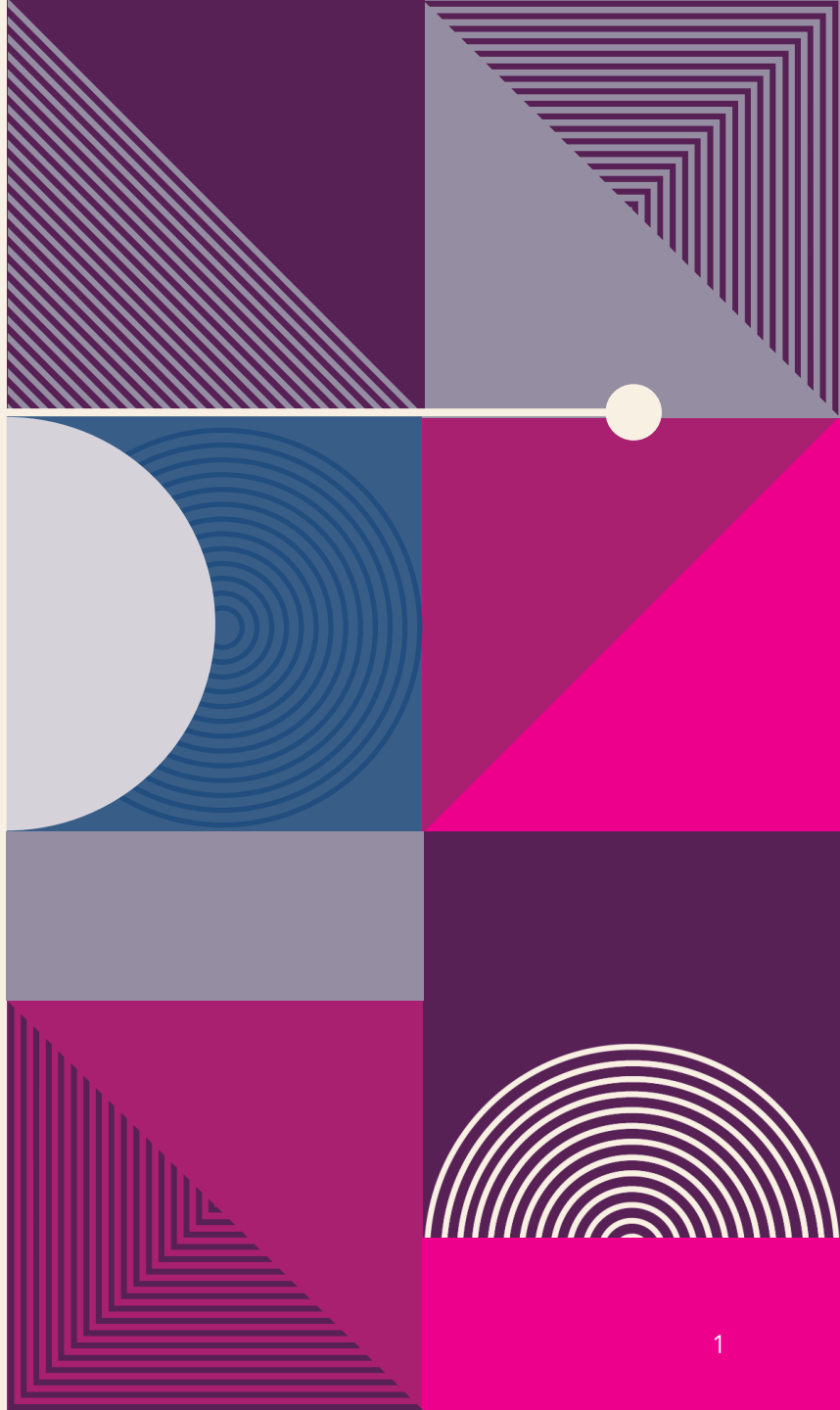
CHECKING ORIENTATION



CHECKING ORIENTATION

There are two ways to figure out orientation,

1. MediaQuery.of(context).orientation.
2. OrientationBuilder.

Creating the Orientation App

Create a new Flutter project and name it **flutter_orientation**. .

