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[7] programming language for apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, backend and web applications.[8]

Dart is an object-oriented, class defined, garbage-collected language[9] 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.[10]

Contents [hide]

Dart

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

https://en.wikipedia.org/wiki/Dart_(programming_language)

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[7] programming language for apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, backend and web applications.[8]

Dart is an object-oriented, class defined, garbage-collected language[9] 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.[10]

Contents [hide]

Dart

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

https://en.wikipedia.org/wiki/Dart_(programming_language)

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)",[2] 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).[3] 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]

History [edit]

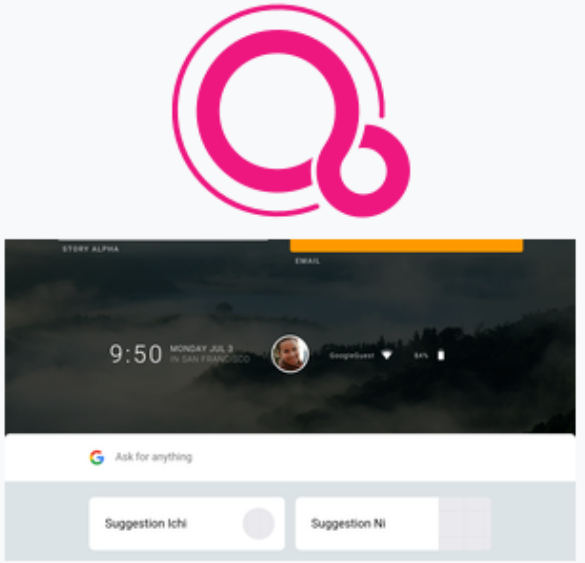
In August 2016, media outlets reported on a mysterious codebase post published on GitHub, revealing that Google was developing a new operating system called "Fuchsia". No official announcement was made, but inspection of the code suggested its capability to run on universal devices, including "dash infotainment 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)[4] rather than on the Linux kernel.[5][6][7][8][9]

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

t https://en.wikipedia.org/wiki/Google_Fuchsia

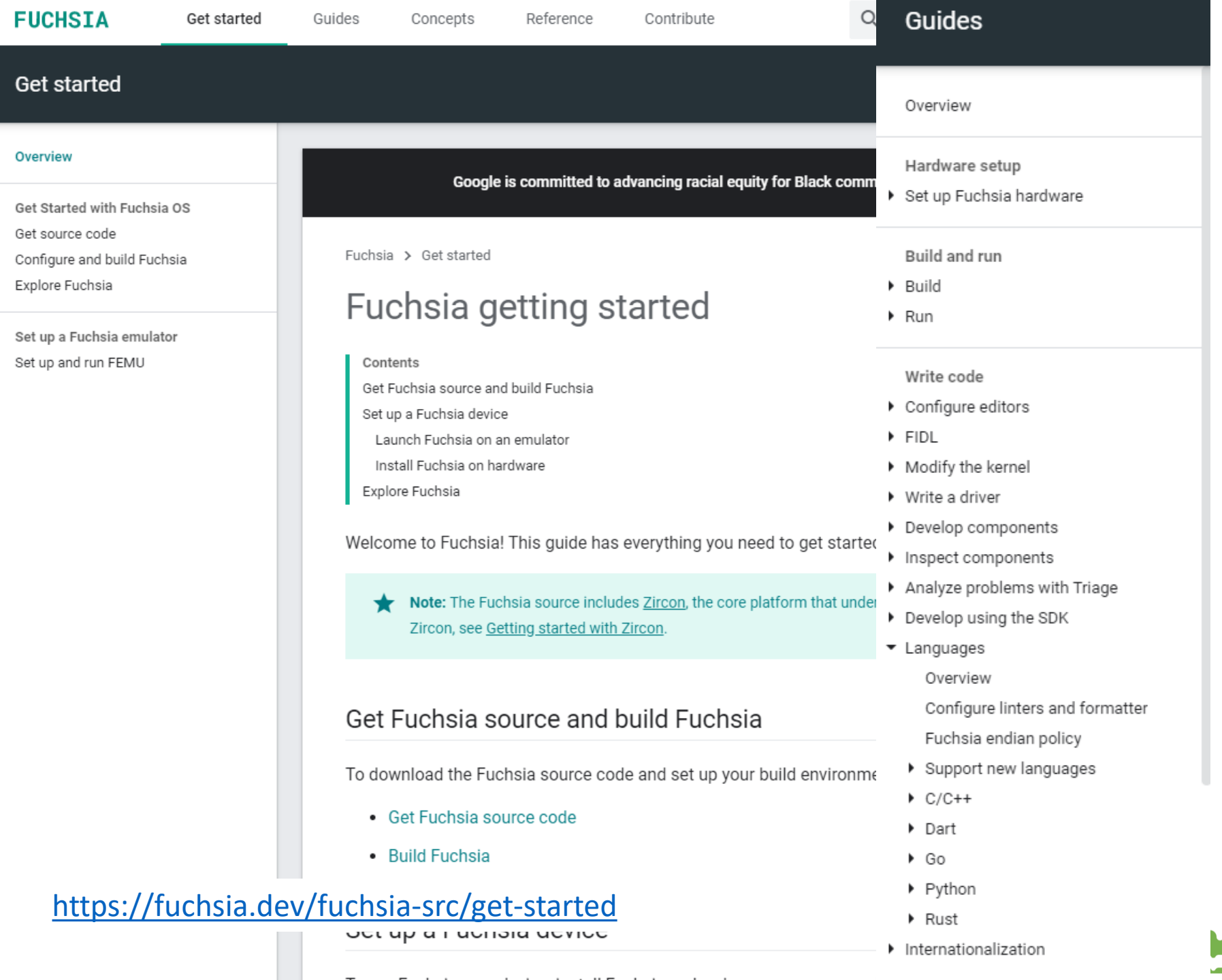
project's seemingly close ties to Android, with some speculating that Fuchsia might be

Fuchsia



Screenshot of the Google Fuchsia GUI

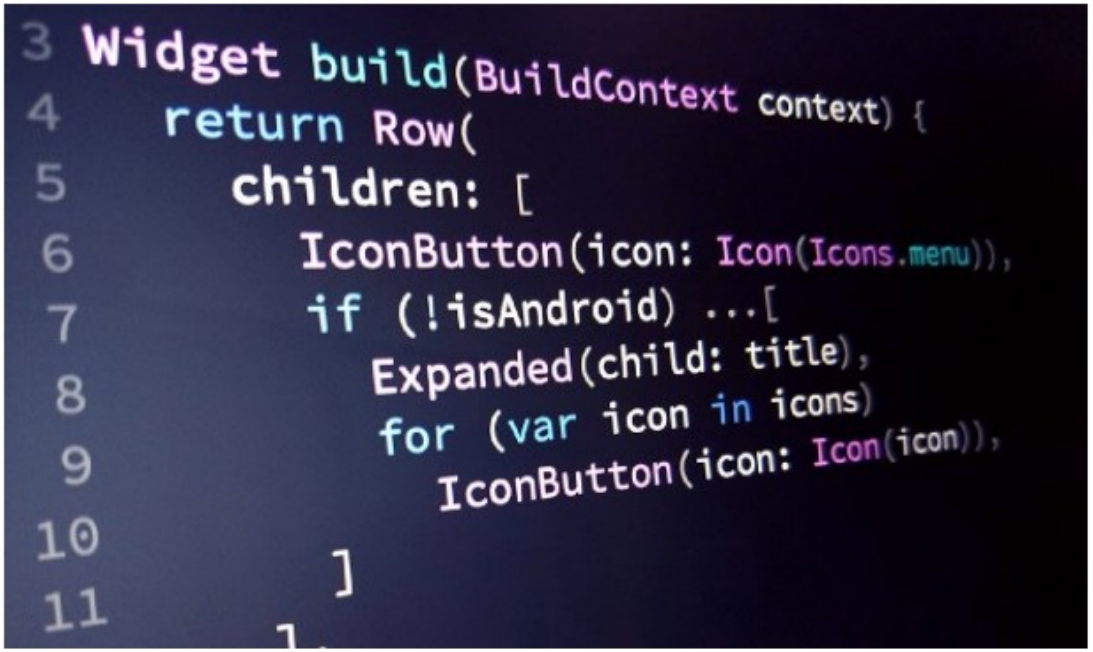
Developer	Google
Written in	C, C++, Dart, Go, Rust, Python[1]
Working state	Current
Source model	Open source
Initial release	August 15, 2016; 4 years ago
Repository	fuchsia.googlesource.com
Available in	English
Platforms	ARM64, x86-64
Influenced by	Android
Default user interface	Ermine
License	BSD, MIT,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



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

Making Dart a Better Language for UI

Bob Nystrom Follow
Mar 19, 2019 · 14 min read



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

Making Dart a Better Language

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

snippet03.html hosted with ❤ by GitHub

You had to write something like:

```
doc.beginTag("p");
doc.appendText("I am an ");
doc.beginTag("strong");
doc.appendText("exciting");
doc.endTag("strong");
doc.appendText("paragraph!");
doc.endTag("p");
```

snippet04.java hosted with ❤ by GitHub

Fortunately, modern languages and APIs ar
though statements are imperative, *expressi*
the above code is gnarly, this one is about c

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

snippet05.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.)

```
MaterialApp(
  title: 'Welcome to Flutter',
  home: Scaffold(
    appBar: AppBar(title: Text('Welcome to Flutter')),
    body: Column(
      children: [
        Text('Hello World'),
        Text('This is Flutter'),
        Text('Written in Dart')
      ]
    )
  )
)
```

snippet06.dart 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

```
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

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

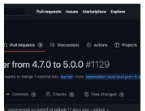
Preparing for the next major release, where Dart only supports sound null safety — Over the last four years, we've evolved Dart into a fast,...

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

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 defa
- Documented specifying a file path wl
- Rewrote Learning Dart as a JavaScri
- 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 URIs 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.

Contents

https://dart.dev/guides/whats-new

Dart language evolution

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.

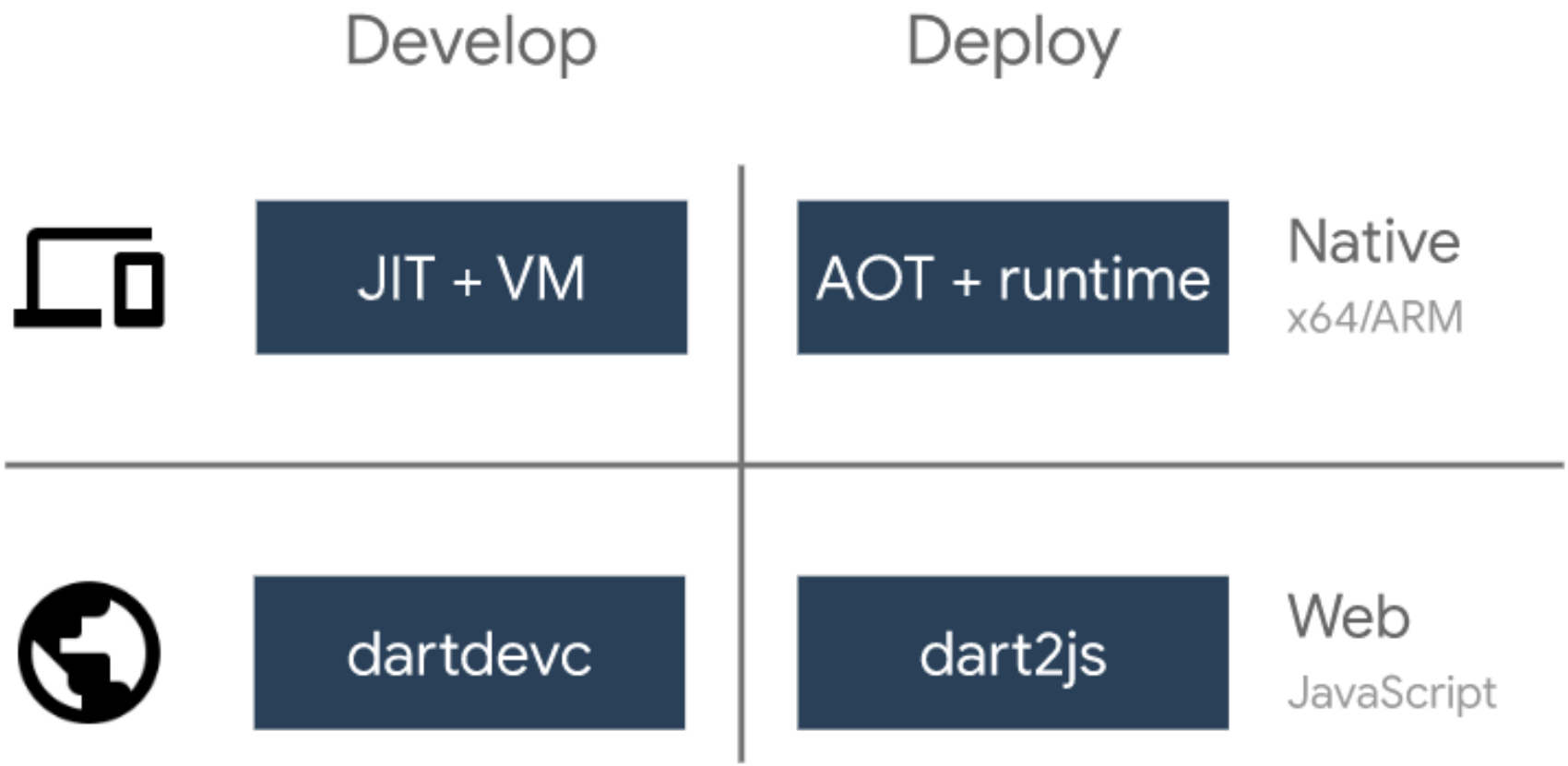
Dart 2.19 also introduced support for unnamed libraries. Library directives, used for appending library-level doc comments and annotations, can and should now be written without a name:

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

Dart 2.18

Released 30 August 2022 | Dart 2.18 announcement

Contents

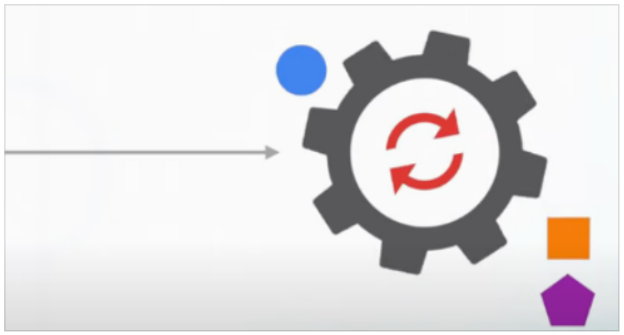


https://dart.dev/platforms

Dart

A client-optimized language for fast apps on any platform

ANNOUNCEMENTS ARCHIVE | DART.DEV Follow



Dart asynchronous programming: Streams

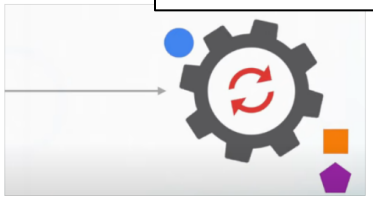
This article covers one of the fundamentals of reactive programming: streams, which are objects of type Stream.

Flutter Devrel Medium
Apr 14 · 9 min read

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



Asynchronous programming: futures, async, await

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 cover https://dart.dev/codelabs/async-await

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

Dart Overview Docs Community Try Dart Get Dart

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

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

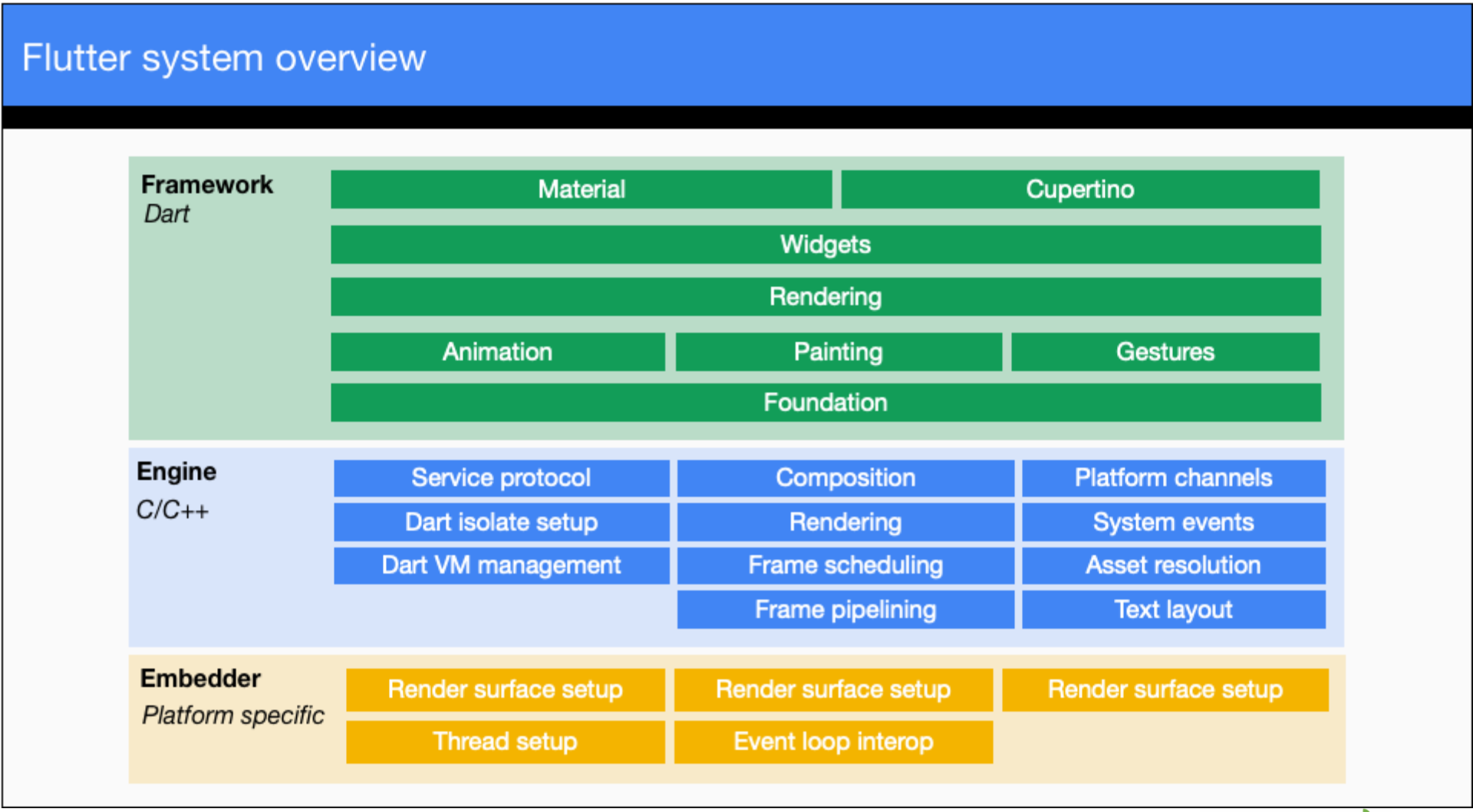
Dart mobile, command-line, and server apps running on the Dart Native platform can use the dart:ffi library to call native C APIs, and to read, write, allocate, and deallocate native memory. *FFI* stands for *foreign function interface.* Other terms for similar functionality include *native interface*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

API documentation is available in the dart:ffi API reference.

Examples

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

How?


Flutter system overview
Framework
Dart
Material
Cupertino
Widgets
Rendering
Animation
Painting
Gestures
Foundation
Engine
C/C++
Service protocol
Composition
Platform channels
Dart isolate setup
Rendering
System events
Dart VM management
Frame scheduling
Asset resolution
Frame pipelining
Text layout
Embedder
Platform specific
Render surface setup
Render surface setup
Render surface setup
Thread setup
Event loop interop

A tour of the Dart language

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.

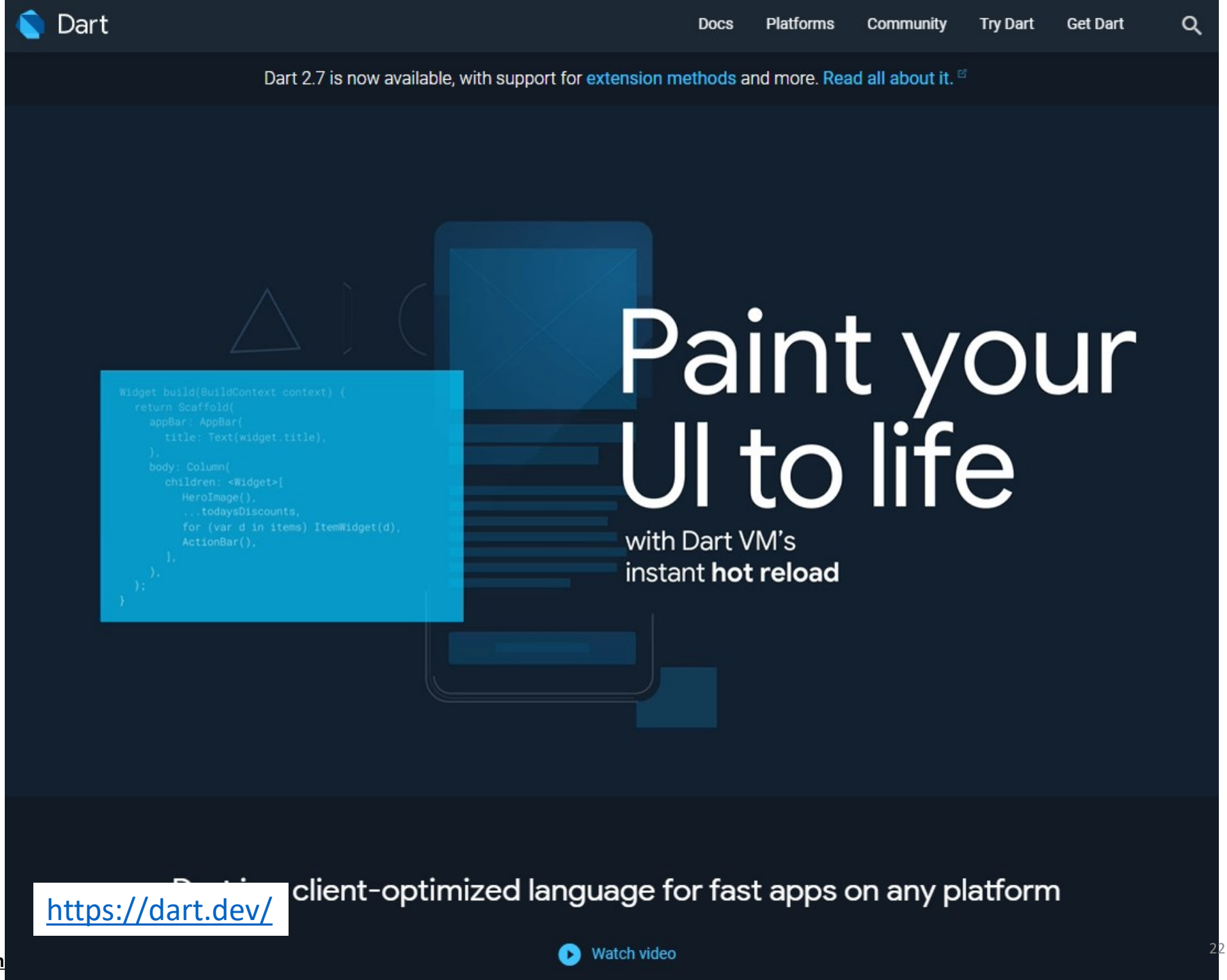
`int`
A type. Some of the other built-in types are `String`, `List`, and `bool`.

`print()`
A handy way to display output.

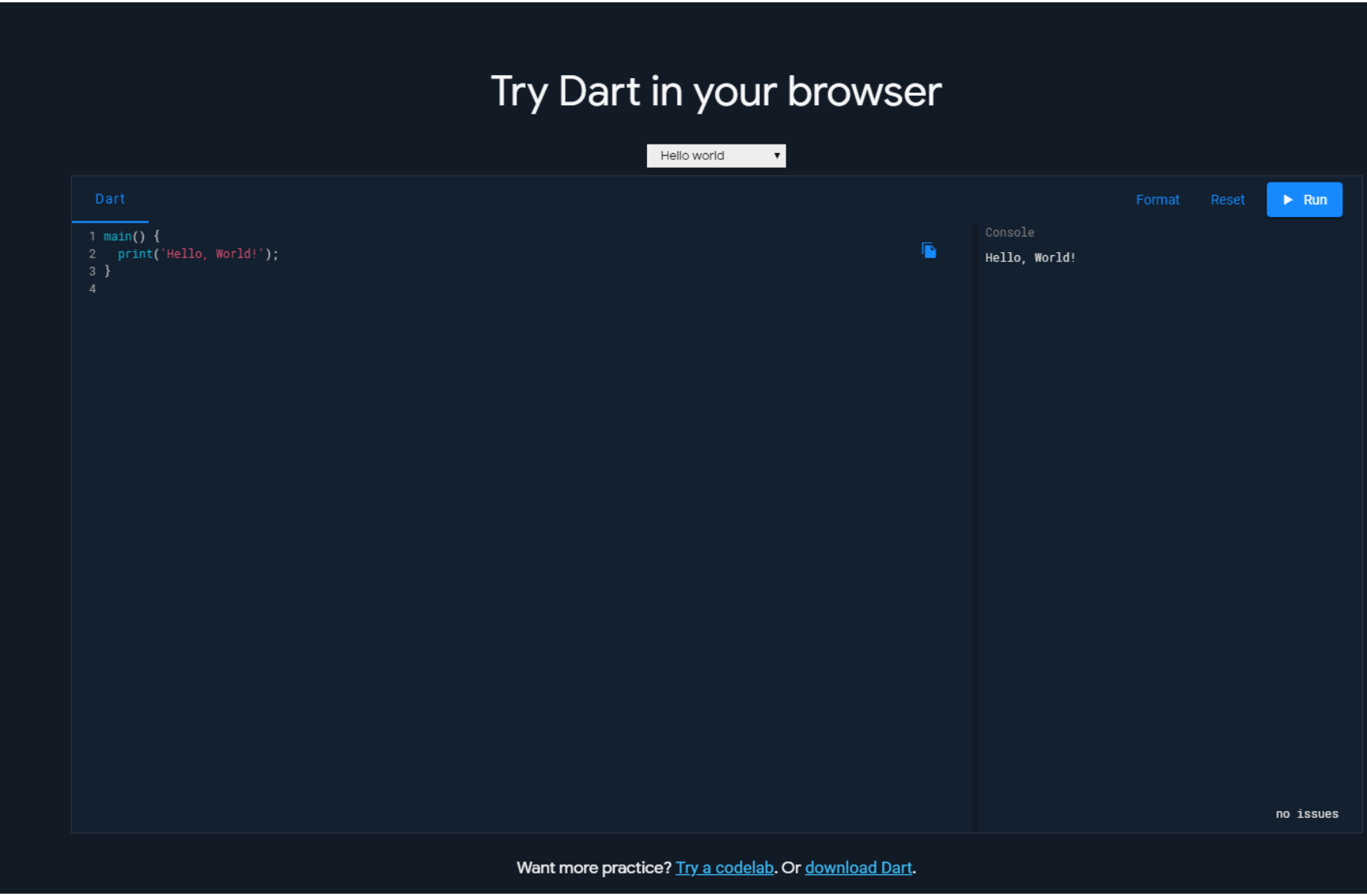
`'...'` (or `"..."`)

Contents

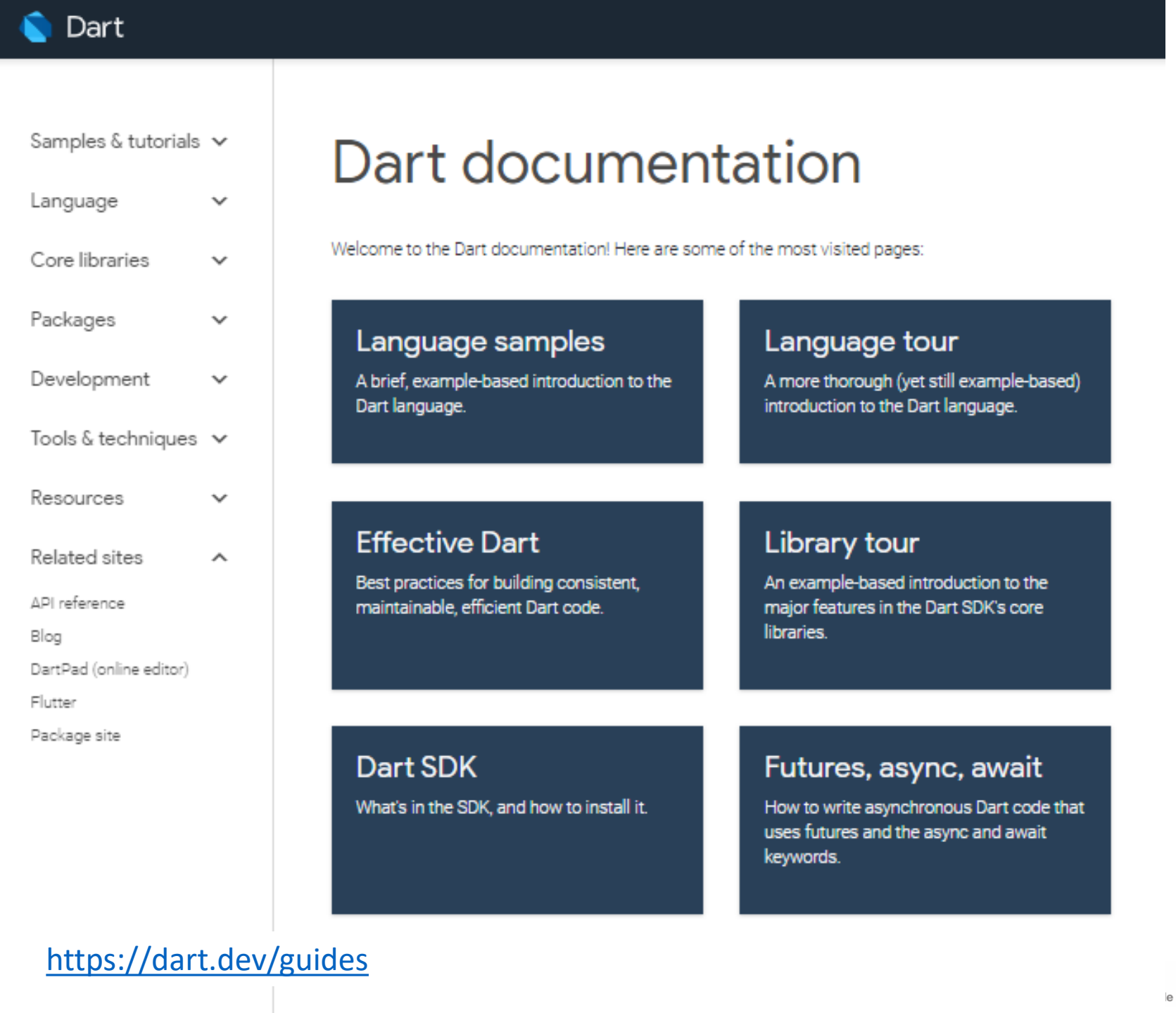
https://dart.dev/guides/language/language-tour



https://dart.dev/



https://dartpad.dev/



https://dart.dev/guides

Samples & tutorials
Language
Effective Dart
Overview
Style
Documentation
Usage
Design
Extension methods
Sound null safety
Evolution
Specification
Tour
Type system
Core libraries
Packages
Development
Tools & 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

- DON'T import libraries that are inside the `src` directory of

Design

- AVOID abbreviations.

Contents

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

Language samples

This collection is not exhaustive—it's just a brief introduction to the language for people who like to learn by example. You might also want to check out the language and library tours, or the Dart cheatsheet codelab.

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

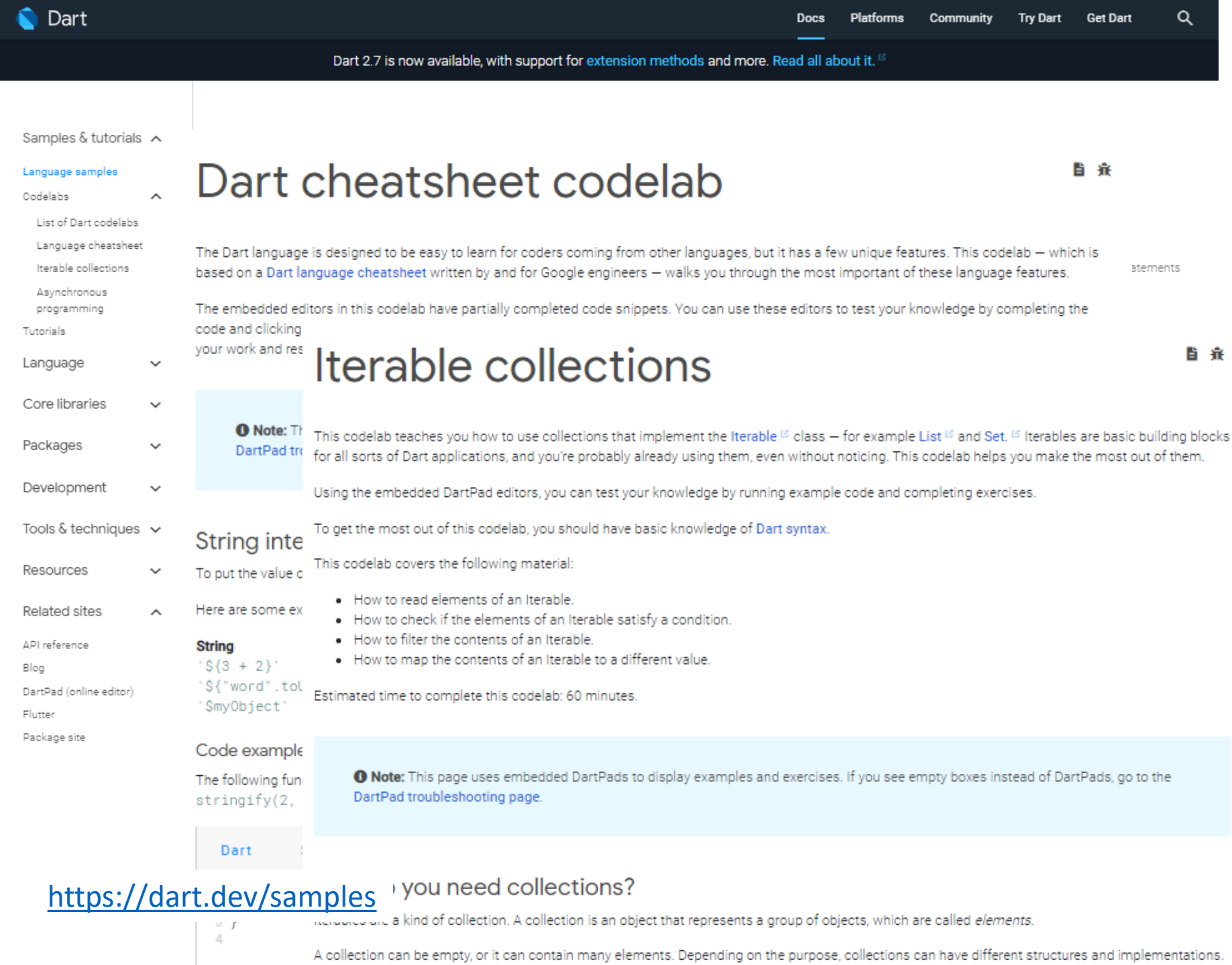
Even in type-safe Dart code, most variables don't need explicit types, thanks to type inference:

