Manejo de Datos en React

Usando Hooks, Fetch y Axios

Tu Nombre

August 2, 2024

Contenido

1. Introducción a las Peticiones HTTP
2. Fetch API
3. Axios
4. Comparación entre Fetch y Axios
5. Mejores Prácticas

Peticiones HTTP en el Frontend

- Fundamentales para aplicaciones web modernas
- Permiten la comunicación cliente-servidor
- Métodos comunes: GET, POST, PUT, DELETE
- Dos principales formas en React: Fetch API y Axios

Introducción a Fetch API

- API nativa del navegador para realizar peticiones HTTP
- Basada en Promesas
- No requiere instalación adicional
- Reemplaza XMLHttpRequest

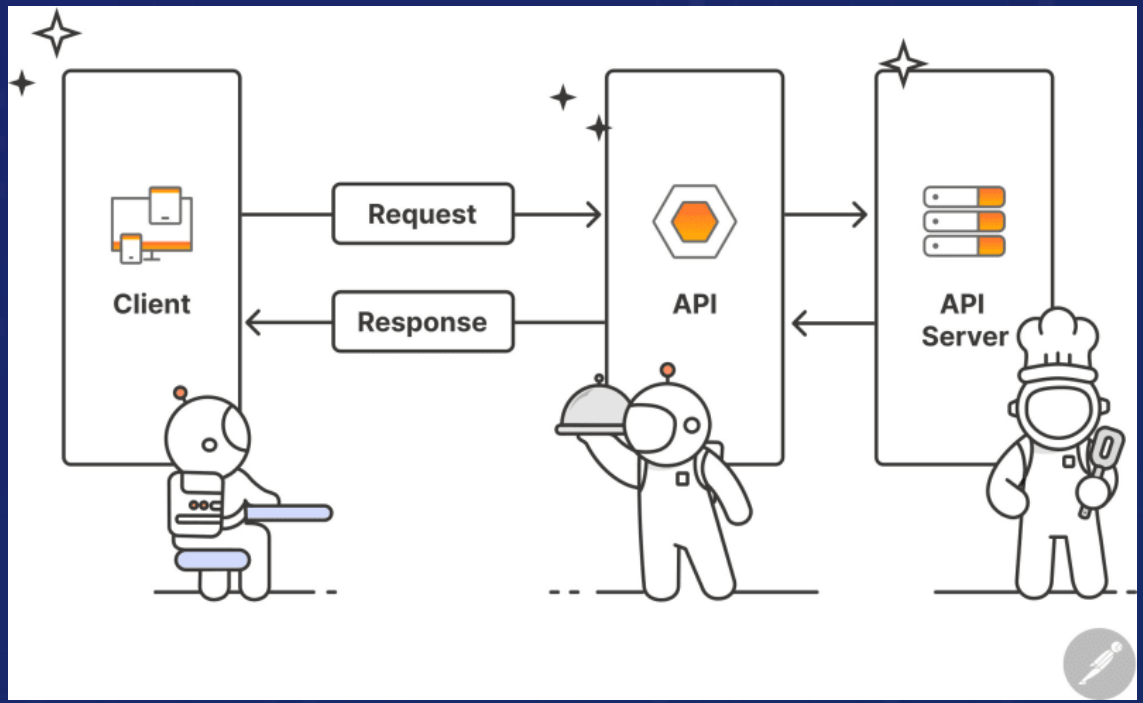

Figura: Funcionamiento de Fetch API

Uso Básico de Fetch

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

- fetch() devuelve una Promise
- Primera .then() convierte la respuesta a JSON
- Segunda .then() maneja los datos
- .catch() maneja errores

Opciones de Fetch

```
fetch('https://api.example.com/data', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify(data)
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

- method: GET, POST, PUT, DELETE, etc.
- headers: cabeceras de la petición
- body: datos a enviar (debe ser string)

Manejo de Errores en Fetch

```
fetch('https://api.example.com/data')
  .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));
```

- Fetch no rechaza la promesa en errores HTTP
- Debemos verificar manualmente response.ok
- Lanzar error si la respuesta no es ok

Introducción a Axios

- Biblioteca cliente HTTP basada en promesas
- Funciona en el navegador y en Node.js
- Ofrece una API más amigable que Fetch
- Requiere instalación: `npm install axios`

Figure: Logo de Axios

Uso Básico de Axios

```
import axios from 'axios';

axios.get('https://api.example.com/data')
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error('Error:', error);
  });
```

- Sintaxis más simple que Fetch
- Transforma automáticamente la respuesta a JSON
- Maneja errores HTTP automáticamente

Peticiones POST con Axios

```
axios.post('https://api.example.com/data', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error('Error:', error);
  });
```

- No es necesario stringificar el body
- Axios establece automáticamente el Content-Type

Configuración Global en Axios

```
axios.defaults.baseURL = 'https://api.example.com';
axios.defaults.headers.common['Authorization'] = AUTH_TOKEN;
axios.defaults.headers.post['Content-Type'] = 'application/jso

axios.get('/data')
  .then(response => console.log(response.data));
```

- Permite establecer configuraciones por defecto
- Útil para URLs base, headers comunes, etc.

Interceptores en Axios

```
// Agregar un interceptor de solicitud
axios.interceptors.request.use(function (config) {
    // Hacer algo antes de que se envíe la solicitud
    return config;
  }, function (error) {
    // Hacer algo con el error de la solicitud
    return Promise.reject(error);
  });

// Agregar un interceptor de respuesta
axios.interceptors.response.use(function (response) {
    // Cualquier código de estado que esté dentro del rango de
    return response;
  }, function (error) {
    // Cualquier código de estado que esté fuera del rango de
```

Fetch vs Axios

Fetch:

- Nativo del navegador
- No necesita instalación
- Menos funcionalidades incorporadas
- Requiere más código para tareas comunes

Axios:

- Necesita instalación
- Más funcionalidades incorporadas
- Sintaxis más simple
- Interceptores de solicitudes/respuestas
- Cancelación de solicitudes

Cuándo Usar Cada Uno

- Usa Fetch cuando:
  - Quieres evitar dependencias externas
  - Estás trabajando en un proyecto pequeño
  - Necesitas soporte nativo del navegador
- Usa Axios cuando:
  - Necesitas una API más robusta y fácil de usar
  - Trabajas en un proyecto grande o complejo
  - Necesitas características avanzadas como interceptores

Mejores Prácticas

- Maneja siempre los errores adecuadamente
- Usa async/await para un código más limpio
- Implementa timeouts para evitar peticiones colgadas
- Considera la cancelación de peticiones cuando sea necesario
- Usa interceptores para lógica común (por ejemplo, tokens de autenticación)
- Implementa retry logic para peticiones fallidas
- Considera el uso de caché para optimizar el rendimiento

Conclusiones

- Tanto Fetch como Axios son herramientas poderosas para realizar peticiones HTTP en React
- Fetch es nativo y simple, mientras que Axios ofrece más funcionalidades
- La elección entre Fetch y Axios depende de las necesidades específicas del proyecto
- Independientemente de la elección, es crucial manejar errores y seguir las mejores prácticas
- El manejo eficiente de datos es fundamental para crear aplicaciones React robustas y eficientes