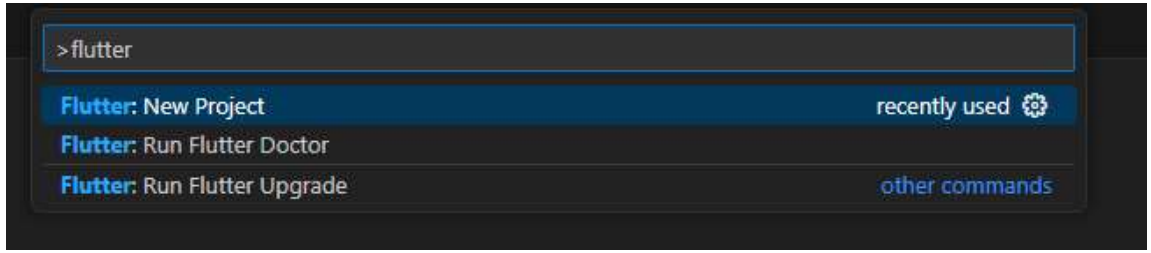
Your first experiences with Flutter

- Create a HelloWorld app in Flutter
- Run the app on Chrome and on Android emulator
- Change color and font and other properties of text widgets
- Observe your code changes directly with “Hot Reload” (no Rebuild needed)
- Learn about columns and rows and their axis alignments
- Know that Flutter offers different kinds of buttons
- Be able to style a button

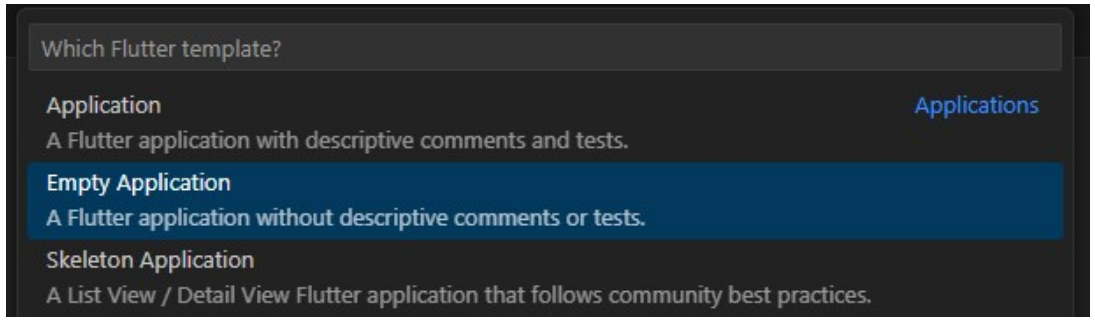
Create a HelloWorld Flutter app in VS Code (part 1)

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

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

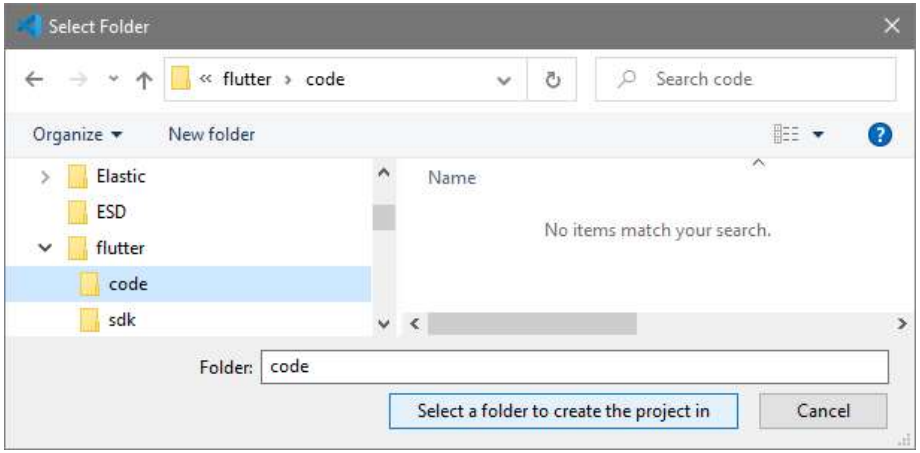


In the next drop-down, select “Empty Application”:

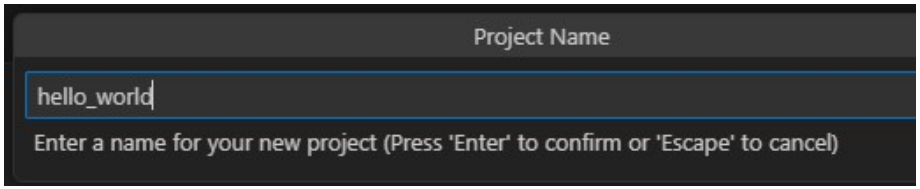


Create a HelloWorld Flutter app in VS Code (part 2)