```
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'
};
```

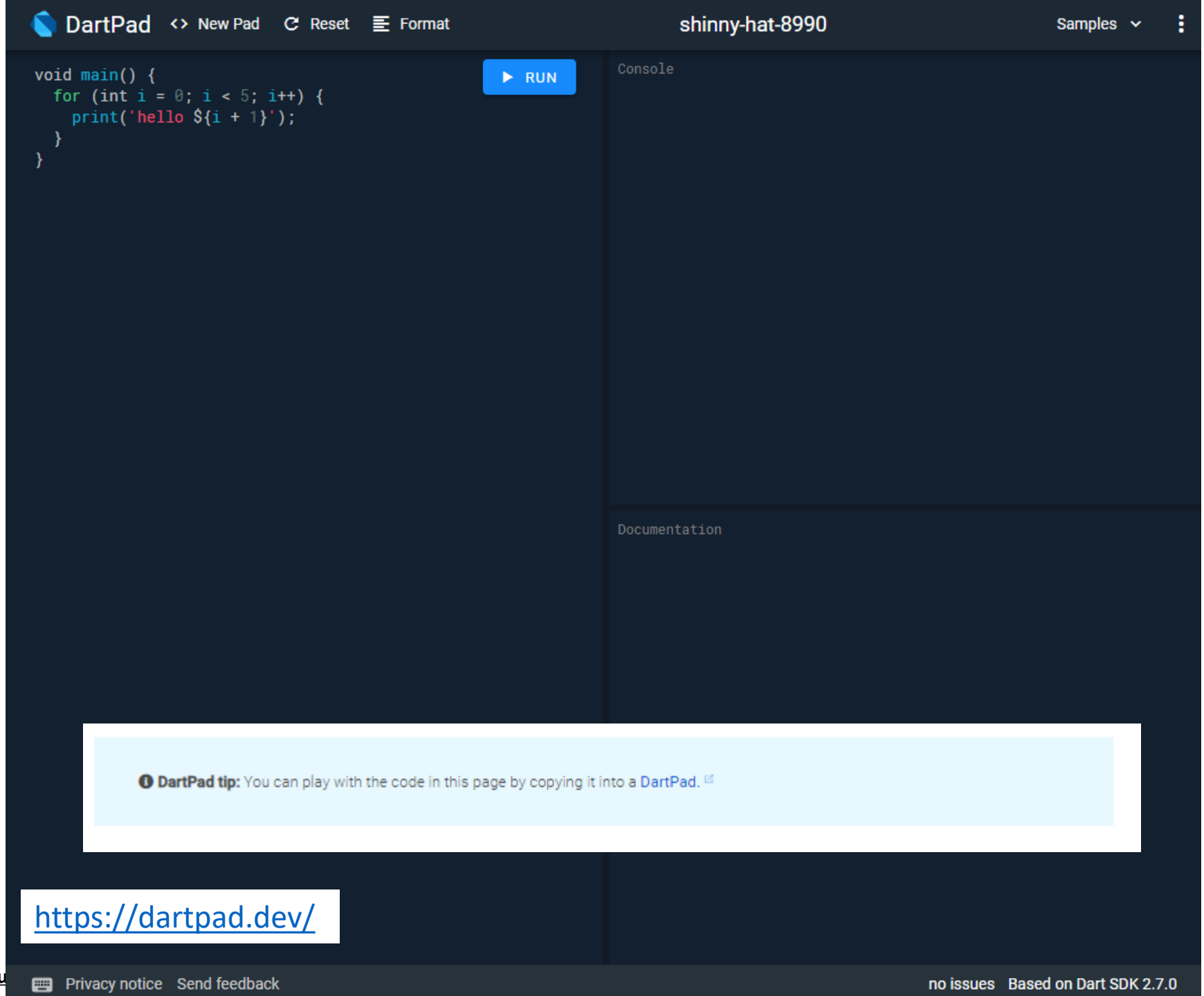
including default values, the `final` and `const` keywords, and static types.

Control flow statements

https://dart.dev/samples



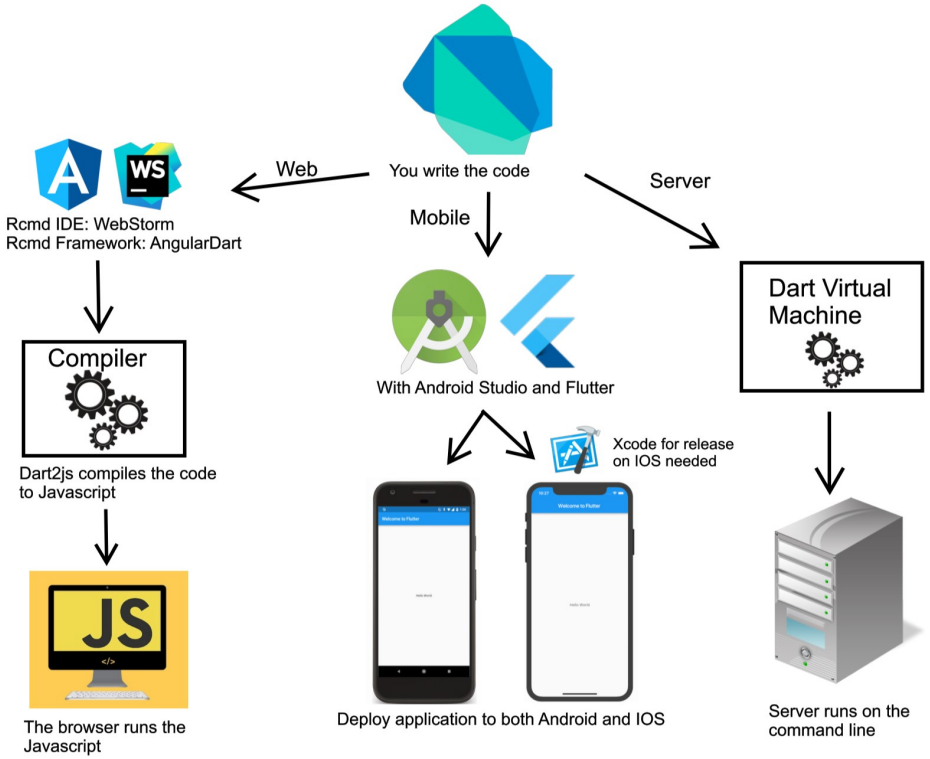
https://dart.dev/samples


DartPad
New Pad
Reset
Format
shinny-hat-8990
Samples
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/
Privacy notice
Send feedback
no issues
Based on Dart SDK 2.7.0

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

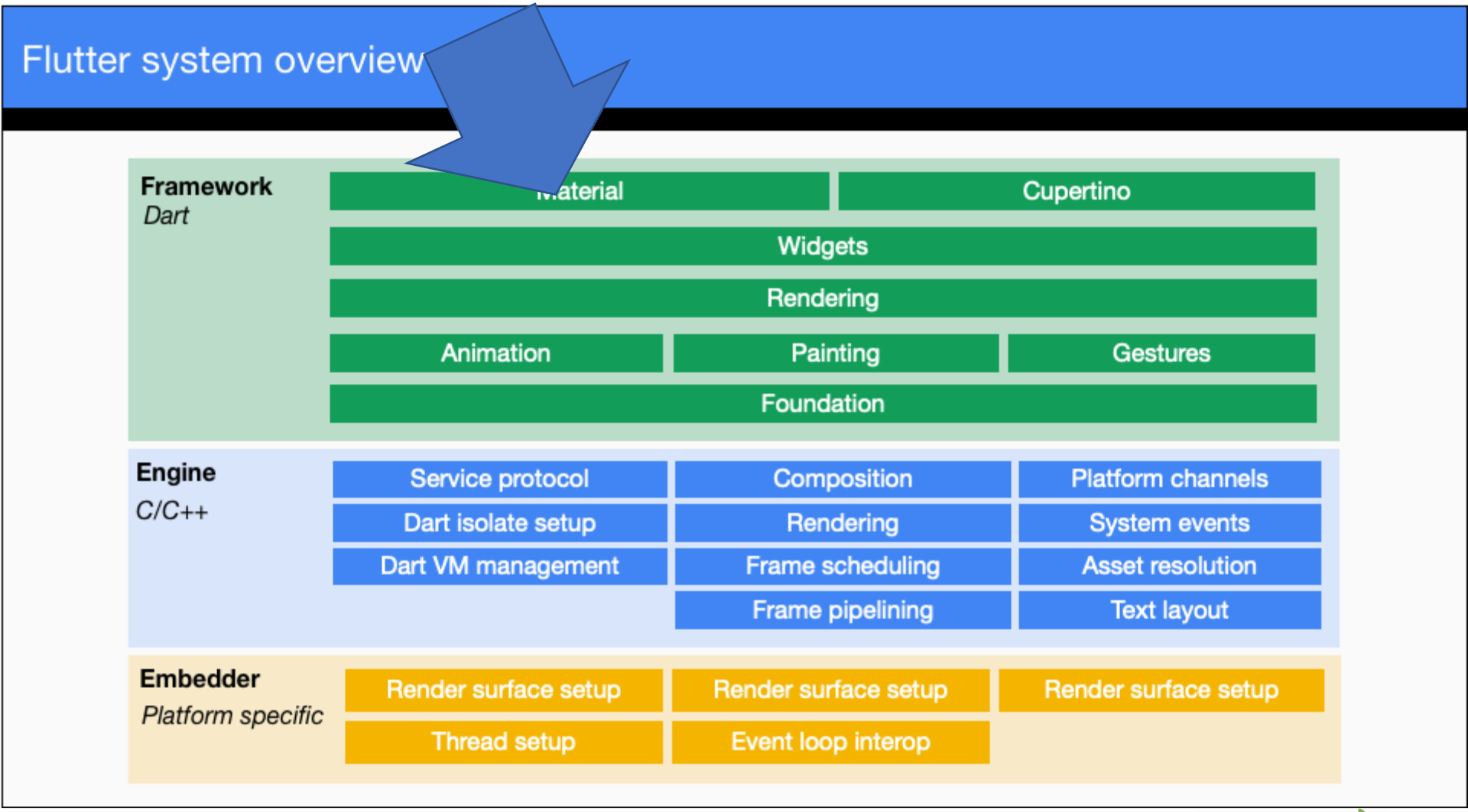


Flutter is built in C and C++ and provides a 2D rendering engine, a React-inspired functional-reactive framework, and a set of Material Design widgets. On Android, the C/C++ code is compiled with Android's NDK, while the most part of the framework and app code in Dart runs on the Dart VM, which generates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



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

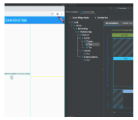
Performance improvements, DevTools updates, new Material You support, a new app template, and more! — Hello and welcome to Flutter 2.5! This is a big release, with the 2nd highest stats in the history of...

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/@csells_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

Search

Chris Sells

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 sta
tooling updates and more! — It's that time of year in the norther
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 Yo
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

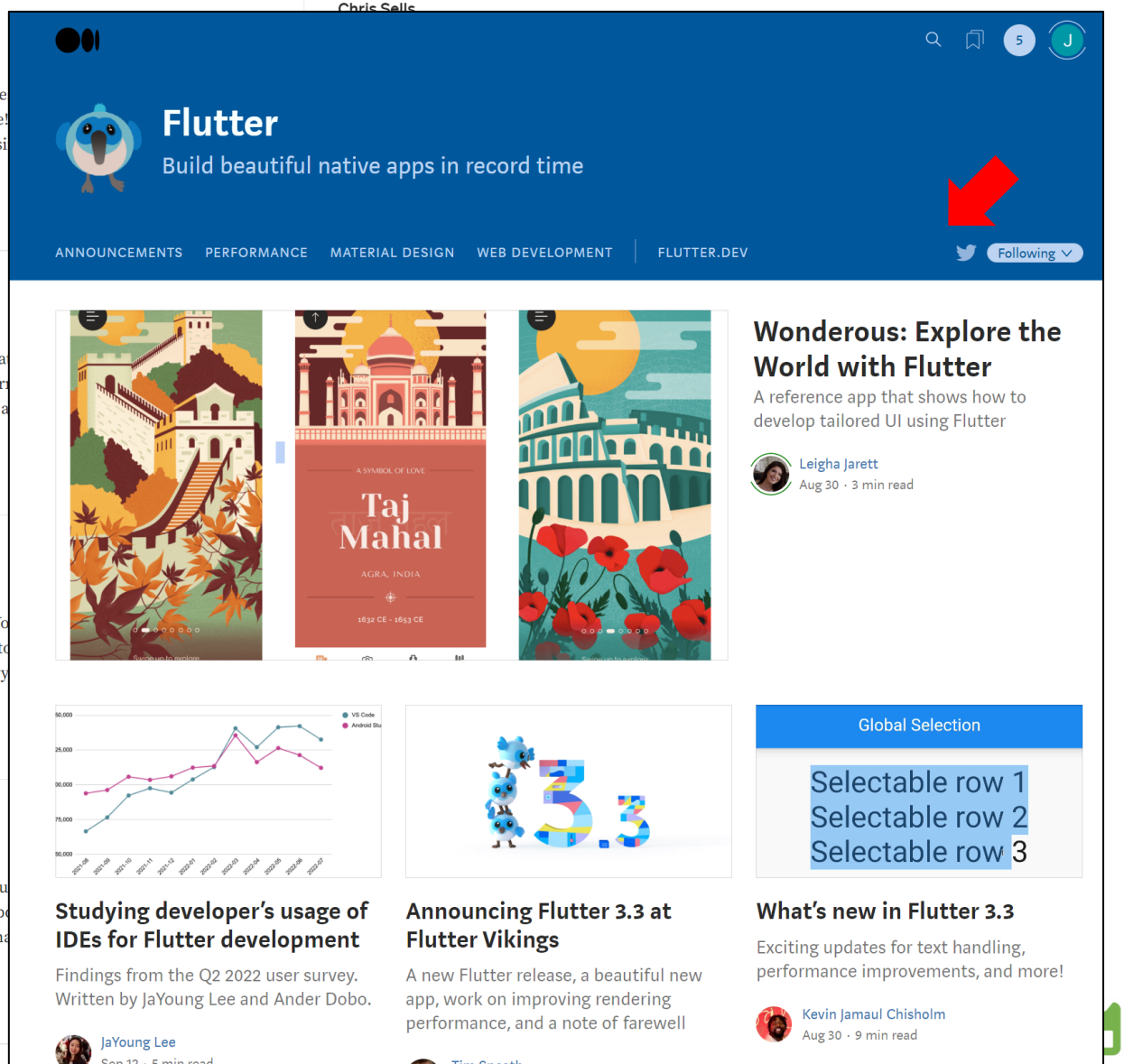
Flutter 18 min read

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

What's new in Flutter 2.2

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

Flutter 22 min read



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.

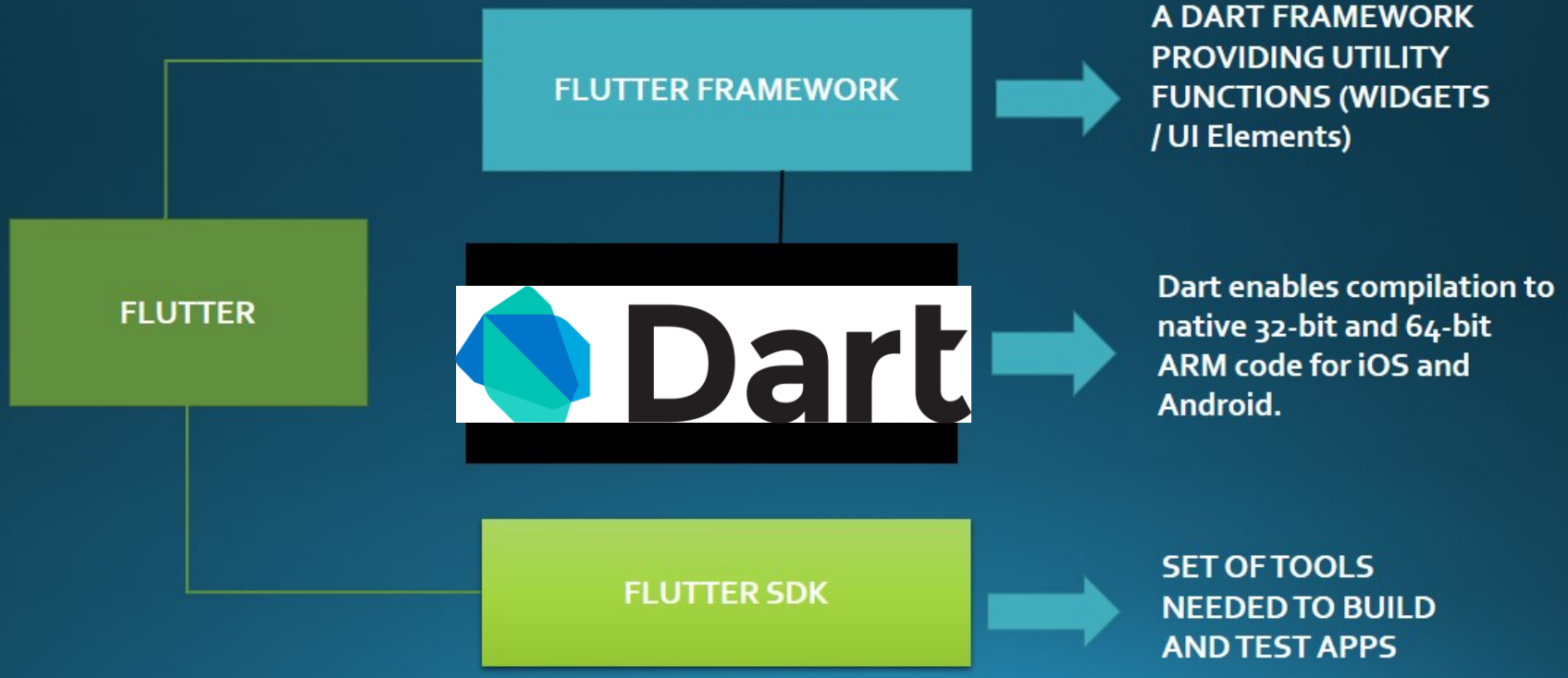
You might also check out What's next for Flutter and Introducing Dart 3 alpha.

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

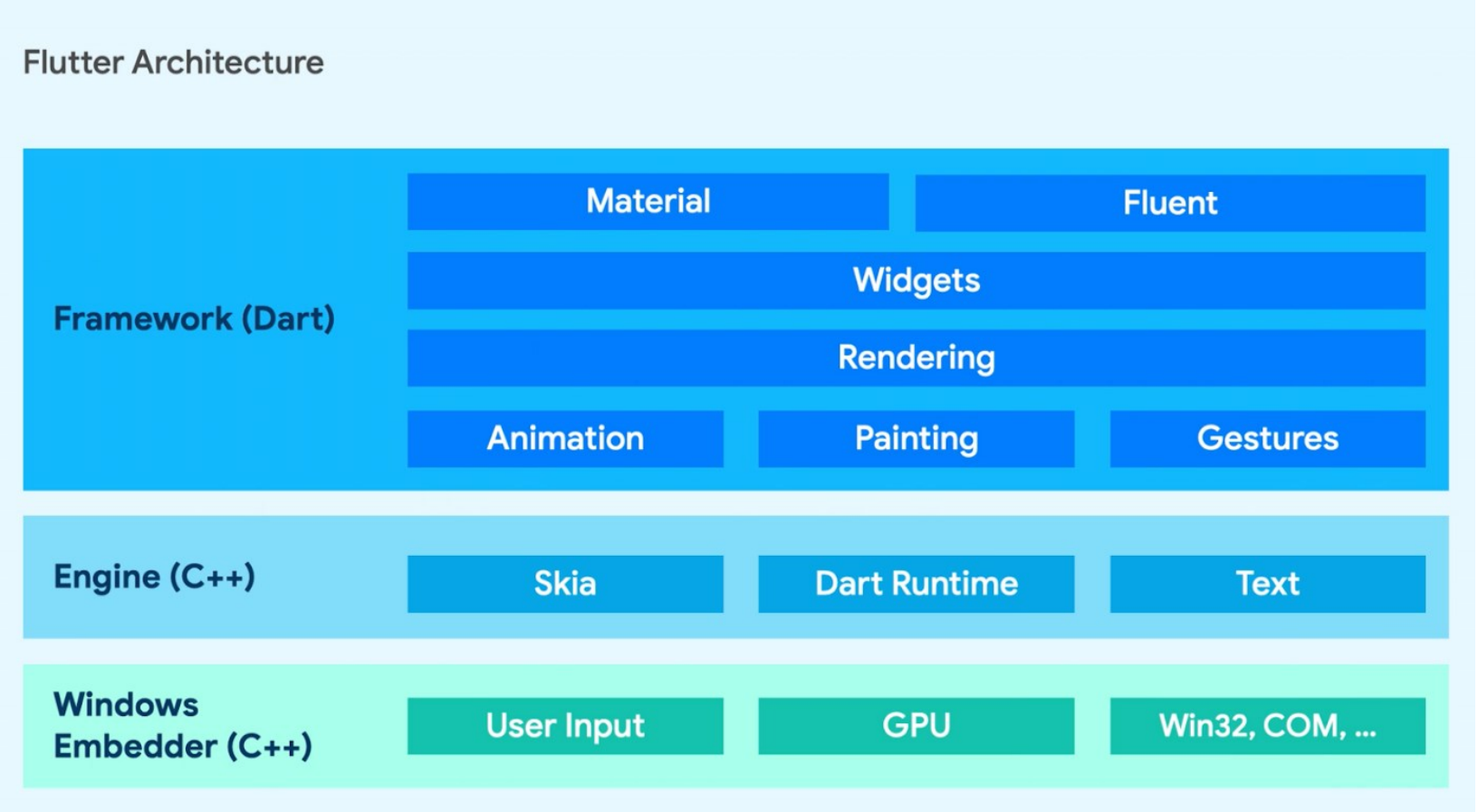
https://docs.flutter.dev/release/whats-new

Flutter VS Dart



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

Flutter in windows (fresh)



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 ★

Cooperation with UX/UIs has never been so good.

by Łukasz Kosman and Jakub Wojtczak

We are sharing the insights after making the first 10 comm

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?
- Ho
- Wh for?

Native code is accessible when needed.

Proof of Concept in Flutter is easy.

DEVs are happy.

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

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

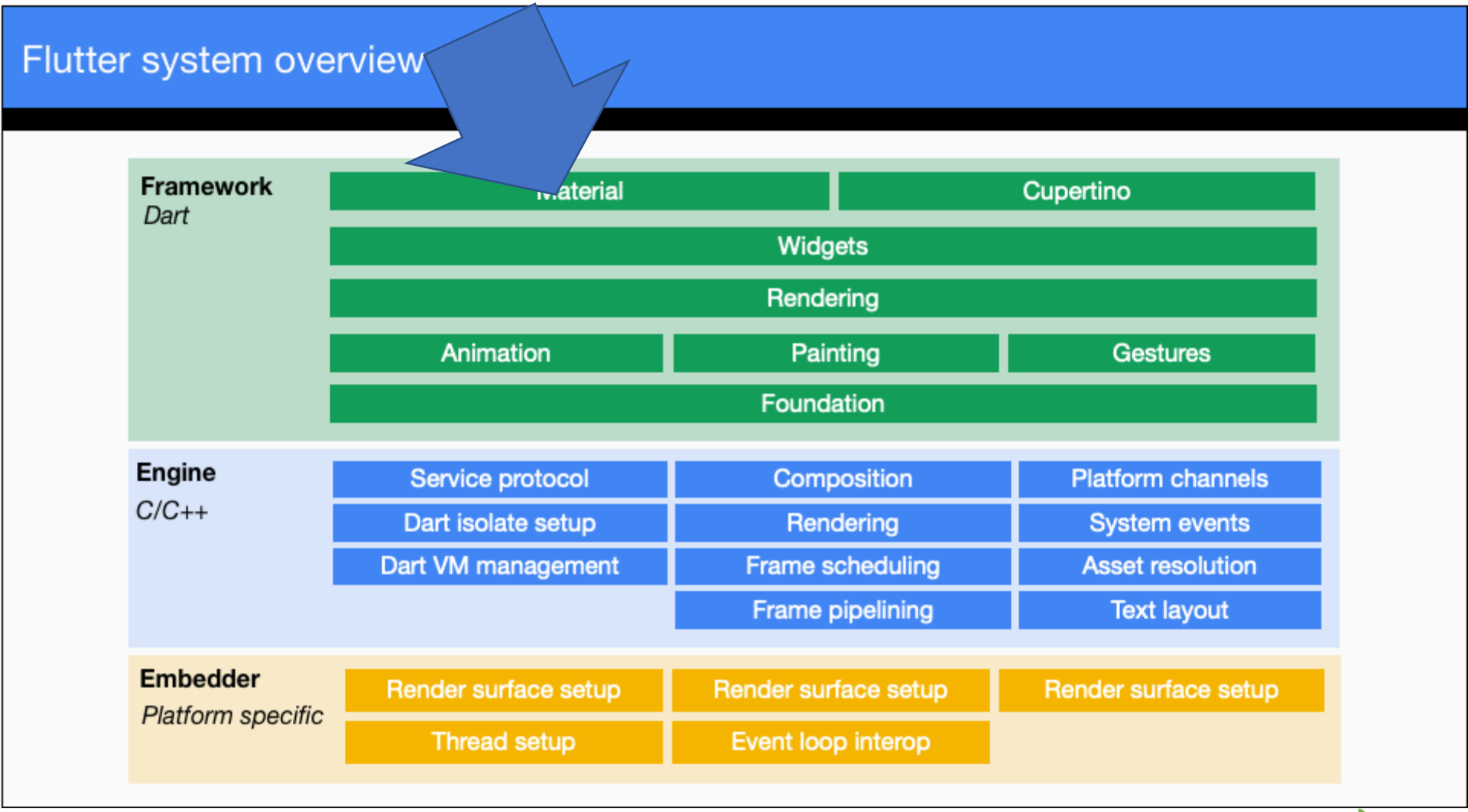
Sep 23, 2020 · 5 min read ★

CONTEXT

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

How?

You program here



For those curious

developers | Platform | Docs | More | Search | English | J

DOCUMENTATION

Overview | Guides | Reference | Samples | Design & 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
- Audi
- Servi

Android Developers > Docs > Guides

Was this helpful?

Create and run a wearable app

On this page

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

with different screen shapes and sizes.

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

https://developer.android.com/docs

welcome to the Flutter Community

FEATURED ABOUT FLUTTER COMMUNITY IS NOT ACCEPTING NEW WRITERS AT THIS

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

welcome to the Flutter Community

FEATURED ABOUT FLUTTER COMMUNITY IS NOT ACCEPTING NEW WRITERS AT THIS

Flutter: Buildi

Souvik Biswas Follow

Jun 25, 2019 · 7 min read

Introduction

This article will help you to build from scratch.

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%

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

https://github.com/fluttercommunity/flutter_wear_plugin

OS . The plugin, wear , created by Matt helped me a lot in managing the screen size of the watch across different screens

Cha Fo **AdaShoelace** for Charlie Foxtrot
Posted on Nov 6, 2020 • Updated on Nov 9, 2020

Rpi 4 meets Flutter and Rust

#flutter #rust #raspberrypi #linux

For those curious

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.

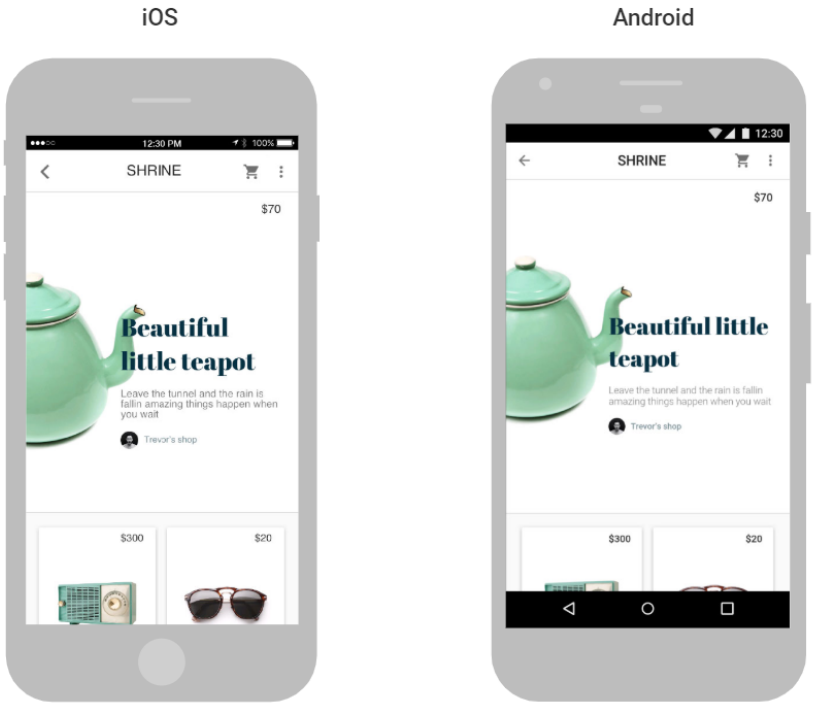
This project has quite a few requirements:

- 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

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.

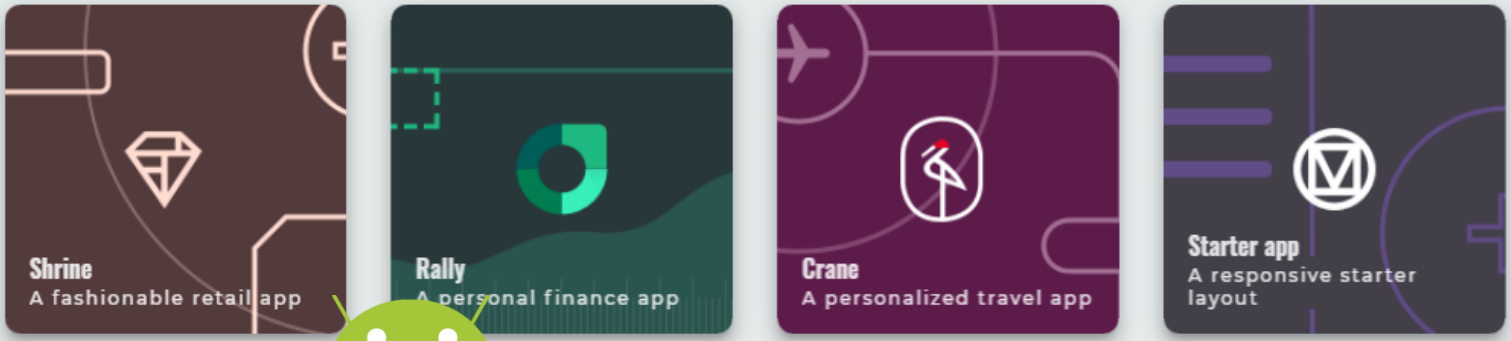


What is Flutter?
https://flutter.dev/docs/resources/technical-overview

Flutter

Themes for both the Android (Material Design) and iOS (Cupertino) platforms.

Gallery



Categorie

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

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

Try Flutter in your browser

Spinning Flutter ▾

Dart | Format | Reset | ▶ Run

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

void main() async {
  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        body: MyApp(),
      ),
    ),
  );
}

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp>
    with SingleTickerProviderStateMixin {
  AnimationController controller;
  Animation<double> animation;

