Dart ([Dart Tutorial - GeeksforGeeks](https://www.geeksforgeeks.org/dart-tutorial/?ref=lbp))

# Introduction:

Dart is an open-source programming language originally developed by Google. It is meant for both the server side as well as the user side.

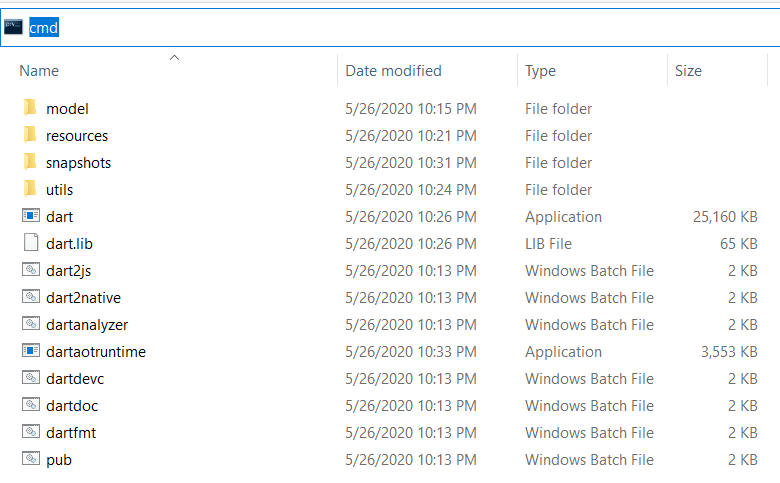
It is object oriented programming language quite similar to Java

Features of dart:

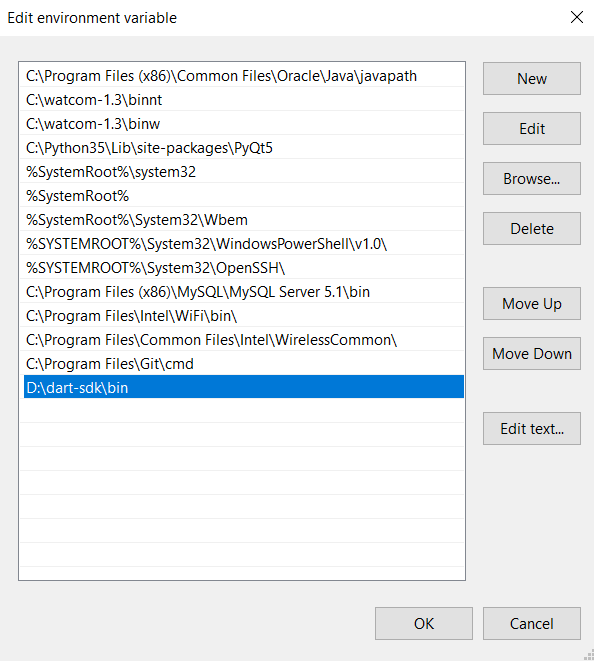
* 1. Asynchronous programming: Dart Supports Asynchronous Programming is where even when a Primary set of tasks are running, Program will respond to other set of tasks and at the end of the execution a final result will be returned.
  2. Flexible compilation and Execution: Dart Compilation is flexible as it supports both JIT(Just-in-Time) compilation as well as AOT(Ahead of Time) compilation. And dart2js adds extra value to it.
  3. Type safe: it is the combination of static and runtime checks to ensure the variable value matches the variable type.
  4. Browser Support
  5. Open Source
  6. Object Oriented Programming
  7. Easy to understand

# Downloading Dart-SDK: (Windows)

Method 1:

1. Directly download form the website and open cmd in the folder type ‘dart’ to complete the process.
2. 

Method 2:

1. choco install dart-sdk
2. choco upgrade dart-sdk
3. Open “Environment variable” and to advance system add the path in the system variable
4. 

## Comment:

1. Single line comment: //
2. Multi-line comment: /\* abc \*/
   1. It similar to java comment
3. Documentation comment: /// abc
   1. Used in packages, software or project
   2. It similar to C# comment

## Variable:

There is 3 types of variable:

1. Static Variable :

* Single declaration: type {variable\_name};
* Multi declaration: type {variable\_1, variable\_2, variable\_3, … variable\_n};
* Type is what data it is? Int, double(float), bool, string, var(depends)

1. Dynamic Variable:

* dynamic variable\_name;
* It can store any value and run the code. It is similar to var declaration
* We can also reassign the value if need

