DART Notes

Dart is a scalable language that we can used to write simple script or full featured applications whether you are creating mobile application , web application , command line application or server side application. Always dart is a solution for that

It is created by GOOGLE.

Run program -> dart .\file_name.dart

Convert dart file to executable file (exe) :

C:\flutter\bin\cache\dart-sdk\bin\dart2js.bat .\file_name.dart

Run .exe file -> .\file_name.exe

Hello World

```
import 'dart:io';

void main() {
  print("Hello World");
  print(12 / 4);
  print(12 / 5);
  print(true);
}

OUTPUT

Hello World
3.0
2.4
true
```

Input

```
import 'dart:io';

void main() {
  stdout.write('Enter Your Name : ');
  String name = stdin.readLineSync();
  stdout.write('Enter Your Age : ');
  int age = int.parse(stdin.readLineSync());
  print(name);
  print(age);

OUTPUT

Ram
20
```

Data Types And Variables

Data Types :

- *All data types are object in a dart so defualt value is none unless we initialize them.*
  1. Numbers
     - Int
     - Double

  2. Strings
  3. Booleans
  4. Lists (also known as Arrays)
  5. Maps
  6. Runes (for expressing Unicode characters in a String)
  7. Symbols

```
import 'dart:io';

void main() {
  int age1 = 10;
  // or
  var age2 = 20;
  int hexValue = 0xEADEBAEE;

  double age3 = 30;
  String name = "vaibhav";
  String str = 'It\'s string';
  bool isValid = true;
  int a = 5, b = 10;

  print(age1);
  print(age2);
  print(hexValue);
  print(age3);
  print(name);
  print(str);
  print(isValid);
  print("Product of $a and $b is ${a * b}");
  print("Sum of 3 and 4 is ${3 + 4}");
}


OUTPUT

10
20
3940465390
30.0
vaibhav
It's string
true
Product of 5 and 10 is 50
Sum of 3 and 4 is 7
```

Final and Const Keyword

- If we never want to change a value of a variable use **final** and **const** keywords.
- final name =”Peter”;
- const PI = 3.14;

- **Difference**
    - **Final** variable can only be set once and it is initialized when accessed.
    - **Const** variable is implicitly (also) final but it is a complie-time constant , i.e. it is initiliazed during compilation.
- Instance variable can be **final** but cannot be **const**
    - If we want a Constant at Class level then make it **static const.**

```
import 'dart:io';

void main() {
  final name = "Vaibhav";
  final String names = "Vibhu";

  const pi = 3.14;
  const double gravity = 9.8;

  print(name);
  print(names);
  print(pi);
  print(gravity);
}

class Circle {
  final color = "red";
  static const p = 3;
}

OUTPUT

Vaibhav
Vibhu
3.14
9.8
```

Conditions

- Exp1 ?? Exp2 , it checks if Exp1 is null then use Exp2 value;
- Switch state applicable for int and string value only not for bool.

```
import 'dart:io';

void main() {
  var marks = 80;
  // If else if
  if (marks >= 90 && marks <= 100) {
    print("Excellent");
  } else if (marks >= 8 && marks < 90) {
    print("Very good");
  } else {
    print("Good");
  }

  // conditional statement
  int a = 10, b = 20;
  a < b ? print("b is greater") : print("a is greater");

  String name;
  String output = name ?? "Vaibhav";

  print(output);
  // Switch Case Statements;
  // Always pass int or string value only . bool value is not work
  String grade = 'A';
  switch (grade) {
    case 'A':
      print("Excellent");
      break;
    case 'B':
      print("Very Good");
      break;
    default:
      print("Good");
  }

}

OUTPUT

Very good
b is greater
Vaibhav
Excellent
```

```
import 'dart:io';

void main() {
  var i;
  // for loop
  for (i = 1; i <= 1; i++) {
    print("For Loop");
  }
  i = 1;

  // while loop
  while (i <= 1) {
    print("While Loop");
    i++;
  }

  // do while loop
  i = 1;
  do {
    print("DO While Loop");
    i++;
  } while (i <= 1);
  List planet = ["Earth", "Mars"];
  for (i in planet) {
    print(i);
  }
}

OUTPUT

For Loop
While Loop
DO While Loop
Earth
Mars
```

Function

- Collection of statements grouped together to perform an operation.
- Functions in Dart are **Objects**.
  - Functions can be assigned to a variable or passed as parameter to other functions.
- All functions in Dart return a value
  - If no return value is specified the function by default return null.
- When we use **FAT ARROW (=>)** we have not need to return any value and write **return keyword.**
- For **optional parameter use square brackets** in passed functon-> **[String name]**

```
F1(int a,int b) {
      // by default return null
}

int F1(int a,int b) {
      // by default return null
}
```

```
import 'dart:io';

void main() {
  // Functons in Dart are Objects
  print(findArea(5, 6));
  find(5, 10);
  findParameter(5, 12);
  print(findPar(5, 12));

  city("Delhi", "Hapur"); // required parameters
  country("India"); // optional parameter

  print(volume(5, b: 10, h: 15));

  print(f1(2, 3));
  print(f2(2, 3, h: 20));
}

int findArea(int l, int b) {
  return l * b;
}

void find(int l, int b) {
  print(l * b);
}

void findParameter(int l, int b) => print("The Perimeter is ${2 * (l + b)}");

int findPar(int l, int b) => 2 * (l + b);

// Required Parameter
void city(String name1, String name2) {
  print("Name 1 ${name1}");
  print("Name 2 ${name2}");
}

// Optional Parameter
void country(String name1, [String name2, String name3]) {
  print("Name 1 ${name1}");
  print("Name 2 ${name2}");
  print("Name 3 ${name3}");
}

// Optional Named Parameters
int volume(int l, {int b, int h}) => l * b * h;

//Default parameter
int f1(int l, int b, {int h = 10}) => l * b * h;

//Overrides the default parameter
int f2(int l, int b, {int h = 10}) => l * b * h;
```