```

UI Output

JUST SCROLL DOWN ON ENTRY

Console

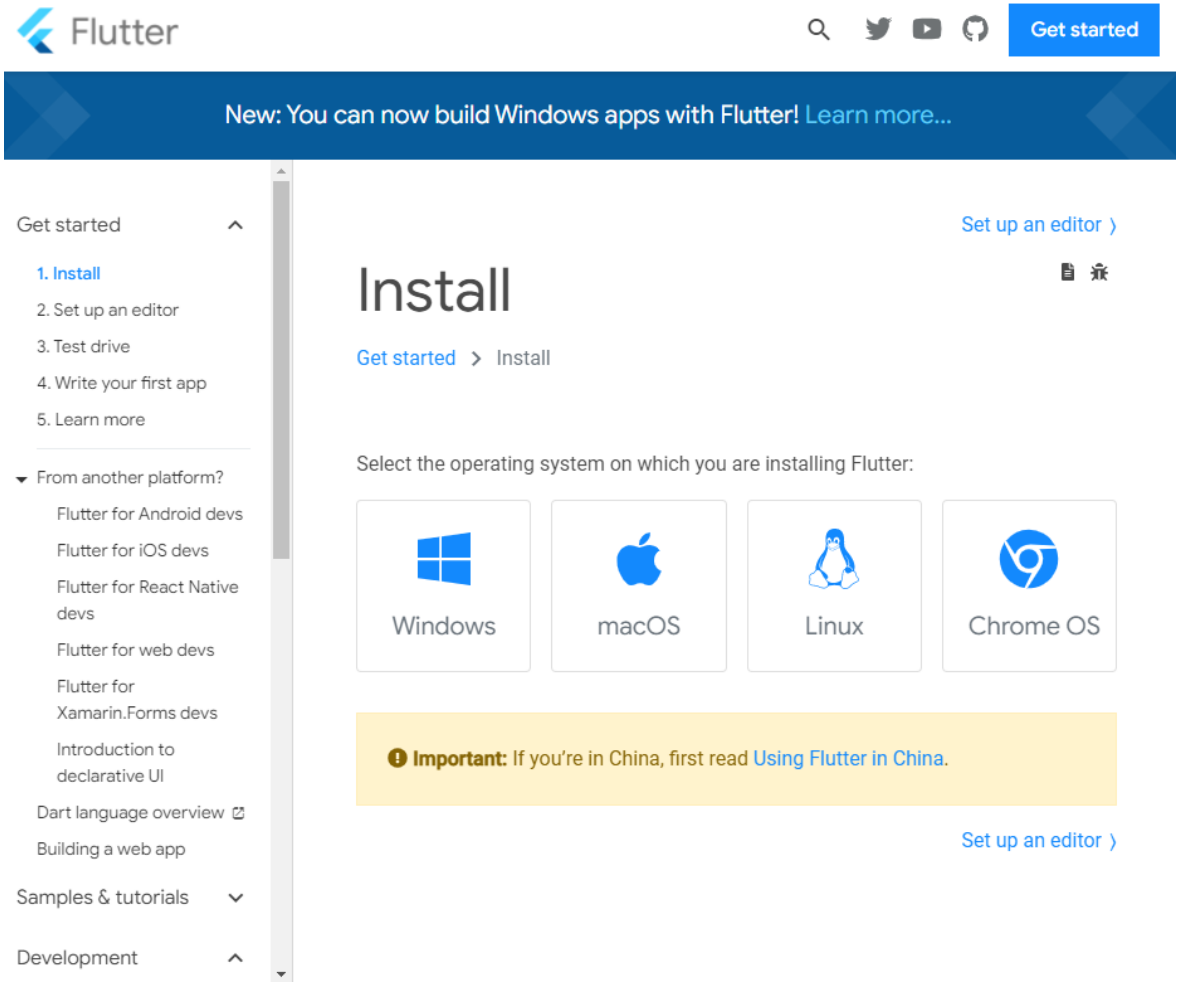
no issues

https://flutter.dev/

Want more practice? Try a codelab.

My first Flutter application

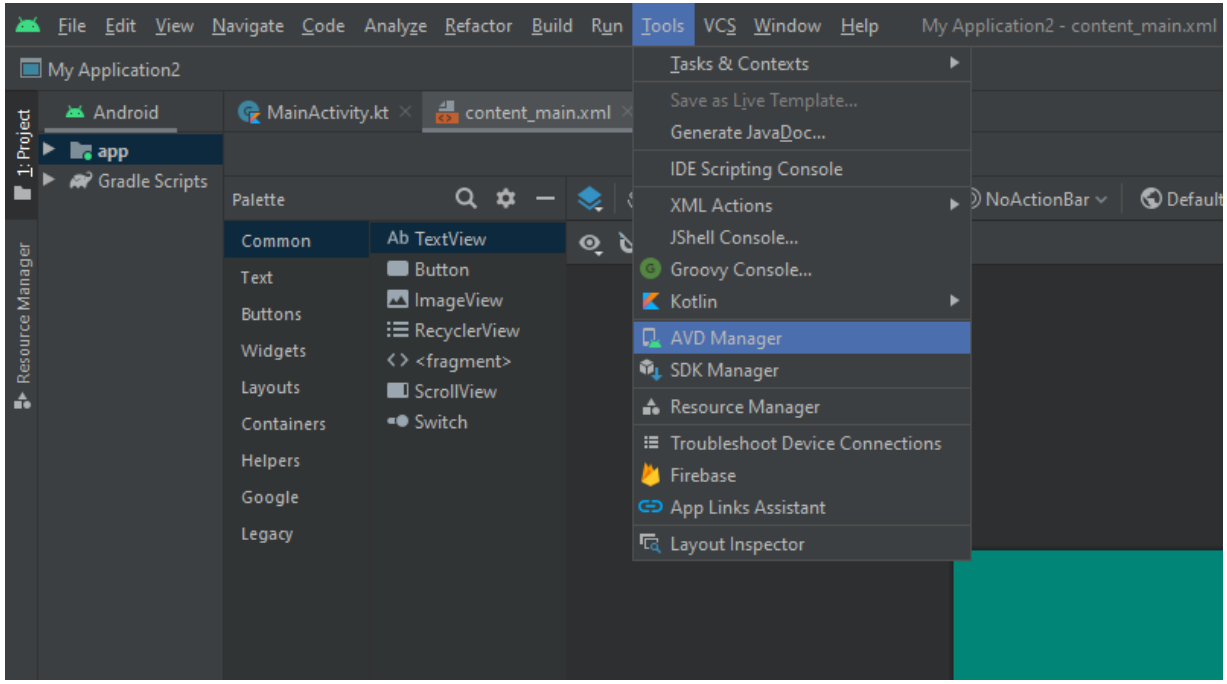
runtime



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

Android runtime

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



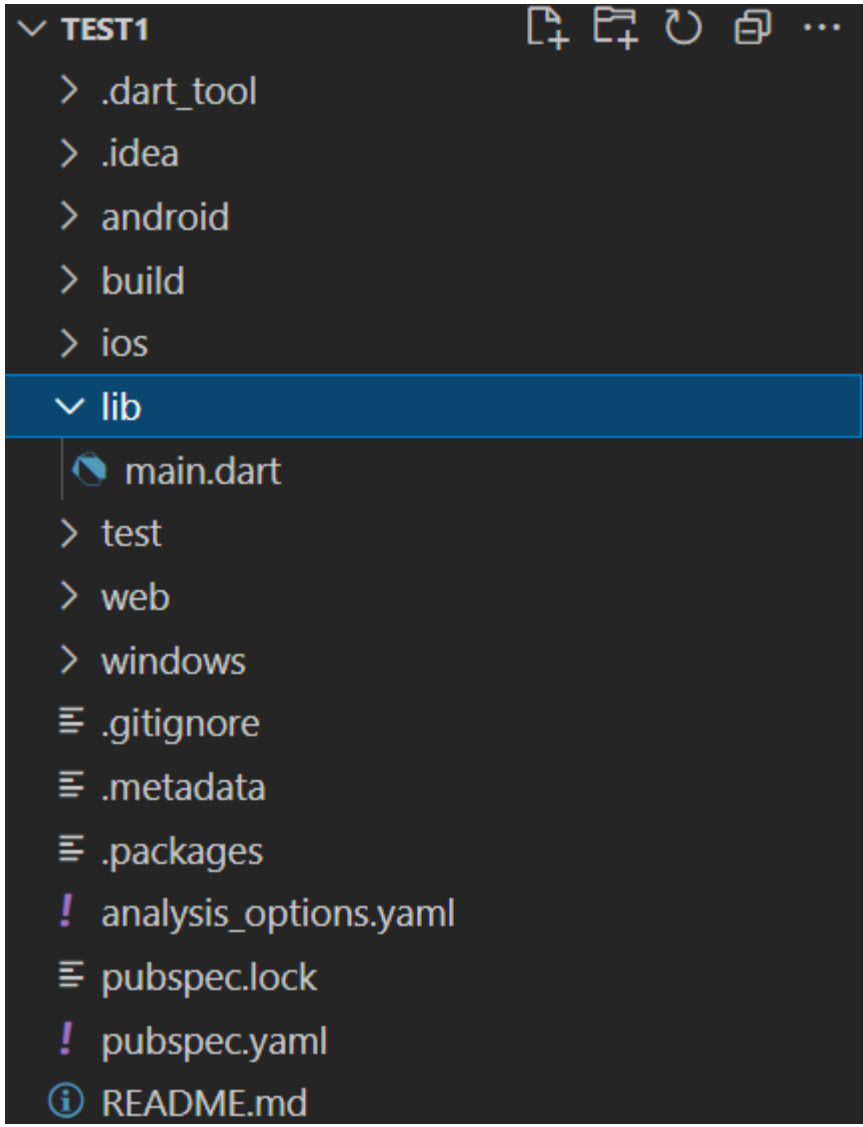
Create project (CLI)

