**INTRODUCTION TO DART**

Dart is an open-source programming language developed by Google. It is used to create mobile, web and desktop applications. It is also used to develop server-side applications. Dart is a strongly typed language, which means that the compiler will detect any errors in the code before the code is compiled.

Currently, Dart is one of the most preferred languages to learn. A solid understanding of Dart is necessary to develop high-quality apps with flutter. According to GitHub, Dart is one of the most loved programming languages in the world.

If you know languages like C, Java, C#, JavaScript, etc. Dart will be easy for you. This course covers Dart and flutter from basic to advanced.

Dart is a powerful and versatile language that can be used for a wide range of applications. Whether you're a beginner or an experienced programmer, it's worth taking the time to learn this language. It's sure to become an important part of your development toolkit.

**Dart Features**

Dart also has several features that make it easier to learn and use compared to other languages. For example, it has a simple syntax and a flexible type system. It also has built-in support for asynchronous programming, which makes it easier to write code that runs concurrently. Below are interesting things about Dart:

Free and open-source.

Object-oriented programming language.

Used to develop android, iOS, web, and desktop apps fast.

Can compile to either native code or JavaScript.

Offers modern programming features like null safety and asynchronous programming.

You can even use Dart for servers and backend.

**Why Learn Dart?**

There are several reasons why learning Dart can be beneficial:

**Easy to Learn**: Dart has a clean and intuitive syntax, making it easy for beginners to grasp the basics of programming.

**Fast Development**: Dart offers a range of features and tools that enable rapid development, allowing developers to build applications more efficiently.

**Platform-Independent**: Dart can be used to develop applications for various platforms, including web, mobile, and desktop, making it a versatile language.

**Strong Type System:** Dart has a strong type system that helps catch errors at compile-time, resulting in more reliable and bug-free code.

**Object Oriented Language**

Dart is an object-oriented language, which means that it uses objects to organize and structure code. Objects are created from classes, which are templates for creating objects. Classes contain properties and methods, which define the behavior of an object.

To get started with Dart, you will need to set up a development environment. This includes installing the Dart SDK and setting up an editor. Once your development environment is set up, you can begin writing code.

**CREATING A BASIC DART PROGRAM**

Creating a basic program in Dart that prints "Hello World" is a great way to get familiar with the Dart programming language. In this lesson, we will be walking through the steps to create a basic Dart program that prints "Hello World".

Step 1: Create a New Dart Project

Once you have installed the Dart SDK, you can create a new Dart project. This can be done using an IDE (Integrated Development Environment) such as Visual Studio Code or IntelliJ, throughout this course we will be using VS Code. Once you have opened VS Code, you can create a new Dart project by selecting the “Open folder” option and select the folder where you want to store your dart program.

Step 2: Write the Program

This is a simple dart program that prints Hello World on screen. Most programmers write the Hello World program as their first program. Lets do it!

void main() {

   print("Hello World!");

}

Then run the code

**Variables In Dart**

Variables are containers used to store value in the program. There are different types of variables where you can keep different kinds of values.

// here variable name contains value Ian.

var name = "Ian";

**Variables Types In Dart**

They are called data types. We will learn more about data types in the next lesson.

* **String**: For storing text value. E.g. “Miriam” [Must be in quotes]
* **int**: For storing integer value. E.g. 10, -10, 8555 [Decimal is not included]
* **double**: For storing floating point values. E.g. 10.0, -10.2, 85.698 [Decimal is included]
* **num**: For storing any type of number. E.g. 10, 20.2, -20 [both int and double]
* **bool**: For storing true or false. E.g. true, false [Only stores true or false values]
* **var:** For storing any value. E.g. ‘Bimal’, 12, ‘z’, true

**Syntax to declare a variable**

type variableName = value;

**Dart variable Examples**

In this example, you will learn how to declare variables and print their values.

void main() {

// declaring variables

String name = "John"; //must be in quotes

String address = "Kenya";  //must be in quotes

num age = 20; //whole numbers

num height = 5.9;//decimal numbers

bool isMarried = false; //boolean

// printing variables value

print("Name is $name");

print("Address is $address");

print("Age is $age");

print("Height is $height");

print("Married Status is $isMarried");

}

Rules on declaring dart variables

Variable names are case sensitive, i.e., “a” and “A” are different.

