


NOTES - WCE GDSC FLUTTER BOOTCAMP

Recording Link:

Session 1 (Installation and Introduction to Flutter) -  Flutter Session-1

Reference Video -  2. Full Installation of Flutter on Windows - Your First Flutter App

Session 2 (Introduction to Basic Widgets and the making of first Application) - [Session 2.mkv](#)


Session 3 (Portfolio Application) - [Session3.mkv](#)

Session 4 (Dice Application) - [Session 4.mkv](#)

Widgets:

1 - **Material App**: MaterialApp is a widget that introduces a number of widgets Navigator, themes that are required to build a material design app.


Documentation - <https://api.flutter.dev/flutter/material/MaterialApp-class.html>

Video -  Flutter Widget Basics - MaterialApp

```
MaterialApp(  
  home: Scaffold(  
    appBar: AppBar(  
      title: const Text('Home'),  
    ),  
  ),  
  debugShowCheckedModeBanner: false,  
)
```

2 - **Scaffold**: Scaffold Widget is used under MaterialApp, it gives you many basic functionalities, like AppBar, BottomNavigationBar, Drawer, FloatingActionButton, etc. It can be thought of as a canvas that you can paint with the help of widgets.

Documentation - <https://api.flutter.dev/flutter/material/Scaffold-class.html>

Video -  Flutter Widget Basics - Scaffold

```
Scaffold(  
  appBar: AppBar(  
    //The basic mobile appbar, looks and works great  
    title: const Text('Sample Code'),  
  ),  
  body: Center(child: Text('You have pressed the button $_count times.')),  
  //Body uses the most space - when using a textfield(or other widgets with focusnodes,  
  wrap the body with a gesturedetector and set an onTap to FocusScope.of(context).unfocus();
```

```

to undofocus when tapping of the widget ;)
floatingActionButton: FloatingActionButton(
  //FAB is great and a defined in the scaffold
  onPressed: () => setState(() => _count++),
  tooltip: 'Increment Counter',
  child: const Icon(Icons.add),
),
);

```

3 - **Container**: A convenience widget that combines common painting, positioning, and sizing widgets.

Documentation - <https://api.flutter.dev/flutter/widgets/Container-class.html>

Video - [Container \(Flutter Widget of the Week\)](#)

```

Container(
  alignment: Alignment.center,
  child: Text('Container'),
  width: 200.0,
  height: 200.0,
  decoration: BoxDecoration(
    color: Colors.purple[200],
    borderRadius: BorderRadius.circular(60.0)
  )
),

```

4 - **Center**: A Widget that helps you orient things in the centre of your screen.

Documentation - [Center class - widgets library - Dart API](#)

Video - <https://youtu.be/c726pWTtxql>

```

Center(
  child: Text('Center'),
)

```

5 - **Image**: This allows you to add images to your application.

5- a) Network Image - <https://docs.flutter.dev/cookbook/images/network-image>

5- b) Asset Image: Video - [Flutter Tutorial for Beginners #8 - Images & Assets](#)

Documentation - [Adding assets and images | Flutter](#)


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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Image from assets"),
        ),
        body: Image.asset('assets/images/lake.jpg'), // <--- image
      ),
    );
  }
}
```

6 - **Text** - This allows you to add text to your application.

Documentation - [Text class - dart:html library](#)

Video -  Flutter Text Widget | How, what and why

```
Text(
  'Hello',
  textAlign: TextAlign.center,
  style: const TextStyle(fontWeight: FontWeight.bold),
)
```

7 - **RichText (New Widget)**: The **RichText** widget displays text that uses multiple different styles.

Documentation - [RichText class - widgets library - Dart API](#)

Video -  RichText (Flutter Widget of the Week)

```
RichText(
  text: TextSpan(
    text: 'Hello ',
    style: DefaultTextStyle.of(context).style,
    children: const <TextSpan>[
      TextSpan(text: 'bold', style: TextStyle(fontWeight: FontWeight.bold)),
      TextSpan(text: ' world!'),
    ],
  ),
)
```

```
1,  
)
```

8 - **SafeArea:**

Documentation - [SafeArea class - widgets library - Dart API](#)

Video - [SafeArea \(Flutter Widget of the Week\)](#)

