The Multiple Methods of Declaring a String

In this lesson, you will look at how to initialize a string in a single line, as well as how to write a string on multiple lines.

We'll cover the following

Strings on a single line#

The first primitive is the string. A string is made of characters. It can be assigned a single quote or a double quote. A string's content can be a number but will behave as characters if between quotes. Both single and double quotes are accepted. However, the guideline of the TypeScript project uses a double quote. By the way, TypeScript is written in TypeScript! It means the language itself is written with the language, hence following its guidance is a good idea.

```
let w = "Value1";
let x = "this is a string with the value " + w;
let y = 'this is a string with the value ' + w;
let z = `this is a string ${w}`;
console.log(w,x,y,z)
```

There is also the possibility of using the backquote for string interpolation. String interpolation allows templating a string to avoid concatenating many strings with the plus sign. It helps to have a cleaner string built. The syntax starts with a backquote and id followed by all desired strings as usual with a single or double-quotes. The value of a variable is embedded in the string by using the dollar sign followed by a curly brace. The insertion of the name of the variable must follow. The use of an ending curly brace is required, and from there, you can write any other characters and close the string by using the second backquote. This is demonstrated in line 2 of the code playground below by way of `${word}`. The placeholder is often a variable, but you can insert any TypeScript expression.

```
let word = "yes!"
let stringInterpolation = `this is a string ${word}`;
console.log(stringInterpolation)
```

Strings on multiple lines

TypeScript can write strings on multiple lines without requiring the use of the backslash *n* `\n` that is required by JavaScript.

```
let multiline1 = "Line1\n" + "Line2\n" + "Line3\n";

let multiline2 = `Line1
Line2
Line3`;

console.log(multiline1);
console.log(multiline2);
```

This shorthand removes the boilerplate of having to bring many characters by allowing you to hit enter to switch lines. Similar to string interpolation, it requires the use of backquotes. Inside the backquote, every changed line will be considered as if you were explicitly using the `\n`. The result of the string interpolation produces multiple strings with backslash *n*.

String interpolation for formatting#

String interpolation has the advantage of providing a placeholder to execute the TypeScript code. The main use case is when you have a string with dynamic areas where values can change.

```
let numberBook = 10;
let storeName = "Amazon";
let title = `My favorite ${numberBook} books on ${storeName}`
console.log(title);
```

But as mentioned, it executes code. Albeit mostly used for variables, you can have TypeScript code executed like below:

```
function getPrice(): number{
    return 100;
}
let description = `The book is about ${2+5} chapters and cost ${getPrice()}$.`;
console.log(description)
```

So far, you have used the implicit way to define a string. However, defining a string explicitly is a matter of using the semicolon and the `string` keyword.

```
let myString: string ="I am a string";
let myString2: string = `so am I`;
```

Back

Switch Scope

Next

What is a Number in TypeScript?

✓ Completed

Report an Issue