DART & FLUTTER BASICS – PART 1

11th November 2023

at Institute of Computer Engineering Technology - ICET

AGENDA

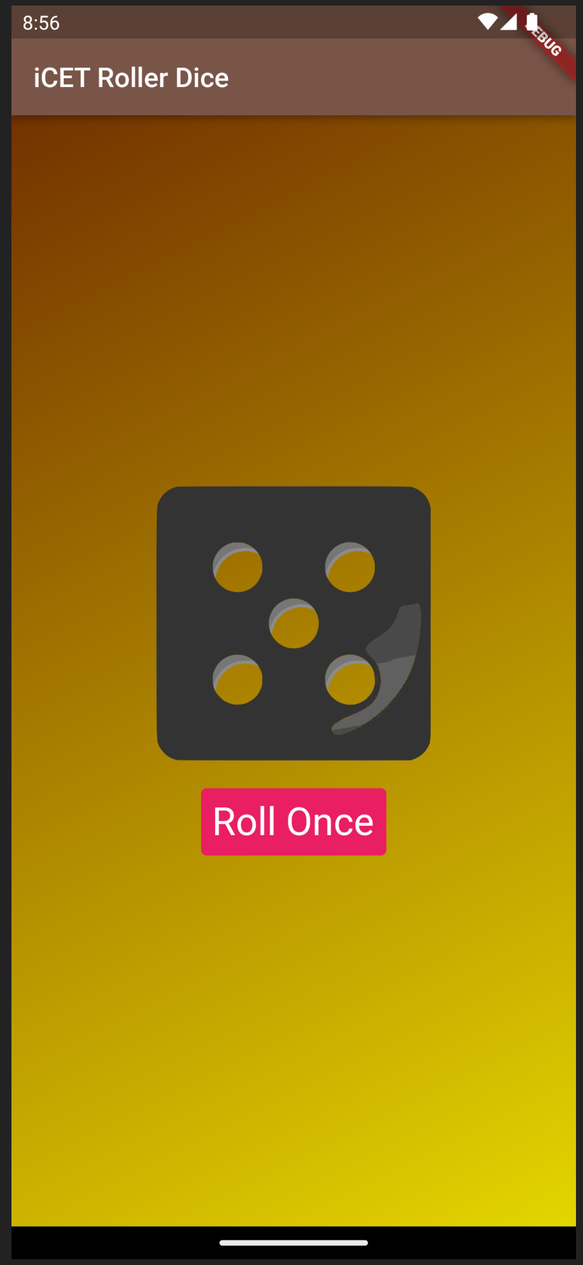
- Recap of Session 1
- Output of Session 2
- Session 2 Outline
  - Flutter & Dart syntax
    - Keywords, Identifiers, Variables, Functions, Arguments, Value Types, Classes, Objects, Generics
  - Understanding & Writing Flutter and Dart code
  - Flutter Widgets
  - Combine widgets & build custom widgets

