29th January 2020

Lookahead and lookbehind

Sometimes we need to find only those matches for a pattern that are followed or preceeded by another pattern.

There's a special syntax for that, called "lookahead" and "lookbehind", together referred to as "lookaround".

For the start, let's find the price from the string like `1 turkey costs 30€`. That is: a number, followed by `€` sign.

Lookahead

The syntax is: `X(?=Y)`, it means "look for `X`, but match only if followed by `Y`". There may be any pattern instead of `X` and `Y`.

For an integer number followed by `€`, the regexp will be `\d+(?=€)`:

```
let str = "1 turkey costs 30€";

alert( str.match(/\d+(?=€)/) ); // 30, the number 1 is ignored, as it's not f
```

Please note: the lookahead is merely a test, the contents of the parentheses `(?=...)` is not included in the result `30`.

When we look for `X(?=Y)`, the regular expression engine finds `X` and then checks if there's `Y` immediately after it. If it's not so, then the potential match is skipped, and the search continues.

More complex tests are possible, e.g. `X(?=Y)(?=Z)` means:

1. Find `X`.
2. Check if `Y` is immediately after `X` (skip if isn't).
3. Check if `Z` is immediately after `Y` (skip if isn't).
4. If both tests passed, then it's the match.

In other words, such pattern means that we're looking for `X` followed by `Y` and `Z` at the same time.

That's only possible if patterns `Y` and `Z` aren't mutually exclusive.

For example, `\d+(?=\s)(?=.*30)` looks for `\d+` only if it's followed by a space, and there's `30` somewhere after it:

```
let str = "1 turkey costs 30€";

alert( str.match(/\d+(?=\s)(?=.*30)/) ); // 1
```

In our string that exactly matches the number `1`.

Negative lookahead

Let’s say that we want a quantity instead, not a price from the same string. That’s a number `\d+`, NOT followed by `€`.

For that, a negative lookahead can be applied.

The syntax is: `X(?!Y)`, it means "search `X`, but only if not followed by `Y`".

```
let str = "2 turkeys cost 60€";

alert( str.match(/\d+(?!€)/) ); // 2 (the price is skipped)
```

Lookbehind

Lookahead allows to add a condition for “what follows”.

Lookbehind is similar, but it looks behind. That is, it allows to match a pattern only if there’s something before it.

The syntax is:

- Positive lookbehind: `(?<=Y)X`, matches `X`, but only if there’s `Y` before it.
- Negative lookbehind: `(?<!Y)X`, matches `X`, but only if there’s no `Y` before it.

For example, let’s change the price to US dollars. The dollar sign is usually before the number, so to look for `$30` we’ll use `(?<=\$)\d+` – an amount preceded by `$`:

```
let str = "1 turkey costs $30";

// the dollar sign is escaped \$
alert( str.match(/(?<=\$)\d+/) ); // 30 (skipped the sole number)
```

And, if we need the quantity – a number, not preceded by `$`, then we can use a negative lookbehind `(?<!\$)\d+`:

```
let str = "2 turkeys cost $60";

alert( str.match(/(?<!\$)\d+/) ); // 2 (skipped the price)
```

Capturing groups

Generally, the contents inside lookaround parentheses does not become a part of the result.

E.g. in the pattern `\d+(?=€)`, the `€` sign doesn’t get captured as a part of the match. That’s natural: we look for a number `\d+`, while `(?=€)` is just a test that it should be followed by `€`.

But in some situations we might want to capture the lookaround expression as well, or a part of it. That's possible. Just wrap that part into additional parentheses.

In the example below the currency sign `(€|kr)` is captured, along with the amount:

```
let str = "1 turkey costs 30€";
let regexp = /\d+(?=(€|kr))/; // extra parentheses around €|kr

alert( str.match(regexp) ); // 30, €
```

And here's the same for lookbehind:

```
let str = "1 turkey costs $30";
let regexp = /(?<=(\$|£))\d+/;

alert( str.match(regexp) ); // 30, $
```

Summary

Lookahead and lookbehind (commonly referred to as "lookaround") are useful when we'd like to match something depending on the context before/after it.

For simple regexps we can do the similar thing manually. That is: match everything, in any context, and then filter by context in the loop.

Remember, `str.match` (without flag `g`) and `str.matchAll` (always) return matches as arrays with `index` property, so we know where exactly in the text it is, and can check the context.

But generally lookaround is more convenient.

Lookaround types:

Pattern	type	matches
`X(?=Y)`	Positive lookahead	`X` if followed by `Y`
`X(?!Y)`	Negative lookahead	`X` if not followed by `Y`
`(?<=Y)X`	Positive lookbehind	`X` if after `Y`
`(?<!Y)X`	Negative lookbehind	`X` if not after `Y`

Tasks

Find non-negative integers

There's a string of integer numbers.

Create a regexp that looks for only non-negative ones (zero is allowed).

An example of use:

```
let regexp = /your regexp/g;

let str = "0 12 -5 123 -18";

alert( str.match(regexp) ); // 0, 12, 123
```

solution

Insert After Head

We have a string with an HTML Document.

Write a regular expression that inserts `<h1>Hello</h1>` immediately after `<body>` tag. The tag may have attributes.

For instance:

```
let regexp = /your regular expression/;

let str = `
<html>
  <body style="height: 200px">
  ...
  </body>
</html>
`;

str = str.replace(regexp, `<h1>Hello</h1>`);
```

After that the value of `str` should be:

```
<html>
  <body style="height: 200px"><h1>Hello</h1>
  ...
  </body>
</html>
```

solution

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…)