



Your first experiences with Flutter

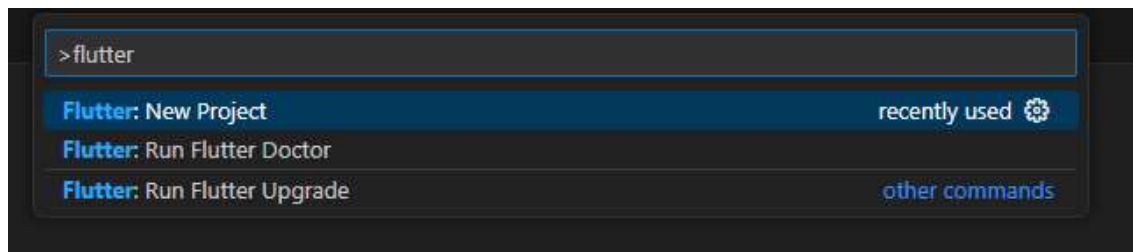
- Create a HelloWorld app in Flutter
- Run the app on Chrome and on Android emulator
- Change color and font of text widgets
- Observe your code changes directly with “Hot Reload” (no Rebuild needed)
- Learn about axis alignments of rows and columns
- Use buttons and images in your UI



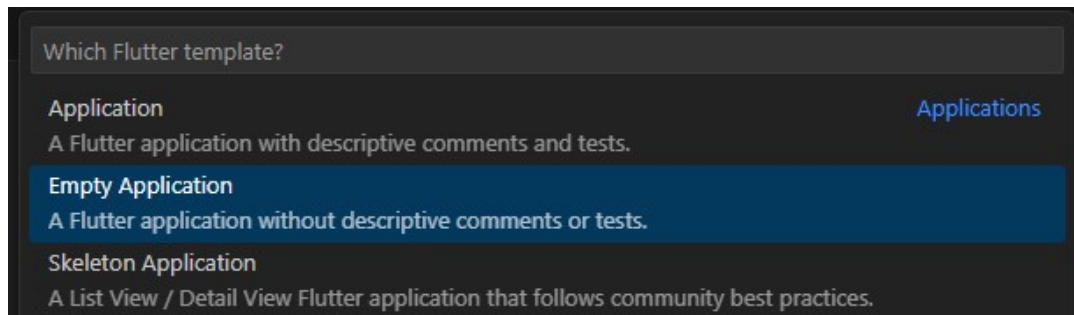
Create a HelloWorld Flutter app in VS Code (part 1)

Open VS Code and select menu “View / Command Palette ...” (or simply press F1).

In the search field enter “flutter” and select “flutter: New Project”



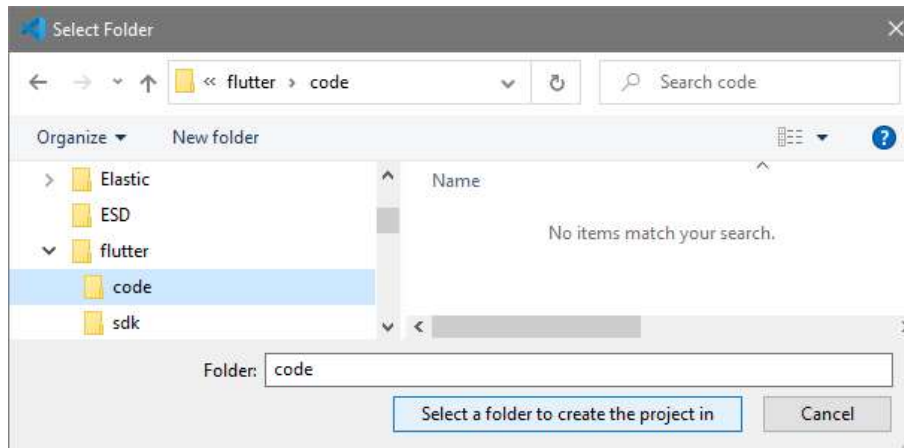
In the next drop-down, select “Empty Application”:



