TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

CÔNG NGHỆ PHẦN MỀM MỚI (MTSE431179)

AXIOS và Fetch API

THS. NGUYỄN HỮU TRUNG

Thông tin Giảng viên

- Ths. Nguyễn Hữu Trung
- Khoa Công Nghệ Thông Tin
- Trường Đại học Sư Phạm Kỹ Thuật TP.HCM
- 090.861.7108
- trungnh@hcmute.edu.vn
- https://www.youtube.com/@baigiai

Nội dung

- React.js là gì?
- Cài đặt React.js
- React.js render HTML
- React.js JSX
- React.js Component
- React.js props
- React.js props xử lý data

Axios với React

- Axios là một HTTP client được xây dựng dựa trên nền tảng Promise do đó nó kế thừa các ưu điểm của Promise. Cho phép thực hiện các hook (intercept) ngay khi gửi request và nhận response. Cho phép hủy yêu cầu, đây là một chức năng mà các thư viện khác không có.
- **Đặc điểm Axios**
  - Tạo XMLHttpRequests từ trình duyệt
  - Thực hiện các http request từ node.js
  - Hỗ trợ Promise API
  - Chặn request và response
  - Chuyển đổi dữ liệu request và response
  - Hủy requests
  - Tự động chuyển đổi về dữ liệu JSON
  - Hỗ trợ phía client để chống lại XSRF

https://axios-http.com/vi/docs/intro

Axios với React

- **Cài đặt**
  - **npm install --save axios**
- **Một request với Axios**
  - Giống như với hàm $.ajax của jQuery, Chúng ta có thể tạo bất kỳ một request HTTP nào bằng cách truyền vào các object option cho Axios, ví dụ như:

```
// Import thư viện axios
      import axios from 'axios';
// Tạo một yêu cầu GET đến một URL cụ thể
    axios.get('http://iotstar.vn/data')
                .then(function(response) {
                        // Xử lý phản hồi từ máy chủ
                        console.log(response.data); })
               .catch(function(error) {
                        // Xử lý lỗi
                        console.error(error);
                }
     );
```

Axios với React

- **Hỗ trợ Promise API trong Axios:** Promise API được hiểu là một tập hợp các phương thức và tính năng liên quan đến Promises trong JavaScript. Promises là một cơ chế xử lý bất đồng bộ được sử dụng để xử lý các hoạt động mà cần thời gian để hoàn thành, chẳng hạn như các yêu cầu HTTP, đọc/ghi vào tệp, hoặc tương tác với cơ sở dữ liệu.

```
import axios from "axios"

function makeRequest(url) {
  return axios.get(url)
    .then(function (response) {
      return response.data;
    })
    .catch(function (error) {
      throw new Error('Request failed: ' + error.response.status);
    });
}

// Usage
makeRequest('https://api.example.com/data')
  .then(function (responseData) {
    // Process responseData
  })
  .catch(function (error) {
    // Handle errors
  });
```

Axios với React

- **Huỷ Request (Cancellation) trong Axios:** Trong một số ít trường hợp như kết nối mạng bị mất trong lúc Axios gửi request, nếu ta không huỷ request đó, nó sẽ được đưa vào trạng thái chờ hoặc call lại liên tục đến server cho đến khi nhận được response trả về. Nếu việc này diễn ra thường xuyên, server sẽ bị quá tải hoặc dẫn đến tràn bộ nhớ.
- Axios đã cung cấp sẵn cho chúng ta phương thức **signal** để giải quyết vấn đề này. Để sử dụng signal, chúng ta sẽ sử dụng thêm **AbortController** được cung cấp bởi WebAPI. AbortController đã có sẵn trong trình duyệt vậy nên chúng ta không cần import từ thư viện khác bên ngoài.

```
1  function newAbortSignal(timeoutMs) {
2    const abortController = new AbortController();
3    setTimeout(() => abortController.abort(), timeoutMs || 0);
4
5    return abortController.signal;
6  }
7
8  axios.get('/foo/bar', {
9     signal: newAbortSignal(5000) // Request sẽ được huỷ sau 5 giây
10 }).then(function(response) {
11    //...
12 });
```

