@CODE.CLASH

Fetch API In JavaScript

NEXT →

1

@CODE.CLASH

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.

NEXT →

2

@CODE.CLASH

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.

NEXT →

3

@CODE.CLASH

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);
```

NEXT →

5

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.

NEXT →

4

@CODE.CLASH

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.

NEXT →

6

@CODE.CLASH

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);
}
```

NEXT →

7

@CODE.CLASH

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();
```

NEXT →

Did you find it Useful?

Leave a comment!

Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like Comment Repost