Dart –

Dart Setup

VS code

Android studio

Github

Dartpad

Dart – Basics and Data Types

- Dart is easy to learn if you know any of Java, C++, JavaScript, etc.
- The simplest "Hello World" program gives the idea of the basic syntax of the programming language. It is the way of testing the system and working environment.
- There are several ways to run the first program, which is given below:
  - Using Command Line
  - Using IDE

File Name

BAD:

- `filesystem.dart`
- `file-system.dart`
- `SliderMenu.dart`

GOOD:

- `slider_menu.dart`
- `file_system.dart`

Variable in Dart

A variable is “a named space in the memory” that stores values. In other words, it acts a container for values in a program.

Variable in Dart

- Identifiers cannot be keywords.
-
- Identifiers can contain alphabets and numbers.
- int numb er1=0;
- Identifiers cannot contain spaces and special characters, except the underscore (_) .

- Variable names cannot begin with a number.

Variable in Dart

abstract	continue	new	this	as
false	true	final	null	default
throw	finally	do	for	try
catch	get	dynamic	rethrow	typedef
if	else	return	var	break
enum	void	int	String	double
bool	list	map	implements	set
switch	case	while	static	import
export	in	external	this	super
with	class	extends	is	const
yield	factory			

Syntax

- A variable must be declared before it is used.
- var name = 'Smith';

```
void main() {
   String name = "1";
}
```

Data Types

- whenever a variable is created, each variable has an associated data type

  INT

  String

  List

  Map

Task

- **Swapping of two number?**

Semicolon

- The semicolon is used to terminate the statement that means, it indicates the statement is ended here. It is mandatory that each statement should be terminated with a semicolon(;).

  We can write multiple statements in a single line by using a semicolon as a delimiter. The compiler will generate an error if it is not use properly.

  Example -
  var msg1 = "Hello World!";

  var msg2 = "How are you?"

Comments

- Comments are the set of statements that are ignored by the Dart compiler during the program execution. It is used to enhance the readability of the source code.

  Generally, comments give a brief idea of code that what is happening in the code.

  We can describe the working of variables, functions, classes, or any statement that exists in the code.

  Programmers should use the comment for better practice.

Comments

- Dart provides three kinds of comments
  - Single-line Comments
  - Multi-line Comments

Single-line Comment

- We can apply comments on a single line by using the // (double-slash). The single-line comments can be applied until a line break.

Example - void

```
main(){

   // This will print the given statement on screen
   print("Welcome to class");
}
```

Multi-line Comment

- Sometimes we need to apply comments on multiple lines; then, it can be done by using /*…..*/. The compiler ignores anything that written inside the /*…*/, but it cannot be nested with the multi-line comments. Let's see the following example.

  Example - void

```
main(){

   /* This is the example of multi-line comment
   This will print the given statement on screen */

   print("Welcome to SMIT");

}
```

Operators

- Arithmetic Operators
- Relational Operators
- Logical Operators

Arithmetic Operators

- +
- -
- *
- /
- %

Arithmetic Operators

- void main(){
- print("Example of Assignment operators");
- var n1 = 10;
- var n2 = 5;
- print("n1+n2 = ${n1+n2}");
- print("n1-n2 = ${n1-n2}");
- print("n1*n2 = ${n1*n2}"); print("n1/n2 = ${n1/n2}");
  print("n1%n2 = ${n1%n2}");
- }

Relational Operators

==

- !=
- <
- >
- <=
- >=
-

Relational Operators

```
void main()
{
int a = 10, b = 20;
 print('Equal to: ${a == b}');
 print('Not Equal to: ${a != b}');
print('Greater than: ${a > b}');
print('Less than: ${a < b}');
print('Greater than or equal to: ${a >= b}');
print('Less than or equal to: ${a <= b}');
 }
```

Logical Operators

- &&
- ||

Logical Operators

```
void main()
    {
    bool isTrue = true, isFalse = false;
    print('AND Operator: ${isTrue && isFalse}');
    print('OR Operator: ${isTrue || isFalse}');
    print('NOT Operator: ${!isTrue}');
     }
```

Increment/decrement

- Dart has special type of operators known as Increment/Decrement operators
- Increment and Decrement operators are used to increment and decrement the value of the particular value by 1 respectively.
- Increment operators are denoted using ++ while decrement operators are denoted using — symbol
- ++a/a++ –a/a–

Increment/decrement

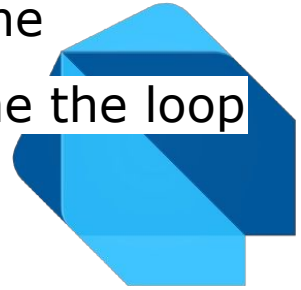
```
void main(){
int a =10, b=20, c=30, d=40;

int preAdd=--a;
int preSub=--b;

print("value of the a is $a and value of pre increment expression is ${preAdd}");
print("value of the b is $b and value of pre decrement expression is ${preSub}");

int postAdd=c++;
int postSub=d--;

print("value of the a is $c and value of post increment expression is ${postAdd}");
print("value of the b is $d and value of post decrement expression is ${postSub}");
}
```

Loops

- A loop represents a set of instructions that must be repeated
- certain instructions require repeated execution. Loops are an ideal way to do the same.