```
SafeArea(  
  bottom: true,  
  left: true,  
  top: true,  
  right: true,  
  maintainBottomViewPadding: true,  
  minimum: EdgeInsets.zero,  
  child: Scaffold(  
    appBar: AppBar(title: Text('SizedBox')),  
  ),  
)
```

Concept of Single Child and Multi-Child Widgets -

Single-Child Layout Widgets are the ones that will accept only one widget as their child
Such as: Container(), Center(), Expanded(), Align(), SizedBox() etc.

Multi-Child Layout Widgets are the ones that will accept more than one widget as their child

Such as: Column(), Row(), Stack(), GridView(), ListView(), Table() etc.

In very simple terms Whenever you see or use a widget with a child property, That widget is called Single-Child Layout

```
Container(child: Text());
```

```
Center(child: Text());
```

```
Expanded(child: Text());
```

```
Align(child: Text());
```

```
SizedBox(child: Text());
```

And Whenever you see or use a widget with a children property, That widget is called Multi-Child Layout*

```
Column(children: [Text(), Text(), Text(), Text()]); Row(children: [Text(), Text(), Text(), Text()]);
```

```
Stack(children: [Text(), Text(), Text(), Text()]);
```

Single-Child Layout Widget is used as Wrapper for other widgets for changing their positions, width, height, adding space, margin, aligning content, etc.

Multi-Child Layout Widget is used for creating a list of widgets in the horizontal or vertical direction, also for stacking widgets one over another, arranging data as well as widgets in a table structure, etc.

9 - Column Widget:

Documentation - [Column class - widgets library - Dart API](#)

Video - https://youtu.be/siFU8c_Heu0

```
Column(children: [Text(),  
Text(),  
Text(),  
Text()  
]);
```

10 - Row Widget:

Documentation - [Row class - widgets library - Dart API](#)

Video - [Flutter Tutorial for Beginners #11 - Rows](#)

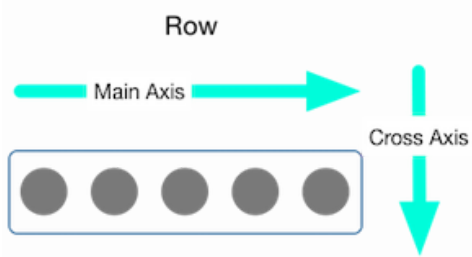
```
Row(children: [Text(),  
Text(),  
Text(),  
Text()  
]);
```

Some Important points about Column and Row Properties

For Row:

mainAxisAlignment = Horizontal Axis

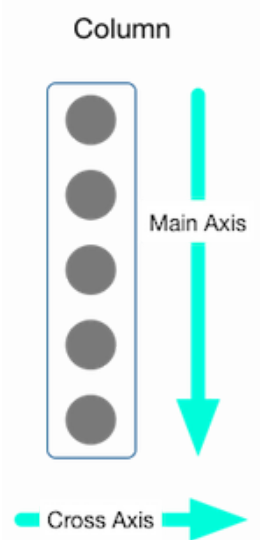
crossAxisAlignment = Vertical Axis



For Column:

mainAxisAlignment = Vertical Axis

crossAxisAlignment = Horizontal Axis



11 - **Circle Avatar:** A circle that represents a user. Typically used with a user's profile image, or, in the absence of such an image, the user's initials. A given user's initials should always be paired with the same background colour, for consistency.

Documentation - [CircleAvatar class - material library - Dart API](#)

Video - [Flutter Widgets | CircleAvatar](#)

```
CircleAvatar(  
  backgroundImage: NetworkImage(userAvatarUrl),  
)
```

12 - **Padding:**

Documentation - [Padding class - widgets library - Dart API](#)

Video - [Padding \(Flutter Widget of the Week\)](#)

```
Padding(  
  padding: EdgeInsets.only(top: 10.0, bottom: 10.0),  
  padding: EdgeInsets.symmetric(vertical: 10.0,),  
  child: Text('Padding'),  
),
```

13 - **Divider** - Adds a line in your application. Works very similar to <hr> tag in HTML

Documentation - [Divider class - material library - Dart API](#)

Video - [Divider \(Flutter Widget of the Week\)](#)

