SLIIT
Discover Your Future

Flutter

Interactivity

Flutter

Adding Interactivity

Recap

- Widgets
- Stateless and State full widgets
- Single child widgets and multiple child widgets
- Image, List view , Grid view ,Stack
- Hot restart and hot reload

Today

Overview

State and State full Widgets

Events

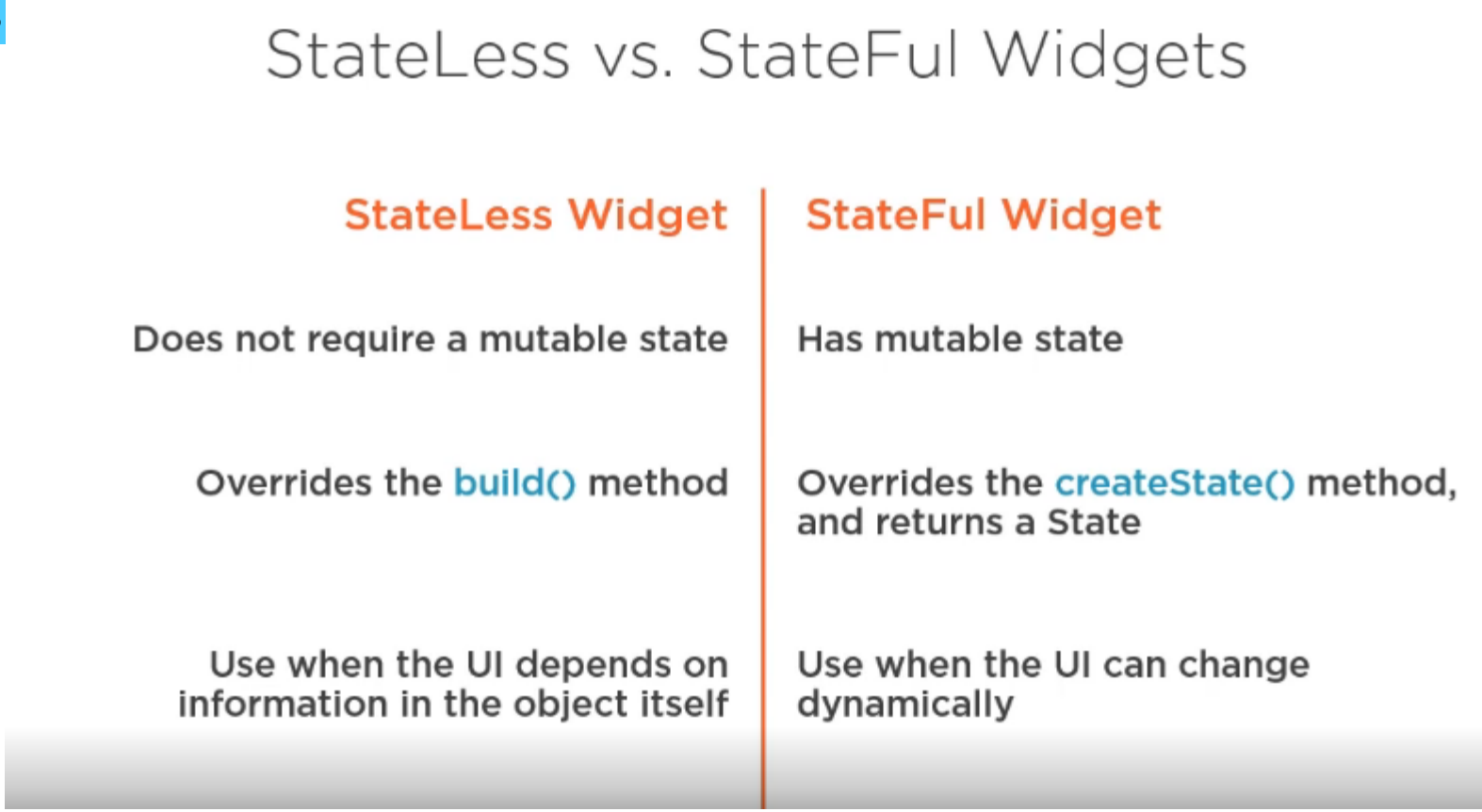
- OnChanged
- OnSubmitted
- Text Widget and TextEditingController
- Widgets
- Dropdown buttons , Dropdownitems

Demo :

- Hello Dilani
- Fuel Consumption Calculator

State

State is information that can be read synchronously when the widget is build and might change during the life time of the widget .



SLIIT
Discover Your Future

Using State Full widgets..