RECAP OF SESSION 1

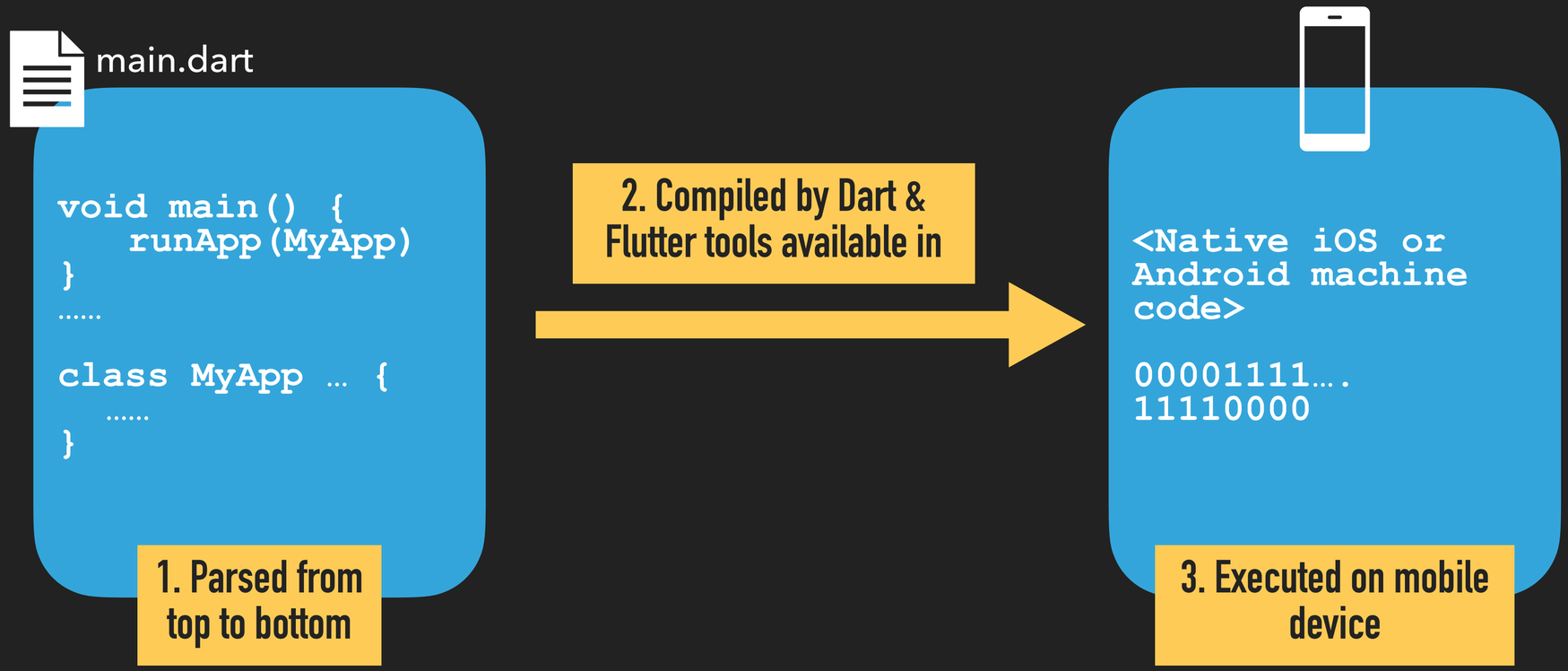
- Introduction about each and everybody
- Course outline
- Local Setup and troubleshoot setup issues
- Create our first Flutter App & Run on Android device or Emulator
- Folder Structure
- Skim through main.dart file
- Flutter flavours

OUTPUT OF SESSION 2

- We are going to build our first **Interactive Flutter App** from the scratch while we are deep diving into Dart & Flutter fundamentals
- What is we are going to build ?
  - Roller Dice Application



How Dart & Flutter is Getting Compiled



Keywords & Identifiers

Keywords

Build into the programming language - ie. Dart -> class, import, void, return, extends

Contextual keywords, build-in identifiers and limited reserve keywords can be use as identifies at some points

Dart Keywords: https://dart.dev/language/keywords

Identifiers

Define by Developers - ie. StatelessWidget, MyApp, FirstWidget, firstName, standard_text_field

`UpperCamelCase` -> Classes, enums types, typedefs and type parameters

`lowerCamelCase` -> class members, variables, parameters, constant names

`lower_with_underscores` -> packages, directories, file names, import prefixes

Identifier Styling: https://dart.dev/effective-dart/style

How Flutter App will Execute

main.dart

```
void main() {
    runApp(MyApp)
}
.....

class MyApp ... {
   .....
}
```

1. main() function get executed automatically

By Dart, when executing the compiled app on the target device. main() function is special function provide by Dart & can't be use any other places in a Flutter project

2. runApp() should call inside main() function

runApp() "tells" Flutter what to display on the screen (i.e., which UI elements to display)

3. To be displayed 'widget tree' should be inside runApp()

A "widget tree" is a combination of (nested) Flutter widgets that build the overall user interface

Dart Functions

Function Definition

```
void calculateTwoNumbers(int num1, int num2) {
  .........
}
```

calculateTwoNumbers is the Function name & it's the identifier

Function Usage/ Execution

```
void main() {
  calculateTwoNumbers(4, 5)
}
```

Function Parameters(Arguments)

