* **Tutorial-1**

**initState Method:**

@[protected](https://api.flutter.dev/flutter/meta/protected-constant.html)

@[mustCallSuper](https://api.flutter.dev/flutter/meta/mustCallSuper-constant.html)

void initState()

*@*[*mustCallSuper*](https://api.flutter.dev/flutter/meta/mustCallSuper-constant.html)*, @*[*protected*](https://api.flutter.dev/flutter/meta/protected-constant.html)

* Called when this object is inserted into the tree. The framework will call this method exactly once for each [State](https://api.flutter.dev/flutter/widgets/State-class.html) object it creates.
* Override this method to perform initialization that depends on the location at which this object was inserted into the tree (i.e., [context](https://api.flutter.dev/flutter/widgets/State/context.html)) or on the widget used to configure this object (i.e., [widget](https://api.flutter.dev/flutter/widgets/State/widget.html)).

**Uri.parse Method:**

[Uri](https://api.flutter.dev/flutter/dart-core/Uri-class.html) parse(

* [String](https://api.flutter.dev/flutter/dart-core/String-class.html) uri,
* [[int](https://api.flutter.dev/flutter/dart-core/int-class.html) start = 0,
* [int](https://api.flutter.dev/flutter/dart-core/int-class.html)? end]

)

* Creates a new Uri object by parsing a URI string. If start and end are provided, they must specify a valid substring of uri, and only the substring from start to end is parsed as a URI. If the uri string is not valid as a URI or URI reference, a [FormatException](https://api.flutter.dev/flutter/dart-core/FormatException-class.html) is thrown.

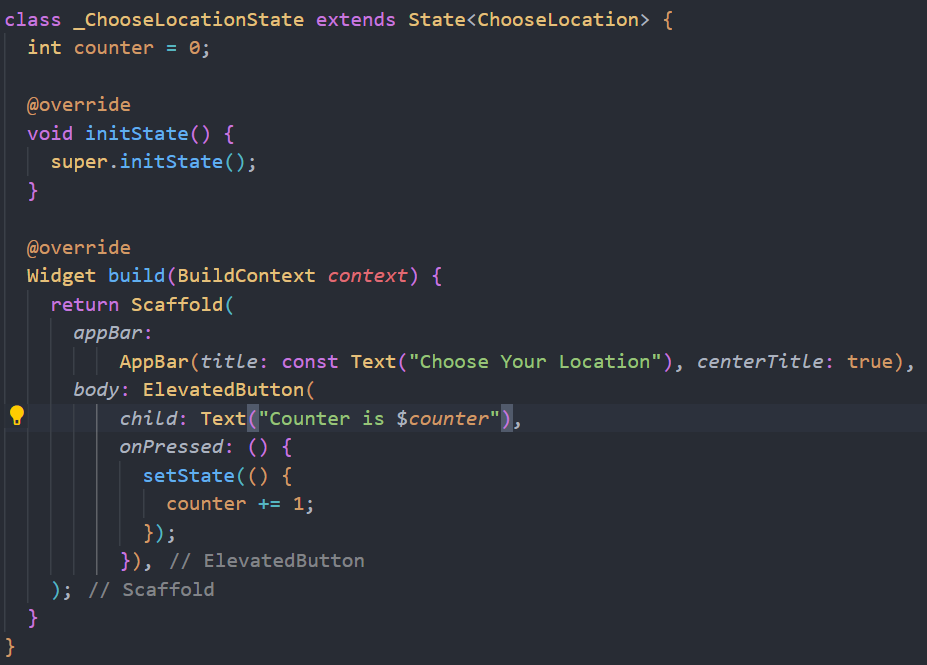
**jsonDecode Method:**

dynamic jsonDecode(

* [String](https://api.flutter.dev/flutter/dart-core/String-class.html) source,
* {[Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? reviver(
  + [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? key,
  + [Object](https://api.flutter.dev/flutter/dart-core/Object-class.html)? value
* )?}

)

* Parses the string and returns the resulting Json object. The optional reviver function is called once for each object or list property that has been parsed during decoding. The key argument is either the integer list index for a list property, the string map key for object properties, or null for the final result.







