



Your first experiences with Flutter

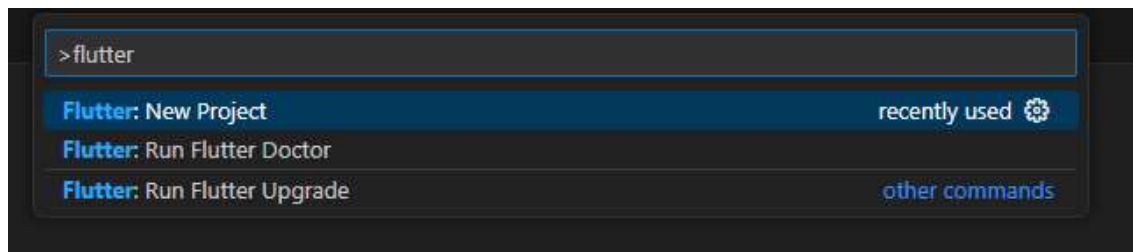
- Create a HelloWorld app in Flutter
- Run the app on Chrome and on Android emulator
- Change color and font and other properties of text widgets
- Observe your code changes directly with “Hot Reload” (no Rebuild needed)
- Learn about columns and rows and their axis alignments
- Know that Flutter offers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



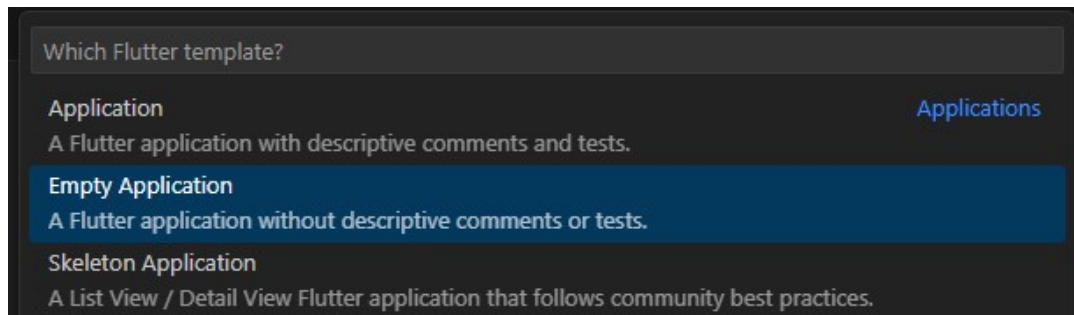
Create a HelloWorld Flutter app in VS Code (part 1)

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

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



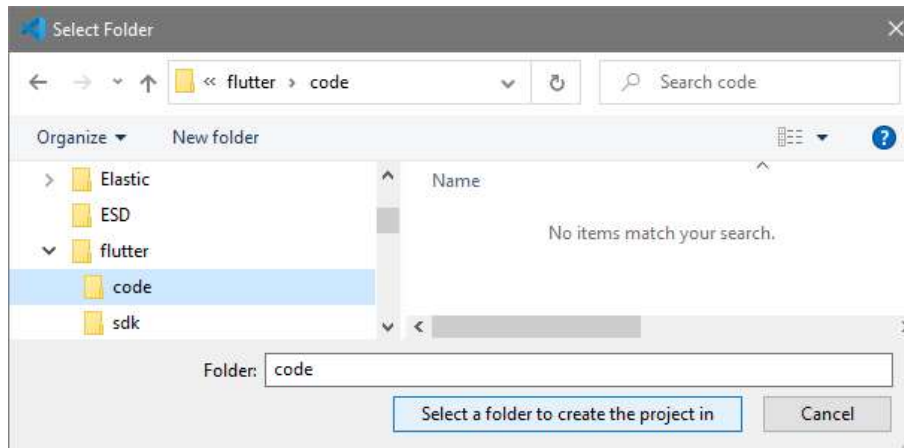
In the next drop-down, select “Empty Application”:



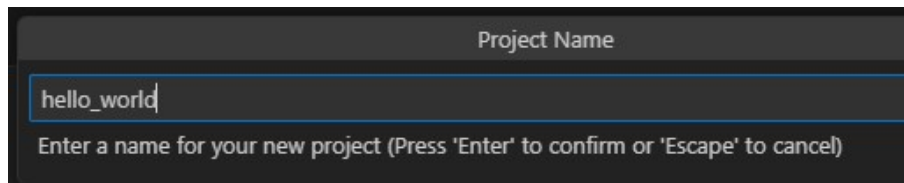


Create a HelloWorld Flutter app in VS Code (part 2)

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



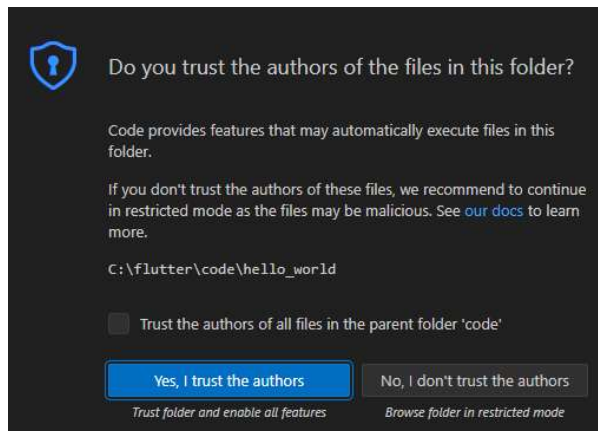
Enter the project name (no blanks or capital letters are allowed):



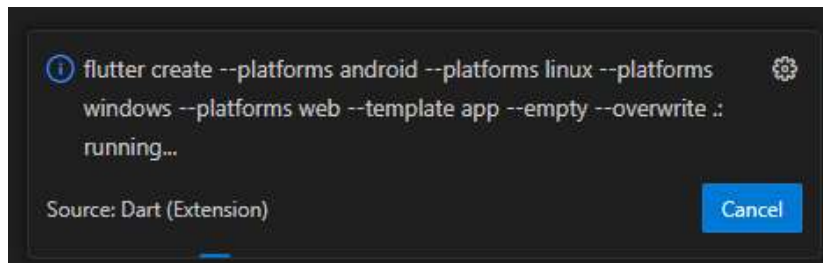


Create a HelloWorld Flutter app in VS Code (part 3)

Allow VS Code to open the new created folder:

