

How to get started with flutter

Background

Flutter is an open source UI software developing kit created by google. It is a perfect framework for a new developer as well as for the more experienced. With flutter you can create applications for platforms like Android, iOS, Linux, Mac, Windows and web browsers.

Now, let's get started.

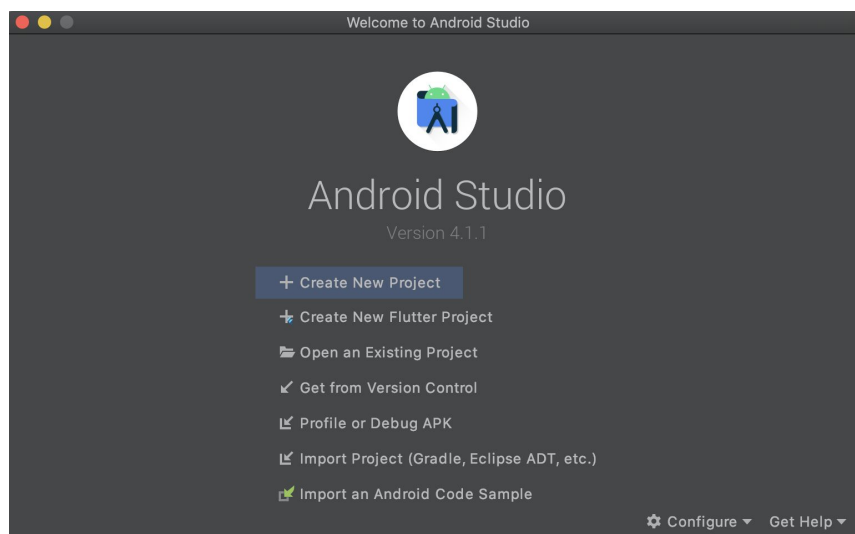
Step 1: Installation

Various editors and developing tools can be used when creating your flutter app. In this guide we are going to use Android studio, follow the [link](#) to install it if you don't already have it installed.

Once installed we want to add a Flutter plugin. Start Android studio and open plugin preferences (Preferences > Plugins on macOS, File > Settings > Plugins on Windows & Linux).

1. Select Browse repositories, select the Flutter plugin and click Install.
2. Click Yes when prompted to install the Dart plugin.
3. Click Restart when it's installed.

Now when Android studio starts it should look something like this.



