

REACT JS

BÀI 1

React JS là gì ?

Tổng quan



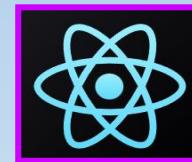
0

Welcome

❑ Các seri đã ra mắt trên tuhoc.cc :

1. Csharp: <http://csharp.tuhoc.cc>
2. Android kotlin: <http://kotlin.tuhoc.cc>
3. Python : <http://python.tuhoc.cc>
4. Java : <http://java.tuhoc.cc>
5. C ++ : <http://c.tuhoc.cc>

6. HTML-CSS-Bootstrap 5: <http://web.tuhoc.cc>
7. Javascript : <http://js.tuhoc.cc>
8. React : <http://react.tuhoc.cc>



Front End

The screenshot shows the YouTube channel page for 'Gà Lại Lập Trình'. At the top, there's a banner with a megaphone, a rocket, and text about new videos daily. Below it is the channel logo featuring a purple robot head. The channel name 'Gà Lại Lập Trình' is displayed in large letters, along with '36,7 N người đăng ký · 848 video'. A red box highlights the 'Danh sách phát' (Playlists) tab in the navigation bar. The main content area shows a grid of video thumbnails for various programming topics like Bootstrap 5, JavaScript, Photoshop, and Microsoft Edge.

Ngôn ngữ lập trình: JavaScript

Giấy phép: MIT License

Kho mã nguồn: github.com/facebook/react

Phát hành lần đầu: 29 tháng 5 năm 2013; 11 năm trước

Phát triển bởi: Meta và cộng đồng

Thiết kế bởi: Jordan Walke



<http://tuhoc.cc>

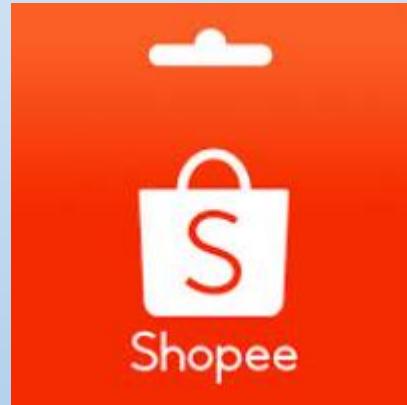
1

Tổng quan React

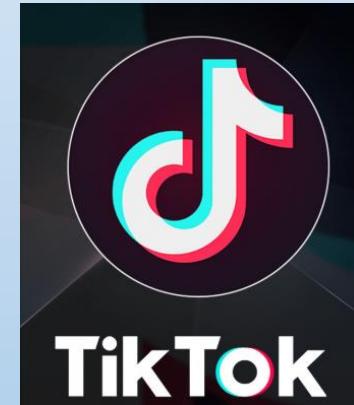
- ✓ *React là một thư viện JavaScript dùng để xây dựng giao diện người dùng (UI - User Interface).*
- ✓ *React được Facebook phát triển, và hiện họ sử dụng vào các sản phẩm của họ như FaceBook, Instagram*
- ✓ *Một số trang nổi tiếng đang sử dụng React:*



<https://www.netflix.com/>



<https://shopee.vn/>



<https://tiktok.com/>



<https://www.lazada.vn/>

2

Cùng nhìn lại kiến thức cũ ?

- ✓ Trong series **HTML-CSS**: <http://web.tuhoc.cc> chúng ta đã được học rằng: Nên tách biệt cấu trúc HTML, CSS để dễ quản lý, chỉnh sửa
- ✓ Tiếp tục trong series Javascript <http://js.tuhoc.cc> chúng ta cũng được học cách để viết khối lệnh **js** tách biệt với **HTML, CSS**

