.

Alexis Villegas Torres

128K ★ 5

**goVSCode Flutter & Dart** 0.0.4

Dedicated to making our lives easier. Used by amazing Tap developer...

Tap Payments

3K

On Your Network: http://10.211.55.3:3000/

Note that the development build is not optimized.



## ► Packages & plugins

▶ Add Flutter to existing app

## ▼ Tools & techniques

Android Studio & IntelliJ

[Visual Studio Code](#)

## ▼ DevTools

Overview

Install from Android Studio &  
IntelliJ

Install from VS Code

Install from command line

Flutter inspector

Timeline view

Memory view

Performance view

Debugger

Logging view

## ▼ Flutter SDK

Upgrading

Releases

Breaking changes

Release notes

Hot reload

Code formatting

AndroidX migration

Testing & debugging



Performance & optimization



Dep <https://flutter.dev/docs/development/tools/vs-code>



Get started ▾

Samples &amp; tutorials ▾

Development ^

- ▶ User interface
- ▶ Data & backend
- ▶ Accessibility & internationalization
- ▶ Platform integration
- ▶ Packages & plugins
- ▶ Add Flutter to existing app
- ▼ Tools & techniques

Android Studio &amp; IntelliJ

Visual Studio Code

▼ DevTools

Overview

Install from Android Studio &amp; IntelliJ

Install from VS Code

Install from command line

[Flutter inspector](#)

Timeline view

Memory view

Performance view

Debugger

Logging view

- Selecting a target device
- Fast edit and refresh development cycle
- Snippets

## Install and run DevTools from VS Code

[Docs](#) > [Development](#) > [Tools](#) > [DevTools](#) > Install and run DevTools from VS Code

## Using the Flutter inspector

[Docs](#) > [Development](#) > [Tools](#) > [DevTools](#) > Using the Flutter inspector

▶ <https://flutter.dev/docs/development/tools/vs-code>



Android Studio & IntelliJ

Visual Studio Code

▼ DevTools

Overview

Install from Android  
Studio & IntelliJ

Install from VS Code

Install from command  
line

Flutter inspector

Timeline view

Memory view

Performance view

Network view

Debugger

App size tool

▶ Flutter SDK

Hot reload

Code formatting

# DevTools

[Docs](#) > [Development](#) > [Tools](#) > DevTools

Topics:

- [Overview](#)
- [Install from Android Studio & IntelliJ](#)
- [Install from VS Code](#)
- [Install from command line](#)
- [Flutter inspector](#)
- [Timeline view](#)
- [Memory view](#)
- [Performance view](#)
- [Network view](#)
- [Debugger](#)
- [App size tool](#)

File Edit Selection View Go Run Terminal Help main.dart - myapp - Visual Studio Code

Which DevTools page?

lib > main.dart > \_MyH

102 // axi  
103 // hor  
104 mainAx  
105 childr  
106 Rais  
107 onPr  
108 chil  
109 ), // Raisedbutton  
110 Text(  
111 | You have pushed the button this many times:',  
112 | ), // Text  
113 Text(  
114 | '\$\_counter',  
115 | style: Theme.of(context).textTheme.headline4,  
116 | ), // Text  
117 | ], // <Widget>[]  
118 | ), // Column  
119 | ), // Center  
120 floatingActionButton: FloatingActionButton(  
121 onPressed: \_incrementCounter,  
122 tooltip: 'Increment'

Ctlr + Alt + D

WATCH

CALL STACK

main.dart - myapp - Visual Studio Code

Which DevTools page?

Open Inspector Page

Open Timeline Page

Open Memory Page

Open Performance Page

Open Network Page

Open Logging Page

Open DevTools in Web Browser

RaisedButton

BREAKPOINT

All Exce  
 Uncaug

script) 18:14

# DevTools - inspector ( in web browser)

The screenshot shows the Dart DevTools Inspector interface running in a web browser. On the left, the **Flutter Inspector** tab is active, displaying the widget tree. The tree starts with **[root]**, then **MyApp**, **MaterialApp**, **MyHomePage**, **Scaffold**, **Center**, **Column**, **RaisedButton**, and finally **Text**. A context menu is open over one of the **Text** nodes, with the option **Map widgets to properties** highlighted.