- > 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.
```

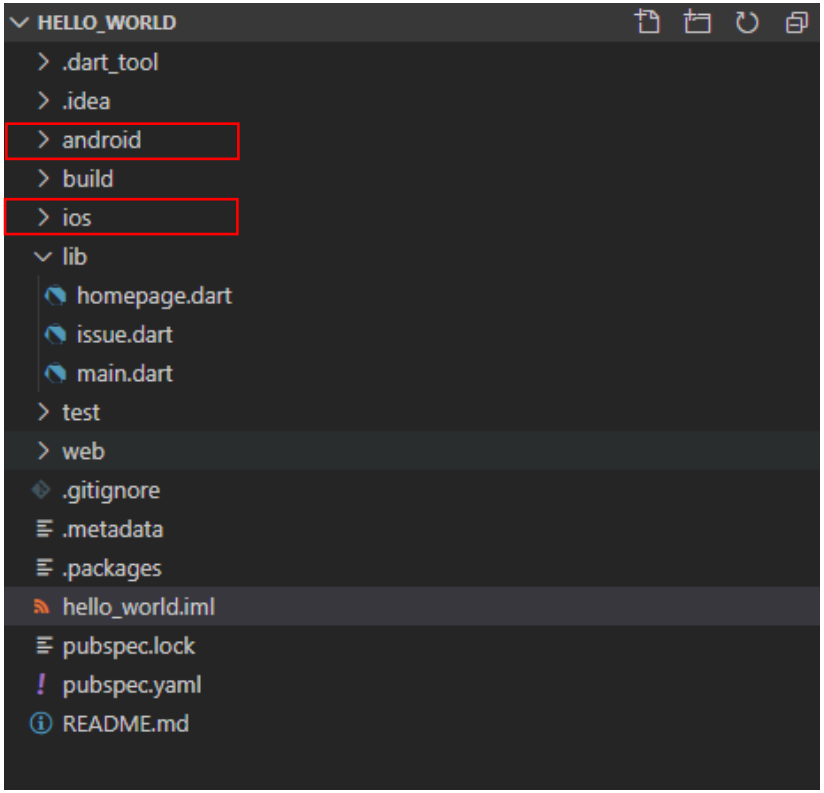
Now you have this

```
    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
```



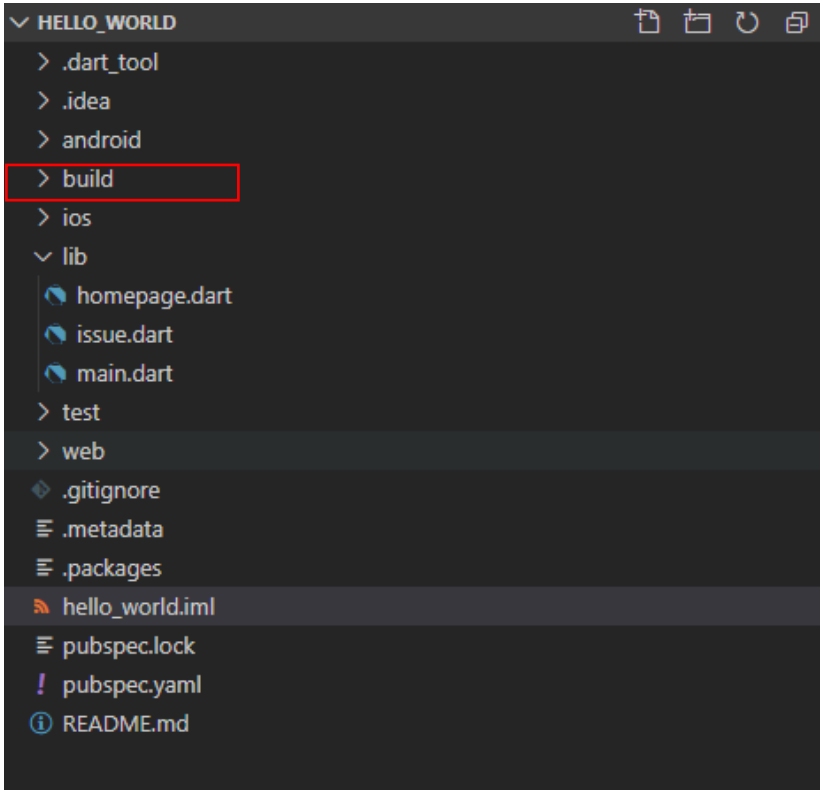
After creating your first Flutter project

- ***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.



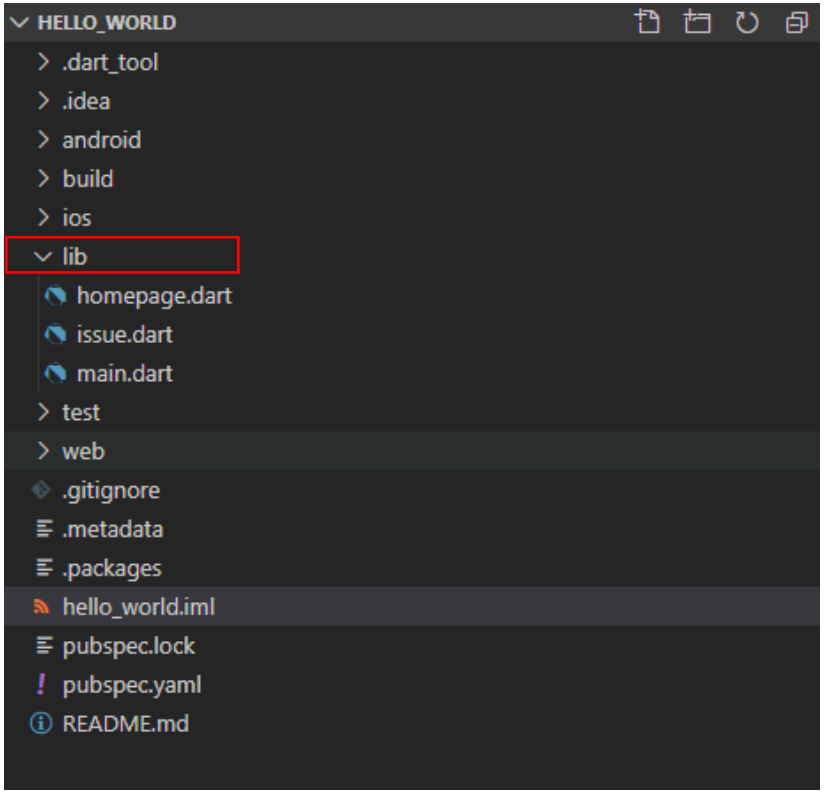
The project structure

- 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.



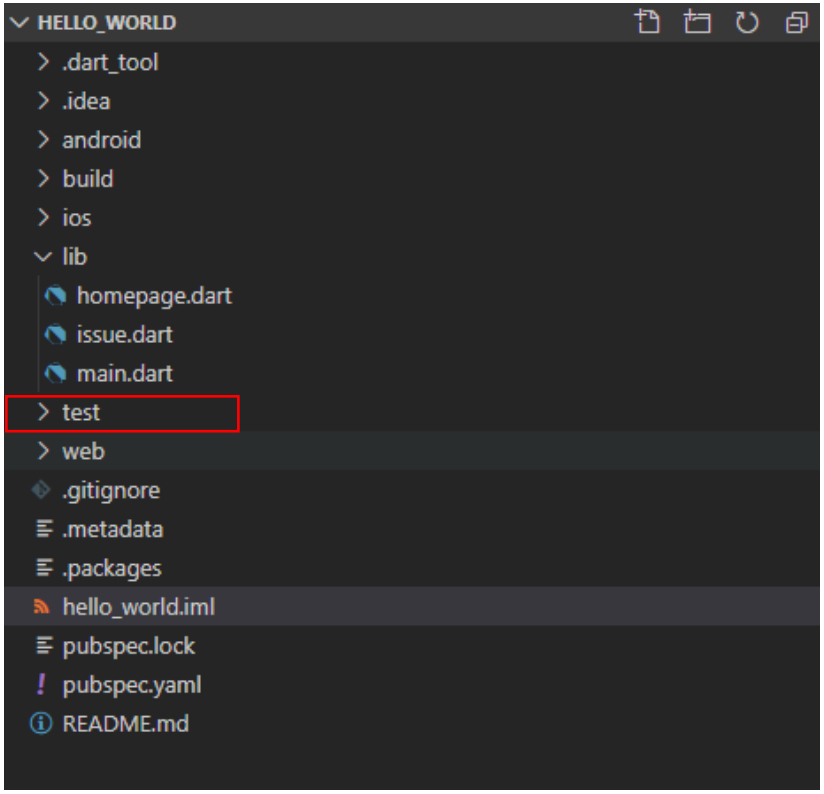
The project structure

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



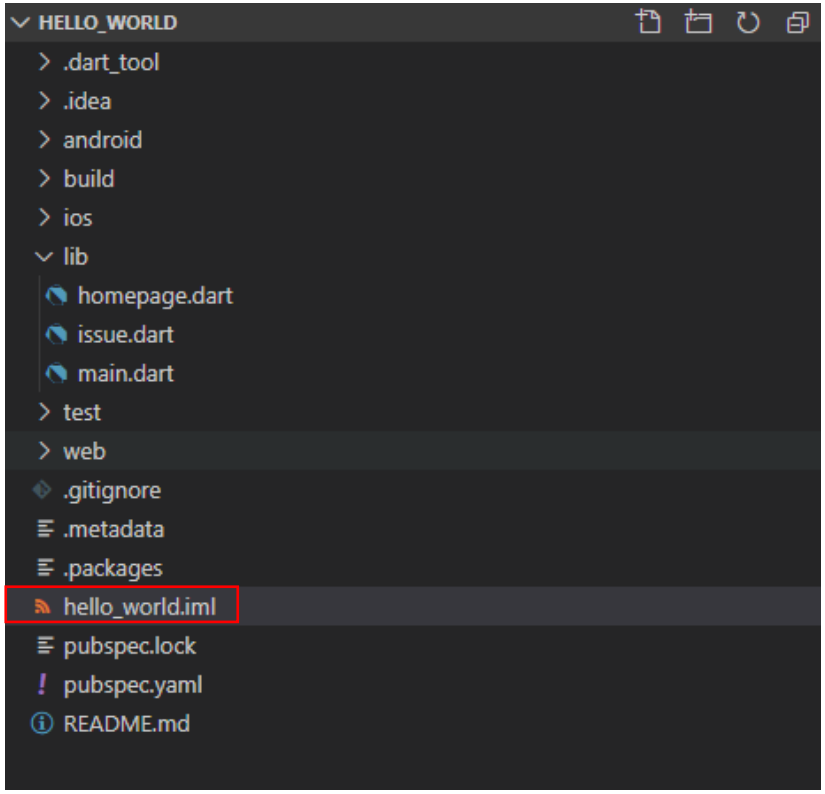
The project structure

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



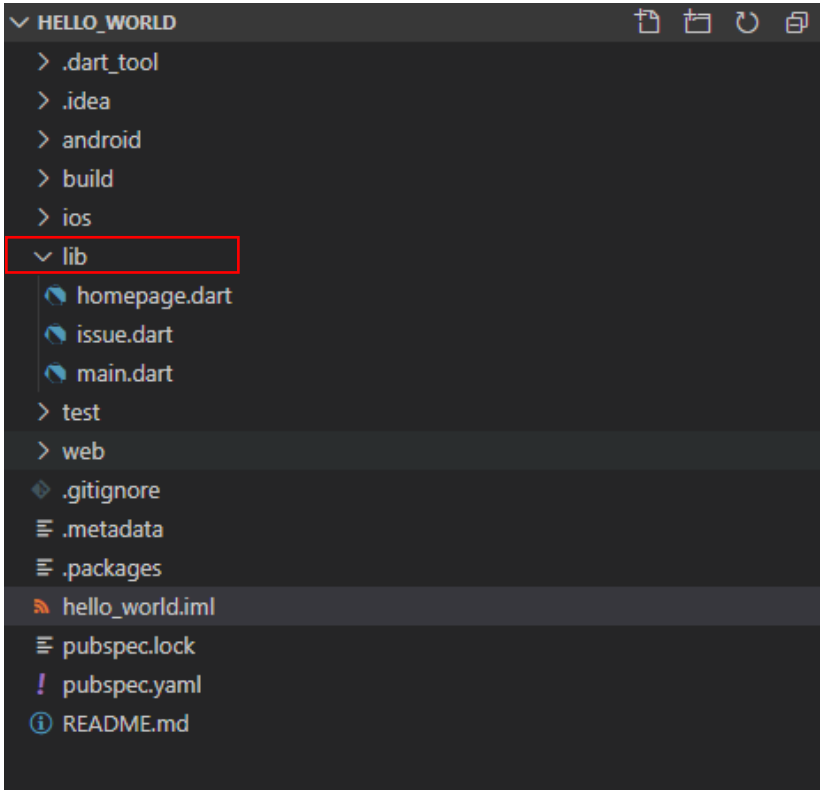
Files In Your Flutter Application

- *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.



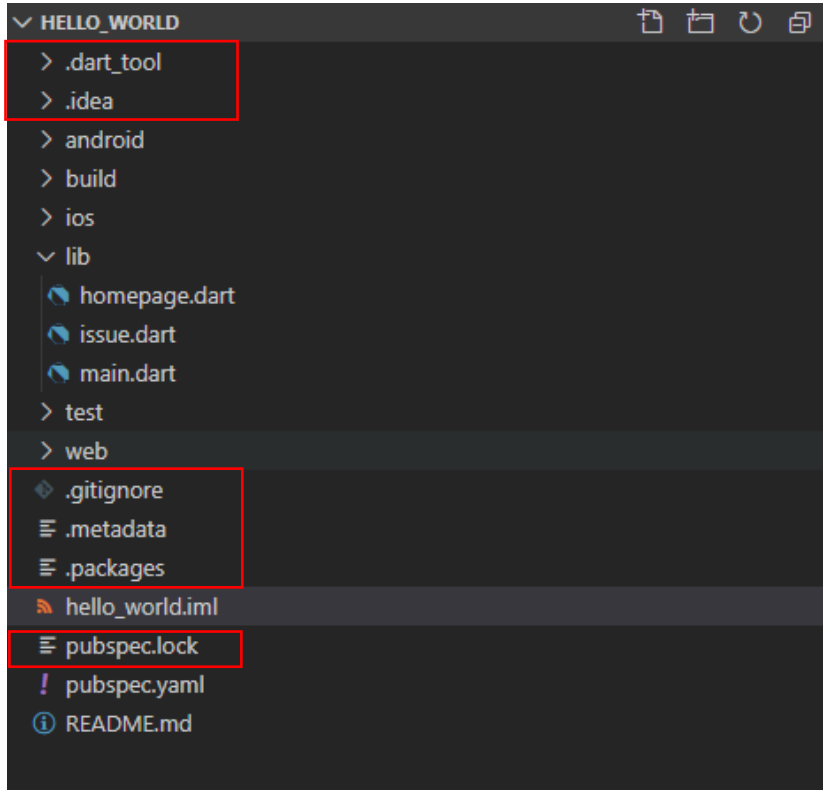
Files In Your Flutter Application

- *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

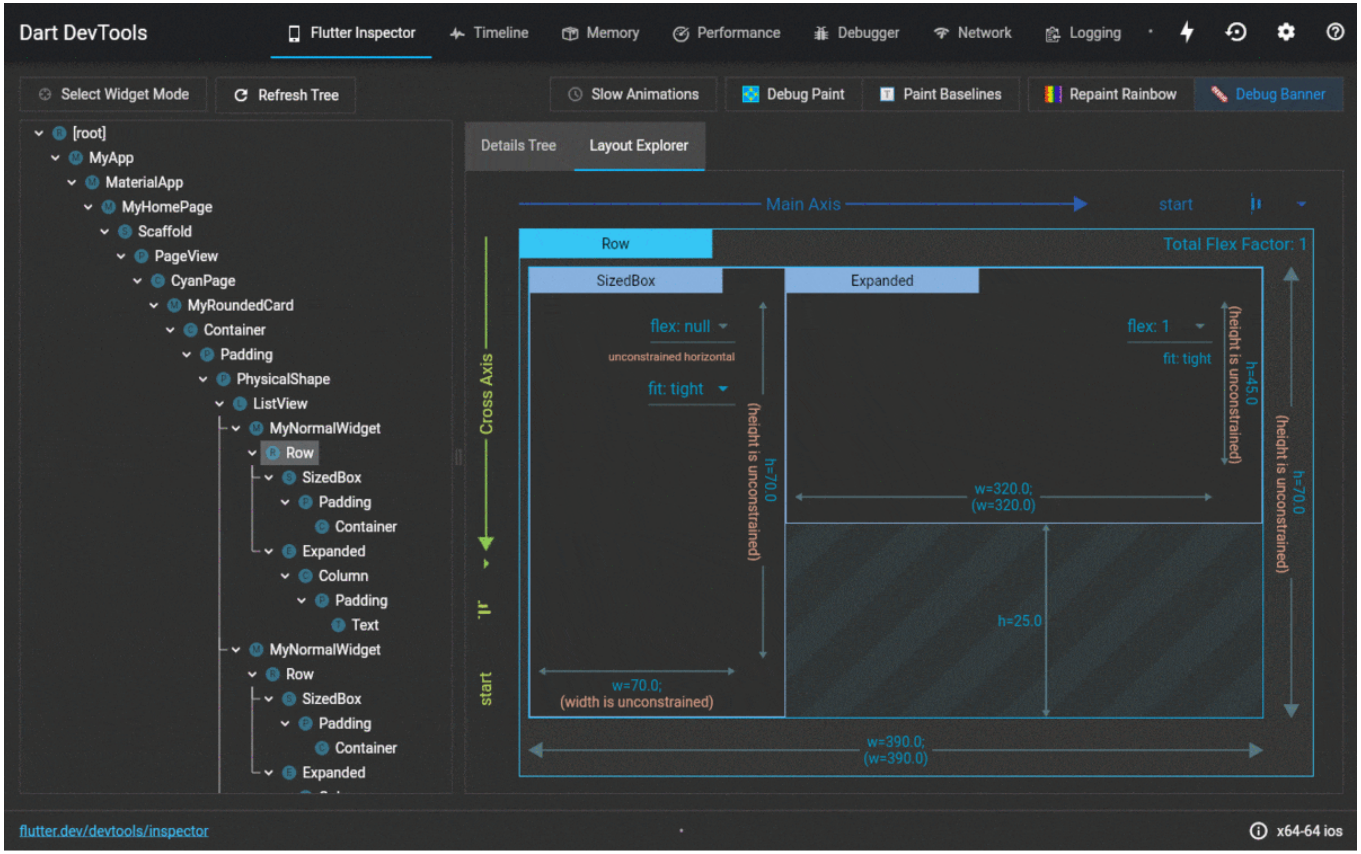
Android → Android SDK

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

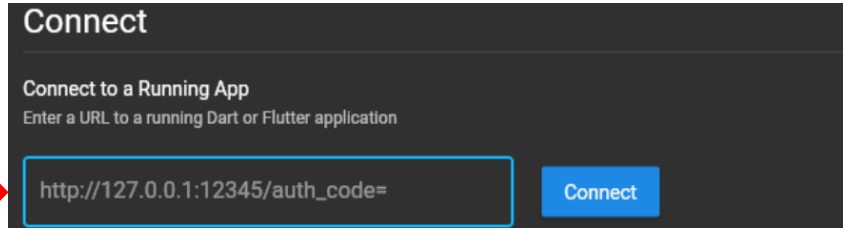
App vs devtools (in chrome, using CLI)

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
```