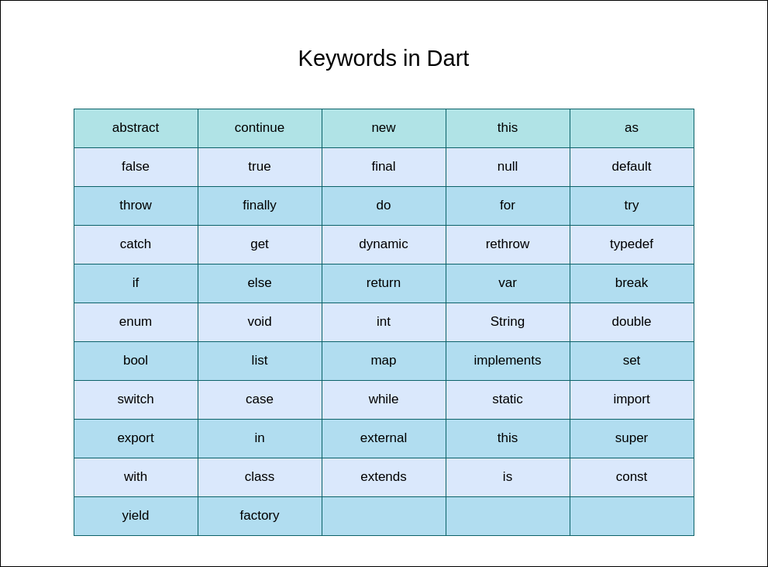
1. Final or const:

* A final variable can only be set once and it is initialized when accessed.
* => final data\_type variable\_name
* A constant variable is a compile-time constant and its value must be known before the program runs.
* => const data\_type variable\_name
* Can’t be changed once declared
* You can declaring it by not specifying the data\_type.

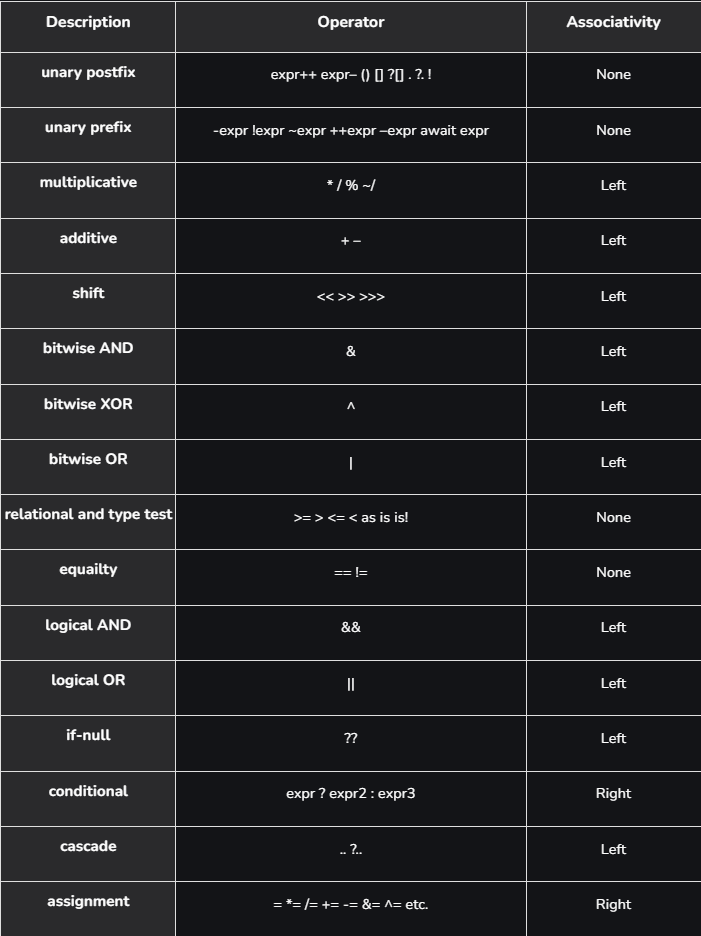
Conditions to Write Variable Name

1. Variable names or identifiers can’t be the keyword.
2. Variable names or identifiers can contain alphabets and numbers.
3. Variable names or identifiers can’t contain spaces and special characters, except the underscore(\_) and the dollar($) sign.
4. Variable names or identifiers can’t begin with a number.

### Keywords in Dart



## Operation in Dart:



# Standard Input and Output

* In dart you must import ‘dart:io’ in order take input and output
* **Taking string :**

// importing dart:io file

import 'dart:io';

void main()

{

print("Enter your name?");

// Reading name of the Geek

String? name = stdin.readLineSync(); // null safety in name string

// Printing the name

print("Hello, $name! \nWelcome to GeeksforGeeks!!");

}

* **Taking Int:**

// Importing dart:io file

import 'dart:io';

void main()

{

// Asking for favourite number

print("Enter your favourite number:");

// Scanning number

int? n = int.parse(stdin.readLineSync()!);

// Here ? and ! are for null safety

// Printing that number

print("Your favourite number is $n");

}

* **Using stdout for output:**

import 'dart:io';

void main()