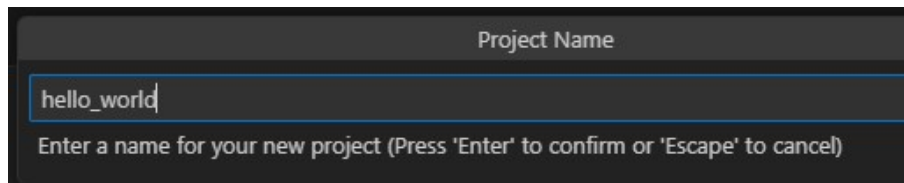


Create a HelloWorld Flutter app in VS Code (part 2)

Select the folder where the new project should be created, e.g. “C:\flutter\code”



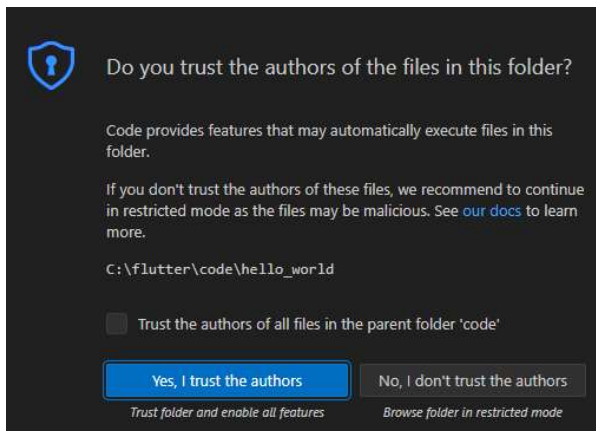
Enter the project name (no blanks or capital letters are allowed):



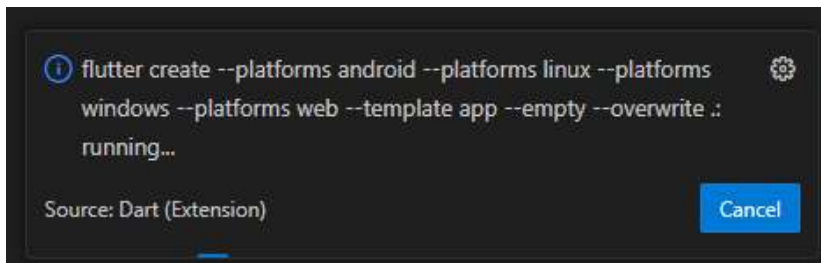


Create a HelloWorld Flutter app in VS Code (part 3)

Allow VS Code to open the new created folder:



Wait until the project is created (you need an Internet connection during this step):





Your first created Flutter app in VS Code

The screenshot shows the Visual Studio Code interface with a Flutter project named 'hello_world'. The Explorer sidebar on the left displays the project structure, including the 'lib' directory and the 'main.dart' file. The main editor window shows the content of 'main.dart', which is a Dart file for a Flutter app. The code includes an import statement for 'package:flutter/material.dart', a 'main' function that calls 'runApp', and a 'MainApp' class that extends 'StatelessWidget'. The 'build' method of 'MainApp' returns a 'MaterialApp' widget with a 'Scaffold' containing a 'Center' widget with a 'Text' widget displaying 'Hello World!'.

```
lib > main.dart > ...
1  import 'package:flutter/material.dart';
2
3  Run | Debug | Profile
4  void main() {
5    runApp(const MainApp());
6  }
7
8  class MainApp extends StatelessWidget {
9    const MainApp({super.key});
10
11  @override
12  Widget build(BuildContext context) {
13    return const MaterialApp(
14      home: Scaffold(
15        body: Center(
16          child: Text('Hello World!'),
17        ), // Center
18      ), // Scaffold
19    ); // MaterialApp
20  }
21
```

