1. [Dart programming Language] - Lecture [1]

[Variable Attributes and behaviors]

1. Introduction

Variables are not just a group of names with values, the contain much more that you think. When you define a variable you also receive a group of attributes and behaviors that are dependent on the ***Data type*** we chose. These attributes and behaviors can greatly reduce the amount of work we have to do ourselves.

In this lecture will study from where does these attributes and behaviors come from, what are they, how to use them, and take a look at different examples.

objectives:

1. Understand the basics of variable attributes and behaviors.
2. Be able to utilize variable attributes and behaviors.
3. Having an insight when to use these attributes and behaviors.

2. Context

Technical Background

One of the most important skills a programmer can have is realizing when the appropriate attribute or behavior is required to be used. This skill will greatly reduce the amount the code we need to write, makes the code more readable and increase the productivity of our coding sessions.

Relevance to Course

In the end of this course we will realize that our code is mostly compromised of variables and their attributes and behaviors used in different situations and circumstances.

Key Terms and Definitions

1. Attribute: a ***variable*** that is tied to a data type, and it describes a property of the variable
2. Behavior: a set of instructions that are executed on the variable.

3. Main Content

1. Attributes:

Each data type defines a set of attributes (and behaviors) for the variable that uses this data type. And we can access these attributes and properties using the dot operator (.), this operator allows us to access all attributes (and behaviors) that are available for this variable. For example:

int myNumber = 18;

this variable (myNumber) has a value of 18 and all the attributes and behaviors that are given to it by the keyword int, let's take a look at some of them:

Example with int myNumber = 18;	Attribute	Description
myNumber.isEven => true	isEven	Returns `true` if the integer is even.
myNumber.isOdd => false	isOdd	Returns `true` if the integer is odd.
myNumber.sign => 1	Sign	Returns `1` for positive, `-1` for negative, `0` for zero.
myNumber.isNegative => false	isNegative	Returns `true` if the integer is negative.

From the previous table we can easily see that attributes give us direct information about our variable without the need to calculate it ourselves.

Here is a list of the most common attributes for different types of data types, ***although memorizing all attributes is not a good idea, but the more a programmer knows about different type of attributes the better he is.***

String name = "Ahmed"

Example	Attribute	Description
name.length => 5	Length	Returns the number of characters in the string.

name.isEmpty => false	is Empty	Checks if the string is empty.
name.isNotEmpty => true	isNotEmpty	Checks if the string is not empty.

2. Behaviors:

Behaviors as we said previously is a set of instructions that are executed on the variable, these instructions (usually) change the value of the variable into a new value. Behaviors are accessed the same way as attributes using the dot operator. Behaviors achieve more work than attributes. For example, a behavior on a variable of type string can change the value of the variable to be all uppercase or lowercase, another example is a behavior that change the value of an integer value to a string value. Let's take a look at some of these examples:

String name = "Ahmed Ali";

Example	Attribute	Description
name. toUpperCase() => "AHMED ALI"	toUpperCase()	Returns the all of the string in uppercase.
name. toLowerCase ()=> "ahmed ali"	toLowerCase()	Returns the all of the string in lowercase.
name.trim() => "AhmedAli"	trim()	Returns the string without any white spaces.

3. Behaviors inputs

A single behavior can be applied in different ways on the same variable, they do not always return the same value. This is one of the most fundamental concepts of behaviors, ***they can change their results,*** some behaviors can change and some of

them cannot, we can determine if a behavior can change its behavior if it has ***input*** or not. If it does not have input, then this behavior will always give the same result, if it has input, then this behavior will give different results depending on the input we gave it. Furthermore, the more inputs a behavior has the more we can change it.

Let's take a look at the structure of a behavior:

variableName.behaviorName(input);

For example:

String name = "Ahmed";
name.toUpperCase(); // this will always return the string AHMED

Since the behavior toUpperCase() has no input it will always have the same result and we cannot control it at any way. Now let's take a look at a behavior that has an input.

String name = "Ahmed";
name.contains("h");
// this behavior will check if the letter h is in the string name or not since Ahmed has the letter h in it, this will return true
name.contains("z");
// this will return false since the letter z is not in Ahmed.

We can clearly see that when we change the input of the behavior the result changes.

String message = "your password is 1234abcd";
message.replaceAll("1234abcd" , "***");
// this will result in "your password is ***
we can see that this behavior has two inputs the first one finds the text we want to replace and the other one is the new replaced text.

String name = "Ali";
name.indexOf("i"); // this will return the index (number of the letter in the string) of the letter. In this case it will return 2. Notice that we start counting from 0;

Now let's go deeper, behaviors have ***optional inputs*** and we can use them or ignore them depending on our needs, the previous behavior has an optional input that determines the place we start counting from.

name.indexOf(“i” , 1); // notice we added a new input in this case, the number 1 indicated that this behavior will start counting from the second letter in the string which will return 1 since the letter i is the directly after the letter l in the name Ali;

Mastering behaviors and their inputs will move your ability in programming into the next level.

4. Practical Examples

Example 1: [Extract first name]

Use the substring and indexOf behaviors to extract the first name from this variable “Ahmed Ali”

Example 2: [Extracting the Domain from an Email Address]

Using the subString behavior extract the domain name from this email ahmed@gmail.com , result should be gmail.com

Example 3: [what will be the result]

```
String sentence = "Dart is fun to learn";
int firstSpaceIndex = sentence.indexOf(' ');
int secondSpaceIndex = sentence.indexOf(' ', firstSpaceIndex + 1);
```

5. Homework

Show the result of all the practice examples.

You can read more about all the behaviors and attributes of strings and others in this link:

1. https://api.dart.dev/stable/3.5.1/dart-core/int-class.html
2. https://api.dart.dev/stable/3.5.1/dart-core/String-class.html

Due date: 29/8/2024