SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER APPLICATIONS

23CAT607- CROSS-PLATFORM APP DEVELOPMENT

I YEAR II SEM

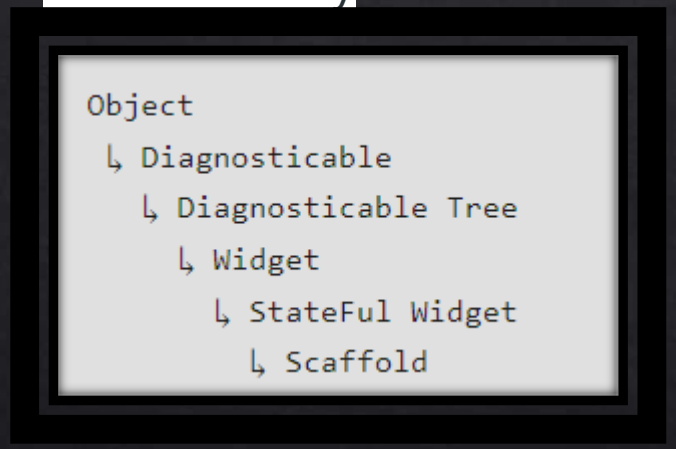
UNIT 2 – FLUTTER BASICS

TOPIC 7 – Scaffold, Material App, AppBar, Rich text widgets

Scaffold class

Scaffold is a class in flutter which provides many widgets or we can say APIs like Drawer, Snack-Bar, Bottom-Navigation-Bar, Floating-Action-Button, App-Bar, etc. Scaffold will expand or occupy the whole device screen. It will occupy the available space. Scaffold will provide a framework to implement the basic material design layout of the application.

class Hierarchy



Properties of Scaffold Class

app-Bar: It displays a horizontal bar which mainly placed at the top of the Scaffold. appBar uses the widget AppBar which has its own properties like elevation, title, brightness, etc.

body: It will display the main or primary content in the Scaffold. It is below the appBar and under the floatingActionButton. The widgets inside the body are at the left-corner by default.

floatingActionButton: FloatingActionButton is a button that is placed at the right bottom corner by default. FloatingActionButton is an icon button that floats over the content of the screen at a fixed place. If we scroll the page its position won't change, it will be fixed.

drawer: drawer is a slider menu or a panel which is displayed at the side of the Scaffold. The user has to swipe left to right or right to left according to the action defined to access the drawer menu. In the Appbar, an appropriate icon for the drawer is set automatically at a particular position. The gesture to open the drawer is also set automatically. It is handled by the Scaffold.

bottomNavigationBar: bottomNavigationBar is like a menu at the bottom of the Scaffold. We have seen this navigationbar in most of the applications. We can add multiple icons or texts or both in the bar as items.

MaterialApp class

MaterialApp Class: MaterialApp is a predefined class or widget in a flutter.

It is likely the main or core component of a flutter app. The MaterialApp widget provides a wrapper around other Material Widgets. We can access all the other components and widgets provided by Flutter SDK. Textwidget, DropdownButton widget, AppBar widget, Scaffold widget, ListView widget, StatelessWidget, StatefulWidget, IconButton widget, TextField widget, Padding widget, ThemeData widget, etc. are the widgets that can be accessed using MaterialApp class.

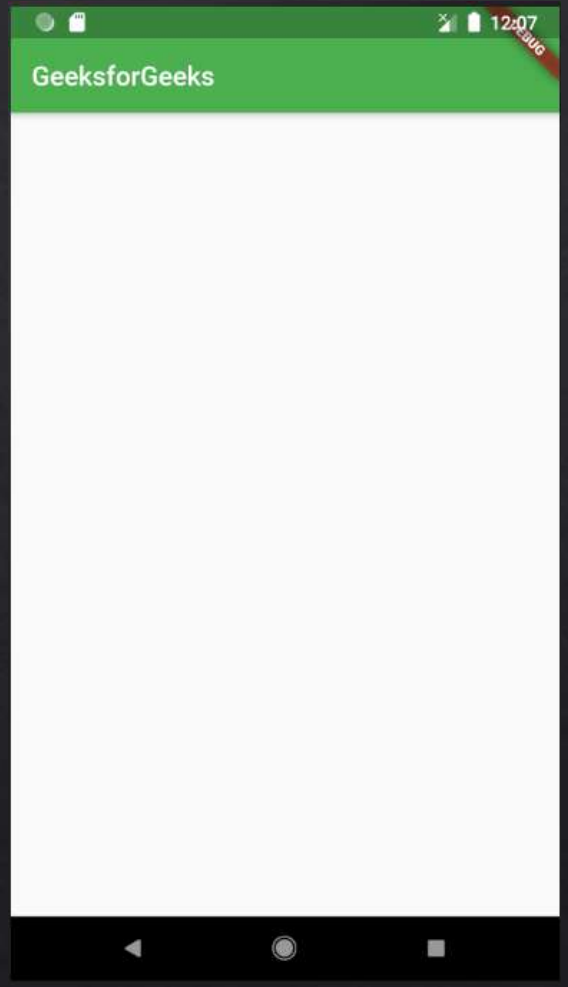
There are many more widgets that are accessed using MaterialApp class. Using this widget, we can make an attractive app that follows the Material Design guidelines.