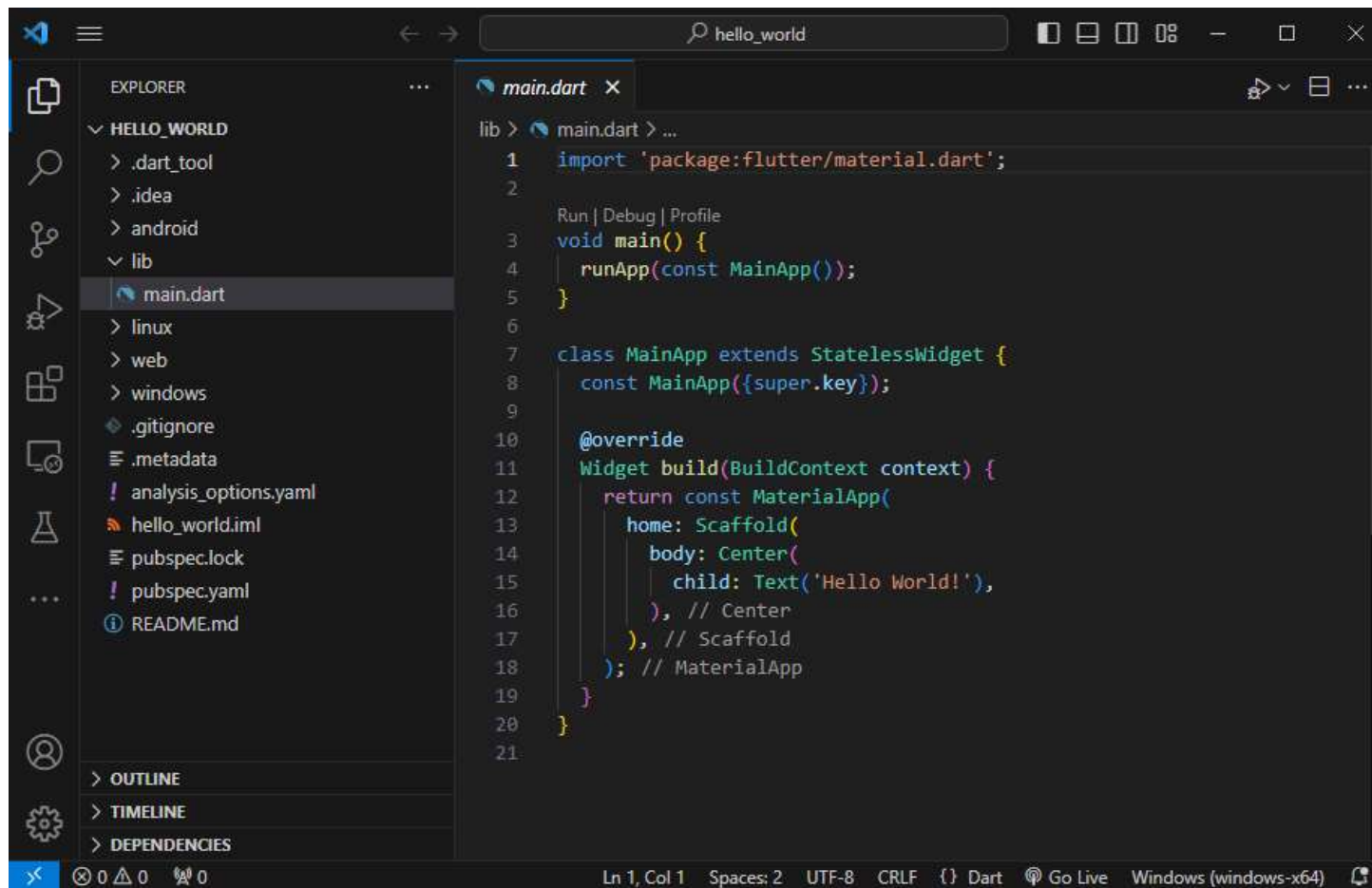


Wait until the project is created (you need an Internet connection during this step):





Your first created Flutter app in VS Code

A screenshot of the Visual Studio Code (VS Code) editor interface. The left sidebar shows the 'EXPLORER' view with a file tree for a project named 'HELLO_WORLD'. The tree includes folders like '.dart_tool', '.idea', 'android', 'lib', 'linux', 'web', 'windows', and files like '.gitignore', '.metadata', 'analysis_options.yaml', 'hello_world.iml', 'pubspec.lock', 'pubspec.yaml', and 'README.md'. The 'lib' folder is expanded, showing 'main.dart' selected. The main editor area displays the code in 'main.dart'. The code is a Dart file for a Flutter app, starting with an import statement for 'package:flutter/material.dart'. It defines a 'main' function that calls 'runApp' with a 'MainApp' instance. 'MainApp' is a class that extends 'StatelessWidget' and overrides the 'build' method. The 'build' method returns a 'MaterialApp' widget, which contains a 'Scaffold' widget. The 'Scaffold' has a 'body' of type 'Center', which contains a 'Text' widget with the text 'Hello World!'. The status bar at the bottom shows 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'CRLF', '{ } Dart', 'Go Live', and 'Windows (windows-x64)'.

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

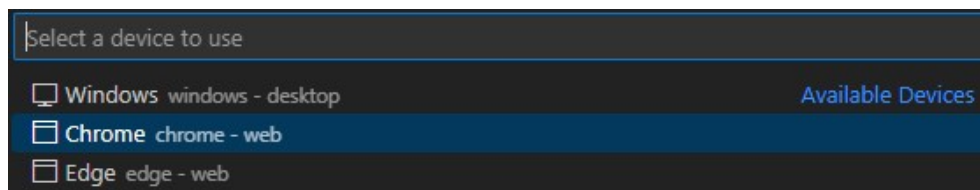


Test your app on Chrome or Edge (part 1)

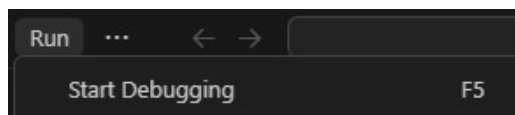
In the bottom line of VS Code, tap the red marked area:



Select Chrome or Edge:



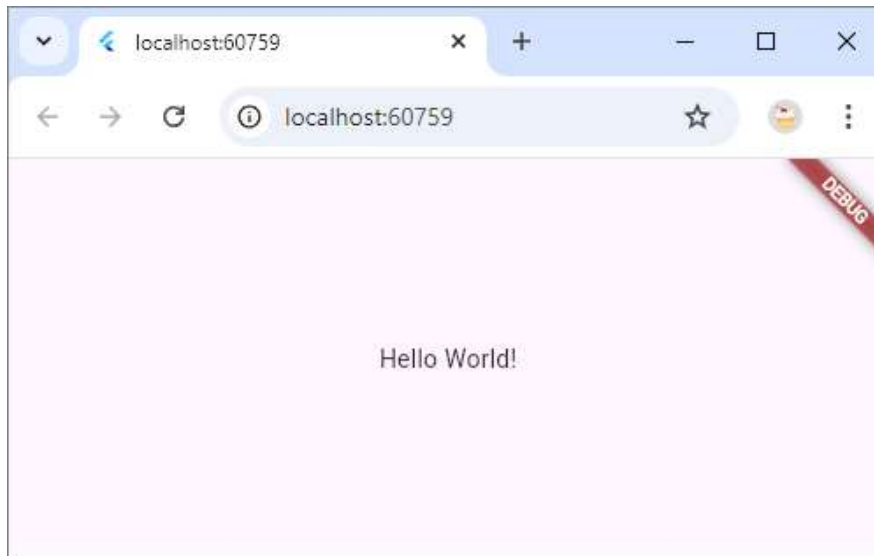
Select menu “Run / Start Debugging” or press F5:





