

Flutter & Dart Complete Guide

Prepared by Eng. Yousef Salah Kassem

DAFL 1
Dart Basics & Data Types

Lecture 1



Chapter
01

Table Of Content

- Introduction
- Main Method
- Hello World
- Variables Declaration and Initialization
- Comments
- Basic Types
- Operators
- Expressions & Statements



Introduction

Introduction

- ◆ Dart is easy to learn if you know any of Java, C++, JavaScript, etc.
- ◆ There are several ways to run the first program, which is given below
 - Using Command Line
 - Running on Browser
 - Using IDE



Chapter

01

Hello World

```
void main() {  
    print("Hello World!");  
}
```

<https://zapp.run/>

Variable Declaration & Initialization

Chapter
01

Identifiers

- ❖ Identifiers are the name which is used to define variables, methods, class, and function, etc.
- ❖ An Identifier is a sequence of the letters([A to Z],[a to z]), digits([0-9]) and underscore(_), but remember that the first character should not be a numeric.



Chapter
01

Identifiers

- ❖ The first character should not be a digit.
- ❖ Special characters are not allowed except underscore (_) or a dollar sign (\$).
- ❖ Two successive underscores (__) are not allowed.
- ❖ The first character must be alphabet(uppercase or lowercase) or underscore.
- ❖ Identifiers must be unique and cannot contain whitespace.
- ❖ They are case sensitive. The variable name **Tushar** and **tushar** will be treated differently.



Chapter
01

Printing and String Interpolation

- ❖ The `print()` function is used to print output on the console, and `$expression` is used for the string interpolation. Below is an example

```
void main() {  
  
    var name = "Tushar";  
    var marks = 78.56;  
    print("My name is ${name} My marks are ${marks}");  
}
```



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?"
```



Whitespace and Line Breaks

- ❖ The Dart compiler ignores whitespaces. It is used to specify space, tabs, and newline characters in our program.
- ❖ It separates one part of any statement from another part of the statement.
- ❖ We can also use space and tabs in our program to define indentation and provide the proper format for the program.
- ❖ It makes code easy to understand and readable.



Block

- ❖ The block is the collection of the statement enclosed in the curly braces. In Dart, we use curly braces to group all of the statements in the block.
- ❖ Consider the following syntax.

Syntax:

```
{ //start of the block
```

```
//block of statement(s)
```

```
// end of the block
```



Live Coding

Comments

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.



- ❖ Dart provides three kinds of comments
 - Single-line Comments
 - Multi-line Comments
 - Documentation 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 MITU Skillologies");

}



Chapter
01

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 MITU Skillologies");  
}
```



Chapter
01

Documentation Comment

- The document comments are used to generate documentation or reference for a project/software package. It can be a single-line or multi-line comment that starts with `///` or `/*`. We can use `///` on consecutive lines, which is the same as the multiline comment.
- These lines ignore by the Dart compiler expect those which are written inside the curly brackets. We can define classes, functions, parameters, and variables. Consider the following example.
- Syntax

```
///This  
///is  
///a example of  
/// multiline comment
```



Live Coding

Data Types

Chapter
01**Data Types**

- ❖ The data types are the most important fundamental features of programming language.
- ❖ In Dart, the data type of the variable is defined by its value.
- ❖ The variables are used to store values and reserve the memory location.
- ❖ The data-type specifies what type of value will be stored by the variable. Each variable has its data-type.
- ❖ The Dart is a static type of language, which means that the variables cannot modify.



❖ Dart supports the following built-in Data types.

- Number
- Strings
- Boolean
- Lists
- Maps
- Runes

Chapter
01

Numbers

- ❖ The Darts Number is used to store the numeric values. The number can be two types - integer and double.
 - ❖ Integer - Integer values represent the whole number or non-fractional values. An integer data type represents the 64-bit non-decimal numbers between -2^{63} to 2^{63} . A variable can store an unsigned or signed integer value. Ex.
- int marks = 80;
- ❖ Double - Double value represents the 64-bit of information (double-precision) for floating number or number with the large decimal points. The double keyword is used to declare the double type variable.

double pi = 3.14;



Strings

- ❖ A string is the sequence of the character. If we store the data like - name, address, special character, etc.
- ❖ It is signified by using either single quotes or double quotes. A Dart string is a sequence of UTF-16 code units.

```
var msg = "Welcome to MITU";  
print("सुस्वागतम");
```



Chapter
01

Strings

- ❖ String is a sequence of the character or UTF-16 code units. It is used to store the text value. The string can be created using single quotes or double-quotes.
- ❖ The multiline string can be created using the triple- quotes. Strings are immutable; it means you cannot modify it after creation.
- ❖ In Dart, The String keyword can be used to declare the string.



