Introduction to Dart Programming Language

انت لا تتعلم المشي باتباع القواعد. انت تتعلم بالممارسة والسقوط.

WHY USE DART?

- Dart is an object-oriented programming (OOP) language.
- class-based style.
- uses a C-style syntax.

What are some of the benefits of using Dart?

– Dart is **ahead-of-time** (**AOT**) compiled to native code. **AOT** compilation is used when compiling your app for release mode (such as to the Apple **App Store** and **Google Play**).

– Dart is **just-in-time** (**JIT**) compiled, making it fast to display your code changes such as via Flutter's **stateful hot reload** feature.

➢ Dart is a client-optimized language for developing fast apps on any platform.

➢ Dart is the open-source programming language originally developed by Google.

➢ It is meant for both server side as well as the user side.

➢ The Dart software development kit (SDK) comes with its compiler – the **Dart VM** and a utility dart2js which is meant for generating **Javascript equivalent of a Dart Script** so that it can be run on those sites also which don't support Dart.

➢ Dart is **Object-oriented language** and is quite similar to that of **Java Programming**.

➢ Dart is extensively use to create single-page websites and web-applications. Best **example** of dart application is **Gmail**.

➢ Dart has built-in sound null safety. This means values can't be null unless you say they can be.
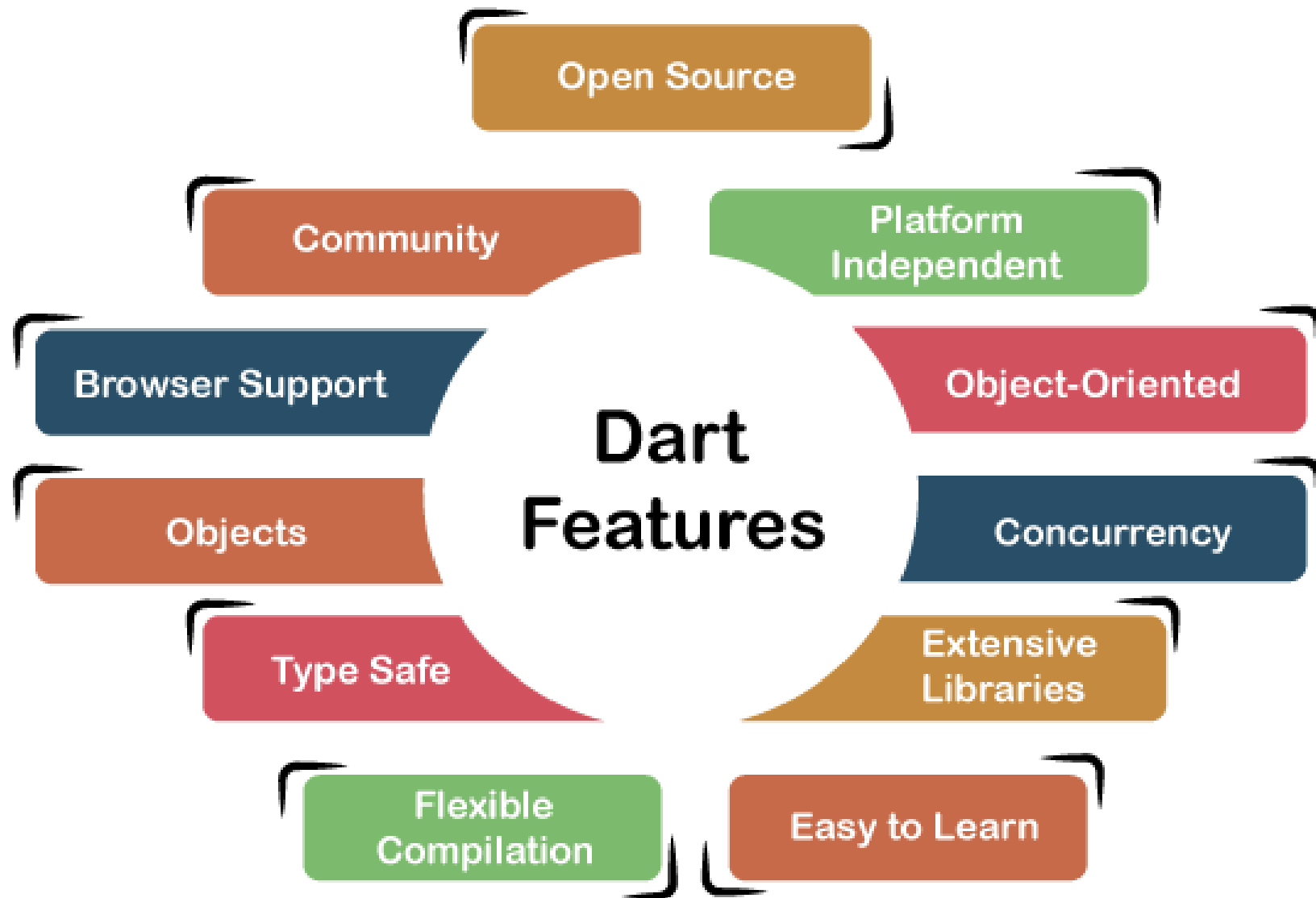
First Code in Dart:

In dart main() function is predefined method and acts as the entry point to the application. A dart script needs the main() method for execution of the code. The program code goes like this:

```dart
void main() {

    print("Welcome to Dart");
}
```

Output:

Welcome to Dart

Execution of program:

1.Online Compiler: The online compiler which support Dart is **Dart Pad**.
2.IDE: The IDEs which support Dart are VsCode, Eclipse, etc.
3.The dart program can also be compiled through terminal by executing the code **dart file_name.dart**.

Comments are a set of statements that are not meant to be executed by the compiler. They provide proper documentation of the code.

Types of Dart Comments:

1. Dart Single line Comment.

2. Dart Multiline Comment.

3. Dart Documentation Comment.

1. **Dart Single line Comment:** Dart single line comment is used to comment a line until line break occurs. It is done using a double forward-slash (//).

```dart
void main()
{
double area = 3.14 * 4 * 4;
// It prints the area
// of a circle of radius = 4
print(area);
}
```

2. Dart Multi-Line Comment: Dart Multiline comment is used to comment out a whole section of code. It uses '/*' and '*/' to start and end a multi-line comment respectively.

```dart
void main()
{
var lst = [1, 2, 3];
/*
It prints
the whole list
at once
*/
print(lst);
 }
```

Dart – Variables

A variable name is the name assign to the memory location where the user stores the data and that data can be fetched when required with the help of the variable by calling its variable name. There are various types of variable which are used to store the data. The type which will be used to store data depends upon the type of data to be stored.

Syntax: To declare a variable:

type variable_name;

Syntax: To declare multiple variables of same type:

type variable1_name, variable2_name, variable3_name, ...variableN_name;

Type of the variable can be among:

1. Integer   2. Double   3. String   4. Booleans   5. Lists   6. Maps

Conditions to write variable name or identifiers are as follows:

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

```dart
void main() {
  int x1 = 10;        // Declaring and initializing a variable
  double x2 = 0.2;    // Declaring another variable
  bool x3 = false;
  // Declaring multiple variable
  String x4 = "0", x5 = "Welcome to Dart";
  // Printing values of all the variables
  print(x1); // Print 10
  print(x2); // Print 0.2
  print(x3); // Print false
  print(x4); // print 0
  print(x5); // Print Welcome to Dart
}
```

```
10
0.2
false
0
Welcome to Dart
```
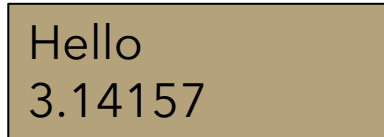
Keywords in Dart:

Keywords are the set of reserved words which can't be used as a variable name or identifier because they are standard identifiers whose function are predefined 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			

Dynamic type variable in Dart:

This is a special variable initialized with keyword **dynamic.** The variable declared with this data type can store implicitly any value during running the program. It is quite similar to **var** datatype in Dart, but the difference between them is the moment you assign the data to variable with **var** keyword it is replaced with the appropriate data type.

```dart
void main() {
  // Assigning value to str variable
  dynamic str = "Hello";

  // Printing variable str
  print(str);

  // Reassigning the data to variable and printing it
  str = "3.14157";
  print(str);
}
```

```
Hello
3.14157
```

Final And Const Keyword in Dart:

These keywords are used to define constant variable in Dart . once a variable is defined using these keyword then its value can't be changed in the entire code. These keyword can be used with or without data type name.

```dart
void main() {
// Assigning value to v1 variable without datatype
  final v1 = "Hello";
// Printing variable v1
  print(v1);

// Assigning value to v2 variable with datatype
  final String v2 = "Hello Again!!";
// Printing variable v2
  print(v2);
}
```

```
Hello
Hello Again!!
```

Use final when you need variables that cannot be reassigned but can be computed at runtime. Use const for values that are known at compile time to enhance performance and ensure immutability.