1. **Open** the **home.dart** file and

- **Add** to the **body** a **SafeArea** with **SingleChildScrollView** as a child.
- **Add Padding** as a child of the **SingleChildScrollView**.
- **Add** a **Column** as a child of the **Padding**.
- In the **Column** children property, **add** the widget class called **OrientationLayoutIconsWidget**(), which you will create next.
- Make sure you add the **const** keyword before the widget class name to take advantage of caching to improve performance.

```
body: SafeArea(
  child: SingleChildScrollView(
    child: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        children: <Widget>[
          const OrientationLayoutIconsWidget(),
],  ),
    ),
  ),
),
```

2. Add the **OrientationLayoutIconsWidget**() widget **class** after **class Home extends StatelessWidget {...}**.

```
class OrientationLayoutIconsWidget extends StatelessWidget {
  const OrientationLayoutIconsWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Orientation _orientation =
MediaQuery.of(context).orientation;
    return Container();
  }
}
```

3. Based on the current **Orientation**, you return a different layout of **Icon** widgets.

- **Use** a **ternary operator** to check whether **Orientation** is **portrait**, and if so, return a single **Row** icon.
- If **Orientation** is **landscape**, return a **Row** of **two Icon** widgets. **Replace** the current return **Container**() with the following code:

```
class OrientationLayoutIconsWidget extends StatelessWidget {
  const OrientationLayoutIconsWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Orientation _orientation = MediaQuery.of(context).orientation;
        return _orientation == Orientation.portrait
        ? Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Icon(
          Icons.school,
          size: 48.0,
        ),
      ],
    )
        : Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Icon(
          Icons.school,
          size: 48.0,
        ),
        Icon(
          Icons.brush,
          size: 48.0,     ),        ],
    );
  }
}
```

4. After **OrientationLayoutIconsWidget**(), add a **Divider** widget and the **OrientationLayoutWidget**() widget **class** to **create**.

```
body: SafeArea(
  child: SingleChildScrollView(
    child: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        children: <Widget>[
          const
OrientationLayoutIconsWidget(),
          Divider(),
          const OrientationLayoutWidget(),
        ],
      ),
    ),
  ),
),
```

5. Create OrientationLayoutWidget() widget class.

```
class OrientationLayoutWidget extends StatelessWidget {
  const OrientationLayoutWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Orientation _orientation = MediaQuery.of(context).orientation;

    return _orientation == Orientation.portrait
        ? Container(
      alignment: Alignment.center,
      color: Colors.yellow,
      height: 100.0,
      width: 100.0,
      child: Text('Portrait'),
    )
        : Container(
      alignment: Alignment.center,
      color: Colors.lightGreen,
      height: 100.0,
      width: 200.0,
      child: Text('Landscape'),    );
  }
}
```

6. After **OrientationLayoutWidget**(), add a **Divider** widget and the **GridViewWidget**() widget **class** that you will **create**.

```
body: SafeArea(
  child: SingleChildScrollView(
    child: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        children: <Widget>[
          const OrientationLayoutIconsWidget(),
          Divider(),
          const OrientationLayoutWidget(),
          Divider(),
          const GridViewWidget(),
        ],
      ),
    ),
  ),
),
```

7. Create GridViewWidget() widget class.

```
class GridViewWidget extends StatelessWidget {
  const GridViewWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Orientation _orientation = MediaQuery.of(context).orientation;

    return GridView.count(
      shrinkWrap: true,
      physics: NeverScrollableScrollPhysics(),
      crossAxisCount: _orientation == Orientation.portrait ? 2 : 4,
      childAspectRatio: 5.0,
      children: List.generate(8, (int index) {
        return Text("Grid $index", textAlign: TextAlign.center,);
      }),
    );
  }
}
```

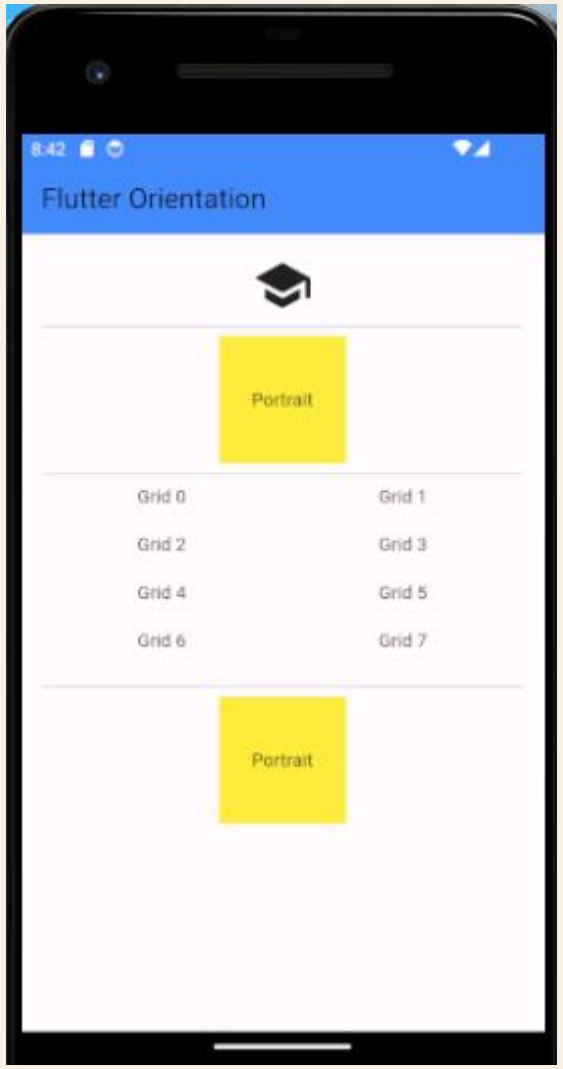
8. After **GridViewWidget**(), add a **Divider** widget and the **OrientationBuilderWidget**() widget **class** that you will create.

