6th September 2019

Escaping, special characters

As we've seen, a backslash `\` is used to denote character classes, e.g. `\d`. So it's a special character in regexps (just like in regular strings).

There are other special characters as well, that have special meaning in a regexp. They are used to do more powerful searches. Here's a full list of them: `[ \ ^ $ . | ? * + ( )`.

Don't try to remember the list – soon we'll deal with each of them separately and you'll know them by heart automatically.

Escaping

Let's say we want to find literally a dot. Not "any character", but just a dot.

To use a special character as a regular one, prepend it with a backslash: `\.`.

That's also called "escaping a character".

For example:

```
alert( "Chapter 5.1".match(/\d\.\d/) ); // 5.1 (match!)
alert( "Chapter 511".match(/\d\.\d/) ); // null (looking for a real dot \.)
```

Parentheses are also special characters, so if we want them, we should use `\(`. The example below looks for a string `"g()"`:

```
alert( "function g()".match(/g\(\)/) ); // "g()"
```

If we're looking for a backslash `\`, it's a special character in both regular strings and regexps, so we should double it.

```
alert( "1\\2".match(/\\/) ); // '\'
```

A slash

A slash symbol `'/'` is not a special character, but in JavaScript it is used to open and close the regexp: `/...pattern.../`, so we should escape it too.

Here's what a search for a slash `'/'` looks like:

```
alert( "/".match(/\//) ); // '/'
```

On the other hand, if we're not using `/.../`, but create a regexp using `new RegExp`, then we don't need to escape it:

```
alert( "/".match(new RegExp("/")) ); // finds /
```

new RegExp

If we are creating a regular expression with `new RegExp`, then we don't have to escape `/`, but need to do some other escaping.

For instance, consider this:

```
let regexp = new RegExp("\d\.\d");

alert( "Chapter 5.1".match(regexp) ); // null
```

The similar search in one of previous examples worked with `/\d\.\d/`, but `new RegExp("\d\.\d")` doesn't work, why?

The reason is that backslashes are "consumed" by a string. As we may recall, regular strings have their own special characters, such as `\n`, and a backslash is used for escaping.

Here's how "\d.\d" is preceived:

```
alert("\d\.\d"); // d.d
```

String quotes "consume" backslashes and interpret them on their own, for instance:

- `\n` – becomes a newline character,
- `\u1234` – becomes the Unicode character with such code,
- …And when there's no special meaning: like `\d` or `\z`, then the backslash is simply removed.

So `new RegExp` gets a string without backslashes. That's why the search doesn't work!

To fix it, we need to double backslashes, because string quotes turn `\\` into `\`:

```
let regStr = "\\d\\.\\d";
alert(regStr); // \d\.\d (correct now)

let regexp = new RegExp(regStr);

alert( "Chapter 5.1".match(regexp) ); // 5.1
```

Summary

- To search for special characters `[ \ ^ $ . | ? * + ( )` literally, we need to prepend them with a backslash `\` ("escape them").
- We also need to escape `/` if we're inside `/.../` (but not inside `new RegExp`).
- When passing a string `new RegExp`, we need to double backslashes `\\`, cause string quotes consume one of them.

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…)