Text () Widget

A Text is a widget in Flutter that allows us to **display a string of text with a single line in our application**. Depending on the layout constraints, we can break the string across multiple lines or might all be displayed on the same line. If we do not specify any styling to the text widget, it will use the closest **DefaultTextStyle** class style. This class does not have any explicit style. In this article, we are going to learn how to use a Text widget and how to style it in our application.

Text Widget Constructor:

The text widget constructor used to make the custom look and feel of our text in Flutter:

```
1. const Text(String data,{
2.     Key key,
3.     TextStyle style,
4.     StrutStyle strutStyle,
5.     TextAlign textAlign,
6.     TextDirection textDirection,
7.     TextOverflow overflow,
8.     bool softWrap,
9.     double textScaleFactor,
10.    int maxLines,
11.    String semanticsLabel,
12.    TextWidthBasis textWidthBasis,
13.    TextHeightBehavior textHeightBehavior
14.    }
15.)
```

The following are the essential properties of the Text widget used in our application:

TextAlign: It is used to specify how our text is aligned horizontally. It also controls the text location.

TextDirection: It is used to determine how textAlign values control the layout of our text. Usually, we write text from left to right, but we can change it using this parameter.

Overflow: It is used to determine when the text will not fit in the available space. It means we have specified more text than the available space.

TextScaleFactor: It is used to determine the scaling to the text displayed by the Text widget. Suppose we have specified the text scale factor as 1.5, then our text will be 50 percent larger than the specified font size.

SoftWrap: It is used to determine whether or not to show all text widget content when there is not enough space available. If it is true, it will show all content. Otherwise, it will not show all content.

MaxLines: It is used to determine the maximum number of lines displayed in the text widget.

TextWidthBasis: It is used to control how the text width is defined.

TextHeightBehavior: It is used to control how the paragraph appears between the first line and descent of the last line.

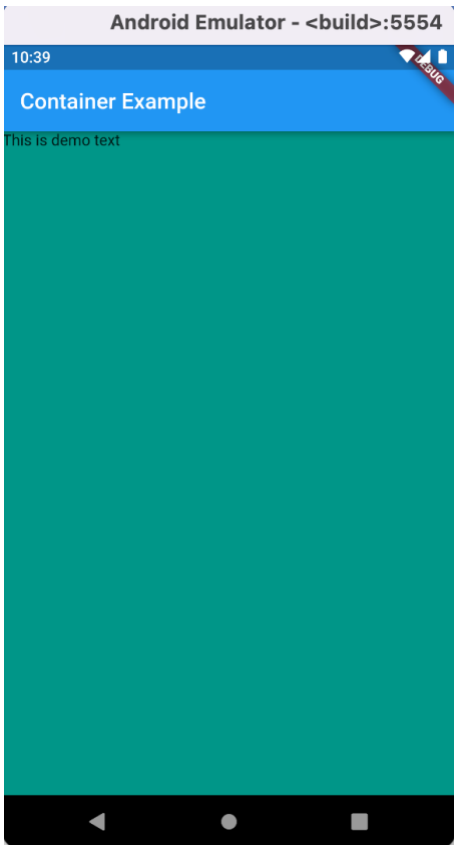
Style: It is the most common property of this widget that allows developers to styling their text. It can do styling by specifying the foreground and background color, font size, font weight, letter and word spacing, locale, shadows, etc. See the table to understand it more easily:

Attributes	Descriptions
foreground	It determines the paint as a foreground for the text.
background	It determines the paint as a background for the text.
fontWeight	It determines the thickness of the text.
fontSize	It determines the size of the text.
fontFamily	It is used to specify the typeface for the font. For this, we need to download a typeface file in our project, and then keep this file into the assets/font folder. Finally, config the pubspec.yaml file to use it in the project.

fontStyle	It is used to style the font either in bold or italic form.
Color	It is used to determine the color of the text.
letterSpacing	It is used to determine the distance between the characters of the text.
wordSpacing	It is used to specify the distance between two words of the text.
shadows	It is used to paint underneath the text.
decoration	We use this to decorate text using the three parameters: decoration, decorationColor, decorationStyle. The decoration determines the location, decorationColor specify the color, decorationStyle determine the shape.