Loops

- The **for** loop is an implementation of a definite loop. The for loop executes the code block for a specified number of times.
- The **while** loop executes the instructions each time the condition specified evaluates to true.
- The **do…while** loop is similar to the while loop except that the do...while loop doesn’t evaluate the condition for the first time the loop executes.



For Loops

```
print(“Dart is fun”);
for (initialization; condition; increment/decrement operation)
{
 body
 }
```

For Loops

```
print(“Dart is fun”);
for (var i = 0; i < 5; i++)
{
i++
i=i+1;
 print('!');
 }
```

For Loops

Display Sum of n Natural Numbers Using For Loop

while Loops

```
while(condition)
{ //statement(s);
// Increment (++) or Decrement (--)
Operation;
 }
```

While Loops

```
void main()
{
int i = 1;
while (i <= 10)
{
 print(i);
 i++;
}
}
```

While Loops

```
void main()
{
int i = 10;
while (i >= 1)
{
 print(i);
 i- -;
}
}
```

Do-While Loops

```
do
{
statement1;
statement2;
…
statementN;
}
while(condition);
```

Do-While Loops

```
void main()
{
int i = 1;
do
{
 print(i);
 i++;
 }
 while (i <= 10); }
```

while Loops

Develop a game:

user should enter a number and our program guess a number from their side if the number is smaller than the guess number, then the program prompt the user to enter greater number. if the number is greater then the guess number, the game show user to enter small number and the program will continue until the number is pound and then excite the game.

Task

1. Write a program to print numbers from 1 to 10.
2. Write a program to calculate the sum of the first 10 natural numbers.
3. Write a program that prompts the user to input a positive integer. It should then print the multiplication table of that number.
4. Write a program to find the factorial value of any number entered through the keyboard.
5. Two numbers are entered through the keyboard. Write a program to find the value of one number raised to the power of another.
6. Write a program that take prompt from user that how much number he want to enter and then find sum of all that numbers.
7. Write a program that prompts the user to input a positive integer. It should then output a message indicating whether the number is a even or odd number.

List in Dart

- List stores a collection of elements in a specific order.
- Lists are commonly used to hold multiple items of the same type, such as integers, strings, or objects.
- lists are ordered collections of objects, and they are represented using the **List**

Characteristics of List

- **Ordered**: Lists maintain the order of elements as they are added.
- **Mutable**: Elements in a list can be modified, added, or removed after the list is created.
- **Dynamic Size**: Lists can grow or shrink in size as elements are added or removed.
- **Index-Based Access**: Elements in a list are accessed using indices, typically starting from 0 for the first element.

Structure of List

List<Datatype> NameofList=[ ];

example:

```
// Creating a list of integers
List<int> numbers = [1, 2, 3, 4, 5];

// Creating a list of strings
List<String> fruits = ['Apple', 'Banana', 'Orange'];

// Creating an empty list
List<double> temperatures = [];
```

Structure of List

List<Datatype> NameofList=[ ];

example:

```
 // Creating a list of integers
  List<int> numbers = [1, 2, 3, 4, 5];


// Creating a list of strings
  List<String> fruits = ['Apple', 'Banana', 'Orange'];


// Creating an empty list
  List<double> temperatures = [];

print(listname);
```

Example of List

1)

```
void main() {
   var num_list = [1,2,3];
   print(num_list);
}
```

2)

```
void main() {
   List<int> lst=[];
   lst.add(12);
   lst.add(13);
   print(lst);
}
```

Properties of List

1)first

Returns the first element in the list.

2)isEmpty

Returns true if the collection has no elements.

3)isNotEmpty

Returns true if the collection has at least one element.

3)Length

Returns the size of the list.

4)Reversed

Returns an iterable object containing the lists values in the reverse order.

