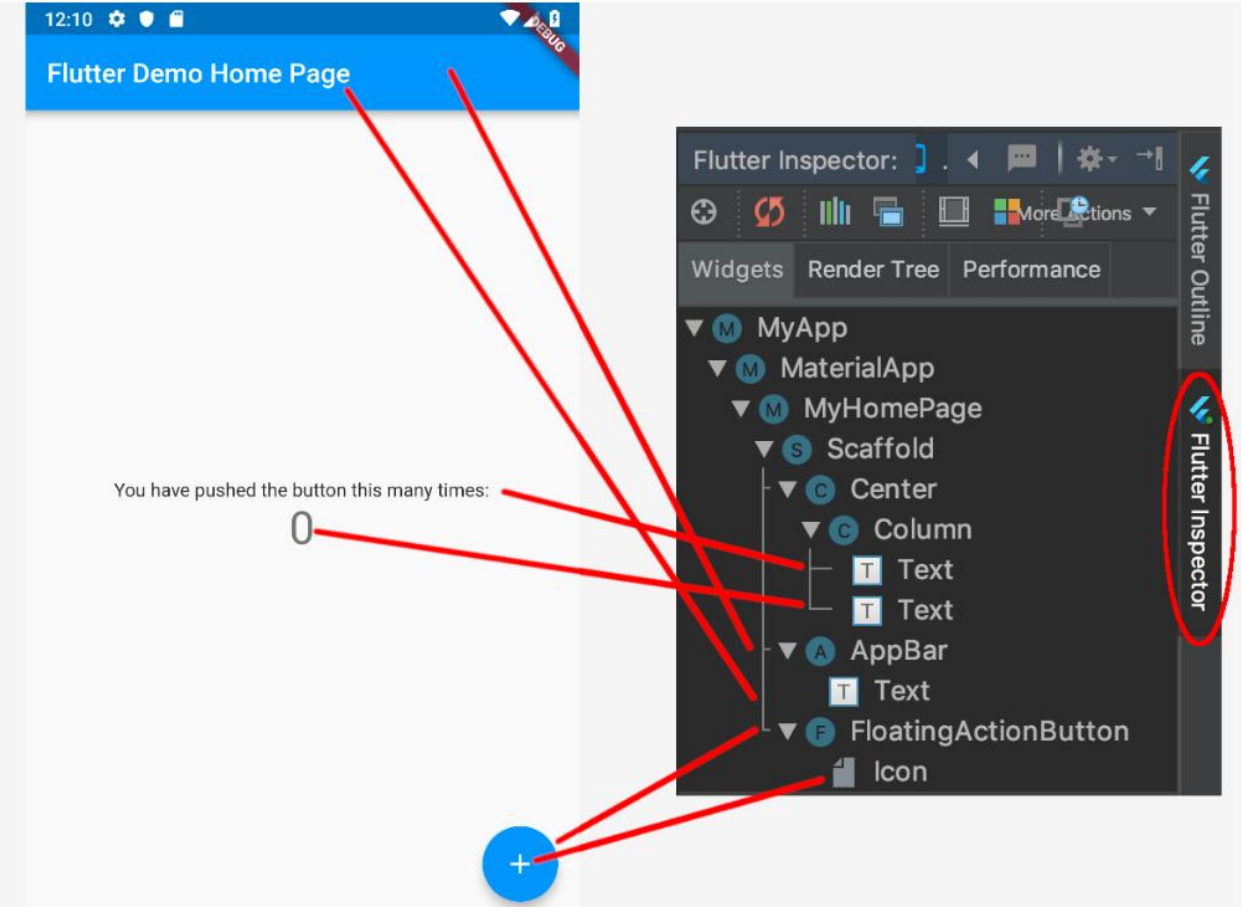
Flutter

TY_OET_Odd2022-23

1. Widgets


12:10
DEBUG
Flutter Demo Home Page
You have pushed the button this many times:
0
Flutter Inspector:
More Actions
Widgets
Render Tree
Performance
MyApp
MaterialApp
MyHomePage
Scaffold
Center
Column
Text
Text
AppBar
Text
FloatingActionButton
Icon
Flutter Outline
Flutter Inspector

