How do I get query params in JavaScript?

Written by Seth Corker on Benevolent Bytes

One of the most common things you might want to do in a web app is working with URLs. Fortunately, JavaScript was designed to work in the browser and there are some useful APIs we can rely on. Let's have a look at the URL API.

Building a URL

To get query parameters from the url in JavaScript, we can use the URL API. Let's create a URL from a humble string and see what we have available:

```
const currentUrl = new URL("https://blog.sethcorker.com?online=true")
```

There are two properties on the URL to take note of: - `.search` — the whole params string, e.g. `'?online=true'` - `.searchParams` — a `URLSearchParams` instance We can see `currentUrl.search` gives us what we want, but it means we have to manually go through and parse out the content ourselves. Let's use `currentUrl. searchParams` instead because it allows us to get back our query params in a more structured way.

Knowing the parameter name

In our example, I know I want the value of the `online` parameter so we can call the `get` method to retrieve the value:

```
currentUrl.searchParams.get('online') // returns 'true'
```

Note: we get back a string `'true'` instead of a boolean, URL params are always strings so you'll need to coerce the value once you get it

If you need to get multiple values for the same name, you can use `.getAll`. This allows you to get all associated values, like in this more complex example:

```
const params = new URLSearchParams("?online=true&search=a%20test&id=1&id=2&id=3")
params.getAll('id') // ['1', '2', '3']
```

You can see we have three params called `id` with different values, `getAll` returns an array of these values.

What if we don't know how many parameters there are? Well, we don't have to resort to the raw string, we can iterate though them instead. ### Iterating though params `URLSearchParams` implements the iterator protocol, which is a fancier way of saying we can use a `for...of` loop on it! (You can also use the `.forEach` method if you prefer.

Take the following example:

```
for(const [name, value] of currentUrl.searchParams){
  console.log(`name: ${name} = ${value}`)
}
```

We're looping through each parameter which returns a two element array, the first element is the parameter name, the second is the value.

Now we can successfully retrieve params from a URL, let's look at making changes and adding our own params.

Modifying params

Adding params to a URL is similar to getting params, we just need to provide the name and a value but, we also need to decide if we want to allow multiple params with the same name. Let's have a look at another example, this time we'll have multiple query/search params, some of which will have the same name.

```
const currentUrl = new URL("https://blog.sethcorker.com?online=true&search=a%20test&id=1&id=2&id=3")
```

set method

This method sets the parameter to a value. In this case, the param already exists so we replace it.

```
currentUrl.searchParams.set('online', 'false')
// "https://blog.sethcorker.com?online=false&search=a%20test&id=1&id=2&id=3"
```

If it didn't exist, we'd add the param to the end like this:

```
currentUrl.searchParams.set('new_param', '1')
// "https://blog.sethcorker.com?online=true&search=a+test&id=1&id=2&id=3&new_param=1"
```

And finally, if there are multiple params, `.set` will remove the other params first before adding it:

```
currentUrl.searchParams.set('id', '99')
// "https://blog.sethcorker.com/?online=true&search=a+test&id=99"
```

append method

Append simply appends a param to the search params. If the param already exists, that's fine, it will add another.

```
currentUrl.searchParams.append('id', '11')
currentUrl.searchParams.append('id', '12')
currentUrl.searchParams.append('id', '13')

currentUrl.toString()
// "https://blog.sethcorker.com/?online=true&search=a+test&id=1&id=2&id=3&id=11&id=12&id=13"
```

This can be really useful for representing lists, a list of IDs in our example, which we can easily read back out later on in our app.

Wrapping up

`URL` and `URLSearchParams` are essential for working with URLs in JS. They make manipulation much easier than manually building strings and do some cool things out of the box like ensuring param values are safely encoded to be put in the URL. I hope this has helped! ## Resources - URLSearchParams on MDN - URL API on MDN