VIETNAM AUSTRALIA Vocational College

React Advance
Bài 6: Sử dụng AJAX kết nối API

daynghevietuc.com

Nội dung

1. Giới thiệu AJAX
2. Sử dung AJAX kết nối API

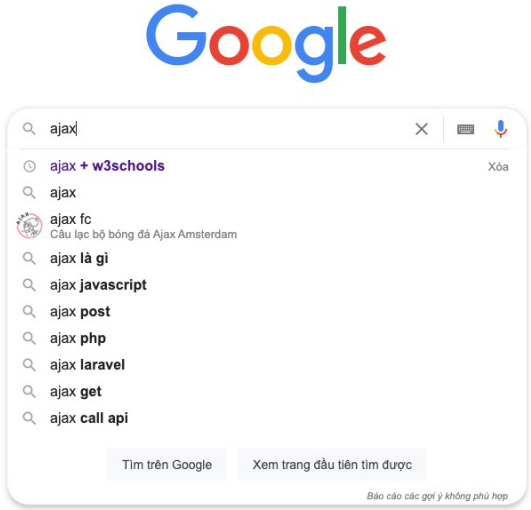
1. Giới thiệu AJAX

Giới thiệu AJAX

- **AJAX** là chữ viết tắt của **Asynchronous JavaScript and XML**
- Đây là một công nghệ giúp chung ta tạo ra những Web động mà hoàn toàn không reload lại trang nên rất mượt và đẹp.
- Vậy ***Asynchronous, JavaScript, XML*** trong từ **AJAX** là gì
  - **Asynchronous** (***Async)*** – *bất đồng bộ*. Bất đồng bộ có nghĩa là một chương trình có thể xử lý không theo tuần tự các hàm. Sẽ không có quy trình, có thể nhảy đi bỏ qua bước nào đó. Ích lợi dễ thấy nhất của bất đồng bộ là chương trình có thể xử lý nhiều công việc một lúc.
  - **JavaScript** là một ngôn ngữ lập trình nổi tiếng. Trong số rất nhiều chức năng của nó là khả năng quản lý nội dung động của website và hỗ trợ tương tác với người dùng.
  - **XML** là một dạng của ngôn ngữ **markup** như **HTML**, chữ đầy đủ của nó là ***eXtensible Markup Language***. Nếu HTML được dùng để hiển thị dữ liệu, XML được thiết kế để chứa dữ liệu.

Giới thiệu AJAX

- Ví dụ thực tế của AJAX
  - Ví dụ như chức năng tự động hoàn thiện từ khóa của Google Search. Nó dự đoán giúp bạn và hoàn thiện tự khóa trong quá trình gõ. Từ khóa thay đổi theo thời gian thực nhưng trang web Google vẫn giữ nguyên trạng thái không load lại trang.
  - AJAX giúp việc trao đổi dữ liệu nội bộ và presentation layer hoạt động đồng thời. Tuy nhiên không ảnh hưởng đến chức năng của nhau