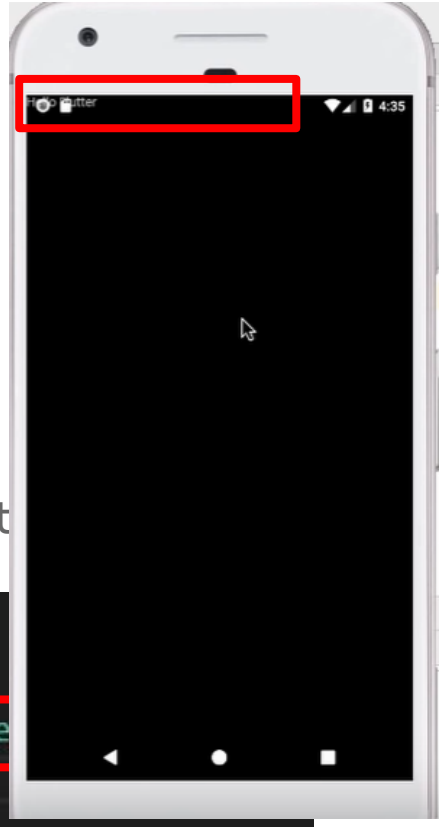
1. First app

1. Ctrl+shift+p ---> create flutter project
2. Delete widget_test.dart inside test
3. Delete everything from main.dart
4. **main.dart-**

   4.1 Import material dart package which contains allt
   design widgets

   4.2 Def

```
import 'package:flutter/material.dart';
void main(){
  runApp(Text("Hello world",textDirection: TextDire
}
```

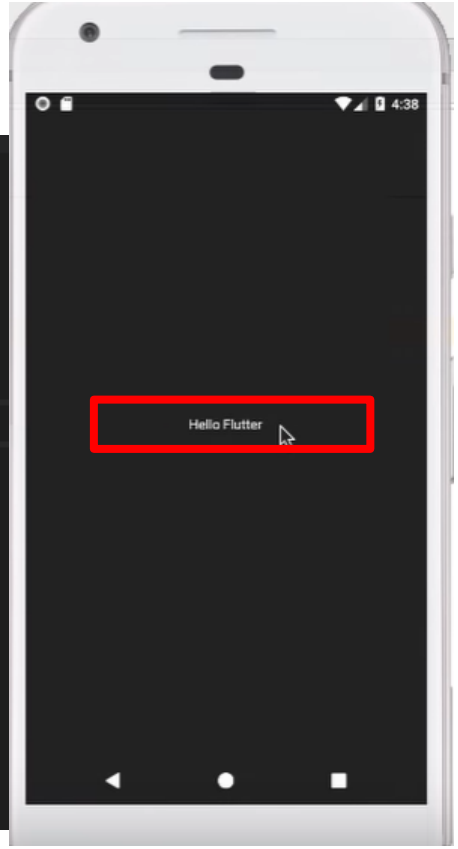


1. First app

main.dart > main

```
import 'package:flutter/material.dart';
void main(){
  runApp(
    Center(
      child: Text(
      "Hello world",
      textDirection: TextDirection.ltr
      ), // Text
      ) // Center

      );
}
```



1. First app

```
void main() {
  runApp(
    Center(
      child: Text(
        "Hello Flutter",
        textDirection: TextDirection:ltr
      )
    );
  )
}
```