Functions can take no parameters or multiple parameters as inputs. Parameters can also called as Arguments

Without Parameters

```
void main() { … }
```

Single Parameter

```
void log(String: text){ … }
```

Multiple Parameter

```
void sum(int: a, int: b){ … }
```

Dart Parameters: https://dart.dev/language/functions#parameters

Named & Positional Arguments

Named arguments

```
void sum({a, b}){ … }
```

```
sum(b: 10, a: 20);
```

Name/s of argument must be specified when passing into function call

Positional Arguments

```
void sum(a, b){ … }
```

```
sum(10, 20);
```

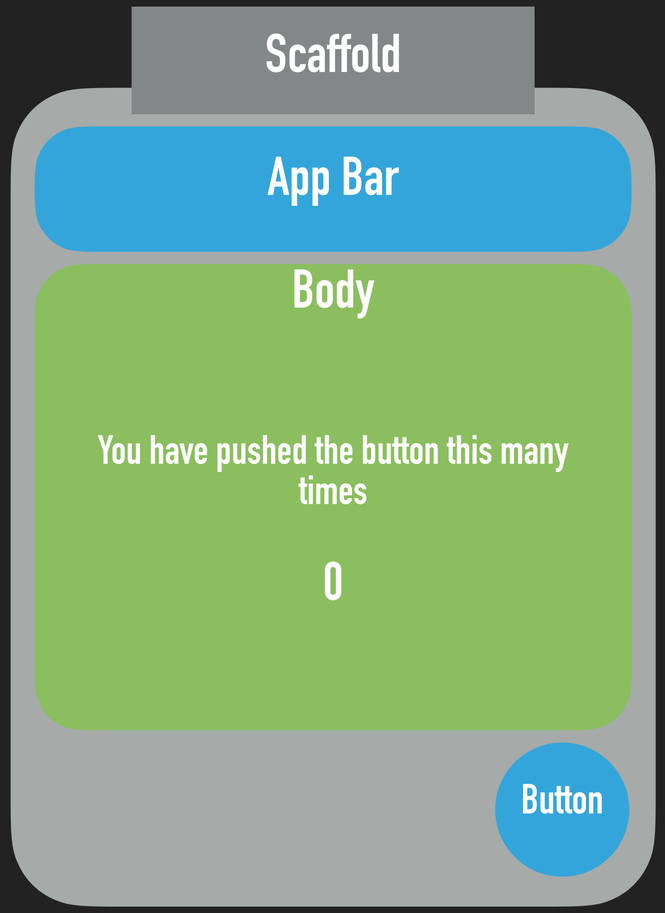
Argument value/s is/are mapped by passing position

Named Parameters: https://dart.dev/language/functions#named-parameters

Positional Parameters: https://dart.dev/language/functions#optional-positional-parameters

Flutter is All About Widgets

- When you are building your Flutter UI you build your UI with dart code, inbuilt flutter widgets and custom widgets
- And those UIs are simply combination of widgets
- Widgets are nested into each other




```dart
Scaffold(
  appBar: AppBar(
    backgroundColor: Theme.of(context).colorScheme.inversePrimary,
    title: Text(widget.title),
  ),
  body: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        const Text('You have pushed the button this many times:',),
        Text(
          '$_counter',
          style: Theme.of(context).textTheme.headlineMedium,
        ),
      ],
    ),
  ),
  floatingActionButton: FloatingActionButton(
    onPressed: _incrementCounter,
    tooltip: 'Increment',
    child: const Icon(Icons.add),
  ),
);
```

Flutter is All About Widgets . . .