Axios với React

- **Huỷ Request (Cancellation) trong Axios:** Ngoài việc sử dụng **AbortController**, chúng ta cũng có thể sử dụng một API khác là **AbortSignal.timeout()**. Tuy nhiên API này chỉ sử dụng được khi ứng dụng web của bạn đang sử dụng NodeJS 17.3 trở lên .

```
axios.get('/foo/bar', {
    signal: AbortSignal.timeout(5000) //Huỷ request sau 5 giây
}).then(function(response) {
    //...
});
```

Axios với React

- Xử lý lỗi (Error Handling) trong Axios
  - Xử lý lỗi HTTP

```
axios.get('https://api.example.com/data')
  .then(function (response) {
    // Xử lý phản hồi thành công
  })
  .catch(function (error) {
    if (error.response) {
      // Lỗi HTTP, ví dụ: error.response.status
      console.log('Lỗi HTTP:', error.response.status);
    } else if (error.request) {
      // Yêu cầu đã được gửi nhưng không nhận được phản hồi
      console.log('Yêu cầu không được phản hồi:', error.request);
    } else {
      // Lỗi khác
      console.log('Lỗi:', error.message);
    }
  });
```

Axios với React

- Xử lý lỗi (Error Handling) trong Axios
  - Xử lý lỗi mạng

```
axios.get('https://api.example.com/data')
  .then(function (response) {
    // Xử lý phản hồi thành công
  })
  .catch(function (error) {
    if (error.code === 'ECONNABORTED') {
      // Lỗi timeout, ví dụ: yêu cầu mất quá nhiều thời gian để hoàn thành
      console.log('Timeout:', error.message);
    } else {
      // Lỗi mạng khác
      console.log('Lỗi mạng:', error.message);
    }
  });
```

Axios với React

- Xử lý lỗi (Error Handling) trong Axios
  - Xử lý lỗi do server tự định nghĩa

```
1  axios.get('https://api.example.com/data')
2    .then(function (response) {
3      // Xử lý phản hồi thành công
4    })
5    .catch(function (error) {
6      if (error.message === 'CustomError') {
7        // Xử lý lỗi tự định nghĩa
8        console.log('Lỗi tự định nghĩa');
9      } else {
10       // Xử lý các loại lỗi khác
11       console.log('Lỗi:', error.message);
12     }
13   });
```

Axios với React

- **Tự động chuyển đổi dữ liệu:** Trong phần cấu hình Axios, Axios đã cung cấp sẵn cho chúng ta 2 lựa chọn để chuyển đổi dữ liệu:
  - **transformResponse**: cho phép bạn chuyển đổi dữ liệu từ response trước khi nó được trả về cho bạn
  - **transformRequest**: cho phép bạn chuyển đổi dữ liệu trước khi gửi nó đi

```
import axios from 'axios';

// Tạo một instance Axios với cấu hình chuyển đổi dữ liệu
const instance = axios.create({
  baseURL: 'https://api.example.com', // Thay thế bằng URL cụ thể của bạn
  transformResponse: [function (data) {
    // Chuyển đổi dữ liệu từ phản hồi trước khi nó được trả về
    // Ví dụ: Chuyển đổi dữ liệu JSON thành đối tượng JavaScript
    return JSON.parse(data);
  }],
  transformRequest: [function (data) {
    // Chuyển đổi dữ liệu trước khi gửi nó đi
    // Ví dụ: Chuyển đổi đối tượng JavaScript thành dữ liệu JSON
    return JSON.stringify(data);
  }],
});

// Sử dụng instance Axios với cấu hình chuyển đổi dữ liệu
instance.get('/data')
  .then(function (response) {
    // Dữ liệu đã được chuyển đổi từ phản hồi
    console.log(response.data);
  })
  .catch(function (error) {
    console.error('Lỗi:', error);
  });
```