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  Filter (e.g. text, !exclude, \esca...  
Launching lib\main.dart on Chrome in debug mode...  
This app is linked to the debug service: ws://127.0.0.1:60792/N_qRFjoX9CM=/ws  
Debug service listening on ws://127.0.0.1:60792/N_qRFjoX9CM=/ws  
Connecting to VM Service at ws://127.0.0.1:60792/N_qRFjoX9CM=/ws  
Connected to the VM Service.
```

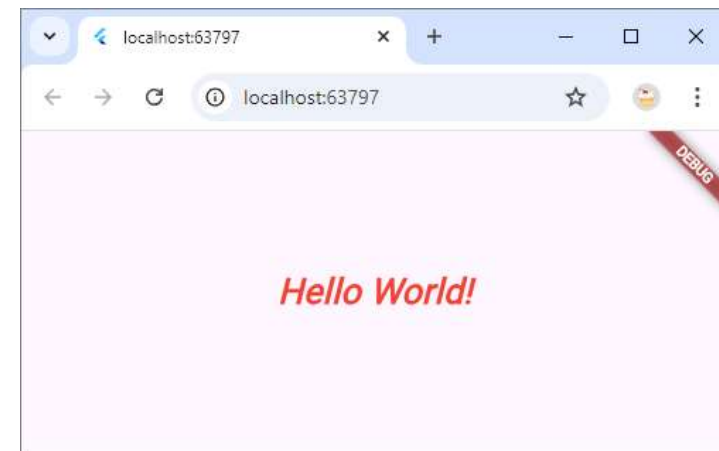


Change some Text properties

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
  }
}
```

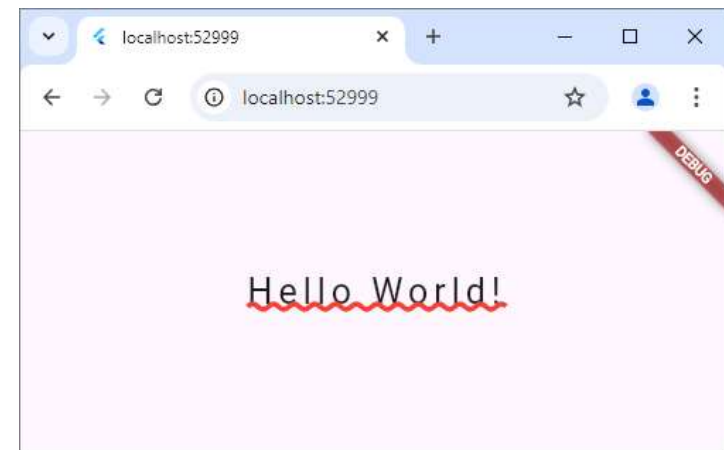
After saving your code, a “Hot Reload” is performed automatically and you can see the changes in Chrome:





More Text properties

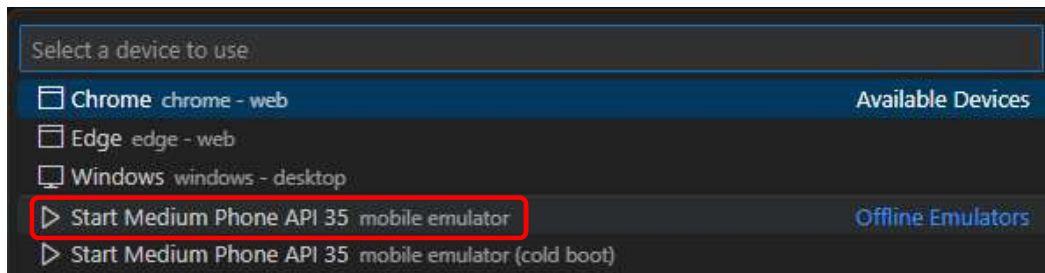
```
body: Center(  
  child: Text('Hello World!',  
    style: TextStyle(  
      fontSize: 25,  
      decoration: TextDecoration.underline,  
      decorationColor: Colors.red,  
      decorationStyle: TextDecorationStyle.wavy,  
      decorationThickness: 3,  
      letterSpacing: 4  
      //wordSpacing: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



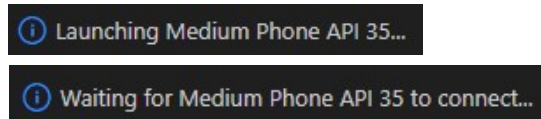


Test your app on Android Emulator (part 1)

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:

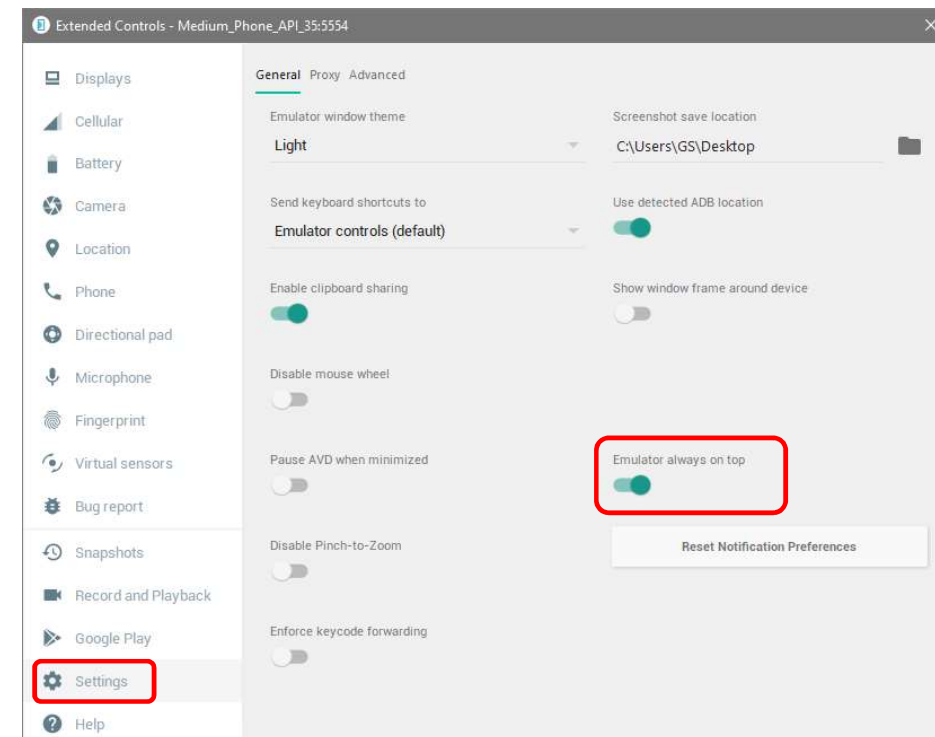
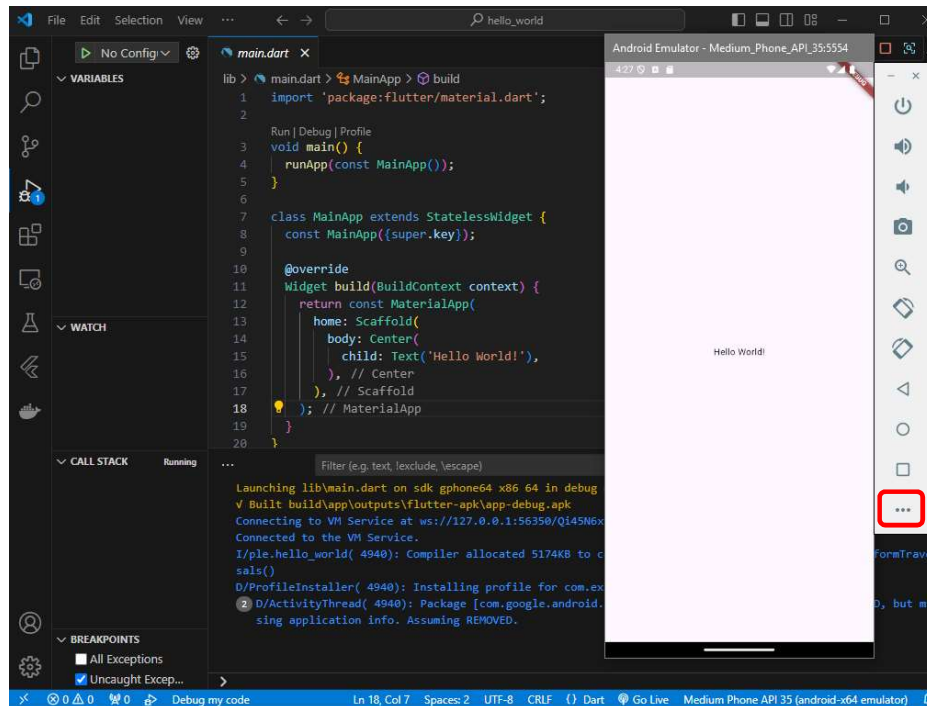
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Filter (e.g. text, !exclude, \escape)  🔍  ☰  ^  ✕

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 version 4 was encountered. This can happen if you use versions of Android Studio and the command-line tools that 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\licenses
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_android_studio
```



