FLUTTER

How To Create a Flutter project in visual studio:

Use the visual studio code terminal to create project with blow code:

flutter create yourProjectName

To run a flutter project, open cmd and navigate to the project folder and open and run:

flutter run

then, whiles it has loaded, if you do any changes in your code, just press "r" on the console and it will hot reload the changes automatically

=====================================================================================================

The pubspec.yaml file

It the file where you need to specify:

* The dependencies you are going to be using in your project.
* The Images you want to use in your project
* The font you want to use in your project

=====================================================================================================

How To fix run error requiring download artifact When You first Create A project and tries to Run it:

When you first create a flutter application and you try running it, you may get some “download failed artifact error”

To fix it…

change your project android/gradle/wrapper/gradle-wrapper.properties "distributionUrl" property to:

distributionUrl=C:/Users/Administrator/.gradle/gradle-6.5-all.zip

or

distributionUrl=https\://services.gradle.org/distributions/gradle-6.5-all.zip

=====================================================================================================

All Flutter Widget are classes

Scaffold, Text, AppBar All widget are classes and whenever you call one, you are required to pass in parameter to it class constructor. But not all parameters are required.

Example:

Home: Scaffold( appBar: AppBar( title:Text(“hi”) ) );

‘’appBar” it’s already defined name parameter in the scaffold class constructor that has a data-type of AppBar class.

So this “appBar” parameter is not custom user defined, it a name parameter created in the Scaffold class constructor by flutter already.

So the format is….

Scaffold( appBar: AppBar(…….) );

Name parameter

Return data-type class constructor of this key. It a class constructor you need to pass in some parameters into it.

=====================================================================================================

Colors & fonts

To choose a color for some widget, let say AppBar widget…

Inside the appBar widget class Constructor, add Background color like:

AppBar(backgroundColor: Colors.red[600]),

After typing the “colors.” press ctr+space and you will get list of different colors to choose from.

To add a font…

Create a root folder in your project directory and name it “fonts”. Then download some font and extract it into the “fonts” folder you just created.

Then in your project, open the pubspec.yaml and uncomment the font section in it and specify the following…

fonts:

    - family: justicefont  # Give your font some name

      fonts:

        - asset: fonts/NotoSansJRegular.otf  # then specify where the font is here with it name in the folder

Then inside main.dart, you can apply the above font to the “Text” widget by using the “TextStyle” name parameter class. The TextStyle class constructor takes in different parameters that will be applied to the Text widget.

  child: Text("hi",

          style: TextStyle(fontFamily:"justicefont", fontSize: 20.0, fontWeight: FontWeight.bold, color: Colors.red, letterSpacing: 2.0)),

=====================================================================================================

Images & Asset

You can use the “Image.network” widget to fetch images from an http request, by passing the url of the image to it.

And you can use the “Image.asset” widget to fetch local images stored in your project “assets” folder.

To fetch network / http image inside main.dart:

home: Scaffold(

  body:  Image.network("https://images.unsplash.com/photo-1619478691745.png”, width: 120, height: 60) // add the url of the image here

)

To fetch Local image, inside main.dart:

Create a folder in your project root with name “assets” and add images to it.

Then inside the “pubspec.yaml” file uncomment the “assets” section and add the path of your image like:

assets:

    - assets/justy.jpg  # add the path of your image here stored in “assets” folder here

Then inside main.dart, display the above image like:

home: Scaffold(

  body:  Image.asset("assets/justy.jpg") // link to the image in the above pubspec.yaml file

)

=====================================================================================================

How To cache Network Images

To cache http network image, use the “cached\_network\_image” dart package. It works like glide in android studio

First add the below dependency to your pubspec.yaml file. Then click on “pub get” pop up on the pubspec.yam file to download the package

dependencies:  
 flutter:  
 sdk: flutter  
 cached\_network\_image: ^3.0.0

Then pass the network image url to the “imageUrl” parameter of the “CachedNetworkImage” constructor to display it…

home: Scaffold(

  body:  CachedNetworkImage(imageUrl: “https://images.unsplash.com/pho.png”, height: 200, width:200) // add the url of the image

)

=====================================================================================================

Icon & Buttons With Icon

To add Icon…

home: Scaffold(

   body:  Icon(

         Icons.airport\_shuttle, // add the icon suggestion by flutter here (you can reasearch how to add your own icons)

         color: Colors.red, // icon color

         size: 50.0 // icon size

      )

       )

To add button with icon…

 home: Scaffold(

   body:  RaisedButton.icon(onPressed: (){print("button clicked");}, // function to call when the button is clicked

          icon: Icon( Icons.airport\_shuttle), // icon

          label: Text("click me"),  // text on the button

          elevation:10.0            // elevate, just like z-index in css

          )

       )

=====================================================================================================

**Flutter Widgets…….**

Scaffold Widget

scaffold() ==> Widget, Represent the whole screen/ or page that you are to place your dom element in it. (it like the HMTL <body> tag)

It constructor have 2 special named parameters:

appBar: For adding appBar to the application, this represent the top Navigation part of your app

body: then this represent the body of your application. You will mainly wrap this widget with safeArea widget to position it righty below the appBar nicely.

=====================================================================================================

The Stack Widget

The stack widget is used to group different widget on top of each other.

Let say you have a container and a Text Widget. And you want to position the text on top of the container, you can use the stack widget.

The stack class constructor has a “children” name parameter that accept list of widget that will be grouped on top of each other.

It also has the “AlignmentDirectional” that let you specify where the widget are grouped in the stack container (remember=> stack() is an invisible layout container)

Example:

home: Scaffold(

   body: Stack(

        alignment: AlignmentDirectional.bottomCenter, // used to align the children in the stack container

        children: [

          // childrens to be arrange on top of each other

          Container(color: Colors.grey, width: 200, height: 200, ), // first child (this will go down of all)

          Container(color: Colors.yellowAccent, width: 100, height: 100, ), // Second child (this will on top of above child)

          Container(color: Colors.blue, width: 50, height: 50, ),  // Third child (this will be on top of above child)

        ],

      )

       )

=====================================================================================================

Container widget

It’s used to put a single child widget into a container containing: padding, margin, width, height, etc.

Example…

 home: Scaffold(

   body:   Container(

        padding: EdgeInsets.all(10.0), // container padding

        child: Text("Justilato"),  // put this "Text" widget into a container, it like putting some html tag into the <div> tag

        width: 300.0, // container width ( double.infinity will let the container stretch to take all the available space)

        height: 100.0, // container height

        color: Colors.red // container color

      ),

       )

=====================================================================================================

SizeBox Widget

Is used to give vertical or horizontal space between two Widget. using it "width" and "height" parameter

To give a vertical space, use it height parameter: SizedBox(height: 10.0)

To give a horizontal space, use it width parameter: SizedBox(width: 10.0)

example: give some vertical space between below two "Text" widget

home: Scaffold(

               body:  Column(

              children: [

                Text("justilato"),

                SizedBox(height: 15.0), // this will give a vertical space between the above and below "Text" Widget.

                Text("how are you doing?")

              ],

            ),

     )

=====================================================================================================

Divider Widget

Is used to give a horizontal line divider. It just like html <hr> tag

All you need is to specify the height and color you want for your divider example:

home: Scaffold(

               body: Divider( height: 20.0, color: Colors.grey[850] ),

     )

=====================================================================================================

Circle Avatar Widget

It used to add an image that will be wrap into a round container by default.

All you need is to give it a backgroundImage and use the radious parameter to specify the width of the image

**example:**

home: Scaffold(

        body: CircleAvatar(

                    onBackgroundImageError: (object, stacktrace){

                      print("error loading image");                    //method to be called if image loading fail

                    },

                    backgroundImage: AssetImage("justy.jpg"),        // image in the assets folder specified in pubspec.yaml file

                    radius: 40.0,                                   // the width & height of the image

                  ),

)

=====================================================================================================

Row & Column Widget

Is used to add different widget into it to be grouped in a row or columns.

Row==> groups it children widget into horizontal alignment

Column==> groups it children widget into vertical alignment

you can use the "CrossAxisAlignment" and the "MainAxisSize" to align the children widget base on the vertical & horizontal direction of it parent container.

Example: Row to align it children horizontally

home: Scaffold(

        body:  Row(

                  children: [

                    Text("Justiceankomah12@gmail.com"),

                    Text("Justiceankomah12@gmail.com"),

                    Text("Justiceankomah12@gmail.com")

                   ]

                )

)

Example: Column to align it children vertically

home: Scaffold(

        body:   Column(

              crossAxisAlignment: CrossAxisAlignment.start, // arragnge the child widget at the beginng (like css flex box)

              children: [

                Divider(height: 20.0, color: Colors.grey[850]),

                Text("name", style: TextStyle(color: Colors.grey)),

                Text("Justice Ankomah", style: TextStyle(color: Colors.yellowAccent,fontWeight: FontWeight.bold, fontSize: 20.0)),

                SizedBox(height: 20.0),

                Text("Flutter level", style: TextStyle(color: Colors.grey)),

                Text("8", style: TextStyle(color: Colors.yellowAccent,fontWeight: FontWeight.bold, fontSize: 20.0)),

                SizedBox(height: 20.0)

              ]

            )

)

====================================================================================================

Expanded Widget

It used to wrapped around a widget to span and take all the available space.

It mainly used to wrap around Row & Column Widget.

Example:

      home: Scaffold(

        body: Row(

            children: [

              Container(  // first widget child

                color: Colors.black26,

                width: 200,

                height: 200,

              ),

              Expanded(  // Let this widget expand to fill all the available space if there is any