Chapter

01

Strings

```
String msg = "Welcome to MITU";
```

or

```
String msg1 = "This is double-quoted string example.";
```

or

```
String msg2 = ''' line1
```

```
line2
```

```
line3'''
```



Strings

- ❖ The + or += operator is used to merge the two string.
- ❖ String Interpolation
 - The string interpolation is a technique to manipulate the string and create the new string by adding another value.
 - It can be used to evaluate the string including placeholders, variables, and interpolated expression.
 - The `${expression}` is used for string interpolation. The expressions are replaced with their corresponding values.

Boolean

- ❖ The Boolean type represents the two values - true and false.
- ❖ The bool keyword uses to denote Boolean Type.
- ❖ The numeric values 1 and 0 cannot be used to represent the true or false value.
- ❖ `bool isValid = true;`



Chapter
01

List

- ❖ The list is a collection of the ordered objects (value).
- ❖ The concept of list is similar to an array. An array is defined as a collection of the multiple elements in a single variable.
- ❖ The elements in the list are separated by the comma enclosed in the square bracket[].
- ❖ The sample list is given below. `var list = [1,2,3]`



- ❖ The maps type is used to store values in key-value pairs. Each key is associated with its value.
- ❖ The key and value can be any type. In Map, the key must be unique, but a value can occur multiple times.
- ❖ The Map is defined by using curly braces ({}), and comma separates each pair.

```
var student = {'name': 'Rajesh', 'age':22, 'Branch': 'Statistics'}
```



Chapter
01

Runes

- ❖ As we know that, the strings are the sequence of Unicode UTF-16 code units. Unicode is a technique which is used to describe a unique numeric value for each digit, letter, and symbol.
- ❖ Since Dart Runes are the special string of Unicode UTF-32 units. It is used to represent the special syntax.
- ❖ For example -The special heart character ♥ is equivalent to Unicode code \u2665, where \u means Unicode, and the numbers are hexadecimal integer.
- ❖ If the hex value is less or greater than 4 digits , it places in a curly bracket ({}). For example - An emoji 😊 is represented as \u{1f600}.

Example

```
void main( ) {  
    var heart_symbol = '\u2665';  
    var laugh_symbol = '\u{1f600}';  
    print(heart_symbol);  
    print(laugh_symbol);  
}
```

Live Coding

Operators

Types of Dart Operators



Chapter

01

Arithmetic Operators

❖ +

❖ -

❖ *

❖ /

❖ %

Chapter
01

Arithmetic Operators

```
void main(){
print("Example of Assignment operators");

varn1= 10;
varn2 = 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}");
}
```

Increment and Decrement

- ❖ ++ and -- operators are known as increment and decrement operators and also known as unary operators, respectively.
- ❖ Unary operators, operate on single operand where ++ adds 1 to operands and -- subtract 1 to operand respectively.
- ❖ The unary operators can be used in two ways - postfix and prefix.
- ❖ If ++ is used as a postfix (like x++), it returns the value of operand first then increments the value of x. If -- is used as a prefix (like --x), it increases the value of x.

Chapter

01

Assignment Operators

❖ =

❖ +=

❖ -=

❖ *=

❖ %=

Chapter

01

Relational Operators

❖ ==

❖ !=

❖ <

❖ >

❖ <=

❖ >=

Type Test Operators

- ❖ **as**
 - It is used for typecast.
- ❖ **is**
 - It returns TRUE if the object has specified type.
- ❖ **is!**
 - It returns TRUE if the object has not specified type.

Chapter

01

Type Test Operators

```
void main() {  
var num = 10;  
var name = "Skillologies";  
print(num is int);  
print(name is! String );  
}
```

Chapter

01

Logical Operators

❖ `&&`

❖ `||`

❖ `!`

Live Coding

Expression And Statement

Chapter
01

The Parse()

- ❖ The parse() function converts the numeric string to the number. Consider the following example

```
void main(){  
var a = num.parse("20.56");  
var b = num.parse("15.63");  
var c=a+b;  
print("The sum is = ${c}");  
}
```

hashCode	It returns the hash code of the given number.
isFinite	If the given number is finite, then it returns true.
isInfinite	If the number infinite it returns true.
isNan	If the number is non-negative then it returns true.
isNegative	If the number is negative then it returns true.
sign	It returns -1, 0, or 1 depending upon the sign of the given number.
isEven	If the given number is an even then it returns true.
isOdd	If the given number is odd then it returns true.

<code>abs()</code>	It gives the absolute value of the given number.
<code>ceil()</code>	It gives the ceiling value of the given number.
<code>floor()</code>	It gives the floor value of the given number.
<code>compareTo()</code>	It compares the value with other number.
<code>remainder()</code>	It gives the truncated remainder after dividing the two numbers.
<code>round()</code>	It returns the round of the number.
<code>toDouble()</code>	It gives the double equivalent representation of the number.
<code>toInt()</code>	Returns the integer equivalent representation of the number.
<code>toString()</code>	Returns the String equivalent representation of the number
<code>truncate()</code>	Returns the integer after discarding fraction digits.

Live Coding

What is Next ?

<https://zapp.run/>



Thank You