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

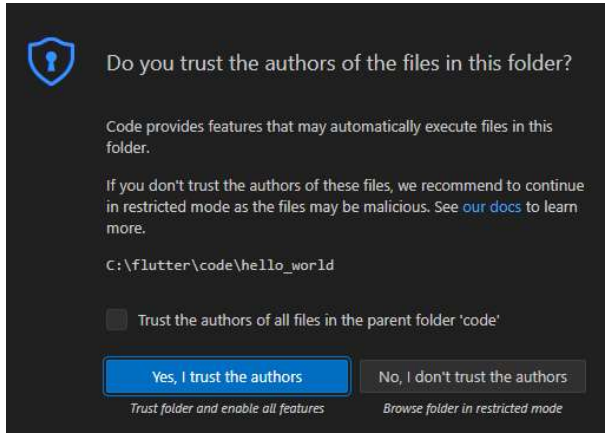


Enter the project name (no blanks or capital letters are allowed):

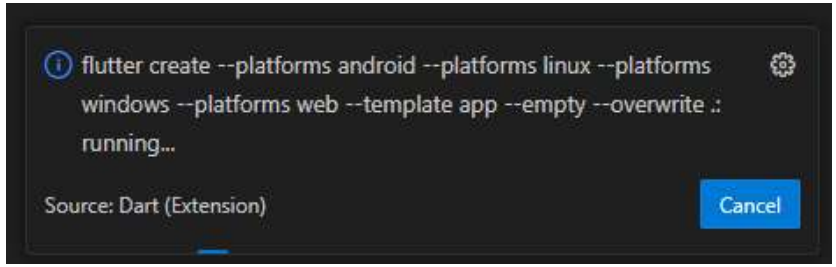


Create a HelloWorld Flutter app in VS Code (part 3)

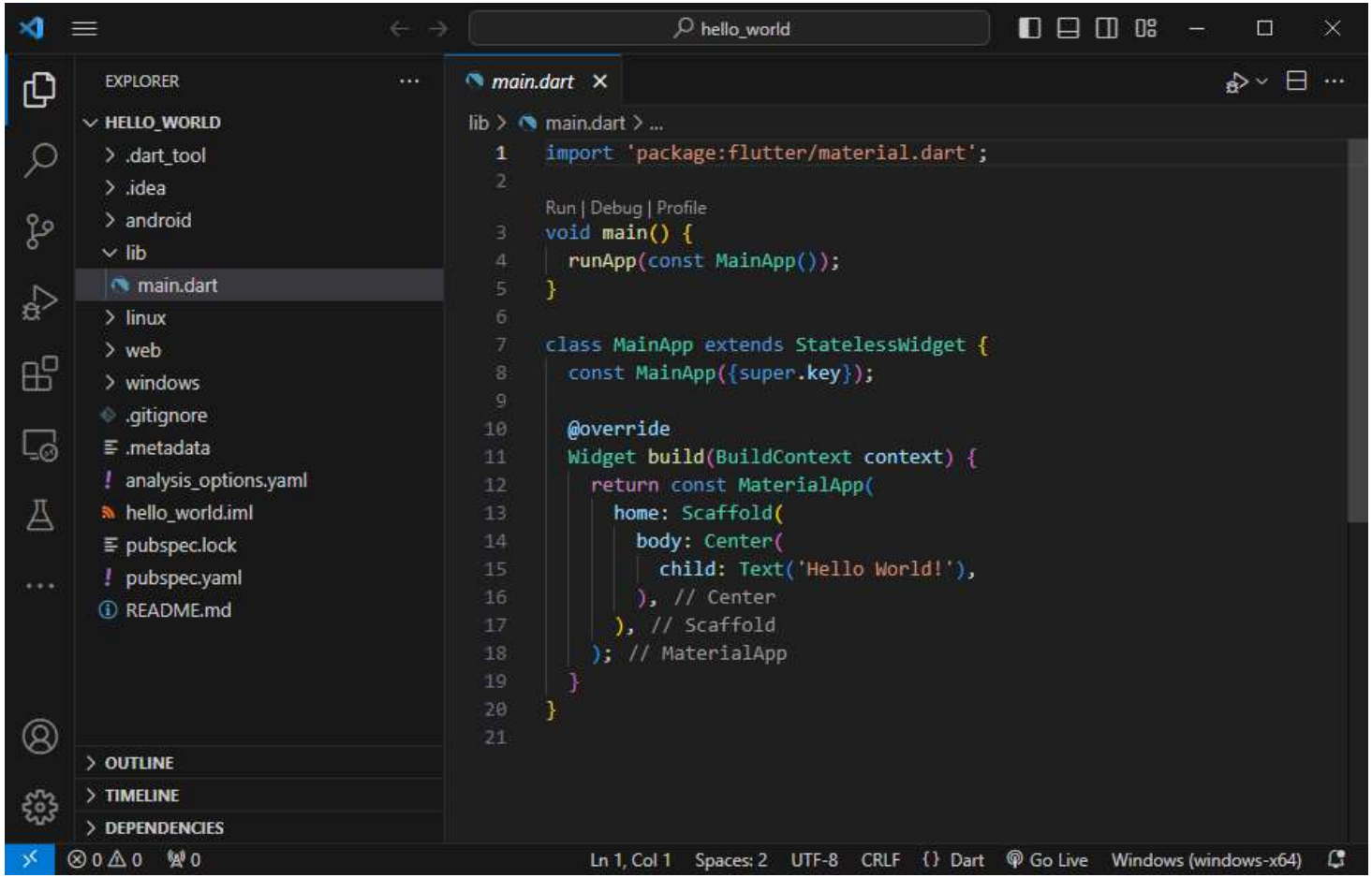
Allow VS Code to open the new created folder:



