08

APPLICATION PROGRAMMING INTERFACE(API)

WHAT YOU WILL LEARN

- JavaScript Fetch API
- Sending a Request
- Reading the Response
- Handling the status codes of the Response

JavaScript Fetch API

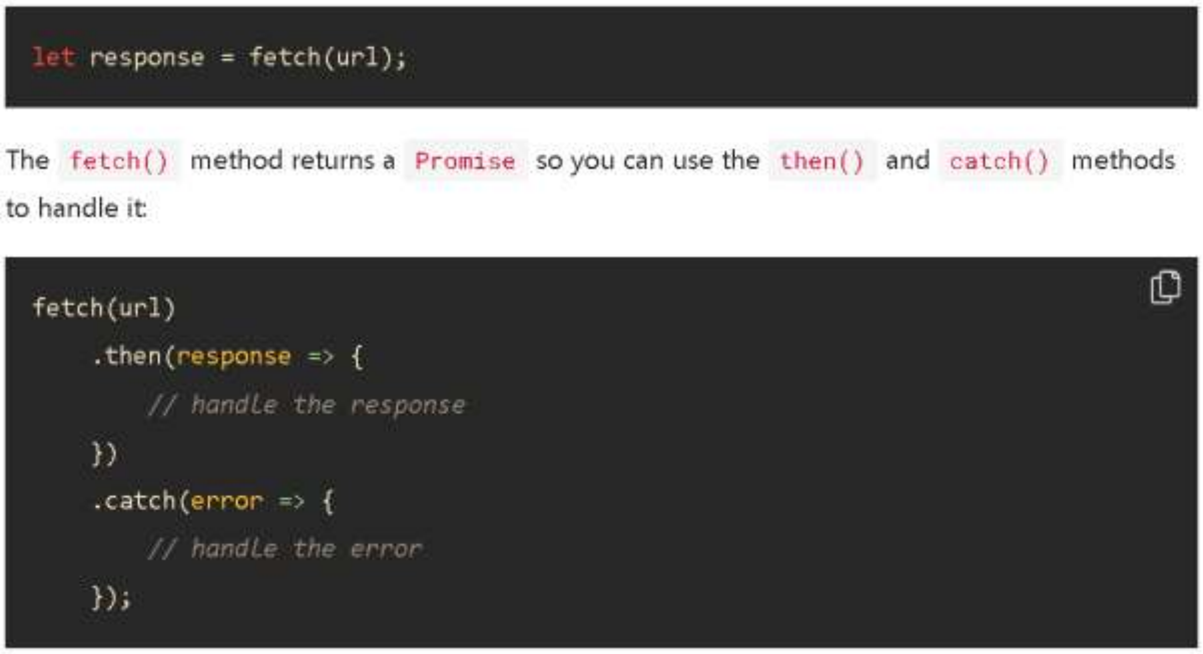
- The Fetch API is a modern interface that allows you to make HTTP requests to servers from web browsers.

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

- If you have worked with XMLHttpRequest (XHR) object, the Fetch API can perform all the tasks as the XHR object does.

Sending a Request

The fetch() requires only one parameter which is the URL of the resource that you want to fetch:



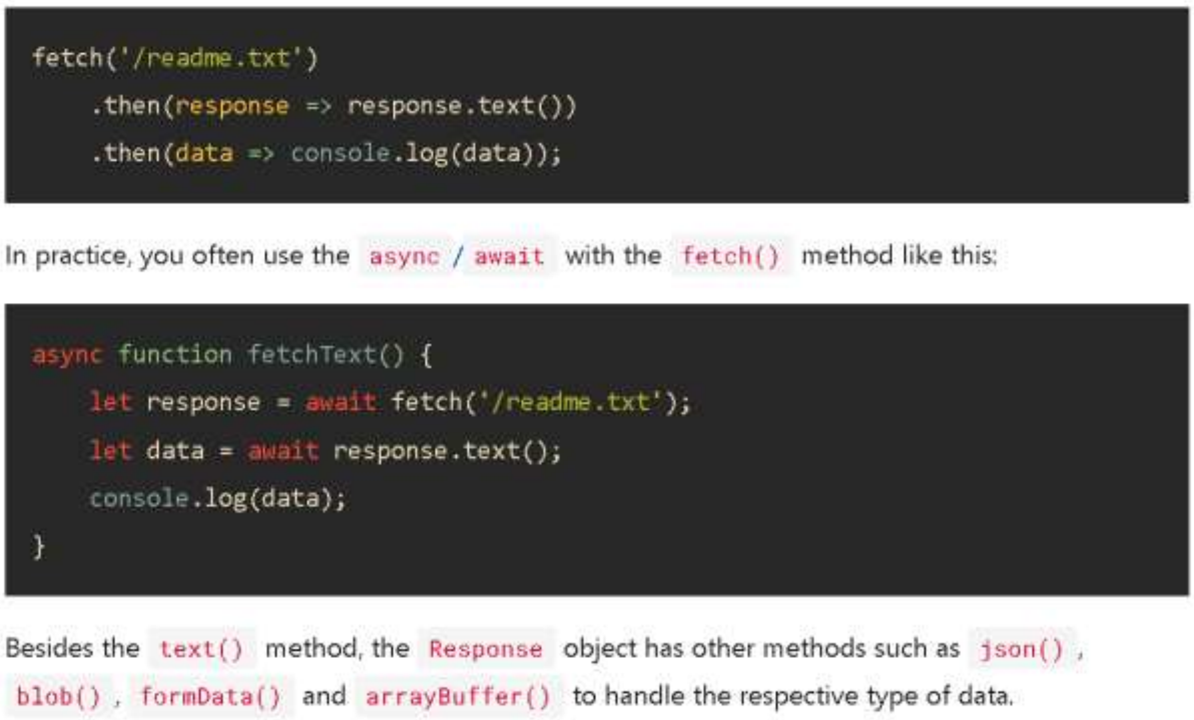
```
let response = fetch(url);
```

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
    });
```

Reading the Response

- 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));
```

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);
}
```

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

Handling the status codes of the Response

The `Response` object provides the status code and status text via the `status` and `statusText` properties. When a request is successful, the status code is `200` and status text is `OK` :

```
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();
```

Output:

```
200
OK
```

If the requested resource doesn't exist, the response code is `404` :

```
let response = await fetch('/non-existence.txt');

console.log(response.status); // 400
console.log(response.statusText); // Not Found
```

Output:

```
400
Not Found
```