Axios với React

- **Interceptors trong Axios:** Interceptors trong Axios là một tính năng mạnh mẽ cho phép bạn can thiệp vào quy trình gửi và nhận yêu cầu HTTP trước và sau khi chúng được gửi. Bạn có thể sử dụng interceptors để thực hiện các tác vụ như thêm tiêu đề, xử lý lỗi, thêm hoặc xóa thông tin từ yêu cầu và phản hồi, và nhiều tác vụ khác
- ***Axios hỗ trợ hai loại interceptor chính:***
  - **Request Interceptors**: Được gọi trước khi yêu cầu được gửi đi. Bạn có thể sử dụng chúng để thêm tiêu đề, thêm token xác thực, hoặc xử lý dữ liệu yêu cầu trước khi nó được gửi.
  - **Response Interceptors**: Được gọi sau khi yêu cầu đã được gửi và phản hồi đã được nhận. Bạn có thể sử dụng chúng để xử lý dữ liệu phản hồi, xử lý lỗi, và thực hiện các tác vụ khác trên phản hồi.

Axios với React

Interceptors trong Axios:

```
import axios from 'axios';

// Tạo một instance Axios
const instance = axios.create({
  baseURL: 'https://api.example.com', // Thay thế bằng URL cụ thể của bạn
});

// Request Interceptor: Trước khi yêu cầu được gửi đi
instance.interceptors.request.use(function (config) {
  // Thực hiện các tác vụ trước khi gửi yêu cầu
  // Ví dụ: Thêm tiêu đề xác thực vào yêu cầu
  config.headers.Authorization = 'Bearer token123';
  return config;
}, function (error) {
  // Xử lý lỗi request
  return Promise.reject(error);
});

// Response Interceptor: Sau khi nhận phản hồi
instance.interceptors.response.use(function (response) {
  // Thực hiện các tác vụ sau khi nhận phản hồi
  // Ví dụ: Xử lý dữ liệu phản hồi
  return response;
}, function (error) {
  // Xử lý lỗi response
  return Promise.reject(error);
});
```

```

// Sử dụng instance Axios đã tạo
instance.get('/data')
  .then(function (response) {
    // Xử lý dữ liệu phản hồi
    console.log(response.data);
  })
  .catch(function (error) {
    // Xử lý lỗi
    console.error('Lỗi:', error);
  });
```

Axios với React

- **GET Requests: URL**: Là địa chỉ của server mà bạn muốn gửi yêu cầu lên. Nó là đối số đầu tiên của phương thức **.get(). Options (Tuỳ chọn):** Các tuỳ chọn này bao gồm:
  - params: Một đối tượng chứa các tham số truy vấn (query parameters) bạn muốn gửi cùng với yêu cầu GET. Ví dụ: { page: 1, limit: 10 }
  - headers: Đối tượng chứa các tiêu đề HTTP bạn muốn gửi cùng với yêu cầu GET. Ví dụ: { Authorization: 'Bearer token123' } để gửi headers xác thực.
  - timeout: Thời gian tối đa cho một yêu cầu GET. Nếu yêu cầu không hoàn thành trong khoảng thời gian này, nó sẽ bị hủy.
  - Và nhiều tuỳ chọn khác, ví dụ: auth, responseType,...
  - **Promise Handling:** Sau khi bạn thực hiện GET request bằng Axios, nó sẽ trả về một promise. Bạn có thể sử dụng .then() để xử lý kết quả thành công và .catch() để xử lý lỗi nếu có. Dữ liệu phản hồi thường được truy cập qua response.data.

Axios với React

- **GET Requests:**

```
1  import axios from 'axios';
2
3  // Thực hiện GET request đơn giản
4  axios.get('https://api.example.com/data')
5    .then(function (response) {
6      // Xử lý dữ liệu phản hồi ở đây
7      const data = response.data;
8      console.log(data);
9    })
10   .catch(function (error) {
11     // Xử lý lỗi ở đây
12     console.error('Lỗi:', error);
13   });
```