← main() : Entry point of application

← runApp(): Inflates the widget and show it on app screen

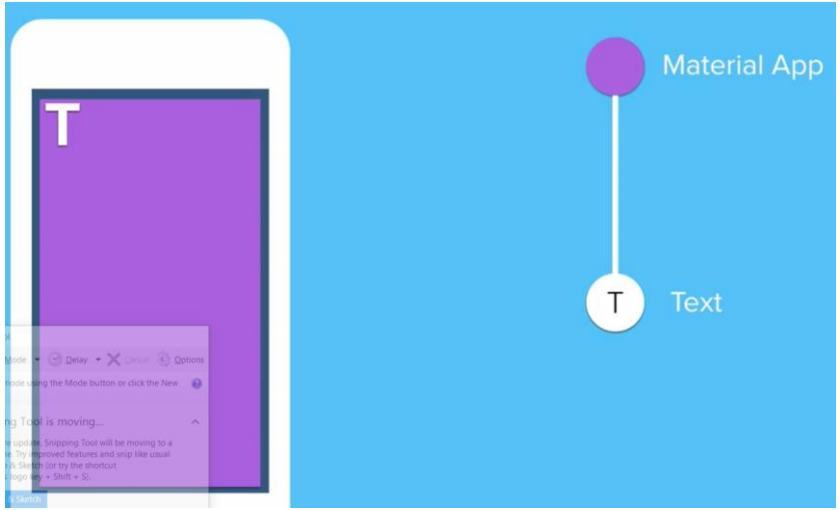
← Center and Text are widgets

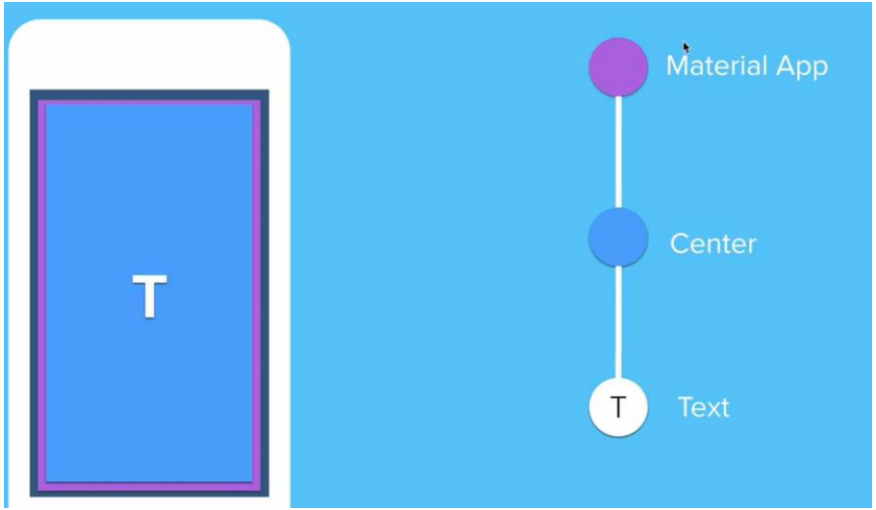
1. Widgets

1. MaterialApp
2. Container
3. Text
4. Scaffold

Github link – https://github.com/meanjali/All-About-Flutter/tree/master/Flutter%20Projects/LEc%201/app1

1. Material App





Material APP

Properties

1. Title
2. Theme
3. Home
4. debugShowCheckedModeBanner

Scaffold Widget - https://api.flutter.dev/flutter/material/Scaffold-class.html

Functionalities

- appbar,
- a floating button,
- a drawer,
- background color,
- bottom navigation bar,
- footer buttons,
- body.

```
//Constructor of Scaffold


const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```

Widgets

3. Scaffold

main.dart > main

```dart
import 'package:flutter/material.dart';
void main(){
  runApp(

MaterialApp(
  title: "My app",
  home:Scaffold(
    appBar: AppBar(title: Text("My First App Screen"),),
    body: Material(
    color: Colors.lightBlueAccent,
    child: Center(
        child: Text(
            "Hello world",
            textDirection: TextDirection.ltr,
            style: TextStyle(color: Colors.white,fontSize: 40.0),
                ), // Text
              ) // Center
              ), // Material
  ) // Scaffold
                ) // MaterialApp

