Axios

Axios is a JavaScript library for making HTTP requests to interact with APIs. It's commonly used in modern web development with frameworks like React, Vue, and Angular because it simplifies working with APIs and handling data.

Key Features of Axios

1. **Promise-Based**: Axios uses promises, making it easy to handle asynchronous operations and data fetching.
2. **Automatic JSON Parsing**: Axios automatically parses JSON responses, so you don't need to use .json() to get the data as you do with the native fetch API.
3. **Global Configuration**: Axios allows you to set a base URL, headers, or timeouts that apply to all requests. This is useful for setting up configurations only once and reusing them across the application.
4. **Error Handling**: Axios automatically catches HTTP errors (like 404 or 500), as well as network errors, making it easy to manage error states in the application.

Installing Axios:

```
npm install axios
```

Importing Axios:

```
import axios from 'axios';
```

Common Axios Methods

1. **GET** (Retrieve Data):

```
axios.get('https://jsonplaceholder.typicode.com/posts').then((res)=>{
        console.log(res.data)
        setItems(res.data.slice(0,5))

    })
```

POST (Send Data):

```
axios.post(`https://jsonplaceholder.typicode.com/posts`,addItem).then((res)=>{
        console.log(res.data)
        setItems([...items,res.data])
        setItemName('')
    })
```

PUT (Update Data):

```
axios.put(`https://jsonplaceholder.typicode.com/posts/${id}`,updateItem).then((res)=>{
            console.log(res.data)
            setItems(items.map((item)=>{
               console.log(item.id == id ? res.data : item)
               return item.id == id ? res.data : item
            }))
```

DELETE (Remove Data):

```
 axios.delete(`https://jsonplaceholder.typicode.com/posts/${id}`).then((res)=>{
         console.log(res.data)
         setItems(items.filter((item)=>{
            console.log(item.id !== id)
            return item.id !== id
         }))
```

Interview questions:

What is Axios, and why is it used?

Answer:
Axios is a JavaScript library for making HTTP requests. It's commonly used in both browser-based and Node.js applications to interact with APIs. Axios simplifies making requests, handling responses, and managing errors. It automatically parses JSON, provides easy error handling, and allows configuration settings like global defaults and interceptors. Axios is often preferred over the native fetch API due to its additional features and ease of use, especially in applications that make multiple API requests.

How do you install and set up Axios in a project?

Answer:
To install Axios, you can use npm or yarn:

```
npm install axios

# or

yarn add axios
```

After installing, you can import Axios into your JavaScript or React file:

```
import axios from 'axios';
```

You can then start making HTTP requests using Axios's various methods, like axios.get(), axios.post(), etc

How does Axios handle JSON responses compared to the fetch API?

Answer:
Axios automatically parses JSON responses, making the data immediately available in response.data. With fetch, you have to manually parse the JSON by calling response.json().

Example: Using Axios:

```
axios.get('https://api.example.com/data')
  .then(response => console.log(response.data));
```

Using fetch:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data));
```

This automatic JSON parsing in Axios makes it easier and quicker to work with API data.

4. What are interceptors in Axios, and how are they useful?

Answer:
Interceptors in Axios are functions that allow you to process or modify requests and responses before they are handled by .then() or .catch(). They are useful for adding global logic to requests, such as attaching authorization tokens, logging, or handling errors in a centralized way.

Example of a Request Interceptor:

```
axios.interceptors.request.use(config => {
  config.headers['Authorization'] = 'Bearer myToken';
  return config;
}, error => Promise.reject(error));
```

This interceptor adds an Authorization header to every outgoing request.

Example of a Response Interceptor:

```
axios.interceptors.response.use(response => response, error => {
  console.error('Error:', error.response ? error.response.status : error.message);
  return Promise.reject(error);
});
```

This interceptor logs error details whenever a response error occurs.

What are the different HTTP methods supported by Axios?

Answer:
Axios supports several HTTP methods, including:

- **GET**: Used to retrieve data.
- **POST**: Used to send new data to the server.
- **PUT**: Used to update existing data.
- **DELETE**: Used to delete data.
- **PATCH**: Similar to PUT, but typically used to update a partial resource.

Each method can be used with Axios's syntax, like axios.get(), axios.post(), axios.put(), etc.

How does Axios handle errors, and how can you use .catch() to manage errors?

Answer:
Axios automatically throws errors for both network issues and HTTP errors (like 404 or 500). The .catch() method can be used to manage these errors. In the .catch() block, you can check error.response for HTTP errors and error.request for network issues.

```
axios.get('https://api.example.com/data')
  .then(response => console.log(response.data))
  .catch(error => {
    if (error.response) {
      // HTTP error
      console.error('HTTP Error:', error.response.status);
    } else if (error.request) {
      // Network error
      console.error('Network Error:', error.request);
    } else {
      console.error('Error:', error.message);
    }
  });
```

How does Axios compare to the fetch API? Why might you choose Axios?

Answer:
Axios has several advantages over fetch, including:

- **Automatic JSON Parsing**: Axios automatically parses JSON responses, while fetch requires .json() to be called manually.
- **Error Handling**: Axios throws errors for both HTTP errors and network errors, whereas fetch only throws network errors by default.
- **Interceptors**: Axios supports request and response interceptors, which fetch doesn't natively support.
- **Request Cancellation**: Axios provides built-in request cancellation, while fetch requires the use of AbortController.

- **Global Configuration**: Axios supports setting global defaults, making it easier to apply configurations across requests.

How do you use Axios for CRUD operations?

Answer:
CRUD operations (Create, Read, Update, Delete) are easily handled with Axios's HTTP methods:

Create (POST):

```
axios.post('https://api.example.com/items', { name: 'Item 1' })
  .then(response => console.log(response.data));
```

Read (GET):

```
  axios.get('https://api.example.com/items')
  .then(response => console.log(response.data));
```

Update (PUT):

```
  axios.put('https://api.example.com/items/1', { name: 'Updated Item' })
  .then(response => console.log(response.data));
```

Delete (DELETE):

```

  axios.delete('https://api.example.com/items/1')
  .then(response => console.log('Item deleted'));
```