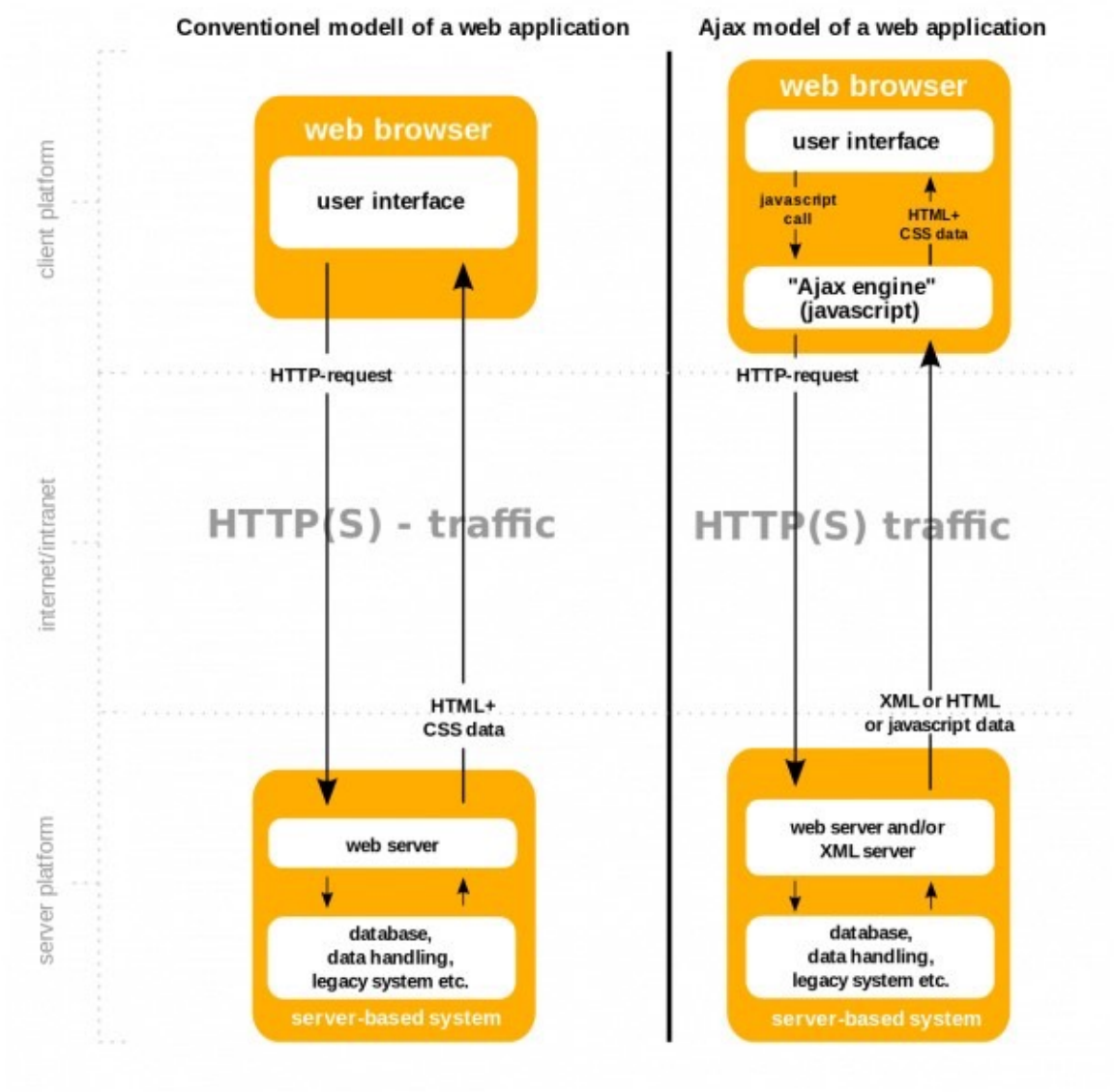
Giới thiệu AJAX

- Các ứng dụng phổ biến của AJAX
  - Hệ thống đánh giá và xếp hạng về sản phẩm - khi mua hàng online. Khi click vào nút đánh giá hay bình chọn, website sẽ nhận kết quả nhưng website không load lại trang.
  - Chat rooms – một số website tích hợp chat vào trang của họ để hỗ trợ chức năng tư vấn tự động với nhân viên bán hàng.
  - Thông báo Trending của Twitter để cập nhật các tin tức mới. Mỗi lần có tweet mới trong các chủ đề nóng, thì Twitter sẽ cập nhật thông tin mới mà không cần load lại toàn page.
  - Tương tự, Facebook cũng sử dụng công nghệ AJAX để thực hiện nhật new-feed hay các chức năng khác của mình.

AJAX hoạt động như thế nào

- Cần lưu ý AJAX không phải dùng một công nghệ duy nhất, cũng không phải ngôn ngữ lập trình.
- AJAX là một bộ kỹ thuật phát triển web. Bộ hệ thống này bao gồm:
  - **HTML/XHTML** làm ngôn ngữ chính và CSS để tạo phong cách.
  - **The Document Object Model (DOM)** để hiển thị dữ liệu động và tạo tương tác.
  - **XML** để trao đổi dự liệu nội bộ và XSLT để xử lý nó. Nhiều lập trình viên đã thay thế bằng **JSON** vì nó gần với **JavaScript** hơn.
  - **XMLHttpRequest** object để giao tiếp bất đồng bộ.
  - Cuối cùng, **JavaScript** làm ngôn ngữ lập trình để kết nối toàn bộ các công nghệ trên lại.