            );
}
```

Widget - Container

Single chi

FLUTTER: OU...

main () → void
- MaterialApp
  - Scaffold
    - SafeArea
      - Container
        - Text "I a...

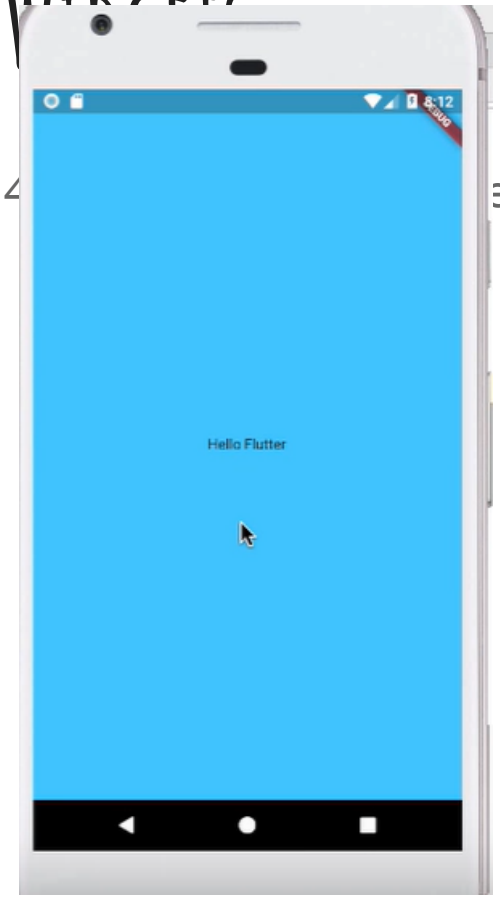
main.dart | pubspec.yaml

lib > main.dart > main

```dart
Run | Debug
void main() {
  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        backgroundColor: Colors.teal,
        body: SafeArea(
          child: Container(
            color: Colors.white,
            child: Text("I am container"),
            height: 200.0,
            width: 100.0,
            //margin: EdgeInsets.all(20.0),
            //margin: EdgeInsets.symmetric(vertical: 10.0,horizontal: 30.0),
            //margin: EdgeInsets.fromLTRB(30.0, 10.0, 50.0, 20.0),
            margin: EdgeInsets.only(left: 30.0),
            padding: EdgeInsets.all(20.0),
          ), // Container
        ), // SafeArea
      ), // Scaffold
    ), // MaterialApp
  );
}
```

Widget - Column and Row

Multi child widget




```
import 'package:flutter/material.dart';
void main(){
  runApp(

MaterialApp(
  title: "My app",
  home: Material(
    color: Colors.lightBlueAccent,
    child: Center(
        child: Text(
            "Hello world",
            textDirection: TextDirection.ltr
                ), // Text
              ) // Center
              ) // Material
            ) // MaterialApp


          );
}
```

Widgets

main.dart > main

```
import 'package:flutter/material.dart';
void main(){
  runApp(

MaterialApp(
  title: "My app",
  home: Material(
    color: Colors.lightBlueAccent,
    child: Center(
        child: Text(
            "Hello world",
            textDirection: TextDirection.ltr,
            style: TextStyle(color: Colors.white,fontSize: 40.0),
                ), // Text
              ) // Center
              ) // Material
            ) // MaterialApp

          );
}
```

1. Widgets

1. Button
    - Floating Action Button
    - Popup Menu Button
    - Flat Button
    - Icon Button
    - Drop Down Button
    - Button Bar

2. Text field

3. List view

1. Widgets

1. Types
- Stateful
- Stateless

Stateless widget

