

Flutter Training

[MUST READ - The Withcenter Policy](#)

[MUST Master - \[The Basic Knowledge of Development - Withcenter Development Practice \]](#)

[Must - Master Programming Language](#)

[Resources](#)

[Important Flutter training Course ** IMPORTANT **](#)

[Important Source Code - For re-using](#)

[First thing to Learn Before Flutter](#)

[Quick start - Create sample project and Run](#)

[To select a device to use to debug run](#)

[Run the project using -d option](#)

[Run from main.dart in VSCode](#)

[Run from VSCode settings](#)

[What is flutter?](#)

[What is Flutter SDK?](#)

[Dart SDK](#)

[Flutter SDK](#)

[So, where is it?](#)

[Then, How to use it?](#)

[pubspec.yaml & SDK constraints](#)

[Flutter SDK constraints](#)

[Then, Where is the Dart SDK?](#)

[Ex\)](#)

[How to install Flutter?](#)

[Things to know first](#)

[Anatomy of StatelessWidget](#)

[Anatomy of StatefulWidget](#)

[First app - Test drive](#)

[Learning Dart Language](#)

[Write your first Flutter app](#)

[Flutter Basic](#)

[UI - Animation, Design, Effects, FORM](#)

[Firebase](#)

[Firebase Installation](#)

[Firebase Authentication](#)

[Firebase Profile Update](#)

[Day 0 - How run Flutter](#)

[Day 1 - StatelessWidget, StatefulWidget, Typing, Types](#)

[Day 2 - Layout - Constraints go down, Sizes go up, Parents set the Position **](#)

[The nature of Container](#)

[The nature of other widgets ??](#)

[Know the nature of the widgets](#)

[Day 3 - How to Use Material 3 and Theme, Layout](#)

[Day 4 - How to use global variables to use consistent UI, spaces and sizes.](#)

[Day 5 - Provider - state management](#)

[Day 6 - Go Router](#)

[Day 7 - HTTP, Modeling](#)

[Day 8 - Firebase](#)

[Firebase Database](#)

[How to get newly added document](#)

[Day 9 - Image & File Upload](#)

[Building a package](#)

Resources

Important Flutter training Course ** IMPORTANT **

Paid Tutorial - Fireship

Learn from [Fireship.io - Dart](#)

Learn from [Fireship.io - Flutter & Firebase](#)

Learn from [Fireship.io - Firebase Security](#)

Free tutorial but very good

<https://heyflutter.com/courses/slkfjfACehTdLLgscSRq/LwfWEy28TUGXyltgc8HU>

Important Source Code - For re-using

- Paid Flutter UI Library and Mock-up Apps - ProKit
- Install it on mobile phone with release mode to see it better.

<https://github.com/thruthesky/prokit>

It works with; Flutter version 3.10.4

```
% .fvm/flutter_sdk/bin/flutter --version
Flutter 3.10.4 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 682aa387cf (3 months ago) • 2023-06-05 18:04:56 -0500
Engine • revision 2a3401c9bb
Tools • Dart 3.0.3 • DevTools 2.23.1
```

First thing to Learn Before Flutter

Most developers who end up quitting without gaining after 20 years of working didn't learn the very basic thing when they first start.

MUST Master - [[The Basic Knowledge of Development - Withcenter Development Practice](#)]

☰ UI/UX - Withcenter Study 2023

Quick start - Create sample project and Run

Install Flutter

1. Check the requirements
To install on Mac, there are requirements.

- See [the MacOS System Requirements](#)
2. Get the Flutter sdk from github
See [the guideline](#)
 3. [Update your path](#)
Remember: 99.99% of developers and people are experiencing difficulties installing because of the path.

Done. You have installed the Flutter on your frame(box, machine).

Create your project

% flutter create project-name

Change directory (go into) to newly created project.

% cd project-name

Run the project using **flutter run** command.

% flutter run

When you do this,

It will build, link, signing the app (of the project)

then, it will ask which device you want to use (to run the app). Or maybe not.

To select a device to use to debug run

First, you can see(list) which devices are available by **flutter devices**

```
% flutter devices
Multiple devices found:
iPhone11MaxPro (mobile)    • 00008030-E    • ios    • iOS 15.6.1 19G82
iPhone 14 Pro Max (mobile)  • 8A752CFC5E    • ios    • com.apple.CoreSimulator.SimRiOS-(simulator)
macOS (desktop)            • macos        • darwin-arm64   • macOS 13.3.1 220a darwin-arm64
```

See the purple colored word. That's the Device ID. It's important to know.

When you run, flutter is asking you which device you want to use. You can select the number.

```
% flutter run  
[1]: iPhone11MaxPro (00008030-000904C80290802E)  
[2]: iPhone 14 Pro Max (8A752C04-06F3-4B21-9B5E-BEEDF313FC5E)  
[3]: macOS (macos)  
[4]: Chrome (chrome)  
Please choose one (To quit, press "q/Q") : 2
```

But isn't it extra work to choose the device? You have lots of, lots of, lots of things to do in a limited amount of time. If you are a pro, then you should work as a pro. And pros don't choose the number.

Run the project using -d option

You can use the `-d` option to specify which device you want to use.

For instance, the device id of `iPhone 14 Pro Max` is
`8A752C04-06F3-4B21-9B5E-BEEDF313FC5E`.

So, you can do this

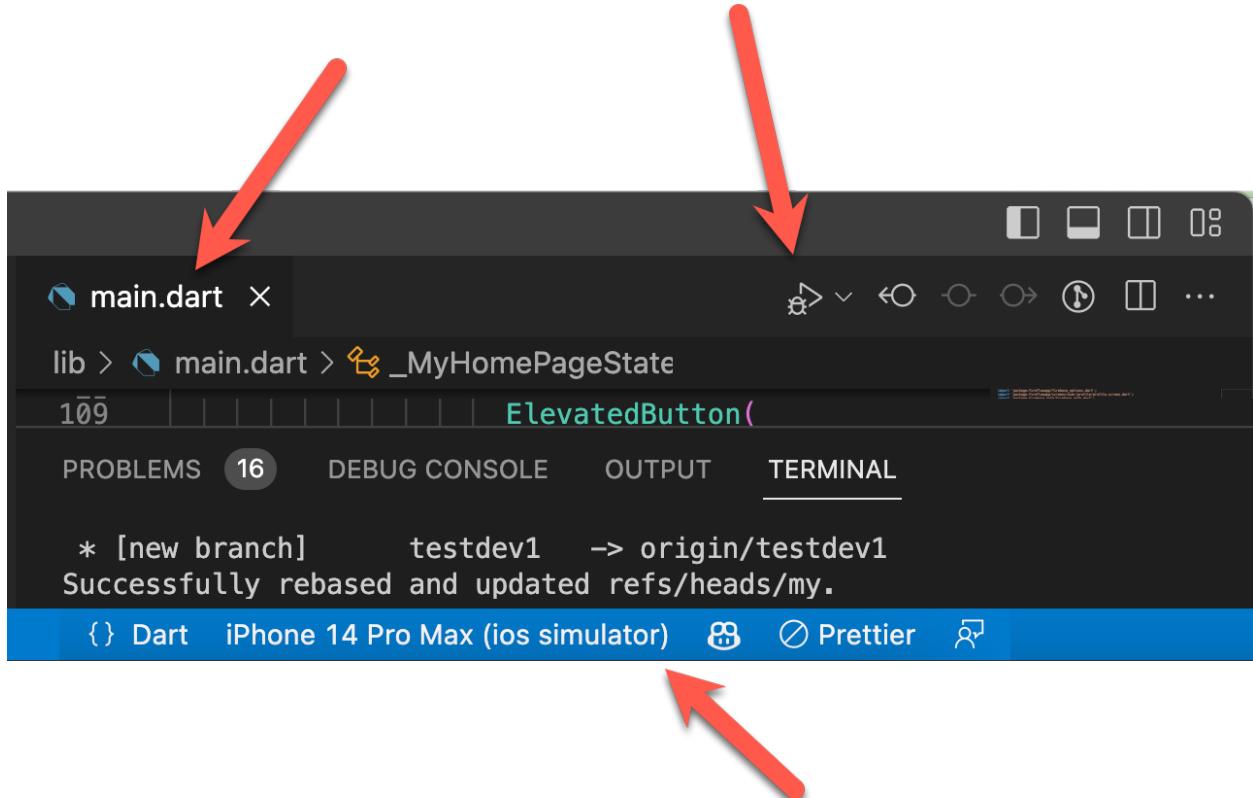
```
% flutter run -d 8A752C04-06F3-4B21-9B5E-BEEDF313FC5E
```

And it will run on iPhone 14 Pro Max.