OUTPUT

```
30
50
The Perimeter is 34
34
Name 1 Delhi
Name 2 Hapur
Name 1 India
Name 2 null
Name 3 null
750
60
120
```

Function

- In Dart ARRAY is known as LIST
- List Types
  - Fixed-length List
    - Length once defined cannot be change
    - All elements are initially **null** until not inintialize
    - Syntax
      - List<datatype> name = List(size);
      - Add elements : name[i]=value;
      - Delete : name[i]=null;

  - Growable List
    - Length is dynamic
    - Syntax
      - List<datatype> name = List(s);
      - Add elements : name.add(value);
      - Delete :
        - **name.remove(element);**
        - **name.removeAt(index);**
        - **name.clear()** : clear the whole list
        - **name[i]=null;**

- **Fixed Size**

```
import 'dart:io';

void main() {
  List<int> n = List(5); // fixed-length list
  print(n[0]);
  n[0] = 1;
  print(n[0]);
  n[0] = 1;
  print(n[1]);
  n[0] = null;
  print(n[0]);

  for (int i in n) {
    print(i);
  }
}

OUTPUT

null
1
null
null
null
null
null
null
null
```

❖ **Growable List**

```
import 'dart:io';

void main() {
List<int> m = List(); // Growable list
  m.add(1);
  m.add(2);
  m.add(3);
  m.add(4);
  print(m);
  m.remove(3);
  print(m);
  m.removeAt(0);
  print(m);
  m.clear();
  print(m);
}

OUPTUT
[1, 2, 3, 4]
[1, 2, 4]
[2, 4]
[]
```

SET

- Unordered Collection of unique elements
    - It does not contain duplicate elements
- We cannot get elements by INDEX , since the items are unordered
- **Syntax:**
    - From list
        - Set<datatype> name = Set.from([val1,val2]);
    - Using Constructor
        - Set <datatype> name = Set();
    - **Insert :** name.add(value);
    - **Delete :** name.remove(value);
    - **Check Set is Empty or not :** name.isEmpty;
    - **Check Element exist or not :** name.contains(value);

HashSET

- Implemenatation of unordered set
- It is based on hash-table based Set implementation

```
import 'dart:io';
```

```
void main() {
  // Method 1 From List
  Set<String> city = Set.from(["Delhi", "Hapur"]);
  city.add("Ghaziabad"); // add elements in a set

  //Method 2 Using Constructor
  Set<int> n = Set();
  n.add(0);
  n.add(2);
  int i;
  String j;
  for (j in city) {
    print(j);
  }

  for (i in n) {
    print(i);
  }

  // check element exist or not
  print(n.contains(2));

  // remove element from set
  city.remove("Delhi");
  print(city);

  // check Set is empty or not
  print(n.isEmpty);
}

OUTPUT

Delhi
Hapur
Ghaziabad
0
2
true
{Hapur, Ghaziabad}
false
```

Map

- It is unordered collection of key-value pair
- Key-value can be of any object type
  - Each KEY in a Map should be unique
  - The VALUE can be repeated
- Commonly called as hash or dictionary
- Size of map is not fixed , it can increase or decrease as per the number of elements
- HashMap
  - Implementation of Map
  - Based on hash-table
- Syntax :
  - Create :
    - Map<datatype,datatype> name =Map();
    - Map<datatype,datatype> name = { key : value , key : value };
  - Insert : name[key]=value;
  - Upadte : name.update(key,(value)=>val);
  - Remove : name.remove(key);
  - Check Length : name.length;
  - Map Empty or not : name.isEmptty;
  - Check key exist or not : name.containsKey(key);
  - Clear Map : name.clear();

```
import 'dart:io';

void main() {
  //Method 1 Using literal
  Map<String, int> m = {"one": 1, "two": 2};
  print(m);

  //Method 2 Using Constructor
  Map<int, String> n = Map();
  n[1] = "One";
  n[2] = "Two";
  print(n);

  for (int key in n.keys) {
    print(key);
  }

  for (String val in n.values) {
    print(val);
  }

  n.forEach((key, value) => print("Key : $key and Value : $value"));

  //Update Value
  n.update(1, (value) => "Ones");
  print(n);

  //remove element
  n.remove(2);
  print(n);

  //check Length
  print(n.length);

  //check map Empty or not
  print(n.isEmpty);

  //check Key
  print(n.containsKey(1));

  //clear Map
  n.clear();
  print(n);
}
```

OUTPUT

```
{one: 1, two: 2}
{1: One, 2: Two}
1
2
One
Two
Key : 1 and Value : One
Key : 2 and Value : Two
{1: Ones, 2: Two}
{1: Ones}
1
false
true
```