VIETNAM AUSTRALIA
Vocational College

React Advance
Bài 7: Sử dụng AXIOS

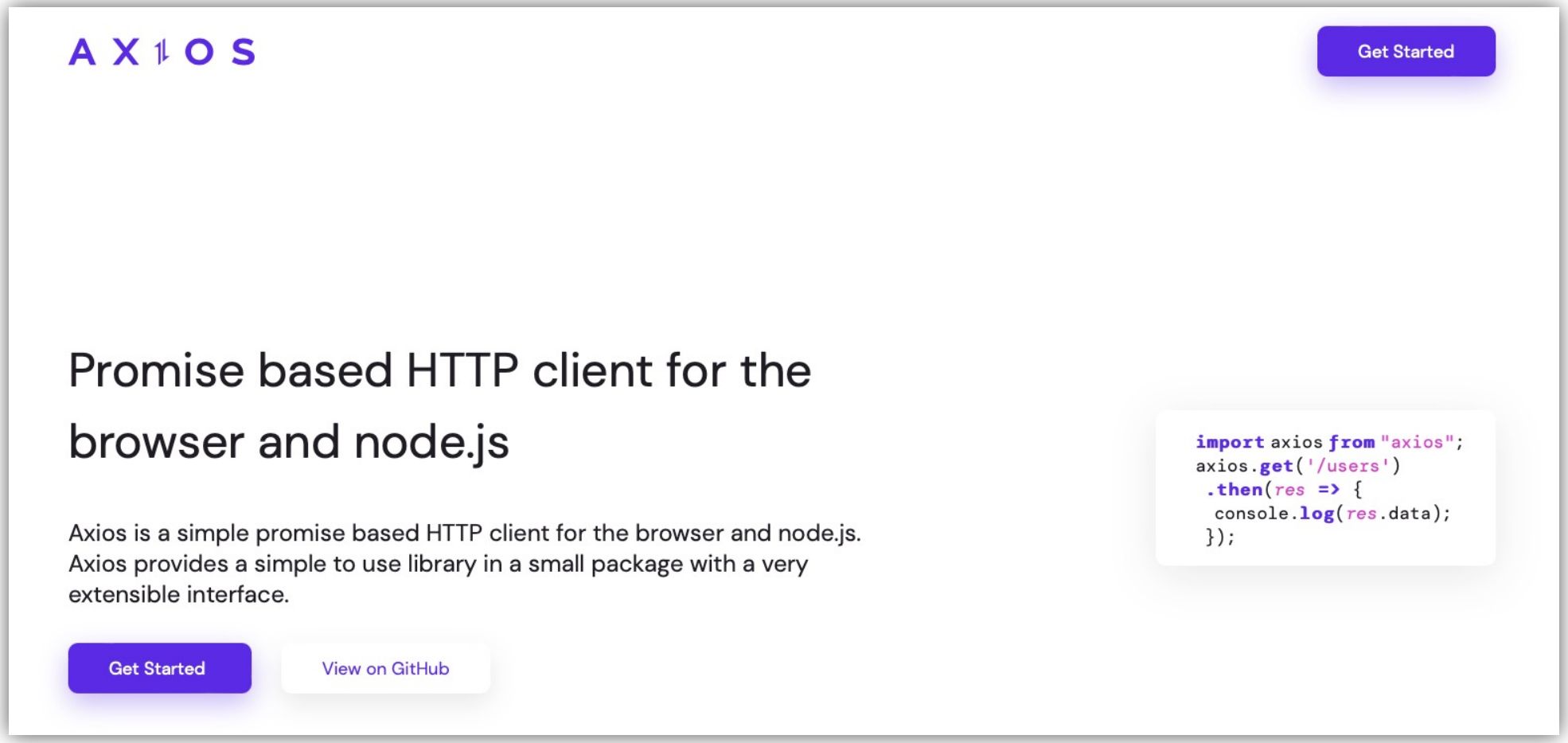
daynghevietuc.com

Nội dung

1. Giới thiệu Axios
2. Axios vs Fetch
3. Cài đặt Axios
4. Transforms và Interceptors
5. Tích hợp Axios vào ứng dụng

1. Giới thiệu AXIOS

Giới thiệu AXIOS



Giới thiệu AXIOS

- **Axios** là một *HTTP client* được viết dựa trên ***Promises*** được dùng để hỗ trợ cho việc xây dựng các ứng dụng API từ đơn giản đến phức tạp và có thể được sử dụng cả ở trình duyệt hay Node.js.
- Việc tạo ra một *HTTP request* dùng để fetch hay lưu dữ liệu là một trong những nhiệm vụ thường thấy mà một ứng dụng Javascript phía client cần phải làm khi muốn giao tiếp với phía server.
- Các thư viện bên thứ 3, đặc biệt là jQuery từ xưa đến nay vẫn là một trong những cách phổ biến để giúp cho các browser API tương tác tốt hơn, rõ ràng mạch lạc hơn và xóa đi những điểm khác biệt giữa các browser với nhau.