                child: Container(  // second widget child

                  color: Colors.yellowAccent,

                  width: 200,

                  height: 200,

                ),

              ),

              Container( // Third widget child

                color: Colors.black26,

                width: 200,

                height: 200,

              ),

            ],

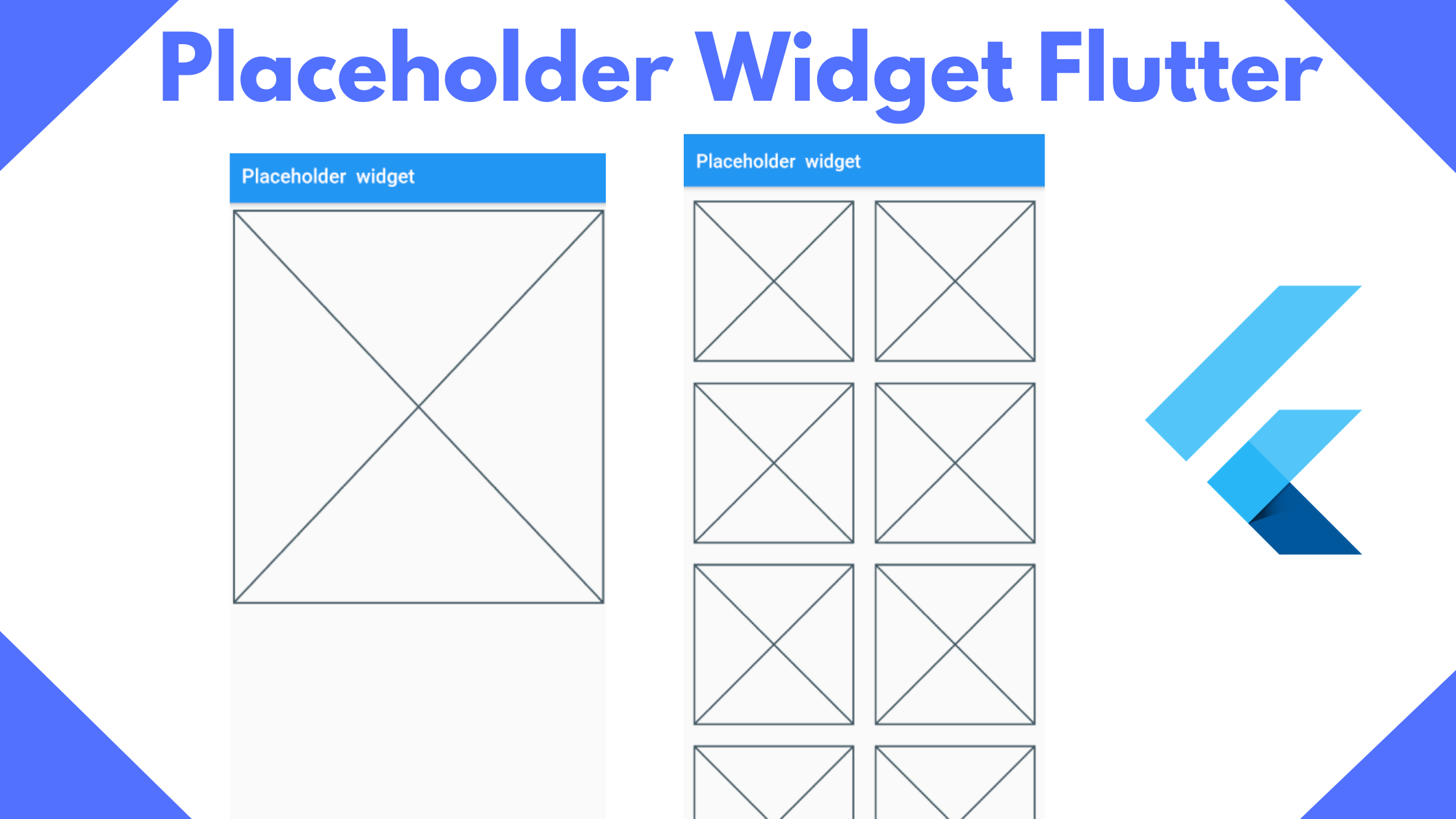
          ),

        ),

====================================================================================================

Placeholder Widget

The placeholder widget is used to see the space a parent widget occupies on a screen. For example, If, you have a container and you want to see the space the container occupies on the screen. You can use the placeholder widget.



**Example:**

 Container(

  height: 100,

  width: 100,

  // Use placeHolder to show the box or the space this Container takes

  child: Placeholder(

    color:Colors.red,

    strokeWidth: 2,

  ),

),

====================================================================================================

Flutter WIDGET SYNTESS is all based on "NAME PARAMETERS"

To create a name parameters, surround the class constructor parameters with curry braces "{}"

All named parameters are not required to be provided, to make a specific parameter strictly required, annotate that parameter with “@required”

Then when passing in value to the constructor in different class, use the name parameters. Example:

// Create a class with a constructor that takes in name parameters

class Quote{

// create properties of the class

  String test;

  String author;

String authorAddress

  // create a constructor of the class that takes in name parameters

  Quote({String test, String author, String authorAddress }){

    this.test=test;

    this.author=author;

this.authorAddress = authorAddress ;

  }

  /\*

  you can simply create the above constructor easily as...

   Quote({this.test, this.author, @required this.authorAddrees});

The above “@required this.authorAddrees” means = that parameter is strictly required to be provided

  \*/

}

// Then to call the constructor in a different class, you need to pass in the values using the name parameters like below;

// the order of the name parmeters is not important, you can mismatch it and Dart will figure it out base on the named parameters

   Quote quote\_obj = new Quote(test: "good boy", author: "jutice");

=====================================================================================================

Dart Private Variables and method

Unlike Java dart doesn’t have private, protect, public access modifies.

So to mark a method or variable as private, you have to add “\_” at the beginning of the variable or method name. This will make such method or variable private to be accessible only in that particular class.

**Example:**

class MyBooks{

  // private variable not accessible to outside class

  var \_bookName  = "Aki-Ola";

 // private Method not accessible to outside class

  \_fetchBookName(){

    print(this.\_bookName);

  }

}

=====================================================================================================

Listtile widget

Is kind of a card that has beginning and ending icon, title and subtitle. You can mainly use it inside a card or column widget.

Home: Scaffold(

      body:Card(

            color: Colors.red,

            elevation: 1,

            child: ListTile(

              title: Text("Name"),  // title

              subtitle: Text("Justice"), // sub title or description

              leading: Icon(Icons.expand),  // beginning icon

              trailing: Icon(Icons.more\_vert), // ending icon

              hoverColor: Colors.black26,  // color to show when it hovered

              tileColor: Colors.yellowAccent,  // color of your listtile

              selectedTileColor: Colors.white, // color of the selected listtile

              isThreeLine: true,  // allow the listile to expand to three lines because (its only title & subtitle = two lines)

              selected: true,    // check if listile is active or inactive

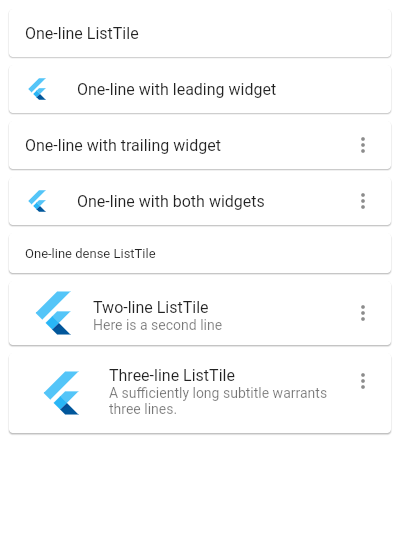
              onTap: (){ print("clicked");},       // function to call when listile is clicked

              onLongPress: (){print("long pressed");},  // function to call when listile is long pressed

            ),

          ),

    );



====================================================================================================

How To define a Named parameter that accept a function

To create a parameter that has a dataType of function, just assign the “VoidCallBack” class to it as it dataType.

Example:

Inside SideMenuIonTab Class

class SideMenuIonTab extends StatelessWidget {

  // Properties

  final String title;

  final VoidCallback ontap; // property that has a dataType of function

  // create a constructor to initialize above properties as named parameters

   // when instance of this costructor is created, it will call the "build" method below automatically

  SideMenuIonTab({ this.title, this.ontap});

  @override

  Widget build(BuildContext context) {

    return ListTile(

      leading: Icon(iconData, size: 28.0, color: Colors.white,),

      title: Text(

        title,

        style: TextStyle(color: Colors.white),

        overflow: TextOverflow.ellipsis,  // show the ... symbol incase the text is big for the screen

      ),

      onTap: ontap,

    );

  }

}

Inside main.dart Calss

home: Scaffold(

  // this will call the build method inside above sideMenuIonTab Class automatically

  body: SideMenuIonTab(

             title: "home",

              ontap: (){print("on clicked");} // pass a function to the above ontap parameter

              ),

)

=====================================================================================================

How to change the color of an item In A List When clicked

You can easily change the color of an items when it clicked by doing what I have explained below.

List<String> colors =["blue", "yellow"];

// define a variable to represent the index of the select item in the list

  int selectedIndex = 0;

  // a function to change the above "selectedIndex" index when that list is clicked

  void onSelected(int index) {

    setState(() => selectedIndex = index);

  }

  ListView.builder(

      shrinkWrap: true,

      itemCount:colors.length,

      itemBuilder:(context, index){

       return InkWell(

         child: Text(

             colors[index], // show each color

             // if  \_selectedIndex not null and is equal to the selected list, set it text color to red. else color is black

             style: TextStyle(

               color: (selectedIndex != null && selectedIndex == index)

             ? Colors.red

             : Colors.black87,) ),

         onTap: (){

           // call the onSelected() method above to set the current selected items index to it

           onSelected(index);

         },

       );

      },

    ),

=====================================================================================================

Scrollbar widget

The scrollbar widget is used to show scrolls overflow around other scroll widget like listview and gridview. It just like the css overflow: true property.

Make sure to always set a different controller for each single Scrollbar. If you don’t understand (learn on youtube)

Example:

 class \_PlaylistScreenState extends State<PlaylistScreen> {

   // create a Scrollbar controller

  ScrollController? \_scrollController;

  @override

  void initState() {

    super.initState();

    \_scrollController = ScrollController();  // Inizialte the controller when app start

  }

  @override

  void dispose() {

    \_scrollController?.dispose();  // terminate the controller when such screen is closed

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

  home: Scaffold(

   // the scrollbar widget

      body: Scrollbar(

              controller: \_scrollController,  // add controller for your scrollbar

              showTrackOnHover: true,  // show the other area of the scrollbar when hovered

              isAlwaysShown: true,    // always show the scrollbar overflow

                child: ListView.builder(

                  controller: \_scrollController,  // add the controller so that it will know the last item in the listview to know the end of the listview

                     itemCount: names.length,

                    itemBuilder:(context,index){

                       return Text(names[index]);

                    }

                ),

              ),

            );

    )

  }

 }

====================================================================================================

BoxDecoration Widget

The boxdecoration widget is a widget use to draw a box in your app. You can give your box a shadow, a color, border radius, add image to it, gradient, you can give it a shape of circle or rectangle.

If you give it a shape of rectangle, then the borderRadius parameter, will control the roundness of the 4 corners of the rectangle.

If you give it a gradient, it will override the color property.

**Example:**

 home: Scaffold(

      body: Container(

        // if you add a color here, flutter will be confused throw error

      width: 500,

      decoration: BoxDecoration(

        color: Colors.yellow, // give it a color

        // add image to it. AssetImage() or NetworkImage()

        image: DecorationImage(image: NetworkImage("https://images.unsplash.com/ben.png")),

         // do you want a "rectangle" or "circle" shape?

        shape: BoxShape.rectangle,

        // give it a gradient (it will override the aobve "color" parameter)

        gradient: LinearGradient(

          colors: [Colors.red,Colors.blue],  // list of your gradient colors

          begin: Alignment.topRight,   // where your gradient should begin

          end: Alignment.bottomRight,  // where your gradient should end

stops:[0,0.3], // stop the gradient at some some point to prevent it from filling the whole box

        ),

        // give it a shadow

        c

        // give your box a radius

        borderRadius: BorderRadius.only( // use .only() if you want to specify individual borders. use .all() if you want to give all corners a border

          bottomLeft: Radius.circular(40),          // give bottomleft of the container radius of 40

          bottomRight: Radius.circular(40),       // give bottomRight of the container radius of 40

          // topRight:  Radius.circular(40),    // give topRight of the container radius of 40

          // topLeft: Radius.circular(40),    // give topLeft of the container radius of 40

        ),

      ),

      ),

    );

====================================================================================================

Inkwell Widget

It is widget that is used to add ontap function, onlongpress function, onHover function, onhover color to any other widget.

All you need to do is to wrap that widget inside the inkwell widget and you are good to go.

Example:

 home: Scaffold(

      body: InkWell(

        onTap: (){print(" on tapped");},  // function to call when clicked

        onLongPress: (){print("long pressed");},   // function to call when longpressed

        /\* function that takes in custome boolean parameter that indicate if the widget is hovered or not\*/

        onHover: (isHoverd){

         if(isHoverd){ print("is hovered");}

          },

        hoverColor: Colors.redAccent,  // color to show when hovered

        // container that the inkwell Widget is wrapped around

        child: Container(

        width: 500,

        color: Colors.blue,

        child: Text("click me"),

        ),

      ),

    );

====================================================================================================

The spacer widget

The spacer widget is used to add spaces between children widget inside row or column Widget.