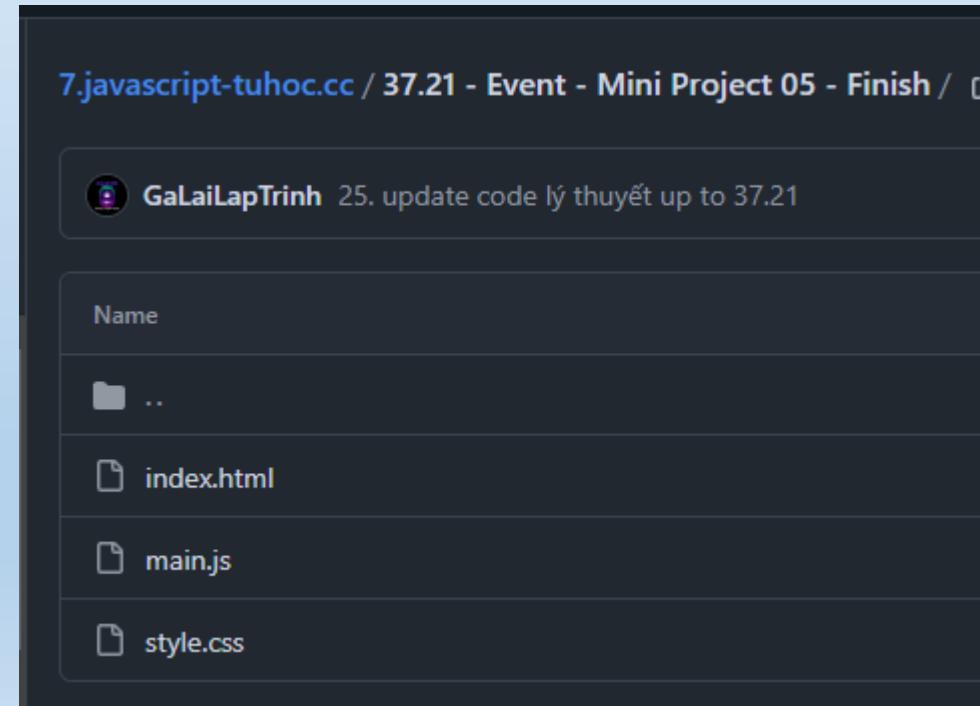
HTML



CSS



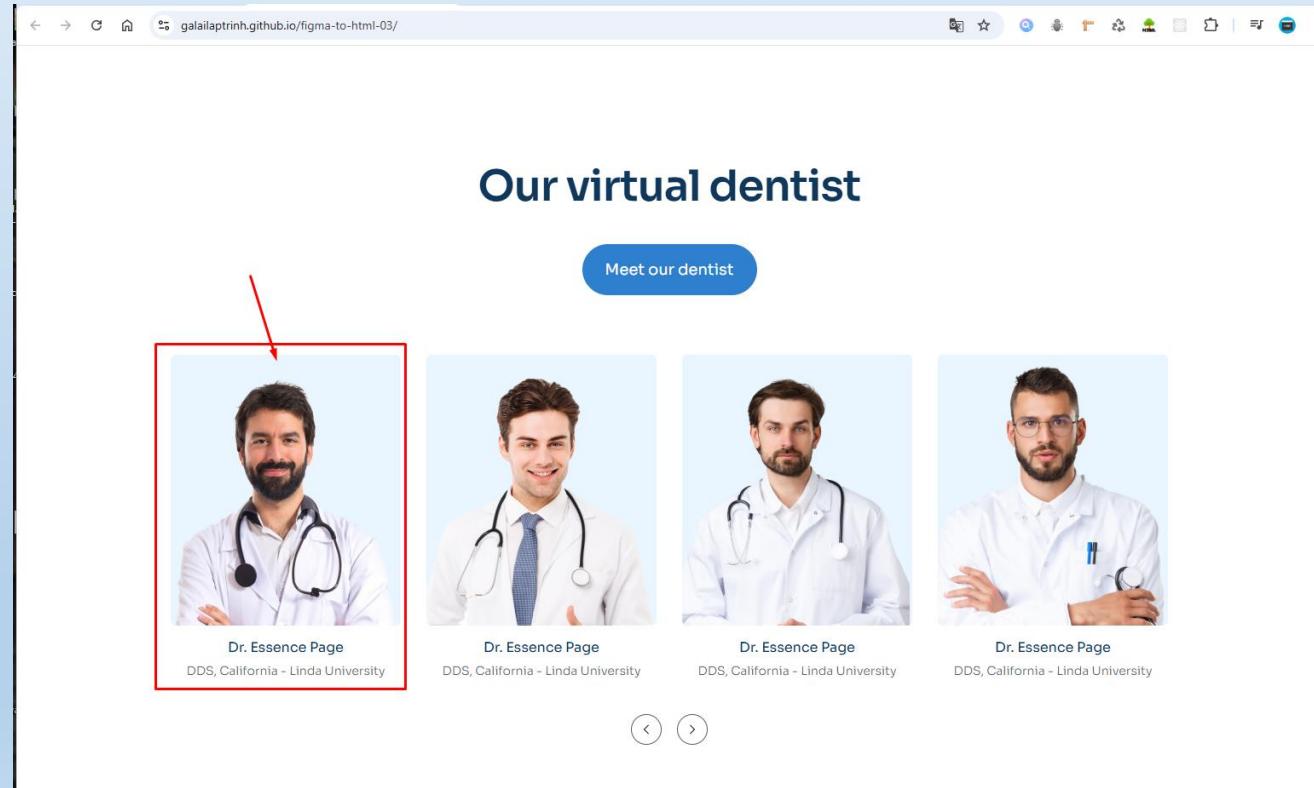
JS



3

React có gì khác biệt ?

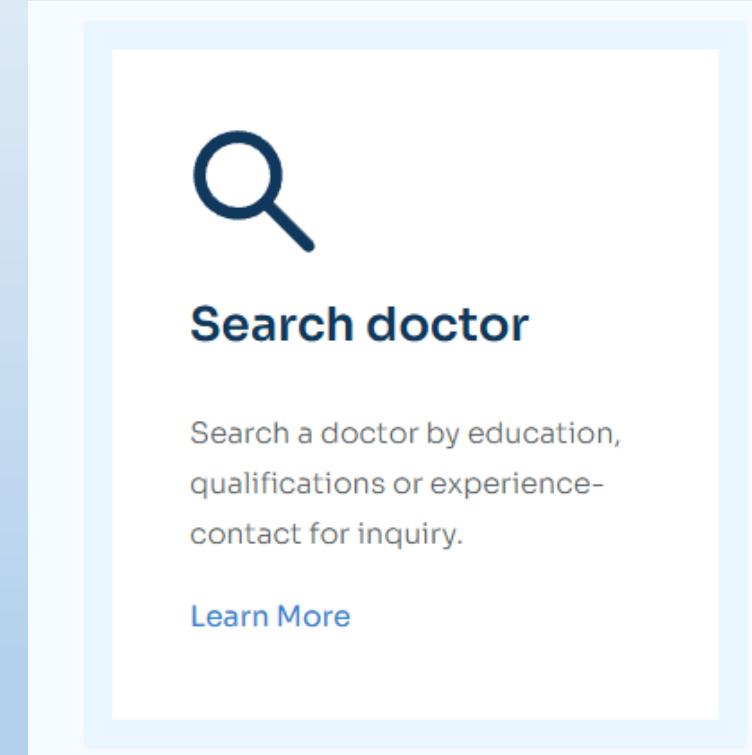