But still, it's extra work. It's not what professionals do.

Run from main.dart in VSCode

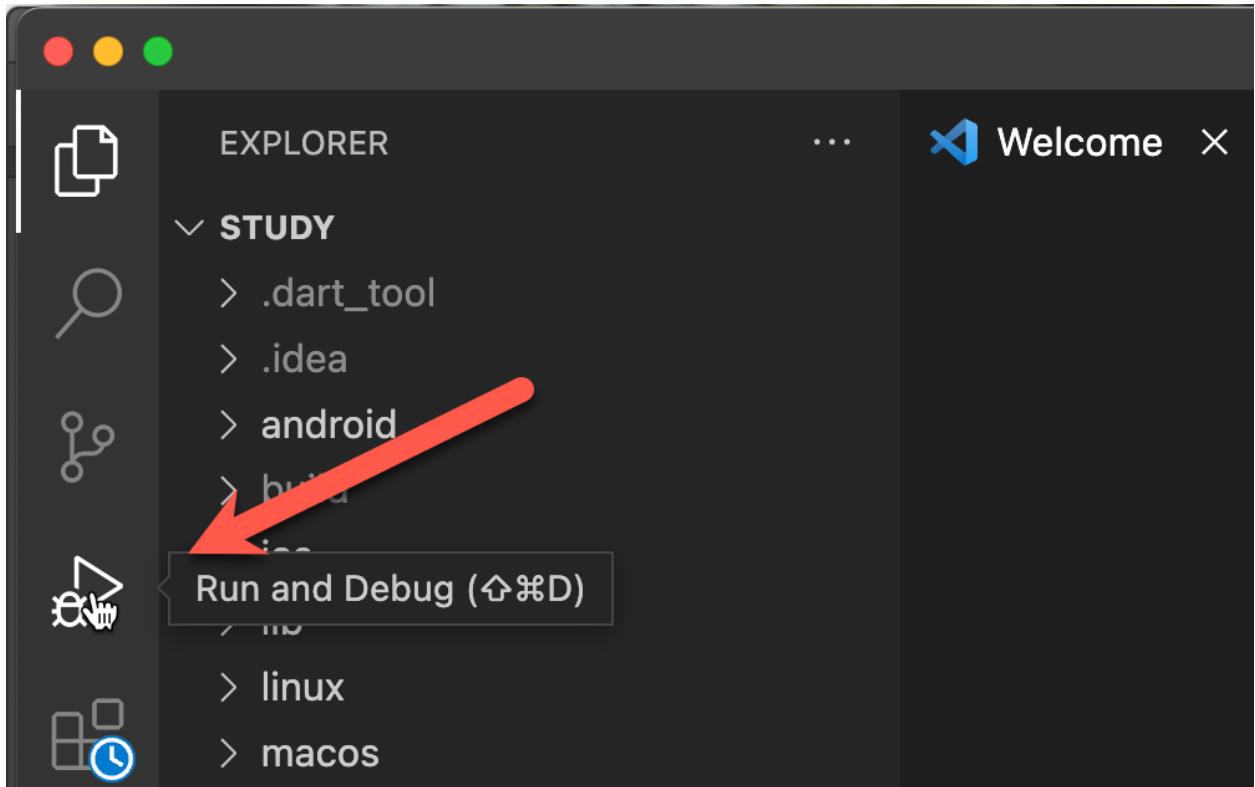
You can open main.dart in VSCode and choose which device you want to run, then press the Debug button to run.



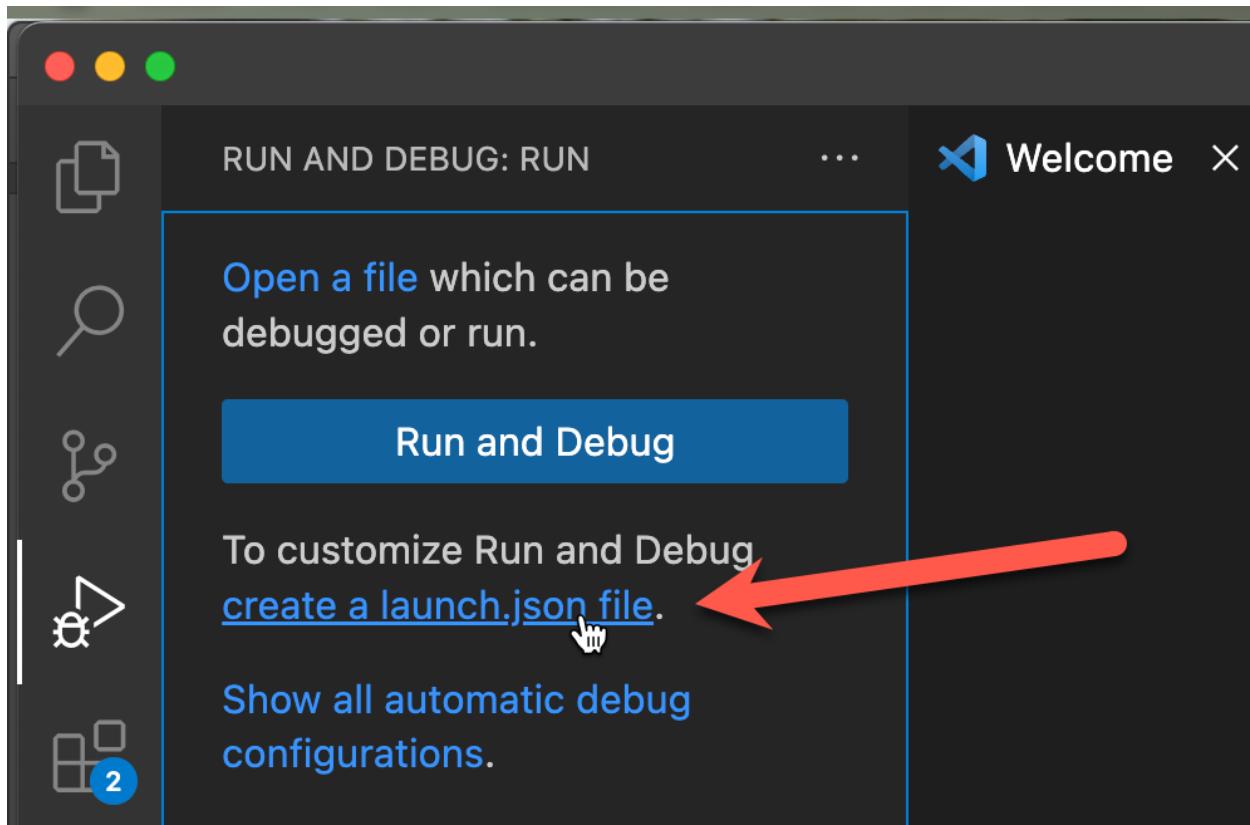
Run from VSCode settings

The better way of running your app is to add the settings on your VSCode settings.