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

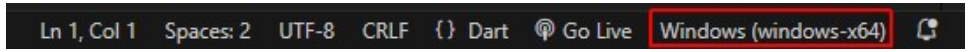


Your first created Flutter app in VS Code



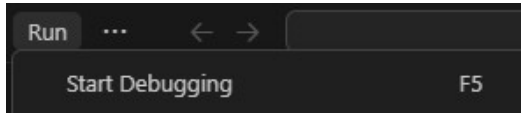
Test your app on Chrome or Edge (part 1)

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



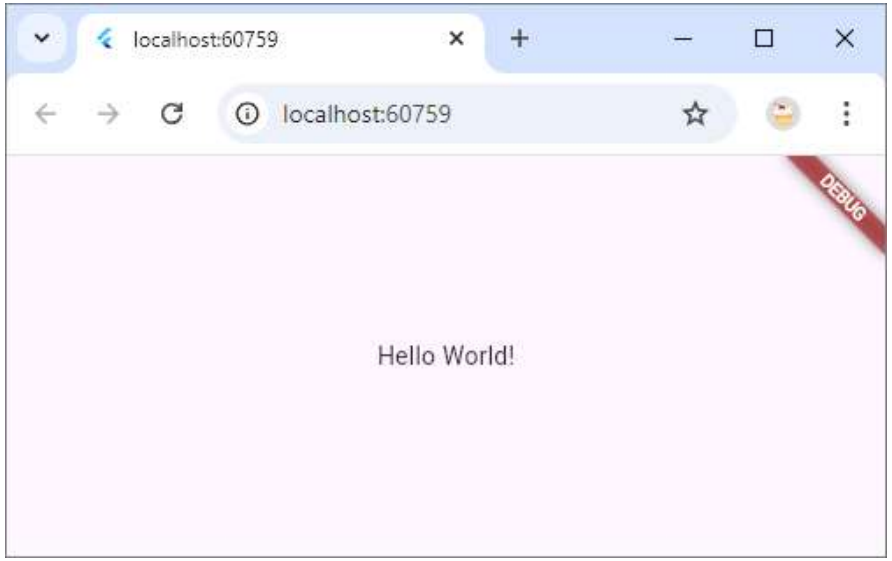
Select Chrome or Edge:

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

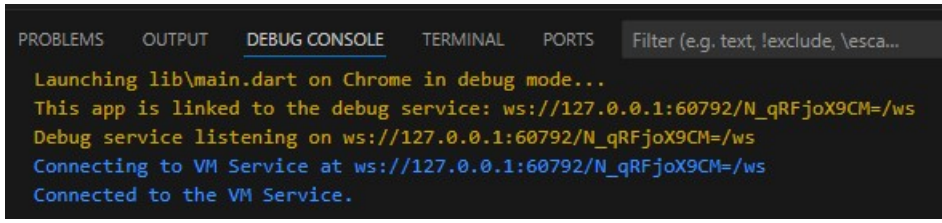


Test your app on Chrome or Edge (part 2)

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



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

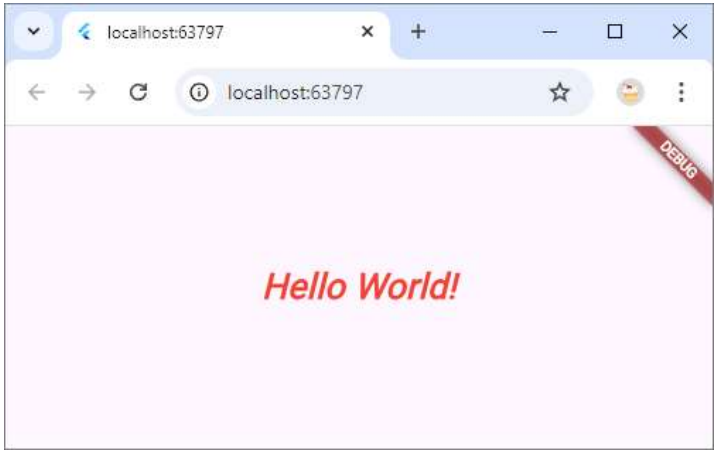


Change some Text properties

Define a style for your Text widget:

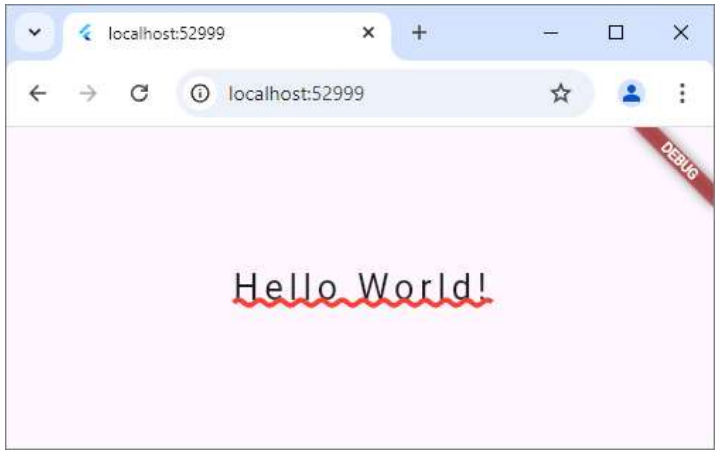
```
@override
Widget build(BuildContext context) {
  return const MaterialApp(
    home: Scaffold(
      body: Center(
        child: Text('Hello World!',
          style: TextStyle(
            color: Colors.red,
            fontSize: 25,
            fontStyle: FontStyle.italic,
            fontWeight: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



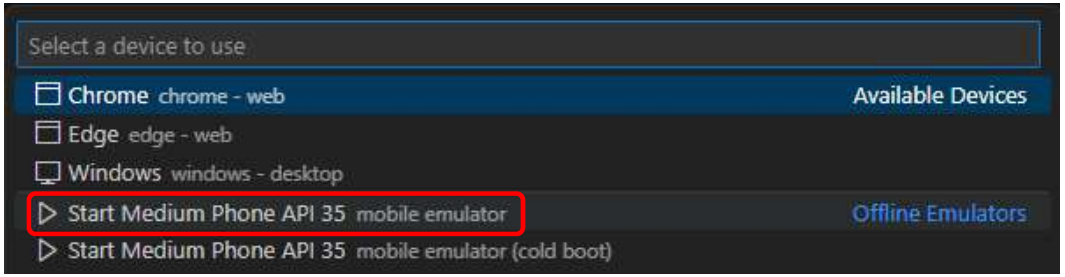
More Text properties