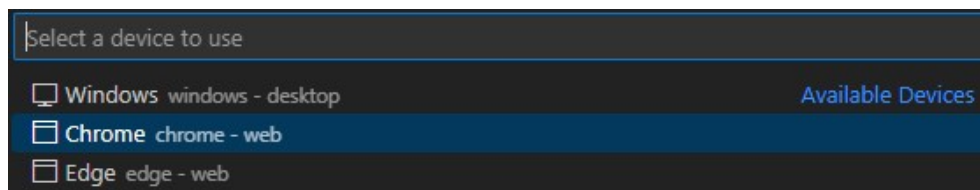


Test your app on Chrome or Edge (part 1)

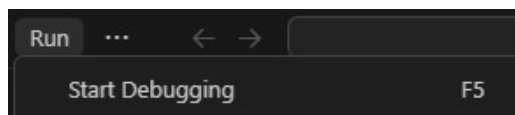
In the bottom line of VS Code, tap the red marked area:



Select Chrome or Edge:



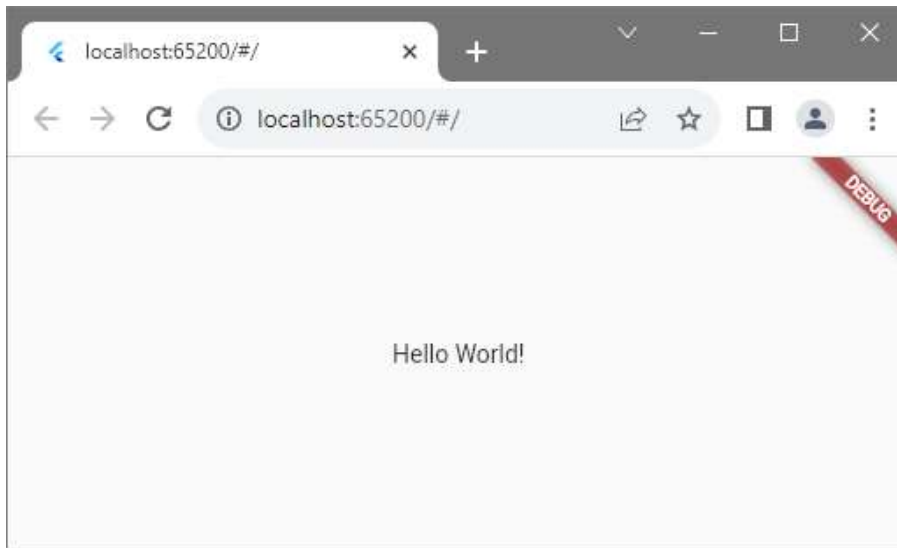
Select menu “Run / Start Debugging” or press F5:





Test your app on Chrome or Edge (part 2)

After some seconds, a Chrome or Edge window should come up showing your app:



and in VS Code you see a “Debug Console”:

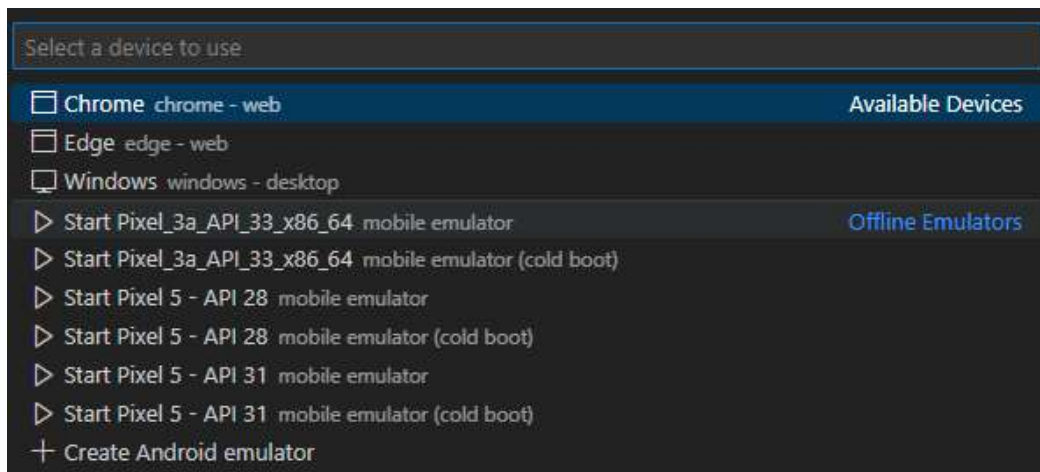
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Launching lib\main.dart on Chrome in debug mode...
This app is linked to the debug service: ws://127.0.0.1:65247/D3R7DtAdEmM=/ws
Debug service listening on ws://127.0.0.1:65247/D3R7DtAdEmM=/ws
Connecting to VM Service at ws://127.0.0.1:65247/D3R7DtAdEmM=/ws
```




Test your app on Pixel Emulator (part 1)

In case your PC has an Intel CPU supporting VT-x, you can select a Pixel emulator for tests:



