Lesson | Wednesday

Introduction to Programming (/introduction-to-programming) / Arrays and Looping (/introduction-to-programming/arrays-and-looping) / Further Exploration: Regular Expressions with Text Analyzer

Text | Cheat sheet

In the last lesson, we covered regular expressions in general. Now let's apply them to the text analyzer application we've built. By doing so, we can see just how powerful regular expressions are.

Remember, this is lesson is a *further exploration* and you are not required to go through it.

Updating Text Analyzer to Use Regular Expressions

We are going to be exploring some new syntax in this lesson. This is a preview of the next section when we'll learn about JavaScript objects. We can create a new regular expression object like this:

```
const word = "red";
const regex = new RegExp(word, "gi");
```

The `new` keyword creates a new regular expression object. We are not going to discuss this syntax further until the next section. For now, just be aware that this is the best way to pass a variable into a regular expression.

When we create a new `RegExp` object, it takes two arguments. The first is the regular expression itself. This can be a variable but it could also be a string as well. The second argument is any flags we want to pass into the regular expression. `"gi"` means that we want the regular expression to be applied to the entire string *and* that we want it to be case-insensitive.

This means we can do the following in the DevTools console:

```
> const text = "RED red red! Green GREEN green.";
> const word = "red";
> const regex = new RegExp(word, "gi");
> text.match(regex);
["RED", "red", "red"]
```

Can you see where this is going? We can make some of our Text Analyzer functions much more concise with regular expressions.

Let's take a look at `numberOfOccurrencesInText()`:

```
// Without regex
function numberOfOccurrencesInText(word, text) {
  if (isEmpty(word)) {
    return 0;
  }
  const textArray = text.split(" ");
  let wordCount = 0;
  textArray.forEach(function(element) {
    if (element.toLowerCase().includes(word.toLowerCase()))
{
      wordCount++;
    }
  });
  return wordCount;
}

// With regex
function numberOfOccurrencesInText(word, text) {
  if (isEmpty(word)) {
    return 0;
  }
  const regex = new RegExp(word, "gi");
  return text.match(regex).length;
}
```

With a regular expression, our function is half as many lines. There is no need to loop at all. We create a regular expression object and store it in `regex`. Then we simply need to call `String.prototype.match()` with `regex` and return the length of the resulting array. Try it out in the DevTools console!

To sum up, regular expressions can be fun but they can also be hard to figure out and easy to mess up. Regardless, they are a very powerful tool to have in your developer toolbox.

Previous (/introduction-to-programming/arrays-and-looping/further-exploration-introduction-to-regular-expressions)
Next (/introduction-to-programming/arrays-and-looping/array-mapping)

Lesson 42 of 50
Last updated February 28, 2023

disable dark mode

(http://www.epicodus.com)