Built-in Widgets	Buttons, Form Inputs, Layouts . . .
Custom Widgets	Developers can build their own widgets based on application requirement on top of Built-in widgets

Custom Gradient Submit Button

Flutter Widget Catalog: https://docs.flutter.dev/ui/widgets

Dart Const

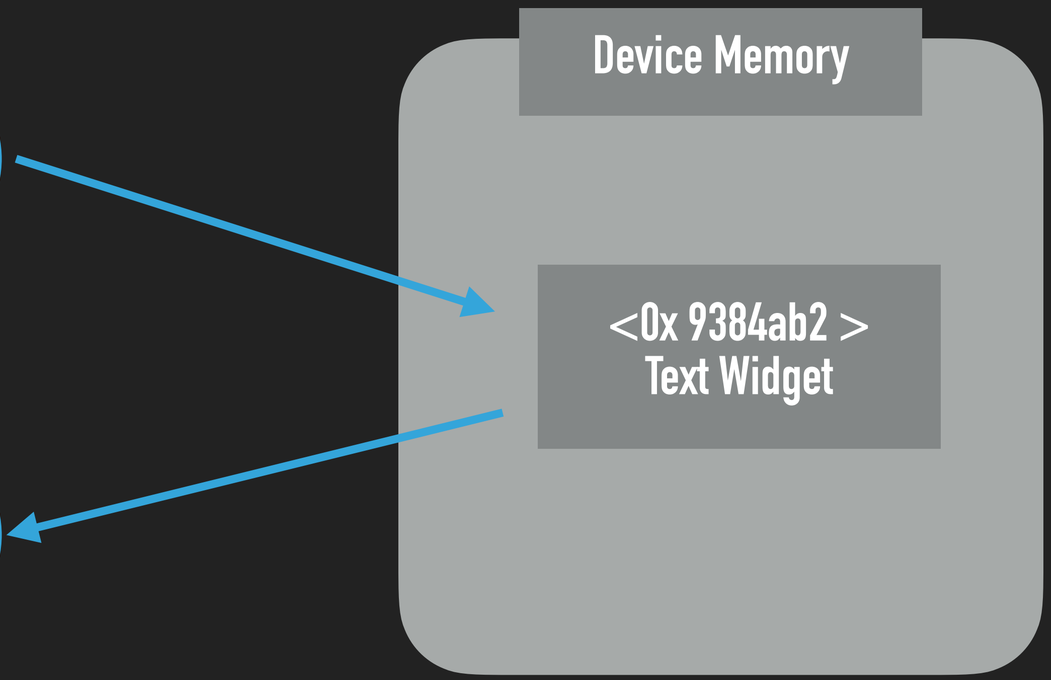
`const` always helps Dart to optimise runtime performance

Define & used 1st time in the program/app

```
const Text('My Name is Shelan")
```