Test your app on Android Emulator (part 2)

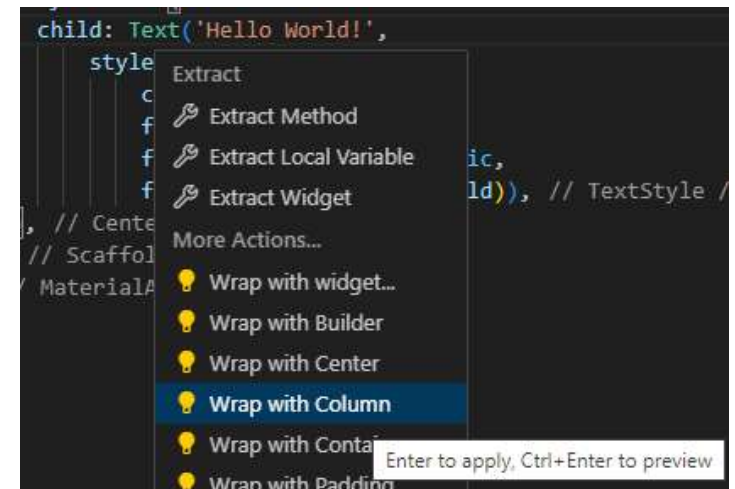
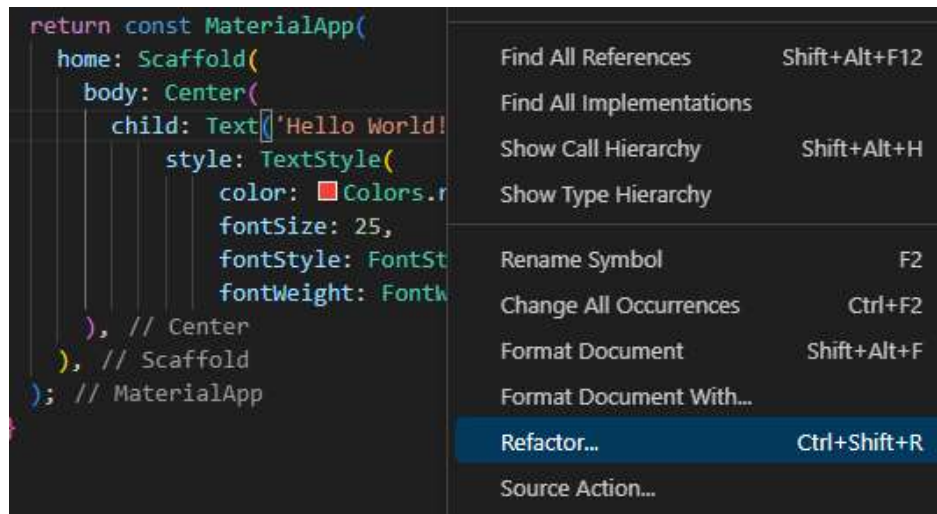
Finally the emulator appears on top of VS Code and stays on-top with setting:





Allow more widgets by introducing a Column

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



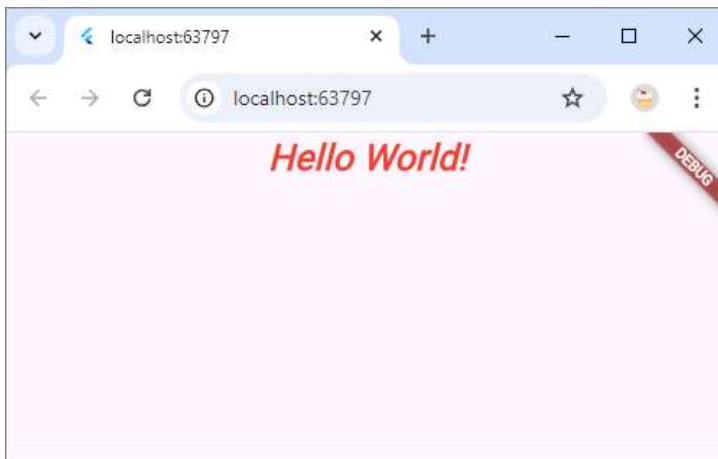
This will add a “Column” widget around your Text. Column widgets can have several children:





Allow more widgets by introducing a Column

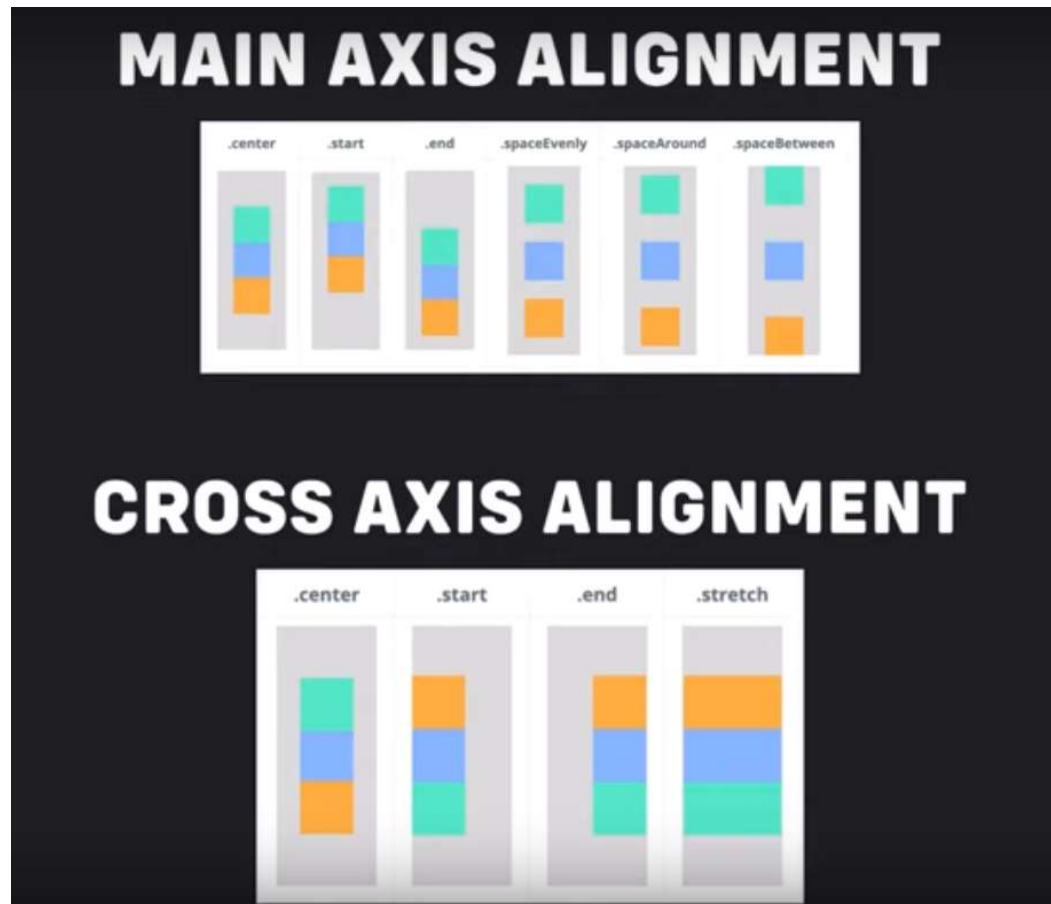
After saving your code, the “Hello World” text will move up, because Columns take the whole space and put their children by default on the top of the column:



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

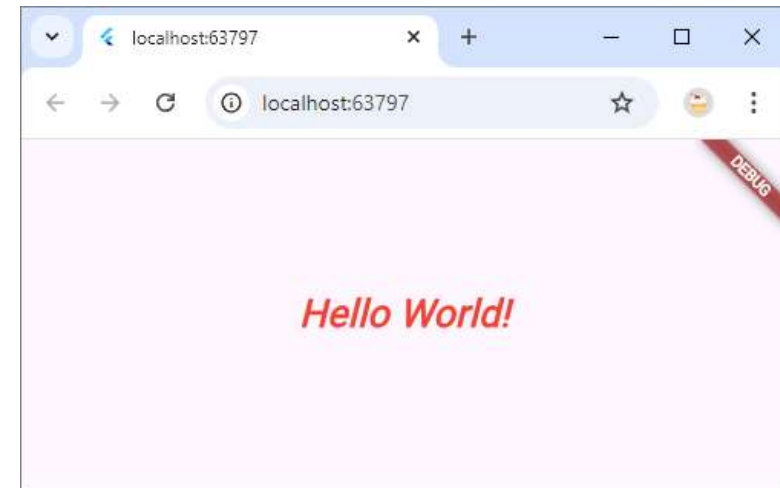


Picture taken from <https://www.youtube.com/watch?v=D4nhaszNW4o>



Center the text again with MainAxisAlignment

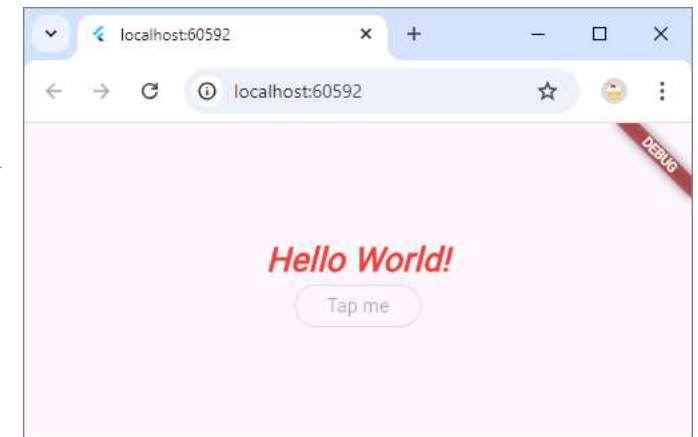
```
Widget build(BuildContext context) {  
  return const MaterialApp(  
    home: Scaffold(  
      body: Center(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.center,  
          children: [  
            Text('Hello World!',  
              style: TextStyle(  
                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
```

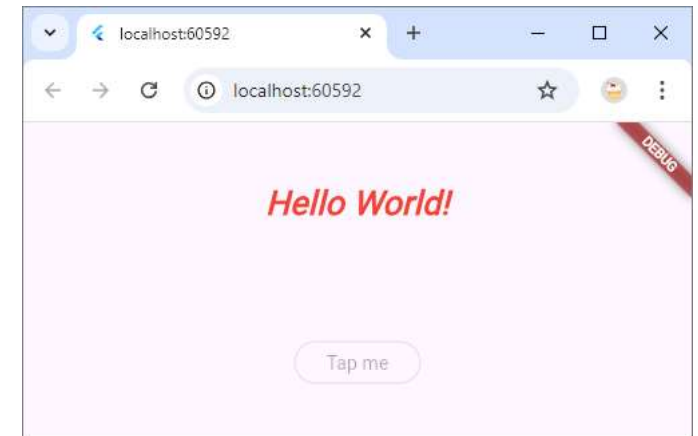


The button is disabled as long as “onPressed” is null.