Extra-Workout:

Let’s start the workout by just displaying a default text using Text widget.



main.dart

lib > main.dart > ...

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

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  final name = 'shyam';
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.teal,
        appBar: AppBar(
          title: const Text('Container Example'),
        ), // AppBar
        body: const SafeArea(
          child:  Text('This is demo text'),
        ), // SafeArea
      ), // Scaffold
    ); // MaterialApp
  }
}
```

As above code, we just display the default text. Now, by using the text property we can change the text.

So first look at the style property:
Style property will takes the TextStyle() widget as value.

```
18          body: const SafeArea(
19            child: Text(
20              'This is demo text',
21              style: TextStyle(
22                fontWeight: FontWeight.bold,
23                fontSize: 20.0,
24                fontStyle: FontStyle.italic,
25                backgroundColor: Colors.yellow,
26                color: Colors.blue,
27                letterSpacing: -1,
28                wordSpacing: 20,
29
30              ), // TextStyle
31            ), // Text
```

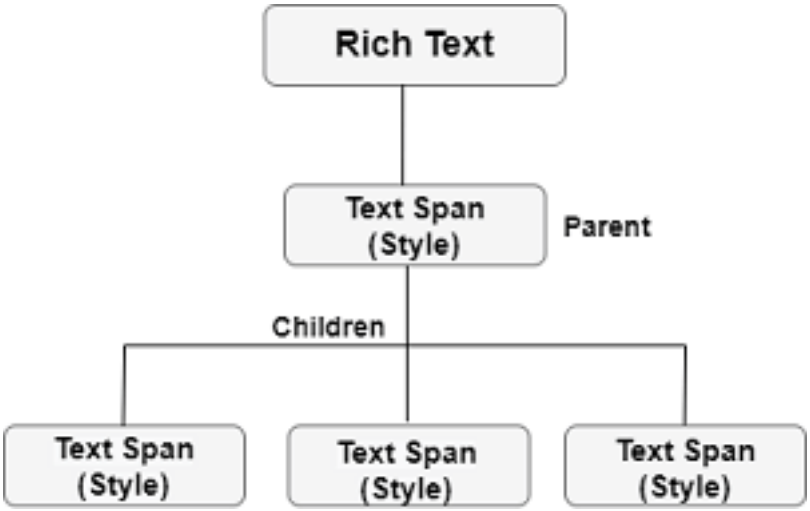
To view the other property of Text widget – put the text inside the container and use textalign, textdirection, overflow to view the impact of those property.

RichText () Widget:

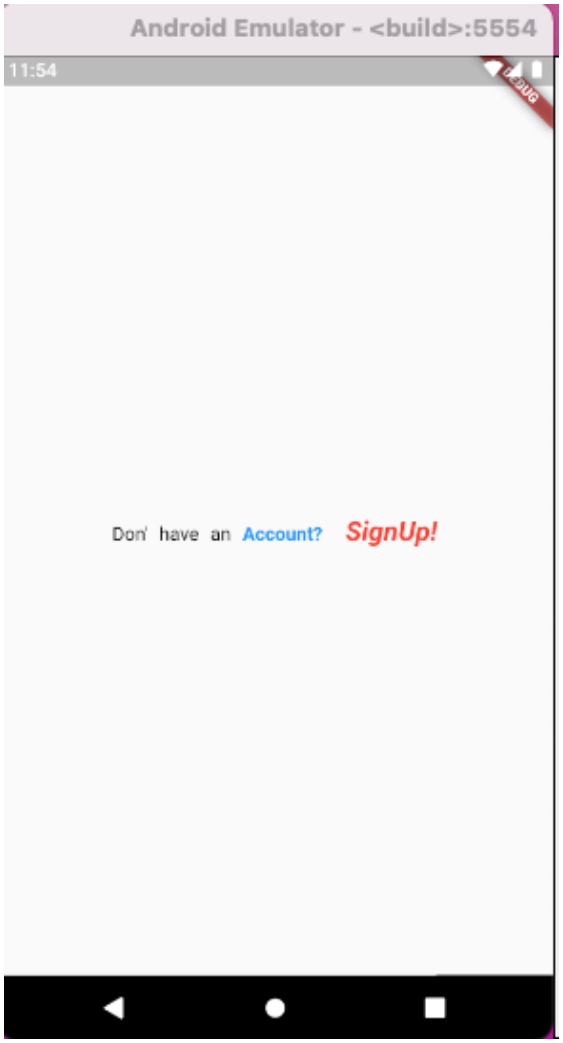
Sometimes we want to **show a line or a paragraph with multiple styles** such as bold, italicized, underlined, different color, different font or everything at once. In that case, we should have to use the RichText widget that allows us to perform multiple test styles without switching many widgets.

RichText is a very useful widget in Flutter, which is used for displaying a paragraph of text on the UI with multiple styles. Inside the widget, we can have different styles by giving it a **tree of TextSpan** widgets. Each TextSpan can set its own style for overriding the default style.

The below image explains the structure of the RichText widget. In this image, the parent TextSpan has its own style property and a text parameter, and then it can contain several children TextSpan who have their own style property.



From the above overview, now we will see how to use this widget in our application.



lib > main.dart > ...