Example:

    home: Scaffold(

      body: Center(

        child: Column(

          children: [

            Container(  // firt child container

              width: 200,

              color: Colors.blue

            ),

            // add a space that flex 2 boxes here

            Spacer(flex:2),

            Container(   // Second child container

                width: 200,

                color: Colors.yellow

            ),

            Container(   // third child container

                width: 200,

                color: Colors.blue

            ),

          ],

        ),

      ),

    );

****

**====================================================================================================**

NOTE==> to create a statful widget, simply move outside the class and type the shortcut "stful"

====================================================================================================

How To Convert A Class To Stateful Widget In Android Studio

To convert a class to a Stateful widget in android studio, Highlight the class name, the "action menu" will be shown immediately.

Then click on the "convert to Stateful widget" feature in the action menu.

And also to see the "action menu" of each widget, simply highlight the name of that widget and it will be shown

====================================================================================================

The setState(){} method and how to display int variable in a Text Widget.

The setState(){} is a method that call the "build" method to rebuild the changes applied to all widget in stateful widget class.

(Thats whenever you want to change some value of some widget like "Text" you need to call it. but make sure the class is converted into stateful widget first) Because stateless widget classdon’t have any context/life to rebuild it widget after the app has started.

use $theIntVariableName format to display an int value in "Text" widget

class \_MyAppState extends State<MyApp> {

  int ninjaLevel=0;

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

        floatingActionButton: FloatingActionButton(

          onPressed: (){

                   // the "setState" to notify flutter that you want to rebuild your widget to apply some changes

              setState(() {

                  ninjaLevl +=1;

              });

          },

          child: Icon(Icons.add)

        ),

         // "$ninjaLevel",  below will display the above "int ninjaLevel=0" variable value

        body: Text("$ninjaLevel", style: TextStyle(color: Colors.yellowAccent,fontWeight: FontWeight.bold, fontSize: 20.0)),

    );

  }

}

====================================================================================================

Provider

Is a different way of managing state in your application instead of calling setState() method all the time.

It the best because once a provider is created, you can get it value in all the screens in your app.

Steps:

1. Create a new class that extends ChangeNotifier class. This class has a method called notifyListeners() which you can call whenever you want to change a value of a data.
2. Listen to the provider changes with Provider.of<T>(context, listen: Bool).T
3. Wrap your top most screen with multiprovider

**Example:**

Inside Counter.dart

// Create a new class that extends ChangeNotifier

// This class is where am storing all my states

// To Make it a Provider, extend ChangeNotifier. This will give you access to the notifyListeners() method, which will notify all the widget listening to it

// when any value in this class changes

class Counter extends ChangeNotifier{

  var \_count = 0;

  // method to increase the counter

   void incrementCounter() {

   \_count += 1;

   print(\_count);

    // call notifyListeners(), this will trigger a change of value in the whole app of whatever widget is listening to it

// so in our case, it will notify all widget listening to the value of the \_count (it like the setState() method )

    notifyListeners();

  }

  // getter to get count value

  int get getCounter {

    return \_count;

  }

}

// inside main.dart

import 'package:provider/provider.dart'; // import provider

void main() {

  // Wrap your Topmost Screen with Multiprovider so that all your app screens can have access to provider

  runApp(

    // Multiprovider allows you to define multiple provider class that extends ChangeNotifier

      MultiProvider(

        // list of provider classes

          providers: [

            //   ChangeNotifierProvider.value(value(value: your class that extends changeNotifier ))

            ChangeNotifierProvider.value(value: Counter(),),

            // you can create and add another provider class here

          ],

          child: MyApp()

     )

  );

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    //get the counter value by using provider

    // set listen to true, because we want to listen whenever the counter value changes

    var counter = Provider.of<Counter>(context, listen: true).getCounter;

    return Center(

      child: Column(

        children:[

          Text(counter.toString()),

          SizedBox(height: 30,),

          InkWell(

            child: Text("click Me"),

            onTap: (){

              //  Provider.of<Counter>() = means what provider do you want me to look for? "Counter" is what I have specify

              // then once you have gotten the provider, you can call any method on it.

              // here we just want to increase the counter, notice I have set listen to false. Because I don't want to listen to any changes

              Provider.of<Counter>(context, listen: false).incrementCounter();

            },

          ),

        ]

      ),

    );

  }

}

You can create another Screen to listen to the count value in the Count class, and it value will be the same as what is set in the MyApp screen

====================================================================================================

How To create custom widget class and export them in a single file

Importing every widget you create into other classes can be difficult in-case you change the name of your widget class file. You will have to go to all classes that import the file and change the name. This is a lot of work.

So the concept is to create a single class that will export all the custom widget class files you will create. So in case of any changes, you will just only alter the single class that export all the widget class files.

First: Create a new package inside “lib” folder and name it “widgets” (it can be anything)

Then create different widget class files inside the above “widget” package. Example Widget/shell.dart

Inside lib/widget/shell.dart // this is in the Widget/shell.dart

import 'package:flutter/material.dart';

class Shell extends StatefulWidget {

  @override

  \_ShellState createState() => \_ShellState();

}

class \_ShellState extends State<Shell> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body:

          Container(

            color: Colors.red,

            width: 300,

            height: 300,

            child: Text("hi"),

          ),

    );

  }

}

Then create a dart file “exportwidget.dart” inside “lib” folder, and export the above Shell.dart class. So that any class who want to import the above file will simply import this file.

Inside lib/exportwidget.dart

export 'widgets/shell.dart';  // Export this class so that any class can import this file

Then inside main.dart, if you want to use the shell.dart widget. Simply import the above “exportwidget.dart”

Inside lib/main.dart

import 'package:flutter\_spotify\_ui/exportwidget.dart';  // import exportwidget.dart which export shell.dart

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

         return MaterialApp(

           title: 'Flutter Spotify UI',

            home: Shell(),   // use the shell widget here (you can even let the "shell" class accept a name parameter if you want to)

         );

  }

  }

===================================================================================================

How To Cycle through list of String And display it in a text

class \_MyAppState extends State<MyApp> {

//create a list of strings

  List<String> quotes=["success is not an accident", "Who is in the mirror is your enemy", "be a man of means"];

 @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

        body: Column(

            // loop through the "quotes" list defined above and create a list and assign each to it ("e" represent each single data in the list)

// then convert it into list

          children: quotes.map((e) => Text(e)).toList(),

        )

     )

 }

}

====================================================================================================

Display list of Data In a Different class

// inside Quote.dart class

class Quote{

// create properties of the class

  String test;

  String author;

  // create a constructor of the class that takes in name parameters

  Quote({String test, String author }){

    this.test=test;

    this.author=author;

  }

}

//inside maind.dart class

class \_MyAppState extends State<MyApp> {

  List<Quote> quotes=[

    Quote(test: "man is going", author: "justice"),

    Quote(test: "Bnnard is going", author: "Shadrack"),

    Quote(test: "Ghana be my country where i live", author: "james")

  ];

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Scaffold(

       body: Column(

            // loop through the above "quotes" List and create "Text" widget for each and display data in it

          children: quotes.map((quotes) => Text('${quotes.test} - ${quotes.author}')).toList(),

        )

     )

   )

}

====================================================================================================

Display Dynamic list of data into a Card

// inside Quote.dart

class Quote{

// create properties of the class

  String test;

  String author;

  // create a constructor of the class that takes in name parameters

  Quote({String test, String author }){

    this.test=test;

    this.author=author;

  }

}

// Inside main.Dart

class \_MyAppState extends State<MyApp> {

  int ninjaLevl=0;

  List<Quote> quotes=[

    Quote(test: "man is going", author: "justice"),

    Quote(test: "Bnnard is going", author: "Shadrack"),

    Quote(test: "Ghana be my country where i live", author: "james")

  ];

// create a method that returns a widget

// the type of below "quote" parameter is dynamic

  Widget quoteTemplate(quote){

    return Card(

      elevation: 10.0,

      child: Padding(

        padding: const EdgeInsets.all(8.0),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.stretch,

          children: [

            Text(quote.test),

            Text(quote.author)

          ],

        ),

      ),

    );

  }

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

        home: Scaffold(

            body: Column(

              children: quotes.map((quotes) => quoteTemplate(quotes)).toList(),

            )

        )

    );

  }

}

*====================================================================================================*

*How To map through list and convert each to a single widget.*

To map through a list and convert it to a widget, use the …ListObjectNamehere.map((e)=> Text(e))

Example:

inside yourLibrary.dart

//list of string

List<String> yourLibrary = [

  'Made For You',

  'Recently Played',

  'Liked Songs',

  'Albums',

  'Artists',

  'Podcasts',

];

   inside main.dart

              Column(  // arange items vertically

              crossAxisAlignment: CrossAxisAlignment.start,

              children: [

// use the ... infront of the List\_Object to convert the above list to a widget like below

                ...yourLibrary.map(

                      (e) => ListTile(

                    dense: true,

                    title: Text(

                      e,

                      style: TextStyle(

                        fontSize: 11.0,

                        color: Colors.white,

                        letterSpacing: 1.0,

                      ),

                      overflow: TextOverflow.ellipsis,

                    ),

                    onTap: (){print("on taped");},

                  ),

                )

              ],

            ),

Note: About the map() method

To simplify things, don’t use arrow functions whiles mapping through something.

The method is like this: map((e)=>(……….).toList()

You can simply do this:

map((e){

// you can do something to the list value before assigning it to a widget

return Text(e.firstName);

}).toList()

This is much simple than the arrow function. Simply remove the => and replace it with curry braces {}

====================================================================================================

WIDGET LIFE CYCLE

flutter has a life cycle for all it widget like a "Text". Does which method is called:

To create the widget

To change the widget value

To remove the widget from the Screen

All widget have only 3 life-cycle methods and it is in this order: "initState()", "build()", "dispose()"

"initState()" ==> This is called only once when the page first load to create all the widget of that page

"build()" ==> This is called immediately after "initState()" is called to set values for the created widgets.

It has a method called "setState()" which can be called whenever the programmer want to change the values in any of the widget.

Calling the "setState()" method will automaticaly call the "build()" method to identify changes in the widget values and apply the new value to the widget

"dispose()" ==> This method is called when the current page that the widget is created within is being replaced by a different page.