```
body: Center(
  child: Text('Hello World!',
      style: TextStyle(
          fontSize: 25,
          decoration: TextDecoration.underline,
          decorationColor: Colors.red,
          decorationStyle: TextDecorationStyle.wavy,
          decorationThickness: 3,
          letterSpacing: 4
          //wordSpacing: -10,
          )), // TextStyle // Text
), // Center
```

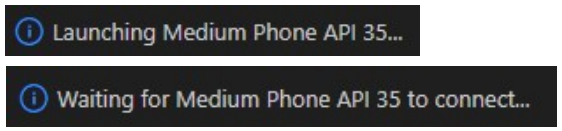


Test your app on Android Emulator (part 1)

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



It takes some time to start
and to connect:



Then press F5 to start debugging.

Be patient! The first build for Android can take **several minutes**, see next slide.

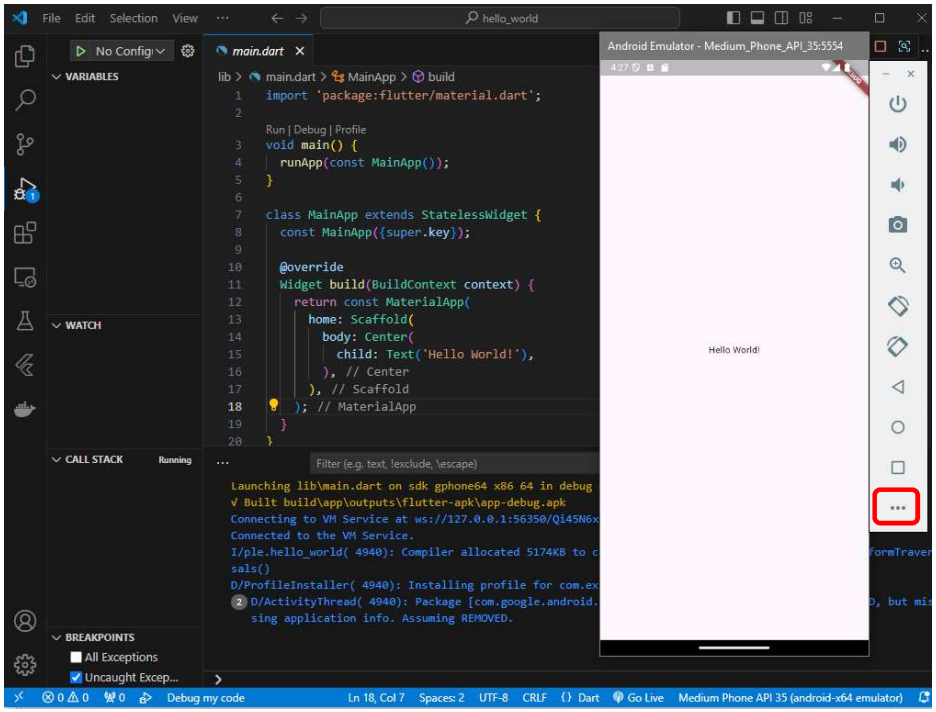
Test your app on Android Emulator (part 2)

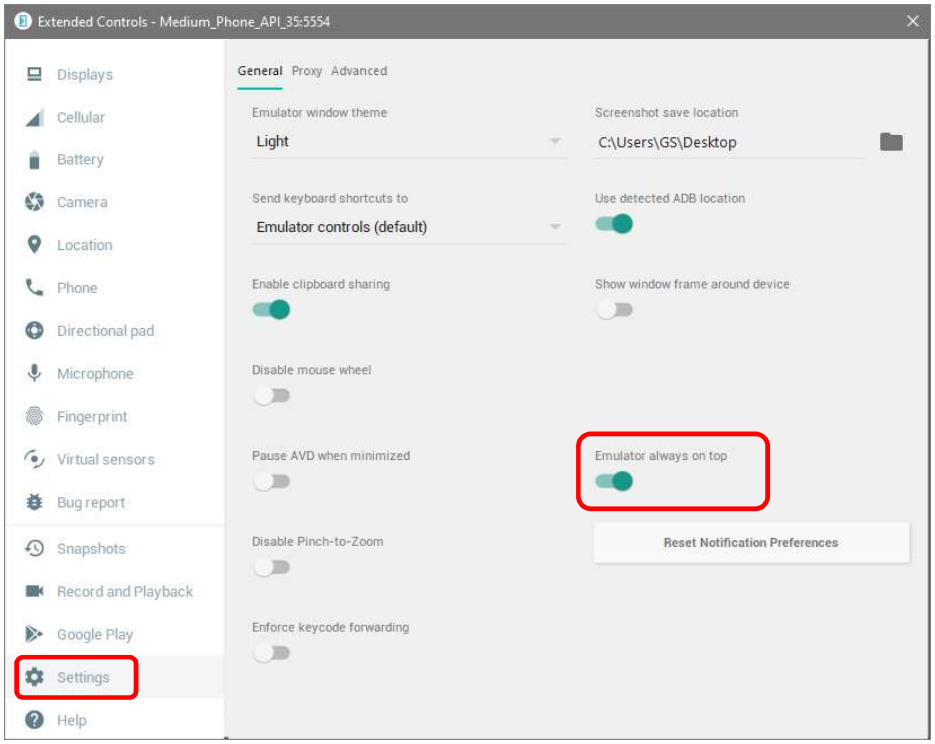
During the first build for Android, Android SDK (Software Development Kit) Build-Tools are downloaded, which takes time:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   Filter (e.g. text, !exclude, \escape)
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
Warning: SDK processing. This version only understands SDK XML versions up to 3 but an SDK XML file of vers
ion 4 was encountered. This can happen if you use versions of Android Studio and the command-line tools tha
t were released at different times.
Checking the license for package Android SDK Build-Tools 33.0.1 in C:\Users\GS\AppData\Local\Android\sdk\li
censes
License for package Android SDK Build-Tools 33.0.1 accepted.
Preparing "Install Android SDK Build-Tools 33.0.1 v.33.0.1".
"Install Android SDK Build-Tools 33.0.1 v.33.0.1" ready.
Installing Android SDK Build-Tools 33.0.1 in C:\Users\GS\AppData\Local\Android\sdk\build-tools\33.0.1
"Install Android SDK Build-Tools 33.0.1 v.33.0.1" complete.
"Install Android SDK Build-Tools 33.0.1 v.33.0.1" finished.
warning: [options] source value 8 is obsolete and will be removed in a future release
warning: [options] target value 8 is obsolete and will be removed in a future release
warning: [options] To suppress warnings about obsolete options, use -Xlint:-options.
3 warnings
√ Built build\app\outputs\flutter-apk\app-debug.apk
Connecting to VM Service at ws://127.0.0.1:54303/3NCQ0uIXKSM=/ws
Connected to the VM Service.
D/ProfileInstaller( 6395): Installing profile for com.example.after_new_anroid_studio
```

Test your app on Android Emulator (part 2)