A variable name can consist of letters and alphabets.

A variable name cannot start with a number.

Keywords are not allowed to be used as a variable name.

Blank spaces are not allowed in a variable name.

Special characters are not allowed except for the underscore (\_) and the dollar ($) sign.

**Extra Note**

dynamic is a type underlying all Dart objects. You shouldn't need to explicitly use it in most cases.

var is a keyword, meaning "I don't care to notate what the type is here." Dart will replace the var keyword with the initializer type, or leave it dynamic by default if there is no initializer.

Use var if you expect a variable assignment to change during its lifetime:

void main() {

var msg = "Hello world.";

msg = "Hello world again.";

  print(msg);

}

/\*

Output:

Hello world again.

\*/

Use final instead of var if you expect a variable assignment to remain the same during its lifetime:

final msg = "Hello world.";

**Static vs Dynamic Variables**

**Static Variables:**

* A static variable is associated with a class rather than a specific instance of that class.
* It is shared among all instances of the class, meaning there’s only one copy of the static variable.
* Memory allocation for static variables happens once during class loading.
* You can access static variables without creating an object of the class.
* Useful for maintaining values that are consistent across all instances.

Example:

class Employee {

  static var emp\_dept;

  var emp\_name;

  int emp\_salary = 0;

  void showDetails() {

    print("Name of the Employee is: $emp\_name");

    print("Salary of the Employee is: $emp\_salary");

    print("Dept. of the Employee is: $emp\_dept");

  }

}

void main() {

  Employee e1 = Employee();

  Employee e2 = Employee();

  Employee.emp\_dept = "MIS";

  e1.emp\_name = 'Rahul';

  e1.emp\_salary = 50000;

  e1.showDetails();

  e2.emp\_name = 'Tina';

  e2.emp\_salary = 55000;

  e2.showDetails();

}

/\*

Output:

Name of the Employee is: Rahul

Salary of the Employee is: 50000

Dept. of the Employee is: MIS

Name of the Employee is: Tina

Salary of the Employee is: 55000

Dept. of the Employee is: MIS

NOTE: The two employee gets the same department. If you want to change that then use:

Employee.emp\_dept = "ICT";

after the assignmnent for the second employee, let say after line 24( before line 25)

\*/

**Dynamic Variables:**

* A dynamic variable can hold values of any type during runtime.
* You can change its data type freely.
* Useful when you need flexibility but be cautious as it lacks compile-time type checking.

Example:

void main() {

  dynamic x = 12; // type: integer

  x = "Hello world"; // type: string

  x = false; // type: boolean

  print(x);

}

/\*

Output:

false

NOTE: the value was change from 12 (which is into) to "Hello world" (which is a String datatype) and to false (which is a bool datatype). Ordinarily it is not allowed in dart. It is possible because it is a dynamic variable

\*/

Note that using dynamic should be done judiciously, as it can lead to runtime errors if misused.

In summary, static variables are class-level and consistent across instances, while dynamic variables allow flexibility but come with runtime type checking. Choose the right one based on your specific use case! 🚀🔍

**Comments in Dart**

Comments are a great way to enhance the readability of your Dart code and make it easier to understand. In Dart, there are two types of comments: single-line and multi-line comments.

**Single-line Comments**

Single-line comments start with two forward slashes (//). Everything after the two forward slashes is ignored by the compiler. For example:

void main() {

// this is a single line comment

}

**Multi-line Comments**

Multi-line comment is a comment that cut across multiple lines. It start with a forward slash and an asterisk (/\*) and end with an asterisk and a forward slash (\*/). The compiler ignores anything between the starting and ending delimiters. For example:

void main() {

/\* This is a multiline comment

It goes across

Multiple lines

\*/

}

Using Comments

Comments are not only useful for providing information about the code, but they can also be used to temporarily disable certain lines of code. For example, if you have a line of code that is causing an error but you do not want to delete it, you can add a comment before it so that the compiler ignores it.

Conclusion

Comments are a great way to make your code easier to read and understand. They can also be used to temporarily disable lines of code that may be causing errors. Make sure to use comments liberally in your code to make it easier to read and debug.