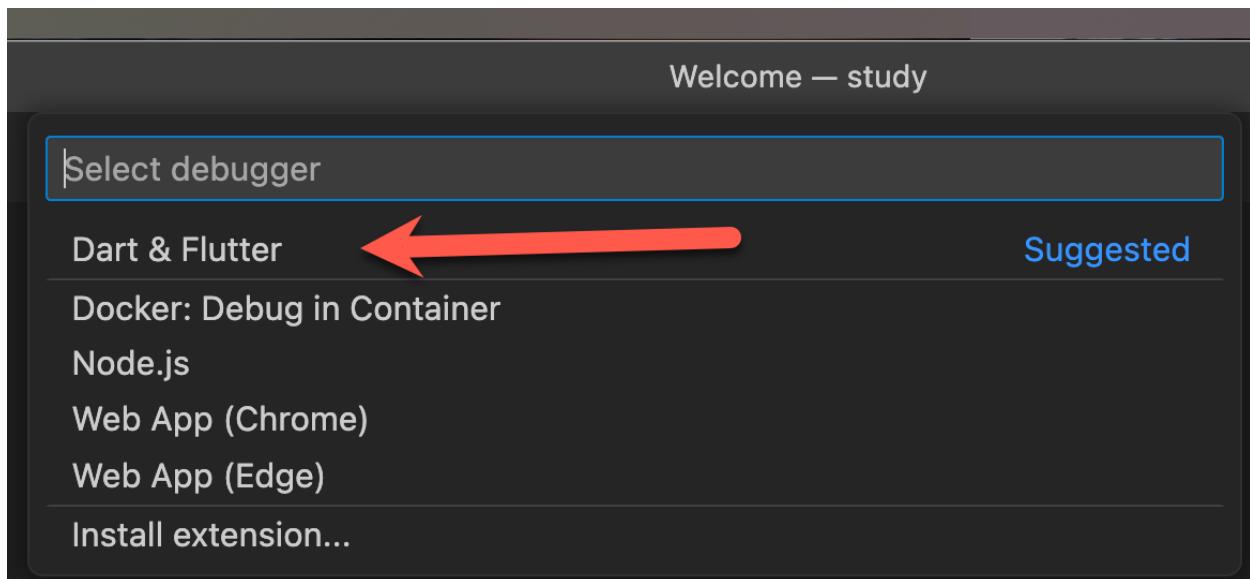
- Click Debug Settings



- Create launch.json file



- Click Dart & Flutter



- Then, add
 - name, request, type, program, flutterMode, deviceld like below.
 - Then, you can click the name on the dropdown in left panel.

The screenshot shows the VS Code interface with the following details:

- Left Panel:** Shows a dropdown menu with options: "study - Simulator - Song", "study - iPhone11ProMax - Song" (highlighted with a blue box), "study (profile mode)", "study (release mode)", "Node.js...", "Dart & Flutter...", and "Add Configuration...".
- Code Editor:** The file "launch.json" is open, showing the configuration for the selected launch target. The code is as follows:

```

1  {
2    // Use IntelliSense to learn about possible attributes.
3    // Hover to view descriptions of existing attributes.
4    // For more information, visit: https://go.microsoft.com/
5    "version": "0.2.0",
6    "configurations": [
7      {
8        "name": "study - Simulator - Song",
9        "request": "launch",
10       "type": "dart",
11       "program": "lib/main.dart",
12       "flutterMode": "debug",
13       "deviceId": "8A752C04-06F3-4B21-9B5E-BEEDF313FC5E"
14     },
15     {
16       "name": "study - iPhone11ProMax - Song",
17       "request": "launch",
18       "type": "dart",
19       "program": "lib/main.dart",
20       "flutterMode": "debug",
21       "deviceId": "00008030-000904C80290802E"
22     }
  ]

```

- Terminal:** Shows the command "flutter devices" and its output, which lists four connected devices:

iPhone11MaxPro (mobile)	00008030-000904C80290802E
iPhone 14 Pro Max (mobile)	• 8A752C04-06F3-4B21-9B5E-BEEDF313FC5E
macOS (desktop)	macos
Chrome (web)	chrome

What is flutter?

What is Flutter SDK?

Dart SDK

Dart is a programming language. It's a language only.

You need to have more tools to develop applications(softwares) with Dart programming language.

Tools something likes

- compiler, builder, linker and more to develop.
- Packager, installer to distribute.

These are SDK. and it's for Dart. so it's called Dart SDK.

Flutter SDK

Dart is a programming language and Dart SDK is for building softwares using Dart programming language.

You can develop your own UI library for your app using Dart programming language. But it's not easy and it's a huge amount of work. So, google provides one for you.

It's called a UI library named Flutter.

But Flutter is merely a UI library. You may really use it for building web/apps with it.

But you will spend lots of time and effort to build without its helper tools like Flutter boilerplate code generation, Flutter UI library run/managing process, Flutter Debug Tools.

With the help of tools, you can build better with ease.

So, Flutter SDK means, Flutter UI library with help tools.

So, where is it?

It's on github. And when you install Flutter, it will clone the source repo into your computer.

In MacOS, it's usually inside ~/bin/flutter

Flutter SDK: **~/bin/flutter**

Flutter UI Library: **~/bin/flutter/packages/flutter**

Then, How to use it?

Since the UI Library is inside **~/bin/flutter/packages**, just import as **import package**:

The **import package**: will point to **~/bin/flutter/packages**

So, **import package:/flutter** will point to **~/bin/flutter/packages/flutter**.

And most of the time, you will use Material Design.

So, you need to use the Material UI Library inside the Flutter UI Library.

To use it, you need to import it. The file name of the Material UI is **material.dart** which exists under **~/bin/flutter/packages/flutter/lib** folder. All the source code of Flutter goes in **/lib** folder. So, it is **~/bin/flutter/packages/flutter/lib** folder.

So, **import package:flutter/material.dart** points to
~/bin/flutter/packages/flutter/lib/material.dart

The **material.dart** has all the widgets you are using when you build Flutter app like

- Text, Container, SizedBox, Row, Column, Stack or whatever are included by **material.dart**

pubspec.yaml & SDK constraints

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

The above is called SDK constraints.

SDK constraints)

```
sdk: ">=2.12.0 <3.0.0"
```

From Official doc: <https://dart.dev/tools/pub/pubspec#sdk-constraints>

SDK constraints is;

A package can indicate which versions of its dependencies it supports, but packages have another implicit dependency: the Dart platform itself

For example, the following constraint says that this package works with any [Dart SDK](#) that's version 3.0.0 or higher:

```
environment:  
  sdk: ^3.0.0
```

It's clearly indicating that the SDK is Dart SDK.

It means, A package can define which version of SDK it supports.

If a dart program(project) has a [pubspec.yaml](#), it's called a package. (from Mr. Song)

A package can be published to [pub.dev](#).

You can set the package (updating pubspec.yaml) not to be published (by accident) using, [publish_to: 'none'](#) option. Especially when it is a flutter application, you should use this option.

```
environment:  
  sdk: ">=2.12.0 <3.0.0" // This package will depends on >=2.12 <3  
  of Dart SDK  
  // the pubspec.yaml contains settings for a dart package.  
  // The sdk constraints tells that this package will only work  
  between >=2.12 and <3 version of Dart SDK.  
  
dependencies:
```

```
flutter:  
  sdk: flutter // This is the Flutter SDK. This package uses  
this version of Flutter SDK.  
  
  app_links: 3.4.1  
  auto_size_text: 3.0.0
```

Flutter SDK constraints

You can specify the flutter sdk version additionally.

```
environment:  
  sdk: '>=1.19.0 <3.0.0'  
  flutter: ^0.1.2
```

Then, Where is the Dart SDK?

Dart and its environment evolves over time. And things change.

Before you have to install Dart SDK and Flutter SDK separately.

But now, the Flutter SDK contains Dart SDK. So, Flutter SDK will install Dart SDK and put the Dart SDK in the same place as Flutter SDK.

It's in `~/bin/flutter`.

And most of the tools are in `~/bin/flutter/bin`.

Ex)

<https://github.com/flutter/flutter> is the flutter.

You put a Text widget.

```
You, 1 second ago | 1 author (You)
class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: const <Widget>[
            Text(
              'Hi, there!!',
            ), // Text
          ],
        ),
      ), // Center
    ); // Scaffold
  }
}
```

Where is it coming from?

The answer is it's coming from Flutter. But what is flutter?

The flutter is <https://github.com/flutter/flutter>

Meaning, the Text widget is coming from <https://github.com/flutter/flutter>.

Where? Tell me exactly where it is coming from.

The answer is:

<https://github.com/flutter/flutter/blob/master/packages/flutter/lib/src/widgets/text.dart>

How to install Flutter?

You can use brew.

```
% brew install flutter
```

You can simply clone

```
% git clone https://github.com/flutter/flutter
```

You can download and install from the official website.

```
environment:  
  sdk: ">=2.19.2 <3.0.0"
```

Things to know first

- Style Guide
 - [Dart style guide](#)
 - [Flutter style guide](#)
- Remove blocks as much as you can.
 - The block is a block of codes like { ... }

```
if ( x ) {  
  // this is block  
}
```

- State

- `const` is built on build time.

Anatomy of StatelessWidget

```
class MyWidget extends StatelessWidget {  
  build(_) => Container();  
}
```

Anatomy of StatefulWidget

```
class MyWidget extends StatefulWidget {  
  createState() => __();  
}  
class __ extends State {  
  build(_) => Container();  
}
```

First app - Test drive

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

Learn how to create a flutter app and test run.
Learn hot reload.