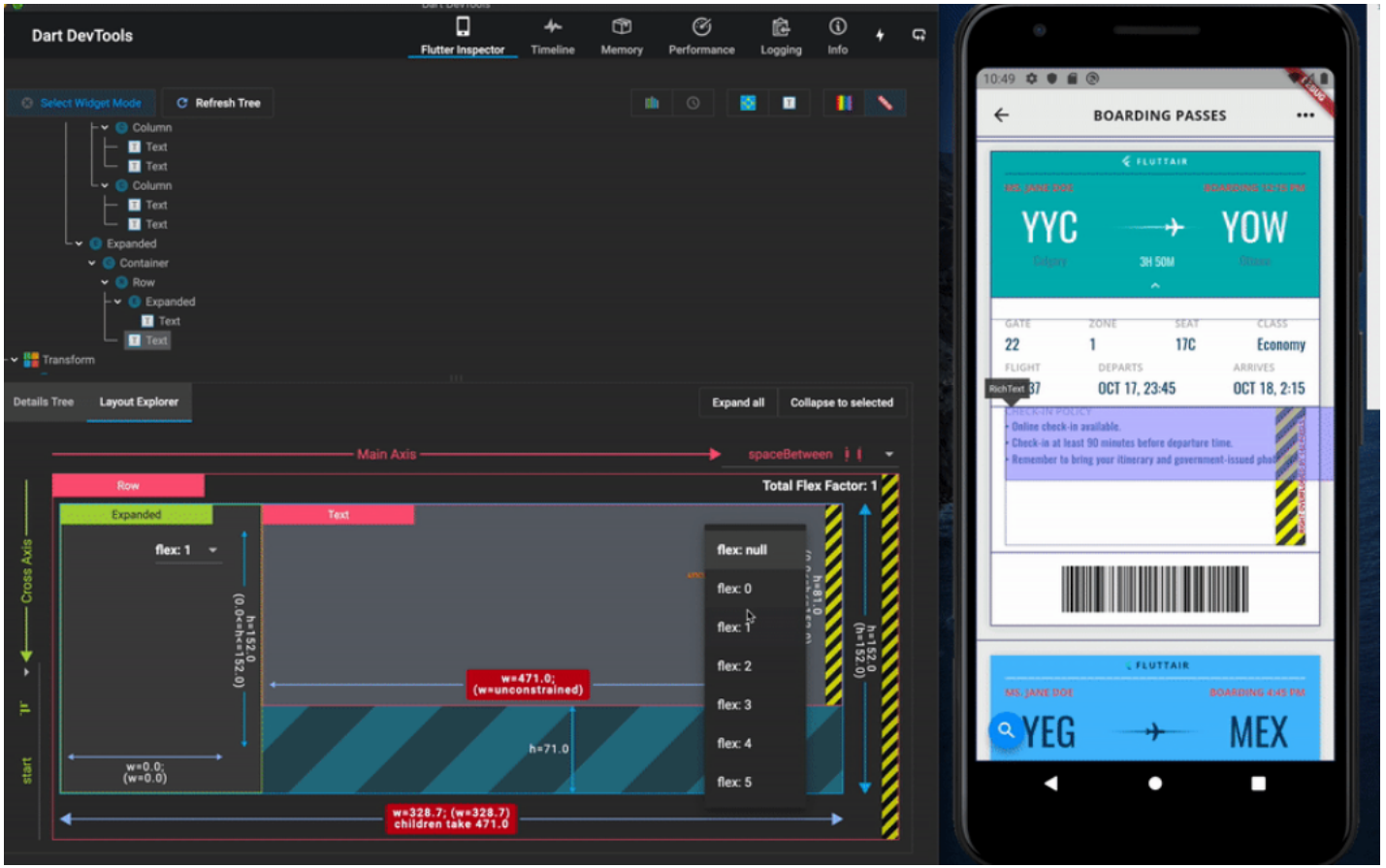
devtools

- Go to `http://localhost:9100`



devtools

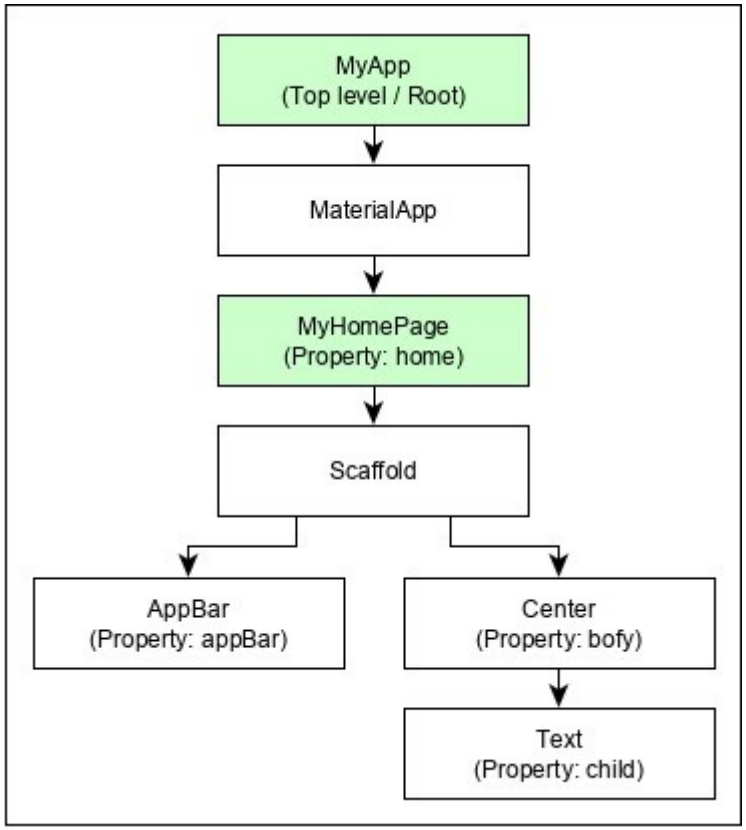
Emulator/chrome/phone



Basic concepts

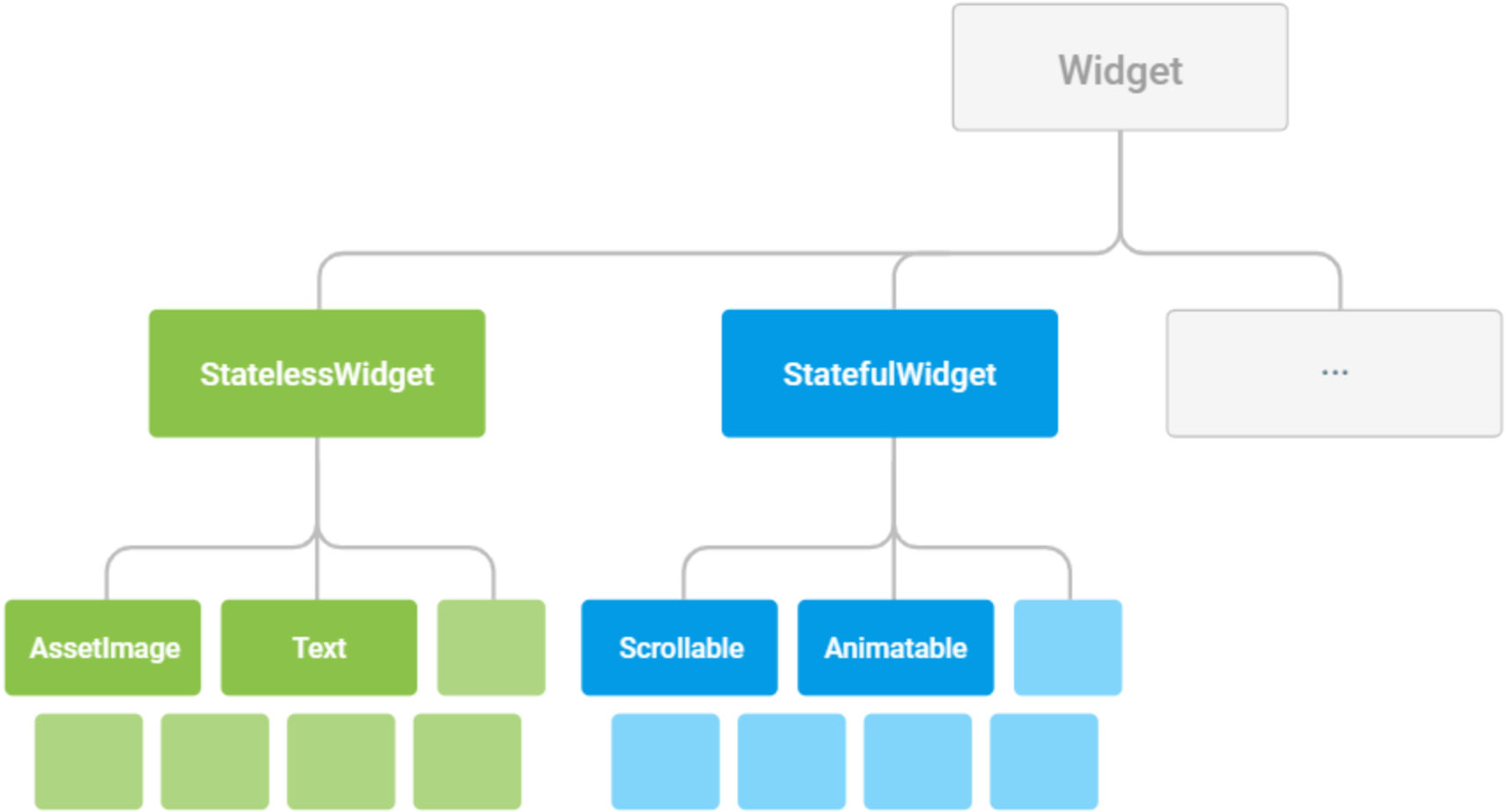
Quick overview

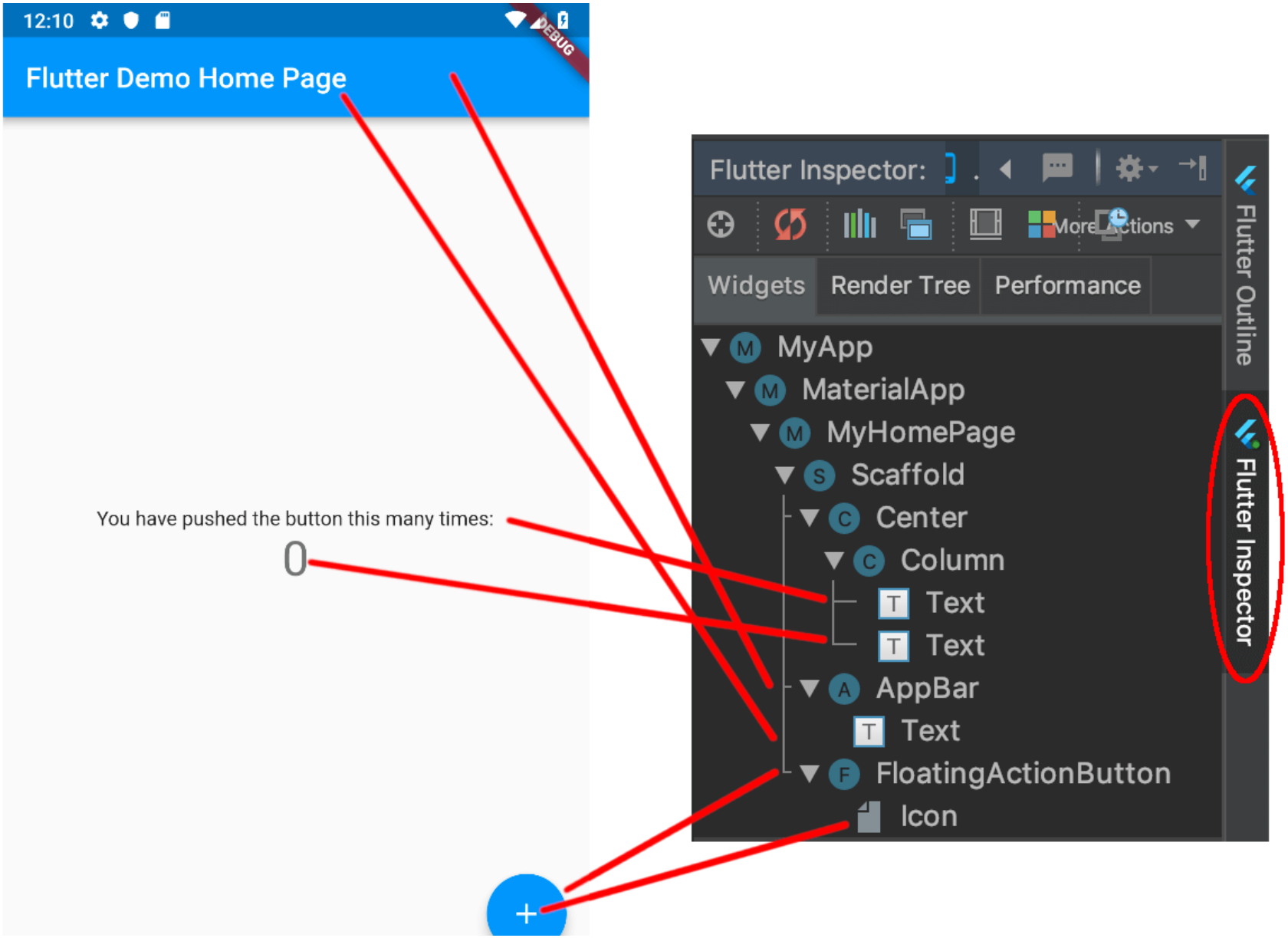
Flutter application architecture



https://www.tutorialspoint.com/flutter/flutter_architecture_application.htm

Everything's a widget

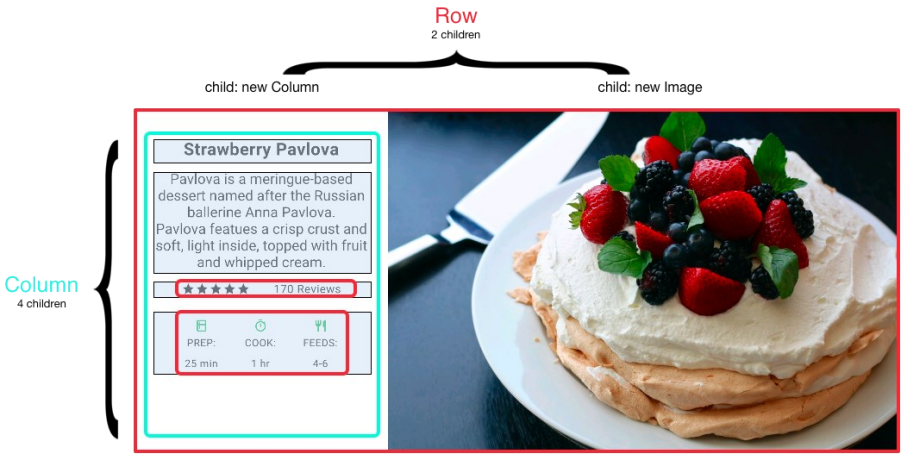


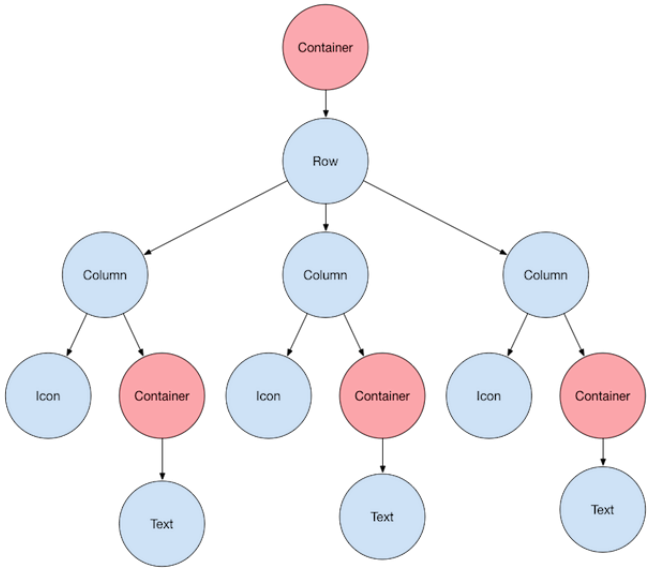


FIRST STEPS WITH FLUTTER- PART 1: EXPLORING WIDGETS
https://pusher.com/tutorials/flutter-widgets

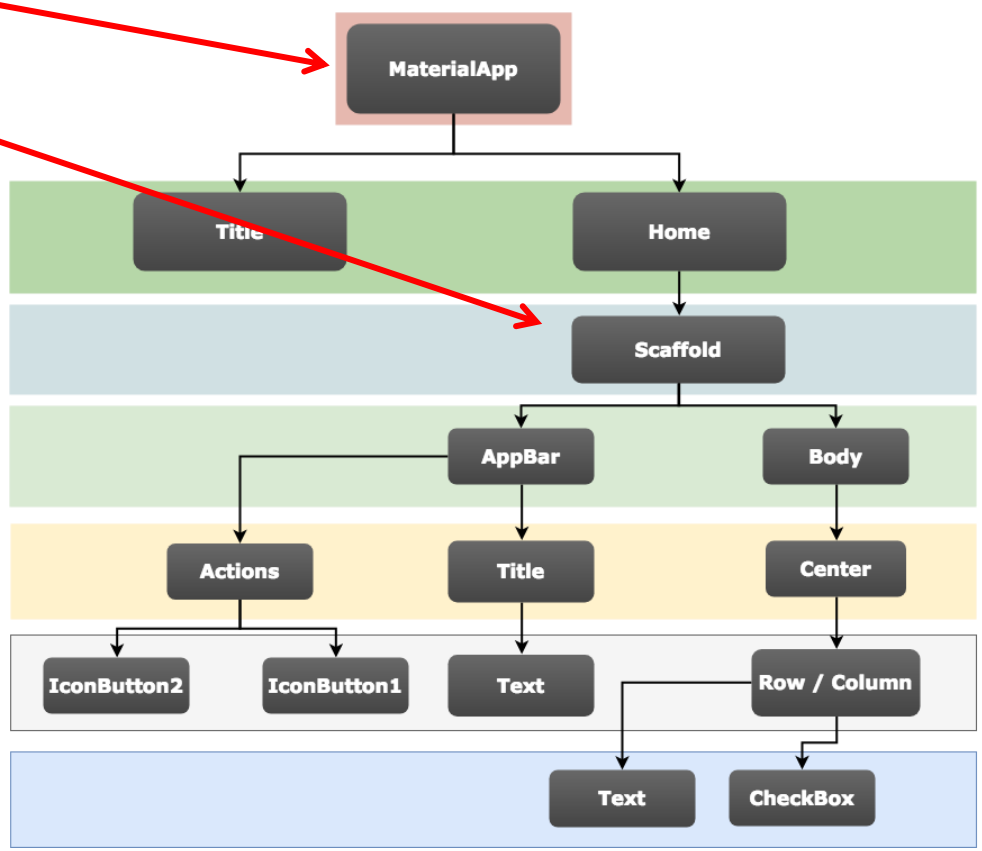
Building widgets

- 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.



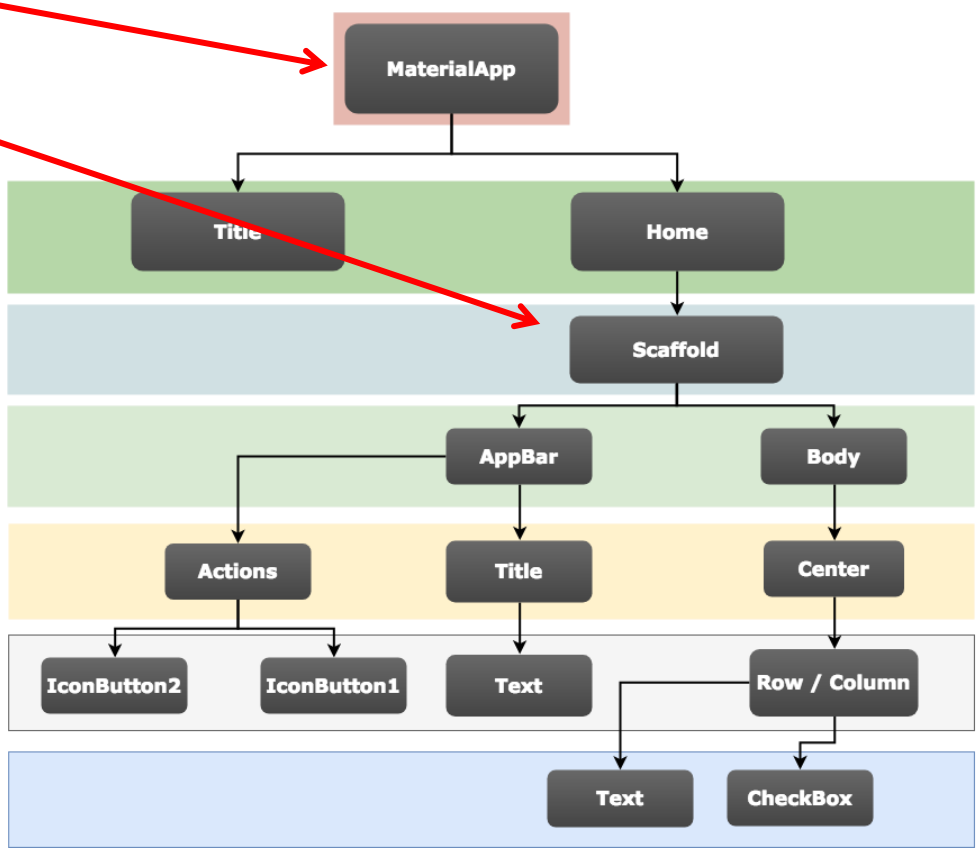


```
class MyAppState extends State<MyApp> {
  bool checkBoxValue = false;
  String  actionText = "Default";
  @override
  Widget build(BuildContext ctxt) {
    return new MaterialApp(
      title: "MySampleApplication",
      home: new Scaffold(
        appBar: new AppBar(
          title: new Text("Hello 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;
                    });
                  }
              )
            ],
          ),
        )
      ),
    );
  }
}
```

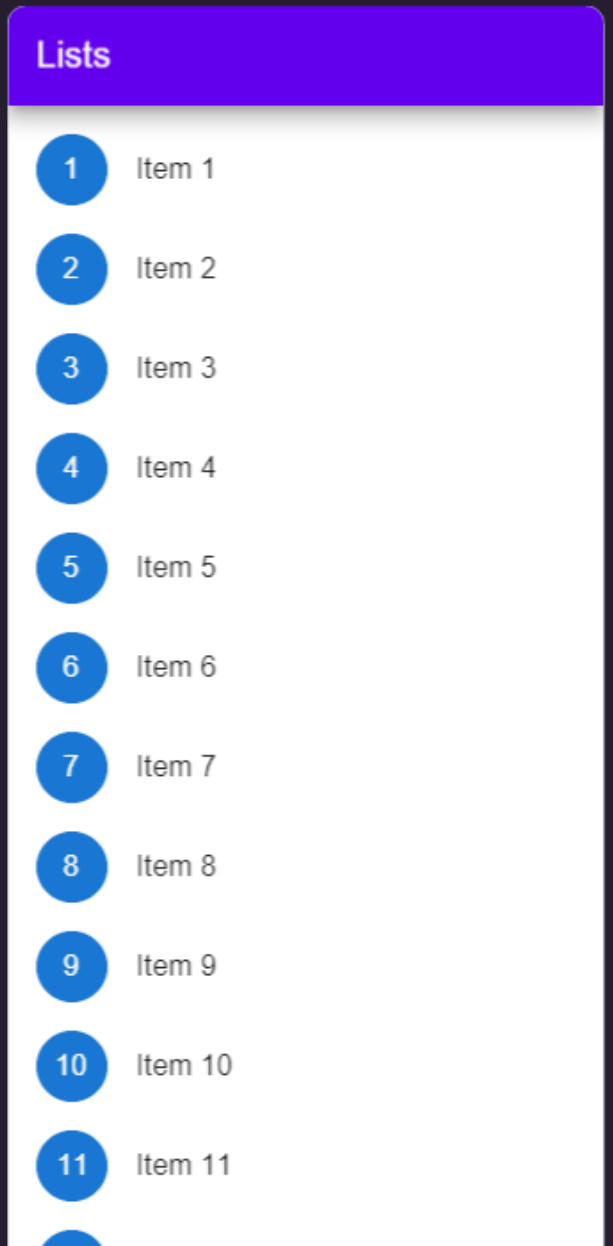


Flutter : From Zero To Comfortable
https://proandroiddev.com/flutter-from-zero-to-comfortable-6b1d6b2d20e

A lots of “{}()”....

