The Fetch API Cheatsheet: Nine of the Most Common API Requests

Almost every project needs to communicate with the outside world. If you're working with JavaScript frameworks, you'll most likely use Fetch API to do that.

But when you're working with the API, do you remember the syntax by heart or do you need a little help?

I have written many articles about JavaScript and related things only to find myself later frequently (re)visiting them to refresh my memory or get some sample code that I know "is there somewhere."

In this article, I aim to create another resource like that. I will list the 9 most common Fetch API requests.

I'm sure you've used them all many times. But wouldn't it be nice to avoid going through old projects to find the syntax of that specific request you used half a year ago? :)

Why Use the Fetch API?

Nowadays, we are spoiled by all the services providing nice SDKs that abstract away the actual API requests. We just ask for data using typical language constructs and don't care about the actual data exchange.

But what if there's no SDK for your chosen platform? Or what if you're building both the server and the client? In these cases, you need to handle the requests on your own. This is how you can do it using the Fetch API.

Simple GET request with the Fetch API

```
fetch('{url}')
    .then(response => console.log(response));
```

Simple POST request with the Fetch API

```
fetch('{url}', {
    method: 'post'
})
    .then(response => console.log(response));
```

GET with an authorization token (Bearer) in the Fetch API

```
fetch('{url}', {
    headers: {
        'Authorization': 'Basic {token}'
    }
})
    .then(response => console.log(response));
```

GET with querystring data in the Fetch API

```
fetch('{url}?var1=value1&var2=value2')
    .then(response => console.log(response));
```

GET with CORS in the Fetch API

```
fetch('{url}', {
    mode: 'cors'
})
    .then(response => console.log(response));
```

POST with authorization token and querystring data in the Fetch API

```
fetch('{url}?var1=value1&var2=value2', {
    method: 'post',
    headers: {
        'Authorization': 'Bearer {token}'
    }
```

```
})
    .then(response => console.log(response));
```

POST with form data in the Fetch API

```
let formData = new FormData();
formData.append('field1', 'value1');
formData.append('field2', 'value2');

fetch('{url}', {
    method: 'post',
    body: formData
})
    .then(response => console.log(response));
```

POST with JSON data in the Fetch API

```
fetch('{url}', {
    method: 'post',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        'field1': 'value1',
        'field2': 'value2'
    })
})
    .then(response => console.log(response));
```

POST with JSON data and CORS in the Fetch API

```
fetch('{url}', {
    method: 'post',
    mode: 'cors',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        'field1': 'value1',
        'field2': 'value2'
    })
```

```
})
    .then(response => console.log(response));
```

How to process the results of the Fetch API request

The Fetch API returns a *Promise*. That's why I'm always using `.then()` and a callback function for processing the response:

```
fetch(...).then(response => {
    // process the response
}
```

But you can also await the result if you're in an async function:

```
async function getData(){
    let data = await fetch(...);
    // process the response
}
```

Now let's take a look at how we can extract the data from the response:

How to check the status code of the Fetch API response

When sending POST, PATCH, and PUT requests, we are typically interested in the return status code:

```
fetch(...)
    .then(response => {
        if (response.status == 200){
            // all OK
        } else {
            console.log(response.statusText);
        }
    });
```

How to get a simple value of the Fetch API response

Some API endpoints may send back an identifier of a new database record that was created using your data:

```
var userId;

fetch(...)
    .then(response => response.text())
    .then(id => {
        userId = id;
        console.log(userId)
    });
```

How to convert JSON data of the Fetch API response

But in most cases, you'll receive JSON data in the response body:

```
var dataObj;

fetch(...)
    .then(response => response.json())
    .then(data => {
        dataObj = data;
        console.log(dataObj)
    });
```

Keep in mind that you can access the data only after both Promises are resolved. This is sometimes a bit confusing, so I always prefer to use async methods and await the results:

```
async function getData(){
    var dataObj;

    const response = await fetch(...);
    const data = await response.json();
    dataObj = data;
    console.log(dataObj);
}
```

Conclusion

These samples should have you covered in most situations.

Is there something that I missed, a request you use on a daily basis? Or something else you're struggling with? Let me know on Twitter, and I'll cover it in another article :-)

Oh, and you can get this cheatsheet in a printable form too.