Operation in List

Accessing Elements in List:

```
 List<int> numbers = [1, 2, 3];
// Accessing elements by index
int firstNumber = numbers[0]; // 1
int lastNumber = numbers[numbers.length - 1]; // 3
```

Updating Elements in List:

```
List<int> numbers = [1, 2, 3];
// Updating an element at a specific index
numbers[1] = 5; // [1, 5, 3]
```

Operation in List

Adding Elements in List:

```
 List<int> numbers = [1, 2, 3];
// Adding a single element at the end
numbers.add(4);
[1,2,3,4]

// Adding multiple elements at the end
numbers.addAll([5, 6]);

[1,2,3,4,5,6]
```

Operation in List

Removing Elements in List:

```
 List<int> numbers = [1, 2, 3];
// Removing an element by value
numbers.remove(2); // [1, 3]
// Removing an element at a specific index
numbers.removeAt(0); // [3]
```

Checking length of List:

```
List<int> numbers = [1, 2, 3];
// Getting the length of the list
int length = numbers.length; // 3
// Checking if the list is empty
bool isEmpty = numbers.isEmpty; // false
```

Operation in List

Iterating Elements in List:

```
 List<int> numbers = [1, 2, 3];

// Using a for loop
for (int i = 0; i < numbers.length; i++) {
  print(numbers[i]);
}

// Using forEach
numbers.forEach((number) => print(number));
```

Task

1. Write a program to calculate the sum of the all elements of the static list.
2. Write a program that prompts the user to input a positive integer, then develop a list in which it take element from user have same length of the user input number.print the list?
3. Write a program that prompts the user to input a positive integer, then develop a list in which it take element from user have same length of the user input number.print the list and find sum of the all elements in the list?
4. Write a program that prompts the user to input a positive integer, then develop a list in which it take element from user have same length of the user input number.print the list and find sum of the all elements in the list? and then find sum of even number in the list and sum of odd numbers in the list. check wether sum of all number and sum of even and odd number is equal or not?

Maps in Dart

- Maps are dictionary-like data types that exist in key-value form.
- Maps are very flexible and can mutate their size based on the requirements.
- No restriction on the type of data that goes in a map data type

Maps in Dart

- Maps are dictionary-like data types that exist in key-value form.
- Maps are very flexible and can mutate their size based on the requirements.
- No restriction on the type of data that goes in a map data type

Maps Syntax in Dart

```
var NameOfTheMap =  Map();
NameOfTheMap[key] = value

var maps = { 'key1': 'val1', 'key2': 'val2',};
```

Maps properties in Dart

1)Keys

Returns an iterable object representing keys

2)values

Returns an iterable object representing values

3)isNotEmpty

Returns true if the Map is an empty Map

4)Length

Returns the size of the Map

Maps properties in Dart

```
void main() {
// Creating Map using is literals
  var maps = {
    'key1': 'valq',
    'key2': 'val2',
    'key3': [1, 2, 3, 4, 5]
  };
  print(maps.keys);
}
```

Functions in Dart

- Function is a set of statements that take inputs, do some specific computation and produces output.
- Functions are created when certain statements are repeatedly occurring in the program and a function is created to replace them.
- Functions make it easy to divide the complex program into smaller sub-groups and increase the code reusability of the program.

Syntax of Functions

```
return_type function_name ( parameters ) {
   // Body of function
   return value;
}
```

Example of Functions

```
void fun()
{
        // Creating function
        print("Welcome to SMIT");
}

void main()
{
        // Calling the function
        fun();
}
```

Example of Functions

```
int add(int a, int b)
{
        // Creating function
        int result = a + b;
        // returning value result
        return result;
}

void main()
{
        // Calling the function
        var output = add(10, 20);

        // Printing output
        print(output);
}
```

Functions with optional parameter

Optional Positional Parameter

To specify it use square (‘[ ]’) brackets

Optional Named parameter

When we pass this parameter it is mandatory to pass it while passing values. It is specify by curly(‘{ }’) brackets.

Optional parameter with default values

Here parameters are assign with default values.

Functions with optional parameter

```
void gfg1(int g1, [ var g2 ])
{
        // Creating function 1
        print("g1 is $g1");
        print("g2 is $g2");
}

void gfg2(int g1, { var g2, var g3 })
{
        // Creating function 1
        print("g1 is $g1");
        print("g2 is $g2");
        print("g3 is $g3");
}

void gfg3(int g1, { int g2 : 12 })
{
        // Creating function 1
        print("g1 is $g1");
        print("g2 is $g2");
}
```

Functions with optional parameter

```
void main()
{
        // Calling the function with optional parameter
        print("Calling the function with optional parameter:");
        gfg1(01);

        // Calling the function with Optional Named parameter
        print("Calling the function with Optional Named parameter:");
        gfg2(01, g3 : 12);

        // Calling function with default valued parameter
        print("Calling function with default valued parameter");
        gfg3(01);
}
```

Lambda/Arrow functions

```
// Lambda/arrow function in Dart

void fun() => print("Welcome to SMIT");

void main()
{
	fun(); // Calling Lambda function
}
```

Git & Github Introduction

What Git?

Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who are collaboratively developing source code during software development.

Why Version Control?

Scenario : Multiple students are doing a project together

What GitHub.com?

- GitHub.com is **a site for online storage of Git repositories**.

- You can get free space for open source projects

Download and install Git

- Here's the standard one: http://git-scm.com/downloads
-

Download and install Git

- Here's the standard one: http://git-scm.com/downloads