Material App

Introduction:

- **MaterialApp** is Futter's one of the most powerful widgets. if you create a basic Flutter app then the first widget you'll see is **MaterialApp**
- **MaterialApp** wraps a number of widgets that are commonly required for material design applications.
- By wrapping your application inside the **MaterialApp**, you're telling your app to use **Android's Material Design**, which is a design system created by Google to help teams build high-quality digital experiences for Android, iOS, Flutter, and the web.

Introduction:

- But if you want to follow **iOS** design patterns, then you have to wrap your app inside **CupertinoApp**. There are many widgets provided by flutter to design your app for **iOS** platform.

MaterialApp

- We can consider this as an application that uses **material design**.
- Before creating **MaterialApp** we have to import **material package** which is provided by flutter SDK.

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

- This package provides us all the widgets that we can use in our application. For example: AppBar, Scaffold, BottomNavigationBar, Card, Chip, BottomSheet, etc.
- MaterialApp must have at least one of home, routes, onGenerateRoute, or builder properties non-null. Without it you will get an error.

Home

- This is a default route of an app.
- It means whatever is defined here is the first thing you will see on the screen.
- It takes Widget as an input.
- Usually, we define home, signIn, signUp, splash screens, but you can put any widget here.

```
MaterialApp(
   home: MyFirstPage(),
);
```

Title

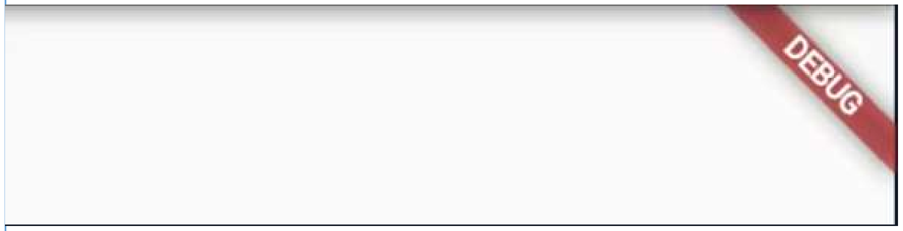
- This takes String as value.
- If you put value in title, you will not see any changes in your app. It will still show an empty blank screen.
- You will see this title when you press the "recent apps" button

```
MaterialApp(
  title: "Widget In Detail",
  home: MyFirstPage(),
);
```

debugShowCheckedModeBanner

- This is a banner that indicates that currently, our app is running in `debug mode.
- The default value of this property is true.
- To remove this banner, simply put false inside it.

```
MaterialApp(
   debugShowCheckedModeBanner: true,
   title: "Widget In Detail",
   home: MyFirstPage(),
);
```



