DART

By: GSS

What is Dart

- Developed by Google
- Object oriented language
- Runs on multiple environment

Installation and Set up

Details

First Program

Dart extension - .dart

Run file - command - dart filename

Fundamentals

1. Static-type
2. Compiled - AOT, JIT
3. Print and read
4. Comments - Inline, block, Documentation
5. Null Safe

Data Types

EVERYTHING IN DART IS AN OBJECT

- Strongly typed and dynamic typed
  a. Int,
  b. double,
  c. String,
  d. bool,
  e. dynamic

Data Types

EVERYTHING IN DART IS AN OBJECT

a. String
b. Type Conversion
c. null
d. Constant

Defining Constants

DART

```dart
void main() {
final city="Mumbai";
final int wheels=4;
print(city);
print(wheels);
const pi=3.14;
const gravity=9.8;
print(pi);
print(gravity);
}
```

Run

```
Mumbai
4
3.14
9.8
```

Control flow statements/conditional statements

```
void main() {
//var marks=50;
var marks=290;
  if(marks>90&&
  print("Your g
}else if(marks>
  print("Your g
  else if(marks
    print("Your
  else if(marks
    print("Your
  else if(marks
    print("Work
  else
    print("Inva
}
```

DART

Run

```
void main() {
  //conditional expression
var a=10,b=50;
int small;
  //exp?true:false
a>b?print("$a is greater") :print("$b is greater");

small=a<b?a:b;
print("$small is smaller");

 //exp1??exp2
String name="Anjali",printName;
  printName=name??"Guest User";
  print(printName);
}
```

```
50 is greater
10 is smaller
Guest User
```

Loops

- for
- while

DART

Run

```dart
void main() {

  // WAP to find the even numbers between 1 to 10

  for (int i = 1; i <= 10; i++) {

    if ( i % 2 == 0) {
      print(i);
    }
  }

  // for ..in loop
  List planetList = ["Mercury", "Venus", "Earth", "Mars"];

  for (String planet in planetList) {
    print(planet);
  }
}
```

```
2
4
6
8
10
Mercury
Venus
Earth
Mars
```

Collections

- List
- Set
- Map

Function Expression

DART

Run

```dart
//Function expression
void main() {

  findPerimeter(4, 2);

  int rectArea = getArea(10, 5);
  print("The area is $rectArea");
}
void findPerimeter(int length, int breadth) => print("The perimeter is ${2 * (length + breadth)}");

int getArea(int length, int breadth) {
  int area = length * breadth;
  return area;
}
/*
 int getArea(int length, int breadth) => length * breadth;*/
```

```
The perimeter is 12
The area is 50
```

Lambda Expression

- Short manner to represent function
- It is used when function returns single expression

Syntax -

```
return_type function_name(arguments) =>
expression;
```

Parameters - Required and optional parameters

DART

Run

```dart
void main() {

  printCities("New York", "New Delhi", "Sydney");
  print("");

  printCountries("USA");  // You can skip the Optional Positional Parameters

}

// Required Parameters
void printCities(String name1, String name2, String name3) {

  print("Name 1 is $name1");
  print("Name 2 is $name2");
  print("Name 3 is $name3");
}

// Optional Positional Parameters
void printCountries(String name1, [String name2, String name3]) {

  print("Name 1 is $name1");
  print("Name 2 is $name2");
  print("Name 3 is $name3");
}
```

```
Name 1 is New York
Name 2 is New Delhi
Name 3 is Sydney

Name 1 is USA
Name 2 is null
Name 3 is null
```

Named parameters

DART

Run

```dart
void main() {
  findVolume(10, breadth: 5, height: 20);
  print("");

  findVolume(10, height: 20, breadth: 5);     //
Sequence doesn't matter in Named Parameter
}


int findVolume(int length, {int breadth, int height}) {

  print("Length is $length");
  print("Breadth is $breadth");
  print("Height is $height");

  print("Volume is ${length * breadth * height}");
}
```

```
Length is 10
Breadth is 5
Height is 20
Volume is 1000

Length is 10
Breadth is 5
Height is 20
Volume is 1000
```