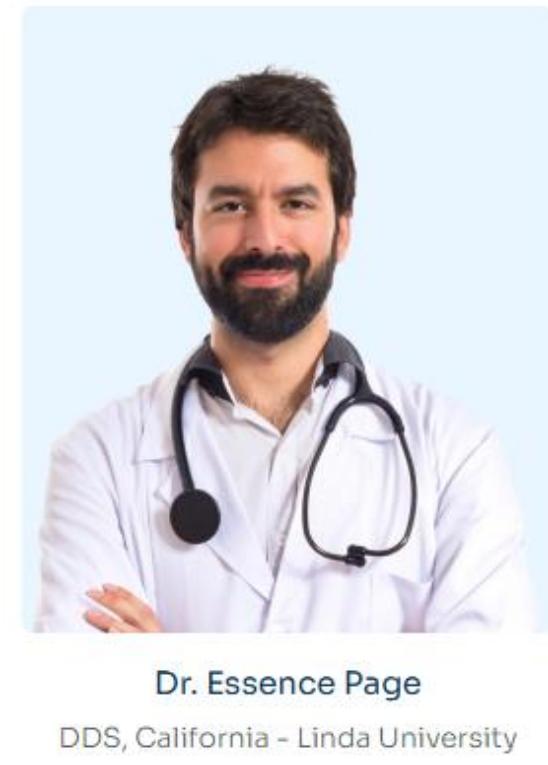
- ✓ *Người ta nhận thấy rằng, trên 1 trang web có nhiều phần lặp lại, nên họ đưa ra ý tưởng chia bố cục trang web thành các thành phần nhỏ hơn (Component)*