Sơ đồ hoạt động của AJAX



So sánh giữa mô hình hoạt thông thường với mô hình AJAX

#	Mô hình thông thường	Mô hình AJAX
1	HTTP được gửi từ trình duyệt lên máy chủ.	Trình duyệt tạo một lệnh gọi JavaScript để kích hoạt ***XMLHttpRequest***.
2	Máy chủ nhận, sau đó phản truy xuất thông tin.	Ở dưới nền, trình duyệt tạo một yêu cầu HTTP gửi lên server.
3	Server gửi dữ liệu được yêu cầu lại cho trình duyệt.	Server tiếp nhận, truy xuất và gửi lại dữ liệu cho trình duyệt.
4	Trình duyệt nhận dữ liệu và tải lại trang để hiển thị dữ liệu lên. Người dùng phải đợi kết thúc quá trình, điều này gây tốn thời gian và làm tăng tải lượng lên server.	Trình duyệt nhận dữ liệu từ server và ngay lập tức hiển thị lên trang. Không cần tải lại toàn bộ trang.

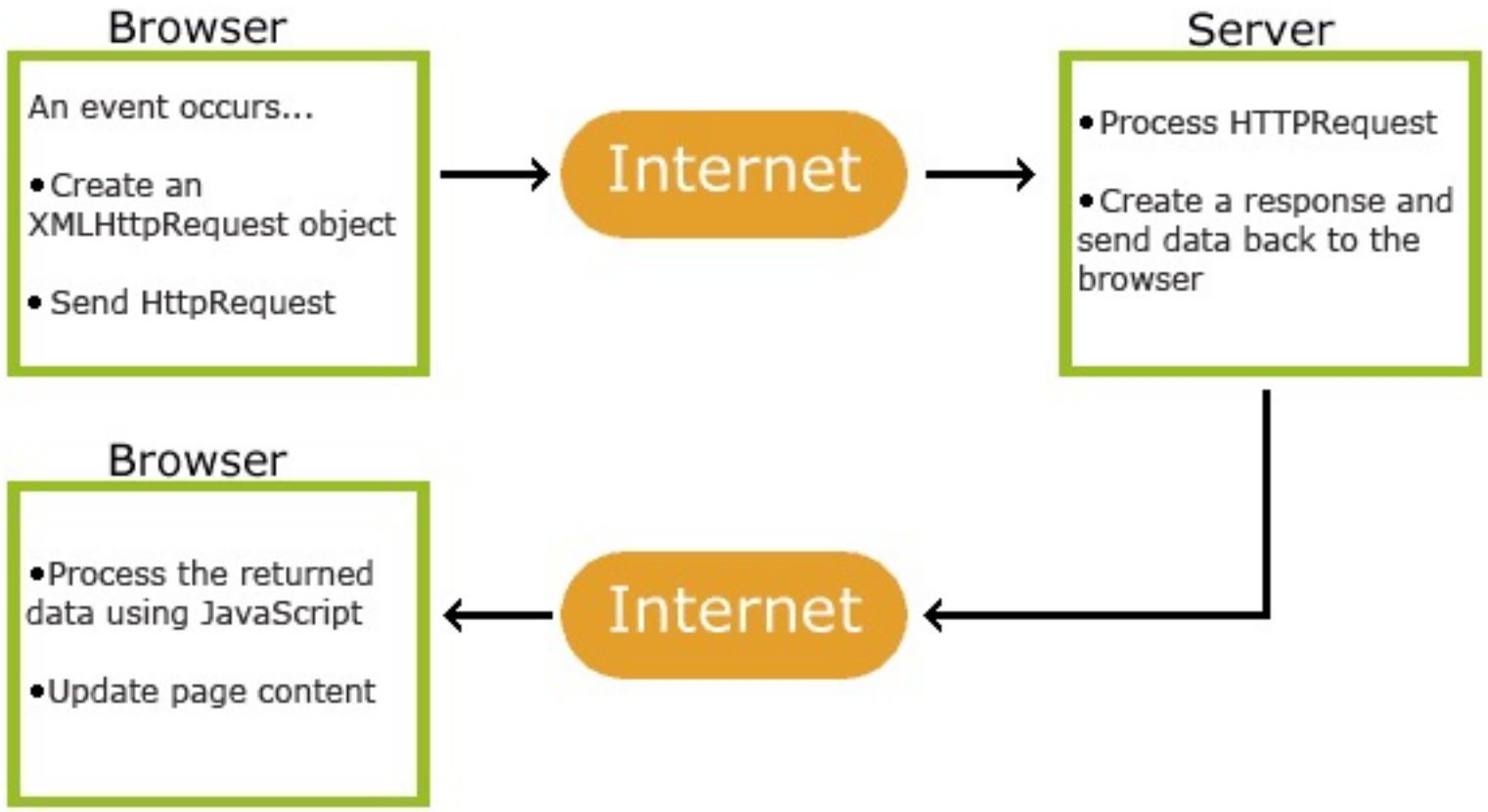
Tại sao nên dùng AJAX

- Có 4 lợi ích chính của việc sử dụng Ajax, cụ thể là:
  - **Callbacks**: Ajax được sử dụng để thực hiện một cuộc gọi lại. **AJAX** thực hiện việc truy xuất và / hoặc lưu dữ liệu mà không gửi toàn bộ trang trở lại máy chủ. Bằng cách gửi lại một phần trang web đến máy chủ, việc sử dụng mạng được giảm thiểu và các hoạt động diễn ra nhanh hơn. Trong các trang web băng thông hạn chế, điều này có thể cải thiện đáng kể hiệu suất mạng. Dữ liệu được gửi đến và đi từ máy chủ một cách tối thiểu.
  - **Thực hiện các cuộc gọi không đồng bộ**: Ajax cho phép bạn thực hiện các cuộc gọi không đồng bộ đến một máy chủ web. Điều này cho phép trình duyệt của người dùng tránh phải chờ tất cả dữ liệu đến trước khi cho phép người dùng hành động một lần nữa.