Step 2: Create your Flutter app

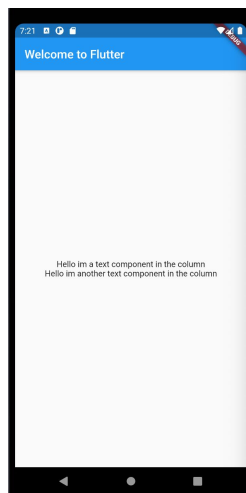
Click on *Create New Flutter Project* to create your first flutter app. This creates all the files you need to get started. Most of the implementation will be done in the file `./lib/main.dart`. For easier guidance remove all the code in `main.dart` and replace it with:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to your first flutter application',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Welcome to Flutter'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text("Hello im a text component in the column"),
              Text("Hello im another text component in the column"),
            ],
          ),
        ),
      ),
    );
  }
}
```

Now it's time to run your app to see so everything works. Open the tab tools > AVD Manager and add a virtual device. Now you can choose your virtual device and run `main.dart`. It should result in something like this depending on your device.



Step 3: Explore flutter Widget

In this step we will explore the layout. If we look at the code in our lib/main.dart the class MyApp extends a StatelessWidget. A StatelessWidget has final properties meaning that no values can change. Flutter also provides a StatefulWidget where the properties can change during its lifetime, let's explore that.

For a quick Stateful boilerplate code place your cursor at the end of all the code and type stfull. The editor will ask if you would like to add a Stateful widget, follow the editor instructions to add the boilerplate. Give your stateful class a name, in this example we will call it MyFirstPage. Add a child in the container of your new class and add a Text(). Now add your new class to MyApp and remove both of the Text components like in the code below:

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return
      MaterialApp(
        title: 'Welcome to your first flutter application',
        home: Scaffold(
          appBar: AppBar(
            title: Text('Welcome to Flutter'),
          ),
          body: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                MyFirstPage(),
                Text("Hello im a text component in the column"),
                Text("Hello im another text component in the column"),
              ],
            ),
          ),
        ),
      );
  }
}

class MyFirstPage extends StatefulWidget {
  @override
  _MyFirstPageState createState() => _MyFirstPageState();
}

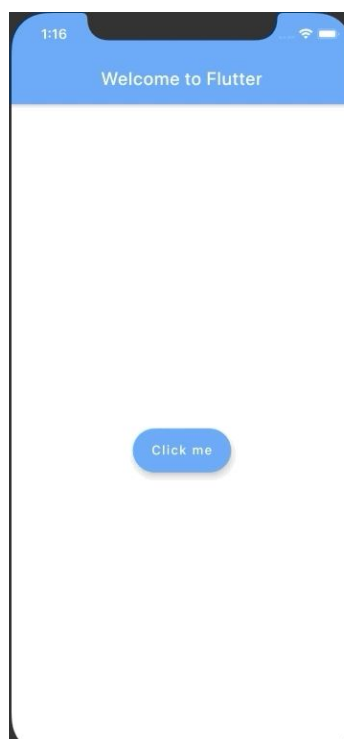
class _MyFirstPage extends State<MyFirstPage> {
  @override
  Widget build(BuildContext context) {
    return Container(
      child: Text("Hello"),
    );
  }
}
```

Now when you run the app you should only see the text “Hello”. Let's keep on creating. In the next step we will add a button element that shows a text when pressed.

Step 4: Interaction and Callback-functions

For making your app more interactable it can be of good use to know how to handle functions and callback functions. We will implement a simple button that when pressed calls the `setState` function which will set a string state and show the text in the app. Look at the code snippet below for inspiration on how it can be done.

```
class _MyFirstPageState extends State<MyFirstPage> {  
  String myTitle = "";  
  
  @override  
  Widget build(BuildContext context) {  
    return Column(  
      mainAxisAlignment: MainAxisAlignment.spaceBetween,  
      children: <Widget>[  
        Text(myTitle),  
        Padding(padding: EdgeInsets.all(15)),  
        FloatingActionButton.extended(  
          onPressed: () {setState(() {  
            myTitle = "You pressed the button";  
          });},  
          label: Text("Click me")  
        ),  
      ],  
    );  
  }  
}
```



Step 5: Navigation

As your app is growing it might be time to create more than one single page. Say that instead of showing a text when you click the button you want to get navigated to the next page. For navigation between pages Flutter has a widget called `Navigation`. With the method `.push` you can add a `Route` to the stack of routes managed by the `Navigation` widget. With the help of the method `.pop` you can in the same way remove a `Route` from the stack.

In the example below we implement a new class called `NextPage` and replace the old code in the `onPressed` function with a `Navigation.push` which routes us to the class `NextPage`. To navigate back we add a `.pop` to the `onPressed` function in our class `NextPage`.

```
class _MyFirstPageState extends State<MyFirstPage> {
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: <Widget>[
        FloatingActionButton.extended(
          onPressed: ()
            {Navigator.push(
              context, MaterialPageRoute(builder: (context) => NextPage()),
            );},
          label: Text("Navigate to next page")
        ),
      ],
    );
  }
}

class _NextPageState extends State<NextPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Next page'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Column(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: <Widget>[
                FloatingActionButton.extended(
                  onPressed: ()
                    {
                      Navigator.pop(context);
                    },
                  label: Text("Go back"),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

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



We hope this helped you to get started with Flutter and building your first Flutter app.