3

React có gì khác biệt ?

- ✓ Các khôi Component này sẽ là 1 khôi mã hoàn chỉnh bao gồm : HTML – CSS - JS



- ✓ Và bạn có thể tạo bao nhiêu component này tùy thích, khi sử dụng chúng ta chỉ cần update lại nội dung cho từng component.

3

React có gì khác biệt ?

- ✓ *Ví dụ với mạng xã hội X, bạn có thể thấy các bài đăng có cấu trúc tương đồng nhau về cấu trúc*

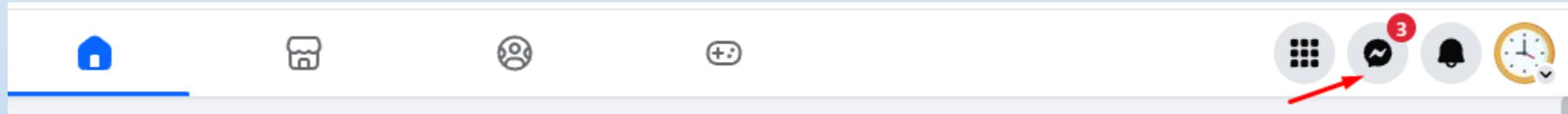


- ✓ *Khi cuộn trang đến điểm cuối nguồn cấp dữ liệu, nó mới phát hiện ra bạn đã đến ngưỡng và nó sẽ yêu cầu máy chủ cung cấp nhiều dữ liệu hơn để hiển thị phần còn lại của ứng dụng web*
- ✓ *Và điểm đặc biệt, là các phần khác của trang vẫn dữ nguyên mà không cần phải tải lại trang (Khác với web truyền thống trước kia chúng ta đã học)*

3

React có gì khác biệt ?

- ✓ Điều này mang lại tác dụng vô cùng to lớn với trải nghiệm người dùng.
- ✓ Ví dụ khi bạn đang dùng facebook, sau đó có tin nhắn đến, React sẽ giúp chúng ta chỉ cập nhật lại component Message để hiển thị lại số lượng tin nhắn chưa được đọc, còn các thành phần khác không bị ảnh hưởng