The **Details Tree** panel on the right shows the properties of the selected **Text** widget. The properties listed include:

- "0"
- debugLabel: (englishLike display1 2014).merge(blackRedmond headline)
- inherit: false
- color: #8a000000
- backgroundColor: null
- family: Segoe UI
- familyFallback: null
- size: 34.0
- weight: 400
- style: null
- letterSpacing: null
- wordSpacing: null
- baseline: alphabetic
- height: null
- locale: null
- foreground: null
- background: null
- decorationColor: null
- decoration: TextDecoration.none
- decorationThickness: null

On the right side of the browser window, the **Flutter Demo Home Page** is visible, showing a disabled button labeled "Disabled Button". Below it, a message says "You have pushed the button this many times: 0".

# DevTools – running in emulator / device

The screenshot shows the Dart DevTools interface. A large blue arrow points from the text "More information on device" in the bottom right of the DevTools window towards the top right of the image, where a screenshot of an Android emulator is displayed.

**Dart DevTools**

Flutter Inspector Timeline Memory Performance Debugger Network Logging App Size

Select Widget Mode Refresh Tree Details Tree Layout Explorer

Slow Animations Debug Paint Paint

[root] MyApp MaterialApp MyHomePage Scaffold Center Column RaisedButton Text Text Text AppBar Text FloatingActionButton Icon

More information on device

Text "0"  
debugLabel: (lerp(englishLike display1 2014 0.7 englishLike display1 2014)).merge  
inherit: false  
color: #8a000000

Text size: 34.0 weight: 400 style: null letterSpacing: null wordSpacing: null baseline: alphabetic height: null locale: null foreground: null background: null decorationColor: null decoration: TextDecoration.none

**Android Emulator - Android\_Accelerated\_x86\_Oreo:5554**

6:22 DEBUG

**Flutter Demo Home ...**

**Disabled Button**  
You have pushed the button this many times:  
-1

+



Building A Blog With Eleventy And Netlify CMS – Pa...

21ST SEPTEMBER '20



Building A Blog With Eleventy And Netlify CMS &#8230;

20TH SEPTEMBER '20



Command-line Interfaces (CLIs) With React And Ink ...

9TH SEPTEMBER '20

## TAGS

Angular

Bootstrap

CSS

Docker

Flutter

Gatsby.js

GraphQL

HTML

Ionic 2

JavaScript

Machine Learning

Meteor

Mobile Development

More videos

- Google Flutter FULL COURSE 2020
- Rethinking reactivity
- Getting Started With strapi
- Flutter TOP 10 WIDGETS FOR FLUTTER DEVELOPERS
- BLoC Pattern

0:00 / 17:27

CodingTheSmartWay

FLUTTER

POSTED BY SEBASTIAN — 3RD NOVEMBER '19

## Getting Started With Flutter – (2) Project Structure

Getting Started With Flutter – (2) Project Structure

<https://codingthesmartway.com/getting-started-with-flutter-project-structure/>

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

# Some recommend resources

# Official: Dart & Flutter

The screenshot shows the official Flutter website at <https://flutter.dev/>. The page features a large banner at the top with a blue gradient background. On the left side of the banner, there's a small image of a mobile application showing a city street with cars and a 'REFRESH' button. On the right side, a smartphone displays a similar scene with text overlaying it: 'Performing hot reload...', 'Reloaded 1 of 8 libraries in 384ms.' Below the banner, the main content area has a light blue background. A large, bold, white text 'Productively build apps' is centered. To the right of this text is a white circle containing a black clock icon. At the very bottom of the page, there's a footer bar with links for 'Docs', 'Showcase', 'Community', a search icon, social media icons for Twitter, YouTube, and GitHub, and a prominent blue 'Get started' button.