Create a Class that Extends a Stateful Widget, that returns a State

Create a State class, with properties that may change

Implement the Build() method

Call the setState() method to make changes

State Full Widgets in Action

```
class HelloInput extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => _HelloInputState();
}
class _HelloInputState extends State<HelloInput> {
  String name = "";
  @override
  Widget build(BuildContext context) {
    return Column(children: <Widget>[
          TextField(
              onChanged: (String string) {
                setState(() {name = string; });
              }),
          Text("Hello " + name + "!")
]);}}
```

Layout widgets

```
DropdownButton<String>(
    onChanged: (value) {
        functionToCall(value);
    }
```

Events

Handle events as properties of Widgets

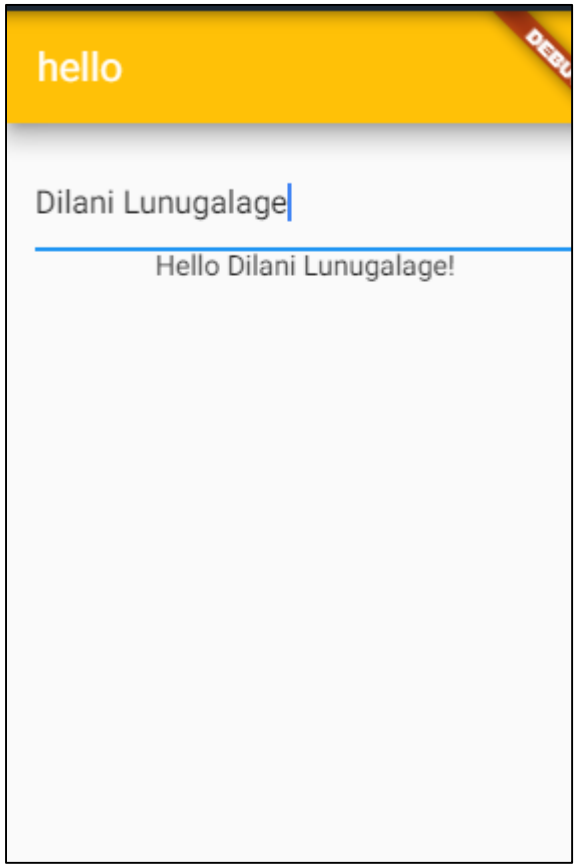
State maintenance widgets

```dart
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Hello You',
      theme: ThemeData(

        primarySwatch: Colors.blue,
      ),
      home: new HelloYou(),
    );
  }
}
```

```dart
class HelloYou extends StatefulWidget{
  @override
  State<StatefulWidget> createState()=>_HelloYouState();

}
```

Statefull Widgets

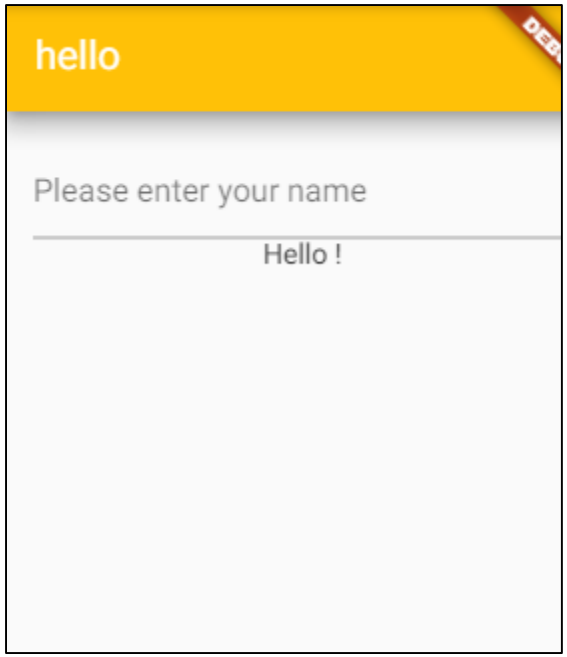
```dart
class _HelloYouState extends State<HelloYou>{
  String name ='';

  @override
  Widget build(BuildContext context) {

    return Scaffold(
      appBar: AppBar(
      title : Text("hello"),
      backgroundColor:Colors.amber

      ),//AppBar
     body:Container(
       padding:EdgeInsets.all(15),
       child:Column(
       children : <Widget>[
         TextField(
         onChanged :(String string) {
           setState((){
               name=string;
            });
          }
        ) ,
       Text ('Hello ' + name + '!' )
       ],//widget
        ), //column
          ) //Container
           ); //Scaffold

  }

}
```




```dart
onSubmitted :(String string) {
```

