Backreferences

Summary: in this tutorial, you'll learn about JavaScript regex backreferences and how to apply them effectively.

Introduction to JavaScript regex backreferences

Backreferences allow you to reference the capturing groups in the regular expressions. Technically speaking, backreferences are like variables in regular expressions.

Here's the syntax of a backreference:

```
\N
```

In this syntax, `N` is an integer such as 1, 2, and 3 that represents the corresponding capturing group number.

Suppose you have a string with the duplicate word `JavaScript` like this:

```
const s = 'JavaScript JavaScript is awesome';
```

And you want to remove the duplicate word (`JavaScript`) so that the result string will be:

```
'JavaScript is awesome'
```

To do so, you can use a backreference in the regular expression.

First, match a word:

```
/\w+\s+/
```

Second, create a capturing group that captures the word:

```
/(\w+)\s+/
```

Third, use a backreference to reference the first capturing group:

```
/(\w+)\s+\1/
```

In this pattern, the `\1` is a backreference that references the (`\w+`) capturing group.

Finally, replace the entire match with the first capturing group using the `String.replace()` method:

```
const s = 'JavaScript JavaScript is cool';
const pattern = /(\w+)\s+\1/;

const result = s.replace(pattern, '$1');

console.log(result);
```

Output:

```
JavaScript is cool
```

JavaScript regex backreference examples

The following examples show some practical applications of backreferences.

1) Using backreferences to get text inside quotes

To get the text inside the double quotes like this:

```
"JavaScript Regex Backreferences"
```

Or single quotes:

```
'JavaScript Regex Backreferences'
```

But not mixed of single and double-quotes:

```
'not match"
```

To do so, you might come up with the following regular expression:

```
/[\'"](.*?)[\'"]/
```

However, this regular expression also matches the text that starts with a single quote (') and ends with a double quote (") or vice versa. For example:

```
const message = `"JavaScript's cool". They said`;
const pattern = /[\'"].*?[\'"]/;

const match = message.match(pattern);

console.log(match[0]);
```

It returns the `"JavaScript'` not `"JavaScript's cool"` .

To resolve it, you can use a backreference in the regular expression:

```
/([\'"]).*?\1/
```

The backreference `\1` references the first capturing group. If the subgroup starts with a single quote, the `\1` matches the single quote. And if the subgroup starts with double quotes, the `\1` matches double-quotes.

For example:

```
const message = `"JavaScript's cool". They said`;
const pattern = /([\'"]).*?\1/;
```

```
const match = message.match(pattern);

console.log(match[0]);
```

Output:

```
"JavaScript's cool"
```

2) Using backreferences to find word that has at least one consecutive repeated character

The following example shows how to use a backreference to find the word that has at least one consecutive repeated character e.g., `apple` (the letter `p` is repeated):

```
const words = ['apple', 'orange', 'strawberry'];
const pattern = /\b\w*(\w)\1\w*\b/;

for (const word of words) {
  const result = word.match(pattern);
  if (result) {
    console.log(result[0], '->', result[1]);
  }
}
```

Output:

```
apple -> p
strawberry -> r
```

Summary

- Use backreferences to reference the capturing groups in a regular expression.