Flutter

Docs Showcase Community

Get started

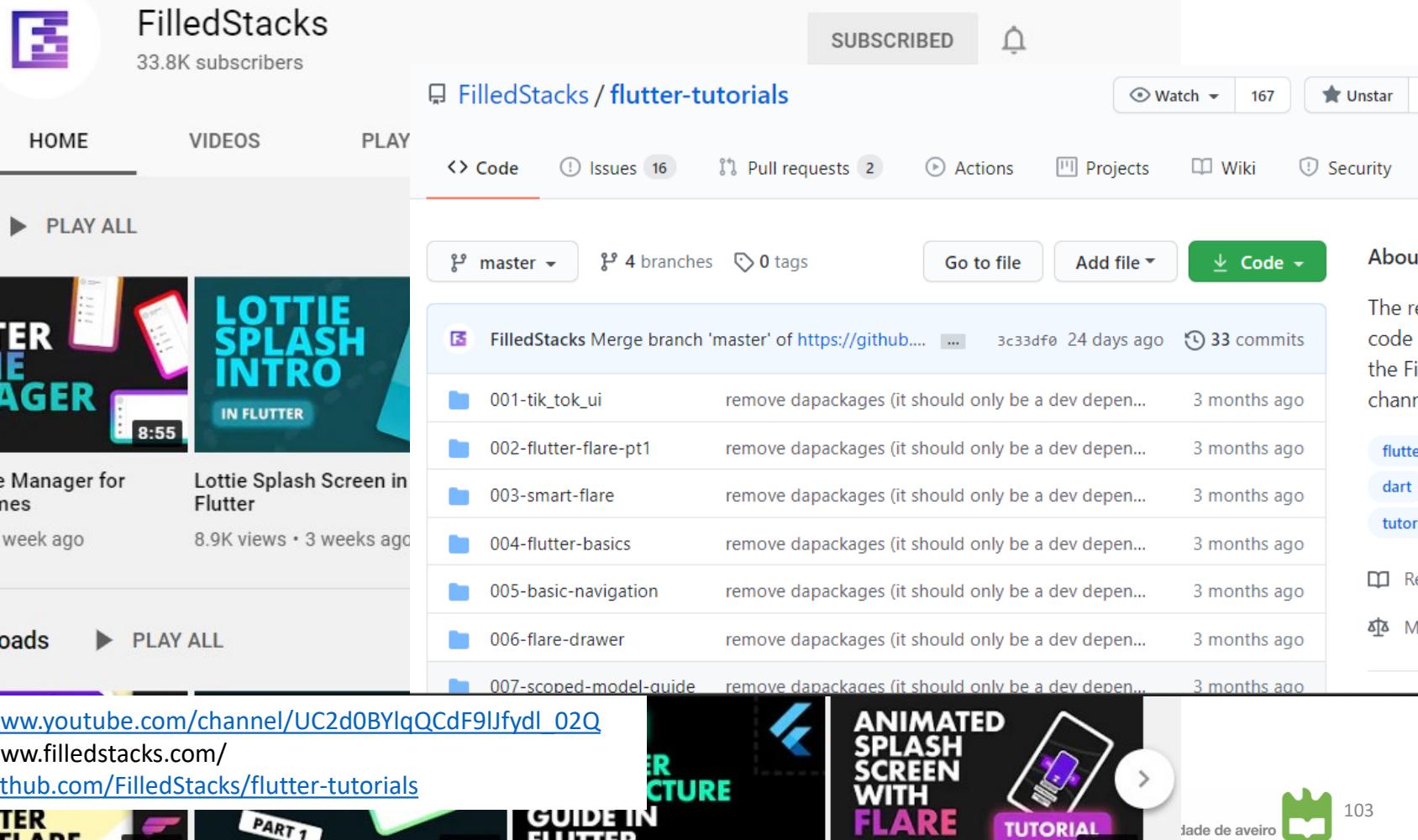
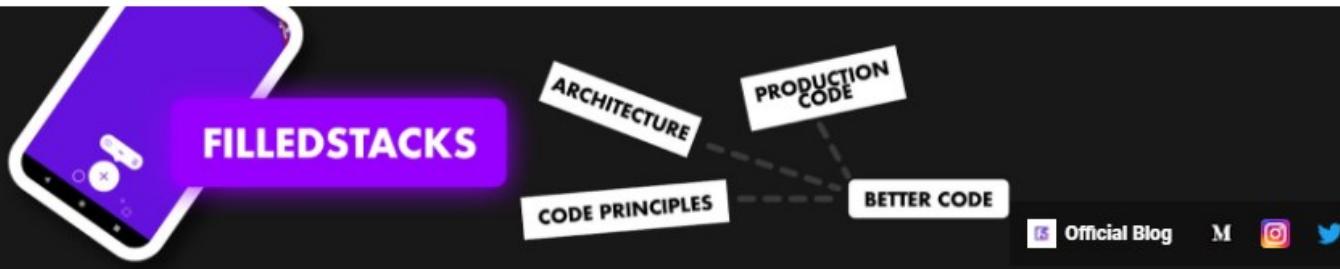
Thanks for attending [Flutter Interact 2019](#)—you can still watch the [session recordings](#)! Flutter 1.12 is live! Check out the [latest announcement](#) and see what's new on the site.

REFRESH

Performing hot reload... Reloaded 1 of 8 libraries in 384ms.

Productively build apps

<https://flutter.dev/>



[Learn ▾](#)[Flutter ▾](#)[Store](#)[Join Chat](#) [Search](#)

...

[Sign In](#)[Create a free account](#)

# Flutter Top Tutorials

Learn multi-platform development with Flutter using our high-quality step-by-step tutorials!