* **/pages/loading.dart**

import 'package:flutter/material.dart';

import 'package:http/http.dart';

import 'dart:convert';

class Loading extends StatefulWidget {

const Loading({Key? *key*}) : super(*key*: *key*);

@override

State<Loading> createState() => \_LoadingState();

}

class \_LoadingState extends State<Loading> {

void getData() async {

final response =

await get(Uri.parse('https://jsonplaceholder.typicode.com/albums/1'));

Map data = jsonDecode(response.body);

print(data['title']);

}

@override

void initState() {

super.initState();

getData();

}

@override

Widget build(BuildContext *context*) {

return const SafeArea(

*child*: Scaffold(

*body*: Text("Loading Screen"),

));

}

}

* **main.dart**

import 'package:flutter/material.dart';

import 'package:lab10/pages/choose\_location.dart';

import 'package:lab10/pages/loading.dart';

void main(List<String> *args*) {

runApp(MaterialApp(

*debugShowCheckedModeBanner*: false,

*initialRoute*: "/",

*routes*: {

"/": (*context*) => const Loading(),

"/home": (*context*) => const Home(),

"/location": (*context*) => const ChooseLocation(),

},

));

}

class Home extends StatefulWidget {

const Home({Key? *key*}) : super(*key*: *key*);

@override

State<Home> createState() => \_HomeState();

}

class \_HomeState extends State<Home> {

@override

Widget build(BuildContext *context*) {

return MaterialApp(

*debugShowCheckedModeBanner*: false,

*home*: Scaffold(

*body*: SafeArea(

*child*: Column(

*children*: [

Center(

*child*: TextButton.icon(

*onPressed*: () {

Navigator.pushNamed(*context*, "/location");

},

*icon*: const Icon(Icons.edit\_location, *size*: 50),

*label*: const Text(

"Location",

*style*: TextStyle(*fontSize*: 30),

)),

),

],

),

),

));

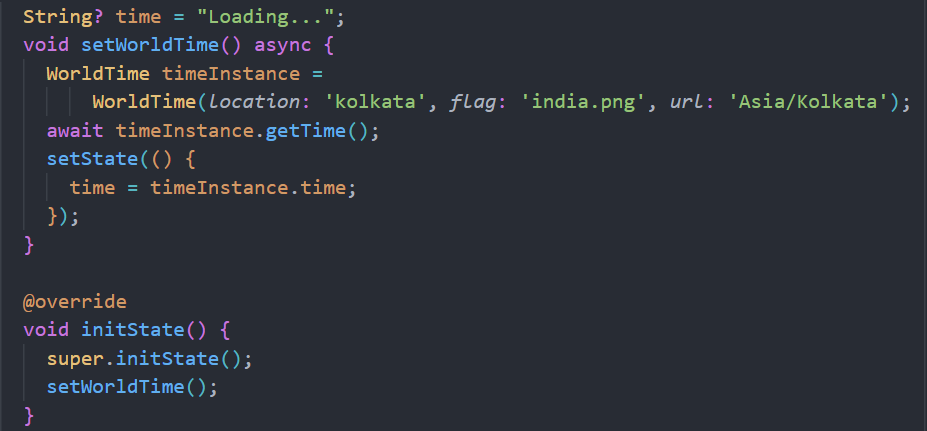
}

}

* **Tutorial-2**

**DateTime Class:**

* An instant in time, such as July 20, 1969, 8:18 pm GMT.
* DateTimes can represent time values that are at a distance of at most 100,000,000 days from epoch (1970-01-01 UTC): -271821-04-20 to 275760-09-13.
* Create a DateTime object by using one of the constructors or by parsing a correctly formatted string, which complies with a subset of ISO 8601. Note: hours are specified between 0 and 23, as in a 24-hour clock.

****

****

****

* **/pages/loading.dart**

import 'package:flutter/material.dart';

import 'package:lab10/services/world\_time.dart';

class Loading extends StatefulWidget {

const Loading({Key? *key*}) : super(*key*: *key*);

@override

State<Loading> createState() => \_LoadingState();

}

class \_LoadingState extends State<Loading> {

String? time = "Loading...";

void setWorldTime() async {

WorldTime timeInstance =

WorldTime(*location*: 'kolkata', *flag*: 'india.png', *url*: 'Asia/Kolkata');

await timeInstance.getTime();

setState(() {

time = timeInstance.time;

});

}

@override

void initState() {

super.initState();

setWorldTime();

}

@override

Widget build(BuildContext *context*) {

return SafeArea(

*child*: Scaffold(

*body*: Center(

*child*: Column(

*children*: [

const Text("Time for your location",

*style*: TextStyle(*fontSize*: 20)),

TextButton(

*child*: Text('$*time*'),

*onPressed*: () {},

),

],

),

),

));

}

}

* **/services/world\_time.dart**

import 'package:http/http.dart';

import 'dart:convert';

class WorldTime {

String? location, time, flag, url;

WorldTime({this.location, this.flag, this.url});

Future<void> getTime() async {

Response response =

await get(Uri.parse('http://worldtimeapi.org/api/timezone/$*url*'));

Map timeData = jsonDecode(response.body);

String dateTime = timeData['datetime'],

offset = timeData['utc\_offset'],

offsetHours = offset.substring(1, 3),

offsetMinutes = offset.substring(4, 6);

DateTime currenttime = DateTime.parse(dateTime);

currenttime = currenttime.add(Duration(

*minutes*: int.parse(offsetMinutes), *hours*: int.parse(offsetHours)));

time = currenttime.toString();

}

}