```
class MyApp extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    return OneOrMoreWidgets;
  }
}
```

Stateful widget

Create a class that extends a “StatefulWidget”, that returns a State in “createState()”

Create a “State” class, with properties that may change

Within “State” class, implement the “build()” method

Call the setState() to make the changes. Calling setState() tells framework to redraw widget

Custom Fonts and Images

Fonts

1. Download and import font file ----> fonts.google.com
2. Declare it in pubspec.yaml.
3. USe in Text widget.

List

Basic **List View**

1. List Tile - for few list items
2. They are scrollable in nature, wrap them in Scaffold widget, otherwise app will throw Runtime exception.

List

Long List

Dynamic Contents

ListView.builder

Parameters

1. `itemCount` - how many times the callback function in `itemBuilder` will be called.
2. `itemBuilder` -

Parsing JSON data - JSON structure #1 : Simple map

Rule #1 : Identify the structure.

Json strings will either have a **Map (key-value pairs)** or a **List of Maps**.

Rule #2 : Begins with curly braces? **It's a map.**

Begins with a Square bracket? **That's a List of maps.**

Parsing JSON data

It is a map. (E.g like, `id` is a key, and `487349` is the value for `id`)

```
{
  "id":"487349",
  "name":"Pooja Bhaumik",
  "score" : 1000
}
```

student.json

Parsing JSON data

2. PODO (Plain Old Dart Object) file for this json structure

```
class Student{
  String studentId;
  String studentName;
  int studentScores;

  Student({
    this.studentId,
    this.studentName,
    this.studentScores
 });


}
```

Parsing JSON data

3.Mapping class members to the json object.

```
Student.fromJson(Map<String, dynamic> parsedJson):
        studentId= parsedJson['id'],
        studentName = parsedJson['name'],
        studentScores = parsedJson ['score'];

}
```

- Maps string key to dynamic value.

```
{
  "name": "Pooja",
  "majors":["CS", "Maths"],
  "subjects": [
    {
      "subjectName": "math",
      "teacher":"Ms S"
    },
    {
      "subjectName": "science",
      "teacher":"Ms P"
    }
  ]
}
```

*key is always a string, value can be anything(for eg. string, list of string, list<object>)

Parsing JSON data

4.

Deserializing Convert the string to a data of primitive data type

Serialization Convert the data to a string

Parsing JSON data

5. call fromJson and retrieve the values from the object

5.1 Snippet #1 : imports

5.2 Snippet #2 : load Json Asset (optional)

5.3 Snippet #3 : load the response

Parsing JSON data - JSON structure #1 : Simple structure with array

Rule #1 : Identify the structure.

Json strings will either have a **Map (key-value pairs)** or a **List of Maps**.

Rule #2 : Begins with curly braces? **It's a map.**

Begins with a Square bracket? **That's a List of maps.**

Parsing JSON data

It is a map. (E.g like, `city` is a key, and `Mumbai` is the value for `city`)

```
{
  "city": "Mumbai",
  "streets": [
    "address1",
    "address2"
  ]
}
```

address.json

Parsing JSON data

address_model.dart

```
class Address {
    final String city;
    final List<String> streets;


    Address({
      this.city,
      this.streets
    });
  }
```

Address.fromJson ----> Map<String, dynamic>

```
Address.fromJson(Map<String, dynamic>
parsedJson) :


      city= parsedJson['city'],
     streets= parsedJson['streets'];
```

Parsing JSON data - #2 Simple nested structure

```
{
  "shape_name":"rectangle",
  "property":{
    "width":5.0,
    "breadth":10.0
  }
}
```

It is a Map.

Property is a object.

Parsing JSON data - #2 Simple nested structure

Model – constructor

```
class Property{
  double width;
  double breadth;

  Property({
    this.width,
    this.breadth
});
}
```

```
class Shape{
  String shapeName;
  Property property;

  Shape({
    this.shapeName,
    this.property
  });
}
```

Parsing JSON data - #2 Simple nested structure

Model – mapping with json data

```
factory Property.fromJson(Map<String, dynamic> json){
  return Property(
    width: json['width'],
    breadth: json['breadth']
  );
}
```

```
factory Shape.fromJson(Map<String, dynamic> parsedJson){
  return Shape(
    shapeName: parsedJson['shape_name'],
    property: Property.fromJson(parsedJson['property'])
  );
}
```

Parsing JSON data - #2 Simple nested structure

Services