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

void main() {
runApp(const GFGapp());
}

class GFGapp extends StatelessWidget {
const GFGapp({Key? key}) : super(key: key);

@override
Widget build(BuildContext context) {
        return MaterialApp(
        title: 'GeeksforGeeks',
        theme: ThemeData(primarySwatch: Colors.green),
        darkTheme: ThemeData(primarySwatch: Colors.grey),
        color: Colors.amberAccent,
        supportedLocales: {const Locale('en', ' ')},
        debugShowCheckedModeBanner: false,
        home: Scaffold(
                appBar: AppBar(title: const Text('GeeksforGeeks')),
        ),
        );
}
}
```



Drawer Widget

Drawer widget is used to provide access to different destinations and functionalities provided in your application.

Syntax:

```
Drawer({Key key, double elevation: 16.0, Widget child, String semanticLabel})
```

Properties:

child: The widgets below this widget in the tree.

hashCode: The hash code for this object.

key: Controls how one widget replaces another widget in the tree.

runtimeType: A representation of the runtime type of the object.

elevation: The z-coordinate at which to place this drawer relative to its parent.

semanticLabel: The semantic label of the dialogue used by accessibility frameworks to announce screen transitions when the drawer is opened and closed.

AppBar Widget

AppBar is usually the topmost component of the app (or sometimes the bottom-most), it contains the toolbar and some other common action buttons.

Constructor of AppBar class:

```
AppBar(
{Key key,
Widget leading,
bool automaticallyImplyLeading: true,
Widget title,
List<Widget> actions,
double elevation,
Color shadowColor,
ShapeBorder shape,
Color backgroundColor,
Brightness brightness,
IconThemeData iconTheme,
IconThemeData actionsIconTheme,
TextTheme textTheme,
...}
)
```

Key Properties of Appbar Widget:

actions: This property takes in a list of widgets as a parameter to be displayed after the title if the AppBar is a row.

title: This property usually takes in the main widget as a parameter to be displayed in the AppBar.

backgroundColor: This property is used to add colors to the background of the Appbar.

elevation: This property is used to set the z-coordinate at which to place this app bar relative to its parent.

shape: This property is used to give shape to the Appbar and manage its shadow.

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

void main() {
runApp(gfgApp()); //MaterialApp
}

MaterialApp gfgApp() {
return MaterialApp(
        home: Scaffold(
        appBar: AppBar(
                title: const Text('GeeksforGeeks'),
        ), //AppBar
        body: const Center(
                child: Text(
                'GeeksforGeeks',
                style: TextStyle(fontSize: 24),
                ), //Text
        ), // center
        ), //Scaffold
        debugShowCheckedModeBanner: false, //Removing
Debug Banner
);
}
```

RichText Widget

The RichText widget is used to display text that uses various different styles. The displayed text is described using a tree of TextSpan objects, each of which has its own associated style that is used for that subtree.

Syntax:

```
RichText(
{Key key,
@required InlineSpan text,
TextAlign textAlign: TextAlign.start,
TextDirection textDirection,
bool softWrap: true,
TextOverflow overflow:
TextOverflow.clip,
double textScaleFactor: 1.0,
int maxLines,
Locale locale,
StrutStyle strutStyle,
TextWidthBasis textWidthBasis: TextWidthBasis.parent,
TextHeightBehavior textHeightBehavior,
```

Properties:

children: The widgets below this widget in the tree.

hashCode: The hash code for this object.

key: Controls how one widget replaces another widget in the tree.

runtimeType: A representation of the runtime type of the object.

text: The text to display in this widget.

textAlign: How the text should be aligned horizontally.

local: This property takes in Locale class as the object. It controls the font used for the text depending on the language used.

maxLines: The maxLines property takes in an int value as the object. It controls the maximum number of lines that can be there for the text to expand and wrap.


Any
questions