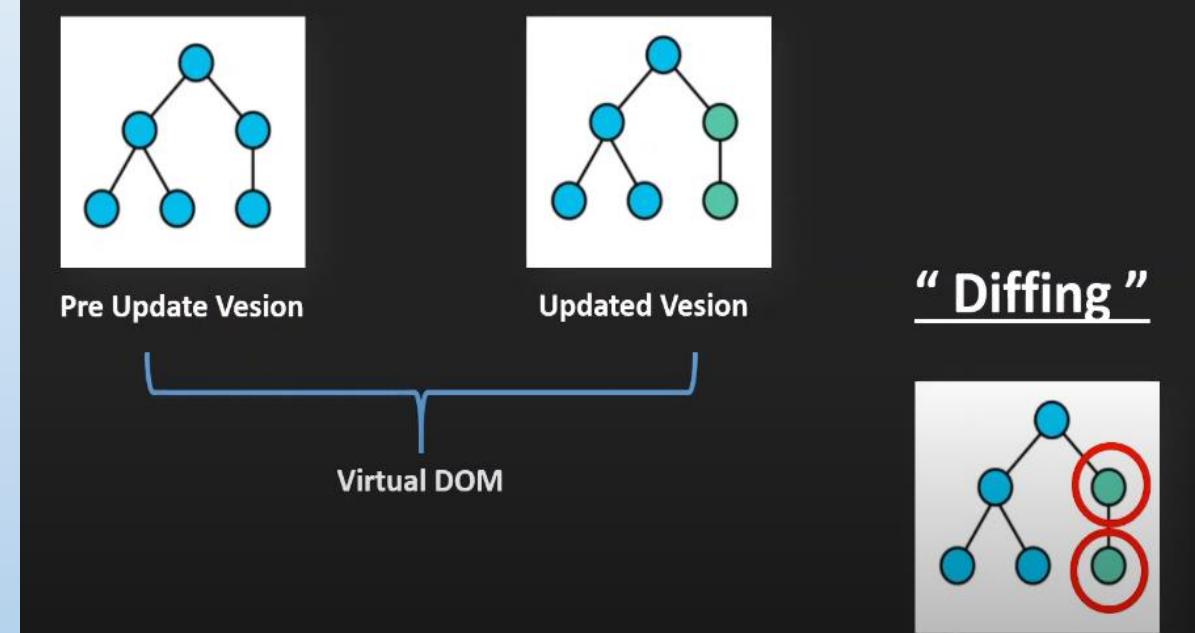
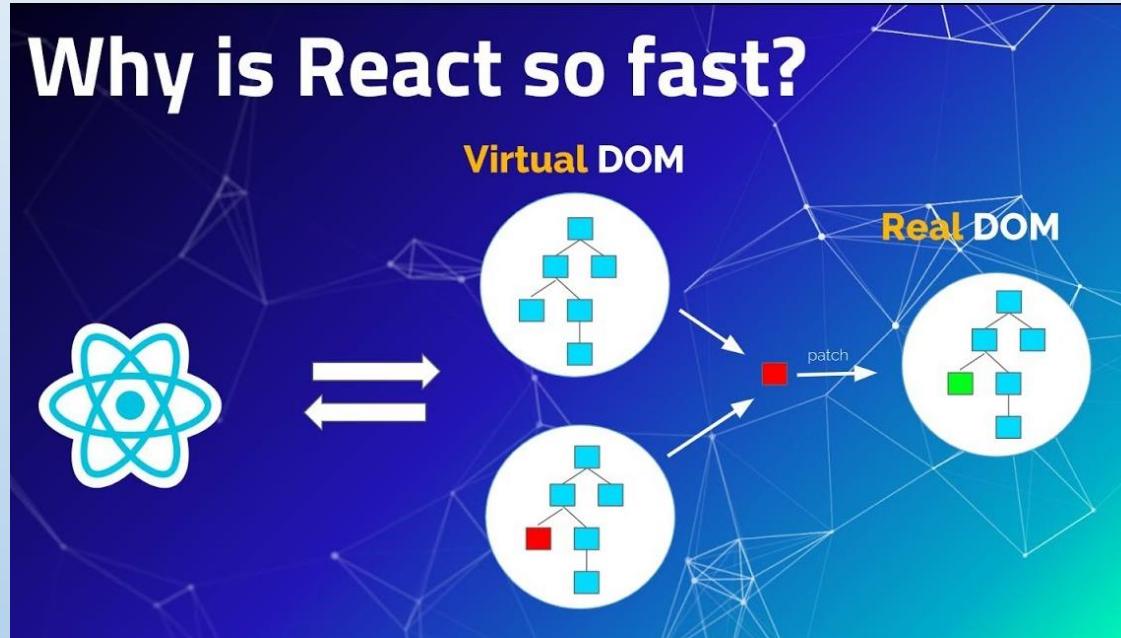
- ✓ Hoặc nếu chia nhỏ các thành phần, ví dụ :
 - + Bài đăng → Component
 - + Comment → Component