Define & used 2nd time in the program/app

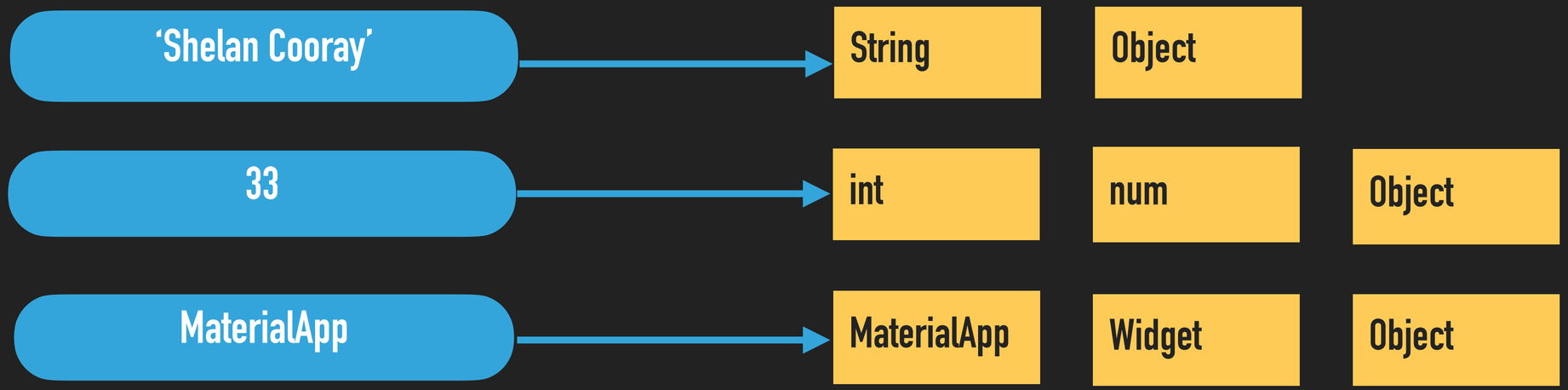
```
const Text('My Name is Shelan")
```



Dart Types

Dart is a Type Safe Language

All values are certain of type



Widget = Object = Data Structure in Memory

Dart Types – Some Core Types

int	**Integer Number**	Numbers without Decimal Places	5, 10, 25, 0, -10, -23
double	**Fractional Numbers**	Numbers with Decimal Places	3.25, 4.5, -2.51
num	**Integer or Fractional Numbers**	Numbers with or without Decimal Places	5, 100, 20.45, -2.4
String	**Text values**	Text wrapped with single or double quotes	‘Shelan Cooray’, “Hello World”
bool	**Boolean values**	Only true or false	true, false
Object	**Any kind of above**	Base type of all types	5, true, “Hi”, 2.75

Dart Generic Types

Generic types are types which are flexible types work with other types

List of names

```
['Shelan', 'Kasun', 'Dineth']
```

List<String>

List of Colors

```
[Color.fromARGB(255, 8, 0, 22),
Color.fromARGB(255, 75, 39, 136)]
```