Learning Dart Language

<https://dart.dev/language>
<https://dart.dev/language/variables>
<https://dart.dev/language/collections>
<https://dart.dev/language/typedefs>
<https://dart.dev/language/functions>
<https://dart.dev/language/control-flow>
<https://dart.dev/language/error-handling>
<https://dart.dev/language/classes>
<https://dart.dev/language/constructors>
<https://dart.dev/language/methods>
<https://dart.dev/language/extend>
<https://dart.dev/language/extension-methods>
<https://dart.dev/language/enum>
<https://dart.dev/language/callable-classes>
<https://dart.dev/language/async>
<https://dart.dev/language/concurrency>
<https://dart.dev/null-safety>
<https://dart.dev/codelabs/async-await>
<https://dart.dev/tutorials/languagestreams>

Write your first Flutter app

<https://github.com/withcenterdev3/your-first-app>

The video

[Follow This YouTube \(Very Good\)](#)

The Codelab

<https://codelabs.developers.google.com/codelabs/flutter-codelab-first#2>

Flutter Basic

- [Introduction to widgets](#)
- [Building layouts tutorial](#)
- [Add interactivity tutorial](#)

UI - Animation, Design, Effects, FORM

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

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

<https://docs.flutter.dev/cookbook/effects/expandable-fab>

<https://docs.flutter.dev/cookbook/effects/gradient-bubbles>

- [Build a form with validation](#)
- [Create and style a text field](#)
- [Focus and text fields](#)
- [Handle changes to a text field](#)
- [Retrieve the value of a text field](#)

- [Add Material touch ripples](#)
- [Handle taps](#)
- [Implement swipe to dismiss](#)

- [Display images from the internet](#)
- [Fade in images with a placeholder](#)
- [Work with cached images](#)

- Create a grid list
- Create a horizontal list
- Create lists with different types of items
- Place a floating app bar above a list
- Use lists
- Work with long lists

- Report errors to a service
 -
-
- Fetch data from the internet
 - Make authenticated requests
 - Parse JSON in the background
 - Send data to the internet
 - Update data over the internet
 - Delete data on the internet
-
- Animate a widget across screens
 - Navigate to a new screen and back
 - Navigate with named routes
 - Pass arguments to a named route
 - Set up app links for Android
 - Set up universal links for iOS
 - Return data from a screen
 - Send data to a new screen

<https://docs.flutter.dev/cookbook/animation/page-route-animation>

<https://docs.flutter.dev/cookbook/animation/animated-container>

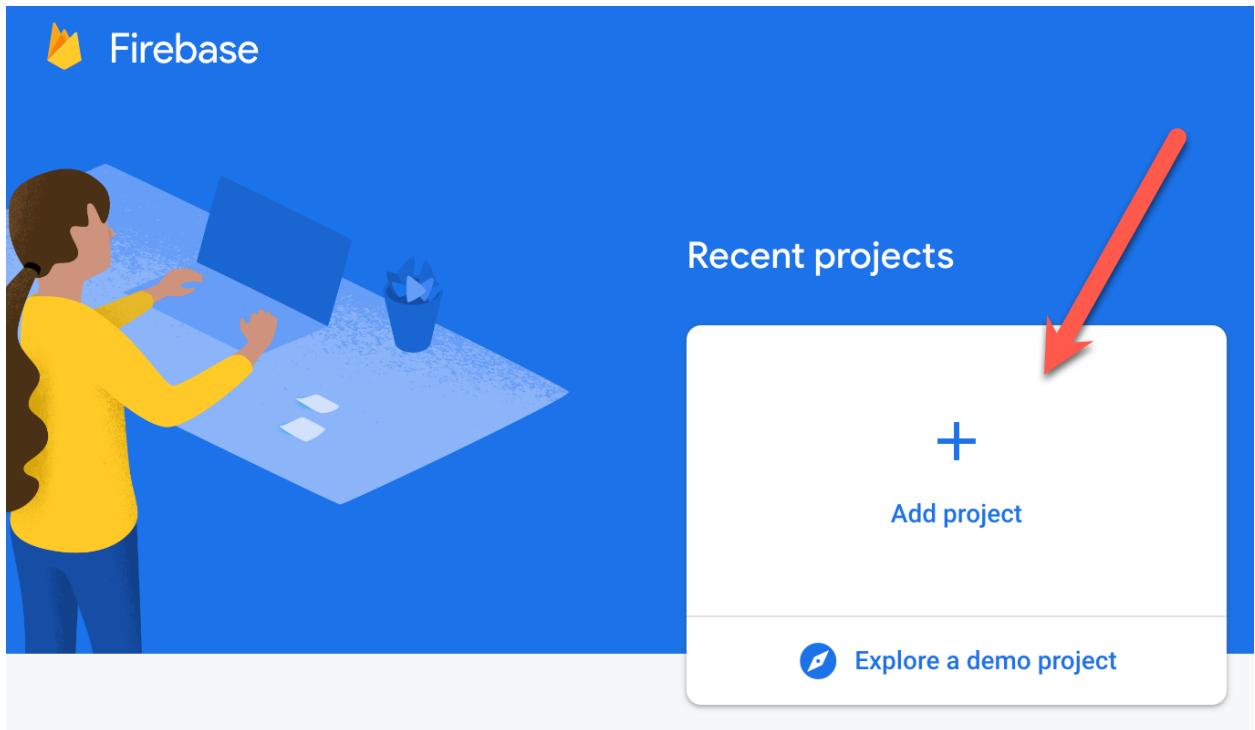
<https://docs.flutter.dev/cookbook/animation/opacity-animation>

Firebase

Firebase Installation

See <https://firebase.google.com/docs/flutter/setup?platform=ios> and follow.

- Create a new Firebase Project (or you can use existing one)



Input the project name.

- × Create a project (Step 1 of 3)

Let's start with a name for
your project[?]

Project name

withcenter-study

 withcenter-study

Continue

No need to enable statistics for learning (or testing)

Create a project (Step 2 of 2)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

- ✗ A/B testing ?
- ✗ Crash-free users ?
- ✗ User segmentation & targeting across ?
Firebase products
- ✗ Event-based Cloud Functions triggers ?
- ✗ Free unlimited reporting ?

Enable Google Analytics for this project
Recommended



[Previous](#)

[Create project](#)

Done.



withcenter-study



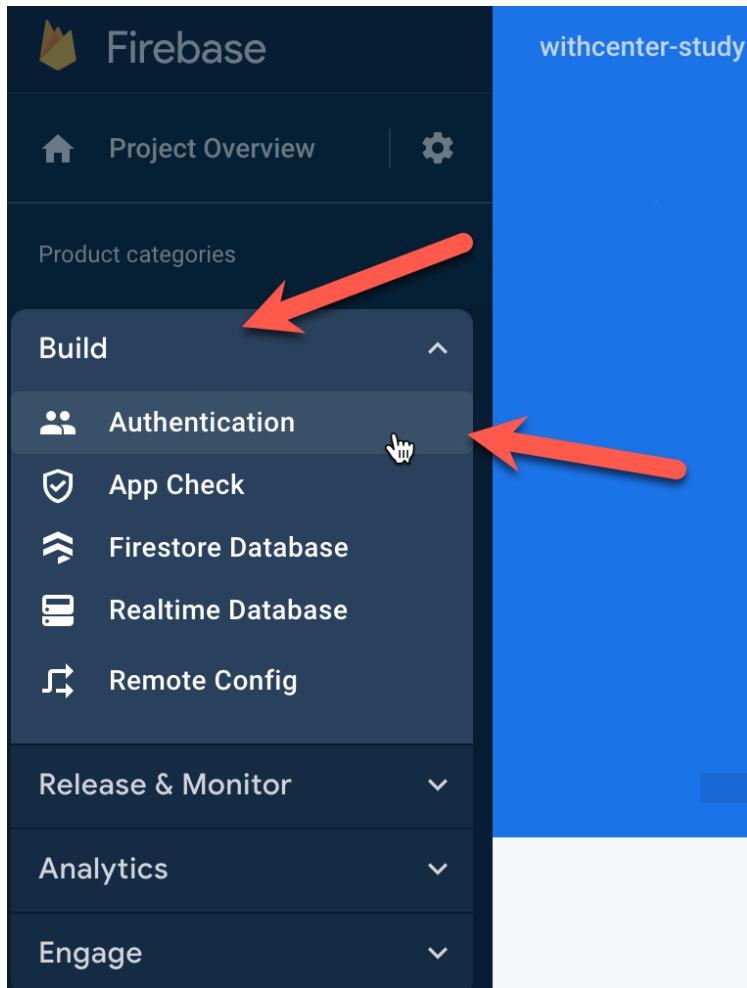
Your new project is ready

Continue

Now you have created your Firebase project.
You can use it as your Database or backend of your application.