Operators in Dart

The operators are special symbols that are used to carry out certain operations on the operands. The Dart has numerous built-in operators which can be used to carry out different functions, for example, '+' is used to add two operands. Operators are meant to carry operations on one or two operands.

1. Arithmetic Operators:

This class of operators contain those operators which are used to perform arithmetic operation on the operands. They are binary operators i.e they act on two operands. They go like this:

Operator Symbol	Operator Name	Operator Description
+	Addition	Use to add two operands
–	Subtraction	Use to subtract two operands
-expr	Unary Minus	It is Use to reverse the sign of the expression
*	Multiply	Use to multiply two operands
/	Division	Use to divide two operands
~/	Division	Use two divide two operands but give output in integer
%	Modulus	Use to give remainder of two operands

2. Relational Operators:

This class of operators contain those operators which are used to perform relational operation on the operands. It goes like this:

Operator Symbol	Operator Name	Operator Description
>	Greater than	Check which operand is bigger and give result as boolean expression.
<	Less than	Check which operand is smaller and give result as boolean expression.
>=	Greater than or equal to	Check which operand is greater or equal to each other and give result as boolean expression.
<=	less than equal to	Check which operand is less than or equal to each other and give result as boolean expression.
==	Equal to	Check whether the operand are equal to each other or not and give result as boolean expression.
!=	Not Equal to	Check whether the operand are not equal to each other or not and give result as boolean expression.

3. Type Test Operators:

This class of operators contain those operators which are used to perform comparison on the operands. It goes like this:

Operator Symbol	Operator Name	Operator Description
is	is	Gives boolean value true as output if the object has specific type
is!	is not	Gives Boolean value false as output if the object has specific type

4. Assignment Operators:

This class of operators contain those operators which are used to assign value to the operands.

Operator Symbol	Operator Name	Operator Description
=	Equal to	Use to assign values to the expression or variable
??=	Assignment operator	Assign the value only if it is null.

```
void main() {
  int a = 5;
  int b = 7;
  // Assigning value to variable c
  var c = a * b;
  print(c);
  var d;      // Assigning value to variable d
   print(d ??= a + b);
  // Again trying to assign value to d
  d ??= a - b; // Value is not assign as it is not null
  print(d); }
```

```
35
12
12
```

5. Logical Operators:

This class of operators contain those operators which are used to logically combine two or more conditions of the operands. It goes like this:

Operator Symbol	Operator Name	Operator Description
&&	And Operator	Use to add two conditions and if both are true than it will return true.
‖	Or Operator	Use to add two conditions and if even one of them is true than it will return true.
!	Not Operator	It is use to reverse the result.

6. Conditional Operators:

This class of operators contain those operators which are used to perform comparison on the operands.

Operator Symbol	Operator Name	Operator Description
condition ? expersion1 : expersion2	Conditional Operator	It is a simple version of if-else statement. If the condition is true than expersion1 is executed else expersion2 is executed.
expersion1 ?? expersion2	Conditional Operator	If expersion1 is non-null returns its value else returns expression2 value.

```
void main() {
  int a = 5;
  // Conditional Statement
  var c = (a < 10) ? "Statement is Correct" : "Statement is Wrong";
  print(c);
  // Conditional statement
  int? n;
  var d = n ?? "n has Null value";
  print(d);
    n = 10;    d = n;
  print(d);}
```

```
Statement is Correct
n has Null value
10
```

Standard Input in Dart:

In Dart programming language, you can take standard input from the user through the console via the use of **readLineSync()** function. To take input from the console you need to import a library, named **dart:io** from libraries of Dart.

About Stdin Class:

This class allows the user to read data from standard input in both synchronous and asynchronous ways. The method **readLineSync()** is one of the methods used to take input from the user.

Taking a string input from user:

```dart
import 'dart:io';
void main() {
  print("Enter your name?");
  // Reading name from user
  String? name = stdin.readLineSync(); // null safety in name string

  // Printing the name
  print("Hello, $name! \n Welcome to Dart!!");
}
```

Taking integer value from user:

```dart
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");
}
```

Standard Output in Dart:

In dart, there are two ways to display output in the console:

1. Using print statement.
2. Using stdout.write() statement.

Printing Output in two different ways:

```dart
import 'dart:io';

void main() {
  // Printing in first way
  print("Welcome to Dart! // printing from print statement");

  // Printing in second way
  stdout.write("Welcome to Dart! // printing from stdout.write()");
}
```

Note:
The **print()** statement brings the cursor to next line while **stdout.write()** don't bring the cursor to the next line, it remains in the same line.