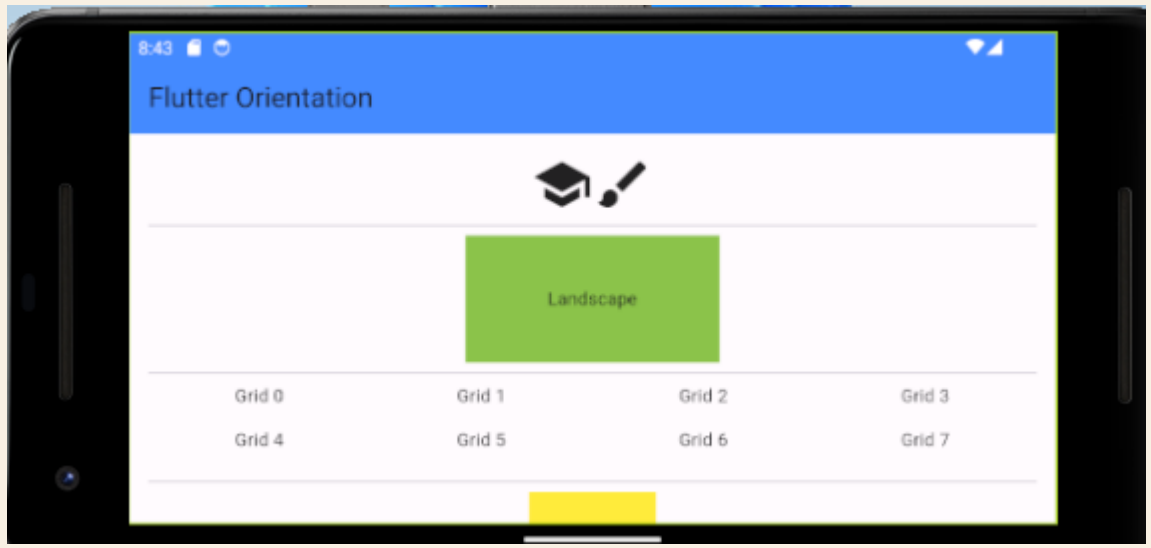
```
body: SafeArea(
  child: SingleChildScrollView(
    child: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        children: <Widget>[
          const OrientationLayoutIconsWidget(),
          Divider(),
          const OrientationLayoutWidget(),
          Divider(),
          const GridViewWidget(),
          Divider(),
          const OrientationBuilderWidget(),
        ],
      ),
    ),
  ),
),
```

9. Create OrientationBuilderWidget() widget **class**.

```
// OrientationBuilder as a child does not give correct Orientation. i.e Child of Column...
// OrientationBuilder as a parent gives correct Orientation
class OrientationBuilderWidget extends StatelessWidget {
  const OrientationBuilderWidget({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return OrientationBuilder(
      builder: (BuildContext context, Orientation orientation) {
        return orientation == Orientation.portrait
            ? Container(
          alignment: Alignment.center,
          color: Colors.yellow,
          height: 100.0,
          width: 100.0,
          child: Text('Portrait'),
        )
            : Container(
          alignment: Alignment.center,
          color: Colors.lightGreen,
          height: 100.0,
          width: 200.0,
          child: Text('Landscape'),
        );
      },
    );
  }
}
```


8:42
Flutter Orientation
Portrait
Grid 0
Grid 1
Grid 2
Grid 3
Grid 4
Grid 5
Grid 6
Grid 7
Portrait


8:43
Flutter Orientation
Landscape
Grid 0
Grid 1
Grid 2
Grid 3
Grid 4
Grid 5
Grid 6
Grid 7

Flutter Navigation and Routing

- Navigation and routing are fundamental components of any mobile application, enabling users to switch between various pages. Mobile apps typically comprise multiple screens that display diverse information.