 @override

  void initState() {

    super.initState();

   print("init State Called");

  }

     @override

     Widget build(BuildContext context) {

      print("Build State Called");

      return Widget;

    }

  @override

  void dispose() {

    super.dispose();

    print("Dispose State Called");

  }

====================================================================================================

async AND "await" function Keywords

Asynchronous operation means run a specific function on a separate thread. it is annotated with keyword "async"

That means such function will run on a background thread without blocking other functions from executing

"await" Is used to say something like, "hei, wait for this function to complete before you execute the function after it"

The reason is that the function after it may expect a parameter from it

Example 1.

class \_ChooseLocationState extends State<ChooseLocation> {

//”async” means Run this "getData()" method on a background thread

  void getData() async{

    String name;

    // run this method on a background thread after 3 seconds

    /\*  "await" means => waite for this method to finish executing before you call the next method below it

        the reason is that the method below needs the value of the "name" variable

    \*/

*await Future.delayed(Duration(seconds: 3),(){*

*print(" 3 seconds");*

*name="justice";*

*});*

    // run this method on a background thread after 2 seconds

    Future.delayed(Duration(seconds: 2),(){

      print(" 2 seconds" + name);

    });

  }

  // call the "getData()" method immediatly this page is loaded

  @override

  void initState() {

    super.initState();

    getData();

  }

}

Example 2.

class \_ChooseLocationState extends State<ChooseLocation> {

// Run this "getData()" method on a background thread

  void getData() async{

    // run this method on a background thread after 3 seconds

    /\*  "await" means => waite for this method to finish executing before you call the next method below it

        the reason is that the method below it expect some value from it

        Then assign the return value "justice" to the variable "userName"

    \*/

  String userName = await Future.delayed(Duration(seconds: 3),(){

         return "justice";

    });

    // run this method on a background thread after 3 seconds

  /\* 'await' means => waite for this method to finish executing before you call the next method below it \*/

  String userAge = await Future.delayed(Duration(seconds: 2),(){

     return "name: " + userName + " age: 21";

    });

  // print "getUserName variable"

  print(userAge);

  }

  // call the "getData()" method immediatly this page is loaded

  @override

  void initState() {

    super.initState();

    getData();

  }

}

====================================================================================================

**Flutter Packages….**

http package

Is used for making http request. Especially for making API calls

First add the dependency to your pubspec.yaml file in the dependency section (Remember where the indentation is):

Then after it, click on “pub get” pop up alert on the pubspect.yaml to download the dependency to your project

dependencies:

flutter:

sdk: flutter

**http: ^0.13.3**

Then inside, main.dart. Import the package and make some http request...

import 'package:http/http.dart';

class \_LoadingState extends State<Loading> {

  //async means 🡺 Run this "getData()" method on a background thread

  void getData() async{

    /\*

    "Response" object represent the response/data coming from the server

Await = means wait for below response to be fetched before executing the next block of code

    \*/

     Response response= await get(Uri.parse("https://jsonplaceholder.typicode.com/todos/1"));

     /\*

     The format is in json, So convert/decode it to a map

“body” represents body of the response not the headers of the response

     \* \*/

     Map<dynamic, dynamic> data = jsonDecode(response.body);

     /\*  Then grab this title value

     userId : 1

     id : 1

     title :  "delectus aut autem"

     completed : false

     \*/

       print(data["title"]);

  }

  // call the "getData()" method immediately this page is loaded

  @override

  void initState() {

    super.initState();

    getData();

  }

}

====================================================================================================

desktop\_window package

Is a dependency for setting maximum and minimum width and height for your application when your app is compelled to run on desktop platforms like windows, Linus, and MacOs.

It does not include web and mobile platforms (That’s when your app is running on a browser or mobile device)

So even if the user is running your app inside a desktop browser, this dependency will have no effect. Because is used to set maximum and minimum width and height when creating a flutter desktop app like vlc media player.

First add the dependency to your pubspec.yaml file. Then after it, click on “pub get” pop up alert on the pubspec.yaml to download the dependency

dependencies:

**desktop\_window: ^0.4.0**

Then inside, main.dart. Import the package and check if the app is compiled to run on desktop OS and change the min&max height and width

import 'package:desktop\_window/desktop\_window.dart';  // import the dependency

// inside the main method, set your app desktop min/max size

void main() async{

  /\* check if the app is running in desktop os and set the minimum and miximum window zise for your app

     The setMinWindowSize() & setMaxWindowSize() returns a dart "Future"

     So am appending the "await" keyword to the methods and also "async" to the above "void main()" method

  \*/

  if(!kIsWeb && (Platform.isWindows || Platform.isMacOS || Platform.isLinux)){

   await DesktopWindow.setMinWindowSize(Size(600, 800));  // set minimum size

   await DesktopWindow.setMaxWindowSize(Size(120, 160));  // set maximum size

  }

  runApp(MyApp());

}

====================================================================================================

custom\_clipper Package

is a package that helps you to achieve various shapes. You can use it to clip part of a rectangle or any shape to achieve something.

You just need to explore their documentation on for various clipping shapes you can use on:

https://pub.dev/packages/flutter\_custom\_clippers and also check the "api reference link" on the website to see how to use it shapes.

You can't learn all the shapes, they just work in similar way so check their api documentaion above.

First you need to add the dependency below to your pubspec.yaml file:

flutter\_custom\_clippers: ^2.0.0 // or check for latest version

and import below code in your main.dart file:

import 'package:flutter\_custom\_clippers/flutter\_custom\_clippers.dart';

/\* All you need to do, is to wrap the widget you want to clip it part with the  "ClipPath() widget" and use "clipper:" to decide which type of clip you want:

There are a lot of "clipper" on the documentaion you should check out \*/

 @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(title: Text("welcome page"),),

      body: ListView(

          padding: EdgeInsets.all(20.0),

          children: <Widget>[

            // use the clipPath widget

      ClipPath(

        // the type of shape of clip you want to have on the child widget

      clipper: SideCutClipper(),

      // the child widget to clip part of it

      child: Container(

        height: 600,

        width: 500  ,

        color: Colors.pink,

        child: Center(child: Text("SideCutClipper()")),

      ),

    ),

    ClipPath(

    clipper: MultipleRoundedCurveClipper(),

    child: Container(

    height: 100,

    color: Colors.pink,

    child: Center(child: Text("MultipleRoundedCurveClipper()")),

    ),

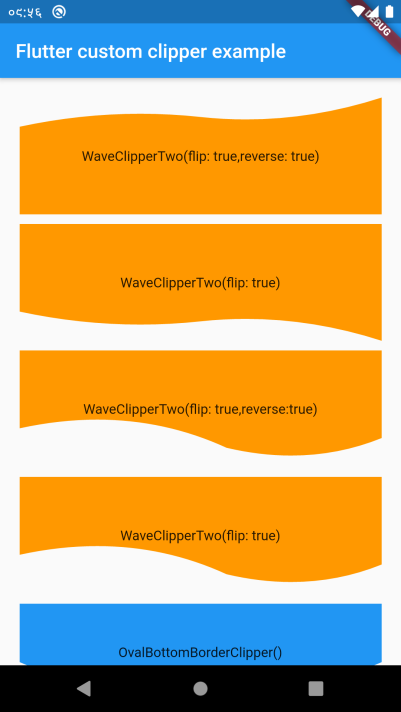
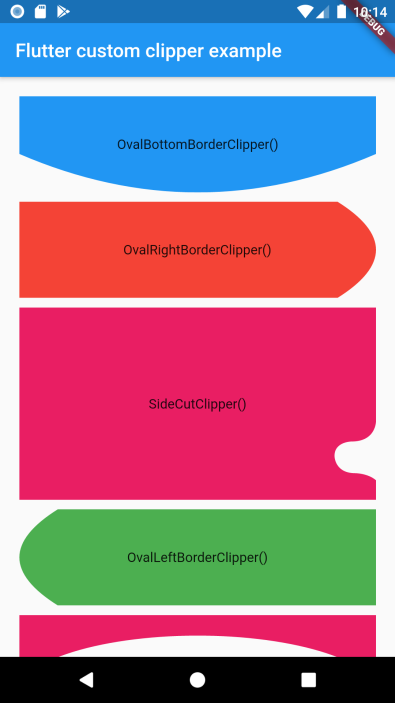
    ),

        ]

    )

    );

  }



**====================================================================================================**

How To Create Splash Screen

When you first start your flutter app, you will see some white blank page before the flutter engine loads your app to the native device. You can change the color and add your app log to it.

flutter\_native\_splash plugin will allow you to do this easily.

First add the plugin to your pubspec.yaml file as a de\_dependency and specify the color and logo you want just as I have done below. Then run the command below in the terminal:

dev\_dependencies:  
 flutter\_test:  
 sdk: flutter  
 flutter\_native\_splash: ^1.2.1 // add the plugin dependency here as a dev\_dependency  
  
  
*// splash screen properties (the plugin is flutter\_native\_splash: ^1.2.1 in above dev\_dependencies)*flutter\_native\_splash:  
 color: "#42a5f5" // the color you want for your splash screen  
 android: true  
 ios: true  
 web: true  
 image: assets/images/logo.png // your app logo, it will be at the center automatically

Then open the terminal/cmd and navigate to your project directory and these in the same order:

* flutter clean
* flutter pub get
* flutter pub run flutter\_native\_splash:create

// then run your application and you will see the splash screen.

================================================================================================

**END OF MY Flutter Packages….**

**====================================================================================================**

How To Remove The Debug Mode Banner From Your App

To remove the debug mode banner from your app, set the “debugShowCheckedModeBanner” name parameter of “MaterialApp” class constructor to false.

Example:

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      debugShowCheckedModeBanner: false, // remove the debuge mode banner from the app

      theme: ThemeData(

        primarySwatch: Colors.white,

      ),

      home: Home(),

    );

  }

}

================================================================================================

Future Builder

Future builder is a widget that assigns the return value of a function that performs asynchronous operation like http request to a widget like a Text().

The reason is that, all async functions returns a dart Future<T> and you cannot directly assign this type to a Widget. And does where the Future builder come in.

**Example:**

// define a function that perform aynchronous http request

  Future<String> checking()async{

    // let say this "b" is the response of the http request

      String b = "hi";

      // return this to any function that calls this future method

       return b;

    }

    // Then inside your build method, assign the above future return value to a Text Widget

     Center(

       // make sure the return-Type is equal to the above function return Type: FutureBuilder<String>

        child:  FutureBuilder<String>(

            future: checking(),  // the above function name

            // snapshot represent the object of the above funtion value. (use: snapshot.data to get the real value)

            builder: (BuildContext context, AsyncSnapshot<String> snapshot){

              return Text(snapshot.data);

            }

        ),

     )

================================================================================================

ListView Widget

Listview is a way of displaying scrollable list of items. It just like listView in android. It has this special parameter:

itemCount: represent the length of the items in your List data.

itemBuilder: This is a method where you build the different list of your widget. (it takes in the widget context, and an index that represent each position in your item)

     home: Scaffold(

        body: ListView.builder(

            physics: ClampingScrollPhysics(),  // add it to make scrolling possible all the time

            padding:   EdgeInsets.all(8),

            scrollDirection: Axis.vertical,

            shrinkWrap: true,

            itemCount: colors.length,  // the length of items

            /\*

             This method is where you build the list of your widget. It takes in two parameters:

             "context" the widget context (widget context means the state of the widget)

             "index" this is an integer parameter that represent each index of the items in your list data or array.

            \*/

            itemBuilder: (context, index){

                return Container(

                  margin: EdgeInsets.all(5),

                  height: 50,

                  color: Colors.grey,

                  child: Text(colors[index]), // the index of different colors in the color list.

                );

            }

        )

      ),

================================================================================================

How to perform if statement in a widget constructor and do something with that widget properties

You can perform some if statement in all widget constructors to change their properties.

Example: Change the color of the selected item in a listview based on if statement.

class \_MylistState extends State<Mylist> {  
 Mymodel mymodel = new Mymodel(); // class that contains list of colors (I have not added it here)  
  
// define a variable to represent the index of the select item in the list  
 int selectedIndex = 0;  
  
 // a function to change the above "selectedIndex" index when that list is clicked  
 void onSelected(int index) {  
 setState(() => selectedIndex = index);  
 }

@override  
Widget build(BuildContext context) {  
 return Column(  
 children: [  
 ListView.builder(  
 shrinkWrap: true,  
 itemCount:mymodel.colors.length,  
 itemBuilder:(context, index){  
 return InkWell(  
  
 child: Text(  
 mymodel.colors[index], // don't mind it, is myModel is a list of colors  
 // if \_selectedIndex not null and is equal to the selected list, set it text color to red. else color is black  
 style: TextStyle(color: selectedIndex != null && selectedIndex == index  
 ? Colors.*red* : Colors.*black87*,) ),  
  
 onTap: (){  
 // call the onSelected() method above to set the current selected items index to it  
 onSelected(index);  
 },  
 );  
 },  
 ),

}

}

================================================================================================

DataTable Widget

This is that is used to show data in a table. Always wrap it around singleChildScrollview Widget for nice horizontal scrolling

Example:

    home: Scaffold(

      body: SingleChildScrollView(

       child:DataTable(

         showBottomBorder: true,  // show a horizontal border-line at the bottom of the table

         // give a Textstyle to the below headings (columns[....])