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

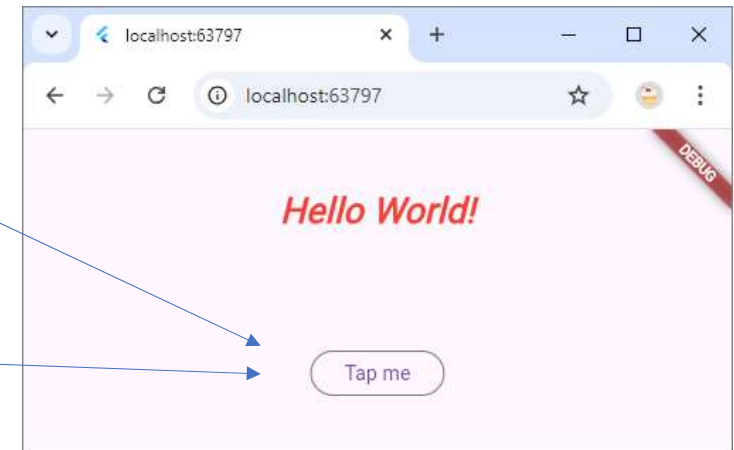
Either with a new function (can be inside or outside the class, normally inside):

```
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
```



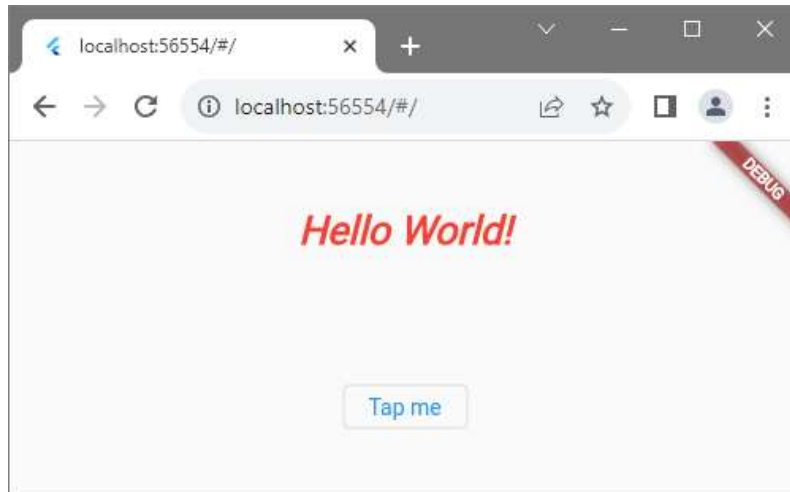
After pressing the button 3 times:



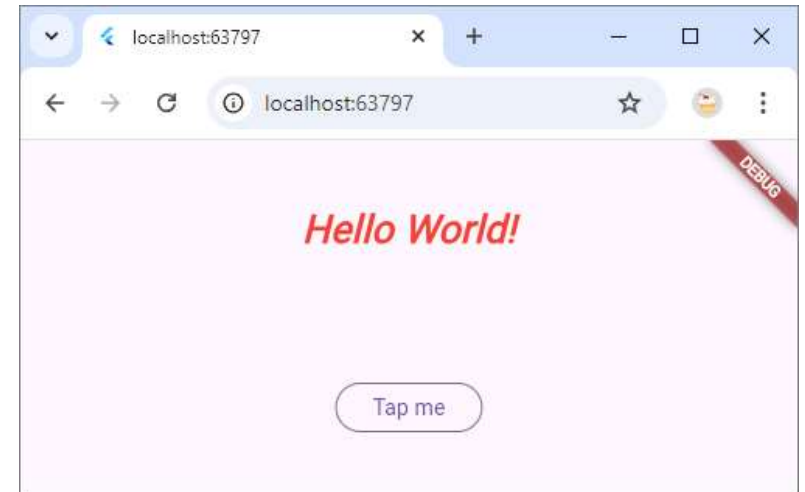


Default styles may change over time

Last year (with older Flutter version):



Now with Flutter version 3.24.3:

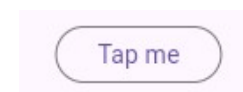




Style the button

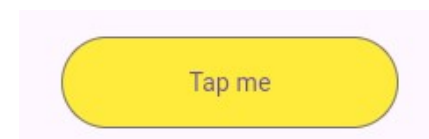
Without style:

```
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
```





Style the button (continued)

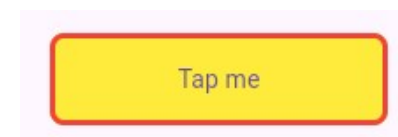
From last page:

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



With more style properties:

```
OutlinedButton(  
  style: OutlinedButton.styleFrom(  
    shape: RoundedRectangleBorder(  
      borderRadius: BorderRadius.circular(8.0),  
    ), // RoundedRectangleBorder  
    side: BorderSide(width: 3, color: Colors.red),  
    minimumSize: Size(200, 60),  
    backgroundColor: Colors.yellow,  
  ),  
  onPressed: () {  
    print("OutlinedButton was pressed");  
  },  
  child: Text("Tap me")) // OutlinedButton
```



Idea for this code was copied from
<https://stackoverflow.com/questions/64322596/how-to-achieve-rounded-corners-on-new-outlinedbutton-widget-in-flutter>



Add an icon inside the button

```
OutlinedButton(  
  style: OutlinedButton.styleFrom(  
    //minimumSize: Size(200, 60),  
    maximumSize: Size(150, 60),  
    backgroundColor: Colors.yellow,  
  ),  
  onPressed: () {  
    print("OutlinedButton was pressed");  
  },  
  child: Row(  
    mainAxisAlignment: MainAxisAlignment.spaceBetween,  
    children: [  
      Icon(Icons.download),  
      Text("Download"),  
    ],  
  ),  
) // Row // OutlinedButton
```

Otherwise the button takes the whole width.





Axis alignments of a Column ... compared to a Row

MAIN AXIS ALIGNMENT



CROSS AXIS ALIGNMENT



CROSS AXIS ALIGNMENT



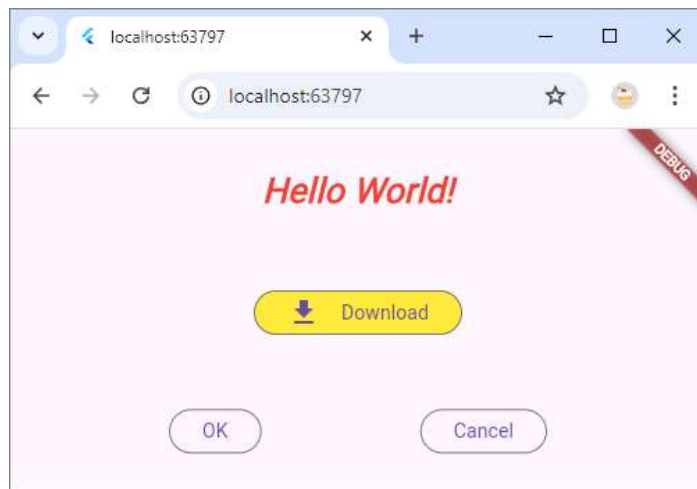
MAIN AXIS ALIGNMENT





Exercise

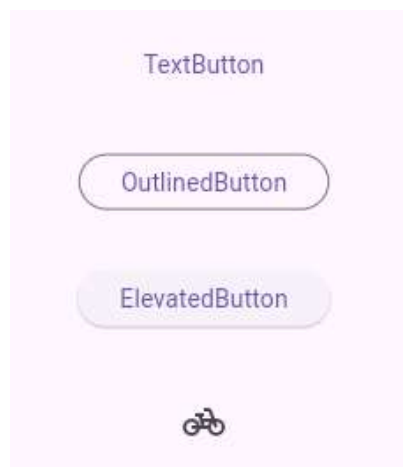
Add an OK and a Cancel button below the Download button:



```
Row(  
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  children: [  
    OutlinedButton(  
      onPressed: () {  
        print("OK was pressed");  
      },  
      child: Text("OK")), // OutlinedButton  
    OutlinedButton(  
      onPressed: () {  
        print("Cancel was pressed");  
      },  
      child: Text("Cancel")), // OutlinedButton  
  ],  
) // Row
```



