An Introduction

In this lesson, we will further explore strings in Scala.

We'll cover the following

In one of the previous chapters, we touched upon string literals. In this chapter, we are going to continue our discussion on strings and go into much more detail. In the second half of this chapter, we will take a **problem/solution** approach where you will be introduced to a problem and through its solution, you will learn of different methods that can be applied on strings in Scala.

Declaring a String

While we have already learned how to declare a string, let's have a quick recap before moving forward.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
// Explicitly mentioning the data type
val myFirstString: String = "A Quick Recap"
println(myFirstString) // Driver Code

// Using type inference
val mySecondString = "Type Inference is Cool!"
println(mySecondString) // Driver Code
```

Using Java's String Methods

If you're coming from another language, it might be beneficial for you to know that strings in Scala are very similar to strings in other programming languages. In actuality, Scala's `String` is built on Java's `String` with added unique features. Let's look at some examples below.

> Note: Java is not a requirement to understand the coming examples. The methods used will be exactly the same whether you know Java or not.

Length of a String

To find the length of a string we use the `length()` method.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val stringVar = "What is the length of this string?"

println(stringVar.length())
```

Feel free to play around with the above code and change the value of `stringVar` to different strings to compare the outputs.

String Concatenation

To concatenate two strings means to join them together. Concatenation of two or more strings is done using the `+` operator.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
//Concatenating without variables
println("Hello " + "World")

//Concatenating using variables
val string1 = "Hello "
val string2 = "World"
print(string1 + string2)
```

We can concatenate other data types with strings as well, such as `Int` or `Float`.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
//Concatenating a string with an integer
println("This is the integer " + 5)

//Concatenating a string with a floating point
val aFloat = 1.5F
println("This is the floating point " + aFloat)
```

These are just a few of the Java `String` methods you can use on Scala strings. We'll play around with strings throughout this chapter and learn how to modify them in different ways. Let's start with Scala's unique and infamous feature: String Interpolation.

String Interpolation

String interpolation is the ability to create new strings or modify existing ones by embedding them with expressions. Interpolation is Scala's more concise and efficient alternative to string concatenation. However, string interpolation is a lot more complex than simple concatenation.

There are a total of three inbuilt interpolation methods:

1. `s`
2. `f`
3. `raw`

We will discuss each one in detail in the coming lessons.

Let's start off with `s` in the next lesson.