Decoration….?

```
TextField(
  decoration:InputDecoration(
    hintText : ('Please enter your name')
    ),//Input Decoration
```



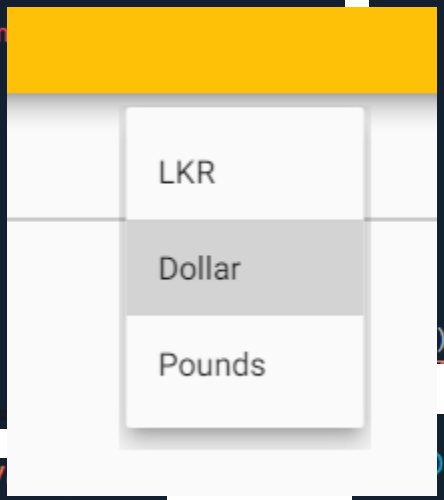
SLIIT
Discover Your Future

Drop Downs

- Generics type

```
DropdownButton<String>(

items : <String>['Dollars','Euros','LKR' ,'poun
value){

  return DropdownMenuItems<String>(

    value :value,
    child :new Text(value),
  );


  }).toList(),
  onChanged:(_) {}
```

```
class _HelloYouState extends State<HelloY
  String name ='';
  final _currencies=['LKR','Dollar','Pounds'];
  String  _currency='LKR';
```

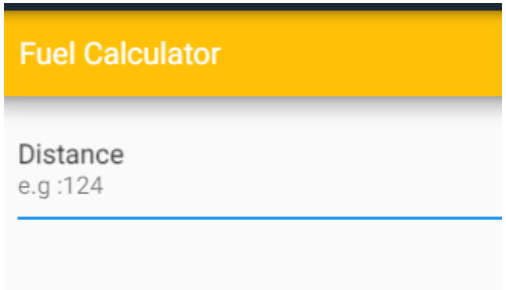


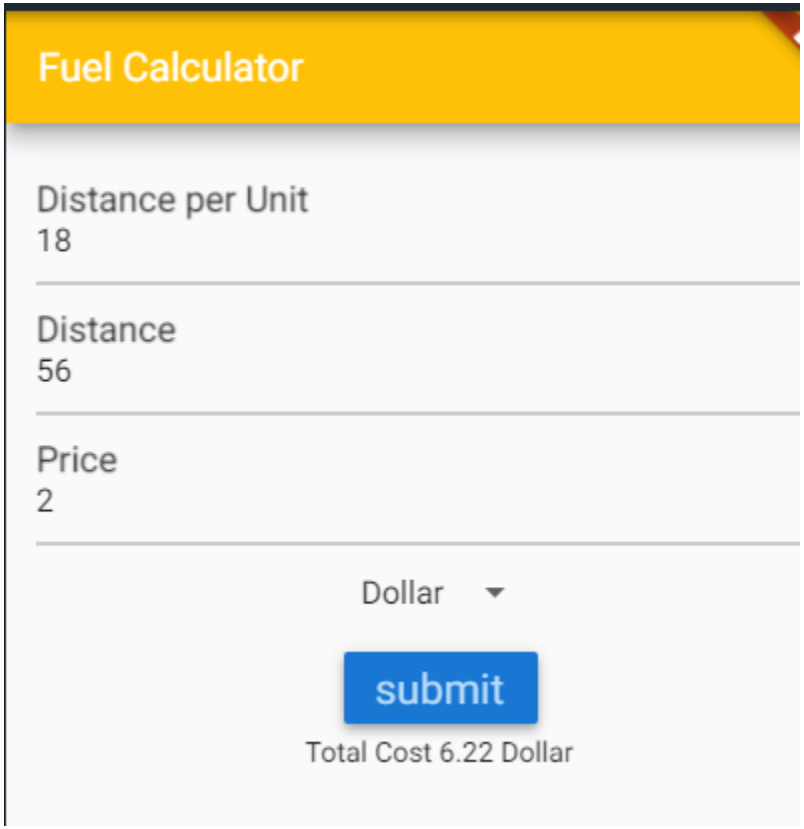

```
  DropdownButton<String>(
  items : _currencies.map((String value

  ){
    return DropdownMenuItem<String>(
    value :value,
    child:Text (value)

    );

    }).toList(),
    value:_currency,
    onChanged:(String value) {
    _onDropdownChanged(value);

      }
),
```

```
ropdownChanged (String value){
 setState((){
    this._currency=value;
  });
}
}
```

Lets Create Fuel Calculator.....