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
```



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

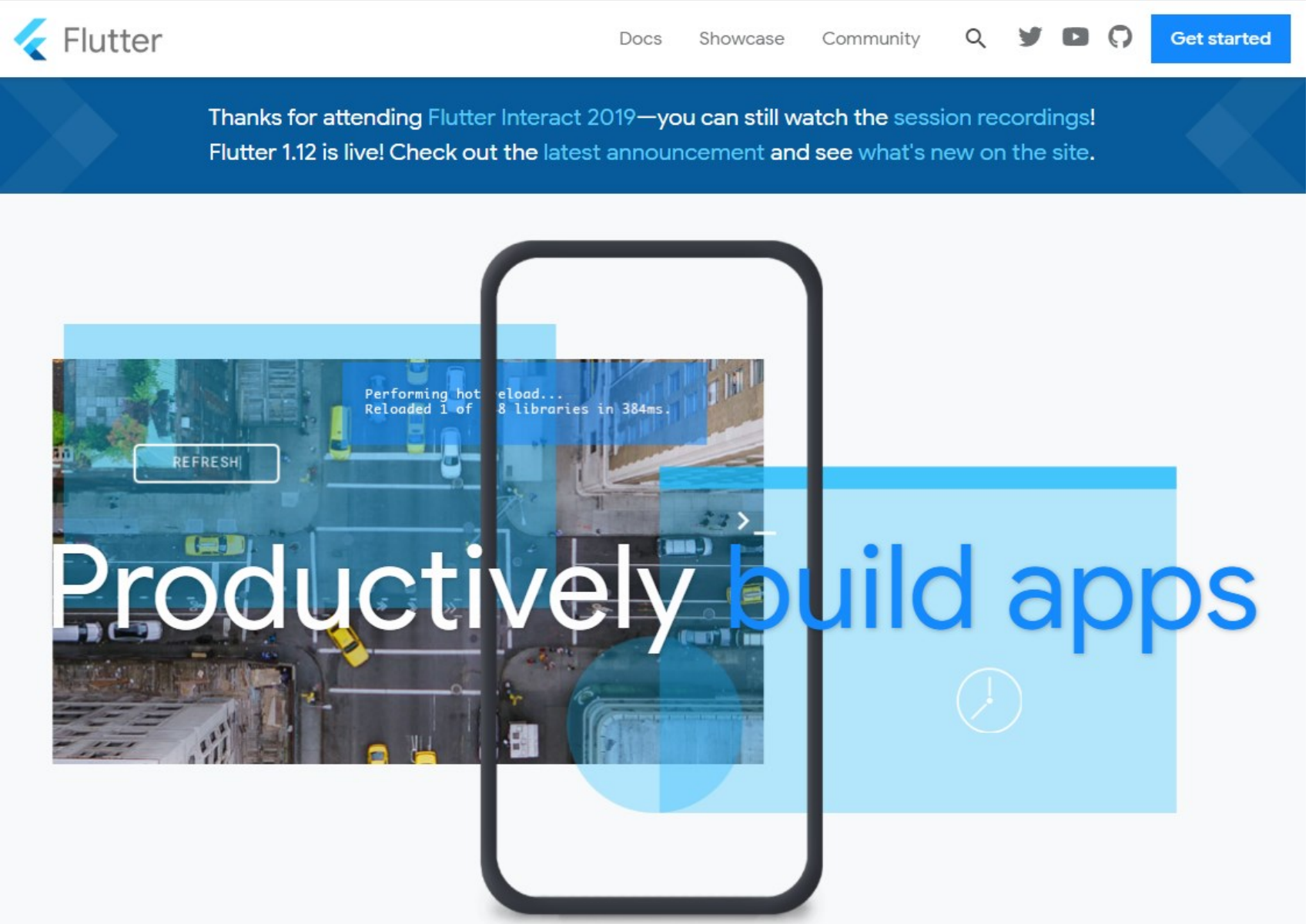
- don't store any state.
- don't store values that might change.

- 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?



https://flutter.dev/

Codelabs

Contents

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

https://flutter.dev/docs/codelabs

Codelabs

Contents

Basic Flutter layout concepts

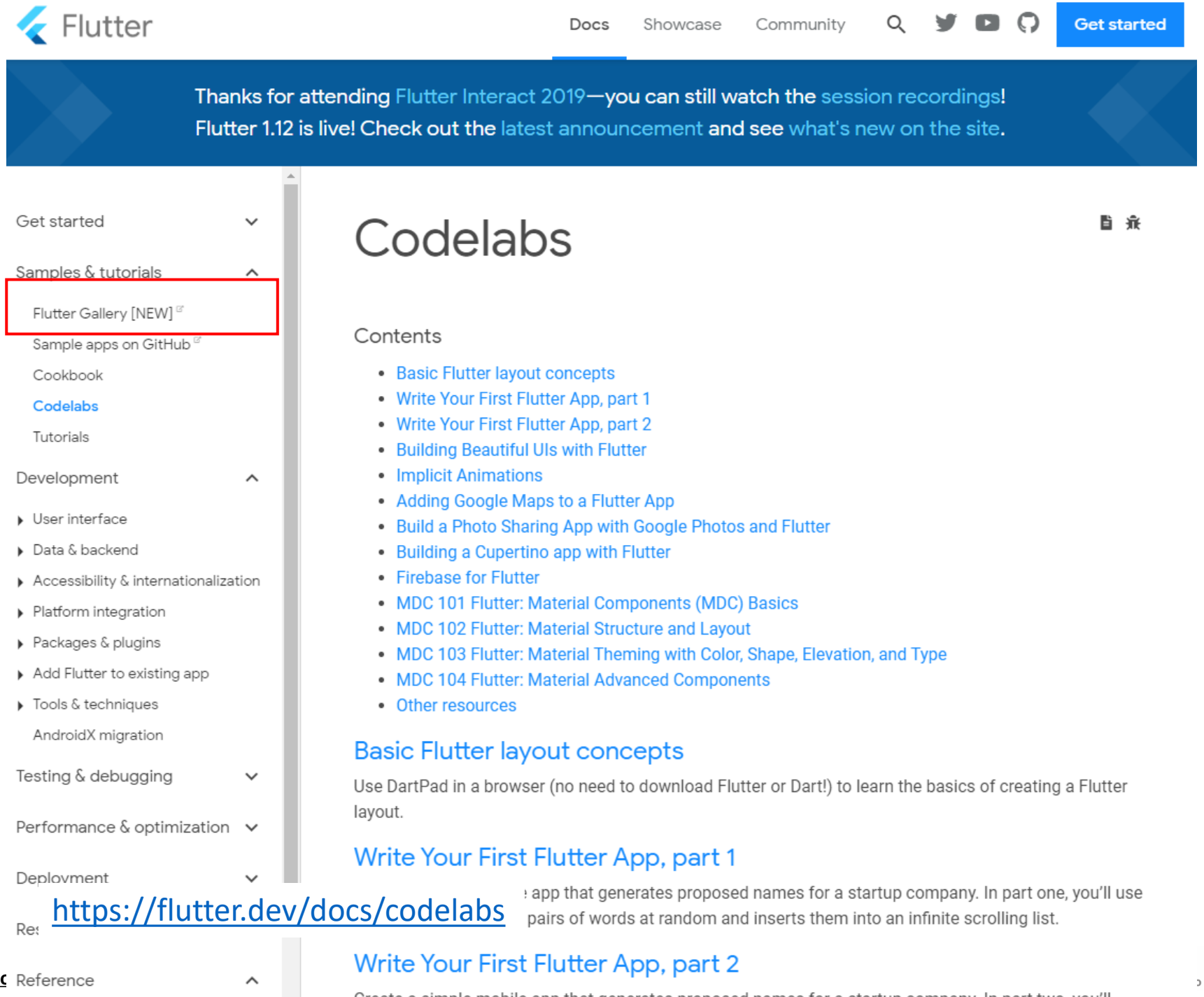
Use DartPad in a browser (no need to download Flutter or Dart!) to learn the basics of creating a Flutter layout.

Write Your First Flutter App, part 1

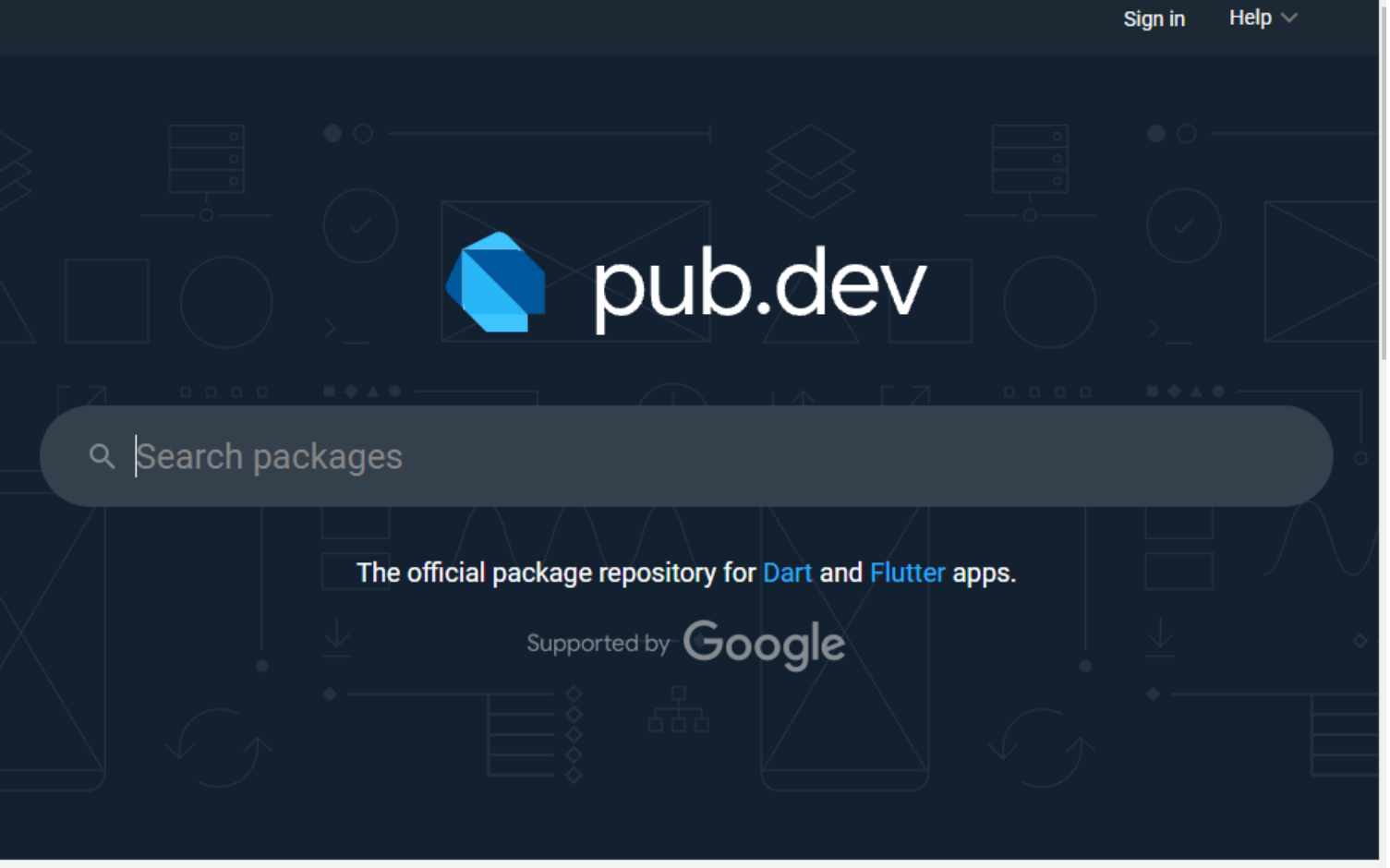
app that generates proposed names for a startup company. In part one, you'll use pairs of words at random and inserts them into an infinite scrolling list.

Write Your First Flutter App, part 2

https://flutter.dev/docs/codelabs



https://flutter.dev/docs/codelabs



Flutter Favorites

Some of the packages that demonstrate the highest levels of quality, selected by the Flutter Ecosystem Committee

infinite_scroll_pagin...

Lazily load and display pages of

flutter_native_splash

Generates native code to
er's default
ash screen with

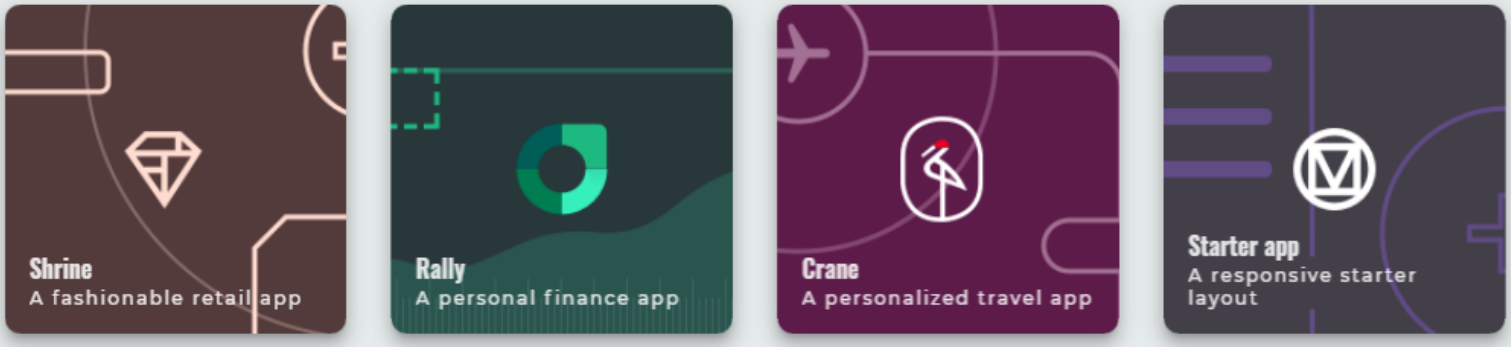
bottom_navy_bar

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

https://pub.dev/

Flutter

Gallery



Categories

MATERIAL

- Banner — Displaying a banner within a list
- Bottom app bar — Displays navigation and actions at the bottom
- Bottom navigation — Bottom navigation with

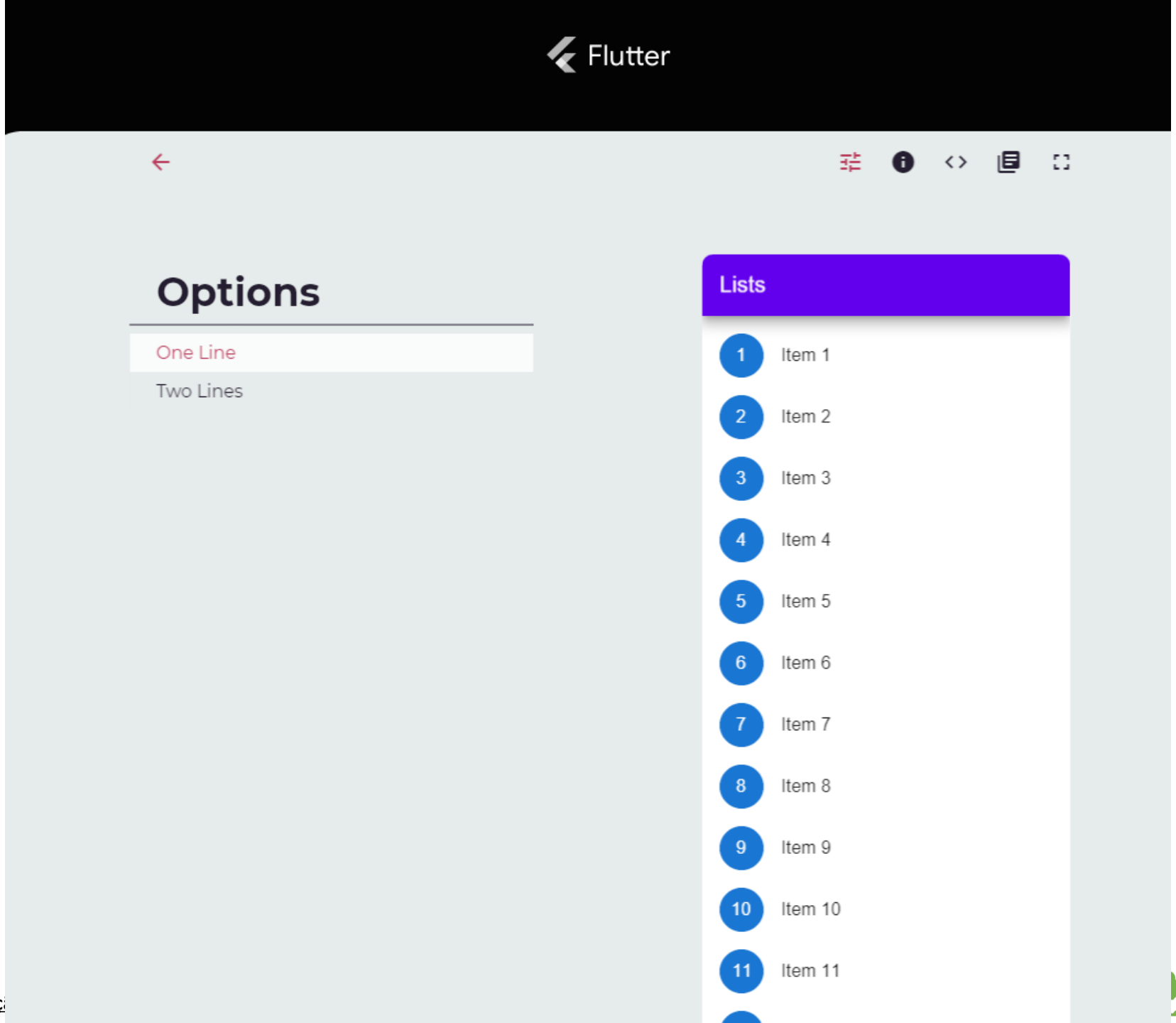
CUPERTINO

- Activity Indicator — iOS-style activity indicators
- Alerts — iOS-style alert dialogs
- uttons — iOS-style buttons

REFERENCE STYLES & MEDIA

- Colors — All of the predefined colors
- Typography — All of the predefined text styles

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


Flutter
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

Flutter › material › ListTile class

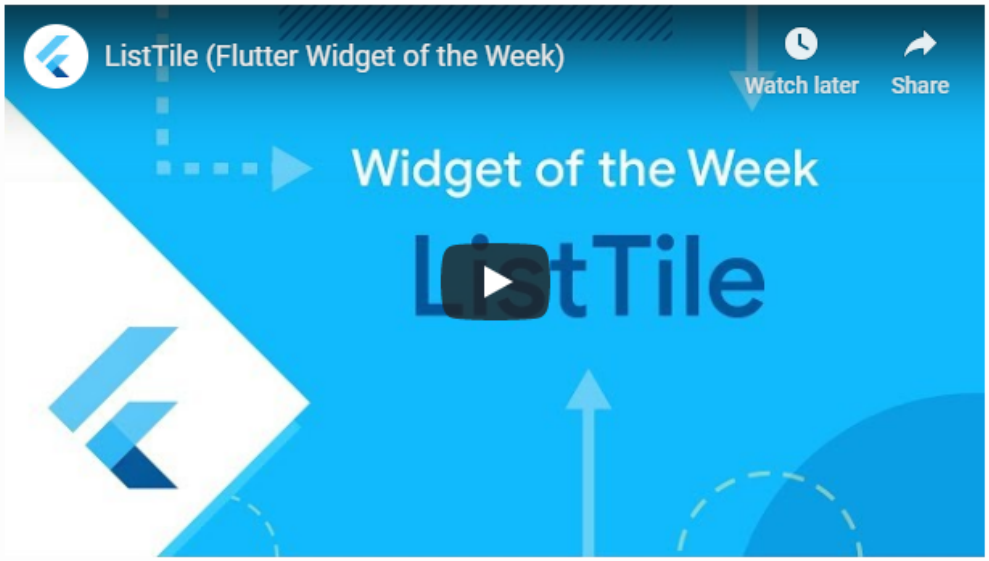
Search API Docs

CLASSES

AboutDialog
AboutListTile
AbsorbPointer
Accumulator
Action
ActionChip
ActionDispatcher
Actions
ActivateAction
AlertDialog
Align
Alignment
AlignmentDirectional
AlignmentGeometry
AlignmentGeometryTween
AlignmentTween
AlignTransition
AlwaysScrollableScrollPh...
AlwaysStoppedAnimation
AndroidView
Animatable

ListTile class

A single fixed-height row that typically contains some text as well as a leading or trailing icon.



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

DefaultTextStyles that wrap the title and subtitle widget are reduced.

CONSTRUCTORS

ListTile

PROPERTIES

contentPadding
dense
enabled
isThreeLine
leading
onLongPress
onTap
selected
subtitle
title
trailing
hashCode
key
runtimeType

METHODS

debugDescribeChildr...

Comp

COPY ALL

Lists

Flutter › material › ListTile class

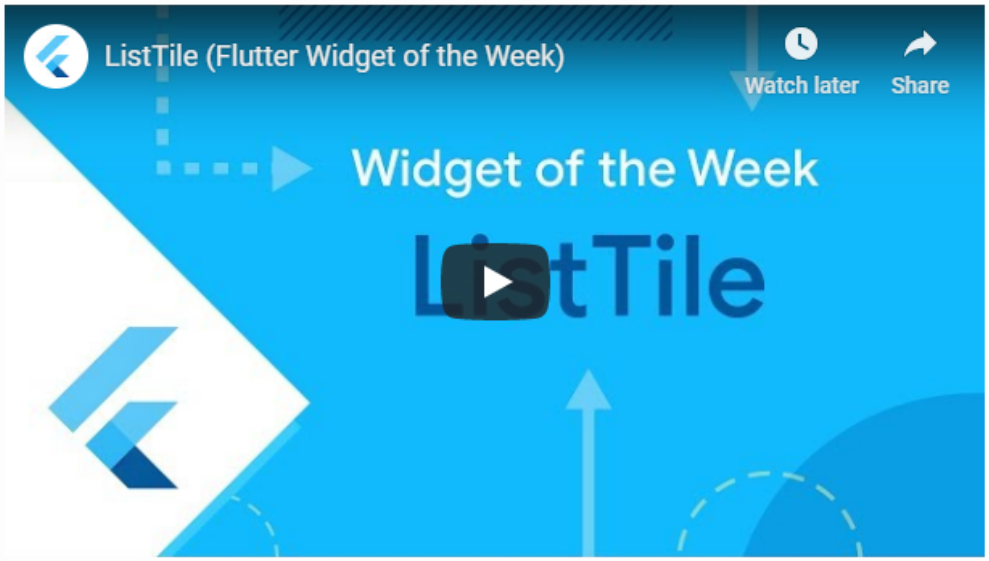
Search API Docs

CLASSES

AboutDialog
AboutListTile
AbsorbPointer
Accumulator
Action
ActionChip
ActionDispatcher
Actions
ActivateAction
AlertDialog
Align
Alignment
AlignmentDirectional
AlignmentGeometry
AlignmentGeometryTween
AlignmentTween
AlignTransition
AlwaysScrollableScrollPh...
AlwaysStoppedAnimation
AndroidView
Animatable

ListTile class

A single fixed-height row that typically contains some text as well as a leading or trailing icon.



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.

CONSTRUCTORS

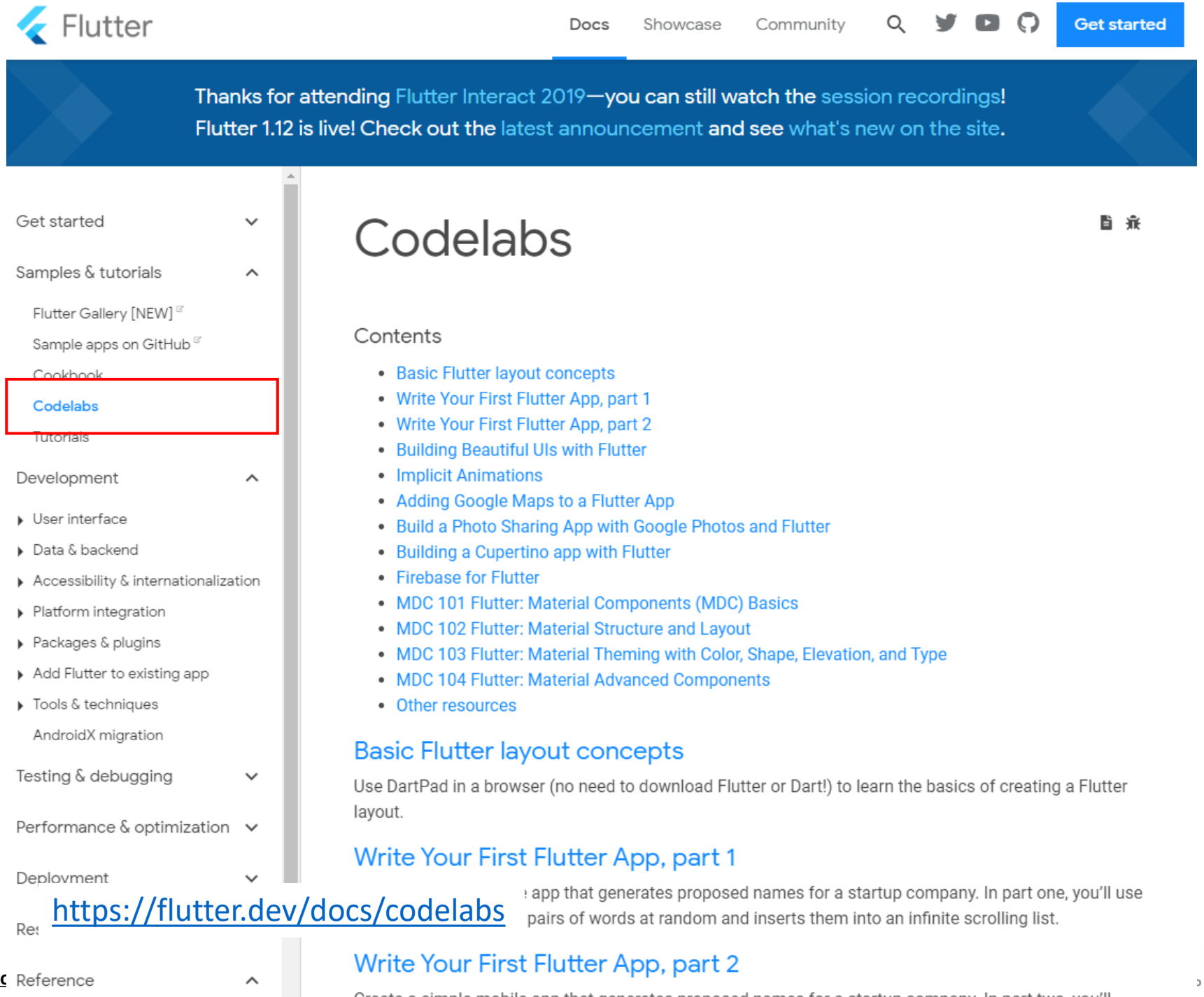
ListTile

PROPERTIES

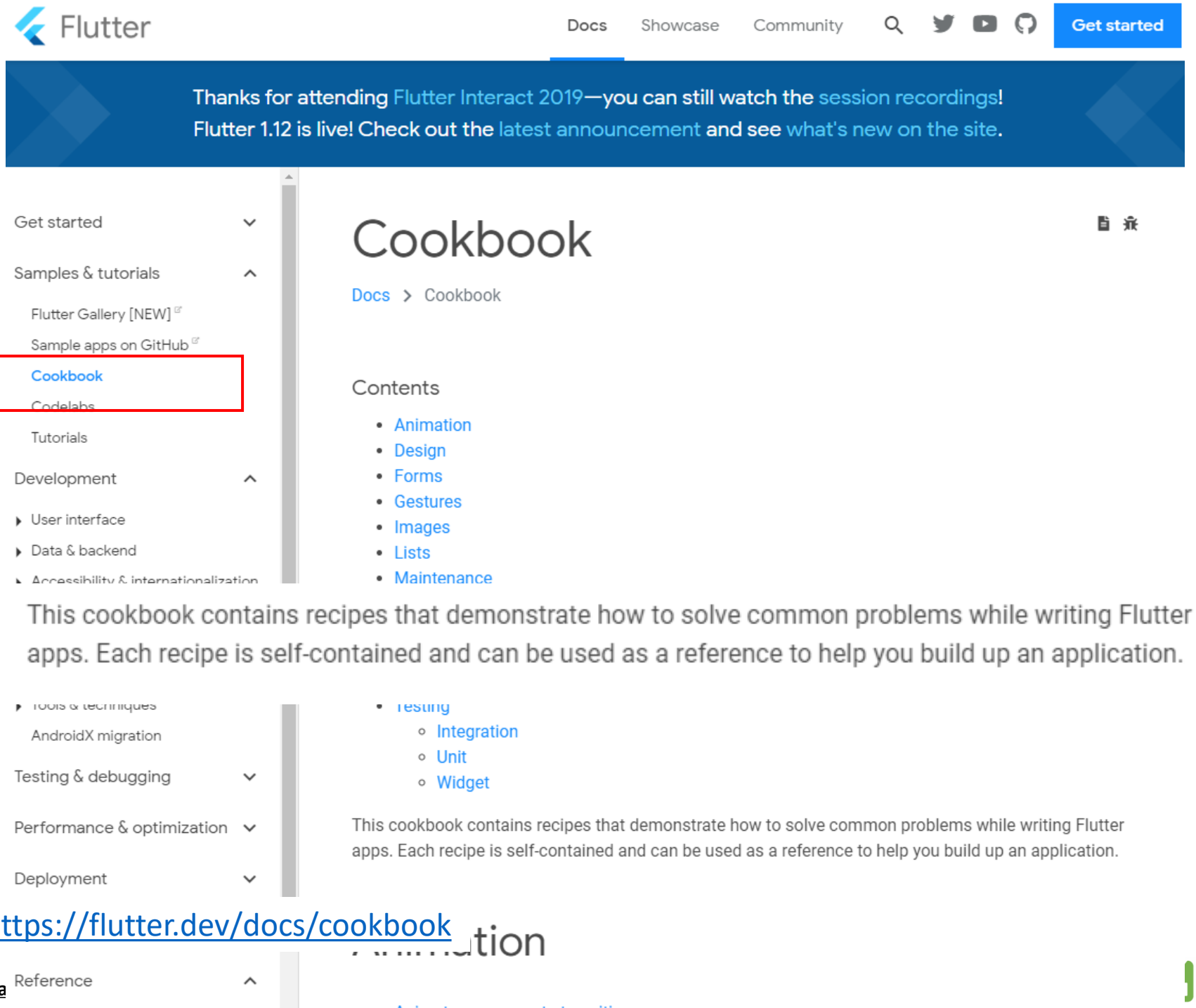
contentPadding
dense
enabled
isThreeLine
leading
onLongPress
onTap
selected
subtitle
title
trailing
hashCode
key