- For example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product.

Routes and routing

- In Flutter, the screens and pages are known as routes, and these routes are just a widget.

In Android, a route is similar to an Activity, whereas, in iOS, it is equivalent to a ViewController.

- 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.

Navigator.push() and Navigator.pop()

- Flutter provides a basic routing class MaterialPageRoute and two methods

– Navigator.push()

– Navigator.pop()

- The Code attached to this lecture shows the how these methods are used.

Navigate between two routes

- The following steps show how to navigate between two routes:-
- Create two routes.
- Navigate to the second route using Navigator.push().
- Return to the first route using Navigator.pop().

There are multiple options for routing. Some create a lot of clutter, others cannot facilitate passing data between routes, and yet others require that you set up a third-party library.

Initial setup

Before we can do routing the right way, we first need to have some pages to navigate between.

There are 2 pages and the second page receives data from the first one. We push MaterialPageRoutes directly to the navigator. The more pages your app has, the worse it gets, and it's easy to get lost in all these routes specified all over the place

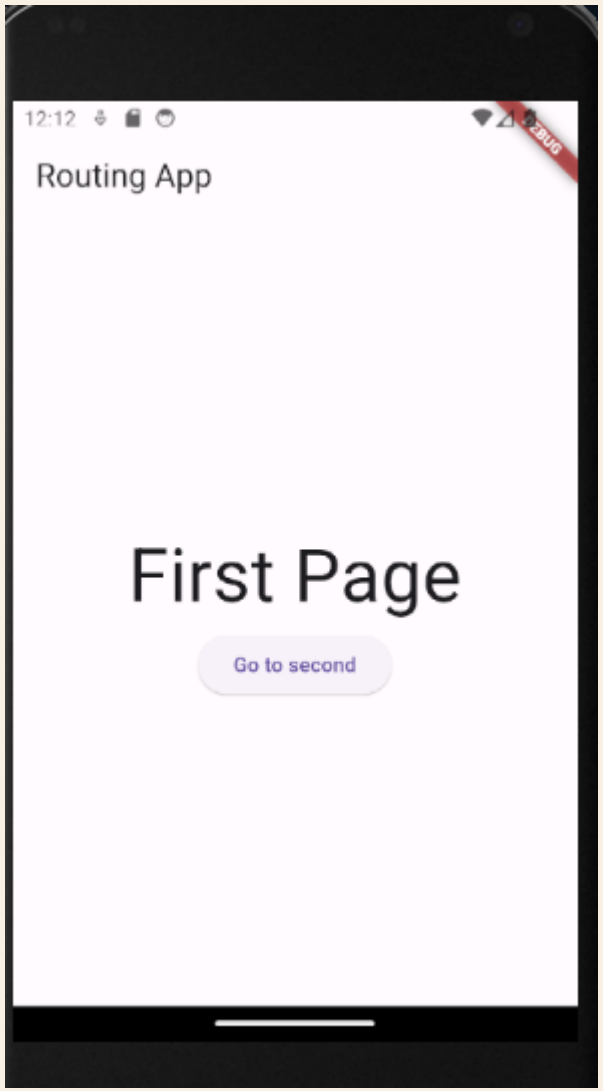
main.dart

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      // Initially display FirstPage
      home: FirstPage(),
    );
  }
}
```

```
class FirstPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Routing App'),      ),
      body: Center(
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            Text(
              'First Page',
              style: TextStyle(fontSize: 50),           ),
            RaisedButton(
              child: Text('Go to second'),
              onPressed: () {
                // Pushing a route directly, WITHOUT using a named route
                Navigator.of(context).push(
                  // With MaterialPageRoute, you can pass data between pages,
                  // but if you have a more complex app, you will quickly get lost.
                  MaterialPageRoute(
                    builder: (context) =>
                        SecondPage(data: 'Hello there from the first page!'),  ),  );  },  )  ],  ),  ), );} }
```



```
class SecondPage extends StatelessWidget {
final String data;
  SecondPage({     Key key,
    @required this.data,  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Routing App'),     ),
      body: Center(
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: <Widget>[
            Text(
              'Second Page',
              style: TextStyle(fontSize: 50),         ),
            Text(
              data,
              style: TextStyle(fontSize: 20),      ),  ],  ),   ),     );
  }
}
```