CodeWithHarry

HTML CSS JS C C++ JAVA PYTHON PHP

Fetch api in Javascript | JavaScript Tutorial In Hindi #42

Overview Q&A Downloads Announcements

Fetch api in Javascript | JavaScript Tutorial In Hindi #42

In this tutorial, we will study about fetch API. The Fetch API is a promise-based API of JavaScript for making asynchronous HTTP requests in the browser similar to XMLHttpRequest (XHR). Unlike XHR, the fetch API is a simple and clean API that uses promises to provides more powerful features to fetch resources from the server. Fetch API is standardized now and is supported by all modern browsers except IE. The fetch() method only has one mandatory argument: the URL of the resource that we want to fetch.

How to use fetch API:-

To use a Fetch API, just pass the URL, the path to the resource we want to fetch, to fetch() method. Here is the syntax:

```
fetch('/js/users.json')
   .then(response => {
     // handle response data
   })
   .catch(err => {
     // handle errors
   });
```

Pass the path of the resource that we want to retrieve as a parameter to fetch(). We cannot block the user interface by waiting until the request finishes. That is why to fetch() returns a Promise, an object which represents a future result. We are using the then method to wait for the server's response. The catch() method is optional. Its purpose is to intercepts errors if the request fails to complete due to network failure or any other reason.
Now let us see how we can extract the JSON from that response once the request completes:

```
fetch('URL here')
.then(res => res.json())
.then(json => console.log(json));
```

We start the request by calling fetch(). When the promise is fulfilled, it returns a response object, which exposes a json method. Within the first then(), we can call this json method to return the response body as JSON.

GET Request:-

GET requests are widely used methods in APIs and websites. The purpose of this method is to retrieve data from the server at the specified resource. The Fetch API uses the GET method for asynchronous requests. Here is an example of get request:

```
fetch('https://api.github.com/orgs/nodejs')
.then(response => response.json())
.then(data => {
  console.log(data) // Prints result from `response.json()` in getRequest
})
.catch(error => console.error(error))
```

As we know, the fetch() method returns a promise. The response returned by the promise is a stream object, which means that it returns another promise when we call the json() method. Call to json() method indicates that we are expecting a JSON response. If we want an XML response, use the text() method

POST Request:-

The purpose of the post request is to send the data to the server and creates a new resource. The resource post request creates subordinate to some other parent resource. When a new resource is posted to the parent, the API service will automatically associate the new resource by assigning it an ID. All we need to do is set the method and body parameters in the fetch() options:

```
let data = {
   first_name: 'John',
   last_name: 'Adams',
   job_title: 'Software Engineer'
};
const options = {
   method: 'POST',
   body: JSON.stringify(data),
   headers: {
     'Content-Type': 'application/json'
}
}
```

```
fetch('https://reqres.in/api/users', options)
   .then(res => res.json())
   .then(res => console.log(res));
```

Code file tut42.js as described in the video

```
console.log('This is my tutorial 42');

// Button with id myBtn
let myBtn = document.getElementById("myBtn");

// div with id content
let content = document.getElementById("content");

// function getData(){
//     console.log("Started getData")
//     url = "harry.txt";
//     fetch(url).then((response)=>{
//         console.log("Inside first then")
//         return response.text();
//     }).then((data)=>{
//         console.log("Inside second then")
//         console.log(data);
//     })
// }

function getData(){
    console.log("Started getData")
    url = "https://api.github.com/users";
    fetch(url).then((response)=>{
        console.log("Inside first then")
        return response.json();
    }).then((data)=>{
        console.log("Inside second then")
        console.log(data);
    })
}
```

```
function postData(){
    url = "http://dummy.restapiexample.com/api/v1/create";
    data = '{"name":"harglry347485945","salary":"123","age":"23"}'
    params = {
        method:'post',
        headers: {
            'Content-Type': 'application/json'
        },
        body: data
    }
    fetch(url, params).then(response=> response.json())
    .then(data => console.log(data)
    )
}

// console.log("Before running getData")
// getData()
// console.log("After running getData")
postData()
```

Previous

Next