Strings and Template Literals

Introduction to Strings and Template Literals

Strings are a very important part of programming. In this lecture, we will learn about an easy way to build strings using something called template literals.

Let's start by creating some new variables about a person, in this case, about me, so that we can then concatenate them into one big string.

For example, we can define variables for the first name, last name, job, and birth year.

Now, let's use these variables to build a string, something like: "I'm Jonas, a 40 or 30 year old teacher." We will use the birth year to calculate the age.

As we already learned in previous lectures, we can use the plus sign to concatenate strings. Let's start by doing that.

When writing strings that include single quotes, such as "I'm" or "isn't", we need to use double quotes to define the string. Otherwise, the single quote will terminate the string prematurely.

For example, we write: "I'm" followed by a space, then concatenate with the first name variable.

It is important that each string starts and ends with the same type of quotation mark, either single or double quotes. Both create strings in the same way.

Continuing, we add a comma and a space, then the age and job description. To calculate the age, we subtract the birth year from the current year.

When performing this calculation inside a string concatenation, we need to put the operation inside parentheses so that the calculation happens before concatenation.

You might wonder how concatenation works when mixing numbers and strings. This is due to type coercion, which means JavaScript converts the number to a string so it can join it with the other strings.

Finally, we add an exclamation mark to finish the sentence and log the result to the console.

However, managing spaces and concatenation with plus signs can be cumbersome and error-prone, especially for complex strings.

Template Literals: A Better Way to Build Strings

Starting with ES6, we have a much better tool for building strings called template literals.

With template literals, we can write a string in a more natural way and insert variables directly into the string, which will be replaced automatically.

A template literal assembles multiple pieces into one final string. To write a template literal, we use backticks (`) instead of single or double quotes.

On an English keyboard, the backtick is located above the tab key. Using backticks tells JavaScript that we are writing a template string.

Inside the template literal, we insert variables using the syntax **${variableName}**. For example, **${firstName}** will be replaced with the value of the variable **firstName**.

We can also include expressions inside the curly braces, such as calculations. For example, **${year - birthYear}** will compute the age dynamically.

This approach eliminates the need for plus signs and manual space management, making the string much easier to write and read.

If you encounter bugs, such as using a variable name incorrectly, reading the error message carefully will help you identify and fix the issue.

Template literals are one of the most used ES6 features because they are amazing and useful in many situations.

Using Backticks for All Strings

We can also use backticks to write strings that do not include any placeholders. This means backticks can be used for any string, not just template literals.

Many developers prefer to use backticks for all strings because it removes the need to decide between single or double quotes and makes inserting variables easier when needed.

Although this is a matter of personal preference, it is a valid and common practice.

Multiline Strings with Template Literals

Another great use case of template literals is creating multiline strings.

Before template literals, writing multiline strings was cumbersome. You had to use the newline character **\n** and concatenate strings across lines.

For example, you would write a string with **\n** to indicate new lines and use backslashes to continue the string on the next line.

This approach worked but was not very clean or readable.

With template literals, you simply hit return to create a new line inside the backticks, and the string will include the line breaks automatically.

This feature is immensely useful, especially when building HTML from JavaScript, as it allows creating multiline HTML elements dynamically in a clean and readable way.

Whenever you need a multiline string, make sure to use template literals because they are much cleaner.

Conclusion

In this lecture, we learned how template literals work in JavaScript, particularly in ES6. They provide a much easier and more readable way to build strings, embed variables and expressions, and create multiline strings.

In the next lecture, we will make our code more fun and take it to the next level by introducing decision-making.

Key Takeaways

- Strings are fundamental in programming, and concatenation can be done using the plus sign.
- Template literals, introduced in ES6, allow embedding variables and expressions directly within strings using backticks and **${}** syntax.
- Template literals simplify string construction, including handling spaces and multiline strings, making code more readable and maintainable.

- Backticks can be used for all strings, offering flexibility and reducing the need to switch between single and double quotes.