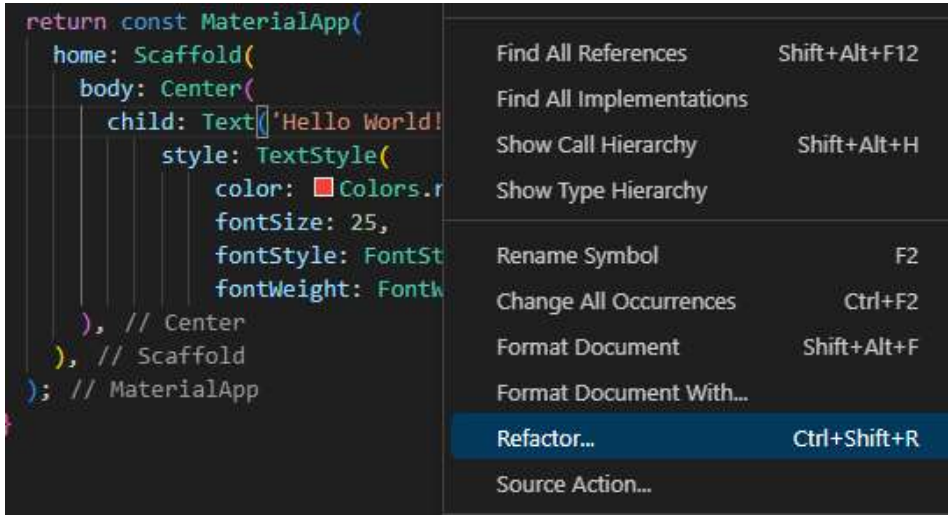
Finally the emulator appears on top of VS Code and stays on-top with setting:

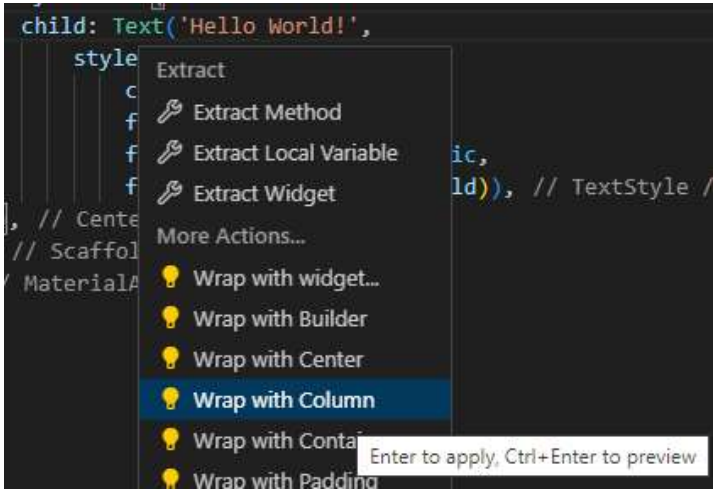




Allow more widgets by introducing a Column

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

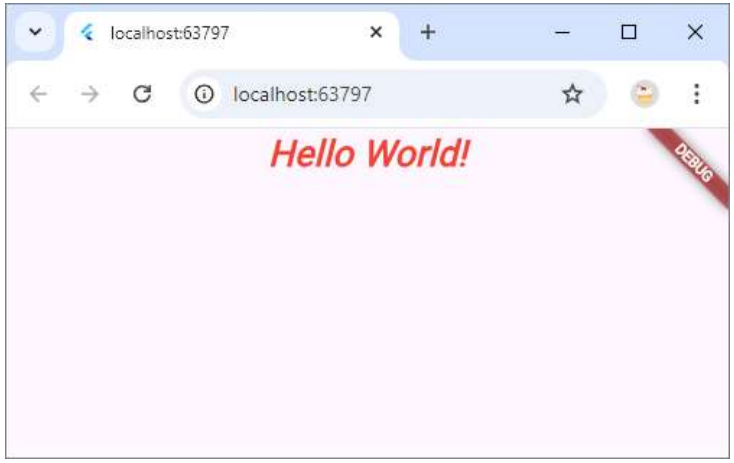




This will add a “Column” widget around your Text. Column widgets can have several children:

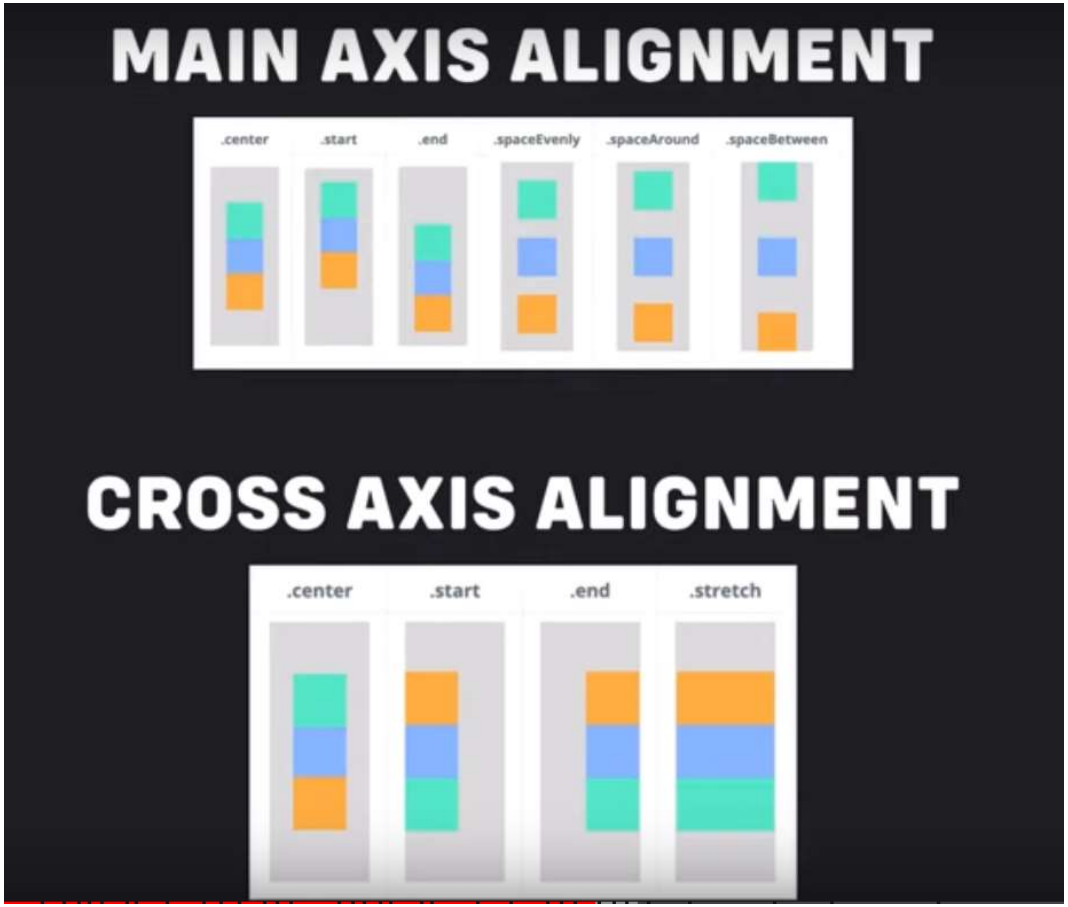
Allow more widgets by introducing a Column