Giới thiệu AXIOS

- Khi ngày càng có nhiều developer thích dùng các *native DOM API* được nâng cấp và cải thiện hơn jQuery.
- Hay các thư viện lập trình front-end UI như React và Vue.js thì việc sử dụng cả một thư viện jQuery chỉ với một giá trị sử dụng đó là tính năng của hàm **$**.*ajax* đã trở nên bất hợp lý hơn bao giờ hết.
- Hãy cùng xem cách để bắt đầu sử dụng ***Axios*** trong code của bạn, và xem một vài tính năng đã góp phần làm cho nó thêm phổ biến đối với các lập trình viên Javascript.

2. Axios vs Fetch

Axios vs Fetch

- Có lẽ bạn cũng đã để ý rằng, các browser hiện đại bây giờ thường đi kèm và xây dựng sẵn các tính năng *Fetch API* mới hơn, vậy tạo sao ta lại không dùng nó luôn cho xong?
- Rất nhiều những sự khác biệt giữa **Axios** và **Fetch** khiến rất nhiều người vẫn lựa chọn **Axios** thay vì **Fetch**.
- Một trong những đặc điểm lớn ta có thể thấy đó là cách mà 2 thư viện xử lý với các *HTTP error code*.
- Khi sử dụng Fetch, nếu khi server trả về các mã lỗi ***4xx*** hay ***5xx***, thì hàm *catch()* của bạn sẽ không được gọi đến và người lập trình viên sẽ có nhiệm vụ phải tự kiểm tra trạng thái của mã trả về để xác định xem liệu *request* đó có thành công hay không.
- Trong khi đó, **Axios** sẽ reject tất cả các promise của request nếu một trong các mã lỗi trên được trả về.

Axios vs Fetch

- Một điểm khác biệt nho nhỏ khác, điều mà không ít các lập trình viên mới trong việc xây dựng API gặp phải đó là việc ***Fetch*** không tự động gửi trả *cookies* về cho server khi tạo một request.
- Ta sẽ cần phải truyền một cách trực tiếp các option để cho cookies có thể được include.
- Còn với ***Axios*** thì bạn không hề phải lo về vấn đề này.
- Sự khác biệt có thể được coi là vô cùng đáng chú ý đối với rất nhiều lập trình viên đó là khả năng cập nhật tiến độ của những lần uploads/downloads.

Vios vs Fetch

- Vì ***Axios*** được xây dựng dựa trên các *XHR API* cũ hơn, ta có thể khai báo các hàm callback cho *onUploadProgress* và *onDownloadProgress* để hiện thị phần trăm thành công tại giao diện cho app của bạn.
- Cho đến hiện tại thì ***Fetch*** vẫn chưa hỗ trợ gì cho việc này.

Vios vs Fetch

- Cuối cùng, ***Axios*** có thể được sử dụng ở cả *browser* và *Node.js* và chính điều này đã tạo cơ hội cho việc chia sẻ code *Javascript* giữa các *trình duyệt* và phần *back-end* hay việc thực hiện *render* cho ứng dụng front-end của bạn ở phía server.

3. Cài đặt Axios

Cài đặt Axios

- Cài đặt thông qua npm package

```
❯ npm install --save axios
```

- Sau đó include package axios vào trong code của ứng dụng

```
// ES2015 style import
import axios from 'axios';

// Node.js style require
const axios = require('axios');
```

- Hoặc có thể tích hợp thông CDN

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

Tạo một request với Axios

- Sử dụng giống như hàm **$**.*ajax()* của *jQuery*, có thể tạo bất kỳ một *request HTTP* bằng cách truyền vào các *object option* cho Axios

```
axios({
  method: 'post',
  url: '/login',
  data: {
    user: 'test',
    lastName: 'test1'
  }
});
```

Tạo một request với Axios

- Chúng ta cung cấp cho **Axios** biết rằng phương thức HTTP nào mà chúng ta muốn dùng
  - Ví dụ: *GET*/*POST*/*PUT*/*DELETE* và URL nào được sử dụng để gửi request lên.
- Đồng thời chúng ta cũng được cung cấp một số dữ liệu để có thể gửi kèm với request dưới dạng các Object Javascript đơn giản gồm các cặp key/value.
- Mặc định, **Axios** sẽ *serialize* các object này thành *JSON* và gắn chúng ở phần body của request.