More Button widgets in Flutter



```
TextButton(  
  onPressed: handlePressed, child: Text("TextButton")), // TextButton  
OutlinedButton(  
  onPressed: handlePressed, child: Text("OutlinedButton")), // OutlinedButton  
ElevatedButton(  
  onPressed: handlePressed, child: Text("ElevatedButton")), // ElevatedButton  
IconButton(  
  onPressed: handlePressed, icon: Icon(Icons.pedal_bike)), // IconButton
```



Exercise

Increase the elevation of the ElevatedButton:



Hint:

Define a style for this button like we did for the OutlinedButton and look for a property controlling the elevation.

Solution:

```
ElevatedButton(  
  style: ElevatedButton.styleFrom(elevation: 20),  
  onPressed: handlePressed, child: Text("ElevatedButton")), // ElevatedButton
```




Appendix: Color class details

```
class Color {  
  /// Construct a color from the lower 32 bits of an [int].  
  ///  
  /// The bits are interpreted as follows:  
  ///  
  /// * Bits 24-31 are the alpha value.  
  /// * Bits 16-23 are the red value.  
  /// * Bits 8-15 are the green value.  
  /// * Bits 0-7 are the blue value.  
  ///  
  /// In other words, if AA is the alpha value in hex, RR the red value in hex,  
  /// GG the green value in hex, and BB the blue value in hex, a color can be  
  /// expressed as `const Color(0xAAARRGGBB)`.  
  ///  
  /// For example, to get a fully opaque orange, you would use `const  
  /// Color(0xFFFF9000)` (`FF` for the alpha, `FF` for the red, `90` for the  
  /// green, and `00` for the blue).  
  const Color(int value) : value = value & 0xFFFFFFFF;
```

The `new` keyword in `Java` is a fundamental part of the language used to create new objects. It dynamically allocates memory for an object and returns a reference to that memory. The `new` keyword is essential for object-oriented programming in Java, as it enables the instantiation of classes.

In `Dart` language there's a new keyword. It is used with class constructors to create new instances. Since Dart 2.0 the new keyword is `optional` and can be omitted. As this keyword doesn't bring any additional value, it is recommended to avoid using it.

```
Color buttonColor =  new Color(0xFFFF9000);
```

Unnecessary 'new' keyword. `dart(unnecessary_new)`




```
Color buttonColor =  Color(0xFFFF9000);
```

Above texts were copied from <https://www.datacamp.com/doc/java/new> and https://next.sonarqube.com/sonarqube/coding_rules?open=dart%3AS7101&rule_key=dart%3AS7101



Appendix: Color class has several constructors

```
/// Construct a color from the lower 8 bits of four integers.
///
/// * `a` is the alpha value, with 0 being transparent and 255 being fully
///   opaque.
/// * `r` is [red], from 0 to 255.
/// * `g` is [green], from 0 to 255.
/// * `b` is [blue], from 0 to 255.
///
/// Out of range values are brought into range using modulo 255.
///
/// See also [fromRGBO], which takes the alpha value as a floating point
/// value.
const Color.fromARGB(int a, int r, int g, int b) :
  value = ((a & 0xff) << 24) |
          ((r & 0xff) << 16) |
          ((g & 0xff) << 8) |
          ((b & 0xff) << 0) & 0xFFFFFFFF;
```

```
Color buttonColor1 =  Color(0xFFFF9000);
Color buttonColor2 =  Color.fromARGB(0xFF, 0xFF, 0x90, 0);
Color buttonColor3 =  Color.fromARGB(255, 255, 144, 0);
```



Appendix: Colors class

```
abstract final class Colors {  
  /// Completely invisible.  
  static const Color transparent = Color(0x00000000);  
  
  /// Completely opaque black.  
  ///  
  ///   
  ///  
  /// See also:  
  ///  
  /// * [black87], [black54], [black45], [black38], [black26], [black12], which  
  ///   are variants on this color but with different opacities.  
  /// * [white], a solid white color.  
  /// * [transparent], a fully-transparent color.  
  static const Color black = Color(0xFF000000);  
}
```

“**abstract**” classed cannot be instantiated.

“**final**” classes cannot be sub-classed

```
static const Color black87 = Color(0xDD000000);
```

```
static const Color black54 = Color(0x8A000000);
```

```
static const Color black12 = Color(0x1F000000);
```

```
Color textColor = Colors.black87;
```

```
static const Color white = Color(0xFFFFFFFF);
```

```
static const Color white70 = Color(0xB3FFFFFF);
```

```
static const Color white12 = Color(0x1FFFFFFF);
```

```
Color textColor = Colors.white70;
```

black 0xFF000000

black12 0x1F000000

black26 0x42000000

black38 0x61000000

black45 0x73000000

black54 0x8A000000

black87 0xDD000000

white 0xFFFFFFFF

white10 0x1AFFFFFF

white12 0x1FFFFFFF

white24 0x3DFFFFFF

white30 0x4DFFFFFF

white38 0x62FFFFFF

white54 0x8AFFFFFF

white60 0x99FFFFFF

white70 0xB3FFFFFF



Appendix: Color swatches*) in Colors class

```
/// The red primary color and swatch.
///
static const MaterialColor red = MaterialColor(
    _redPrimaryValue,
    <int, Color>{
        50: Color(0xFFFFEBEE),
        100: Color(0xFFFFCDD2),
        200: Color(0xFFEF9A9A),
        300: Color(0xFFE57373),
        400: Color(0xFFE53535),
        500: Color(_redPrimaryValue),
        600: Color(0xFF539393),
        700: Color(0xFF32F2F2),
        800: Color(0xFF628282),
        900: Color(0xFFB71C1C),
    },
);
static const int _redPrimaryValue = 0xFFFF44336;
```

```
Color alarmColor = Colors.red.shade700;
```

```
static const MaterialAccentColor redAccent = MaterialAccentColor(
    _redAccentValue,
    <int, Color>{
        100: Color(0xFFFF8A80),
        200: Color(_redAccentValue),
        400: Color(0xFFFF1744),
        700: Color(0xFFD50000),
    },
);
static const int _redAccentValue = 0xFFFF5252;
```