```
Future<String> _loadShapeAsset() async {
  return await rootBundle.loadString('assets/shape.json');
}


Future loadShape() async {
  String jsonString = await _loadShapeAsset();
  final jsonResponse = json.decode(jsonString);
  Shape shape = new Shape.fromJson(jsonResponse);
  print(shape.property.breadth);
}
```

Parsing JSON data

Command Prompt

```
C:\Flutter Projects>cd Stateless-Widget

C:\Flutter Projects\Stateless-Widget>cd "json_serializaton - Copy"

C:\Flutter Projects\Stateless-Widget\json_serializaton - Copy>flutter packages pub run build_runner build
[INFO] Generating build script...
[INFO] Generating build script completed, took 778ms

[INFO] Creating build script snapshot......
[INFO] Creating build script snapshot... completed, took 27.6s

[INFO] Initializing inputs
[INFO] Building new asset graph...
[INFO] Building new asset graph completed, took 2.3s

[INFO] Checking for unexpected pre-existing outputs....
[INFO] Checking for unexpected pre-existing outputs. completed, took 6ms

[INFO] Running build...
[INFO] Generating SDK summary...
[INFO] 7.9s elapsed, 0/4 actions completed.
[INFO] Generating SDK summary completed, took 7.9s

[INFO] 9.1s elapsed, 0/4 actions completed.
[INFO] 10.1s elapsed, 0/4 actions completed.
[INFO] 11.2s elapsed, 1/4 actions completed.
[INFO] 12.2s elapsed, 1/4 actions completed.
[INFO] 25.3s elapsed, 2/4 actions completed.
[INFO] 26.9s elapsed, 4/4 actions completed.
[INFO] Running build completed, took 27.1s
```

Async, await and future builder

```
 await Future.delayed(Duration(seconds: 3));
```

```
bool isLoading=true;

body:isLoading ? Center(child:CircularProgressIndicator(),

: ListView …….
```

As loading of data is done change state of this variable to false

Using Camera in flutter

File structure

camera_screen.dart is the camera preview screen where you can click a picture and toggle between front or back camera,

preview_screen.dart is the screen where you will see the preview of the image you clicked and will have the option to share that image with your friends.

main.dart is the root widget of your app.

Using Camera in flutter

Dependencies

1. **camera**: A Flutter plugin for iOS and Android allowing access to the device cameras.
2. **path_provider**: A Flutter plugin for finding commonly used locations on the filesystem. Supports both iOS and Android.
3. **path**: A comprehensive, cross-platform path manipulation library for Dart.
4. **esys_flutter_share**: A Flutter plugin for sharing files and text with other applications.

Using Camera in flutter

main.dart

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

class CameraApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CameraScreen(),
    );
  }
}

void main() => runApp(CameraApp());
```

Using Camera in flutter

camera_screen.dart

```
class _CameraScreenState extends State {
  CameraController controller;
  List cameras;
  int selectedCameraIdx;
  String imagePath;
```

1. `CameraController controller`
2. `List cameras`
3. `selectedCameraIdx`
4. `imagePath`

Using Camera in flutter

camera_screen.dart

```
@override
void initState() {
  super.initState();
  // 1
  availableCameras().then((availableCameras) {

    cameras = availableCameras;
    if (cameras.length > 0) {
      setState(() {
        // 2
        selectedCameraIdx = 0;
      });

      _initCameraController(cameras[selectedCameraIdx]).then((void v) {});
    }else{
      print("No camera available");
    }
  }).catchError((err) {
    // 3
    print('Error: $err.code\nError Message: $err.message');
  });
}
```

1. `availableCameras()`
2. `selectedCameraIdx`
3. `catchError()`

Using Camera in flutter

camera_screen.dart

```dart
// 1, 2
Future _initCameraController(CameraDescription cameraDescription) async {
  if (controller != null) {
    await controller.dispose();
  }

  // 3
  controller = CameraController(cameraDescription, ResolutionPreset.high);

  // If the controller is updated then update the UI.
  // 4
  controller.addListener(() {
    // 5
    if (mounted) {
      setState(() {});
    }

    if (controller.value.hasError) {
      print('Camera error ${controller.value.errorDescription}');
    }
  });

  // 6
  try {
    await controller.initialize();
  } on CameraException catch (e) {
    _showCameraException(e);
  }

  if (mounted) {
    setState(() {});
  }
}
```