runtimeType

METHODS

debugDescribeChildr...

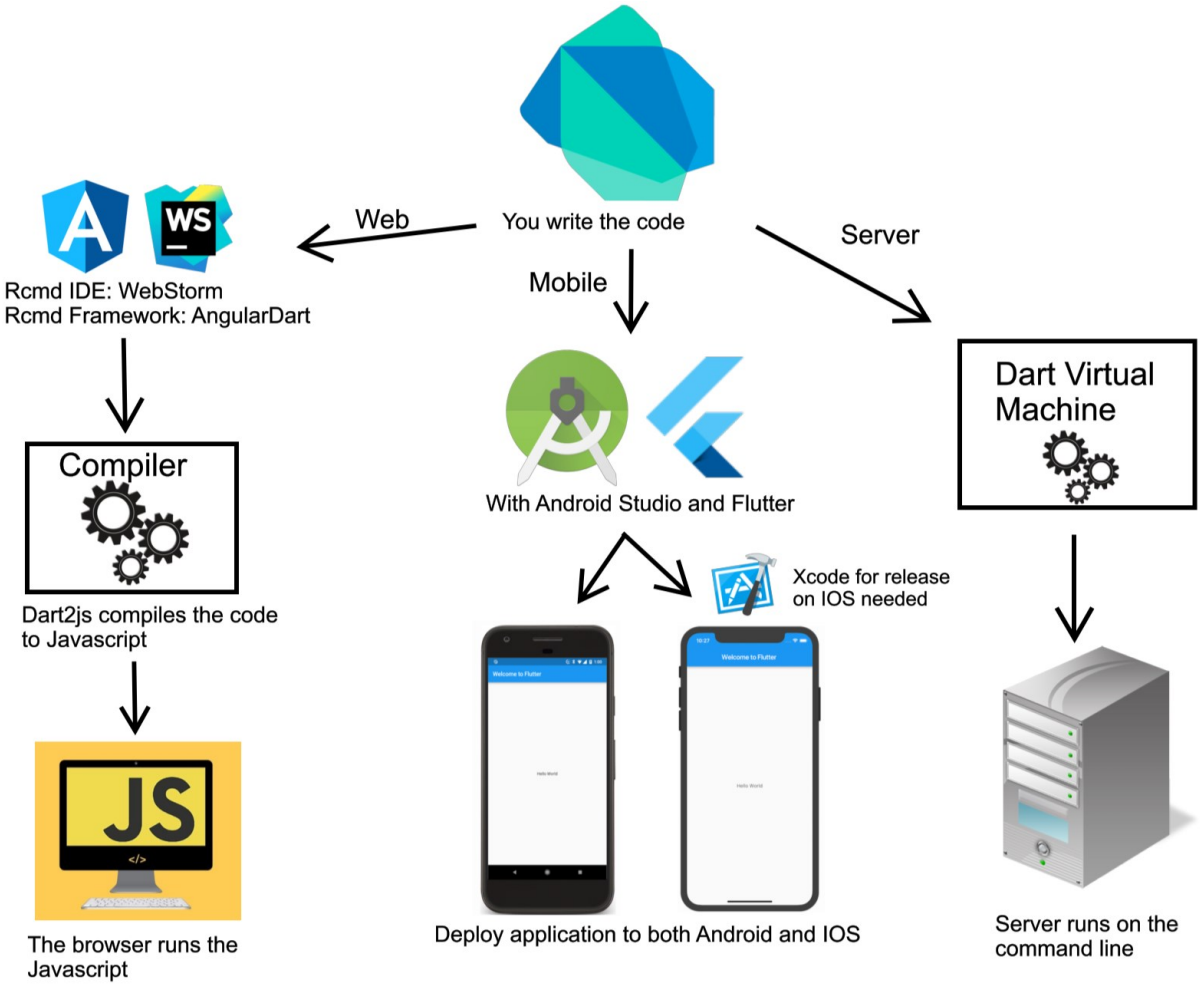


https://flutter.dev/docs/codelabs



https://flutter.dev/docs/cookbook

Flutter setup



developers | Platform | Android Studio | Google Play | Jetpack | More | Search | LANGUAGE | SIGN IN

Android Studio

DOWNLOAD | WHAT'S NEW | USER GUIDE | PREVIEW

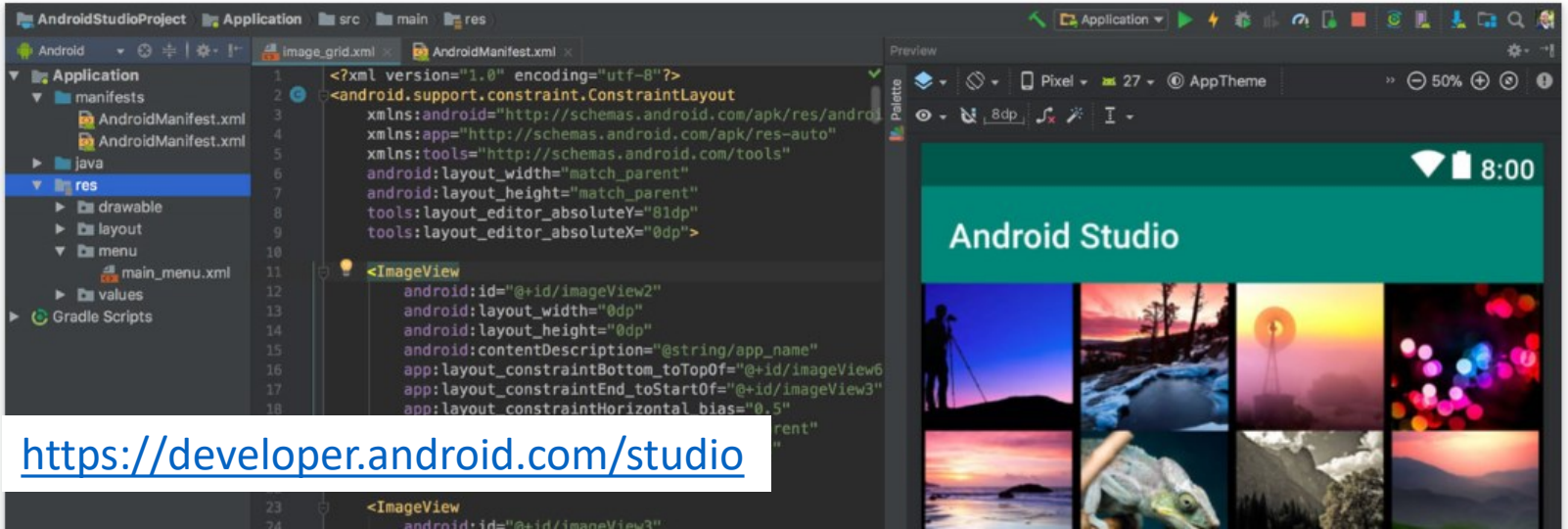
android studio

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

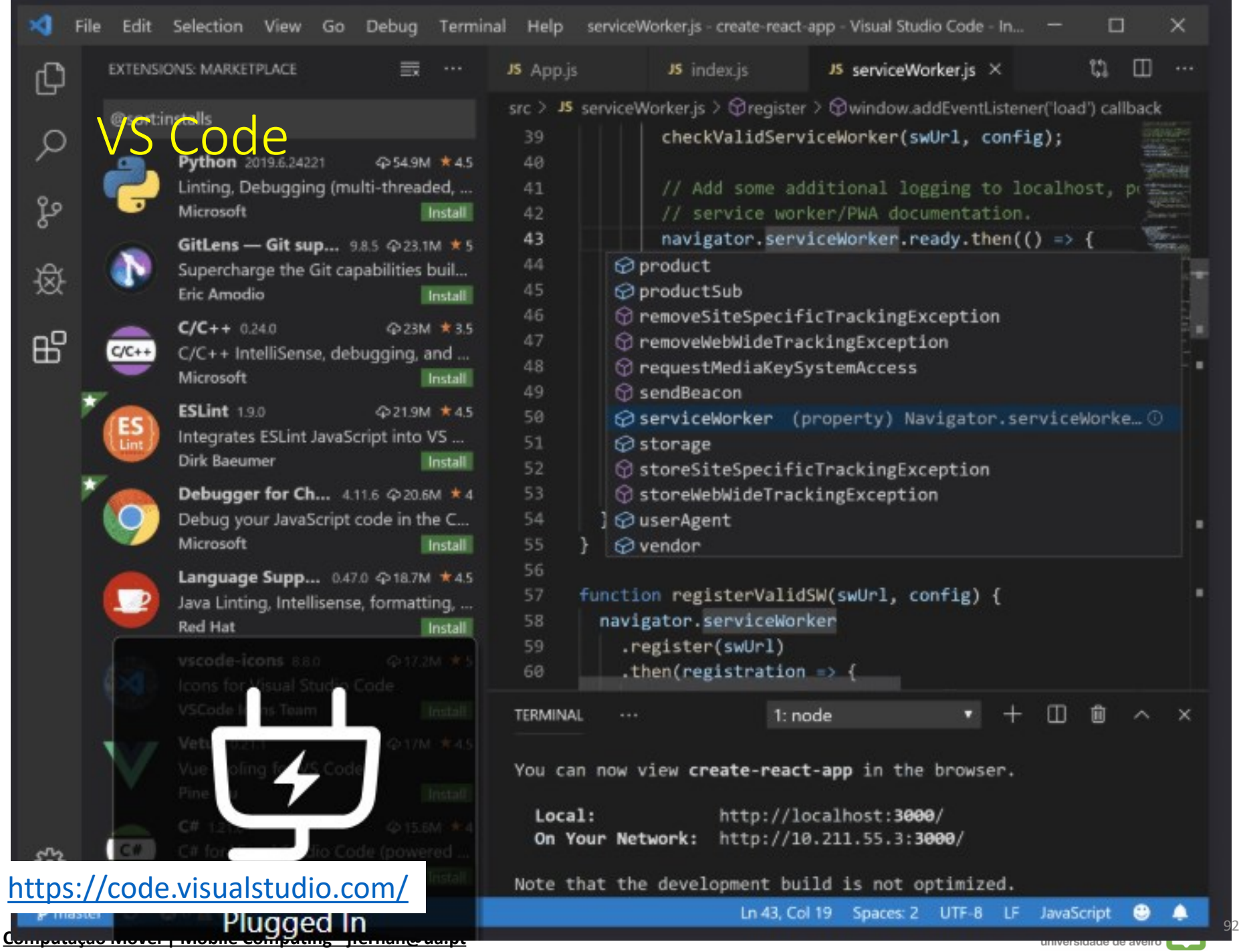
DOWNLOAD ANDROID STUDIO

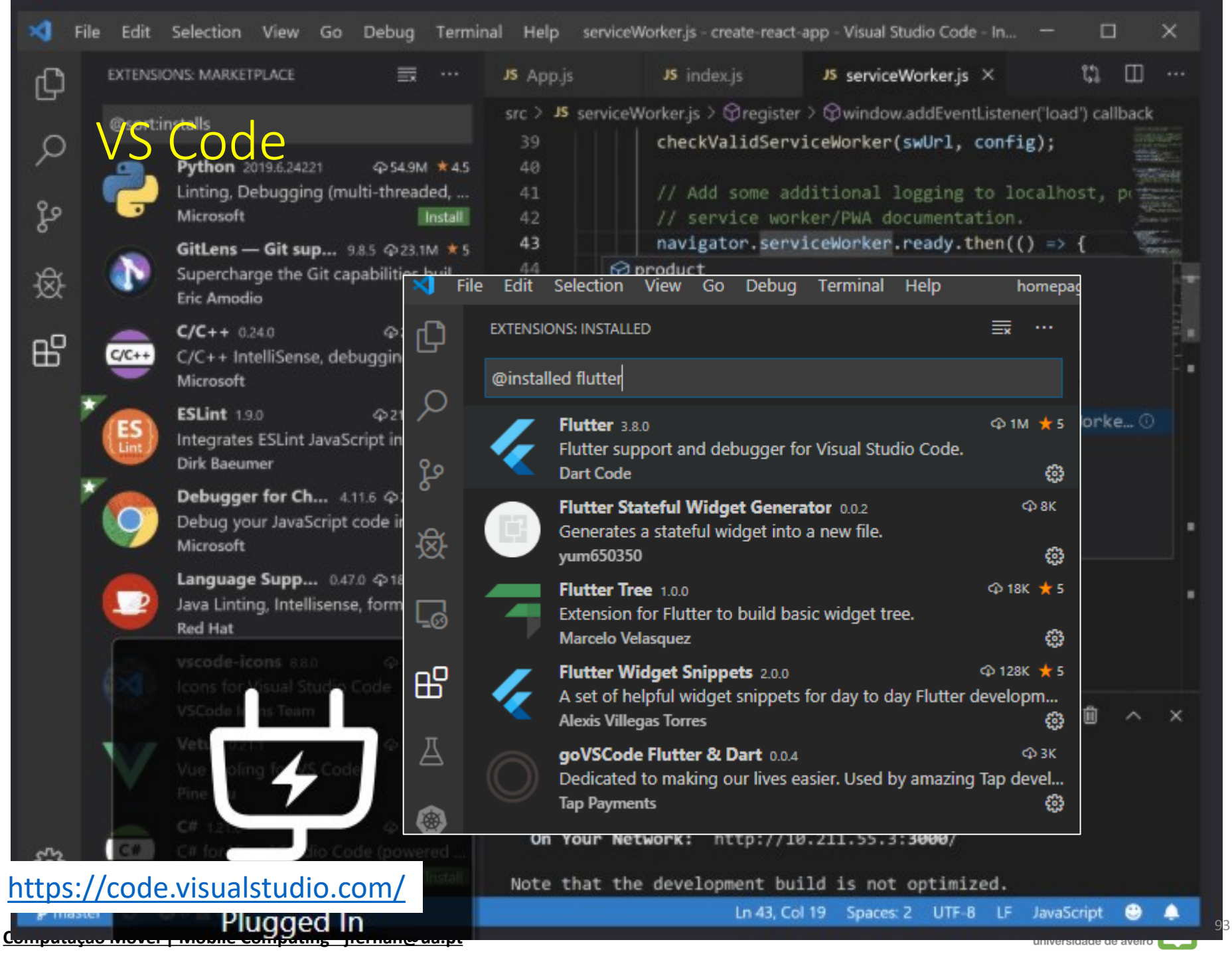
3.5.3 for Windows 64-bit (718 MB)

DOWNLOAD OPTIONS | RELEASE NOTES



https://developer.android.com/studio

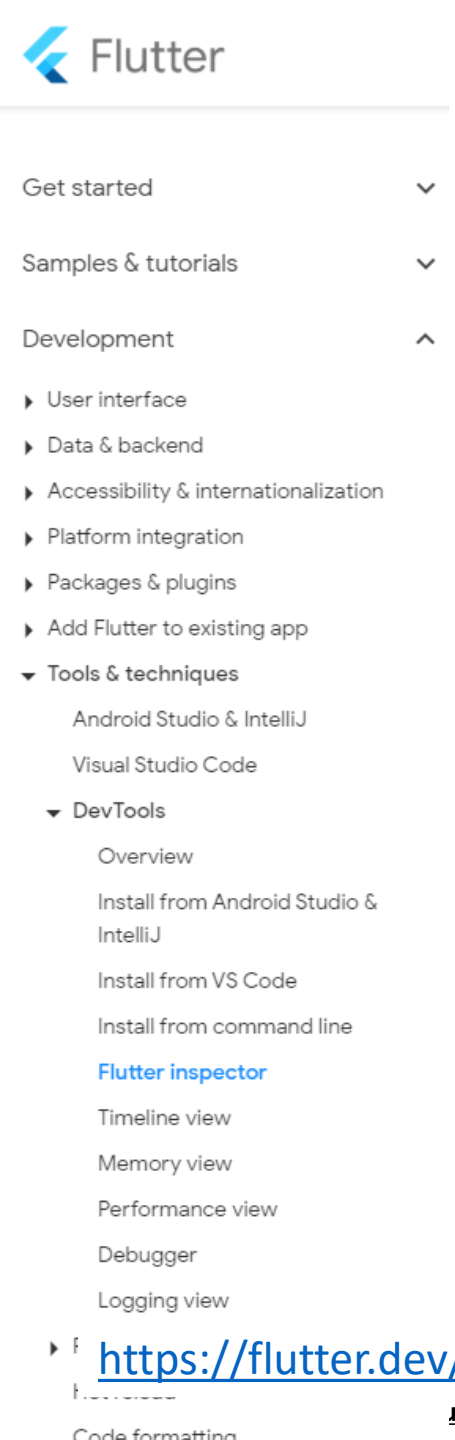

File Edit Selection View Go Debug Terminal Help serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
VS Code
Python 2019.6.24221 54.9M 4.5
Linting, Debugging (multi-threaded, ...
Microsoft
Install
GitLens — Git sup... 9.8.5 23.1M 5
Supercharge the Git capabilities buil...
Eric Amodio
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
JS App.js
JS index.js
JS serviceWorker.js
src > JS serviceWorker.js > register > window.addEventListener('load') callback
checkValidServiceWorker(swUrl, config);
// Add some additional logging to localhost, p
// service worker/PWA documentation.
navigator.serviceWorker.ready.then(() => {
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 => {
TERMINAL
1: node
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
https://code.visualstudio.com/
Plugged In


VS Code
File Edit Selection View Go Debug Terminal Help serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
App.js
index.js
serviceWorker.js
src > serviceWorker.js > register > window.addEventListener('load') callback
checkValidServiceWorker(swUrl, config);
// Add some additional logging to localhost,
// service worker/PWA documentation.
navigator.serviceWorker.ready.then(() => {
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
Eric Amodio
C/C++ 0.24.0
C/C++ IntelliSense, debuggin
Microsoft
ESLint 1.9.0
Integrates ESLint JavaScript in
Dirk Baeumer
Debugger for Ch... 4.11.6
Debug your JavaScript code i
Microsoft
Language Supp... 0.47.0
Java Linting, Intellisense, form
Red Hat
File Edit Selection View Go Debug Terminal Help
EXTENSIONS: INSTALLED
@installed flutter
Flutter 3.8.0 1M 5
Flutter support and debugger for Visual Studio Code.
Dart Code
Flutter Stateful Widget Generator 0.0.2 8K
Generates a stateful widget into a new file.
yum650350
Flutter Tree 1.0.0 18K 5
Extension for Flutter to build basic widget tree.
Marcelo Velasquez
Flutter Widget Snippets 2.0.0 128K 5
A set of helpful widget snippets for day to day Flutter developm...
Alexis Villegas Torres
goVSCode Flutter & Dart 0.0.4 3K
Dedicated to making our lives easier. Used by amazing Tap devel...
Tap Payments
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
https://code.visualstudio.com/
Plugged In

Flutter

- 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
- Resources

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



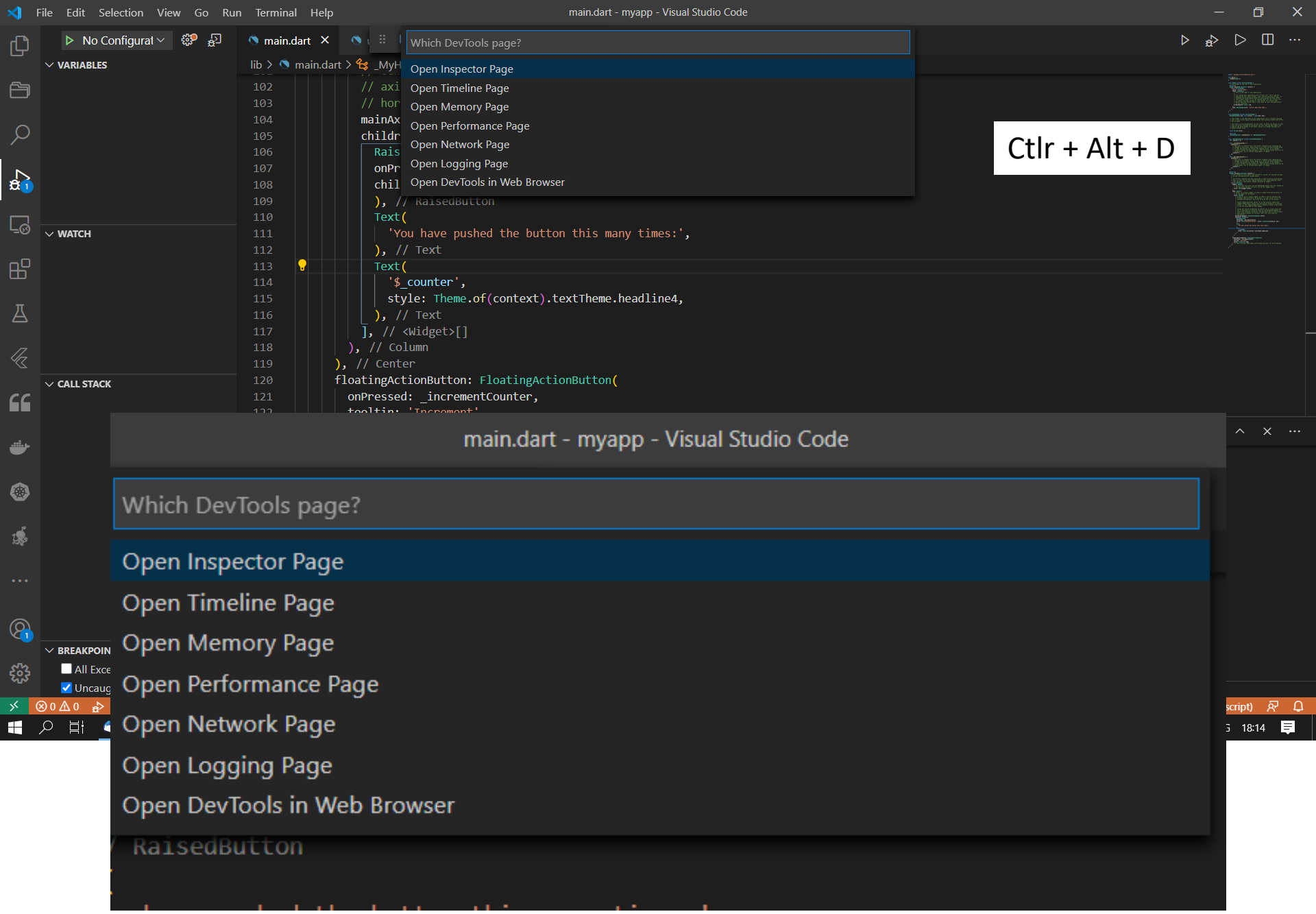
- Selecting a target device
- Fast edit and refresh development cycle
- Snippets

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

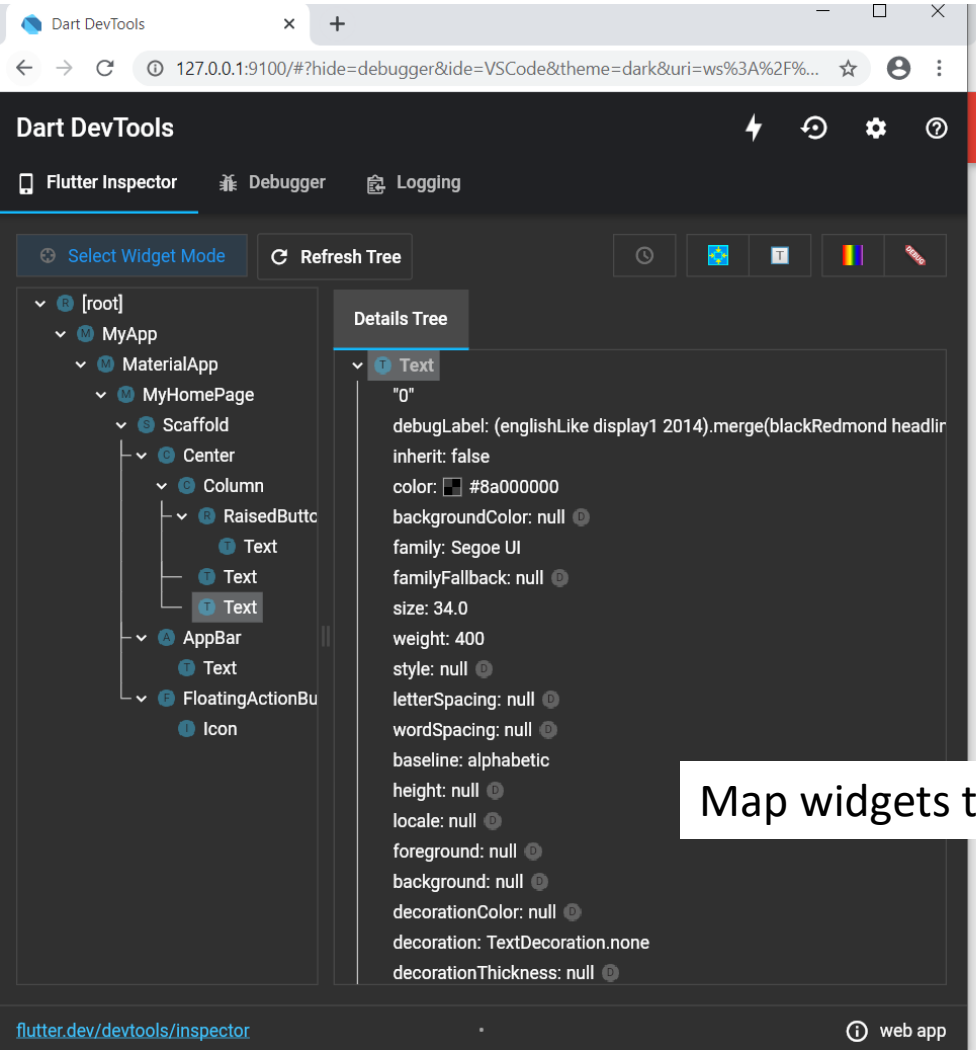
DevTools

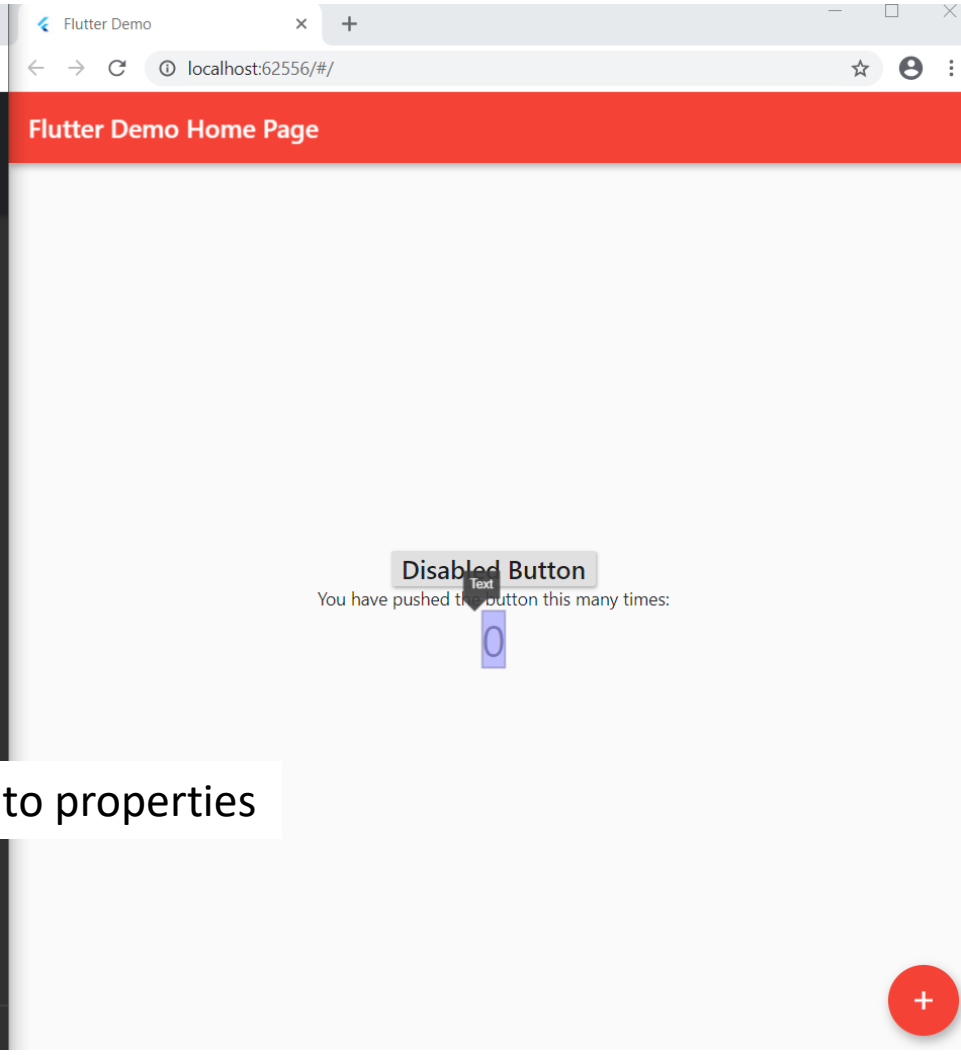
Topics:

https://flutter.dev/docs/development/tools/devtools


File Edit Selection View Go Run Terminal Help
main.dart - myapp - Visual Studio Code
No Configurat
main.dart
VARIABLES
WATCH
CALL STACK
BREAKPOIN
All Exce
Uncaug
lib > main.dart > _MyH
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
Ctlr + Alt + D
'You have pushed the button this many times:',