{

// Printing in first way

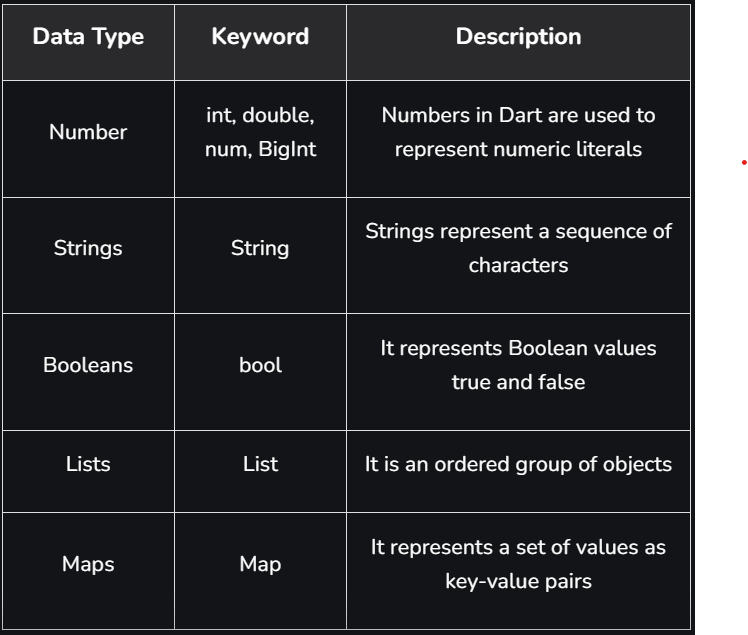
print("Welcome to GeeksforGeeks! // printing from print statement");

// Printing in second way

stdout.write("Welcome to GeeksforGeeks! // printing from stdout.write()");

}

# Datatype:



1. Number:

* It is used to hold the numeric value (64-Bit Max)
* Int; double(float); num(Both int and double)
* Properties:
  + *hashcode:* This property is used to get the hash code of the given number.
  + *isFinite:* If the given number is finite, then this property will return true.
  + *isInfinite:* If the number is infinite, then this property will return true.
  + *isNan:* If the number is non-negative then this property will return true.
  + *isNegative:* If the number is negative then this property will return true.
  + *sign:* This property is used to get -1, 0, or 1 depending upon the sign of the given number.
  + *isEven:* If the given number is an even then this property will return true.
  + *isOdd:* If the given number is odd then this property will return true.
* Methods:
  + *abs():* This method gives the absolute value of the given number.
  + *ceil():* This method gives the ceiling value of the given number.
  + *floor():* This method gives the floor value of the given number.
  + *compareTo():* This method compares the value with other numbers.
  + *remainder():* This method gives the truncated remainder after dividing the two numbers.
  + *round():* This method returns the round of the number.
  + *toDouble():* This method gives the double equivalent representation of the number.
  + *toInt():* This method returns the integer equivalent representation of the number.
  + *toString():* This method returns the String equivalent representation of the number
  + *truncate():* This method returns the integer after discarding fraction digits.

1. Strings:

* It is a sequence of UTF-16 code units. The keyword string used to represent literals.
* It is denoted in single or double quotes.
* You can put the value of an expression inside a string by using ${expression}, is used to concatenate the string easily.
* Raw string are useful when you want to define a string that has a lot of special characters

=> var gfg = r’This is a raw string’;

=> print(gfg) // This is a raw string

1. Booleans:

* It represents True or False.
* bool variable\_name;

1. Lists:

* Is similar to array in other programming languages. It is used to represent a collection of objects.
* Declaration of List
  + Variable size list:
    - Correct way to declare a variable-sized list

List<int> variable\_name = [];

* + - Alternative declaration

List<int> variable\_name2;

* + Fixed size list:
    - Fixed size don’t mean we can’t change the size, but we have predefined limit that it can take.

List<int> variable\_name1 = List<int>.filled(size, 0);

void main() {

// Creating a fixed-size list using List.filled

List<String> gfg = List<String>.filled(3, "default");

// Modifying the elements of the list

gfg[0] = 'Geeks';

gfg[1] = 'For';

gfg[2] = 'Geeks';

// Printing the entire list

print(gfg); // Output: [Geeks, For, Geeks]

// Printing a specific element

print(gfg[0]); // Output: Geeks

}

1. Maps:

* It is a key : value pair, it is a dynamic collection.
* Declaration of Map:
  + There is two type
  + Empty Map:
    - Method 1
      * Map? map\_name;
    - Method 2
      * Map<key\_datatype, value\_datatype>? map\_name;
    - Method 3
      * var map\_name = new map();
  + Declaring map with elements inside it:
    - Method 1
      * Map x = {

key1 : value1;

key2 : value2;

};

* + - Method 2
      * Map<key\_datatype, value\_datatype> map\_name {

key1 : value1;

key2 : value2;