  - **Thân thiện với người dùng**: Vì không phải post lại trang lên server, các ứng dụng hỗ trợ Ajax sẽ luôn nhanh hơn và thân thiện với người dùng hơn.
  - **Tăng tốc độ**: Mục đích chính của Ajax là **cải thiện tốc độ**, **hiệu suất** và **khả năng sử dụng của một ứng dụng web**. Một ví dụ tuyệt vời của Ajax là tính năng xếp hạng phim trên Netflix. Người dùng đánh giá một bộ phim và xếp hạng cá nhân của họ cho bộ phim đó sẽ được lưu vào cơ sở dữ liệu của họ mà không cần chờ trang làm mới hoặc tải lại

AJAX nên sử dụng ở đâu?

- **Ajax nên được sử dụng ở bất cứ nơi nào trong một ứng dụng web**, nơi một lượng nhỏ thông tin có thể được lưu hoặc lấy ra từ máy chủ mà không cần tải lại toàn bộ trang web.
- Ví dụ, với một trang web bán hàng, khi người dùng chọn thành phố giao hàng giao hàng một hộp thoại dropdown được tải lại và chỉ chứa giá trị là tên các huyện của thành phố đã được chọn.

Cơ chế hoạt động của AJAX



Cơ chế hoạt động của AJAX

1. Phát sinh sự kiện trong trang web như Page load, button click
2. Khởi tạo đối tượng XMLHttpRequest bằng JavaScript
3. Đối tượng XMLHttpRequest gửi một request đến Server
4. Server xử lý request
5. Server sẽ gửi response (phản hồi) về trang Web.
6. Response được đọc bởi JavaScript.
7. JavaScript sẽ thực hiện render lại (cấu trúc HTML DOM) để cập nhật nội dung trên Website