Enable User Authentication. So, your app(or your users) can create accounts and login.

Open Build → Authentication page



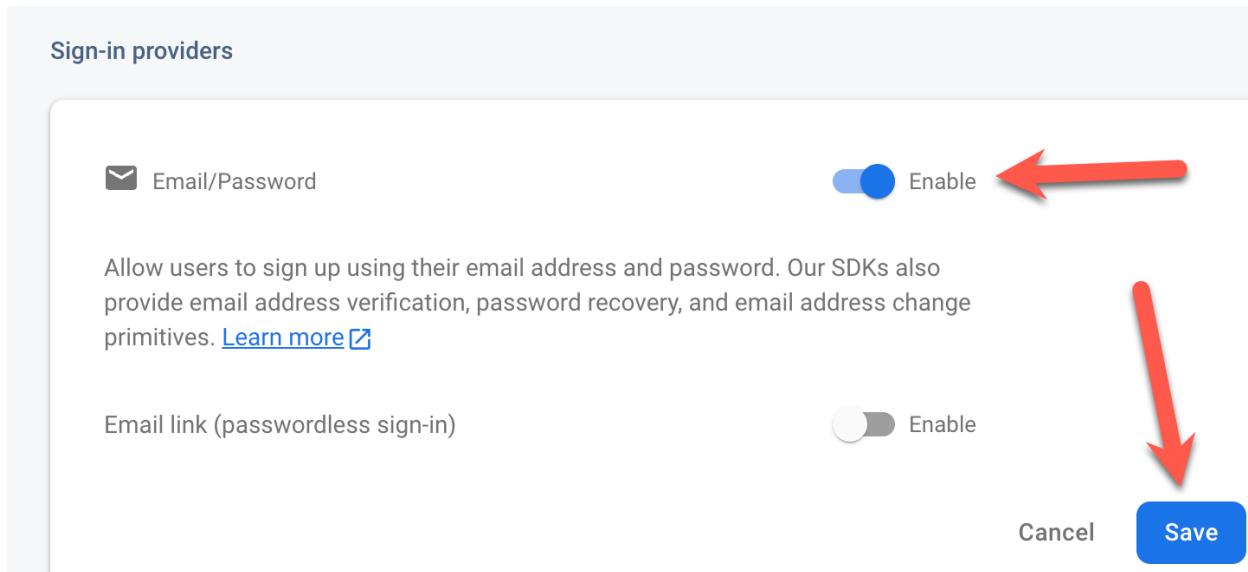
Then, press [Get started]

The screenshot shows the Firebase Project Overview page for a project named "withcenter-study". The left sidebar contains links for Project Overview, Authentication (which is highlighted in blue), Product categories, Build, Release & Monitor, and Analytics. The main content area is titled "Authentication" with the sub-instruction "Authenticate and manage users variety of providers without ser code". A large orange arrow points from the "Get started" button towards the "Email/Password" sign-in method option in the next screenshot.

Then, click [Email/Password] in Build → Authentication → Sign-in method.

The screenshot shows the Authentication settings page for the same project. The top navigation bar includes Project Overview, Authentication (highlighted in blue), Users, Sign-in method (which is underlined in blue, indicating it is selected), Templates, Usage, Settings, and Extensions. Below this, the "Sign-in providers" section is visible. A red arrow points to the "Email/Password" option under the "Native providers" heading.

And enable [Email/Password] option. Then [Save]



So far, you have created your own Firebase project and enabled the Authentication feature. So, your users can register and login with their emails.

Create Flutter App

Create your flutter app like below. The project name is **ff**.

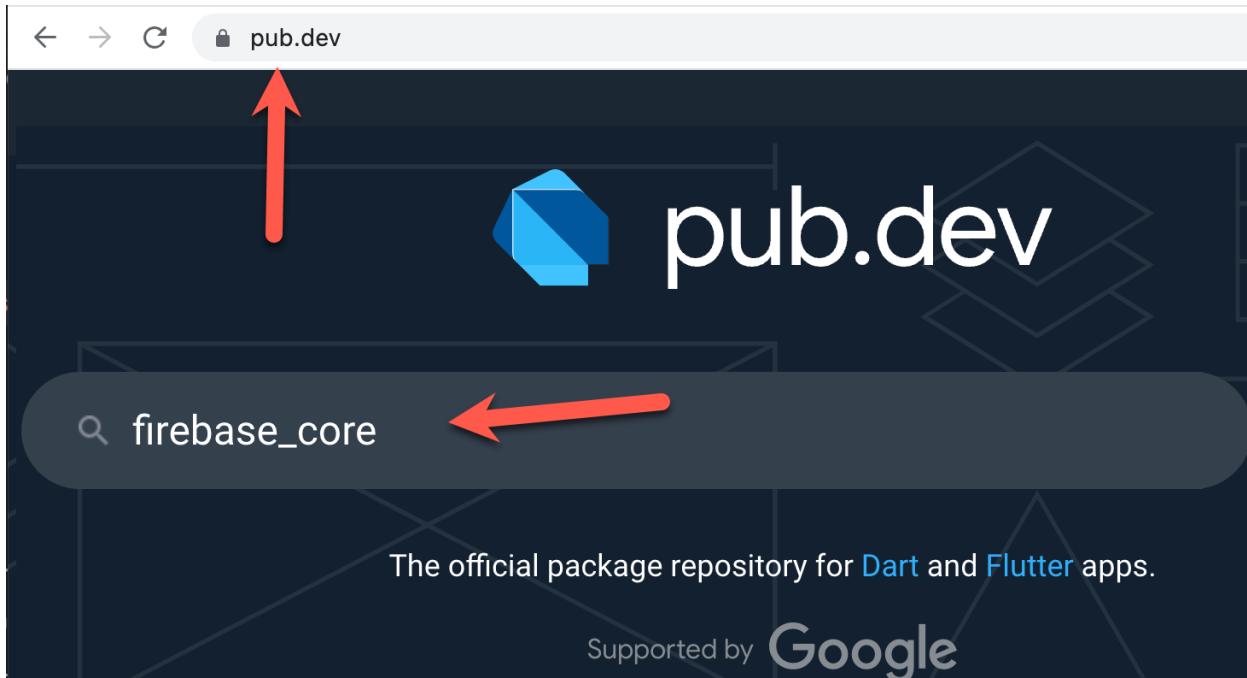
```
% flutter create ff  
% cd ff
```

Do the Firebase setup to your Flutter project. So, your app can use(connect) to Firebase.

```
% firebase login  
% dart pub global activate flutterfire_cli  
% flutterfire configure  
// → Then, select your firebase project.  
// → update *build.gradle files.
```

Install Firebase Core Library(plugin)

Open <https://pub.dev> and search for `firebase_core`.



Open the `firebase_core` package page.

A screenshot of the firebase_core package page on pub.dev. At the top, there is a search bar with a magnifying glass icon containing the text "firebase_core". Below the search bar, the text "RESULTS 142 packages" is displayed. A red arrow points from the bottom left towards the package name "firebase_core" in the results list. The package details are shown below:

firebase_core

Flutter plugin for Firebase Core, enabling connecting to multiple Firebase apps.

v 2.11.0 (40 hours ago) [firebase.google.com](https://github.com/firebase/firebase.goog...) BSD-3-Clause

SDK FLUTTER PLATFORM ANDROID IOS MACOS WEB WINDOWS

Copy the latest version.