        // headingTextStyle: TextStyle(color:Colors.blue,backgroundColor: Colors.blueGrey),

         // give a Textstyle to all the datas in the table excluding heading

         // dataTextStyle: TextStyle(color:Colors.white,backgroundColor: Colors.black),

         // define list of heading for your table

         columns: [

           DataColumn(label: Text("Names")),

           DataColumn(label: Text("Age")),

         ],

         // List of datas that will go under above headings column

         rows: [

           // first data

           DataRow(

             cells: [

             DataCell(Text("justice")),

                // showEditIcon: true,   // show and editable icon to indicate that this cell is editable

               //  placeholder: true),  // display this cell as placeholder

             DataCell(Text("24")),

           ],

           //  selected: true,  // mark this row as the selected one

           ),

           // second data

           DataRow(

               cells: [

             DataCell(Text("Albert")),

             DataCell(Text("19")),

           ],

           ),

         ],

       ),

     ),

    );

HOW TO LOOP THROUGH DATA USING MAP AND ASSIGN IT TO DATA-TABLE

    child: DataTable(

        // heading of the table

          columns: [

            DataColumn(label: Text("TITLE", style: Theme.of(context).textTheme.subtitle2)),

            DataColumn(label: Text("ARTIST")),

            DataColumn(label: Text("ALBUM")),

            DataColumn(label: Icon(Icons.access\_time)),

          ],

// TABLE DATA:  loop through the data

        rows: tracks.map((e) {

          return DataRow(

            cells: [

              DataCell(

                Text(e.title, style: Theme.of(context).textTheme.subtitle2),

              ),

              DataCell(

                Text(e.artist, style: Theme.of(context).textTheme.subtitle2),

              ),

              DataCell(

                Text(e.album, style: Theme.of(context).textTheme.subtitle2),

              ),

              DataCell(

                Text(e.duration, style: Theme.of(context).textTheme.subtitle2),

              ),

            ],

          );

        }).toList(),

      ),

================================================================================================

ClipRRect Widget

The cliprrect widget is used to add border-radius around other widget.

     home: Scaffold(

        body: ClipRRect(

                  borderRadius: BorderRadius.circular(50), // the radious number

                  child: Container(  // the widget to give it a radius

                    height: 50,

                    child: Text("Helloo Word"),

                  ),

                );

      ),

================================================================================================

CIRCULAR PROGRESS INDICATOR widget

This is used to show a loading progress in a page.

     home: Scaffold(

        body: SafeArea(child: CircularProgressIndicator(

                  backgroundColor: Colors.blue,

                ));

      ),

Example: To show loading progress while page is loading data from server

class \_HomeState extends State<Home> {

// Set loading to true to make the progress  indicator scroll

  bool \_loading =true;

  //initstate Method (called only once, immediately this page load)

  @override

  void initState() {

    super.initState();

  // method to perfrom load data from server

  void getNews() async {

    News news = News();

    await news.getNews();

    newslist = news.news;

    // set loading to false when the data is loaded from the server to hide the loading indicator

    setState(() {

      \_loading = false;

    });

  }

  }

  // Build method (called after iniState() and whenever setState() method is called to update widget values)

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(.... ),

      body: SafeArea(

        child: \_loading  // if loading is true, display a loader, else display the content

            ? Center(

          child: CircularProgressIndicator(),

        ) :

        SingleChildScrollView(

          child: Container(.....)

        ),

  }

}

================================================================================================

PageView Widget

PageView is a widget used to scroll between widget or Pages. It just like viewpager2 in android.

The PageView constructor takes in one particular parameter called “pagecontroller”

Pagecontroller is used to control which page to show when the app load first & which page to show when the user scrolls.

Pagecontroller has a method called…

controller.animateToPage(…) 🡺 which is a function that takes in the page index to scroll to, duration, and it animation (calling this function will scroll to the exact page indexed passed into it )

The PageView constructor takes in another parameter “onPageChanged” method which will be called whenever the page changes

Example:

class \_MyHomePageState extends State<MyHomePage> {

// set a variable to represent the page index (i will change it value when the page changes)

   int pageIndex = 0;

   /\* PageController it used to set the page number that should be shown

  when you scroll the page, this number will be increased automatically

  it "initialPage" parameter is used to set which page should load first when the app runs\*/

   final PageController controller = PageController(initialPage: 0);

  @override

  Widget build(BuildContext context) {

   return ( Scaffold(

     // Create an appBar with floating button that will be clicked to scroll to different page

     appBar: AppBar(title: Text("my PageView"),centerTitle: true,

     actions: [

       FloatingActionButton(

         child: Text("scroll"),

         onPressed: (){

             /\* controller.animateToPage() calling this function will navigate to a different page

                it takes in 3 parameters:

                1st ==> the page index to animate to

                2nd ==> the duration to animate to the page, use Duration(milliseconds: 300)

                3rd ==> curve, the animation style to use to animate to the page

                Note ==> am increasing the pageIndex so that it will navigat the user to the next page when it clicks

              \*/

             controller.animateToPage(++pageIndex, duration: Duration(milliseconds: 300), curve: Curves.easeIn);

         }

         )

     ],),

     // the body of the pages

     body: Column(

       children: [

         Expanded(

           child: PageView(  // the pageview widget

                 controller: controller,   // set the controller which controlls which page to show

                 scrollDirection: Axis.horizontal,  // the scroll direction (horizontal or vertical)

                 /\* List of different pages that return widget. here am only showing just widget

                  But in real world, it should be different pages that return widget\*/

                 children: [

                   Text("page one"),

                   Text("page 2"),

                   Text("page 3"),

                 ],

                // method to be called whenever a page changes

               // it takes in one custom parameter which represent the current page index or number

                onPageChanged: (index){

                   // set the pageIndex to the index of the page

                   pageIndex=index;

                },

               ),

         ),

       ],

     ),

   ));

  }

}

================================================================================================

Shared preference

Share preference allows you to store data in key value pairs in the user phone. When your app get uninstalled, this data will be deleted.

So don't use shared preference to store users data's that you will need in the future for further processing.

Note: You need to stop the debugging after writing shared preference code, then start again before it will work.

First add the shared preference dependency to your pubspec.yaml file:

dependencies:

flutter:

sdk: flutter

shared\_preferences: "<check newest version>"

SAVE DATA

First get the object of the shared prefernece. And depending on the type of data you want to save, use any of these methods

setInt(), setBool(), setString() = each methods takes in the keyName, Value Ex:

// get shared preferences bellow code is inside a method that is annotated with async

final prefs = await SharedPreferences.getInstance();

// use the object to save or set value

prefs.setInt('counter', 5);

RETRIEVE/GET DATA

To retrieve a particular data, use the shared preference object with any of it methods getInt(), getBool(), getString() depending on the

type of the data. Each methods takes in the key of the value that was used to store the data. And you can add "?? valueToReturn" at

the end of any of these above methods to use this value if the main value is not found. Ex:

// use the shared preference object bellow code is inside a method that is annotated with async

final prefs = await SharedPreferences.getInstance();

// get data from the "counter" key. If it doesn't exist, return or use default value 0.

final counter = prefs.getInt('counter') ?? 0;

REMOVE DATA

to remove data, use the shared preference remove() method. It takes in the key of the data you want to remove. Ex:

bellow code is inside a method that is annotated with async

final prefs = await SharedPreferences.getInstance();

prefs.remove('counter');

CHECK IF A KEY EXIST

Use the containsKey() method to check if a particular key exist or not.

final sharePrefs = await SharedPreferences.*getInstance*();

if(sharePrefs.containsKey(“username”))  
 // Do something here  
  
}

================================================================================================

Media Query

Media query help you to create responsive app for different divices.

Using *mediaQuery.of* automatically causes the widgets to rebuild themselves according to the current device sizes and layout every time they change.

Format: The formmat is that, You just get the height or width of the user device and adjust your widget height and width based on the user

Device size.

Example. get the width of the user device and if it is 100. And you have 3 boxes widget give each a width of 30% of the device. So if you run the app on

different device of 200 width, each of your widget will have 30% of the device height.

Note: to be able to use mediaquery, you need to import this in your

‘package:flutter/material.dart’;

MediaQuery.of(context).size.width // get width of the device

MediaQuery.of(context).size.height // get height of the user device

MediaQuery.of(context).size / / to get the object of both size (width & height for further processing)

context above just represent the "life cycle or life" of the screen.

Remember:

You can deduct the statubar height, appbar height and the bottom Nav height from the app body content to get only the height of the app body.

so your app height will be (statusbar height + appbar height + bottom nav height + app body height)

appbar height = kToolbarHeight

statusbar height = MediaQuery.of(context).padding.top

bottom nav height = kBottomNavigationBarHeight

body height = MediaQuery.of(context).size.height

 // To get the height of the body content, you just need to do:

// app body height - statusbar height - appbar height -bottom nav height

MediaQuery.of(context).size.height - MediaQuery.of(context).padding.top - kToolbarHeight - kBottomNavigationBarHeight

//To get the height of the status bar

// statusbar height -  app body height - appbar height -bottom nav height

MediaQuery.of(context).padding.top - MediaQuery.of(context).size.height - kToolbarHeight - kBottomNavigationBarHeight

// use the same logic to retrieve any single of the above height.

**Example::**

/\* Create a seperate dart file "sizes\_helpers.dart" the name can be anything and put in below code.

Am just creating a method that return the user device size object, widht and height.

and am also printing the user device size to the terminal for debuging purpose. \*/

  //inside sizes\_helpers.dart

import 'package:flutter/material.dart';

// method to return the size object of the user screen

Size displaySize(BuildContext context) {

     print('Size = ' + MediaQuery.of(context).size.toString());

  return MediaQuery.of(context).size;

}

// method to return the height of the user screen

double displayHeight(BuildContext context) {

  print('Height = ' + displaySize(context).height.toString());

  return displaySize(context).height;

}

// method to return the width of the user screen

double displayWidth(BuildContext context) {

  print('Width = ' + displaySize(context).width.toString());

  return displaySize(context).width;

}

/\*

Then inside main.dart import the above file. And adjust your widget size automatically based on the user device size

 \*/

  // inside main.dart

import 'package:flutter/material.dart';

import 'package:sizes/sizes\_helpers.dart';

class \_HomePageActivityState extends State<HomePageActivity> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text("Home page"),),

    body: Center(

    child: Column(

      children: <Widget>[

        Container(

    color: Colors.red,

      width: displayWidth(context) \* 0.25, //displayWidth() represent the widht of the use device from the sizes\_helpers.dart file

      child: Text(

        'Box width 25% of screen width and text size 3% of screen width',

        textAlign: TextAlign.center,

        style: TextStyle(fontSize: displayWidth(context) \* 0.03),

      ),

    ),

        Container(

          color: Colors.green,

          width: displayWidth(context) \* 0.5,

          child: Text(

            'Box width 50% of screen width and text size 6% of screen width',

            textAlign: TextAlign.center,

            style: TextStyle(fontSize: displayWidth(context) \* 0.06),

          ),

        ),

        Container(

          color: Colors.blue,

          width: displayWidth(context),

          child: Text(

            'Box width equal to screen width and text size 10% of screen width',

            textAlign: TextAlign.center,

            style: TextStyle(fontSize: displayWidth(context) \* 0.1),

          ),

        ),

      ]

    ),

    ),

    );

  }

}

/\*

Note the figures, multiplying it by the user device will give:

0.50 is the same as 0.5 = 50%

0.06 = 6%

0.6 is the same as 0.60 =50%

And…..

    userdeviceheight/2,  //half of the user device height size

        userdevicewidth: width/2,  //half of the user device width size

\*/

Device Orientation

You can also use media query to get the orientation of the user device and adjust your widget accordingly.