```
const Divider(  
  thickness: 5, // thickness of the line  
  indent: 20, // empty space to the leading edge of divider.  
  endIndent: 20, // empty space to the trailing edge of the divider.  
  color: Colors.black, // The color to use when painting the line.  
  height: 20, // The divider's height extent.  
),
```

14 - **Card** - A material design card: a panel with slightly rounded corners and an elevation shadow. A card is a sheet of Material used to represent some related information, for example, an album, a geographical location, a meal, contact details, etc.

Documentation - [Card class - material library - Dart API](#)

Video - [Flutter Tutorial for Beginners #19 - Cards](#)

```
Card(  
  shape: RoundedRectangleBorder(  
    borderRadius: BorderRadius.circular(15.0),  
  ),  
  color: Colors.lightBlueAccent.withOpacity(0.5),  
),
```

15 - **ListTile** - A single fixed-height row that typically contains some text as well as a leading or trailing icon.

Documentation- [ListTile class - material library - Dart API](#)

Video - [ListTile \(Flutter Widget of the Week\)](#)

```
ListTile(  
  title: Text('ListTile'),  
  isThreeLine: true,  
  subtitle: Text('Secondary text\nTertiary text'),  
  leading: Icon(Icons.label),  
  trailing: Text('Metadata'),  
),
```

16 - **Icons** -

Documentation - [Icon class - widgets library - Dart API](#)

Video - <https://youtu.be/ABmqtl7ec7E>

```
Icon(  
  Icons.home,  
  color: Colors.blue,  
  size: 40.0,  
  textDirection: TextDirection.ltr,  
  semanticLabel: 'Icon', // Announced in accessibility modes (e.g TalkBack/VoiceOver).  
  This label does not show in the UI.  
)
```

17 - **Buttons** -

Documentation - [Flutter Buttons - Javatpoint](#)

Video - <https://youtu.be/ABmqtl7ec7E>

```
FlatButton(  
  color: Colors.red,  
  splashColor: Colors.black12,  
  onPressed: ()  
{  
  },  
  child: Text(  
    "Nouvelle Texte"  
  ),  
)
```


18 - Expanded -

Documentation - <https://api.flutter.dev/flutter/widgets/Expanded-class.html>

Video - [📺 Expanded \(Flutter Widget of the Week\)](#)

```
Expanded(  
  child: Container(  
    height: 100.0,  
    color: Colors.green,  
    width: 100.0,  
  ),  
)
```

Note about State Widget - Stateful and Stateless Widgets

Stateless Widget:

1. A widget that has an immutable state.
2. Stateless Widgets are static widgets.
3. They do not depend on any data change or any behaviour change.
4. Stateless Widgets do not have a state, they will be rendered once and will not update themselves, but will only be updated when external data changes.
5. For Example Text, Icon, RaisedButton are Stateless Widgets.

Stateful Widget:

1. A widget that has a mutable state.
2. Stateful Widgets are dynamic widgets.
3. They can be updated during runtime based on user action or data change.
4. Stateful Widgets have an internal state and can re-render if the input data changes or if the Widget's state changes.
5. For Example Checkbox, Radio Button, Slider are Stateful Widgets

State Widgets - <https://www.youtube.com/watch?v=Nui8IXel3Yk>

Read a little about **onPressed Property** - <https://googleflutter.com/flutter-button-onpressed/>

Read a little about **SetState Method** - <https://api.flutter.dev/flutter/widgets/State/setState.html>

Video - https://www.youtube.com/watch?v=TVXM6YZ_wcg

Variables in Dart

A variable is “a named space in the memory” that stores values. In other words, it acts a container for values in a program. Variable names are called identifiers. Following are the naming rules for an identifier –

- Identifiers cannot be keywords.
- Identifiers can contain alphabets and numbers.
- Identifiers cannot contain spaces and special characters, except the underscore (_) and the dollar (\$) sign.
- Variable names cannot begin with a number.

Type Syntax

A variable must be declared before it is used. Dart uses the var keyword to achieve the same. The syntax for declaring a variable is as given below –

```
var name = 'Smith';
```

All variables in dart store a reference to the value rather than containing the value. The variable called name contains a reference to a String object with a value of “Smith”.



Dart supports type-checking by prefixing the variable name with the data type. Type-checking ensures that a variable holds only data specific to a data type. The syntax for the same is given below –

