Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Fetch API in JavaScript

Definition: The Fetch API is a modern interface that allows you to make network requests similar to XMLHttpRequest (XHR). It provides a more powerful and flexible feature set for working with network requests, and it returns promises, making it easier to work with asynchronous operations.

Key Features:

- **Promise-Based:** Fetch returns promises, allowing for easier and cleaner handling of asynchronous operations.
- **Readable Stream:** Fetch provides access to the response body as a readable stream.
- **Modern API:** Simplifies the process of making HTTP requests compared to the older XMLHttpRequest API.

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

url: The URL to which the request is sent.
options (optional): An object containing any custom settings that you want to apply to the request. These options include method, headers, body, mode, credentials, cache, redirect, referrer, and referrerPolicy.

Making a Simple GET Request:

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
    .then(response => {
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        return response.json(); // Parse the JSON from the response
    })
    .then(data => {
        console.log(data); // Use the JSON data
    })
    .catch(error => {
        console.error('There has been a problem with your fetch operation:', error);
    });
```

Handling Different Response Formats:

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
    .then(response => response.json())
```

```
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Making a POST Request:

To send data to the server, you need to use the POST method and include the data in the request body.

```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    title: 'foo',
    body: 'bar',
    userId: 1
  })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

Handling Errors:

To handle errors in fetch requests, use the catch method to catch any network errors and check the ok property of the response object.

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    return response.json();
  })
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Customizing Fetch Requests:

The Fetch API provides a wide range of options to customize your request.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'GET', // GET, POST, PUT, DELETE, etc.
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer token'
  },
```

```
    mode: 'cors', // no-cors, cors, same-origin
    cache: 'default', // default, no-cache, reload, force-cache, only-if-cached
    credentials: 'same-origin', // include, same-origin, omit
    redirect: 'follow', // manual, follow, error
    referrer: 'client', // no-referrer, client
    referrerPolicy: 'no-referrer', // no-referrer, no-referrer-when-downgrade, origin, origin-
when-cross-origin, unsafe-url
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

Interview Questions:

What is the Fetch API?

- **Answer:** The Fetch API is a modern interface that allows you to make network requests similar to XMLHttpRequest (XHR). It returns promises and provides a more powerful and flexible feature set for working with network requests in JavaScript.

How do you make a basic GET request using the Fetch API?

```
    fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

What does the Fetch API return and how do you handle it?

- **Answer:** The Fetch API returns a promise that resolves to a Response object. You can handle it using the then method to process the response and catch to handle errors. Methods like json(), text(), and blob() can be used to read the response body.

How can you handle errors in Fetch API requests?

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    return response.json();
  })
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

What are the advantages of using Fetch API over XMLHttpRequest?

- **Answer:**
  - Fetch API uses promises, which make it easier to work with asynchronous operations.
  - It provides a more powerful and flexible feature set.
  - It simplifies the process of making HTTP requests and handling responses.
  - It has a cleaner and more modern syntax.

How do you make a POST request with Fetch API?

```
fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        title: 'foo',
        body: 'bar',
        userId: 1
    })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

How can you handle different types of responses (JSON, text, blob) using Fetch API?

```
        // JSON
fetch('https://jsonplaceholder.typicode.com/posts/1')
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.error('Error:', error));

// Text
fetch('https://jsonplaceholder.typicode.com/posts/1')
    .then(response => response.text())
    .then(data => console.log(data))
    .catch(error => console.error('Error:', error));

// Blob
fetch('https://jsonplaceholder.typicode.com/posts/1')
    .then(response => response.blob())
    .then(data => {
        const url = URL.createObjectURL(data);
        console.log(url);
    })
    .catch(error => console.error('Error:', error));
```

Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

What options can you customize in a Fetch API request?

- **Answer:** You can customize various options in a Fetch API request, such as:
  - method: HTTP request method (e.g., GET, POST, PUT, DELETE).
  - headers: HTTP headers to include in the request.
  - body: The body of the request (for methods like POST).
  - mode: Request mode (e.g., cors, no-cors, same-origin).
  - credentials: Whether to include credentials (e.g., omit, same-origin, include).
  - cache: Cache mode (e.g., default, no-cache, reload, force-cache, only-if-cached).
  - redirect: How to handle redirects (e.g., follow, error, manual).
  - referrer: Referrer of the request (e.g., client, no-referrer).
  - referrerPolicy: Referrer policy (e.g., no-referrer, origin, unsafe-url).

How do you abort a Fetch request?

```
const controller = new AbortController();
const signal = controller.signal;

fetch('https://jsonplaceholder.typicode.com/posts/1', { signal })
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => {
        if (error.name === 'AbortError') {
            console.log('Fetch aborted');
        } else {
            console.error('Fetch error:', error);
        }
    });

// Abort the fetch request after 1 second
setTimeout(() => controller.abort(), 1000);
```