Nếu có 1 bài đăng mới hoặc 1 bình luận mới, thì chỉ bài đăng đó được tự cập nhật, mà không ảnh hưởng đến các thành phần khác, không phải load lại toàn bộ trang web.

4

React update component thế nào ?

- ✓ **Virtual DOM** (*Document Object Model ảo*) là một bản sao của *DOM* thật
- ✓ Nó không được hiển thị trực tiếp trên trình duyệt mà chỉ tồn tại trong bộ nhớ của ứng dụng.

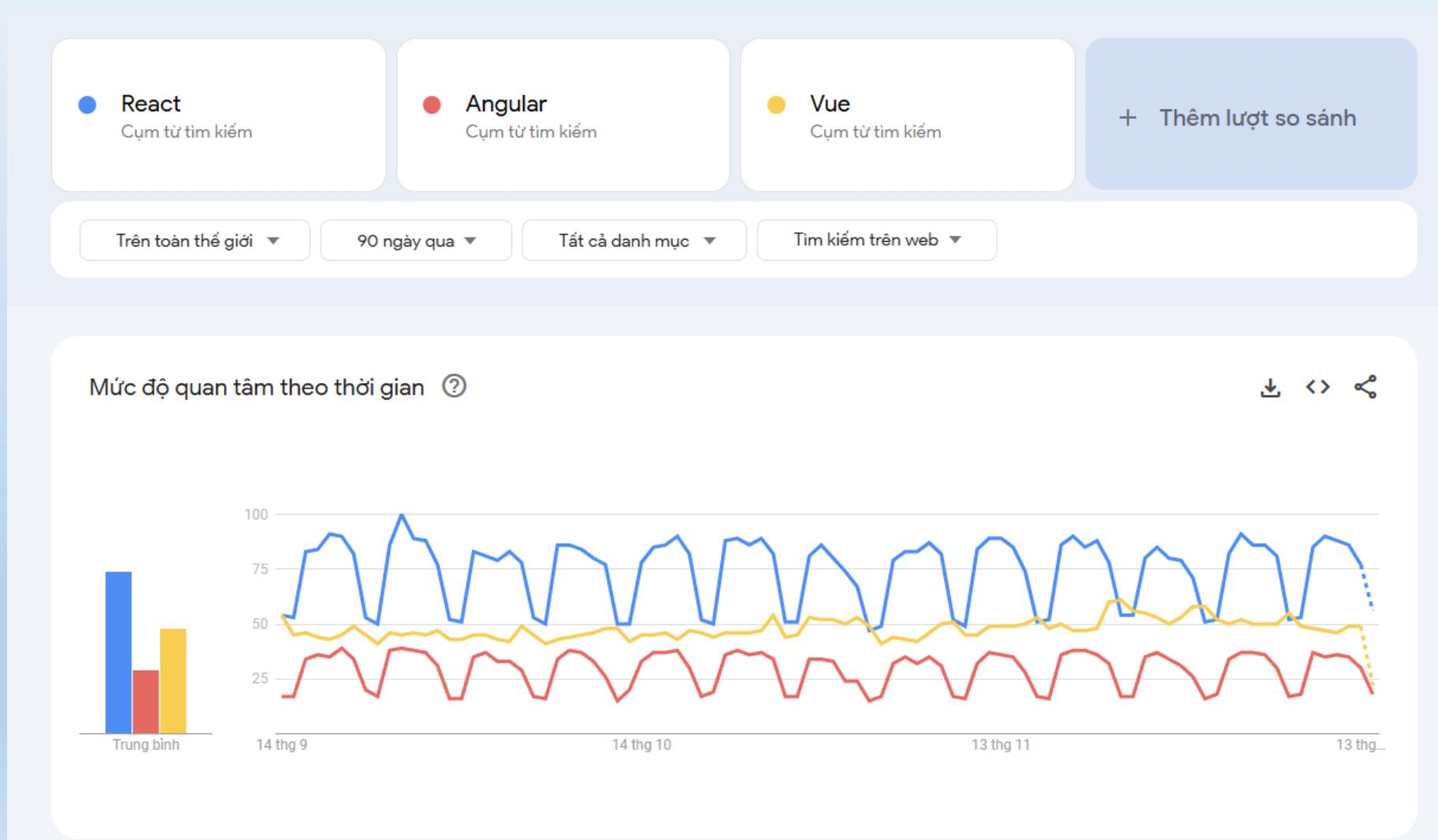


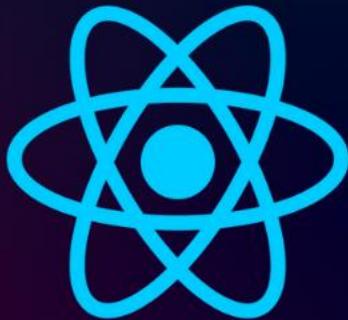
1. **Render lần đầu:** React tạo một bản *Virtual DOM* và hiển thị giao diện.
2. **Update virtual DOM :** Khi có thay đổi trong state hoặc props, React tạo một *Virtual DOM* mới.
3. **Diffing Algorithm:** So sánh giữa *Virtual DOM* cũ và mới để xác định sự khác biệt.
4. **Cập nhật DOM thật:** Chỉ những phần thay đổi được áp dụng vào *DOM* thật.

5

React Trends

<https://trends.google.com.vn/trends/explore?date=today%203-m&q=React,Angular,Vue&hl=vi>





REACT JS



BÀI 2

JavaScript ES6

Named Export
Default Export



1

Giới thiệu về Module

- ✓ Một module là một file chứa mã JavaScript (JS) mà bạn có thể tái sử dụng ở các file khác.
- ✓ Giúp chia nhỏ ứng dụng thành các phần nhỏ, dễ quản lý và bảo trì.

Step 1: Thêm thuộc tính type="module"

```
<body>
    <!-- nhúng js --&gt;
    &lt;script type="module" src=".main.js"&gt;&lt;/script&gt;
&lt;/body&gt;</pre>
```

- ✓ Trong JavaScript, export được sử dụng để chia sẻ các hàm, biến, class từ một module. Có hai cách chính để export:
 - *Named Export* : Cho phép export nhiều phần tử từ một file.
 - *Default Export* : Mỗi module chỉ có một default export.

2

Named Export

- ✓ **Named Export** : Cho phép export nhiều phần tử từ một file - hàm, biến, class .

JS named.js X

```
//Cách 1: khai báo hàm, biến, class rồi export
const myVar = 10;

function myFunction() {
    console.log("xin chào đây là hàm!");
}

export {myVar, myFunction};
```

```
//Cách 2 : export ngay từ khi khai báo
export const myVar = 10;

export function myFunction() {
    console.log("xin chào đây là hàm!");
}
```

2

Named Import

- ✓ *Named Export : Tại nơi cần sử dụng, ta thực hiện Import từ Named Export*

```
<body>
  ...
  <!-- nhung js -->
  <script type="module" src="./main.js"></script>
</body>
```

Step 1: Thêm thuộc tính type="module"

JS main.js X

```
// 1. lấy một số phần tử cần thiết
import {myVar} from "./named.js";
console.log(myVar);
```

```
//2. hoặc chỉ lấy một số phần tử cần thiết
import {myVar, myFunction} from "./named.js";
console.log(myVar);
myFunction();
```

```
//3. hoặc lấy tất cả
import * as name from "./named.js";
console.log(name.myVar);
name.myFunction();
```

1. Lúc này, name sẽ là một đối tượng chứa tất cả các thành phần được export từ file named.js

3

Default Export

- ✓ *Default Export* : Mỗi module chỉ có một default export - hàm, biến, class ..

