Server Communication in React (AJAX, Axios & APIs)

By
Narasimha Rao T
Microsoft.Net FSD Trainer
Professional Development Trainer
tnrao.trainer@gmail.com

1. What is AJAX?

- **AJAX** = *Asynchronous JavaScript and XML*
- A technique to **send/receive data from a server asynchronously** without reloading the entire page.
- Modern apps often exchange data in **JSON** instead of XML.
- Used in SPAs (Single Page Applications) like React for smooth user experience.

Example use cases:

- Fetching data (news, products, weather info).
- Submitting forms without page reload.
- Real-time search suggestions.

2. Why do we need to communicate with a server/API?

- React apps are mostly **frontend only**.
- To make apps dynamic, we need **data from backend** (e.g., users, posts, payments).
- APIs help:
  - Fetch data from DB (GET).
  - Send data to DB (POST).
  - Update existing data (PUT/PATCH).
  - Remove data (DELETE).

3. Multiple Ways to Communicate with Server

1. **Fetch API (built-in)**

```
fetch("https://api.example.com/data")
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err));
```

2. **Axios (third-party library)** – more powerful and easier to use.

3. **Other libraries**

4. Axios Package

- A **popular HTTP client** for making API requests.
- Features:
  - Promise-based.
  - Automatic JSON transformation.
  - Interceptors (modify requests/responses).
  - Handles request cancellation.
  - Better error handling than Fetch.

5. Axios Setup

Install:

```
npm install axios
```

Basic Usage:

```
import axios from "axios";

axios.get("https://api.example.com/data")
  .then(res => console.log(res.data))
  .catch(err => console.error(err));
```

Fetching Products Data in React

```
import React, { useEffect, useState } from "react";
import axios from "axios";

function Products() {
  const [products, setProducts] = useState([]);


    axios.get("https://fakestoreapi.com/products")
        .then((res) => {
            setProducts(res.data);
        })
        .catch((err) => {
            setError(err.message);
        }
    );
}
```

6. CRUD Operations using Axios

- ## GET (Read Data)

```
useEffect(() => {
  axios.get("/api/users")
    .then(res => setUsers(res.data))
    .catch(err => console.error(err));
}, []);
```

- ## POST (Create Data)

```
axios.post("/api/users", { name: "John", age: 25 })
  .then(res => console.log("User Created:", res.data));
```

PUT (Update Data)

```
axios.put("/api/users/1", { name: "John Updated" })
  .then(res => console.log("User Updated:", res.data));
```

DELETE (Remove Data)

```
axios.delete("/api/users/1")
  .then(res => console.log("User Deleted:", res.data));
```

7. Connecting & Securing API

- APIs are often **protected**.
- Security Measures:
  - **Authentication** (verify user).
  - **Authorization** (check permissions).
  - **JWT Tokens** (most common).

8. Adding JWT Tokens in Request Header

```
const token = localStorage.getItem("token");

axios.get("/api/profile", {
  headers: {
    Authorization: `Bearer ${token}`
  }
})
.then(res => console.log(res.data))
.catch(err => console.error(err));
```

9. Error & Loading State in React

Always handle **loading** & **error states** for better UX.

```
const [loading, setLoading] = useState(false);
const [error, setError] = useState(null);

useEffect(() => {
  setLoading(true);
  axios.get("/api/data")
    .then(res => setData(res.data))
    .catch(err => setError(err.message))
    .finally(() => setLoading(false));
}, []);
```

- Show a **spinner** while loading.
- Show **error message** if request fails.

10. CORS (Cross-Origin Resource Sharing)

- **Security feature** in browsers.
- Prevents frontend (e.g., `http://localhost:3000`) from calling backend on a **different domain** (`http://api.example.com`).
- If not configured, you'll see:
  - ❌ "CORS Policy: No 'Access-Control-Allow-Origin' header..."
- **Fix**: Backend must allow the frontend origin. Example (Express.js):

```
app.use(cors({ origin: "http://localhost:3000" }));
```

11. Common Interview Questions

1. What is AJAX and how does it work in React?
2. Difference between Fetch API and Axios?
3. How do you handle errors in Axios requests?
4. How would you add JWT authentication to Axios requests?
5. Explain CORS and how to solve related issues.

6. What are Axios interceptors and when to use them?
7. What is the difference between PUT and PATCH in REST APIs?
8. How do you manage loading and error states when making API calls in React?
9. Can you explain the difference between synchronous and asynchronous requests?
10. Why is it important to secure API endpoints?

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com