pub.dev

firebase_core 2.11.0

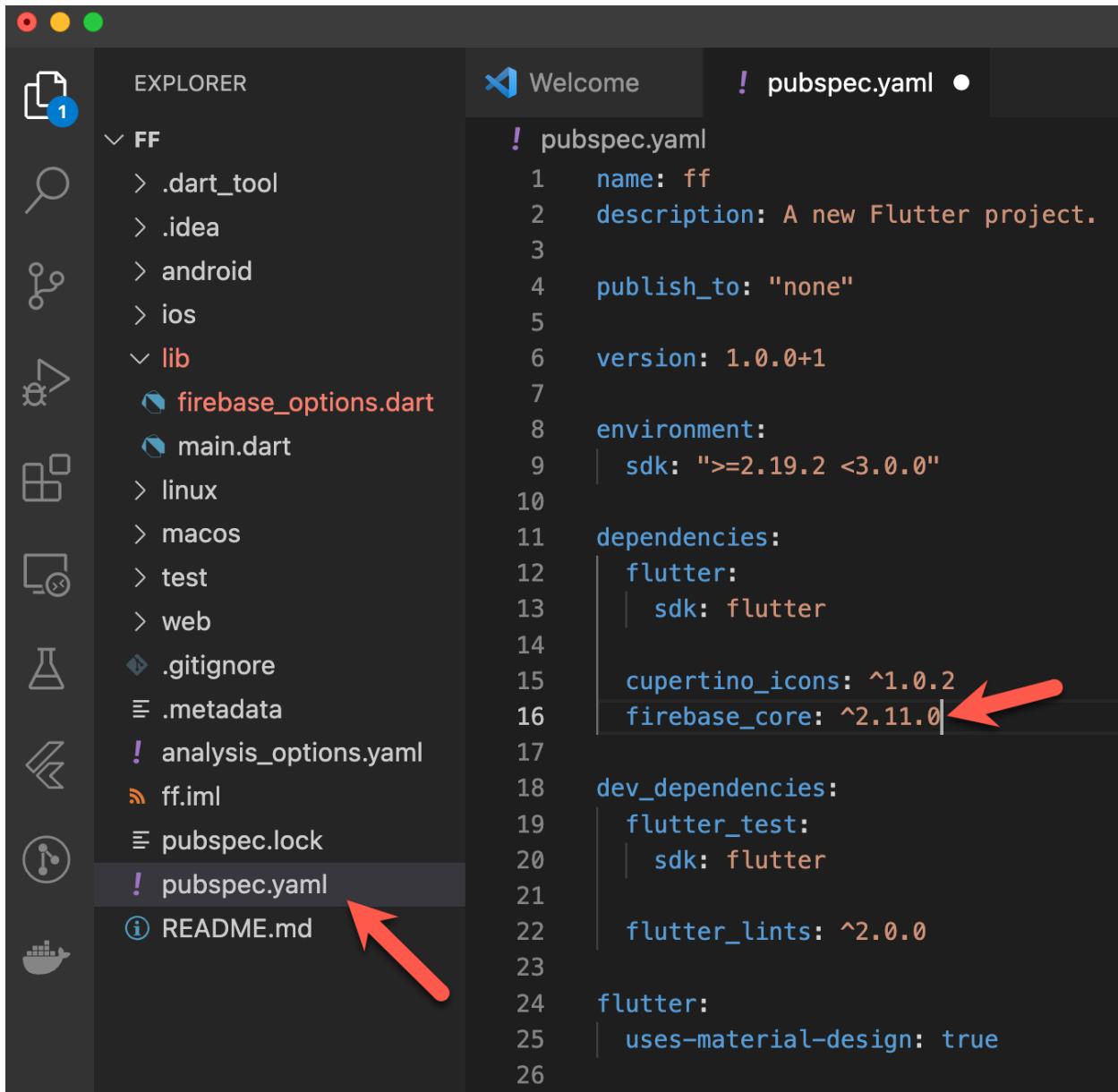
Published 40 hours ago • firebase.google.com Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS MACOS WEB WINDOWS

Readme Changelog Example Installing Versions Scores

Open pubspec.yaml file in VSCode.

And paste the latest version into the dependencies.



```
! pubspec.yaml
1   name: ff
2   description: A new Flutter project.
3
4   publish_to: "none"
5
6   version: 1.0.0+1
7
8   environment:
9     | sdk: ">=2.19.2 <3.0.0"
10
11 dependencies:
12   flutter:
13     | sdk: flutter
14
15   cupertino_icons: ^1.0.2
16   firebase_core: ^2.11.0
17
18 dev_dependencies:
19   flutter_test:
20     | sdk: flutter
21
22   flutter_lints: ^2.0.0
23
24 flutter:
25   | uses-material-design: true
```

- Add Firebase Auth package to code email registration and login. So, your users can actually sign up and sign in.

Install `firebase_auth` just as exactly the same way of installing `firebase_core` package.

So far, you have

- Created your firebase project.
- Enabled Authentication in Firebase project. So your users can register/login by their emails.
- Create a Flutter app.
 - And setup your flutter app with Firebase.
- And installed Firebase Core plugin in your Flutter app to connect and use Firebase.
- Then, lastly, you have installed the Firebase Authentication package to build user registration and login.

Firebase Authentication

- Code for firebase connection. So, when your app is running, your app can actually use the Firebase.

With the code below, your app is now using Firebase.

lib/main.dart)

```
import 'package:ff/firebase_options.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp());
}
```

Add registration code

With the code below, your user(or you) can register into Firebase.

lib/main.dart)

```
import 'package:ff/firebase_options.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

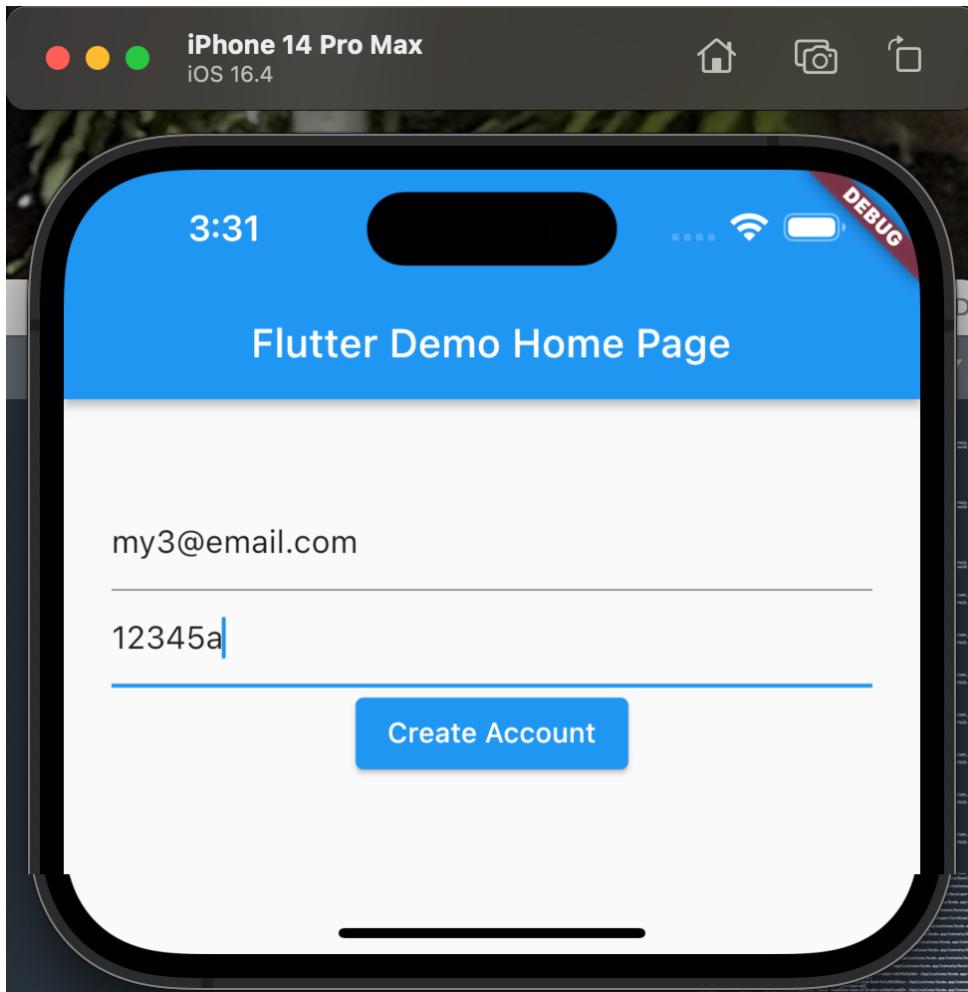
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
```

```
const MyHomePage({super.key, required this.title});  
  
final String title;  
  
@override  
State<MyHomePage> createState() => _MyHomePageState();  
}  
  
class _MyHomePageState extends State<MyHomePage> {  
final email = TextEditingController();  
final password = TextEditingController();  
  
@override  
Widget build(BuildContext context) {  
return Scaffold(  
appBar: AppBar(  
title: Text(widget.title),  
) ,  
body: Center(  
child: Padding(  
padding: const EdgeInsets.all(24),  
child: Column(  
mainAxisAlignment: MainAxisAlignment.center,  
children: <Widget>[  
TextField(  
controller: email,  
decoration: const InputDecoration(hintText: 'Input Email'),  
) ,  
TextField(  
controller: password,  
decoration: const InputDecoration(hintText: 'Input Password'),  
) ,  
ElevatedButton(  
onPressed: () async {
```

```
try {
    final credential = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(
            email: email.text,
            password: password.text,
        );
    print(credential.user);
} on FirebaseAuthException catch (e) {
    if (e.code == 'weak-password') {
        print('The password provided is too weak.');
    } else if (e.code == 'email-already-in-use') {
        print('The account already exists for that email.');
    }
} catch (e) {
    print(e);
}
},
child: const Text('Create Account'))
],
),
),
),
),
);
}
}
```

You will see a screen like below with the code above.