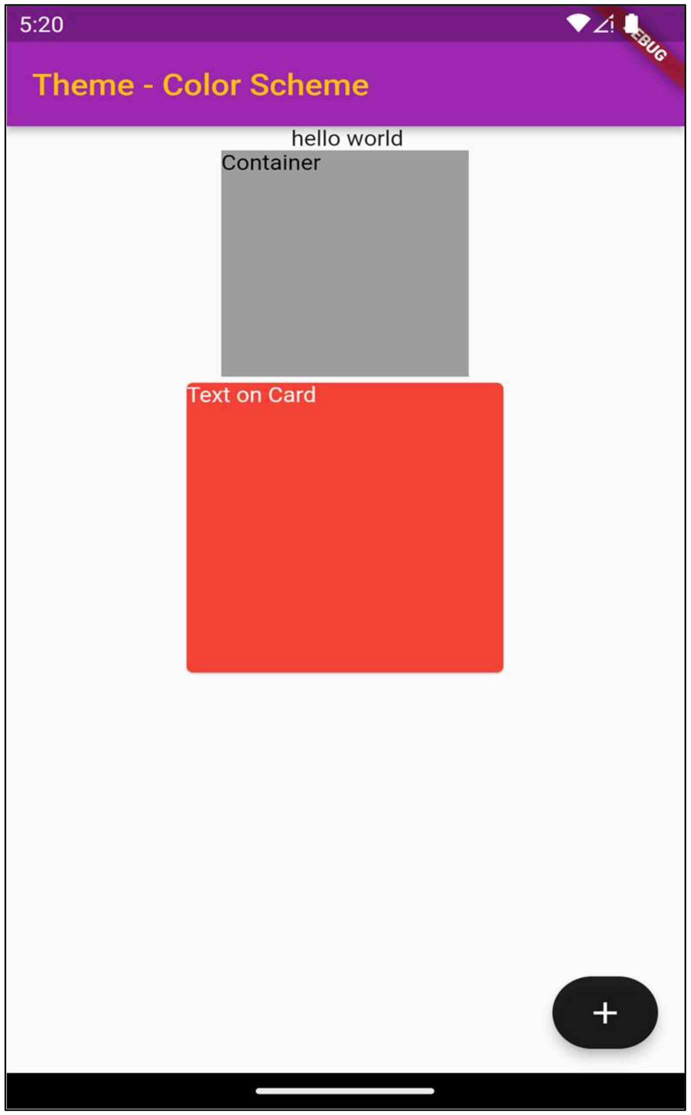
darkTheme

- By applying the ThemeData in the darkTheme property, we are telling our app to use this particular ThemeData when the system requests for DarkTheme.

```
MaterialApp(
     darkTheme: ThemeData.dark(),
   )
```

Themes:

- Themes are an integral part of UI for any application. Themes are used to design the fonts and colors of an application to make it more presentable. In Flutter, the Theme widget is used to add themes to an application



primary = Appbar color

onPrimary = Appbar Text and Icons

Background = background color

onBackground = text on background

Surface = Card Color

onSurface = Text on Card color

Secondary = floating action button

onSecondary = Icon on floating action button

Error = error

onError = error

```
    return MaterialApp(
theme: ThemeData(
        colorScheme: const ColorScheme(
            brightness: Brightness.light,) // .dark
            primary: Colors.purple,
            onPrimary: Colors.amber,
            secondary: Colors.black87,
            onSecondary: Colors.white,
            error: Colors.red,
            onError: Colors.red,
            background: Colors.blue,
            onBackground: Colors.black,
            surface: Colors.red,
            onSurface: Colors.white),
      ),
```

Apply a theme

- To apply your new theme, use the **Theme.of(context)** method when specifying a widget’s styling properties.

```
 Text(
              “Mobile Applications",
              style: TextStyle(
                color:
Theme.of(context).colorScheme.onBackground,
              ),
            ),
```

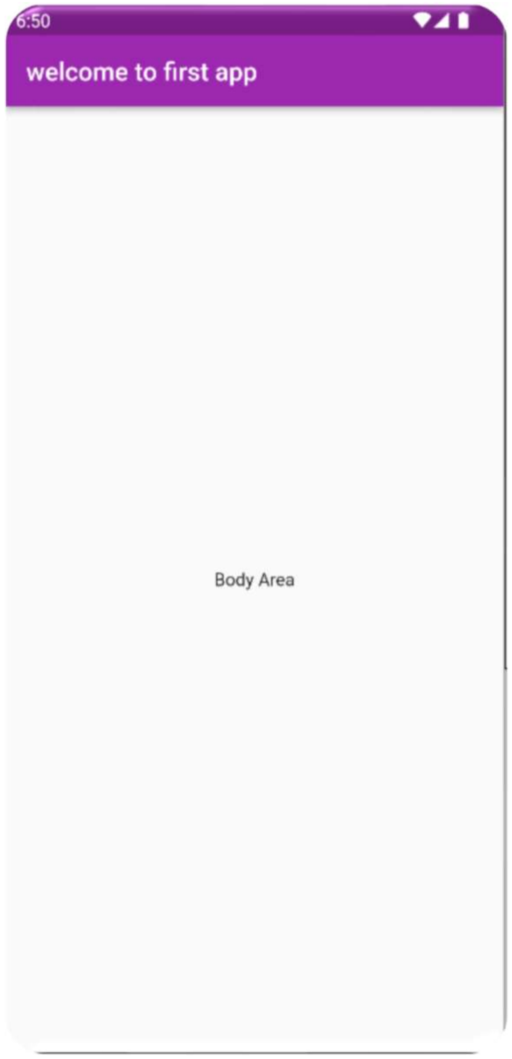
themeMode

- This property determines which theme will be used by the application if both theme and darkTheme are provided.
- The default value of themeMode is ThemeMode.system, which means whatever the theme of the system will be applied by default by our app.

```
MaterialApp(
   themeMode: ThemeMode.dark,
   theme: ThemeData(
     brightness: Brightness.light,
),
   darkTheme: ThemeData(
     brightness: Brightness.dark,
),
   home: HomePage(),
),
```