- **POST Requests:**
  - Chúng ta đã sử dụng **axios.post()** để thực hiện một POST request đến URL 'https://api.example.com/login'.
  - Chúng ta đã tạo đối tượng postData chứa dữ liệu mà chúng ta muốn gửi đi trong yêu cầu POST. Đây có thể là một đối tượng JavaScript, chuỗi JSON hoặc dữ liệu bất kỳ mà bạn muốn gửi.
  - Trong axios.post(), chúng ta truyền postData làm tham số thứ hai. Axios sẽ tự động chuyển đổi nó thành dữ liệu gửi đi trong yêu cầu. (hay còn gọi là thêm body)
  - Khi POST request được thực hiện, chúng ta sử dụng .then() để xử lý kết quả thành công (ví dụ: dữ liệu phản hồi sau khi đăng nhập thành công) và .catch() để xử lý lỗi hoặc xử lý dữ liệu phản hồi sau khi đăng nhập thất bại

❑ POST Requests:

```
import axios from 'axios';

// Dữ liệu bạn muốn gửi đi trong request (ví dụ: đối tượng JavaScript)
const postData = {
  username: 'user123',
  password: 'pass456',
};

// Token xác thực của bạn
const authToken = 'Bearer your_auth_token_here';

// Tạo một đối tượng tiêu đề (headers) để truyền Authorization Token
const headers = {
  'Authorization': authToken,
};

// Thực hiện POST request với tiêu đề Authorization
axios.post('https://api.example.com/login', postData, { headers: headers })
  .then(function (response) {
    // Xử lý dữ liệu phản hồi sau khi đăng nhập thành công
    const responseData = response.data;
    console.log(responseData);
  })
  .catch(function (error) {
    // Xử lý lỗi hoặc xử lý dữ liệu phản hồi sau khi đăng nhập thất bại
    console.error('Lỗi:', error);
  });
```

Axios với React

- **POST Requests**

axios.post(`http://iotstar.vn/users`, { user }) .then(res => { console.log(res); console.log(res.data); })

```
render() {
  return (
    <div>
      <form onSubmit={this.handleSubmit}>
        <label>
          Person Name:
          <input type="text" name="name" onChange={this.handleChange} />
        </label>
        <button type="submit">Add</button>
      </form>
    </div>
  )
  }
}
```

```
import React from 'react';
import axios from 'axios';

export default class PersonList extends React.Component {
  state = {
    name: '',
  }

  handleChange = event => {
    this.setState({ name: event.target.value });
  }

  handleSubmit = event => {
    event.preventDefault();

    const user = {
      name: this.state.name
    };

    axios.post(`https://jsonplaceholder.typicode.com/users`, { user })
      .then(res => {
        console.log(res);
        console.log(res.data);
      })
  }
```

Axios với React

- **PUT Request:** gửi dữ liệu cập nhật lên server

```
import axios from 'axios';

// Dữ liệu bạn muốn gửi để cập nhật tài nguyên (ví dụ: đối tượng JavaScript)
const updateData = {
  name: 'New Name',
  description: 'Updated description',
};

// Token xác thực của bạn
const authToken = 'Bearer your_auth_token_here';

// Tạo một đối tượng tiêu đề (headers) để truyền Authorization Token
const headers = {
  'Authorization': authToken,
};

// Thực hiện PUT request với tiêu đề Authorization
axios.put('https://api.example.com/resource/123', updateData, { headers: headers })
  .then(function (response) {
    // Xử lý dữ liệu phản hồi sau khi cập nhật thành công
    const responseData = response.data;
    console.log(responseData);
  })
  .catch(function (error) {
    // Xử lý lỗi hoặc xử lý dữ liệu phản hồi sau khi cập nhật thất bại
    console.error('Lỗi:', error);
  });
```

Axios với React

- **DELETE Requests**

axios.delete(`http://iotstar.vn/${this.state.id}`) .then(res => { console.log(res); console.log(res.data); })

```
import React from 'react';
import axios from 'axios';

export default class PersonList extends React.Component {
  state = {
    id: '',
  }

  handleChange = event => {
    this.setState({ id: event.target.value });
  }

  handleSubmit = event => {
    event.preventDefault();

    axios.delete(`https://jsonplaceholder.typicode.com/users/${this.state.id}`)
      .then(res => {
        console.log(res);
        console.log(res.data);
      })
  }
```

```
  }

  render() {
    return (
      <div>
        <form onSubmit={this.handleSubmit}>
          <label>
            Person ID:
            <input type="text" name="id" onChange={this.handleChange} />
          </label>
          <button type="submit">Delete</button>
        </form>
      </div>
    )
  }
}
```