Once you register, you will see your account in the Firebase Authentication page.

The screenshot shows the Firebase console's Authentication section. On the left, there's a sidebar with Project Overview, Authentication (which is selected and highlighted in blue), and other tools like Build, Release & Monitor, and Analytics. The main area is titled 'Authentication' and has tabs for Users, Sign-in method, Templates, Usage, Settings, and Extensions. The 'Users' tab is active. Below it is a search bar with the placeholder 'Search by email address, phone number, or user UID'. A table lists users with columns for Identifier, Providers, Created, Signed In, and User UID. One user, 'my3@email.com', is listed with a provider icon (envelope) and a timestamp of May 6, 2023.

Identifier	Providers	Created	Signed In	User UID
my3@email.com	✉	May 6, 2023	May 6, 2023	dS7s9IMdFeZeS2lmLj

- Let's display user email when user register or login (instead of the registration form).

The code below shows user email information when user registered or logged in (instead of register or login form.)

```
import 'package:ff/firebase_options.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
```

```
return MaterialApp(
  title: 'Flutter Demo',
  theme: ThemeData(
    primarySwatch: Colors.blue,
  ),
  home: const MyHomePage(title: 'Flutter Demo Home Page'),
);
}

class MyHomePage extends StatefulWidget {
const MyHomePage({super.key, required this.title});

final String title;

@Override
State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
final email = TextEditingController();
final password = TextEditingController();

User? user;

@Override
void initState() {
  super.initState();

  FirebaseAuth.instance.authStateChanges().listen((user) {
    if (user == null) {
      print('User is currently signed out!');
    } else {
      print('User is signed in!');
    }
    setState(() {
      this.user = user;
    });
  });
}
}
```

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text(widget.title),
        ),
        body: Center(
            child: Padding(
                padding: const EdgeInsets.all(24),
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                        if (user == null) ...[
                            TextField(
                                controller: email,
                                decoration: const InputDecoration(hintText: 'Input Email'),
                            ),
                            TextField(
                                controller: password,
                                decoration: const InputDecoration(hintText: 'Input Password'),
                            ),
                        ],
                        Row(
                            children: [
                                ElevatedButton(
                                    onPressed: () async {
                                        try {
                                            final credential = await FirebaseAuth.instance
                                                .signInWithEmailAndPassword(
                                                    email: email.text,
                                                    password: password.text,
                                                );
                                            print(credential.user);
                                        } on FirebaseAuthException catch (e) {
                                            if (e.code == 'user-not-found') {
                                                print('No user found for that email.');
                                            } else if (e.code == 'wrong-password') {
                                                print('Wrong password provided for that user.');
                                            } else {
                                                print(e.code);
                                                print(e.message);
                                            }
                                        }
                                    }
                                )
                            ],
                        ),
                    ],
                ),
            ),
        ),
    );
}
```

```
        } catch (e) {
            print(e);
        }
    },
    child: const Text('Login'),
),
const Spacer(),
ElevatedButton(
    onPressed: () async {
        try {
            final credential = await FirebaseAuth.instance
                .createUserWithEmailAndPassword(
                    email: email.text,
                    password: password.text,
                );
            print(credential.user);
        } on FirebaseAuthException catch (e) {
            if (e.code == 'weak-password') {
                print('The password provided is too weak.');
            } else if (e.code == 'email-already-in-use') {
                print('The account already exists for that email.');
            }
        } catch (e) {
            print(e);
        }
    },
    child: const Text('Register'),
),
],
),
],
),
if (user != null) ...[
    Text('User: ${user!.email}'),
    ElevatedButton(
        onPressed: () async {
            await FirebaseAuth.instance.signOut();
        },
        child: const Text('Sign Out'),
    ),
]
],
```

```
    ) ,  
    ) ,  
    ) ,  
    ) ;  
}  
}
```

Firebase Profile Update

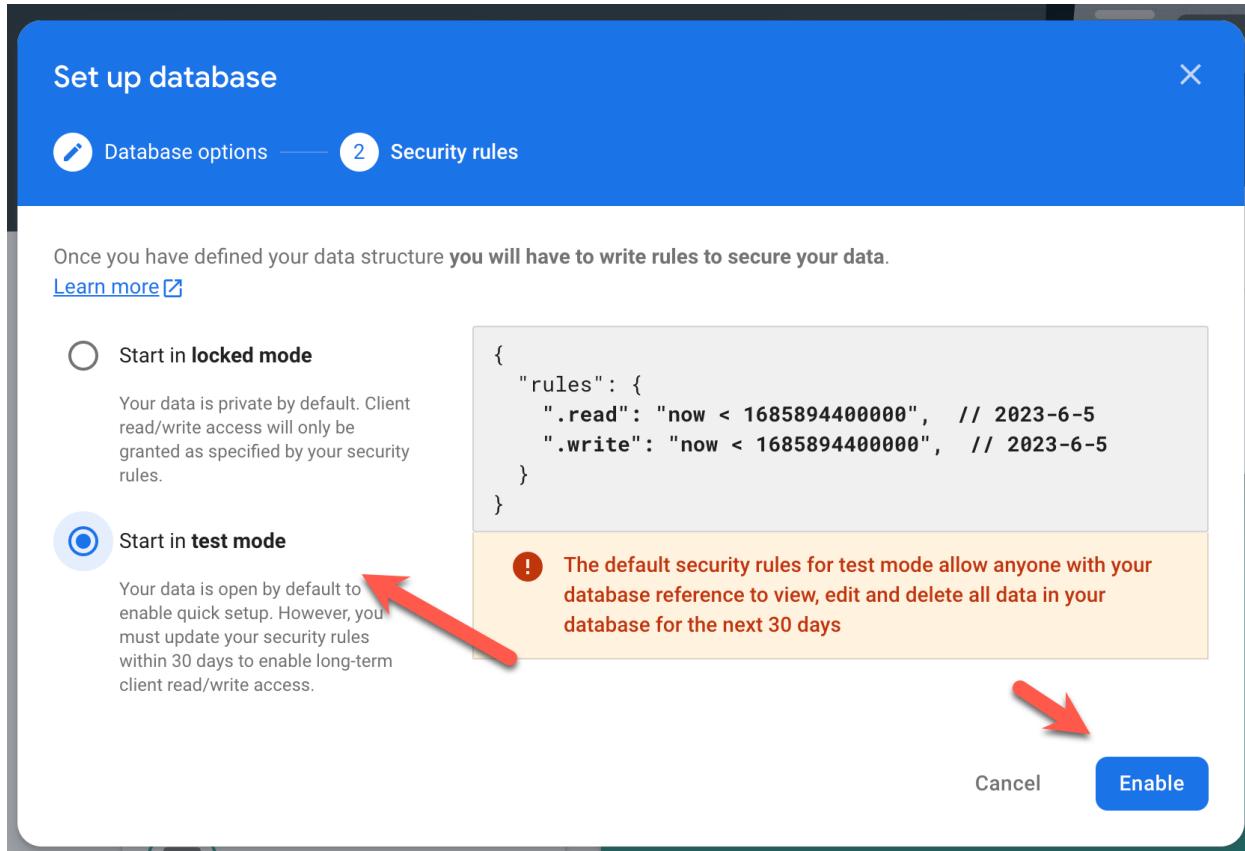
- Create a profile page. Use go_router.

Follow the [Go_Route Office document](#)

- Create the realtime database

The screenshot shows the Firebase console interface. On the left, there's a sidebar with 'Authentication' and 'Realtime Database' listed under 'Project shortcuts'. Below this is a 'Build' section with 'Authentication', 'App Check', 'Firestore Database', and 'Realtime Database' listed. 'Realtime Database' is highlighted with a blue background and has a small icon of a person with a gear. To the right, the main area is titled 'Realtime Database' with the subtitle 'Store and sync data in real time'. A large 'Create Database' button is centered. Two red arrows point from the 'Realtime Database' item in the sidebar towards the 'Create Database' button.

The screenshot shows a modal dialog titled 'Set up database'. It has two tabs at the top: 'Database options' (selected) and 'Security rules'. The main content area says 'Your location setting is where your Realtime Database data will be stored.' Below it, 'Realtime Database location' is set to 'Singapore (asia-southeast1)'. At the bottom right are 'Cancel' and 'Next' buttons, with a red arrow pointing down to the 'Next' button.



Install `firebase_database`.

After installing the `firebase_database` package. You must run `futterfire_cli` again. By doing this, it will add database information into the setup.

Update user display name into realtime database.

See the official documents. <https://firebase.google.com/docs/database/flutter/start>

See the complete working study source in github:
<https://github.com/thruthesky/study-flutter/tree.firebaseio-profile>

Day 0 - How run Flutter

% flutter run

Day 1 - StatelessWidget, StatefulWidget, Typing, Types

- Variable types in Dart

Day 2 - Layout - Constraints go down, Sizes go up, Parents set the Position **

This is the most important part **

<https://docs.flutter.dev/ui/layout>

<https://docs.flutter.dev/ui/layout/constraints>

Good understanding on how layout works.

<https://docs.google.com/document/d/1IXCs1eDw58J4hd6yfdLFDHf8S4KwWzAanwf4BKqYrF0/edit#heading=h.t2x75s299otu>

The nature of Container

This is Mr. Song's own thoughts. It is welcome to prove that this thought is wrong.

- A container with no constraints
 - will fit into the child. (as small as its child.)
 - and no child will fit into the parent's available space.

The screenshot shows a code editor with a Dart file named `home.screen.dart` open. The code defines a `_HomeScreenState` class that extends `State<HomeScreen>`. It overrides the `build` method to return a `Scaffold` widget. The `Scaffold` has an `AppBar` with the title 'Home'. Below the `AppBar` is a `bottomNavigationBar` which is a `UserReady` object. The `UserReady` object's `builder` property is set to a `MainBottomNavigationBar` widget with a user parameter. The `Scaffold`'s `body` contains a `Container` with a grey color. A tooltip is visible over the `Container` code, stating 'You, 3 seconds ago • Uncommit'. The code editor has a dark theme with syntax highlighting. To the right of the editor is a mobile application interface for an iPhone 14 Pro Max running iOS 16.4. The app's title bar says 'Home'. The bottom navigation bar has four items: 'Home' (selected), 'Profile', 'Messages', and 'Menu'. The status bar at the top of the phone screen shows the time as 10:07.

The nature of other widgets ??

This is Mr. Song's own thoughts. It is welcome to prove that this thought is wrong.

Just like a container, if a widget has no constraints, then it will act as Container

Especially some widget that cannot have constraints will shrink its size on the cross axis only.
Like Column, Row, and others.

Know the nature of the widgets

- Row, Column will shrink its cross axis size to its child size. Not the main axis.
- Align - will take as much available space from its parent.

Day 3 - How to Use Material 3 and Theme, Layout

Don't use Column, Row, Wrap for Adaptive layout.

➡ Design for every device with Flutter and Material 3

Use grid for the layout;

https://pub.dev/packages/flutter_layout_grid

See this video: ➡ Design for every device with Flutter and Material 3

01: GRID - Use grid for better RWD

