String.prototype.replace()

Summary: in this tutorial, you'll learn how to use the JavaScript String `replace()` method to return a new string with one or more matches replaced by a new string.

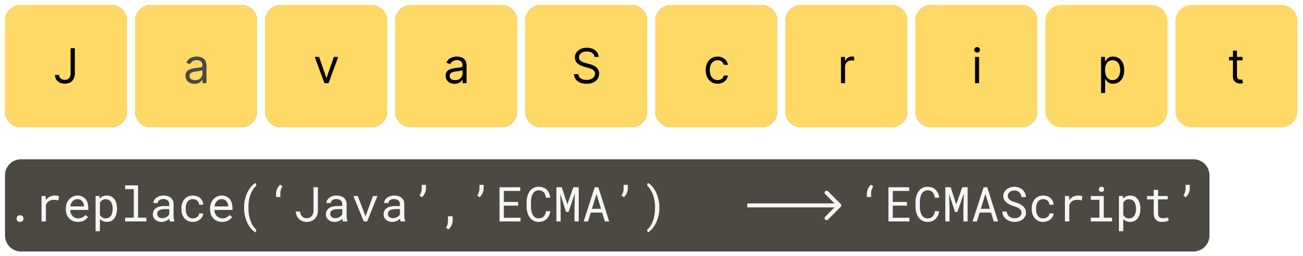
Introduction to the JavaScript String replace() method

The replace() method returns a new string with the first occurrence of a substring by a new one.

Here's the syntax of the replace() method:

```
let newStr = str.replace(substr, newSubstr);
```

Note that the `replace()` method doesn't change the original string but returns a new string.



To return a new string with all occurrences of a substring replaced by a new one, you use the replaceAll() method.

JavaScript String replace() method examples

Let's take some examples of using the JavaScript String `replace()` method.

1) Basic JavaScript String replace() method example

The following example uses the `replace()` method to return a new string with the `'JS'` replaced by `'JavaScript'` in the string `"JS will, JS will rock you!"` :

```
let str = "JS will, JS will rock you!";
let newStr = str.replace("JS", "JavaScript");

console.log({ newStr });
```

Output:

```
{ newStr: 'JavaScript will, JS will rock you!' }
```

The output indicates that the `replace()` method replaces only the first occurrence of the substring `JS` with the new substring `JavaScript`.

To return a new string with all occurrences of a substring replaced by a new substring, you can use a newer method called `replaceAll()`:

```
let str = "JS will, JS will rock you!";
let newStr = str.replaceAll("JS", "JavaScript");

console.log({ newStr });
```

Output:

```
{ newStr: 'JavaScript will, JavaScript will rock you!' }
```

Alternatively, you can use a regular expression.

Using regular expressions

The `replace()` method fully supports regular expressions:

```
let newStr = str.replace(regexp, newSubstr);
```

In this syntax, the `replace()` method finds all matches in the `str` and returns a new string with all matches replaced by the `newSubstr`.

The following example uses the global flag (`g`) to replace all occurrences of the `JS` in the `str` by the `JavaScript` :

```
let str = "JS will, JS will rock you!";
let newStr = str.replace(/JS/g, "JavaScript");

console.log({ newStr });
```

Output:

```
JavaScript will, JavaScript will rock you!
```

If you want to ignore cases for searching and replacement, you can use the ignore flag (`i`) in the regular expression like this:

```
let str = "JS will, Js will rock you!";
let newStr = str.replace(/JS/gi, "JavaScript");

console.log({ newStr });
```

Output:

```
{ newStr: 'JavaScript will, JavaScript will rock you!' }
```

Note that the `replaceAll()` method performs a case-sensitive search for matching the string:

```
let str = 'JS will, Js will rock you!';
let newStr = str.replaceAll('JS', 'JavaScript');

console.log({ newStr });
```

Output:

```
{ newStr: 'JavaScript will, Js will rock you!' }
```

Using a replacement function

Instead of passing a `newSubstr` to the second parameter of the `replace()` method, you can pass a *replacement function* as follows:

```
let newStr = str.replace(substr | regexp, replacer);
```

In this syntax, the `replace()` method calls the `replacer` function after the match has been performed and then uses the result of the function as the replacement string.

If you use the global flag (`g`) in the regular expression, the `replace()` method will invoke the `replacer` function for every match.

For example, if there are three matches, the `replace()` method will invoke the `replacer()` function three times.

The `replacer()` function has the following syntax:

```
function replacer(match, p1, p2, ..., offset, string);
```

In this syntax:

- `match` is the matched substring.
- `p1` , `p2` , ... `pn` are the nth string found by a parenthesized capture group provided by the regular expression.
- `offset` is the offset of the matched substring within the string being searched.
- `string` is the whole string being examined.

The following example uses the `replace()` function to change the substrings `apples` and `bananas` to uppercase. It passes a replacer function into the `replace()` function:

```
let str = "I like to eat, eat, eat apples and bananas";
let re = /apples|bananas/gi;

let newStr = str.replace(re, (match) => {
    console.log({match});
    return match.toUpperCase();
});

console.log(newStr);
```

Output:

```
{match: "apples"}
{match: "bananas"}
I like to eat, eat, eat APPLES and BANANAS
```

Summary

- Use the `replace()` method to return a new string with a substring replaced by a new one.
- Use the `replaceAll()` to replace all occurrences of a string with a new substring.
- Use a regular expression with the global flag (`g`) to replace all occurrences of a substring with a new one.