Example

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.purple,
      ),
      home: Scaffold(
        appBar: AppBar(
          title: const Text("welcome to first app"),
        ),
        body: const Center(child: Text("Body
Area")),
      ),
    ); }}
```



route

If you want to navigate via **namedRoutes**, you have to first define all the routes in the application's top-level routing table. i.e, in MaterialApp's routes property.

You can think of routes as a table where each screen is binded with a particular path. For example, "/home" is binded with HomeScreen() widget.

- The ***initialRoute*** property defines which route the app should start with.

route

HomePage()=
“ / “

Page1()=
“/Login_Screen “

Page2()=
“/Profile“

Page3()=
“/ Search“

Example

```
return MaterialApp(
    initialRoute: "/",
     routes: {
       "/": (context) => HomePage(),
       "/login_Screen": (context) => Page1(),
       "/Profile": (context) => Page2(),
       "/Search": (context) => Page3(),
     },
  );
```

route

- Now you can use **Navigator.pushNamed(context, "/login_Screen");** for navigation.

```
class HomePage extends StatelessWidget {
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: ElevatedButton(
        onPressed: () => Navigator.pushNamed(context, "/login_Screen"),
        child: Text('To Second Screen'),
      ),),);
}}
```

Flutter Navigation and Routing

- Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information.
- In Flutter, the screens and pages are known as **routes,**

Flutter Navigation and Routing

- In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as **routing.** Flutter provides a basic routing class **MaterialPageRoute** and two methods **Navigator.push()** and **Navigator.pop()** that shows how to navigate between two routes. The following steps are required to start navigation in your application

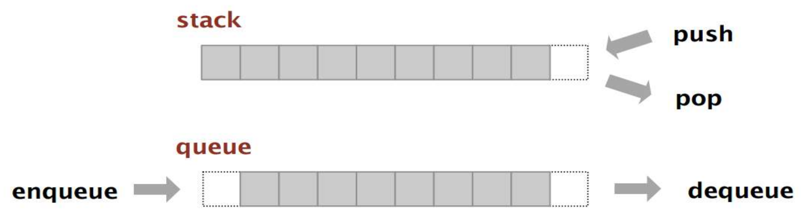
Step 1: First, you need to create two routes.