Khởi tạo đối tượng XMLHttpRequest

- Đối với các tình duyệt mới (Chrome, FireFox, IE7+, Edge, Safari, Opera) đều hỗ trợ sẵn đối tượng XMLHttpRequest

```
var xhttp = new XMLHttpRequest();
```

- Đối với một số trình duyệt cũ thì phải sử dụng đối tượng ActiveXObject thay thế cho XMLHttpRequest

```
if (window.XMLHttpRequest) {
    // code for modern browsers
    xmlhttp = new XMLHttpRequest();
} else {
    // code for old IE browsers
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Access across domain

- Vì lý do bảo mật, các trình duyệt hiện đại không cho pháp truy cập vào các trang có tên miền khác nhau.
- Điều này có nghĩa là các trang web và file xml phải được đặt trên cùng một máy chủ và cùng tên miền.
- Các ví dụ trên W3School, tất cả các file xml đều nằm trong tên miền W3School.
- Nếu muốn sử dụng các ví dụ này các trang web và file xml phải được nằm trên cùng máy chủ và tên miền của bạn.

XMLHttpRequest object methods

Phương thức (method)	Ý nghĩa
new XMLHttpRequest()	Khởi tạo đối tượng XMLHttpRequest
abort()	Hủy (cancel) request hiện tại
getAllResponseHeader()	Trả về thông tin của Header
getResponseHeader()	Trả về thông tin của một Header cụ thể
open(method, url, async, user, psw)	Định nghĩa request • ***method***: phương thức request GET hay POST • ***url***: định danh endpoint đến server • ***async***: true (asynchronous) hay false (synchronous) • ***user***: là username (option) • ***psw***: là password (option)
send()	Gửi request về server – dành cho method GET
send(string)	Gửi request về server – dành cho method POST
setRequestHeader()	Thêm một lable/value cho header và gửi đi

XMLHttpRequest object properties

Propery (thuộc tính)	Ý nghĩa
onreadysatechange	Định nghĩa hàm được gọi khi thuộc tính readState thay đổi giá trị
readyState	Giữ trạng thái củ đối tượng XMLHttpRequest • 0 – request chưa được khởi tạo • 1 – kết nối server được thiết lập • 2 – request đã được server tiếp nhận • 3 – server đang xử lý request • 4 – hoàn thành xử lý và phản hồi về cho client
responseText	Trả về response data dưới dạng chuỗi (String)
responseXML	Trả về response data dưới dạng XML
status	Trạng thái của request • 200 – OK • 403 – Lỗi “Forbidden” • 404 – Lỗi “Not Found”
statusText	Trả về status-text ví dụ “OK” hay “Not Found)

Gửi request tới server

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Method (phương thức)	Ý nghĩa
open(method, url, async)	Đặc tả loại request • ***method***: loại request là POST hay GET • ***url***: địa chỉ server • ***async***: true (asynchronous) hay flase (synchronous)
send()	Gửi request tới server (GET)
send(string)	Gửi request tới server (POST

GET hay POST

- Phương thức GET đơng giản và nhanh hơn POST
- Tuy nhiên phương thức POST được sử dụng khi
  - Cache file – cần lưu một file hay database trên server
  - Gửi một lượng dữ lieu lớn lên server
  - Gửi một user input lên server
    - Nội dung có chứa các ký tự đặc biệt
    - Nội dung cần phải được bảo mật

GET request

```
xhttp.open("GET", "demo_get.asp", true);
xhttp.send();
```

Hay cần truyền kèm theo parameter *t* với giá trị là một số được random

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();
```

Hay cần truyền kèm theo parameter *fname* và *lname*

```
xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
xhttp.send();
```

POST request

```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

Hay cấu phần request header và cần truyền kèm theo parameter *fname* và *lname*

```
xhttp.open("POST", "demo_post2.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

Thuộc tính onreadystatechange

```
xhttp.onreadystatechange = function () {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Server Response

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

Sử dụng Call-back function

```
loadDoc("url-1", myFunction1);

loadDoc("url-2", myFunction2);

function loadDoc(url, cFunction) {
  var xhttp;
  xhttp=new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      cFunction(this);
    }
 };
  xhttp.open("GET", url, true);
  xhttp.send();
}

function myFunction1(xhttp) {
  // action goes here
}
function myFunction2(xhttp) {
  // action goes here
}
```

responseText property và responseXML property

- responseText property

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

- responseXML property

