6th September 2019

Backreferences in pattern: \N and \k<name>

We can use the contents of capturing groups `(...)` not only in the result or in the replacement string, but also in the pattern itself.

Backreference by number: \N

A group can be referenced in the pattern using `\N`, where `N` is the group number.

To make clear why that's helpful, let's consider a task.

We need to find quoted strings: either single-quoted `'...'` or a double-quoted `"..."` – both variants should match.

How to find them?

We can put both kinds of quotes in the square brackets: `['"](.*?)['"]`, but it would find strings with mixed quotes, like `"...'` and `'..."`. That would lead to incorrect matches when one quote appears inside other ones, like in the string `"She's the one!"`:

```
let str = `He said: "She's the one!".`;

let regexp = /['"](.*?)['"]/g;

// The result is not what we'd like to have
alert( str.match(regexp) ); // "She'
```

As we can see, the pattern found an opening quote `"`, then the text is consumed till the other quote `'`, that closes the match.

To make sure that the pattern looks for the closing quote exactly the same as the opening one, we can wrap it into a capturing group and backreference it: `(['"])(.*?)\1`.

Here's the correct code:

```
let str = `He said: "She's the one!".`;

let regexp = /(['"])(.*?)\1/g;

alert( str.match(regexp) ); // "She's the one!"
```

Now it works! The regular expression engine finds the first quote `(['"])` and memorizes its content. That's the first capturing group.

Further in the pattern `\1` means "find the same text as in the first group", exactly the same quote in our case.

Similar to that, `\2` would mean the contents of the second group, `\3` – the 3rd group, and so on.

Please note:

If we use `?:` in the group, then we can't reference it. Groups that are excluded from capturing `(?:...)` are not memorized by the engine.

Don't mess up: in the pattern `\1`, in the replacement: `$1`

In the replacement string we use a dollar sign: `$1`, while in the pattern – a backslash `\1`.

Backreference by name: \k<name>

If a regexp has many parentheses, it's convenient to give them names.

To reference a named group we can use `\k<имя>`.

In the example below the group with quotes is named `?<quote>`, so the backreference is `\k<quote>`:

```
let str = `He said: "She's the one!".`;

let regexp = /(?<quote>['"])(.*?)\k<quote>/g;

alert( str.match(regexp) ); // "She's the one!"
```

Previous lesson Next lesson

Share Tutorial map

Comments

- If you have suggestions what to improve - please submit a GitHub issue or a pull request instead of commenting.
- If you can't understand something in the article – please elaborate.
- To insert a few words of code, use the `<code>` tag, for several lines – use `<pre>`, for more than 10 lines – use a sandbox (plnkr, JSBin, codepen…)