Các option cho Request

Option	Ý nghĩa
baseURL	nếu bạn chỉ định một base URL, nó sẽ được đính vào trước bất cứ một URL tương đối nào mà bạn sử dụng.
headers	một object gồm các cặp key/value có thể gửi trong header của request.
params	một object gồm các cặp key/value mà sẽ được serialize và đính vào URL dưới dạng một query string.
responseType	nếu bạn mong chờ một format trả về khác ngoài JSON, thì bạn có thể set thuộc tính này thành *arraybuffer, blob, document, text* hay *stream*
auth	truyền vào một object gồm 2 trường username và password được sử dụng để cung cấp quyền truy cập cho các request yêu cầu HTTP Basic auth.

Các hàm tiện lợi

- Giống như **jQuery**, **Axios** cũng có những hàm ngắn để thực hiện các kiểu request khác nhau.
- Các hàm get, delete, head và options đều nhận vào 2 tham số là URL và một object option dùng để config request.

```
axios.get('/products/5');
```

Các hàm tiện lợi

- Các hàm *post, put và patch* nhận một object dữ liệu vào thành tham số thứ 2 của chúng và *object option* dùng để *config* là tham số thứ 3.

```
axios.post(
  "/products",
  {
    name: "Waffle Iron",
    price: 21.5
  },
  { options }
);
```

Nhận về một response

- Khi đã tạo ra một request, **Axios** trả về một promises mà sẽ resolve ngay cả khi kết quả trả về là một object response hay một object error.

```
axios.get('/product/9')
  .then(response ⇒ console.log(response))
  .catch(error ⇒ console.log(error));
```

Response Object

- Khi một request được thực hiện thành công, thì hàm then() của bạn sẽ nhận được một response object với các thuộc tính như sau:
  - ***data***: Payload được trả về từ server. Mặc định, Axios mong chờ JSON và sẽ parse nó trở thành một Javascript Object cho bạn.
  - ***status***: mã HTTP được trả về từ server.
  - ***statusText***: message của HTTP status được trả về bởi server.
  - ***headers***: tất cả các header được gửi về từ server.
  - ***config***: thiết lập ban đầu của request.
  - ***request***: một object XMLHttpRequest thực khi request được thực hiện ở trình duyệt.

Error Object

- Nếu có bất kỳ vấn đề gì xảy ra với request, promise sẽ bị reject với một error object chứa những thuộc tính sau đây.
  - ***message***: message của error ở dạng text.
  - ***response***: response object được trả về.
  - ***config***: thiết lập ban đầu của request.
  - ***request***: một object XMLHttpRequest thực khi request được thực hiện ở trình duyệt.

4. Transforms và Interceptors

Transforms

- **Axios** cung cấp các function để biến đổi dữ liệu đến và đi, dưới dạng 2 config option mà bạn có thể set khi thực hiện một request:
  - *transformRequest*
  - *transformResponse*
- Cả 2 thuộc tính đều là các mảng, giúp ta có thể tạo một luồng các function mà dữ liệu có thể đi qua.

Transforms

- Bất kỳ function nào được truyền vào transformRequest đều được ứng dụng vào cho các request PUT, POST và PATCH.
- Chúng nhận vào dữ liệu cần request và object header dưới dạng tham số và phải trả về một object đã được chỉnh sửa thay đổi.

Transforms

```
const options = {
  transformRequest: [
    (data, headers) => {
      // do something with data
      return data;
    }
  ]
}
```

Transforms

- Các function cũng có thể được thêm vào *transformResponse* tương tự như vậy, nhưng chỉ được gọi khi có response data từ server, và trước khi response được truyền qua bất cứ hàm *then()* nào.
- Vậy ta có thể sử dụng transform vào mục đích gì?
- Một trường hợp sử dụng đó là khi API mong chờ dữ liệu trả về dưới format cụ thể khác như XML hay kể cả CSV.
- Ta có thể set up một vài hàm transform để convert dữ liệu gửi đi và trả về.
- Mặc định, hàm transformRequest và *transformResponse* biến đổi dữ liệu thành JSON.

Interceptors

- Trong khi transform cho phép bạn điều chỉnh dữ liệu gửi đi và trả về, **Axios** cũng cho phép bạn thêm các function gọi là ***interceptors***.
- Giống transform, những function này có thể được đính vào để gọi đến khi một request được tạo ra, hay một response được trả về.

Interceptors

```
// Add a request interceptor
axios.interceptors.request.use((config) => {
  // Do something before request is sent
  return config;
}, (error) => {
  // Do something with request error
  return Promise.reject(error);
});

// Add a response interceptor
axios.interceptors.response.use((response) => {
  // Do something with response data
  return response;
}, (error) => {
  // Do something with response error
  return Promise.reject(error);
});
```

Interceptors

- Như bạn có thể thấy ở ví dụ trên, interceptor có những điểm khác biệt quan trọng so với transforms.
- Thay vì chỉ nhận dữ liệu hay headers, interceptor nhận vào toàn bộ config của request hay response object.
- Khi tạo một interceptor, bạn cũng có thể chọn cung cấp một hàm xử lý error cho phép bạn bắt và xử lý lỗi hợp lý.