```
...
return LayoutGrid(
    columnSizes: List.generate(12, (index) => 1.fr),
    rowSizes: [kToolbarHeight.px, 1.fr],
    children: const [
        GridPlacement(
            columnStart: 1,
            columnSpan: 12,
            child: Center(child: Text('Header')),
        ),
        GridPlacement(
            columnStart: 2,
            columnSpan: 10,
            child: Center(child: Text('Content')),
        ),
    ],
);
...

```

02: Color

Material 3 Color Scheme

Primary	P-40	Secondary	S-40	Tertiary	T-40	Error	E-40				
On Primary	P-100	On Secondary	S-100	On Tertiary	T-100	On Error	E-100				
Primary Container	P-90	Secondary Container	S-90	Tertiary Container	T-90	Error Container	E-90				
On Primary Container	P-10	On Secondary Container	S-10	On Tertiary Container	T-10	On Error Container	E-10				
Primary Fixed	P-90	Primary Fixed Dim	P-80	Secondary Fixed	S-90	Secondary Fixed Dim	S-80	Tertiary Fixed	T-90	Tertiary Fixed Dim	T-80
On Primary Fixed	P-10	On Secondary Fixed	S-10	On Tertiary Fixed	T-10						
On Primary Fixed Variant	S-30	On Secondary Fixed Variant	S-30	On Tertiary Fixed Variant	T-30						
Surface Dim	N-87	Surface	N-98	Surface Bright	N-98	Inverse Surface	N-20				
Surf. Container Lowest	N-100	Surf. Container Low	N-96	Surf. Container High	N-92	Surf. Container Highest	N-90	Inverse On Surface	N-95		
On Surface	N-10	On Surface Var.	NV-30	Outline	NV-50	Outline Variant	NV-80	Inverse Primary	P-80		
						Scrim	N-0	Shadow	N-0		

Day 4 - How to use global variables to use consistent UI, spaces and sizes.

xs, sm, lg, xl, 2xl, 3xl

fontXs, Sm, Lg, Xl, 2xl,
iconXs, Sm, Lg, Xl, 2xl,
spaceXs, spaceSm, Md, Lg, Xl, 2xl, 3xl

Page() widget which has a scrollable Column.

primaryColor, backgroundColor, primaryTextColor, backgroundTextColor,

Day 5 - Provider - state management

From Mr. Song - It may be wrong.

The state is the value of a variable. So, let's think about it as a **variable**.

Then, what is **state management**?

Some times (or more often in Flutter) **the variable shows(displays) a wrong value**.
And we need to manage it (to make it show correct value). This is called state management.

For instance) the code below does not show the correct value on screen.

```
class MyState extends State<StScreen> {  
    String abc = 'Sample value';
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('StScreen'),
    ),
    body: Container(
      width: double.infinity,
      padding: const EdgeInsets.all(24),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text('StScreen; variable: $abc'),
          ElevatedButton(
            onPressed: () {
              abc = 'New value';
              debugPrint('abc: $abc');
            },
            child: const Text('Change value'),
          ),
        ],
      ),
    ),
  );
}
```

So, what we need to do in this case is simply to call `setState()` which will manage the state.

```
setState(() {
  abc = 'New value';
});
```

Managing state means, to show the actual value on the screen. You can use `setState()` if the logic is simple and for the class member variable of the stateful widget.

What if you need to update a state which is across multiple widgets and screens? `setState()` may not work in this case.

And there are lots of helpful libraries for managing states. To name few,

- [Provider](#)
 - [Official Doc](#)
- Riverpod
- Getx
- Redux
- MobX
- BLoC
- Git it
- states_rebuilder
- Triple Pattern
- solidart
- flutter_reactive_widget

Day 6 - Go Router

Day 7 - HTTP, Modeling

<https://docs.flutter.dev/cookbook/networking/fetch-data>

Day 8 - Firebase

Firebase Database

How to get newly added document

Since it listens to a single (newly added) data, the snapshot has a single document information.

ex)

```
StreamBuilder<DatabaseEvent>(
    stream: ForumService.instance
        .categoryRef(widget.category)
        .orderByChild('orderNo')
        .limitToFirst(1)
        .onChildAdded,
    builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
            return CircularProgressIndicator();
        }
        if (snapshot.hasError) {
            return Text('Error: ${snapshot.error}');
        }

        final post = PostModel.fromSnapshot(snapshot.data!.snapshot);
```

Day 9 - Image & File Upload

iPhone 14 Pro Max - 1



File Upload

1. Let the user choose from which resource they will upload.
2. Get the image(or file)
3. Compress and adjust if possible(especially for images)
4. Upload it
5. Display

File Upload



Hello



Choose a media



From Photo Gallery



From Camera

Building a package

See official Doc

<https://docs.flutter.dev/packages-and-plugins/developing-packages>