Colors.red[50]	0xFFFFEBEE
Colors.red.shade50	
Colors.red[100]	0xFFFFCDD2
Colors.red.shade100	
Colors.red[200]	0xFFEF9A9A
Colors.red.shade200	
Colors.red[300]	0xFFE57373
Colors.red.shade300	
Colors.red[400]	0xFFE53535
Colors.red.shade400	
Colors.red	0xFFFF44336
Colors.red[600]	0xFF539393
Colors.red.shade600	
Colors.red[700]	0xFFD32F2F
Colors.red.shade700	
Colors.red[800]	0xFFC62828
Colors.red.shade800	
Colors.red[900]	0xFFB71C1C
Colors.red.shade900	

Colors.redAccent[100]	0xFFFF8A80
Colors.redAccent.shade100	
Colors.redAccent	0xFFFF5252
Colors.redAccent[400]	0xFFFF1744
Colors.redAccent.shade400	
Colors.redAccent[700]	0xFFD50000
Colors.redAccent.shade700	

*) "swatch" in German: Muster

Red 50	#FFEBEE	Pink 50	#FCE4EC	Purple 50	#F3E5F5	Deep Purple 50	#EDE7F6	Indigo 50	#E8EAF6	Blue 50	#E3F2FD	Light Blue 50	#E1F5FE	Cyan 50	#E0F7FA	Teal 50	#E0F2F1	Green 50	#E8F5E9
100	#FFCDD2	100	#F8BBD0	100	#E1BEE7	100	#D1C4E9	100	#C5CAE9	100	#BBDEFB	100	#B3E5FC	100	#B2EBF2	100	#B2DFDB	100	#C8E6C9
200	#EF9A9A	200	#F48FB1	200	#CE93D8	200	#B39DDB	200	#9FA8DA	200	#90CAF9	200	#81D4FA	200	#80DEEA	200	#80CBC4	200	#A5D6A7
300	#E57373	300	#F06292	300	#BA68C8	300	#9575CD	300	#7986CB	300	#64B5F6	300	#4FC3F7	300	#4DD0E1	300	#4DB6AC	300	#81C784
400	#EF5350	400	#EC407A	400	#AB47BC	400	#7E57C2	400	#5C6BC0	400	#42A5F5	400	#29B6F6	400	#26C6DA	400	#26A69A	400	#66BB6A
500	#F44336	500	#E91E63	500	#9C27B0	500	#673AB7	500	#3F51B5	500	#2196F3	500	#03A9F4	500	#00BCD4	500	#009688	500	#4CAF50
600	#E53935	600	#D81B60	600	#8E24AA	600	#5E35B1	600	#3949AB	600	#1E88E5	600	#039BE5	600	#00ACC1	600	#00897B	600	#43A047
700	#D32F2F	700	#C2185B	700	#7B1FA2	700	#512DA8	700	#303F9F	700	#1976D2	700	#028BD1	700	#0097A7	700	#00796B	700	#388E3C
800	#C62828	800	#AD1457	800	#6A1B9A	800	#4527A0	800	#283593	800	#1565C0	800	#0277BD	800	#00838F	800	#00695C	800	#2E7D32
900	#B71C1C	900	#880E4F	900	#4A148C	900	#311B92	900	#1A237E	900	#0D47A1	900	#01579B	900	#006064	900	#004D40	900	#1B5E20
A100	#FF8A80	A100	#FF80AB	A100	#EA80FC	A100	#B388FF	A100	#8C9EFF	A100	#82B1FF	A100	#80D8FF	A100	#84FFFF	A100	#A7FFEB	A100	#B9F6CA
A200	#FF5252	A200	#FF4081	A200	#E040FB	A200	#7C4DFE	A200	#536DFE	A200	#448AFF	A200	#40C4FF	A200	#18FFFF	A200	#64FFDA	A200	#69F0AE
A400	#FF1744	A400	#F50057	A400	#D500F9	A400	#651FFF	A400	#3D5AFE	A400	#2979FF	A400	#00B0FF	A400	#00E5FF	A400	#1DE9B6	A400	#00E676
A700	#D50000	A700	#C51162	A700	#AA00FF	A700	#6200EA	A700	#304FFE	A700	#2962FF	A700	#0091EA	A700	#00B8D4	A700	#00BFA5	A700	#00C853

Light Green 50	#F1F8E9	Lime 50	#F9FBE7	Yellow 50	#FFFDE7	Amber 50	#FFF8E1	Orange 50	#FFF3E0	Deep Orange 50	#FBE9E7	Brown 50	#EFEBE9	Gray 50	#FAFAFA	Blue Gray 50	#ECEFF1	Black	#000000
100	#DCEDC8	100	#F0F4C3	100	#FFF9C4	100	#FFECB3	100	#FFE0B2	100	#FFCCBC	100	#D7CCC8	100	#F5F5F5	100	#CFD8DC	White	#FFFFFF
200	#C5E1A5	200	#E6EE9C	200	#FFF59D	200	#FFE082	200	#FFCC80	200	#FFAB91	200	#BCAAA4	200	#EEEEEE	200	#B0BEC5		
300	#AED581	300	#DCE775	300	#FFF176	300	#FFD54F	300	#FFB74D	300	#FF8A65	300	#A1887F	300	#E0E0E0	300	#90A4AE		
400	#9CCC65	400	#D4E157	400	#FFEE58	400	#FFCA28	400	#FFA726	400	#FF7043	400	#8D6E63	400	#BDBDBD	400	#78909C		
500	#8BC34A	500	#CDDC39	500	#FFEB3B	500	#FFC107	500	#FF9800	500	#FF5722	500	#795548	500	#9E9E9E	500	#607D8B		
600	#7CB342	600	#C0CA33	600	#FDD835	600	#FFB300	600	#FB8C00	600	#F4511E	600	#6D4C41	600	#757575	600	#546E7A		
700	#689F38	700	#AFB42B	700	#FBC02D	700	#FFA000	700	#F57C00	700	#E64A19	700	#5D4037	700	#616161	700	#455A64		
800	#558B2F	800	#9E9D24	800	#F9A825	800	#FF8F00	800	#EF6C00	800	#D84315	800	#4E342E	800	#424242	800	#37474F		
900	#33691E	900	#827717	900	#F57F17	900	#FF6F00	900	#E65100	900	#BF360C	900	#3E2723	900	#212121	900	#263238		
A100	#CCFF90	A100	#F4FF81	A100	#FFFF8D	A100	#FFE57F	A100	#FFD180	A100	#FF9E80								
A200	#B2FF59	A200	#EEFF41	A200	#FFFF00	A200	#FFD740	A200	#FFAB40	A200	#FF6E40								
A400	#76FF03	A400	#C6FF00	A400	#FFEA00	A400	#FFC400	A400	#FF9100	A400	#FF3D00								
A700	#64DD17	A700	#AEEA00	A700	#FFD600	A700	#FFAB00	A700	#FF6D00	A700	#DD2C00								



Important: never modify Flutter sources