**Example:**

/\*

You can also get the user device orientation do some changes to your widget

their orientation changes.

\*/

class Home extends StatelessWidget {

var orientation, size,height,width;

  @override

  Widget build(BuildContext context) {

    // getting the orientation of the app

    orientation = MediaQuery.of(context).orientation;

    //size of the widow

    Size size = MediaQuery.of(context).size;

   double height = size.height;

    double width = size.width;

    return Scaffold(

      appBar: AppBar(

        title: Text("Geeks For Geeks"),

        backgroundColor: Colors.green,

      ),

      // checking the orientation and reunnng if statments and adjusting widget sizes

      body: orientation == Orientation.portrait?Container(

        color: Colors.blue,

        height: height/4,

        width: width/4,

      ):Container(

        height: height/3,

        width: width/3,

        color: Colors.red,

      ),

    );

  }

}

================================================================================================

**Resources**

* <https://pub.dev/> == to see all dart packages you can use in flutter.

If you see any particular plugin you want to use, click on it. And on your right side you will see “API reference” link to learn how to use the plugin Or (check the link under “Libraries” after clicking the “api reference” link ). And on top side, you will see “readme”, “installation” and more on how to install and use the app.

* <https://flutter.dev/docs> ==== To expolore and use the flutter documentation

================================================================================================

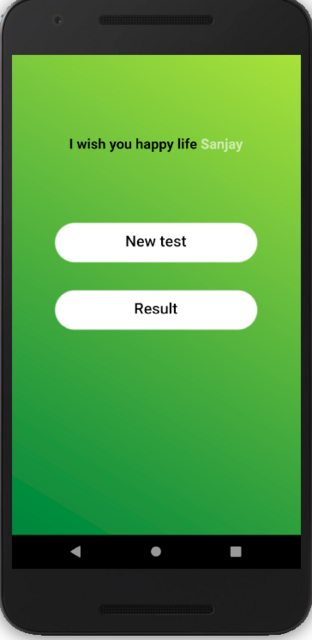
How To Hide and show the statusbar

First import below code and read to the end of this tip before you take action.

import 'package:flutter/services.dart';

**1.Hide Statusbar**

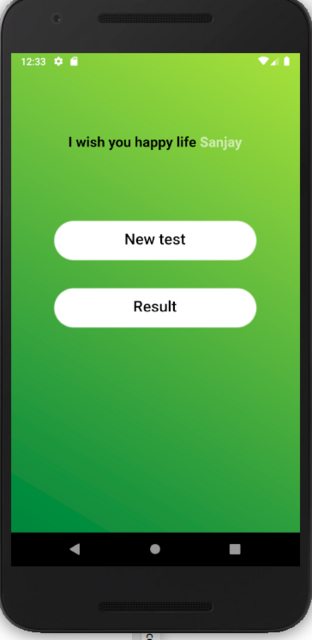
SystemChrome.*setEnabledSystemUIMode*(SystemUiMode.manual, overlays: [SystemUiOverlay.bottom]);

**[](https://i.stack.imgur.com/E1fUvl.png)**2. Transparant Statusbar

SystemChrome.setSystemUIOverlayStyle(SystemUiOverlayStyle(

statusBarColor: Colors.transparent,

));

[](https://i.stack.imgur.com/qciMSl.png)

3.Show Statusbar

SystemChrome.setEnabledSystemUIOverlays(SystemUiOverlay.values);

**You need to put this code in :**

1.For Single Screen

@override

void initState() {

// put it here

super.initState();

}

@override  
void dispose() {  
 SystemChrome.*setEnabledSystemUIMode*(  
 SystemUiMode.manual, overlays: [SystemUiOverlay.top, SystemUiOverlay.bottom]);  
 super.dispose();  
}

2.For All pages in main.dart:

void main() {

// put it here

runApp(...);

}

================================================================================================

Force device to stay in portrait or landscape mode when device orientation changes

Put all below code in build() method.

* Force portrait mode

*// force the app to stay to protraite mode when device orientation changes*SystemChrome.*setPreferredOrientations*([DeviceOrientation.portraitUp,DeviceOrientation.portraitDown]);

* Force landscapeLeft mode

SystemChrome.setPreferredOrientations(DeviceOrientation.landscapeLeft,DeviceOrientation.landscapeRight])

================================================================================================

Routing

The Navigator.pushNamed() method will put another screen on top of the other so that when the user clicks the back button, he can be able to go back to the previous screen.

The Navigator.ReplacementNamed() method will replace the current screen with the new one. When the user clicks the back button, he can’t go back to the previous screen.

//inside main.dart

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  // This widget is the root of your application.

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

  // route

      initialRoute: "/", // app starting route

      routes: {

        "/":(context) => HomeActivityState(),  // app starting route url

        "/aboutus" :(context)=>Aboutus(),  // about us page ulr

        "/contactus": (context)=> Contactus(), // contact us page url

      },

      debugShowCheckedModeBanner: false,

      // home: HomeActivity(), // home screen activity (comment this since starting up url is set above)

    );

  }

}

// then inside other activities.dart widget

 InkWell(

        child: Text("About", style: TextStyle(fontSize: 15, color: Colors.grey)),

               onTap: (){

                 // go to about us page when this widget is clicked. there will be automatice back-icon on the appbar on about us page

              Navigator.pushNamed(context, "/aboutus");

     }

 )

================================================================================================

How To Break a text in the Text() Widget

To break a text in the Text() widget just use “\n” it just the same as the HTML <br/>.

**Example:**

Text(“my name is: \n Justice Ankomah”);

Result:

My name is:

Justice Ankomah

================================================================================================

How To Navigate between screens automatically

Just put this below code in the initState() method. Then set the time you want to delay before navigating to the other page. And pass in the other page url. Ex:

*// init method*@override  
void initState() {  
*// put this code in your*   
 Future.delayed(Duration(seconds: 4),()async{  
 Navigator.*pushNamed*(context, "/homepageactivity");  
 });  
 super.initState();  
}

================================================================================================

How To Align And Arrangement Widget Like the css: position property and margin.

Use the Align() or position() Widget to align element any way you like. These two widget are the best ways to align element anyhow inside and outside parent widget. They takes in negatives values too.

Check how to use it if you want to align elements anyhow.

Note: When any of these above widget is not having effect on your other widget. It means there may not be enough space for the widget to move to.

Use multiple stack(), positioned(), Align(), SizeBox() to play around element in stack and you can achieve whatever layout you want the css position:absolute property will have.

Just take your time and play around with these widget, when you are not getting the layout you want.

================================================================================================

Sqlite & sqflite Database

Sqlite database helps you to store large structural data locally on the user device. The flutter sqflite plugin/packge available on pub.dev provides class and methods to helps you work with sqlite. So sqflite is just a package that helps you to work with sqlite, it not a different database on its own.

If you want to store some small data in key-value-pairs then consider using shared preferences.

Sqlite DataBase DataTypes

Int in dart is referred as Integer in sqlite

String in dart is referred as Text in SQLite

First Add the dependency to your pubspect.yaml file:

sqflite: ^2.0.0+4

Then to be able to write and read from the database. You need to create a database and set up connection.

Example:

// Inside DineWithDbClass

import 'package:flutter/material.dart';

import 'package:sqflite/sqflite.dart';

class DineWithDbClass {

  // name of your database table

  String tableName="recentlyviewedItemsTable";

  // define a database variable

  Database \_dineWithJulDb;

  // this is the method that others classes will call to get opened database connection

  // define a singleton get method for the above \_dineWithJulDb databse to inizilate it when it is null

  Future<Database> get dineWithJulDb async {

    if (\_dineWithJulDb == null) {

      // if the database is null, initialize it by calling the below method

      \_dineWithJulDb = await initializeDatabase();

    }

    // return an opened "dinewithjuldb.db" database conection to class who calls this method

    return \_dineWithJulDb;

  }

  // custom method that create and open a database connection

  Future<Database> initializeDatabase() async{

    // get the folder where databases are stored on the user phone

    var dir = await getDatabasesPath();

    // create a file called "dinewithjuldb.db" in the database directory folder in the user device

    var path = dir + "dineWithjuldb.db";

    // open a database connection

  var database = await openDatabase(

      path,  //the above path/directory of the database to open

      version: 1,  // the version of your database

      //onCreate is the parameter that takes in a function to create a database and it version

      // db = databaseName

      // verion = database version

      onCreate: (Database db, int version) async{

            // When creating the db, create below table in it

            // the ''' helps to type string in multi-line

          await db.execute('''

            create table $tableName(

              id integer primary key autoincrement,

            itemname text not null,

            itemid integer

            )

          ''');

      },

  );

  // return an opened "dinewithjuldb.db" database conection to the above "get dinewithJulDb" method

  return database;

  }

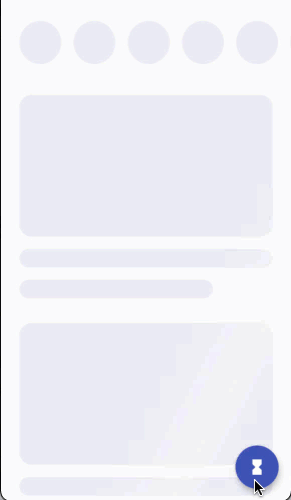
}

================================================================================================

**Shimmer Effect**

Is a way of showing nice beautiful animation with effect while data is loading from the server.

Look at below image, it shows while data is loading from the server, it comes with an animation but it not visible this image. So you can check online for how it looks actually.



Draw the shimmer shapes

The shapes to show while data is loading from the server is different. That’s, not what the contents will be rendered with when it fully loaded.

So the goal is to create shapes or UI with light gray shapes and apply the shimmer effect on it. while contents is loading from the server. The shapes doesn’t have to match exactly the UI of your widget, just something similar.

================================================================================================

FORMS AND TextFormField

In flutter, every form you create must have its own special “key” for tracking such particular form. The reason is that there may be 2 or more forms in the same screen.

Also, Each TextFormField must have it own special controller for tracking it.

The Form is just like any other widget. And all you need is to create one and wrap all it TextFormField in a Column or a Row.

