React Tip

S.O.L.I.D

03

Dependency Inversion Principle in JavaScript Projects

Step 1: Create an abstraction for the HTTP client

```
export class HttpClient {
  async get(url, config) {
    throw new Error('Method not implemented');
  }

  async post(url, data, config) {
    throw new Error('Method not implemented');
  }

  async put(url, data, config) {
    throw new Error('Method not implemented');
  }

  async delete(url, config) {
    throw new Error('Method not implemented');
  }
}
```

Start by creating a generic interface that defines the methods your HTTP client should implement. This ensures that if you ever replace Axios with another library, the core API of your application remains unaffected.

Step 2: Implement the HTTP client using Axios

```
import axios from 'axios';
import { HttpClient } from './HttpClient';

export class AxiosHttpClient extends HttpClient {
  constructor(baseURL) {
    super();
    this.client = axios.create({ baseURL });
  }

  async get(url, config) {
    return this.client.get(url, config);
  }

  async post(url, data, config) {
    return this.client.post(url, data, config);
  }

  async put(url, data, config) {
    return this.client.put(url, data, config);
  }

  async delete(url, config) {
    return this.client.delete(url, config);
  }
}
```

Next, create a concrete implementation of this interface using Axios.

Step 3: Use the abstraction in your code

```
const httpClient = new AxiosHttpClient('url');
httpClient.get(`user/${id}`)
```

Instead of relying directly on Axios, depend on the abstraction (HttpClient). This approach makes it easier to swap Axios for another library in the future (e.g., fetch).

Did you like the content?

Help me by reacting to this post