```
xmlDoc = xhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
  txt += x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("demo").innerHTML = txt;
xhttp.open("GET", "cd_catalog.xml", true);
xhttp.send();
```

getAllResponseHeaders()

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getAllResponseHeaders();
  }
};
```

getResponseHeader()

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getResponseHeader("Last-Modified");
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Response XML file

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
    }
  };
  xhttp.open("GET", "cd_catalog.xml", true);
  xhttp.send();
}
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table = "<tr><th>Title</th><th>Artist</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i < x.length; i++) {
    table +=
      "<tr><td>" +
      x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
      "</td><td>" +
      x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
      "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
```

2. Sử dụng AJAX kết nối API

Sử dụng AJAX trong project React

- Nên gọi AJAX ở đâu trong vòng đời component?
  - Nên gọi AJAX ở componentDidMount của các phương thức vòng đời *componentDidMount*. Điều này để bạn có thể sử dụng setState để cập nhật component khi có dữ liệu được lấy.
- Ví dụ: Sử dụng dữ liệu của AJAX xét vào state
  - Component dưới đây thể hiện cách thực hiện gọi AJAX trong *componentDidMount* để xét dữ liệu vào state.
- Ví dụ API trả về một object JSON như sau:˜

```
{
  "items": [
    {
      "id": 1,
      "name": "Apples",
      "price": "$2"
    },
    {
      "id": 2,
      "name": "Peaches",
      "price": "$5"
    }
  ]
}
```

Sử dụng AJAX trong project React

```jsx
import React from 'react'

class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      error: null,
      isLoaded: false,
      items: []
    };
  }

  componentDidMount() {
    fetch("https://api.example.com/items")
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            items: result.items
          });
        },
        // Note: it's important to handle errors here
        // instead of a catch() block so that we don't swallow
        // exceptions from actual bugs in components.
        (error) => {
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
  }
```

```jsx
  render() {
    const { error, isLoaded, items } = this.state;
    if (error) {
      return <div>Error: {error.message}</div>;
    } else if (!isLoaded) {
      return <div>Loading ...</div>;
    } else {
      return (
        <ul>
          {items.map(item => (
            <li key={item.id}>
              {item.name} {item.price}
            </li>
          ))}
        </ul>
      );
    }
  }
}

export default MyComponent;
```

Sử dụng AJAX trong project React – sử dụng React Hooks

```jsx
import React, {useState, useEffect} from 'react'

function MyComponent() {
  const [error, setError] = useState(null);
  const [isLoaded, setIsLoaded] = useState(false);
  const [items, setItems] = useState([]);

  // Note: the empty deps array [] means
  // this useEffect will run once
  // similar to componentDidMount()
  useEffect(() => {
    fetch("https://api.example.com/items")
      .then(res => res.json())
      .then(
        (result) => {
          setIsLoaded(true);
          setItems(result);
        },
        // Note: it's important to handle errors here
        // instead of a catch() block so that we don't swallow
        // exceptions from actual bugs in components.
        (error) => {
          setIsLoaded(true);
          setError(error);
        }
      )
  }, [])
```

```jsx
  if (error) {
    return <div>Error: {error.message}</div>;
  } else if (!isLoaded) {
    return <div>Loading ... </div>;
  } else {
    return (
      <ul>
        {items.map(item => (
          <li key={item.id}>
            {item.name} {item.price}
          </li>
        ))}
      </ul>
    );
  }
}

export default MyComponent;
```

Bài tập về nhà

- Xây dựng một ứng dụng React đơn giản để hiện thị thông tin profile của cá nhân gồm:
  - Họ tên
  - Hình đại diện (Avatar)
  - Số diện thoại
  - Email
  - Địa chỉ
  - Sở thích
- Layout các bạn tự thiết kế hoặc hiển thị mặt định
- Thang điểm:
  - Tạo project ExpressJS API để cung cấp các thông tin trên (5 điểm)
  - Tạo project ReactJS sử dụng AJAX để call API (5 điểm)