```
String name = 'Smith';  
int num = 10;
```

Consider the following example –

```
void main() {  
  String name = 1;  
}
```

The above snippet will result in a warning since the value assigned to the variable doesn't match the variable's data type.

Output

```
Warning: A value of type 'String' cannot be assigned to a variable of type 'int'
```

Functions in Dart

Functions are the building blocks of readable, maintainable, and reusable code. A function is a set of statements to perform a specific task. Functions organize the program into logical blocks of code. Once defined, functions may be called to access code. This makes the code reusable. Moreover, functions make it easy to read and maintain the program's code.

A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.

Sr.No	Functions & Description
1	<p>Defining a Function</p> <p>A function definition specifies what and how a specific task would be done.</p>
2	<p>Calling a Function</p> <p>A function must be called so as to execute it.</p>
3	<p>Returning Functions</p> <p>Functions may also return value along with control, back to the caller.</p>
4	<p>Parameterized Function</p> <p>Parameters are a mechanism to pass values to functions.</p>

Optional Parameters

Optional parameters can be used when arguments need not be compulsorily passed for a function's execution. A parameter can be marked optional by appending a question mark to its name. The optional parameter should be set as the last argument in a function.

We have three types of optional parameters in Dart –

Sr.No	Parameter & Description
1	<p>Optional Positional Parameter</p> <p>To specify optional positional parameters, use square [] brackets.</p>
2	<p>Optional named parameter</p> <p>Unlike positional parameters, the parameter's name must be specified while the value is being passed. Curly brace {} can be used to specify optional named parameters.</p>
3	<p>Optional Parameters with Default Values</p> <p>Function parameters can also be assigned values by default. However, such parameters can also be explicitly passed values.</p>

Recursive Dart Functions

Recursion is a technique for iterating over an operation by having a function call to itself repeatedly until it arrives at a result. Recursion is best applied when you need to call the same function repeatedly with different parameters from within a loop.

Example

```
void main() {
  print(factorial(6));
}
factorial(number) {
  if (number <= 0) {
    // termination case
    return 1;
  } else {
    return (number * factorial(number - 1));
    // function invokes itself
  }
}
```

It should produce the following output –

720

Lambda Functions

Lambda functions are a concise mechanism to represent functions. These functions are also called as Arrow functions.

Syntax

```
[return_type]function_name(parameters)=>expression;
```

Example

```
void main() {  
  printMsg();  
  print(test());  
}  
printMsg()=>  
print("hello");  
  
int test()=>123;  
// returning function
```


It should produce the following output –

hello 123

Some more Widgets :

19 - Navigator:

Documentation - [Navigator class - widgets library - Dart API](#)

Video :  Flutter Tutorial - How To Navigate to New Screen and Back [2021] Navigator Push...

```
ListTile(  
  title:Text("Settings",style:TextStyle(fontSize:20.0),),  
  leading:Icon(Icons.home),  
  onTap:(){  
    Navigator.push(context,MaterialPageRoute(builder:  
      (context)=>A()  
    ));  
  },  
),
```

20 - Webview:

Documentation - [webview_flutter | Flutter Package](#)

Video: [Flutter Tutorial - WebView App | The Right Way \[2021\] 1/3 Load URL, HTML, Java...](#)

```
class A extends StatefulWidget {
  const A({Key? key}) : super(key: key);

  @override
  _AState createState() => _AState();
}

class _AState extends State<A> {
  @override
  Widget build(BuildContext context) {
    return WebView(
      initialUrl: 'https://www.youtube.com/watch?v=N7BvXrOYQTY',

    );
  }
}
```

21 - URL Launcher : A Flutter plugin for launching a URL

Documentation: https://pub.dev/packages/url_launcher

Video: [Flutter Tutorial - How To Use URL Launcher \[2021\] Open URL In Web Browser / In ...](#)

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

const String _url = 'https://flutter.dev';

void main() => runApp(
  const MaterialApp(
    home: Material(
      child: Center(
        child: RaisedButton(
          onPressed: _launchURL,
          child: Text('Show Flutter homepage'),
        ),
      ),
    ),
  ),
);
```

```
void _launchURL() async {  
  if (!await launch(_url)) throw 'Could not launch $_url';  
}
```

GDSC WCE