**Example:**

// put these 2 properties in class as variables, their not widget

 // A unique key for this form only

  final loginFormKey = GlobalKey<FormState>();

    //====controllers for formtextfield======

  //access it to get each inpute field value with: phoneController.value.text.toString()

  final phoneController = TextEditingController();

// add Form to any of your widget

Form(

                // add the form key

                key: loginFormKey,

                // Wrap all your forminput either in Column or Row

                child: Column(children: [

                  //Phone Number TextFormField

                  TextFormField(

                      controller: phoneController, // assign the above phoneController to it

                      style: TextStyle(

                        color: black,

                      ),

                      cursorColor: Colors.black,

                      // this represent the text type either emailAddress, text, number, multiline

                      keyboardType: TextInputType.number,

                      // InputDecoration is where you have to style the Form inpute fields

                      decoration: InputDecoration(

                        focusColor: ColorsData.black,

                        hoverColor: ColorsData.black,

                        prefixIcon: Padding(

                          padding: const EdgeInsets.only(left: 8.0),

                          // add icon to the inpute filed

                          child: Icon(

                            Icons.call,

                            color: Colors.black,

                            size: 40.0,

                          ),

                        ),

                        labelText: "Phone Number",

                        labelStyle: TextStyle(

                          color: Colors.black,

                          fontFamily: "Sixtwo",

                        ),

                        hintText: "required",

                        border: OutlineInputBorder(

                          borderRadius: BorderRadius.circular(50),

                          borderSide: BorderSide(color: Colors.black, width: 3),

                        ),

                        contentPadding: EdgeInsets.all(2),

                        focusedBorder: OutlineInputBorder(

                          borderRadius: BorderRadius.circular(50),

                          borderSide:

                              BorderSide(color: Colors.lightBlue, width: 3),

                        ),

                      ),

                      // Use validator to validate the inpute form "phoneValue" is the text the user will type in the input.

                      // return null if the valiation passes

                      validator: (String phoneValue) {

                        if (phoneValue.isEmpty || phoneValue == null) {

                          return "This field is required";

                        } else if (phoneValue.length < 10 ||

                            phoneValue.length > 10) {

                          return "Phone Number must be 10 digits";

                        } else {

                          return null;

                        }

                      }),

                  // add another TextFormField Here If You Want

                  // Submit button, you can move this out of the column if you want

                  ElevatedButton(onPressed: () {

                    //Check using the form Key to see If all the validation performed in the "TextFormField" are correct and do something

                    if (loginFormKey.currentState.validate()) {

                      // if validation is successful

                    } else {

                      // if validation is not successful

                      return;

                    }

                  }),

                ])),

================================================================================================

HOW TO CREATE A STATEFUL DIALOG

All Dialogs in flutter are stateless by default. Meaning, when your dialog is started, When any value changes it will not reflect because it is stateless.

You can fix this by passing a “BuildContext” as a parameter to the dialog method and by wrapping the dialog contents with statefulBuilder.

**Example:**

// Create a method with "BuildContext context" as a parameter that will be called by widget

updateUserAddress(BuildContext context) {

  // the dialog to show when the method am inside is called (showDialod method belows is already createdy by flutter)

  showDialog(

    context: context,

    builder: (context) {

      // Wrap your AlertDialog in a StatufulBuilder

      return StatefulBuilder(builder: (context, setState) {

        return AlertDialog(

          // Title of the dialog

          title: Text('Hi Welcome!'),

          // The main message or content of your dialog

          content: Column(children: [

            Text("content one"),

            Text("content two"),

          ]),

// Cancel & Confirm action buttons

          actions: [

            // cancel button

            TextButton(

              //  Navigator.pop(context), will exit the dialog

              onPressed: () => Navigator.pop(context),

              child: Text(

                'Cancel',

              ),

            ),

            // okay button

            TextButton(

              onPressed: () {},

              child: Text('Confirm'),

            )

          ],

        );

      });

    },

  );

}

// Then inside the build method call the above dialog method to execute the dialog

@override

Widget build(BuildContext context) {

  return ElevatedButton.icon(

      onPressed: () {

        // call the method by passing in the context

        updateUserAddress(context);

      },

      label: Text("Show Dialog"));

}

================================================================================================

HOW TO USE THEMES

We mainly use themes to share colors and font-styles throughout the app. This helps to maintain code and if any changes needs to be done, you just have to adjust it in a single place.

All you need to do is to define some ThemeData and apply it to any widgets you want in your app.

**Example:**

// Inside the material widget in main.dart define your themes in the ThemeData

 MaterialApp(

      title: 'Flutter Demo',

      // This "ThemeData" overrides Theme.dart that represent all the themes you can use

      theme: ThemeData(

        // define a primary color

        primaryColor: Colors.blue,

        // Create A text Themes

        textTheme: TextTheme(

          // Single TextTheme (headLine1 is already defined)

          headline1: TextStyle(

            color: Colors.red,

            fontSize: 15,

            fontWeight: FontWeight.bold,

          ),

          // You can add another textStyle here if you want

        ),

      ),

    );

    // Then apply the above theme data to any of your app widget.

    // using:  Theme.of(context).....

      ListTile(

              // apply the primaryColor specified in the theme

              tileColor: Theme.of(context).primaryColor,

              title: Text("helloo",

                // apply the textTheme "headline1" specified in the theme to your text style

                style: Theme.of(context).textTheme.headline1,

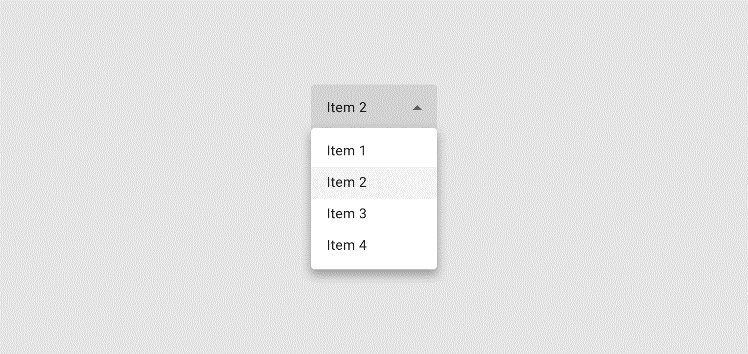
              ),

            ),

================================================================================================

Dropdown-Menu

A dropdown-menu represent a list of menu items that will be shown when a dropDownBotton is clicked.



**Example:**

class JusticeAPP extends StatefulWidget {

  const JusticeAPP({Key? key}) : super(key: key);

  @override

  \_JusticeAPPState createState() => \_JusticeAPPState();

}

class \_JusticeAPPState extends State<JusticeAPP> {

   // dropDown value

  String? dropDownValue;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body:   // DropDownButtons represent the buttons that will be clicked to open the dropDowmMenuItem.

              // DropdownButton is a generic class DropdownButton<T>(). The generic <T> represent what dataType the dropDown value is going to be?

              // here am specifying String

              DropdownButton<String>(

                // the Default value for your dropdown ()

                value: dropDownValue,

                  // the hint text to show on the DropdownButton

                  hint: Text("choose color"),

                // <underline> Underline the DropdownButton. If you don't need any underline, just use:

                //   underline: SizedBox(),

                  underline: Divider(color: Colors.black,),

                // Icon to show on the DropdownButton

                icon: Icon(

                  Icons.more\_vert,

                  color: AppColors.white.withOpacity(0.7),

                  size: displayUserDeviceWidth(context) \* 0.07,

                ),

                  // onChanged function means, do something when user select different DropdownMenuItem

                  // so this is where to call other methods when some DropdownMenuItem is clicked

                  onChanged: (String? newValue){

                    log("$newValue");

                  setState(() {

                    dropDownValue = newValue;

                  });

                  },

                  // List of dropDownMenuItem to show when the DropdownButton is clicked

                  items:[

                    // Each DropdownMenuItem should have a <value> and child widget

                    DropdownMenuItem(value:"blue", child: Text("Blue")),

                    DropdownMenuItem(value: "red", child: Text("Red")),

                  ]

              ),

    );

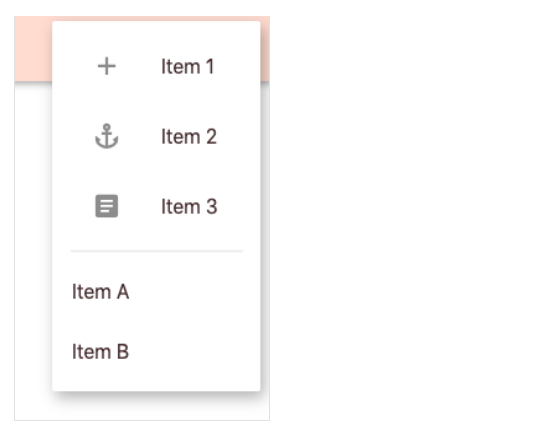
  }

}

================================================================================================

POPUP-Menu

A popup-menu is also like a dropdown-menu. But the little difference is that:



**Example:**

class JusticeAPP extends StatefulWidget {

  const JusticeAPP({Key? key}) : super(key: key);

  @override

  \_JusticeAPPState createState() => \_JusticeAPPState();

}

class \_JusticeAPPState extends State<JusticeAPP> {

  // popUpMenu Value

  String? popUpMenuValue;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: // PopupMenuButton represent the buttons that will be clicked to open the PopupMenuItem.

          // PopupMenuButton is a generic class PopupMenuButton<T>(). The generic <T> represent what dataType the PopupMenu value is going to be?

          // here am specifying String

          PopupMenuButton<String>(

              // the background for the popupMenu

              color: AppColors.blue,

              // Default popupMenu <it Optional>

              initialValue: popUpMenuValue,

              // icon to show on the PopupMenuButton to be clicked to open the PopupMenuItem.

              icon: Icon(Icons.more\_vert),

              // onSelected function means, do something when user select different PopupMenuItem

              // so this is where to call other methods when some PopupMenuItem is clicked

              onSelected: (String? newValue) {

                setState(() {

                  popUpMenuValue = newValue;

                });

                log("$popUpMenuValue");

              },

              // when clicks the <PopupMenuButton> and clicks somewhere outside the popupMenu to to cancel it

              // the <onCanceled> method will be called

              onCanceled: () {

                log("popup cancelled");

              },

              // list of <PopupMenuItem> to show. You can include a <PopupMenuDivider>

              itemBuilder: (context) {

                return <PopupMenuEntry<String>>[

                  // Each <PopupMenuItem> should have it <value> and child <widget>

                  PopupMenuItem(

                    child: ListTile(

                      leading: Icon(Icons.add),

                      title: Text('Item 1'),

                    ),

                    value: "Item 1",

                  ),

                  PopupMenuItem(

                    child: ListTile(

                      leading: Icon(Icons.anchor),

                      title: Text('Item 2'),

                    ),

                    value: "Item 2",

                  ),

                  PopupMenuItem(

                    child: ListTile(

                      leading: Icon(Icons.article),

                      title: Text('Item 3'),

                    ),

                    value: "Item 3",

                  ),

                  // Popup Menu Divider

                  PopupMenuDivider(),

                  PopupMenuItem(

                    child: Text('Item A'),

                    value: "Item A",

                  ),

                  PopupMenuItem(

                    child: Text('Item B'),

                    value: "Item B",

                  ),

                ];

              }),

    );

  }

}

================================================================================================

Slivers

Slivers is all about creating or defining a layout of items with scroll animation.

**Sliver Types:**

There 3 types of slivers.

