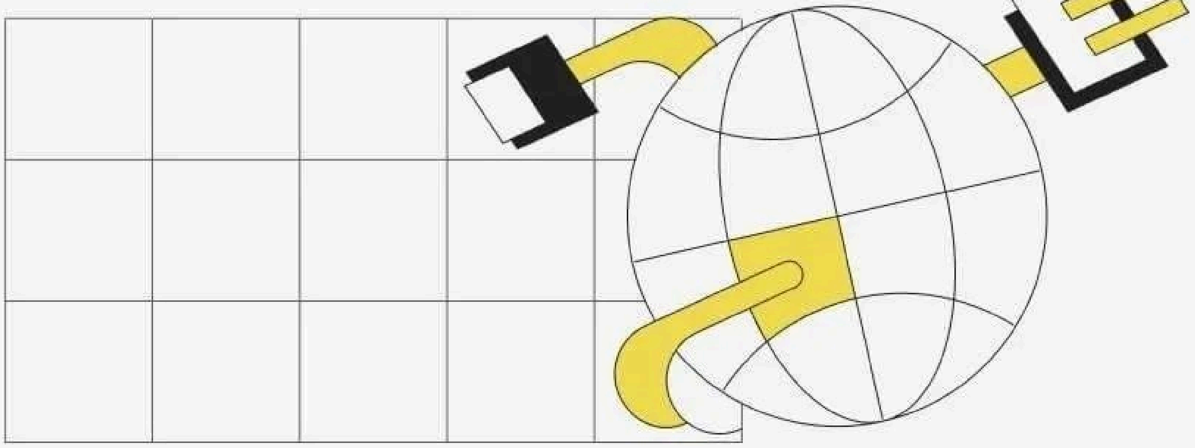
JS

Master The JavaScript Fetch API



Follow For More

Web Development | Design | Coding

Swipe

🔍 What Is The Fetch API?

Fetch is a **promise based JavaScript API** for making **asynchronous HTTP requests**.

Fetch is a simple, powerful and flexible way to **get** or **send** data from/to a **server.**

Even if the name **implies** that you can only "fetch" data, you can actually make any type of request: **GET, POST, PUT, PATCH, DELETE.**

Each **fetch call** returns a **promise**. This allows you to easily **handle the data** that you receive and the **errors** you might get.

Let's take a look at how it works!

Next

🚀 Basic Example: GET

Let's say we need to get a list of users from the seerver:

```javascript
1 fetch('https://jsonplaceholder.typicode.com/users')
2   .then(response => response.json())
3   .then(data => console.log(data))
4   .catch(err => console.error(err))
```

We call **fetch()** and give it a **request URL** as its **parameter**. Since we know that fetch will return a **promise**, we use **.then()** to access the **server's response**. The **response object** returned on **line 2** contains **all** of the **response's data**, including **headers, status code, status message.**

Since we know that we're expecting a **JSON response**, we can call the **.json()** method to be able to acces the actual data in the chained **.then()** call.

We can also use a **.catch()** block to handle possible errors **thrown by the server.**

Next

💡 Possible Response Types

Not all calls will return **JSON** responses, so it's useful to be aware that the **response object** returned by **fetch** has multiple **methods you can use:**

```javascript
// creates a clone of the response
response.clone()

// creates a new response with a different URL
response.redirect()

// returns a promise that resolves with an ArrayBuffer
response.arrayBuffer()

// returns a promise that resolves with a FormData Object
response.formData()

// returns a promise that resolves with a Blob
response.blob()

// returns a promise that resolves with a string
response.text()

// returns a promise that resolves with JSON
response.json()
```