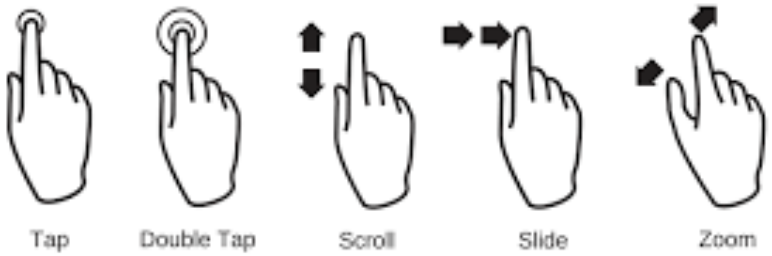
```
Widget build(BuildContext context) {
  TextStyle textStyle =Theme.of(context).textTheme.headline5;
  return Scaffold(
      appBar: AppBar(
          title : Text("Fuel Calculator"),
          backgroundColor:Colors.amber

      ),//AppBar
      body:Container(
        padding:EdgeInsets.all(15),
        child:Column(
          children : <Widget>[
            TextField(
                decoration: InputDecoration(
                labelText: "Distance",
                hintText: "e.g :124",
                labelStyle: textStyle,
            ),
```




Fuel Calculator
Distance per Unit
18
Distance
56
Price
2
Dollar
submit
Total Cost 6.22 Dollar

Introduction to Gestures

- Gestures are primarily a way for a user to interact with a mobile (or any touch based device) application. Gestures are generally defined as any physical action / movement of a user in the intention of activating a specific control of the mobile device.
- Gestures are as simple as tapping the screen of the mobile device to more complex actions used in gaming applications.
- Tap: Touching the surface of the device with fingertip for a short period and then releasing the fingertip.



Widely used gestures

- Tap: Touching the surface of the device with fingertip for a short period and then releasing the fingertip.
- Double Tap: Tapping twice in a short time.
- Drag: Touching the surface of the device with fingertip and then moving the fingertip in a steady manner and then finally releasing the fingertip.
- Flick: Similar to dragging, but doing it in a speeder way.
- Pinch: Pinching the surface of the device using two fingers.
- Spread/Zoom: Opposite of pinching.
- Panning: Touching the surface of the device with fingertip and moving it in any direction without releasing the fingertip.

Gesture Events

Tap

- onTapDown
- onTapUp
- onTap
- onTapCancel

Double tap

- onDoubleTap

Long press

- onLongPress

Vertical drag

- onVerticalDragStart
- onVerticalDragUpdate
- onVerticalDragEnd

Horizontal drag

- onHorizontalDragStart
- onHorizontalDragUpdate
- onHorizontalDragEnd

Pan

- onPanStart
- onPanUpdate
- onPanEnd

onTap……

```dart
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      home:Scaffold(
        appBar:AppBar(
          title : Text('Welcome to flutter'),
        ) ,
        body: Center(
           child: Container(
                  padding: EdgeInsets.all(12.0),
                  decoration: BoxDecoration(
                    color: Theme.of(context).buttonColor,
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                  child: Text('My Button'),
                )
```

```dart
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      home:Scaffold(
        appBar:AppBar(
          title : Text('Welcome to flutter'),
        ) ,
        body: Center(
            child: GestureDetector(
                onTap:(){
                  print('MyButton was tapped!');
                },
                child: Container(
                  padding: EdgeInsets.all(12.0),
                  decoration: BoxDecoration(
                    color: Theme.of(context).buttonColor,
                    borderRadius: BorderRadius.circular(8.0),
                  ),
                  child: Text('My Button'),
                )
            )
        ),

      ),
    );
```

Lets See example of Gestures

- https://flutter.dev/docs/cookbook

- Flutter also provides a low-level gesture detection mechanism through Listener widget. It will detect all user interactions and then dispatches the following events:
- PointerDownEvent
- PointerMoveEvent
- PointerUpEvent
- PointerCancelEvent

Advanced gestures

- Flutter also provides a small set of widgets to do specific as well as advanced gestures. The widgets are listed below:
- Dismissible: Supports flick gesture to dismiss the widget.
- Draggable: Supports drag gesture to move the widget.
- LongPressDraggable: Supports drag gesture to move a widget, when its parent widget is also draggable.
- DragTarget: Accepts any Draggable widget.
- IgnorePointer: Hides the widget and its children from the gesture detection process.
- AbsorbPointer: Stops the gesture detection process itself and so any overlapping widget also can not able to participate in the gesture detection process and hence, no event is raised.
- Scrollable: Support scrolling of the content available inside the widget

Rest API with Flutter

- https://www.c-sharpcorner.com/article/flutter-rest-api/