Flutter Course

Section No.1

instructor
YOUSSEF HOSSAM

Table of contents

01

Introduction

Brief about the instructor ,Member introducing

02

Road map

Course tobics - Expectations

03 Requirements

What do we need to get the best results?

- ❑ Laptop - Android studio requirements
- ❑ Microphone
- ❑ Share Screen

04

Course system

Tasks – Quizzes - Projects - Attendance

Introduce yourself

What is dart ?

Dart is an open-source, general-purpose programming language that was developed by Google. It was first unveiled in 2011 and has since gained popularity, primarily as the programming language used for building applications using the Flutter framework.

- Dart is **simple**
- **High Performance**
- **Cross-Platform Development**

Overall, Dart is a versatile language that can be used for a variety of purposes, including mobile app development, web development, and more.

– What are we need to know in dart ?

1- Variables and Data Types: Dart has a strong static type system. You should be familiar with basic data types like **int**, **double, String, bool**, and understand how to declare variables with explicit types or with type inference.

2- **Control Flow**: Understand common control flow structures like **if, else, switch, for, and while loops.**

3- **Functions: Learn how to define functions using the void keyword for functions that don’t return a value or specify a return type using Type functionName() { ... }.**

4-**Lists and Maps:** Dart provides data structures like lists and maps. You should be comfortable working with lists (arrays) and maps (key-value pairs).

– What are we need to know in dart ?

5- **Classes and Objects:** Dart is an object-oriented language. Learn how to create classes and instantiate objects. Understand constructors, methods, and properties of classes.

6- **Null Safety**: Dart introduced null safety, which means you should understand how to work with nullable and non-nullable variables using ? and ! annotations.

7- **Asynchronous Programming:** Dart has built-in support for asynchronous programming using async, await, and Future. You'll use these extensively for handling network requests, file I/O, and other asynchronous tasks.

Data Types

Numbers

int

```
int age = 30;
Int x=0;
Integer y=55.5;
```

double

```
double price = 9.99;
Double z =10;
```

Strings

Strings are used to represent text and are enclosed in single (') or double (") quotes

Var

```
var s1 = ’Single quotes work well for string literals.’;
var s2 = ”Double quotes work just as well.”;
var s3 = ’It\’s easy to escape the string delimiter.’;
var s4 = ”It’s even easier to use the other delimiter.”;
```

String

```
String greeting = ’Hello, Dart!’;.
```

String methods - Soon

Booleans

Booleans represent two values: true and false and are used for logical operations.

```
bool isRaining = true;
bool isSunny = false;
```

Lists

Lists are ordered collections of objects. They can contain elements of any data type.

```
List<int> numbers = [1, 2, 3, 4, 5];
List<String> fruits = ['apple', 'banana', 'cherry'];
```

Sets

A set in Dart is an unordered collection of unique items. Dart support for sets is provided by set literals and the Set type.

```
var halogens = {'fluorine', 'chlorine', 'bromine', 'iodine', 'astatine'};
```

Comments

1. Single-Line Comments: **Single-line comments are used to add comments on a single line of code. They start with two slashes (“//”)**

2. Multi-Line Comments: They start with /* and end with */. Everything between the /* and */ is treated as a comment,

```
// This is a single-line comment in Dart
int x = 42;
 // You can add comments after code on the same line
```

```
/*
  This is a multi-line comment in Dart.
  You can use it to provide detailed
explanations or to comment out
  large blocks of code.

  int y = 100;
  int z = 200;
*/
```

Operators

1. Arithmetic Operators: **+:** Addition -: Subtraction *: Multiplication /: Division %: Modulus (remainder after division) ~/: Integer Division (divides and returns the integer part)

2. Assignment Operators:

=: Assigns the value on the right to the variable on the left.
+=: Adds the value on the right to the variable.
-=: Subtracts the value on the right from the variable.
*=: Multiplies the variable by the value on the right.
/=: Divides the variable by the value on the right.
%=: Assigns the remainder of the variable divided by the value on the right.

```
int x = 5;
x += 3;  // x is now 8
```

```
int a = 10;
int b = 3;

int sum = a + b;       // 13
int difference = a - b;  // 7
int product = a * b;     // 30
double quotient = a / b; // 3.3333
int remainder = a % b;   // 1
int integerDivision = a ~/ b; // 3
```

Operators

3. Comparison Operators: ==: Equal to !=: Not equal to <: Less than >: Greater than <=: Less than or equal to >=: Greater than or equal to