Step 2: Then, navigate to one route from another route by using the **Navigator.push()** method.

Step 3: Finally, navigate to the first route by using the **Navigator.pop()** method.

Navigator

- The **Navigator.push()** method is used to navigate/switch to a new route/page/screen. Here, the **push()** method adds a page/route on the stack and then manage it by using the **Navigator.**
- we need to use **Navigator.pop()** method to close the second route and return to the first route. The **pop()** method allows us to remove the current route from the stack, which is managed by the Navigator.

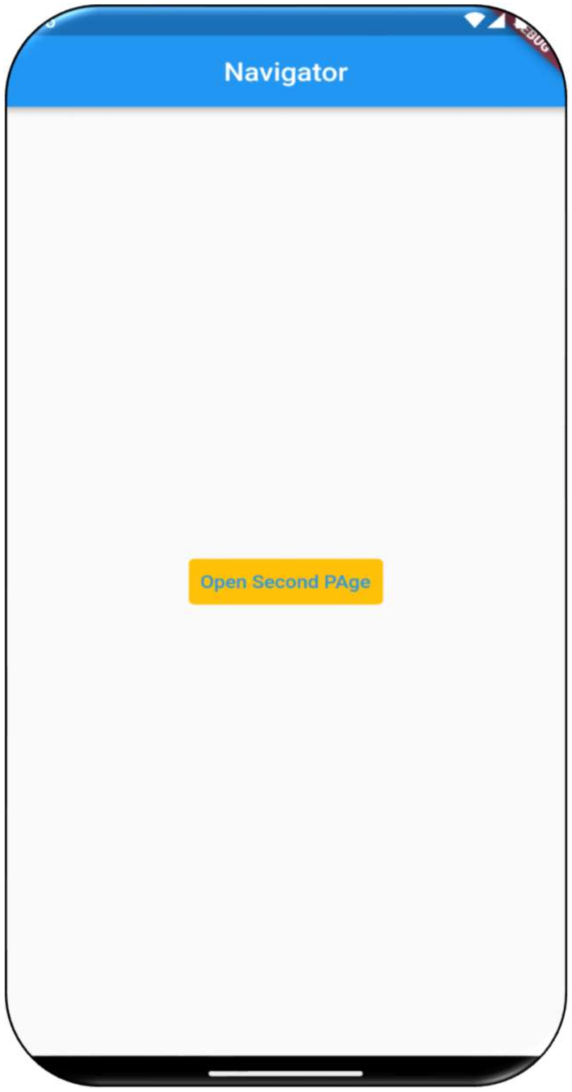


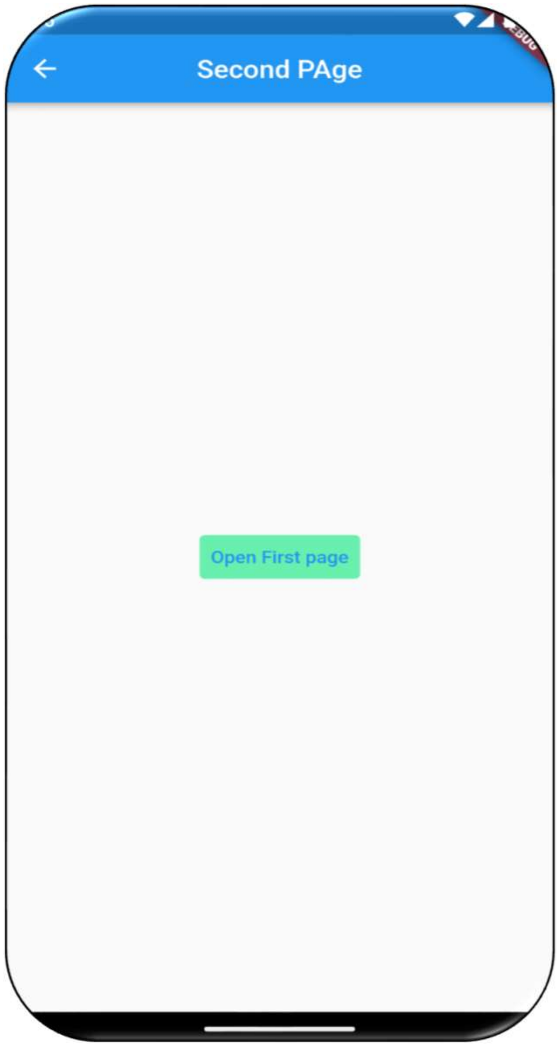
```
TextButton(
  onPressed: () {
  Navigator.push(
   context,
   MaterialPageRoute(
    builder: (context) => SecondPage(),
        ));
     },
 child: Text("Open Second PAge"),
 style: TextButton.styleFrom(backgroundColor: Colors.amber),
    ),
```

FirstPage.dart

```
TextButton(
     onPressed: () {
      Navigator.pop(context);
     },
     child: Text("Open First page"),
     style: TextButton.styleFrom(backgroundColor: Colors.greenAccent),
    ),
```

SecondPage.dart


Navigator
Open Second PAge


Second PAge
Open First page

Pass Data from Page1 to Page2

```dart
class _HomeScreenState extends State<HomeScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Navigator'),
      ),
      body: Center(
        child: ElevatedButton(
    onPressed: () {
    Navigator.push(
    context,
  MaterialPageRoute(
  builder: (context) => SecondScreen(
  text: "Data form page 1",
 ),
 ));
},
            child: Text("Click Me")),
      ),
    );
  }
```

```dart
class SecondScreen extends StatelessWidget
{
  final String text;
const SecondScreen(
{super.key, required this.text});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Seccond Screen'),
      ),
      body: Center(
        child: Text('Second Screen $text'),
      ),
    );
  }
}
```