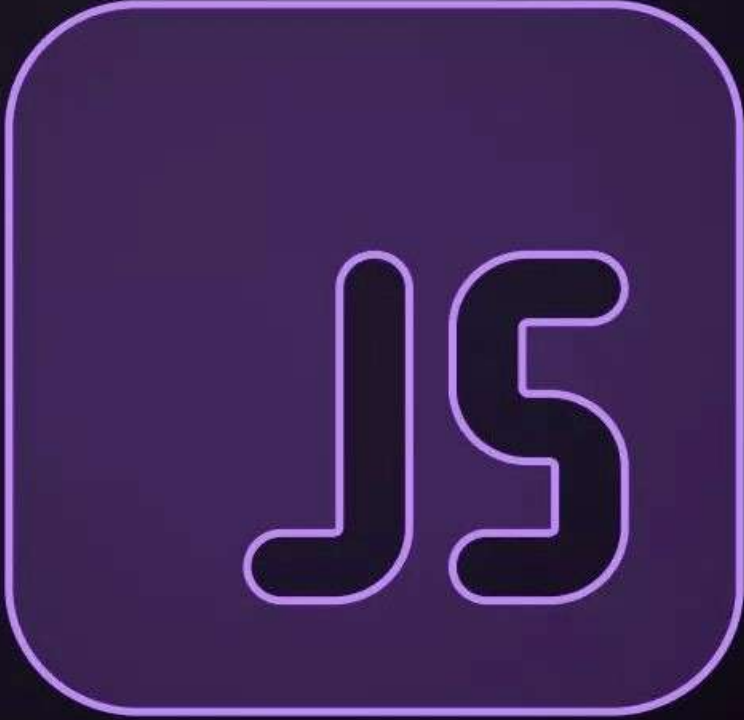


Fetch API

JAVASCRIPT

A large, rounded square with a purple gradient background. Inside the square, the letters 'JS' are written in a white, outlined, sans-serif font. The 'J' and 'S' are connected at the top and bottom.

JS

Hey Everyone 🖐️

JavaScript is everywhere. Millions of webpages are built on JS.

In this Post, you'll learn about the JavaScript Fetch API and how to use it to make asynchronous HTTP requests.

Do Like, save and Share This Post If You Found This Helpful.

Fetch API

The **Fetch API** is a modern interface that allows you to make **HTTP requests to servers** from web browsers.

- If you have worked with **XMLHttpRequest (XHR)** object.
- Fetch API can perform **all the tasks** as the XHR object does.
- Fetch API is much **simpler and cleaner**.
- It uses the **Promise** to deliver more **flexible features** to make requests to servers from the web browsers.

Sending a Request

The `fetch()` method is available in the **global scope** that instructs the web browsers to send a request to a URL.

- The `fetch()` requires only **one parameter** which is the **URL** of the resource that you want to fetch.
- When the **request completes**, the **promise** will resolve into a **Response** object.

```
let response = fetch(url);
```

Reading the Response

The `fetch()` method returns a `Promise` so you can use the `then()` and `catch()` methods to handle it.

```
fetch(url)
  .then((response) => {
    // handle the response
  })
  .catch((error) => {
    // handle the error
  });
```

- If the contents of the `response` are in the `raw text format`, you can use the `text()` method.

The `text()` method returns a Promise that resolves with the complete contents of the fetched resource.

```
fetch('/readme.txt')  
  .then(response => response.text())  
  .then(data => console.log(data));
```

Besides the `text()` method, the Response object has other methods such as `json()`, `blob()`, `formData()` and `arrayBuffer()` to handle the respective type of data.

In practice, you often use the `async/await` with the `fetch()` method like this:

```
async function fetchText() {  
  let response = await fetch('/readme.txt');  
  let data = await response.text();  
  console.log(data);  
}
```

Handling the status codes

The `Response` object provides the status code and status text via the `status` and `statusText` properties.

```
async function fetchText() {  
  let response = await fetch('readme.txt');  
  
  console.log(response.status); //200  
  console.log(response.statusText); //OK  
  
  if (response.status === 200) {  
    let data = await response.text();  
    // handle data  
  }  
}  
  
fetchText();
```


☒ **Hello Everyone!**

☒ **Learn JavaScript for Free with
an AI-Powered Platform!** ☒☒

Ready to master JavaScript? ☒☒
Look no further! Discover the
power of coding with our state-
of-the-art AI-powered platform.
☒☒

Level up your development
journey: **<https://clean-code.io>** ☒☒

NEXT →

**i post about Tech,
Coding and Career**

Follow for more content like this