It takes some time to start:

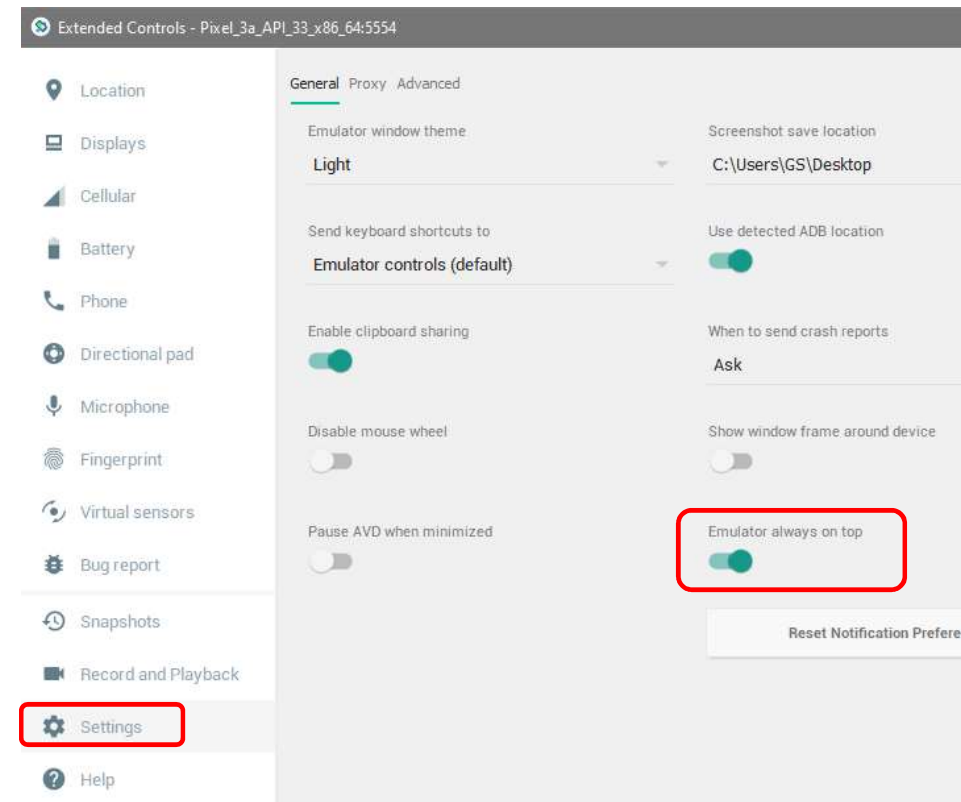
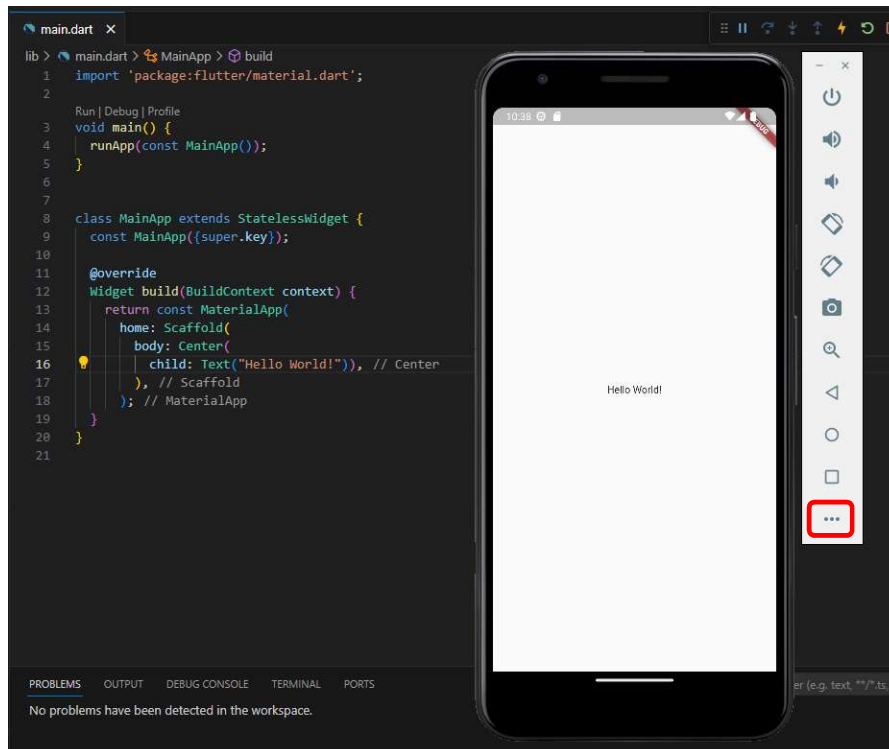
 Waiting for Pixel_3a_API_33_x86_64 to connect...

Then press F5 to start debugging. Take care: first build may take more than a minute.



Test your app on Pixel Emulator (part 2)

Emulator appears on top of VS Code and stays on-top with setting:



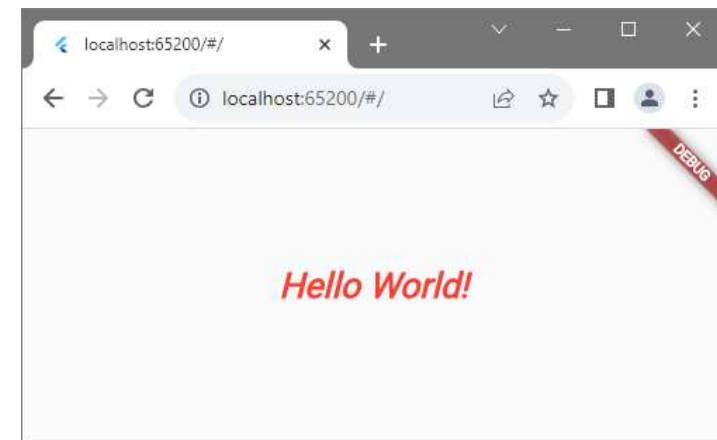


Change some Text properties

Define a style for your Text widget:

```
@override
Widget build(BuildContext context) {
  return const MaterialApp(
    home: Scaffold(
      body: Center(
        child: Text('Hello World!',
          style: TextStyle(
            color: Colors.red,
            fontSize: 25,
            fontStyle: FontStyle.italic,
            fontWeight: FontWeight.bold), // TextStyle // Text
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

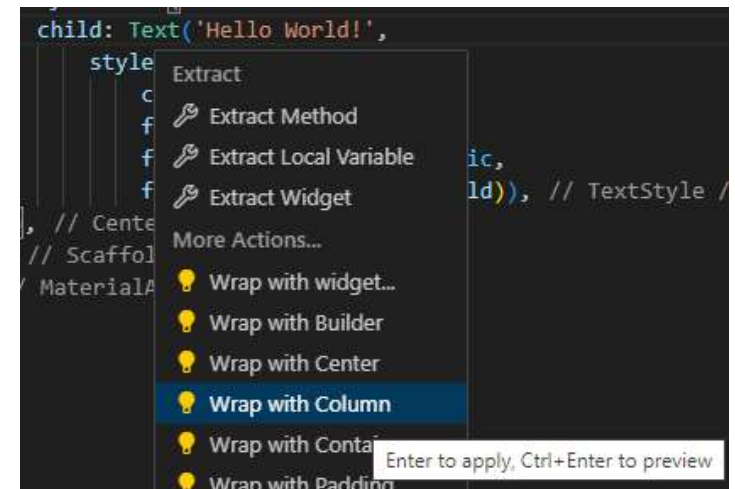
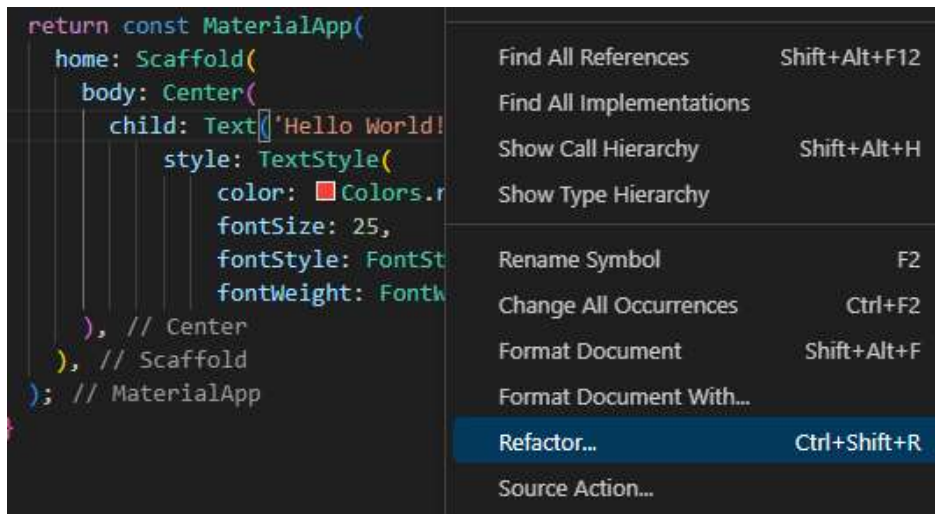
After saving your code, a “Hot Reload” is performed automatically and you can see the changes in Chrome:





Allow more widgets by introducing a Column

Right-Click on your Text widget and select “Refactor”, then select “Wrap with Column”:



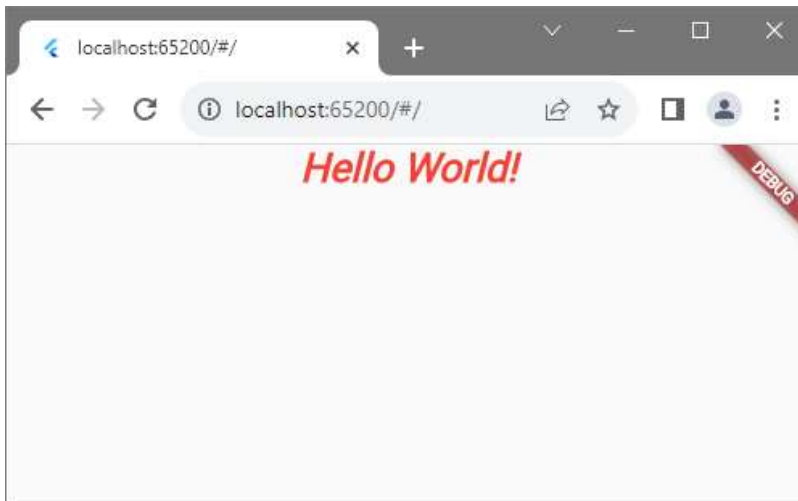
This will add a “Column” widget around your Text. Column widgets can have several children:





