Theory: Template literals

9 minutes 0 / 5 problems solved

If you ever tried to add variables to strings of text, you probably noticed that the resulting program looks difficult to read.

Previously, we only used single quotes `''` or double quotes `""` to create a string. This led to various restrictions and inconveniences, so it's time we learn to manage strings without unnecessary complications.

§1. Using template literals

Suppose we need to display a message about today's temperature in some city:

```
Now, the temperature in ... is ... degrees Celsius.
```

In place of `...` we need to write certain values.

We can do it this way:

```
let city = "Paris";
let temp = "24";


console.log("Now, the temperature in " + city + " is " + temp + " degrees Celsius.
");
```

Here we declare two variables, for example, `city` and `temp`, and further build the result string using a sequence of concatenations. Of course, this code works correctly, but it is long, and we have to keep a close eye on gaps so that we don't accidentally get solid text in the output.

JavaScript provides a more convenient way to manage strings: **template literals**. To put a value of a variable to a string, write the dollar sign `$` before the variable's name, put it in curly brackets `{}`; enclose the string in reverse quotes `` ` ` `` instead of double or single:

```
let city = "Paris";
let temp = "24";


console.log(`Now, the temperature in ${city} is ${temp} degrees Celsius.`);
```

This code is more readable than that with concatenations. You can run these scripts and see for yourself that they display the same message:

```
Now, the temperature in Paris is 24 degrees Celsius.
```

§2. Multi-line strings

Symbols of a new line are part of the template literals. Now you can quickly and effortlessly write a multi-line program. For example:

```
console.log(`String text line 1
String text line 2`);
```

The output of this code will be as follows:

```
String text line 1
String text line 2
```

This is quite handy, because now there is no need to duplicate functions or puzzle over where to put line break characters (`\n`).

Current topic:

Template literals

Topic depends on:

× Variables

× Arithmetic operators

Topic is required for:

ForEach method

JavaScript Code Style

Table of contents:

↑ Template literals

§1. Using template literals

§2. Multi-line strings

§3. Syntactic sugar

Feedback & Comments

§3. Syntactic sugar

Thanks to the template literals, it became convenient not only to insert values of variables, but also to insert whole expressions. Take a look at this example:

```
let a = 1;
let b = 2;

console.log(`The sum of numbers ${a} and ${b} is not equal to ${ 10 * a + b }.`);
// The sum of numbers 1 and 2 is not equal to 12.
```

Without the template literals, this code would look like this:

```
let a = 1;
let b = 2;

console.log("The sum of numbers " + a + " and " + b + " is not equal to "+ (10 * a
 + b) + ".");
// The sum of numbers 1 and 2 is not equal to 12.
```

Template literals are a truly useful innovation in ES6. It allows for flexible line management, easy creation of multi-line programs and easy insertion of expressions into strings.

Report a typo

112 users liked this theory. **2** didn't like it. **What about you?**

Start practicing

Comments (1) Hints (0) Useful links (0) Show discussion