List<Color>

Dart Generics: https://dart.dev/language/generics

Dart Classes

Dart is an Object Oriented Language & in Dart every value is an Object

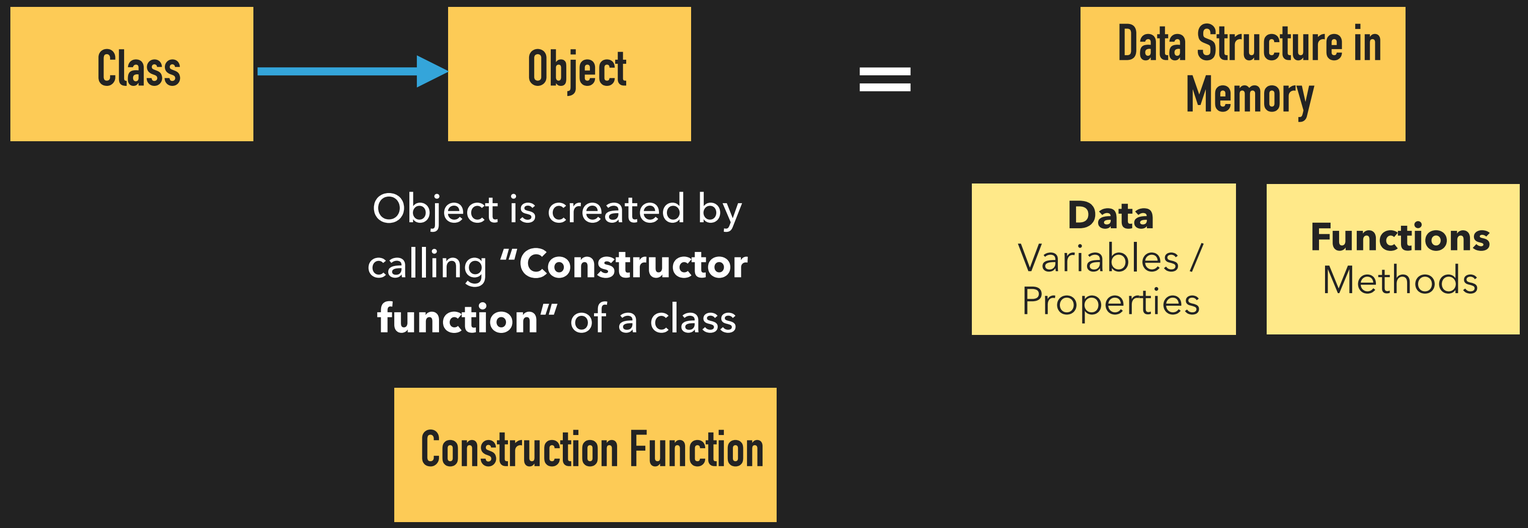
Primitive Values

Text

`"Hello World", "Shelan"`

Numbers

`10, 30.4, 0`



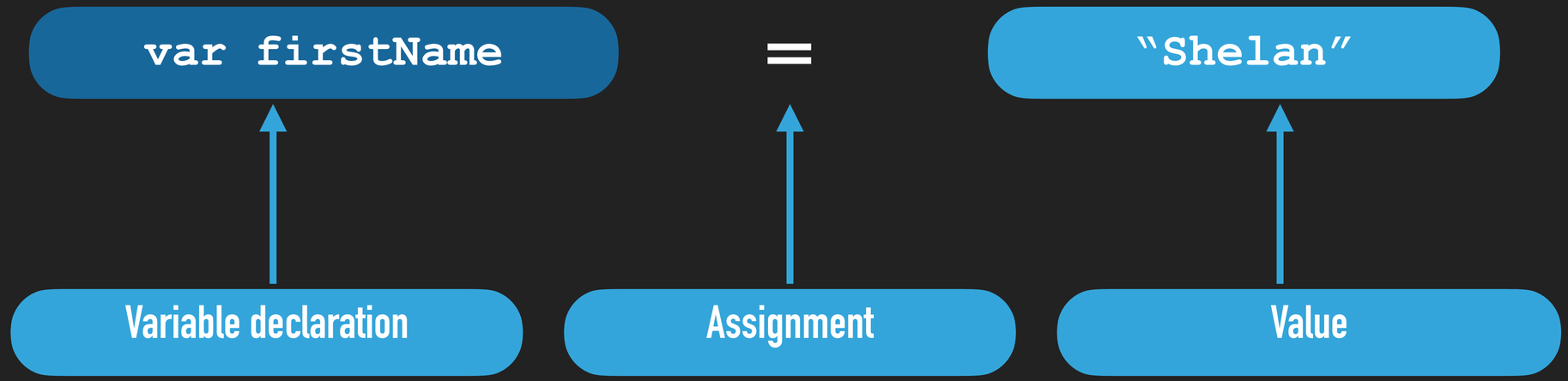
In Flutter -> Scaffold, Widget, Colors, Gradient and etc. are created based on a Class (blue print)

Dart Class: https://dart.dev/language/classes

Dart Class: https://dart.dev/language/constructors

Dart Variables

Variables are Data Containers which hold values of each & every assignment



Dart Variables: https://dart.dev/language/variables

Dart Variables

`var`

Creates a new variable that will be re-assigned at some point

Use the type (e.g., String) instead of var if the variable has no initial value

Otherwise, the type can be inferred by Dart

`final`

Create a new variable that will (and can) never be re-assigned

Prefer over var to avoid unintended re-assignments (e.g., by other developers)

`const`

Create a new compile-time constant

Will (and can) never be re-assigned & value is “hardcoded” (fixed) at compile-time

Can’t be used if some code must be executed in order to derive the value

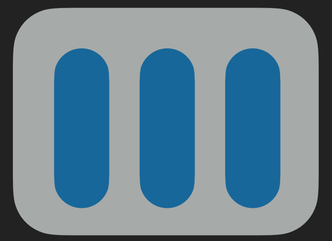
Flutter Column & Row

`Column()`

Main Axis: Vertical

Cross Axis: Horizontal

By default, occupies the **entire available height** but **only the width required** by its content (children)



`Row()`

Main Axis: Horizontal

Cross Axis: Vertical

By default, occupies the **entire available width** but **only the height required** by its content (children)

Flutter StatelessWidget & StatefulWidget &

`StatelessWidget`	`StatefulWidget`
Don’t manage any internal data or state	Manage internal state
Only update if parent widget/s get updated (re-rendered)	When state change widget is updated and UI will be change
Always use if you don’t have any state to manage inside your widget	Only use when your widget have have dynamic state to manage

Take Home Work

- Extend the Roller Dice Application to play by 2 users
- Add a ‘Play New Game’ Button
- Then Player 1 & Player 2 ‘Role Once’ buttons should appear
- Allow Players to play until 5 rounds
- After 5 rounds you have to display who won the game based on 5 round results with a Summary of each round
- Additional features are Welcome

Roller Dice Application Github Repo: https://github.com/shlncooray/icet_flutter_session_2

THANK YOU VERY MUCH FOR YOU PARTICIPATION

Shelan Cooray