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 either by setting the **mainAxisSize** property of the column to “MainAxisSize.min” or by setting the **mainAxisAlignment** property of the column:

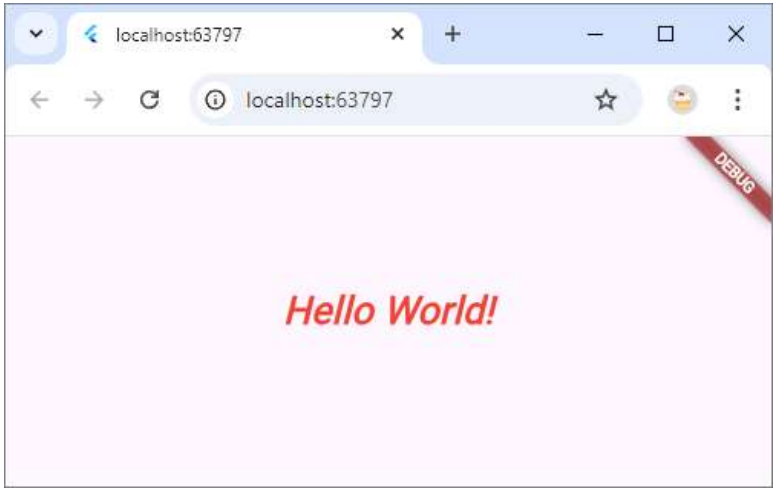
Axis alignments of a Column



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

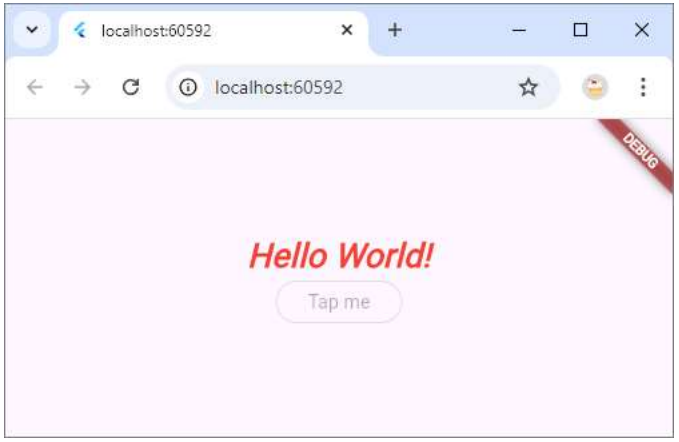
Center the text again with MainAxisAlignment

```
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

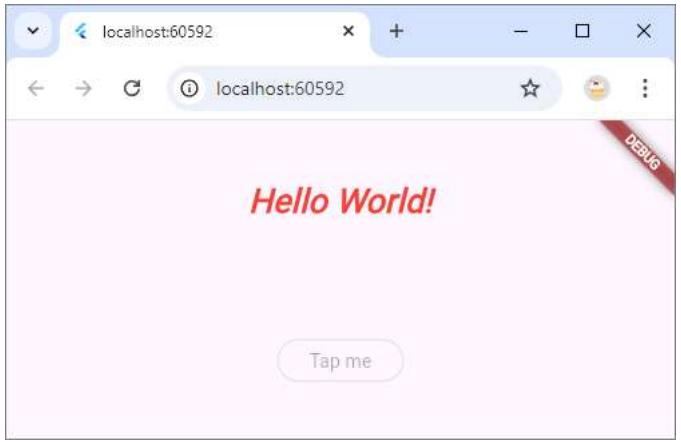
```
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text('Hello World!',
          style: TextStyle(
              color: Colors.red,
              fontSize: 25,
              fontStyle: FontStyle.italic,
              fontWeight: FontWeight.bold)),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