## Newest Tutorials

### Your First Flutter App

Getting Started



If you're a complete beginner to Flutter development and wondering how to get started, this is the course for you.



Sep 8 2020 · Video Course (3 hrs, 54 mins)

### Creating Reusable Custom Widgets in Flutter



<https://www.raywenderlich.com/flutter>

Learn how to design and create your own custom widgets in Flutter that you can use in any of your projects or share with the world.

### Mastering Git, 1st edition, Is Now Available in Full!

Announcements



Mastering Git, 1st Edition, is now available in full with five new chapters! This book is the easiest and fastest way to get hands-on experience...

Aug 19 2020 · Article (5 mins)

### Will raywenderlich.com Books Be Updated in 2020?

Announcements



We're happy to announce we'll be updating many of our books in the coming months — and we have some great new titles underway!

Aug 17 2020 · Article (10 mins)

Learn how to identify and resolve common errors that Flutter developers face during app development.

Aug 4 2020 · Video Course (1 hr, 18 mins)

### Beginning Flutter Debugging

Core Concepts



### Flutter Interview Questions and Answers

Core Concepts



In this article, you'll work through a series of Flutter and Dart job interview questions and answers.

# LEARN & DISCOVER PUSHER

**447** tutorials to help you build great apps

**CHANNELS****BEAMS**

Enter a search term

**Search**

ANDROID, BEAMS, JAVA ...

## Send Push Notifications to your Android app

We will walk you through the steps to send push notifications to

<https://pusher.com/tutorials>

[READ MORE →](#)

ANDROID, BEAMS, GO, KOTLIN ...

## Notify Slack users of push notification status with webhooks

Webhooks make it extremely easy to send push notifications to your iOS and Android users with a single request. I...

[READ MORE →](#)



## LEARN FLUTTER

absolute beginners

### Flutter Tutorial

- ① [Flutter - Home](#)
- ② [Flutter - Introduction](#)
- ③ [Flutter - Installation](#)
- ④ [Creating Simple Application in Android Studio](#)
- ⑤ [Flutter - Architecture Application](#)
- ⑥ [Introduction to Dart Programming](#)
- ⑦ [Flutter - Introduction to Widgets](#)
- ⑧ [Flutter - Introduction to Layouts](#)
- ⑨ [Flutter - Introduction to Gestures](#)
- ⑩ [Flutter - State Management](#)
- ⑪ [Flutter - Animation](#)
- ⑫ [Flutter - Writing Android Specific Code](#)



## Flutter Tutorial

[!\[\]\(58f43f206e70dc5eaeb3fa36e74f56dd\_img.jpg\) PDF Version](#)[!\[\]\(2b62736cbb4c4d5e1fbfefd164999cc6\_img.jpg\) Quick Guide](#)[!\[\]\(6ebc67a815c259aa96ca434187239a8c\_img.jpg\) Resources](#)[!\[\]\(3541849cd4bd19ccbe199e4ee558d31c\_img.jpg\) Job Search](#)[!\[\]\(6c42de673d308d4fb46523ddda1df503\_img.jpg\) Discussion](#)

Flutter is an open source framework to create high quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides a simple, powerful, efficient and easy to understand SDK to write mobile application in Google's own language, *Dart*. This tutorial walks through the basics of Flutter framework, installation of Flutter SDK, setting up Android Studio to develop Flutter based application, architecture of Flutter framework and developing all type of mobile applications using Flutter framework.

## Audience

This tutorial is prepared for professionals who are aspiring to make a career in the field of mobile applications. This tutorial is intended to make you comfortable in getting started with Flutter framework and its various functionalities.

<https://www.tutorialspoint.com/flutter/index.htm>

⑬ [Flutter - Accessing REST API](#)

This tutorial is written assuming that the readers are already aware about what a Framework is and the readers have a sound knowledge on Object Oriented Programming and basic knowledge on



# Curated list of links

- Curated by me
  - Not meant JUST for classes
  - Useful for a programmer
- Try to get you a filtering on typical “google” search on subjects
  - details on widgets
  - on building UI
  - Technical “curiosities”

Some links that might give some context...

## 10 Flutter Tips I Learned After 3 Years of Flutter Development

Things that helped boost my productivity



Robert Brunhage

Follow



Feb 9 · 4 min read



<https://betterprogramming.pub/10-flutter-tips-i-learned-after-3-years-of-flutter-development-7a7dbb697fe2>

Some links that might give some context...

## Popular apps that use Dart



Manpreet Singh Sep 8 · 3 min read ★



...

<https://preettheman.medium.com/popular-apps-that-use-dart-81592aea1a39>

# The END