Next

📄 Accessing Response Info

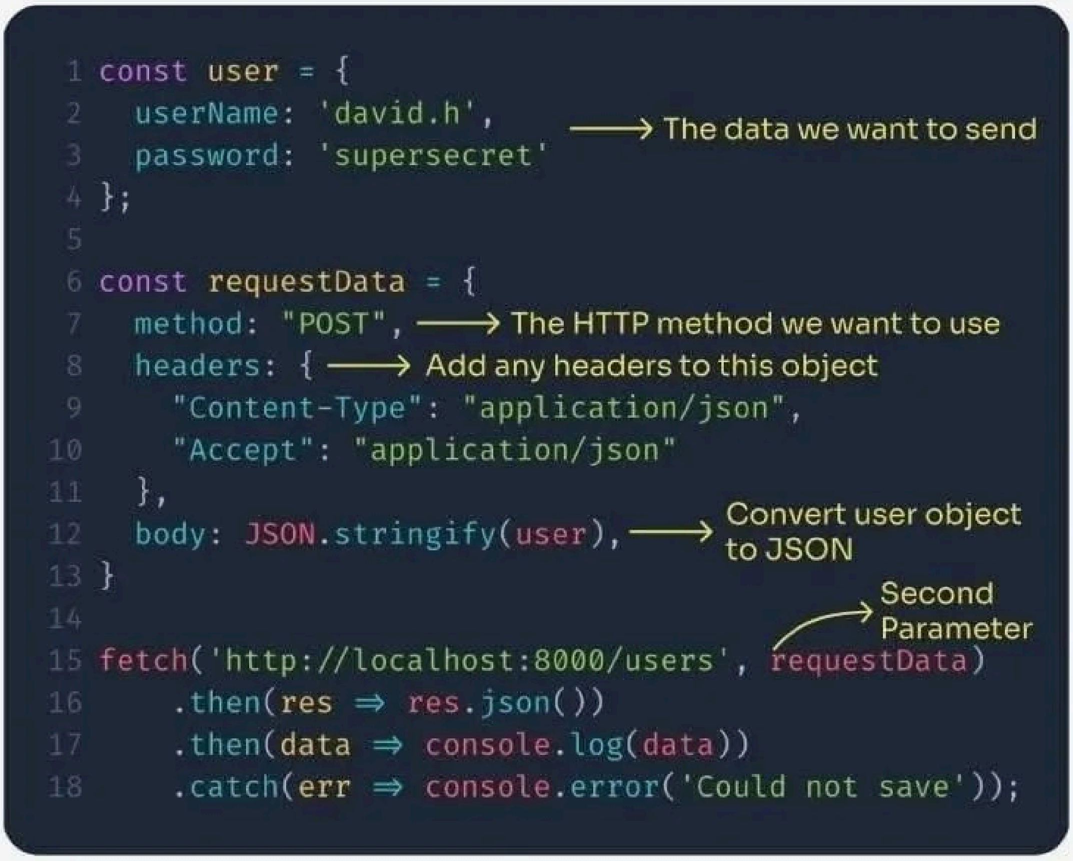
Besides the methods we use to manipulate the data in our response, we also have **access** to some other fields that might hold **useful information:**

```javascript
fetch('https://jsonplaceholder.typicode.com/posts')
  .then(response => {
    // accessing response headers
    console.log(response.headers.get('content-type'));

    // the HTTP response status code
    console.log(response.status);

    // true if status code is between 200 and 299
    console.log(response.ok);

    // status message of the response e.g. `Not Found`
    console.log(response.statusText);

    // true if there was a redirect
    console.log(response.redirected);

    // the response type (e.g., `basic`, `cors`)
    console.log(response.type);

    // the full url of the request
    console.log(response.url);
  });
```

Next

✍🏻 Making Write Requests

You can make **POST, PUT** or **PATCH** requests using **fetch** by adding a **second parameter**, an **object** that will contain the necessary details. Here's how:



```
const user = {
  userName: 'david.h',
  password: 'supersecret'
};

const requestData = {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Accept": "application/json"
  },
  body: JSON.stringify(user),
}

fetch('http://localhost:8000/users', requestData)
    .then(res ⇒ res.json())
    .then(data ⇒ console.log(data))
    .catch(err ⇒ console.error('Could not save'));
```

The data we want to send
The HTTP method we want to use
Add any headers to this object
Convert user object to JSON
Second Parameter

Done 🎉 You've just made a HTTP POST request to the server using fetch!

Next

❌ Making Delete Requests

You can also **delete** resources with **fetch** by using **DELETE** as the method, **like so:**

```javascript
const requestData = {
  method: "DELETE",
  headers: {
    "Content-Type": "application/json",
    "Accept": "application/json"
  }
}

fetch('http://localhost:8000/users/6', requestData)
    .then(res => res.json())
    .then(data => console.log(data))
    .catch(err => console.error('Could not delete'));
```

Here we add the **user's ID** in the **URL** so that the server knows which user we want to delete and **make the request.**

Now you're more than ready to work with APIs using JavaScript's fetch!

Next