Default parameters

DART

Run

```dart
void main() {

  findVolume(10);     // Default value comes into action
  print("");

  findVolume(10, breadth: 5, height: 30);     // Overrides the old value with new one
  print("");

  findVolume(10, height: 30, breadth: 5);     // Making use of Named Parameters with Default values
}


void findVolume(int length, {int breadth = 2, int height = 20}) {

  print("Lenght is $length");
  print("Breadth is $breadth");
  print("Height is $height");
  print("Volume is ${length * breadth * height}");
}
```

```
Lenght is 10
Breadth is 2
Height is 20
Volume is 400

Lenght is 10
Breadth is 5
Height is 30
Volume is 1500

Lenght is 10
Breadth is 5
Height is 30
Volume is 1500
```

DART

Run

```dart
void main() {

  // Example One: Passing Function to Higher-Order Function
  Function addNumbers = (a, b) => print(a + b);
  someOtherFunction("Hello", addNumbers);


  // Example Two: Receiving Function from Higher-Order Function
  var myFunc = taskToPerform();
  print(myFunc(10));      // multiplyFour(10)         // number * 4       // 10 * 4       // OUTPUT: 40
}



// Example one: Accepts function as parameter
void someOtherFunction(String message, Function myFunction) {       // Higher-Order Function

  print(message);
  myFunction(2, 4);       // addNumbers(2, 4)    // print(a + b);   // print(2 + 4)       // OUTPUT: 6
}


// Example two: Returns a function
Function taskToPerform() {       // Higher-Order Function

  Function multiplyFour = (int number) => number * 4;
  return multiplyFour;
}
```

```
Hello
6
40
```

abstra

The base

A functior
type, and

Open libra

DART

Run

```dart
doSomething(List values, Function func) {
  for (var v in values) {
    var r = func(v);
    print("Input: $v Output: $r");
  }
}

double_num(n) {
   return 2*n;
}

main() {
  doSomething([1, 2, 3], (n) => n*n);

  doSomething([4, 5], double_num);
}
```

```
Input: 1 Output: 1
Input: 2 Output: 4
Input: 3 Output: 9
Input: 4 Output: 8
Input: 5 Output: 10
```

Closures

A closure is a function wrapped inside an outer/parent function. It has access to the variables in the outer/parent function.

A function can modify variables defined in the parent scopes.

OOPS

Classes and Objects

Classes - a template (blueprint) for creating objects (the real *thing*).

Class

Syntax -

```
class class_name {
  // Properties (Instance Variables)
  // Constructor
  // Methods (Functions)
  // Getters and Setters
}
```

Example -

```
class Mobile {
 String color;
 String brandName;
 String calling() {
   return "Mobile can do calling";
 }
 String musicPlay() {
   return "Mobile can play Music";
 }
}
```

Object

Syntax of creating object

```
var object_name = new class_name (arguments);

var myMobile = new Mobile();
```

Assessing properties and methods of class

Syntax -

```
// Accessing Properties
object_name.property_name;
// Accessing Methods
  object_name.method_name;
```

Example -

```
// Accessing properties
   myMobile.color;
   myMobile.brandName;
// Accessing methods
   myMobile.calling();
   myMobile.musicPlay();
```

Constructors

- Default, Parameterised - Refer Github

Encapsulation and Inheritance

- Refer Github

Inheritance

- Single level
- Multilevel

Abstract Class

Refer Github

Interface

Refer Github

Mixin

Mixin is a class that contains methods for use by other classes without having to be the parent class of those other classes.

```
class B {  //B is not allowed to extend any other
class other than object
 method(){
     ....
 }
}

class A with B {
....
......
}
void main() {
 A a = A();
 a.method();  //we got the method without inheriting
B
}
```