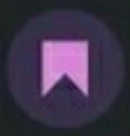

Fetch API
Ultimate Guide

If you want to load data from an API then the fetch API is the best way to do that in JavaScript. The **fetch API is a promise based API.**

Call the fetch API

Just pass a URL to the fetch function.

```
fetch("https://jsonplaceholder.typicode.com/users")
```

This will return a promise that contains the response data. This **response data contains properties for the status as well as methods** for converting the raw response data to JSON, text, or other formats.

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(res => {
    console.log(res.ok) // true
    console.log(res.status) // 200
    return res.json()
  })
```

Fetch Options

method

This option allows you to set which HTTP verb you want to use (GET, POST, PUT, DELETE, etc).

```
fetch("https://jsonplaceholder.typicode.com/users/2", {
  method: "DELETE"
})
```

body

If you are modifying the method then chances are you will need to pass data along with your request. That is where the body option comes in. The body does not accept objects so if you want to pass JSON to your API you must first convert it to a string.

```
fetch("https://jsonplaceholder.typicode.com/users", {
  method: "POST",
  body: JSON.stringify({ name: "Kyle" })
})
```

headers

You need to set the proper headers to tell your API that you are sending along JSON information. This headers option lets you set any HTTP header that you want.
This below set of code is everything you need to do in order to pass JSON to an API.

```
fetch("https://jsonplaceholder.typicode.com/users", {
  method: "POST",
  body: JSON.stringify({ name: "Kyle" }),
  headers: { "Content-Type": "application/json" }
})
```

Advanced Fetch Uses

- fetch API will not throw an error if you get back a 404, 500, or any other error HTTP response.
- The only way you can determine if a request failed is to check the ok property of the response.

```
fetch("https://jsonplaceholder.typicode.com/users/-1")
  .then(res => {
    console.log(res.ok) // false
    console.log(res.status) // 404
})
```

Failed request Code

```
fetch("https://jsonplaceholder.typicode.com/users/-1")
  .then(res => {
    if (res.ok) return res.json()
    return Promise.reject(res)
  })
  .then(data => console.log(data))
  .catch(res => console.error(res.status)) // 404
```

If the response is ok then just keep all code the same as normal, otherwise return a rejected promise that contains the response so I can handle it in a .catch.

Custom fetch function

```
function jsonFetch(url, { body, headers, ...options } = {}) {
  return fetch(url, {
    headers: { "Content-Type": "application/json", ...headers }
    body: JSON.stringify(body)
    ...options
  })
  .then(res => {
    if (res.ok) return res.json()
    return Promise.reject(res)
  })
  .then(res => res.json())
}
```

This custom function will take care of all the extra code I need to send JSON data and will still allow me to utilize all the custom options of fetch. It also handles throwing errors for things like 404s.

Turn on post notification

THANKS FOR READING

FOLLOW FOR MORE AMAZING CONTENT

Like

Save