Interceptors

- Request interceptors có thể được sử dụng để thực hiện những điều như: lấy token từ localStorage và gửi kèm các request.
- Response interceptor có thể được dùng để bắt lỗi 401 và redirect đến trang login trong việc xác thực.

Third-Party Add-Ons

- Là một thư viện phổ biến, **Axios** được hưởng lợi rất nhiều từ một hệ sinh thái các thư viện bên thứ 3 thừa hưởng từ tính năng của Axios.
  - ***axios-mock-adaptor***: cho phép bạn dễ dàng mock request để thực hiện việc kiểm tra code.
  - ***axios-cache-plugin***: một wrapper được dùng để lựa chọn lưu Cache các GET request.
  - ***redux-axios-middleware***: nếu bạn dùng Redux, middleware này cung cấp một cách gọn gàng để dispatch một Ajax request với các actions.
- Một danh sách các add-ons và extensions khác có sẵn trên Github.

5. Tích hợp Axios vào ứng dụng React

GET request

- Danh sách User

src > components > JS demo.js > ...

```
1  import React from 'react';
2  import axios from 'axios';
3
4  export default class PersonList extends React.Component {
5    state = {
6      persons: []
7    }
8
9    componentDidMount() {
10     axios.get(`https://jsonplaceholder.typicode.com/users`)
11       .then(res => {
12         const persons = res.data;
13         this.setState({ persons });
14       })
15       .catch(error => console.log(error));
16   }
17
18   render() {
19     return (
20       <ul>
21         { this.state.persons.map(person => <li>{person.name}</li>)}
22       </ul>
23     )
24   }
25 }
```

POST request

src > components > demo.js > ...

```
1  import React from "react";
2  import axios from "axios";
3
4  export default class PersonList extends React.Component {
5    state = {
6      name: "",
7    };
8
9    handleChange = (event) => {
10     this.setState({ name: event.target.value });
11   };
12
13   handleSubmit = (event) => {
14     event.preventDefault();
15
16     const user = {
17       name: this.state.name,
18     };
19
20     axios
21       .post(`https://jsonplaceholder.typicode.com/users`, { user })
22       .then((res) => {
23         console.log(res);
24         console.log(res.data);
25       });
26   };
27
28   render() {
29     return (
30       <div>
31         <form onSubmit={this.handleSubmit}>
32           <label>
33             Person Name:
34             <input type="text" name="name" onChange={this.handleChange} />
35           </label>
36           <button type="submit">Add</button>
37         </form>
38       </div>
39     );
40   }
41 }
```

DELETE request

src > components > demo.js > ...

```
1  import React from "react";
2  import axios from "axios";
3
4  export default class PersonList extends React.Component {
5    state = {
6      id: "",
7    };
8
9    handleChange = (event) => {
10     this.setState({ id: event.target.value });
11   };
12
13   handleSubmit = (event) => {
14     event.preventDefault();
15
16     axios
17       .delete(`https://jsonplaceholder.typicode.com/users/${this.state.id}`)
18       .then((res) => {
19         console.log(res);
20         console.log(res.data);
21       });
22   };
23
24   render() {
25     return (
26       <div>
27         <form onSubmit={this.handleSubmit}>
28           <label>
29             Person ID:
30             <input type="text" name="id" onChange={this.handleChange} />
31           </label>
32           <button type="submit">Delete</button>
33         </form>
34       </div>
35     );
36   }
37 }
```

Đối tượng response sẽ được trả về từ Server

```
{
  data: {}, // Dữ liệu cần lấy từ máy chủ

  status: 200, // Mã trạng thái HTTP của yêu cầu

  statusText: 'OK', // Mô tả trạng thái tương ứng với mã trạng thái ở trên

  headers: {}, // Thông tin header của hồi đáp (response)

  config: {}, // config được thiết lập trước khi gửi request

  request: {} // là thực thể của ClientRequest nếu sử dụng Node.js và XMLHttpRequest trong trình duyệt
}
```

Bài tập về nhà

- Xây dựng một ứng dụng React đơn giản để hiện thị thông tin sản phẩm gồm:
  - Mã sản phẩm
  - Tên sản phẩm
  - Giá
  - Tên hãng sản xuất
- Layout các bạn tự thiết kế hoặc hiển thị mặt định
- Thang điểm:
  - Xây dựng ứng dụng ExpressJS app để cung cấp các API của thoogn tin sản phẩm như trên (2 điểm)
  - Tạo được ứng dụng React và sử dụng Axios để load và hiển thị đầy đủ thông tin sản phẩm đầy đủ từ API được khởi tạo. (6 điểm)
  - Layout đẹp và hiển thị được nhiều Column (2 điểm)