}

* + - Method 3
      * var map\_name{

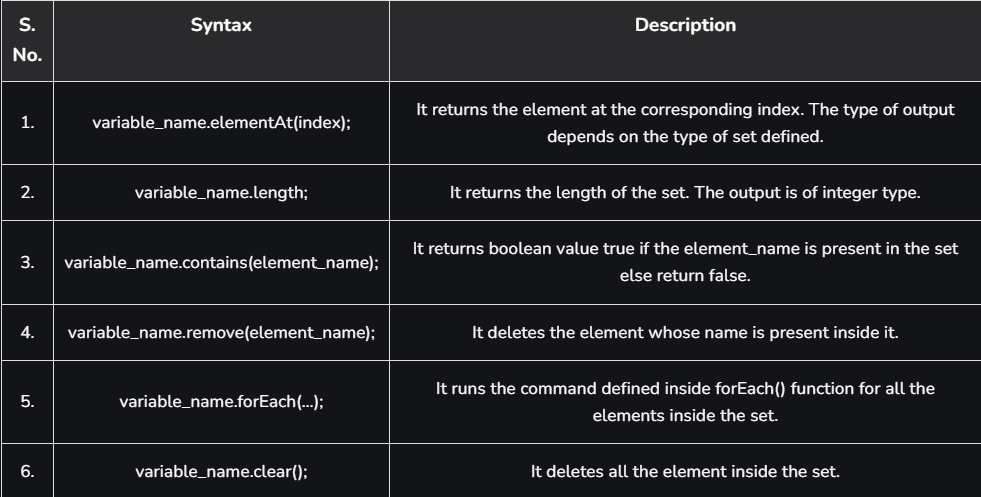
key1 : value1;

key2 : value2;

}

1. Set:

* Set is special case in list where all the inputs are unique i.e. it doesn’t contain any repeated input.
* It can also be interpreted as an unordered array with unique value/input.
* Adding element in set: v\_n.add() or v\_n.addall()



1. Still working on Dart data types

# Control Flow

## Switch case:

* It accepts only condition and not declaring variables.
* Break help in controlling the flow which is must, if not I will throw error.
* The default case is optional.
* You can also build nested switch.

Syntax:

Switch (expression) {

Case value1: {

// body of value1

} break; // if satisfied

Case value 2: {

// body of value 2

} break;

… Continues

Default: {

// body of default case

} break;

}

## Loops:

For loop:

Syntax:

For(initialization; condition; text expression) {

// body of the loop

}

Eg:

// Printing GeeksForGeeks 5 times

void main()

{

for (int i = 0; i < 5; i++) {

print('GeeksForGeeks');

}

}

For … in loop:

In dart takes an expression or object as an iterator.

void main()

{

var GeeksForGeeks = [ 1, 2, 3, 4, 5 ];

for (int i in GeeksForGeeks) {

print(i);

}

} // 1, 2, 3, 4, 5 in vertical order

For each … loop: iterates over all elements in some container/collective and passes the elements to some specific function

Syntax: collection.foreach(void f(value))

Void main() {

Var geeksforgeeks = [1, 2, 3, 4, 5];

Geeksforgeeks.foreach((var num) => print(num)); }

While loop: run until the condition is satisfied.

Syntax:

While (condition) {

Text expression;

// body of loop

}

Eg:

Void main() {

Var gfg = 4;

Int i = 1;

While (I <= gfg) {

Print(‘hello geek’);

i++;

}

}

Do … while loop: the body of the loop will first execute and then it checks for condition is true or not.

Syntax:

Do{

Text expression;

// body of loop

}

Eg:

Void main () {

Var gfg = 4;

Int I = 1;

Do {

Print(‘Hello world’);

I++

}

} while (I <= gfg);

}

## Loop control (Break & continue):

**Break:** used to break the flow of control of the loop i.e. if it is used within a loop then it will terminate the loop whenever encountered. It will bring the flow of control out of the nearest loop.

**Continue:** to the contrary of break, continue statement encounters a loop it skips the iterator to the next iterator.

## Labels in Dart:

Goto and label statement are used to jump from one point to another in c programming, but no in java, Dart don’t have goto but labels can be used with continue and break statements which helps them take bigger leap in the code.

Eg: in Break

Void main() {

// defining the label

Geek1: for(int I = 0; I < 3; I++) {

If (I < 2) {

Print(“you are inside the loop geek”);

// breaking with label

Break geek1;}

Print(“You are still inside the loop ”); }

}

Eg: in continue:

Void main() {

// defining the label

Geek1: for(int I = 0; I < 3; i++) {

If (I < 2) {

Print(“you are inside the loop geek”);

// breaking with label

Continue geek1; }

Print(“You are still inside the loop ”); }

}

Output:

You are inside the loop Geek

You are inside the loop Geek

You are still inside the loop