1. `_initCameraController`
2. `CameraDescription`
3. `CameraController`
4. `addListener()`
5. `mounted`
6. `try/catch`

Using Camera in flutter

camera_screen.dart

```
Widget _cameraPreviewWidget() {
  if (controller == null || !controller.value.isInitialized) {
    return const Text(
      'Loading',
      style: TextStyle(
        color: Colors.white,
        fontSize: 20.0,
        fontWeight: FontWeight.w900,
      ),
    );
  }

  return AspectRatio(
      aspectRatio: controller.value.aspectRatio,
      child: CameraPreview(controller),
    );
}
```

Using Camera in flutter

camera_screen.dart

```
Widget _cameraTogglesRowWidget() {
  if (cameras == null || cameras.isEmpty) {
    return Spacer();
  }

  CameraDescription selectedCamera = cameras[selectedCameraIdx];
  CameraLensDirection lensDirection = selectedCamera.lensDirection;

  return Expanded(
    child: Align(
      alignment: Alignment.centerLeft,
      child: FlatButton.icon(
          onPressed: _onSwitchCamera,
          icon: Icon(_getCameraLensIcon(lensDirection)),
          label: Text(
              "${lensDirection.toString().substring(lensDirection.toString().indexOf('.') +
1)}")),
    ),
  );
}
```

Using Camera in flutter

camera_screen.dart

```
IconData _getCameraLensIcon(CameraLensDirection direction) {
  switch (direction) {
    case CameraLensDirection.back:
      return Icons.camera_rear;
    case CameraLensDirection.front:
      return Icons.camera_front;
    case CameraLensDirection.external:
      return Icons.camera;
    default:
      return Icons.device_unknown;
  }
}
```

Using Camera in flutter

camera_screen.dart

```
void _onSwitchCamera() {
  selectedCameraIdx =
  selectedCameraIdx < cameras.length - 1 ? selectedCameraIdx + 1 : 0;
  CameraDescription selectedCamera = cameras[selectedCameraIdx];
  _initCameraController(selectedCamera);
}
```

Using Camera in flutter

camera_screen.dart

```
void _onCapturePressed(context) async {
  try {
    // 1
    final path = join(
      (await getTemporaryDirectory()).path,
      '${DateTime.now()}.png',
    );
    // 2
    await controller.takePicture(path);
    // 3
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => PreviewImageScreen(imagePath: path),
      ),
    );
  } catch (e) {
    print(e);
  }
}
```

1. `join(..)`
2. `takePicture(path)`
3. `PreviewImageScreen`.

Using Camera in flutter

preview_screen.dart

```
Future<ByteData> getBytesFromFile() async {
  Uint8List bytes = File(widget.imagePath).readAsBytesSync() as Uint8List;
  return ByteData.view(bytes.buffer);
}
```

```
onPressed: () {
  getBytesFromFile().then((bytes) {
    Share.file('Share via:', basename(widget.imagePath),
        bytes.buffer.asUint8List(), 'image/png');
  });
},
```

STATE MANAGEMENT

1. setState
2. InheritedWidget & InheritedModel
3. Provider & Scoped Model
4. Redux
5. BLoC / Rx
6. MobX

CONSTRAINTS

- Allow steady development velocity without sacrificing code quality
- Separate presentation logic from business logic
- Easy to understand; hard to break.
- Predictable and widely adopted

Inherited Widget

A widget that **provides** data to be inherited (or simply accessed) by other widgets down the widget tree

Navigation

1. MaterialPageRoute
2. Named Routes

a. Specify a map of routes
b. Make a function returning routes