Stateless widget Vs Statelfulwidget

**Flutter Stateless Widgets**

The widgets whose state can not be altered once they are built are called stateless widgets. These widgets are immutable once they are built i.e. any amount of change in the variables, icons, buttons, or retrieving data can not change the state of the app. Below is the basic structure of a *stateless widget. Stateless* widget overrides the *build*() method and returns a widget.For example, we use *Text* or the *Icon* in our Flutter application where the state of the widget does not change in the *runtime*. It is used when the UI depends on the information within the object itself. Other examples can be *Text*, *RaisedButton*, *IconButtons*.

**Example of Stateless Widgets:**

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Container();

}

}

So let us see what this small code snippet tells us. The name of the stateless widget is MyApp which is being called from the runApp() and extends a stateless widget. Inside this MyApp a build function is overridden and takes BuildContext as a parameter. This BuildContext is unique to each and every widget as it is used to locate the widget inside the widget tree.

The build function contains a container which is again a widget of Flutter inside which we will design the UI of the app.  In the stateless widget, the build function is called only once which makes the UI of the screen.

**Example:**Stateless Widget

Dart

import 'package:flutter/material.dart';

//This function triggers the build process

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

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

home: Scaffold(

backgroundColor: const Color.fromARGB(255, 230, 255, 201),

appBar: AppBar(

leading: const Icon(Icons.menu),

backgroundColor: Colors.green,

title: const Text(

"GeeksforGeeks",

textAlign: TextAlign.start,

),

), // AppBar

body: const Center(

child: Text(

"Stateless Widget",

style: TextStyle(color: Colors.black, fontSize: 30),

),

), // Container

), // Scaffold

); // MaterialApp

}

}

## ****Flutter Stateful Widgets****

The widgets whose state can be altered once they are built are called stateful Widgets. These states are mutable and can be changed multiple times in their lifetime. This simply means the state of an app can change multiple times with different sets of variables, inputs, data. Below is the basic structure of a stateful widget. Stateful widget overrides the createState() method and returns a State. It is used when the UI can change dynamically. Some examples can be CheckBox, RadioButton, Form, TextField.

Classes that inherit “Stateful Widget” are immutable. But the State is mutable which changes in the runtime when the user interacts with it.

**Example**

import 'package:flutter/material.dart';

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

class MyApp extends StatefulWidget {

const MyApp({Key? key}) : super(key: key);

@override

// ignore: library\_private\_types\_in\_public\_api

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

@override

Widget build(BuildContext context) {

return Container();

}

}

So let us see what we have in this code snippet. The name of the Stateful Widget is MyApp which is called from the *runApp()* and extends a stateful widget. In the MyApp class, we override the create state function. This createState() function is used to create a mutable state for this widget at a given location in the tree. This method returns an instance for the respected state subclass. The other class which is \_MyAppState extends the state, which manages all the changes in the widget. Inside this class, the build function is overridden which takes the BuildContext as the parameter. This build function returns a widget where we design the UI of the app. Since it is a stateful widget the build function is called many times which creates the entire UI once again with all the changes.

**Example:**Stateful Widget

Dart

import 'package:flutter/material.dart';

//This function triggers the build process

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

// StatefulWidget

class MyApp extends StatefulWidget {

const MyApp({Key? key}) : super(key: key);

@override

// ignore: library\_private\_types\_in\_public\_api

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

home: Scaffold(

backgroundColor: Color.fromARGB(255, 230, 255, 201),

appBar: AppBar(

leading: const Icon(Icons.menu),

backgroundColor: Colors.green,

title: const Text(

"GeeksforGeeks",

textAlign: TextAlign.start,

),

), // AppBar

body: const Center(

child: Text(

"StateFul Widget",

style: TextStyle(color: Colors.black, fontSize: 30),

),

), // Container

), // Scaffold

); // MaterialApp

}

}

Stateless widget is useful when the part of the user interface you are describing does not depend on anything other than the configuration information and the BuildContext whereas a Stateful widget is useful when the part of the user interface you are describing can change dynamically.