DELETE Requests

```
1  import axios from 'axios';
2
3  // Token xác thực của bạn
4  const authToken = 'Bearer your_auth_token_here';
5
6  // Tạo một đối tượng tiêu đề (headers) để truyền Authorization Token
7  const headers = {
8    'Authorization': authToken,
9  };
10
11 // Thực hiện DELETE request với tiêu đề Authorization
12 axios.delete('https://api.example.com/resource/123', { headers: headers })
13   .then(function (response) {
14     // Xử lý dữ liệu phản hồi sau khi xóa thành công
15     console.log('Xóa thành công');
16   })
17   .catch(function (error) {
18     // Xử lý lỗi hoặc xử lý dữ liệu phản hồi sau khi xóa thất bại
19     console.error('Lỗi:', error);
20   });
```

Axios với React

- **Cấu hình Interceptors:** Một ứng dụng của Interceptors thường thấy nhất là xử lý trường hợp gửi Request bị lỗi. Giả sử bạn muốn đăng xuất người dùng ra khỏi website khi bất cứ việc gửi Request nào trong ứng dụng React của bạn bị lỗi

```
import axios from 'axios';

// Tạo một instance Axios
const instance = axios.create({
  baseURL: 'https://api.example.com', // Thay thế bằng URL cụ thể của bạn
});

// Intercept cho yêu cầu
instance.interceptors.request.use(function (config) {
  return config;
}, function (error) {
  return Promise.reject(error);
});

// Intercept cho phản hồi
instance.interceptors.response.use(function (response) {
  // Thực hiện xử lý dữ liệu phản hồi ở đây
  return response;
}, function (error) {
  // Xử lý lỗi phản hồi
  if (error.response.status === 401) {
    // Gọi API đăng xuất nếu trả về mã trạng thái 401
    // Thay thế 'logout' bằng URL của API đăng xuất thực tế
    return instance.post('/logout')
      .then(function () {
        // Sau khi đăng xuất thành công, bạn có thể thực hiện các tác vụ khác
        // ví dụ: chuyển hướng người dùng đến trang đăng nhập
      })
      .catch(function (logoutError) {
        // Xử lý lỗi khi gọi API đăng xuất
        return Promise.reject(logoutError);
      });
  } else {
    return Promise.reject(error);
  }
});
```

Axios với React

- Ngoài ra còn có rất nhiều các option mới cho request, nhưng dưới đây là một số option phổ biến nhất:
  - **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
- Ví dụ đối với option **baseURL**: Ta tạo 1 file mới và đặt tên là api.js.

```
import axios from 'axios';
export default axios.create({
        baseURL: `http://iotstar.vn/`
});
```

```
import API from '../api';

export default class PersonList extends React.Component {
  handleSubmit = event => {
    event.preventDefault();

    API.delete(`users/${this.state.id}`)
      .then(res => {
        console.log(res);
        console.log(res.data);
      })
  }
}
```

Axios với React

- **Đối tượng response được trả về từ server**

```
{
    // Dữ liệu cần lấy từ máy chủ
    data: {},
     // Mã trạng thái HTTP của yêu cầu
    status: 200,
    // Mô tả trạng thái tương ứng với mã trạng thái ở trên
    statusText: 'OK',
    // Thông tin header của hồi đáp (response)
    headers: {},
    // config được thiết lập trước khi gửi request
    config: {},
    // là thực thể của ClientRequest nếu sử dụng Node.js và
    XMLHttpRequest trong trình duyệt
    request: {}
}
```

Hướng dẫn sử dụng Axios

- **Cấu hình Axios trong ReactJS:** Để sử dụng Axios trong dự án ReactJS, trước tiên chúng ta cần tạo một dự án React. Ở đây mình sẽ sử dụng build tool là ViteJS để tạo một dự án React nhanh chóng
  - **npm create vite@latest**
  - Sau đó nhập tên Project muốn tạo và chọn Framework là **React**
  - Tiếp đến chúng ta cài đặt axios từ thư viện npm: **npm i axios**
    - Để sử dụng axios thì ta import thư viện vào component:

import axios from "axios"