Allow more widgets by introducing a Column

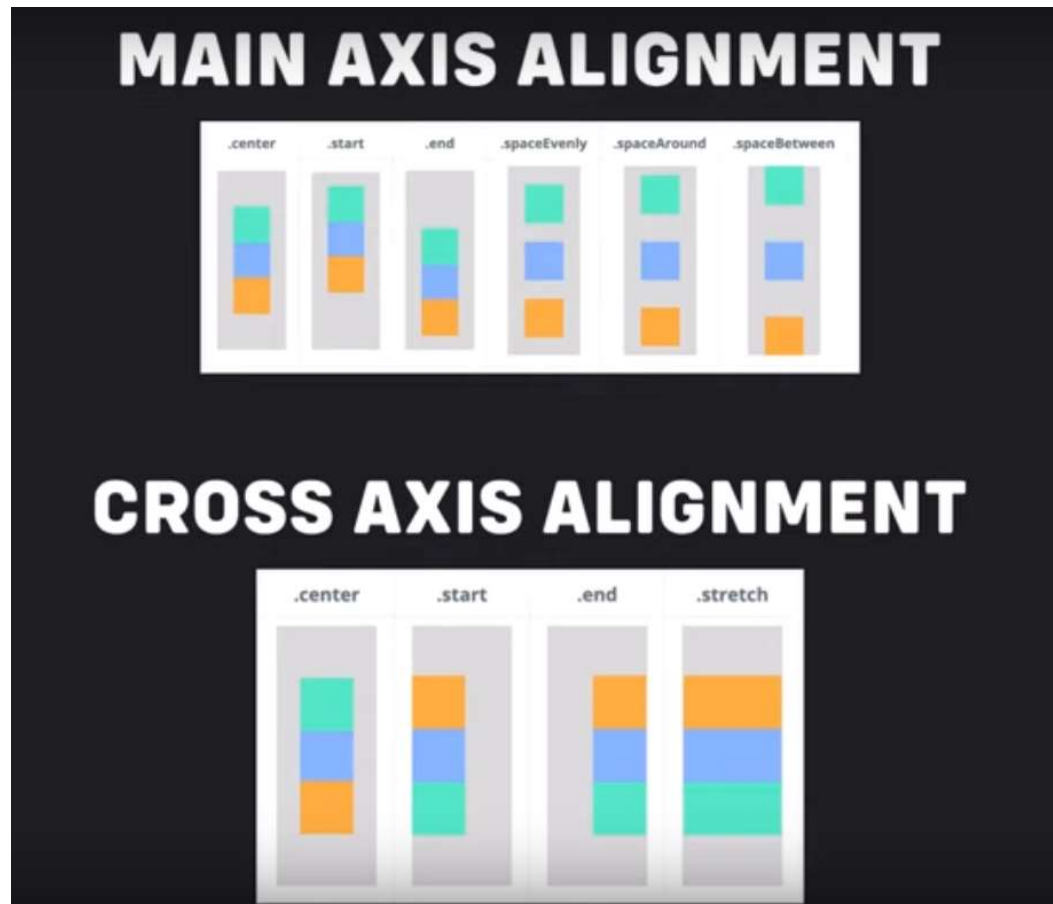
After saving your code, the “Hello World” text will move up, because Columns take the whole space and put their children by default on the top of the column:



You can change this by setting the `MainAxisAlignment` property of the Column:



Axis alignments of a Column

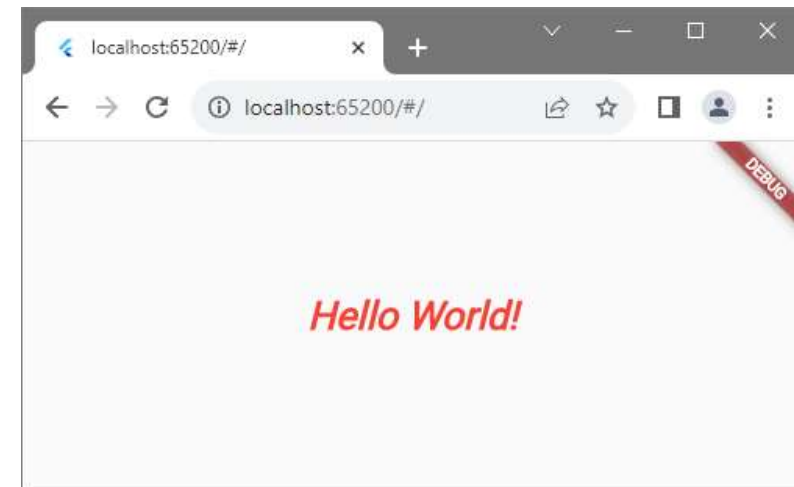


Picture taken from <https://www.youtube.com/watch?v=D4nhaszNW4o>



Center the text again with MainAxisAlignment

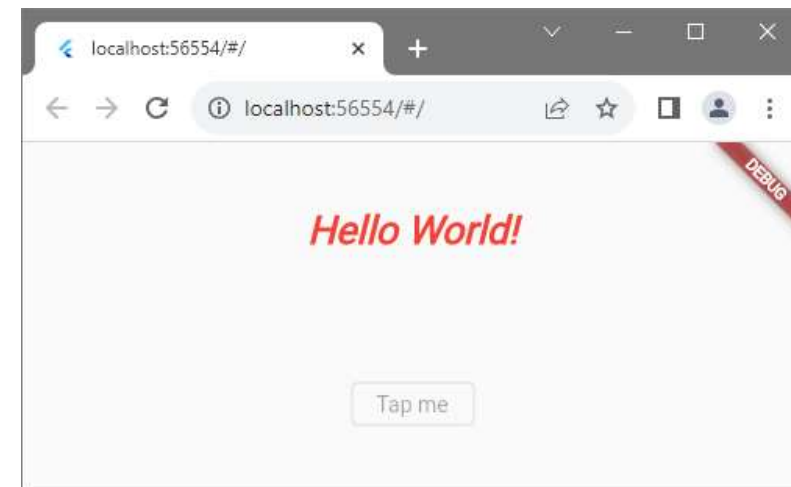
```
Widget build(BuildContext context) {  
  return const MaterialApp(  
    home: Scaffold(  
      body: Center(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.center,  
          children: [  
            Text('Hello World!',  
              style: TextStyle(  
                color: Colors.red,  
                fontSize: 25,  
                fontStyle: FontStyle.italic,  
                fontWeight: FontWeight.bold)), // TextStyle // Text  
          ],  
        ), // Column  
      ), // Center  
    ), // Scaffold  
  ); // MaterialApp  
}
```





Add an OutlinedButton to the UI

```
child: Column(  
  mainAxisAlignment: MainAxisAlignment.spaceAround,  
  children: [  
    Text('Hello World!',  
      style: TextStyle(  
        color: Colors.red,  
        fontSize: 25,  
        fontStyle: FontStyle.italic,  
        fontWeight: FontWeight.bold)), // TextStyle // Text  
    OutlinedButton(onPressed: null, child: Text("Tap me"))  
  ],  
)
```



The button is disabled as long as “onPressed” is null.



Define an “onPressed” handler

Either with a new function (can be inside or outside the class, normally inside):

```
OutlinedButton(onPressed: handlePressed, child: Text("Tap me"))
```

```
void handlePressed() {  
  print ("in handlePressed");  
}
```

Or use an anonymous function:

```
OutlinedButton(  
  onPressed: () {  
    print("OutlinedButton was pressed");  
  },  
  child: Text("Tap me")) // OutlinedButton
```

