Day 5: Template Literals


by AvmnuSng

JavaScript: Template Literals

Template Literals

Table Of Contents

Template literals (formerly known as *template strings*) are string literals that allow for embedded expressions. We typically use them to express strings spanning multiple lines or for string interpolation, which essentially allows us to create a template with one or more placeholders for inserting variable text at a later time.

While traditional strings are wrapped in single or double quotes, template literals are wrapped in backtick (`` ` ``) characters. A template literal can contain placeholders, which are preceded by a dollar sign (`$`) and wrapped in curly braces (`{}`). For example, in the template literal `` `${expression}` ``, the ***expression*** text between the placeholders is passed to a function. The default function simply concatenates the template literal's parts into a single string.

Any time we see an expression preceding a template literal, we call the expression a *tag* and the template string a *tagged template literal*. In these instances, we call the tag expression (typically a function) with the processed template literal, which we can then manipulate before outputting the final string.

Multi-line Strings

\- EXAMPLE

Print a Multi-Line String Using Normal Strings

```
1 console.log("first line\n" + "second line");
2 console.log("first line" + "\nsecond line");
3 console.log("first line\nsecond line");
```

Output

Run

Print a Multi-Line String Using Template Literals

```
1 console.log(`first line
2 second line`);
```

Output

Run

Regardless of which approach we choose, the output is the same.

Expression Interpolation

- EXAMPLE

Print a Line Using Normal Strings

```
const a = 2;
const b = 3;

console.log(
    'The sum of a and b is ' + (a + b) + '.\n'
    + 'The product of a and b is ' + (a * b) + '.'
);
```

Output

Run

Print a Line Using Template Literals

```
const a = 2;
const b = 3;

console.log(`The sum of a and b is ${a + b}.
The product of a and b is ${a * b}.`);
```

Output

Run

Tagged Template Literals

Tagged template literals allow us to use a function to modify the output of a template literal. In this construct:

1. The first argument contains an array of string literals.
2. The subsequently processed arguments are the values of the substitution expressions.

After processing all the arguments, the function returns the manipulated string.

- EXAMPLE

```
var a = 5;
var b = 10;

function foo(strings, ...values) {
    console.log("." + strings[0] + ".");
    console.log("." + strings[1] + ".");
    console.log("." + strings[2] + ".");
    console.log("." + strings[3] + ".");
    console.log(values[0]);
    console.log(values[1]);
    console.log(values[2]);
}

foo`Sum ${a + b}
Product ${a * b}
Division ${b / a}`;
```

Output

Run

Now we can see that, the total number of string literals is one more than the number of total cooked expressions. The first string literal is the string before the first cooked

expression, the last string literal is the string after the last cooked expression and other literals are in between the cooked expressions.

We can also return from tagged templates:

- EXAMPLE

```
var a = 5;
var b = 10;

function foo(strings, ...values) {
    let a = values[0];
    let b = values[1];

    return `Sum ${a + b}
Product ${a * b}
Division ${b / a}`;
}

console.log(foo`Num1 ${a + 10}
Num2 ${b * 2}
Num3 ${b / a}`);
```

Output

Run