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

Bring some space between text and button

```
body: Center(
  child: Column(
    //mainAxisSize: MainAxisSize.min,
    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"))
    ],
  ), // Column
), // Center
```



Define an “onPressed” handler

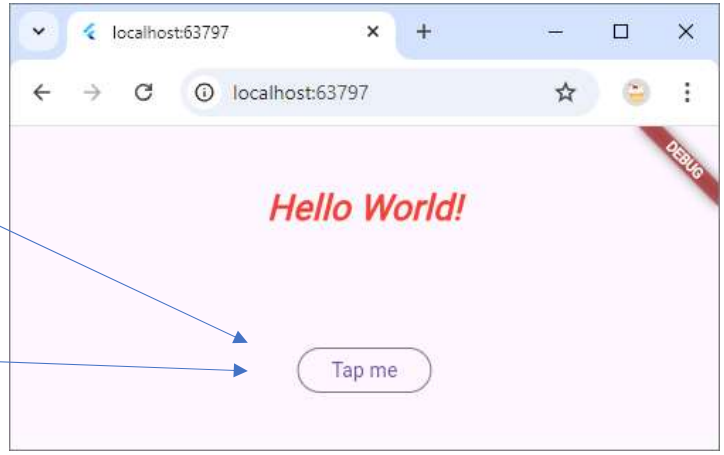
Either with a new function (can be inside or outside the class, normally inside):

```
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
```

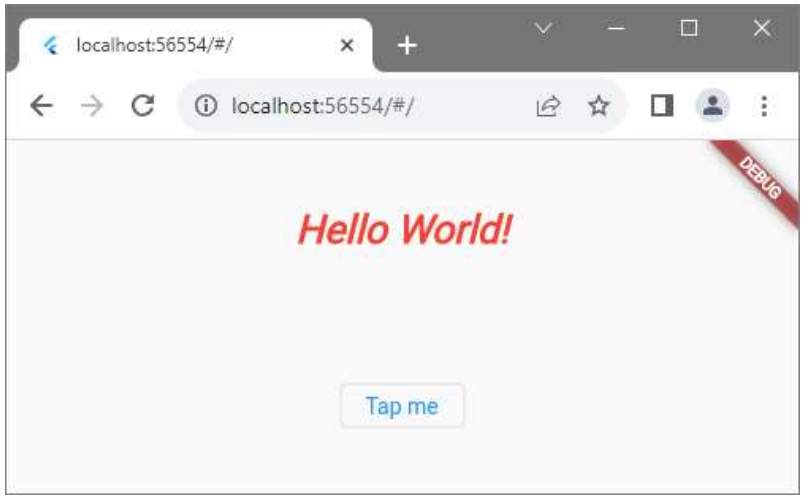


After pressing the button 3 times:

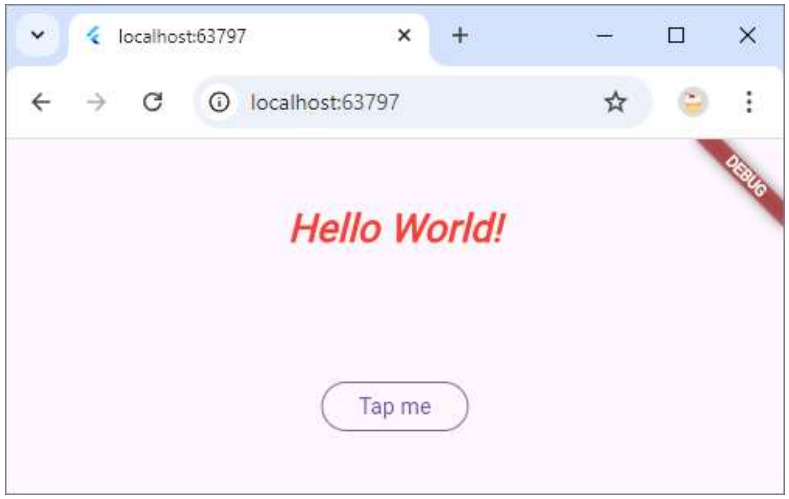
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
3 OutlinedButton was pressed
```

Default styles may change over time

Last year (with older Flutter version):



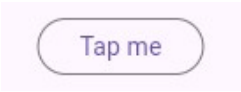
Now with Flutter version 3.24.3:



Style the button

Without style:

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



With style:

```
OutlinedButton(
    style: OutlinedButton.styleFrom(
      minimumSize: Size(200, 60),
      backgroundColor: Colors.yellow,
    ),
    onPressed: () {
      print("OutlinedButton was pressed");
    },
    child: Text("Tap me")) // OutlinedButton
```