Lesson Tuesday

Intermediate JavaScript (/intermediate-javascript) / Test-Driven Development and Environments with JavaScript (/intermediate-javascript/test-driven-development-and-environments-with-javascript) / ES6 Template Literals

Text

Template literals are a spoonful of syntactic sugar and another popular feature in ES6. They're especially useful when concatenating strings or inserting variables into strings.

A lot of other languages have this same feature, too. For example, template literals are called string interpolation in both Ruby and C#. It's also a lot easier to use than concatenation, especially with longer strings.

Template Literals

Here's an example of how we'd insert a variable into a string prior to ES6. (In the following example, ignore the fact that we wouldn't use `let` prior to ES6.)

```
let language = "JavaScript";
let adjective = "fun";

let concatenatedString = language + " is " + adjective + "
to learn!";
```

String concatenation is useful but it quickly becomes painful. We can make this process relatively painless using template literals. Here's an example using the same variables from above:

```
let stringLiteral = `${language} is ${adjective} to learn!
`;
```

There are two syntax rules to remember with template literals:

1. First, we need to surround the string with the backtick symbol instead of quotes.
2. Second, any variable or expression needs to be put inside of `${ }`.

Using Template Literals for Expressions

Note that template literals `${ }` can contain any expression, not just a variable. For instance, we could do this:

```
let length = 2;
let width = 3;
let rectangleArea = `The area of a rectangle with length
${length} and width ${width} is ${length * width}.`;
```

In the example above, we're calculating `length * width` inside the template literal. We can compute any JS code inside a template literal as long as it's inside of `${}`. Keep in mind that while you *can*

do this, you probably *shouldn't*, since it could quickly make your code less readable.

Creating Multi-Line Strings

One last thing. Backticks can be used to create multi-line strings. It's unlikely you'll need this functionality much, but it's still good to know. Here's an example to try out in the DevTools console:

```
> const multiLineString = `This is not a good
haiku about JavaScript code:
template literals.`;
> console.log(multiLineString);
This is not a good
haiku about JavaScript code:
template literals.
```

If we don't print `multiLineString` in a `console.log()`, we'll see our line breaks are preserved by the new line character `\n`:

```
'This is not a good\nhaiku about JavaScript code:\ntemplate
literals.'
```

To see more examples, visit the MDN documentation on template literals (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals).

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/es6-arrow-notation)
Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/working-with-multiple-business-logic-files)

(http://www.epicodus.com)