'$_counter',
style: Theme.of(context).textTheme.headline4,
floatingActionButton: FloatingActionButton(
onPressed: _incrementCounter,
RaisedButton
18:14

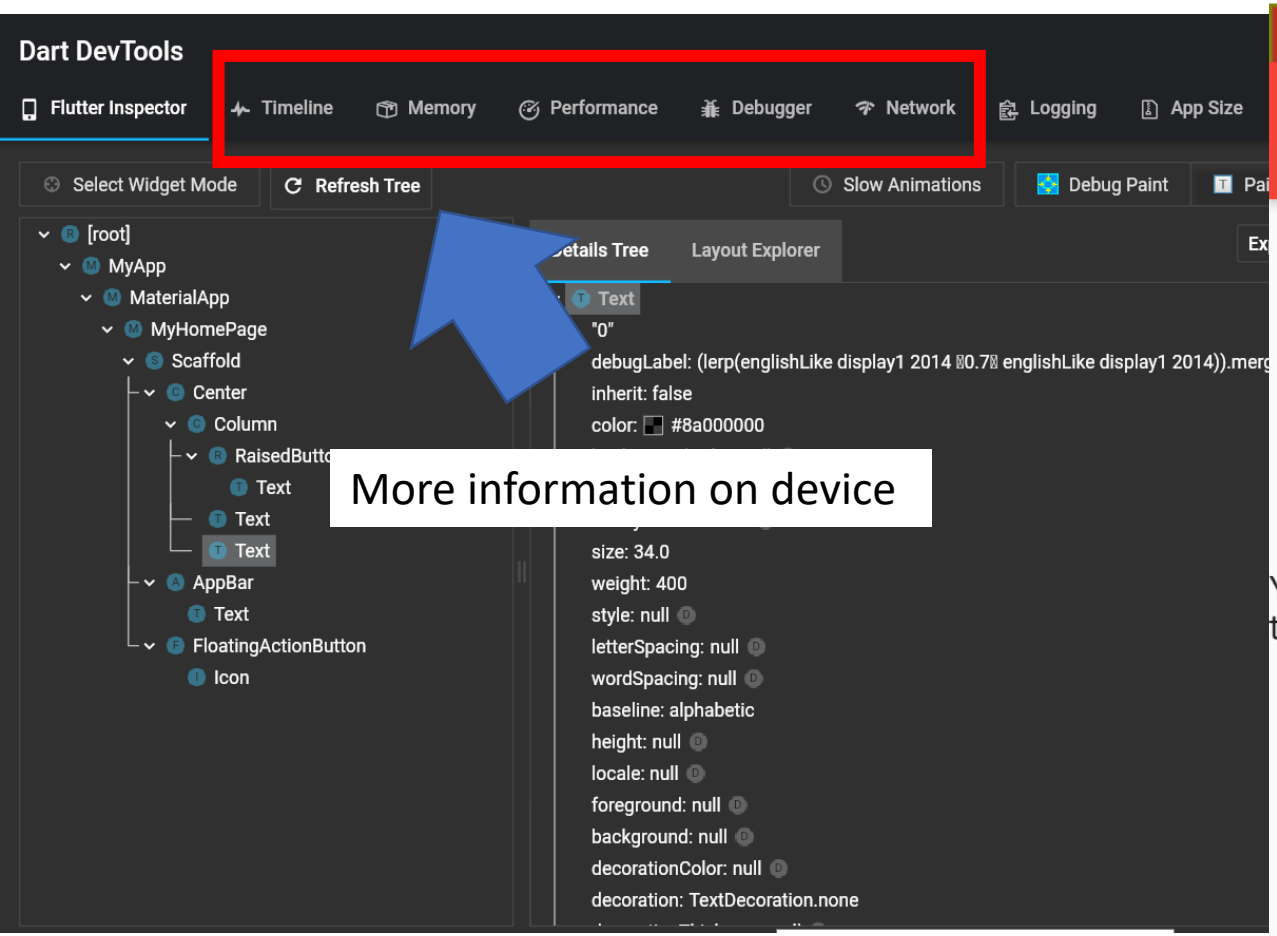
DevTools - inspector (in web browser)



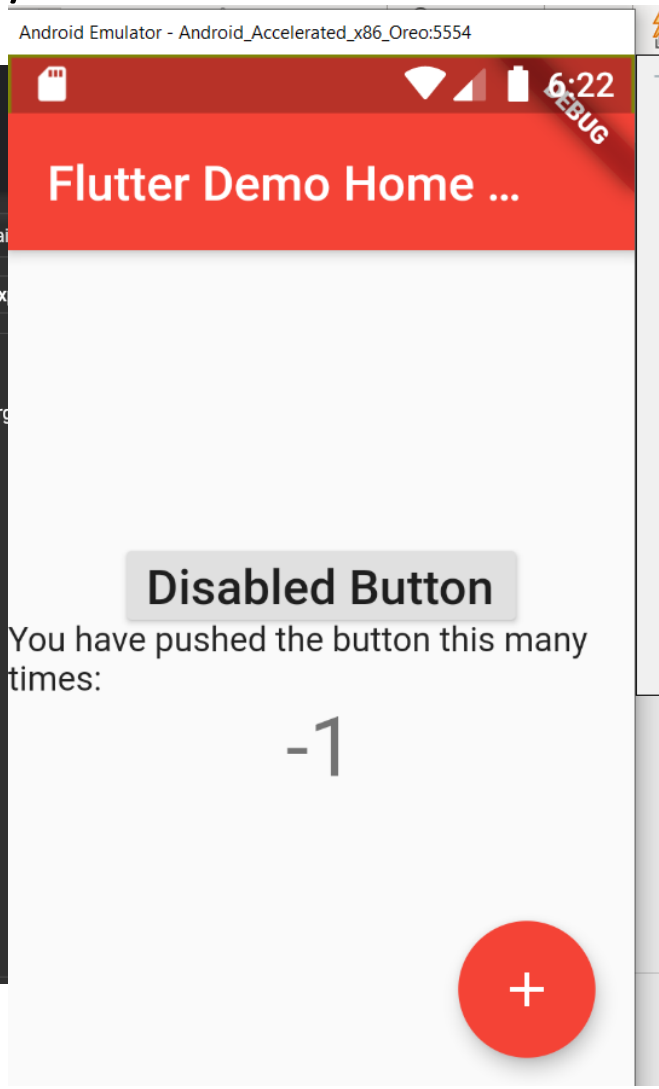


Map widgets to properties

DevTools – running in emulator / device



More information on device



FLUTTER

POSTED BY SEBASTIAN — 3RD NOVEMBER '19

Getting Started With Flutter – (2) Project Structure

LATEST ARTICLES

Building A Blog With Eleventy And Netlify CMS – Pa...
21ST SEPTEMBER '20

Building A Blog With Eleventy And Netlify CMS &#82...
20TH SEPTEMBER '20

Command-line Interfaces (CLIs) With React And Ink ...
9TH SEPTEMBER '20

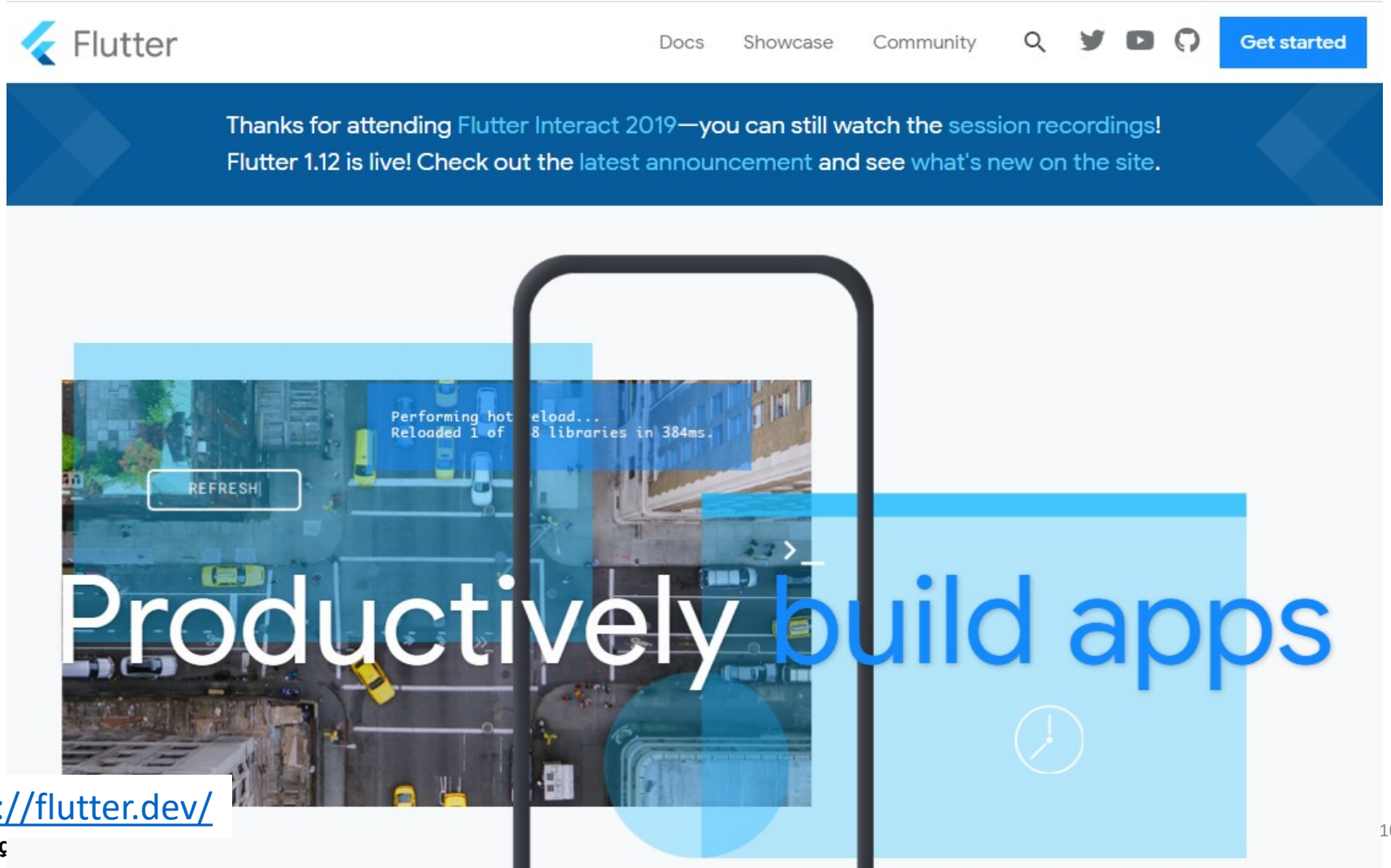
TAGS

Angular
Bootstrap
CSS
Docker
Flutter
Gatsby.js
GraphQL
HTML
Ionic 2
JavaScript
Machine Learning
Meteor
Mobile Development

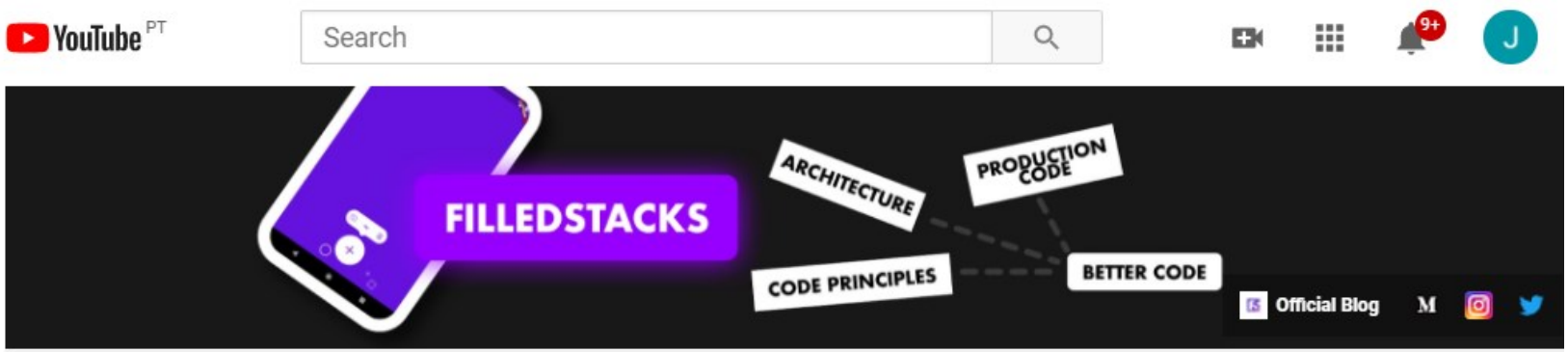
Getting Started With Flutter – (2) Project Structure
https://codingthesmartway.com/getting-started-with-flutter-project-structure/
https://www.youtube.com/watch?v=mzzWciSRgBs

Some recommend resources

Official: Dart & Flutter



https://flutter.dev/



FilledStacks

33.8K subscribers

SUBSCRIBED

HOME VIDEOS PLAY

Uploads ▶ PLAY ALL

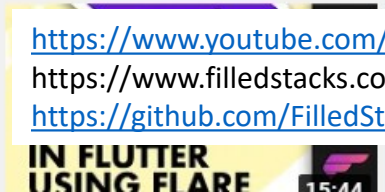
Flutter Theme Manager for Multiple Themes

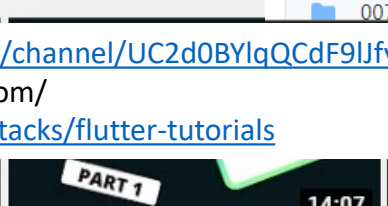
6.7K views • 1 week ago

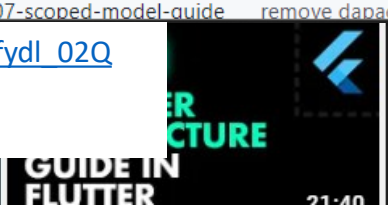
Lottie Splash Screen in Flutter

8.9K views • 3 weeks ago

Popular uploads ▶ PLAY ALL







FilledStacks / flutter-tutorials

Watch 167 Unstar

Code Issues 16 Pull requests 2 Actions Projects Wiki Security

master 4 branches 0 tags Go to file Add file Code

FilledStacks Merge branch 'master' of https://github.... ... 3c33df0 24 days ago 33 commits

001-tik_tok_ui	remove dapackages (it should only be a dev depen...	3 months ago
002-flutter-flare-pt1	remove dapackages (it should only be a dev depen...	3 months ago
003-smart-flare	remove dapackages (it should only be a dev depen...	3 months ago
004-flutter-basics	remove dapackages (it should only be a dev depen...	3 months ago
005-basic-navigation	remove dapackages (it should only be a dev depen...	3 months ago
006-flare-drawer	remove dapackages (it should only be a dev depen...	3 months ago
007-scoped-model-guide	remove dapackages (it should only be a dev depen...	3 months ago

About

The re
code f
the Fil
chann

flutter
dart
tutori

Re

MI

https://www.youtube.com/channel/UC2d0BYlqQCdF9lJfydI_02Q
https://www.filledstacks.com/
https://github.com/FilledStacks/flutter-tutorials

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)

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)

Creating Reusable Custom Widgets in Flutter

custom widgets in Flutter that you can use in any of your projects or share with the world.

Beginning Flutter Debugging

Core Concepts

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

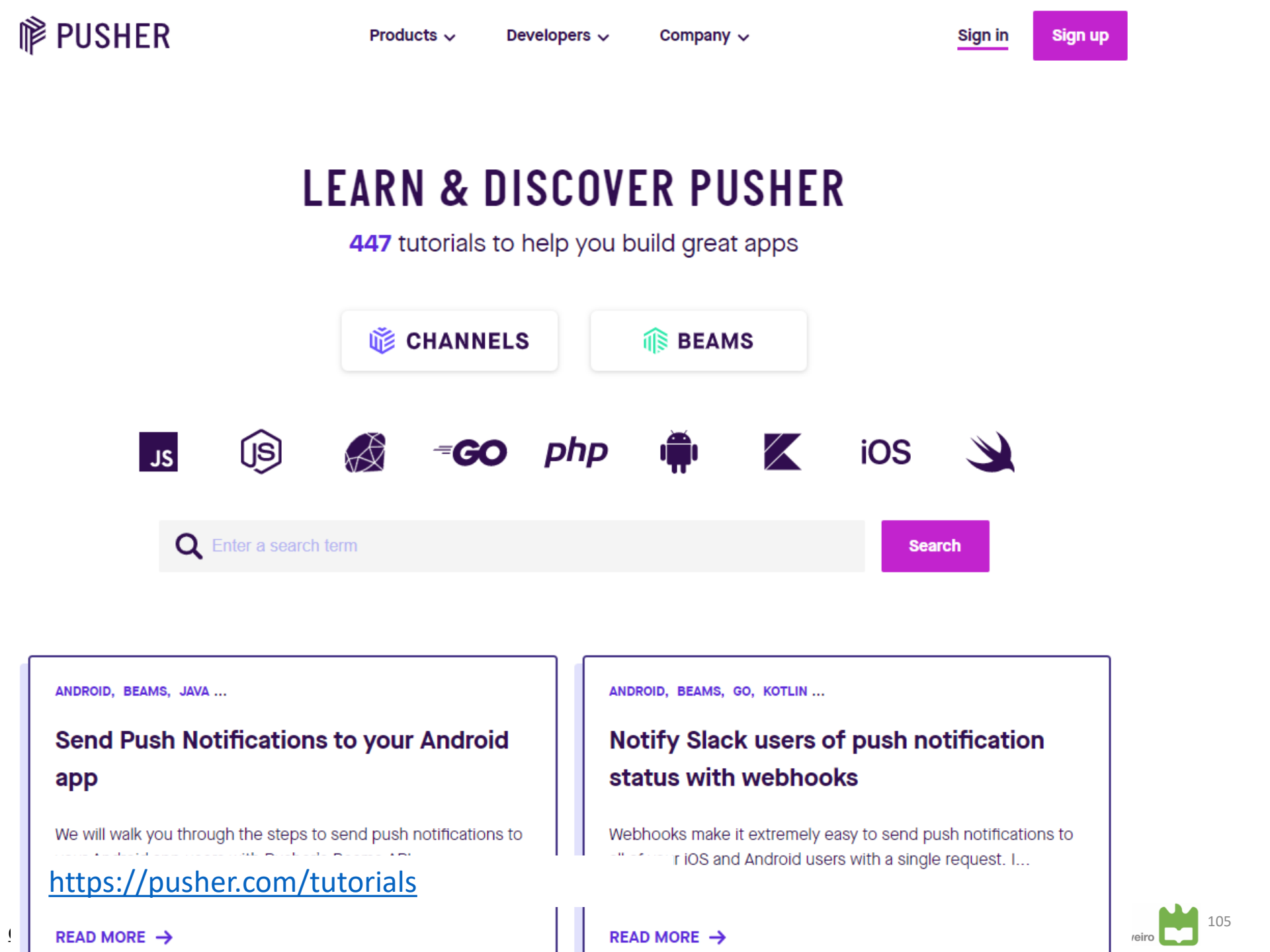
Aug 4 2020 · Video Course (1 hr, 18

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.

https://www.raywenderlich.com/flutter



https://pusher.com/tutorials

LEARN FLUTTER

absolute beginners

Flutter Tutorial

PDF Version | Quick Guide | Resources | Job Search | 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

This tutorial is written assuming that the readers are already aware about what a Framework is and ...at 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...

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

Some links that might give some context...

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

The END