Tạo một Axios Instance

```
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
[4/4] 🔨  Building fresh packages...
success Installed "create-vite@4.4.1" with binaries:
      - create-vite
      - cva
✔ Project name: … vite-project
? Select a framework: › - Use arrow-keys. Return to submit.
    Vanilla
    Vue
❯   React
    Preact
    Lit
    Svelte
    Solid
    Qwik
    Others
```

Hướng dẫn sử dụng Axios

- **Cấu hình Axios trong ReactJS:**
  - **Tạo một Axios Instance**

```
1  import axios from 'axios';
2
3  // Tạo một instance Axios
4  const instance = axios.create({
5    baseURL: 'https://api.example.com', // Thay thế bằng URL cụ thể của bạn
6  });
7
8  // Sử dụng instance để thực hiện GET request
9  instance.get('/data')
10   .then(function (response) {
11     // Xử lý dữ liệu phản hồi ở đây
12     console.log(response.data);
13   })
14   .catch(function (error) {
15     // Xử lý lỗi ở đây
16     console.error('Lỗi:', error);
17   });
```

So sánh Fetch API và Axios

- **Điểm giống nhau giữa Fetch API và Axios**
  - **HTTP Requests**: Cả Fetch API và Axios đều được sử dụng để thực hiện yêu cầu HTTP, bao gồm GET, POST, PUT, DELETE và các loại yêu cầu khác.
  - **Promise-Based**: Cả hai cung cấp hỗ trợ Promise, cho phép xử lý yêu cầu và phản hồi dễ dàng bằng .then() và .catch().
  - **CORS (Cross-Origin Resource Sharing)**: Cả Fetch API và Axios đều hỗ trợ việc xử lý CORS cho yêu cầu từ một nguồn khác.
  - **Chuyển đổi Dữ liệu**: Cả hai cho phép bạn chuyển đổi dữ liệu yêu cầu và phản hồi, bao gồm chuyển đổi JSON thành đối tượng JavaScript.

So sánh Fetch API và Axios

- **Điểm khác nhau giữa Fetch API và Axios**
  - **Khả năng Tích hợp**: Fetch API là một phần của tiêu chuẩn JavaScript và được tích hợp sẵn trong các trình duyệt hiện đại. Axios cần được cài đặt bổ sung thông qua npm hoặc yarn.
  - **Xử lý JSON Tự động**: Axios tự động chuyển đổi dữ liệu JSON phản hồi thành đối tượng JavaScript, trong khi Fetch API yêu cầu bạn gọi .json() trên đối tượng phản hồi để chuyển đổi nó.
  - **Xử lý Lỗi**: Axios cung cấp cách xử lý lỗi dễ đọc hơn và có thêm các tính năng như interceptors để xử lý lỗi. Fetch API trả về Promise khiến việc xử lý lỗi HTTP trở nên phức tạp hơn và đòi hỏi kiểm tra trạng thái của phản hồi.
  - **Interceptors**: Axios hỗ trợ interceptors, cho phép bạn can thiệp vào quy trình gửi và nhận yêu cầu HTTP. Fetch API không có interceptors tích hợp.

So sánh Fetch API và Axios

- **Điểm khác nhau giữa Fetch API và Axios**
  - **Browser Compatibility**: Fetch API được tích hợp trong các trình duyệt hiện đại, trong khi Axios cung cấp tính nhất quán trên nhiều trình duyệt khác nhau và phiên bản.
  - **Xử lý Trạng Thái HTTP**: Axios thường cung cấp một cách dễ dàng hơn để kiểm tra và xử lý trạng thái HTTP cụ thể, bao gồm 401 (Unauthorized) và các trạng thái khác.
  - **Khả năng Hủy Request**: Axios hỗ trợ hủy các yêu cầu HTTP bằng cách sử dụng Cancel Token, trong khi Fetch API không có tính năng hủy yêu cầu tích hợp.
  - **Plugin**: Axios hỗ trợ cài đặt các plugin mở rộng để thực hiện các tác vụ khác nhau, trong khi Fetch API yêu cầu viết mã tùy chỉnh nếu bạn muốn mở rộng chức năng của nó.