@CODE.CLASH

# Axios API

JAVASCRIPT


1

@CODE.CLASH

# What is Axios

Axios is a popular JavaScript library used for making HTTP requests from a web browser or Node.js.

- Whether you're building a web app or a server-side application, Axios makes it easy to work with APIs and other HTTP services.
- In this post, we'll show you how to get started with Axios, from installation to making basic GET and POST requests and handling errors, using async/await syntax.

2

@CODE.CLASH

# Setting Up Axios

To start using Axios, you'll need to install it in your project. You can do this using NPM or a CDN.

Here's an example of how to install Axios using NPM:

```
npm install axios
```

Once you've installed Axios, you can include it in your HTML or JavaScript code.

Here's an example of how to include Axios in your Code:

```
// In HTML
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

// OR if you're using a module bundler
import axios from 'axios';
```

3

@CODE.CLASH

# Basic GET Request

let's make a simple GET request to retrieve data from an API using async/await syntax.

Here's an example of how to make a GET request with Axios using async/await:

```
const fetchData = async() => {
  try {
    const response = await axios.get('https://example.com/data');
    console.log(response.data);
  } catch (error) {
    console.error(error);
  }
}

fetchData();
```

We're using the async/await keywords to create an asynchronous function and wait for the response from the API before logging the data to the console.

4

@CODE.CLASH

# Basic POST Request

Now, let's make a simple POST request to send data.

Here's an example of how to make a POST request with Axios using async/await:

```
const postData = async() => {
  try {
    const response = await axios.post('https://example.com/data', {
      firstName: 'Imtiyaz',
      lastName: 'Nandasaniya'
    });
    console.log(response.data);
  } catch (error) {
    console.error(error);
  }
}

postData();
```

The data we're sending is an object with two properties: firstName and lastName.

If the request is successful, the response data will be logged to the console.

5

@CODE.CLASH

# Handling Errors

HTTP requests can sometimes fail, for various reasons such as network errors or server-side errors.

It's important to handle these errors gracefully in your code.

Fortunately, Axios provides several options for handling errors and exceptions.

Here's an example of how to handle a 404 error using `async/await

```
const fetchDataasync = async() => {
  try {
    const response = await axios.get('https://example.com/data');
    console.log(response.data);
  } catch (error) {
    if (error.response.status === 404) {
      console.error('Data not found');
    } else {
      console.error(error);
    }
  }
}

fetchData();
```

6

@CODE.CLASH

- In this example, we're using the if statement to check if the HTTP response status is 404, which indicates that the requested data was not found.
- If the status is 404, we're logging a custom error message to the console.
- If the status is anything else, we're logging the original error message to the console.

7

# Conclusion

- Axios is a powerful JavaScript library that makes it easy to work with HTTP requests in your web apps or Node.js projects.
- Whether you need to make a simple GET request or a more complex POST request, Axios provides an intuitive and flexible API that you can use with async/await syntax to create robust and reliable applications.
- With this introduction to Axios and async/await, you're well on your way to becoming a skilled JavaScript developer.
- As always, I hope you enjoyed the post and learned something new.
- If you have any queries then let me know in the comment box.