4. Logical Operators:

&& (AND): Returns true if both conditions are true.
|| (OR): Returns true if at least one condition is true.
! (NOT): Returns the opposite of the condition.

```
int a = 5;
int b = 7;

bool isEqual = (a == b);  // false
bool isNotEqual = (a != b);  // true
bool isLessThan = (a < b);  // true
bool isGreaterThan = (a > b);  // false
```

```
bool isTrue = true;
bool isFalse = false;

bool andResult = isTrue && isFalse;  // false
bool orResult = isTrue || isFalse;   // true
bool notResult = !isTrue;            // false
```

Operators

5. Increment and Decrement Operators:

++: Increment (add 1 to a variable)

--: Decrement (subtract 1 from a variable)

6. Conditional Operator (Ternary Operator):

The conditional operator allows you to return different values based on a condition.

condition ? valueIfTrue : valueIfFalse

```
int count = 5;
count++;  // Increment count by 1, count is now 6
count--;  // Decrement count by 1, count is now 5 again
```

```
int age = 20;
String message = age >= 18 ? "You are an
adult" : "You are not an adult";
```

User Input

1- import 'dart:io';

2- DataType? dataTypeName = stdin.readLineSync();

```
import 'dart:io';

void main(List<String> arguments) {

print('Enter your name: ');
 String name = stdin.readLineSync();
 print(name);
}
```

Practise Time

Exercise 1: Calculate Area and Perimeter
Write a Dart program that calculates the area and perimeter of a rectangle. Prompt the user for the length and width of the rectangle as input. Use appropriate data types and formulas for the calculations.

Exercise 2**: User Profile**
Build a Dart program that creates a user profile. Prompt the user for their name, age, and favorite color. Store this information in appropriate variables with suitable data types. Then, display a message that includes the user's name, age, and favorite color.

Exercise 3**: Temperature Conversion**
Create a Dart program that converts temperatures between Celsius and Fahrenheit. Prompt the user for a temperature value and a choice of conversion (Celsius to Fahrenheit or Fahrenheit to Celsius). Perform the conversion and display the result.

Maps

in general, a map is an object that associates keys and values. Both keys and values can be any type of object. Each *key* occurs only once, but you can use the same *value* multiple times.

```
var Studens = {
  // Key:    Value
  'first': ‘Ahmed',
  'second': ‘Ali',
  'fifth': ‘Akram'
};
```

String Methods

Dart provides a variety of string manipulation methods that allow you to work with strings effectively. Here are some commonly used string methods in Dart:

Length

```
String myString = 'Hello, Dart!';
int length = myString.length; // 12
```

isEmpty and isNotEmpty

```
String emptyString = '';
bool isEmpty = emptyString.isEmpty; // true
```

toUpperCase and toLowerCase

```
String text = 'Dart Programming'; String uppercaseText = text.toUpperCase(); //
"DART PROGRAMMING"
String lowercaseText = text.toLowerCase(); // "dart programming"
```

String Methods

Dart provides a variety of string manipulation methods that allow you to work with strings effectively. Here are some commonly used string methods in Dart:

Substring

```
String text = 'Dart Programming';
String substring = text.substring(5, 12); // "Program“
```

indexOf and lastIndexOf

```
String text = 'Dart Programming is fun, and Dart is cool.';
 int firstIndex = text.indexOf('Dart'); // 0
 int lastIndex = text.lastIndexOf('Dart'); // 27
```

toUpperCase and toLowerCase

```
String text = 'Dart Programming'; String uppercaseText = text.toUpperCase(); //
"DART PROGRAMMING"
String lowercaseText = text.toLowerCase(); // "dart programming"
```

String Methods

Dart provides a variety of string manipulation methods that allow you to work with strings effectively. Here are some commonly used string methods in Dart:

replaceFirst and replaceAll

```
String text = 'I love Dart, Dart is great!';
String replaced = text.replaceFirst('Dart', 'Flutter'); // "I love Flutter, Dart is great!"
String allReplaced = text.replaceAll('Dart', 'Flutter'); // "I love Flutter, Flutter is great!"
```

Contains

```
String text = 'Dart is a great language.';
bool containsDart = text.contains('Dart'); // true
```

Practise Time

Exercise 1**: Email Validator**
Create a Dart program that validates whether a given string is a valid email address. Use string methods to check for the presence of the "@" symbol, proper domain format, and other email address criteria.

Exercise 2**: Sentence Reversal**
Write a Dart program that takes a sentence as input and reverses the order of words in it. For example, if the input is "Hello World," the program should output "World Hello."
These exercises will help you practice various string methods in Dart, such as splitting, reversing, and checking for specific patterns, while also improving your problem-solving skills.

Exercise 3**: Uppercase and Lowercase Conversion**
Write a Dart program that takes a string as input and converts it to uppercase and lowercase. Print both the uppercase and lowercase versions of the string.

Questions !

Thank You !

YOUSSEF HOSSAM