```dart
void main() {…
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: SafeArea(
          child: Center(
            child: RichText(
              text: const TextSpan(
                text: 'Don\' have an ',
                style: TextStyle(
                  color: Colors.black,
                  wordSpacing: 5,
                ), // TextStyle
                children: [
                  TextSpan(
                    text: 'Account?  ',
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                      color: Colors.blue,
                      wordSpacing: 5,
                    ), // TextStyle
                  ), // TextSpan
                  TextSpan(
                    text: 'SignUp!',
                    style: TextStyle(
                      color: Colors.red,
                      fontSize: 20,
                      fontStyle: FontStyle.italic,
                      fontWeight: FontWeight.bold,
                    ), // TextStyle
                  ), // TextSpan
                ],
              ), // TextSpan
            ), // RichText
          ), // Center
        ), // SafeArea
      ), // Scaffold
    ); // MaterialApp
  }
}
```

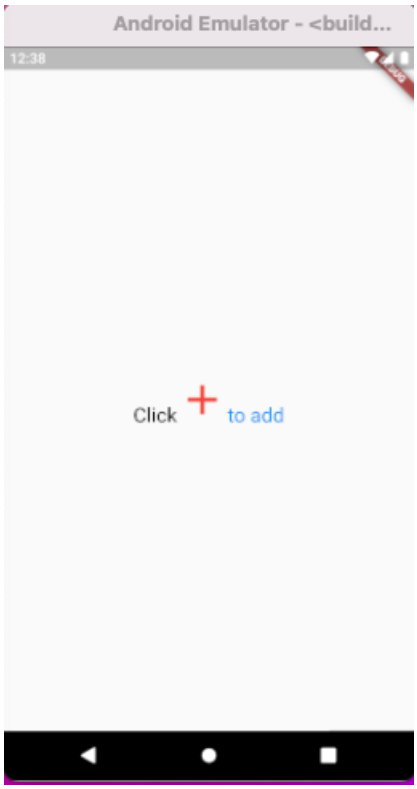
Now, try to underline the any of the word that is displayed in the screen.

To underline the text, we have to use decoration property of TextStyle under the style property of Text or TextSpan. (Do yourself).

```
TextSpan(
  text: 'Account?',
  style: TextStyle(
    decoration: TextDecoration.underline
    decorationThickness: 5,
    fontWeight: FontWeight.bold,
    color: Colors.blue,
    // wordSpacing: 10,
  ), // TextStyle
), // TextSpan
```

How to show an icon in a text widget?

- To show an icon in a text widget, flutter provides a WidgetSpan() inside the RichText().



```
15                child: Center(
16                  child: RichText(
17                    text: const TextSpan(children: [
18                      TextSpan(
19                        text: 'Click',
20                        style: TextStyle(
21                          color: Colors.black,
22                          fontSize: 20,
23                        ), // TextStyle
24                      ), // TextSpan
25                      WidgetSpan(
26                        child: Icon(
27                          Icons.add,
28                          color: Colors.red,
29                          size: 50,
30                        ), // Icon
31                      ), // WidgetSpan
32                      TextSpan(
33                        text: 'to add',
34                        style: TextStyle(fontSize: 20, color: Colors.blue),
35                      ), // TextSpan
36                    ]), // TextSpan
37                  ), // RichText
38                ), // Center
```