1. Sliver-To-Sliver:

Sliver-parent Widget that Takes Sliver-child Widget Example: SliverPadding

  // Parent widget

 SliverPadding(

    padding: EdgeInsets.all(5),

      // child widget

      sliver: SliverAppBar(),

  ),

1. Sliver-To-Box:

Sliver-parent widget that takes a non-sliver-child-widget Example, SliverAppBar

 // Sliver Parent widget

 SliverAppBar(

  // Non-sliver-child widget

    title: Text("My New Title"),

  ),

1. Sliver-To-Many-Sliver:

Parent-sliver-widget that takes in List of Sliver widget as its child Example: SliverList

**How to Create A sliver Example:**

// list of colors to show

  List<String> colors =["Red","yellow","pink","white","black","blue", "indigo","violet","gray","green"];

  Widget build(BuildContext context) {

    return Scaffold(

      body:

              // Use CustomeSrollView because you want the Item to scroll

              // You can't use singleChildScrollview becuase that doesn't take in slivers Widget

              CustomScrollView(

                // The scroll direction

                scrollDirection: Axis.vertical,

                // This suppse to be children, but bacuase flutter want developers to know that, it takes in sliver widget as

                // children, that's why they made it <slivers> instead of <children>

                slivers: [

                  // Create a silver appBar

                  SliverAppBar(

                    backgroundColor: Colors.red,

                    centerTitle: true,

                    title: Text("My Title"),

                    // let the appbar stay fixed without being scrolled

                    // pinned: true,

                    // when I scroll upwards, hide the appBar, when I scroll downwards, show the appbar

                    // for certaine reasons in flutter, if you set floating: true, you need to remove the "pinned" property

                    floating: true,

                    // expand the height of the appBar

                    expandedHeight: 100,

                    // add some widget just below the appBar

                    // Placeholder() is also a widget that draws a box it can be Text() or any widget

                    flexibleSpace: Placeholder(),

                  ),

                  // Create sliver List

                  // This is just like a listView, just that you use this to work with Slivers

                  SliverList(

                    // Build the widget that will be returned

                      delegate: SliverChildBuilderDelegate(

                        // index represent the index of the item (its int)

                            (context, index){

                          return Container(

                            color: Colors.blue,

                            margin: EdgeInsets.all(2),

                            height: 90,

                            child: Text(colors[index]),

                          );

                        },

                        childCount: colors.length,

                      )

                  ),

                  // YOu can use the widget SliverPadding to add padding to all slivers like SliverList, SliverAppBar and many more

                  SliverPadding(

                    padding: EdgeInsets.all(5),

                    // remember the "child" is replaced with "sliver"

                    sliver: SliverAppBar(

                      // you can use FlexibleSpaceBar or any widget like Text()

                      flexibleSpace: FlexibleSpaceBar(

                        background: FlutterLogo(),

                         title: Text("My New Title"),

                      ),

                    ),

                  ),

                  // Use SliverGrid to show a grid of Items

                  SliverGrid(

                    // Define the spacing and the number of items to show horizontally here

                    gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

                      // Vertical space between the items

                      mainAxisSpacing: 10.0,

                      //horizontal space between the ites

                      crossAxisSpacing: 10.0,

                      // Number of items to show horizontally

                      crossAxisCount: 3,

                      childAspectRatio: 1

                    ),

                    // Build the widget that will be returned

                    delegate: SliverChildBuilderDelegate(

                      // index represent the index of the item (its int)

                        (context, index){

                          return Container(

                            color: Colors.red,

                            margin: EdgeInsets.all(2),

                            height: 30,

                            width: 30,

                            child: Text(colors[index]),

                          );

                        },

                      // The length of the item, so that the grid will know when to stop

                      childCount: 30,

                    )

                  ),

                ],

              ),

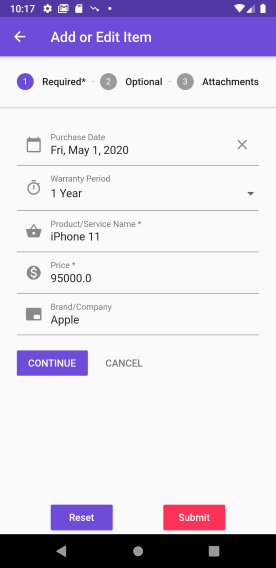
    );

  }

================================================================================================

Steppers

Steppers actually way of showing presentable forward & backwards widget.



**Example:**

// All the code here is inside a widget class

     //Create a var to represent the Stepper index

  int \_currentStep = 0;

// Create a method that takes in context parameter and return list of steppers to show

List<Step> \_steppers(BuildContext context){

List<Step> \_steps =[

      // First stepper

        Step(

          // this stepper heading

          title: Text("First stepper” )

              // content of this stepper

          content: Container(),

          // change this stepper color to indicate that it active

          // \_currentStep is an int var that should be declared in the class

          isActive: (\_currentStep==0),

          // when this steper is not shown, show it icon that it complete

          state: StepState.complete,

        ),

        // Second stepper

        Step(

          // this stepper heading

          title: Text("Second Stepper"),

              // content of this stepper

          content: Container()

        )

];

// return this above list of steppers

 return \_steps;

}

 @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Stepper(

        // call the above method that returns list of steppers

        steps: \_steppers(context),

              physics: ClampingScrollPhysics(),

              // stepper direction (vertical or horizontal)

            type: StepperType.horizontal,

            elevation: 0,

            // which stepper to show first

            currentStep: this.\_currentStep,

            // When the stepper is heading is tapped ("step" is int and is the index of the stepper)

            onStepTapped: (step){

              // navigat to that stepper

             setState(() {

               \_currentStep=step;

             });

            },

            onStepContinue: (){

              // move to the next stepper widget when the continue button is clicked

            setState(() {

                this.\_currentStep = this.\_currentStep + 1;

            });

            },

            // when priveios button is clicked

            onStepCancel: (){

            setState(() {

            // move back to the previous stepper

                this.\_currentStep = this.\_currentStep - 1;

            });

            },

            // the continue and previous buttons widget to show (build it here)

            controlsBuilder: (context,{onStepContinue,onStepCancel}){

              return

              Row(

                children: [

                  // Continue Button

                  InkWell(

                    ontap:(){

                      // call the onStepContinue property above

                     onStepContinue!();

                    }

                  )

                  SizedBox(

                    width: displayUserDeviceWidth(context) \* 0.04,

                  ),

                  // Cancel/previous Button

                  InkWell(

                    ontap:(){

                      // call the onStepCancel property above

                     onStepCancel!();

                    }

                  )

                ],

              );

            },

          ),

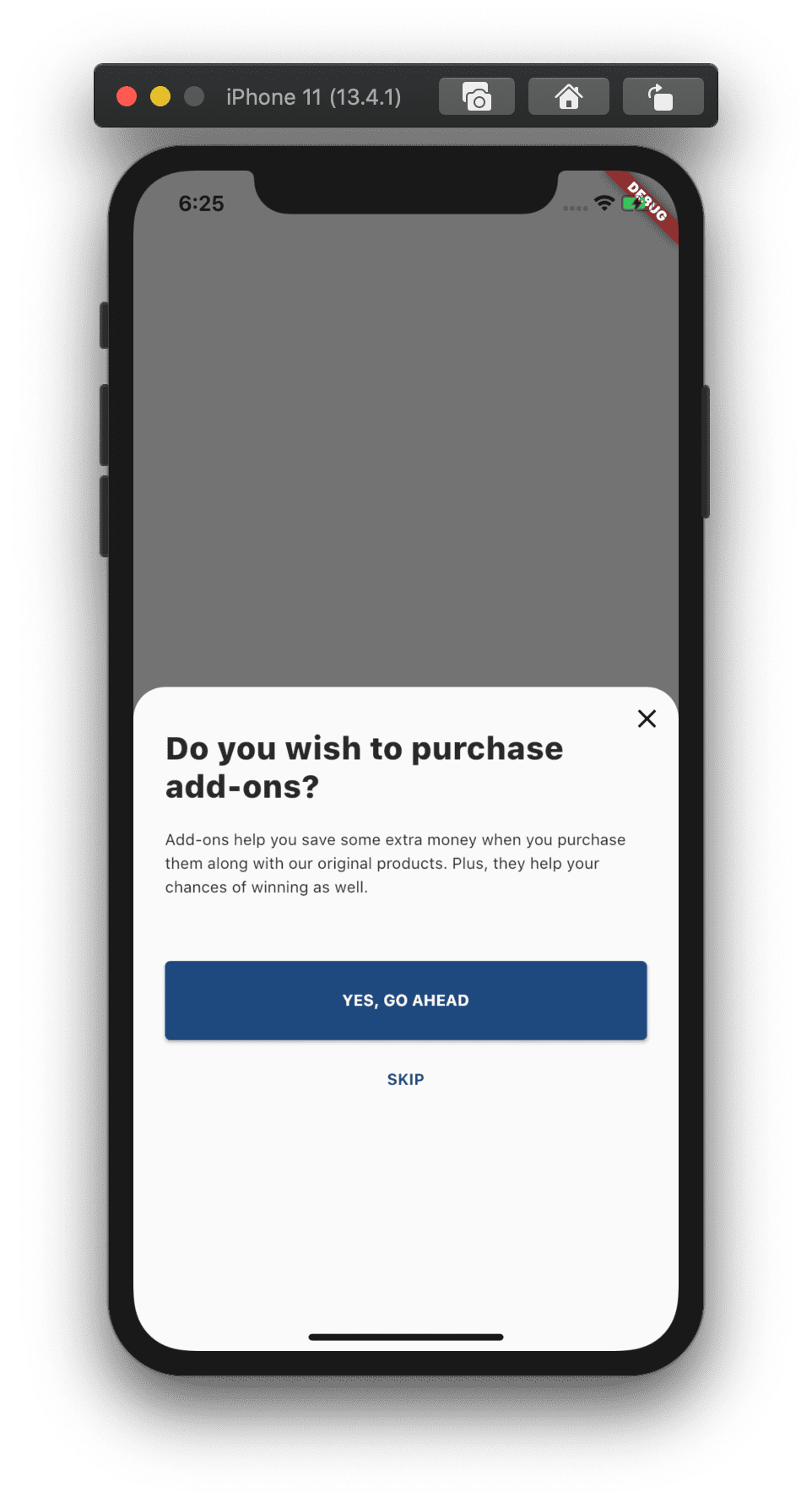
    );

  }

================================================================================================

Flutter button Sheet

Button sheet is for displaying a widget on the button of the user Device/screen when some action happens like a button click.



**Example:**

  // Build method

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        body: InkWell(

          child:Text("Show bottomSheet"),

           onTap: (){

             // call below method to show bottomsheet

              \_addCar\_Sheet(context);

            },

        )

  }

   // Create a method outside the build-method that shows a BottomSheet

  \_addCar\_Sheet(BuildContext context){

      // This method will be called automatically when above method is called

      showModalBottomSheet(

        // when the outside of the button sheet is clicked, dismist it or not

          isDismissible: false,

          // shape of the bottom sheet

          shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.vertical(top: Radius.circular(25.0))

              ),

          backgroundColor: Colors.white,

          context: context,

          // should you scroll the bottom sheet?

          isScrollControlled: true,

          // the widget builder

          builder: (context) {

            // always return a stateful builder

            return StatefulBuilder(

                builder: (BuildContext context, StateSetter setState) {

                  // The widget to return

                  return Text("Am Inside the bottom Sheet");

                }

            );

          }

      );

 }

================================================================================================

TIPS

* Navigator.of(context).push(MaterialPageRoute(builder: (context) => NewScreen())); 🡺 with back screen

Navigator.*of*(context).pushReplacement(MaterialPageRoute(builder: (context) => AboutPage())); 🡺 with no back screen

This is very good navigation method to navigate to a new screen, it comes with a nice default material page animation.

* Note: You can learn more about sqlite on <https://www.sqlitetutorial.net/>
* Note: Check the available dataTypes of Sqlite on https://www.sqlite.org/datatype3.html
* Note ==> space indentation in the pubspec.yaml file is very important, so when getting errors check your space indentation of your pubspec.yaml file