JS default.js X

```
1 let df = "Đây là biến default";
2 export default df;
```

```
function total(a, b) {
|   return a + b;
}
export default total;
```

JS main.js X

```
//4. import default
import df from "./default.js";
console.log(df);
```

Đây là biến default

```
//4. import default
import df from "./default.js";
console.log(df(1, 2));
```

4

Lưu ý

- ✓ Nếu file named.js có một Default Export, nó sẽ không được gộp vào đối tượng name khi sử dụng import * as

JS named.js X

```
//file có 1 export default  
let df2 = "Đây là biến default 2 trong named";  
export default df2;
```

JS main.js X

```
import * as name from "./named.js";  
console.log(name.myVar);  
name.myFunction();  
console.log(name.df2); //undefined vì df2 là default export
```

JS main.js X

```
//import riêng default export  
import df2 from "./named.js";  
console.log(df2);
```

Đây là biến default 2 trong named

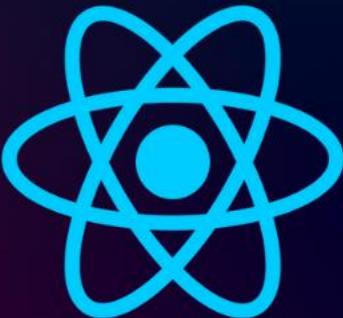
5

Đổi tên khi import

| Trường hợp | Cách đặt tên (bí danh) mới |
|----------------|---|
| Default Export | <i>Đặt tên tùy ý khi import, không cần as</i> |
| Named Export | <i>Dùng từ khóa as</i> |

```
//5.1 import default: đặt tên tùy ý
import anyName from "./default.js";
console.log(anyName(1, 2));
```

```
//5.2 import named: đặt tên với as
import {myVar as varAlias, myFunction as funcAlias} from "./named.js";
console.log(varAlias); //10
```

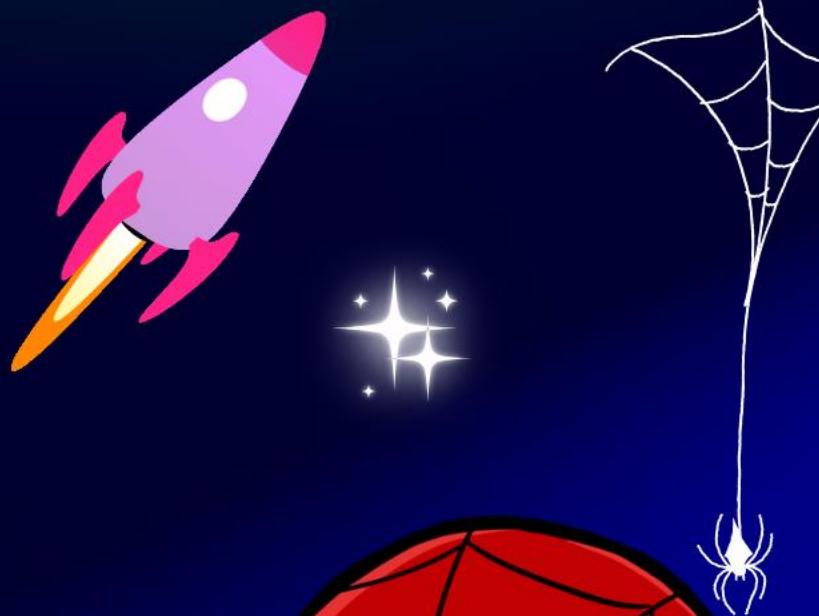


REACT JS

BÀI 3

Tạo Dự Án React

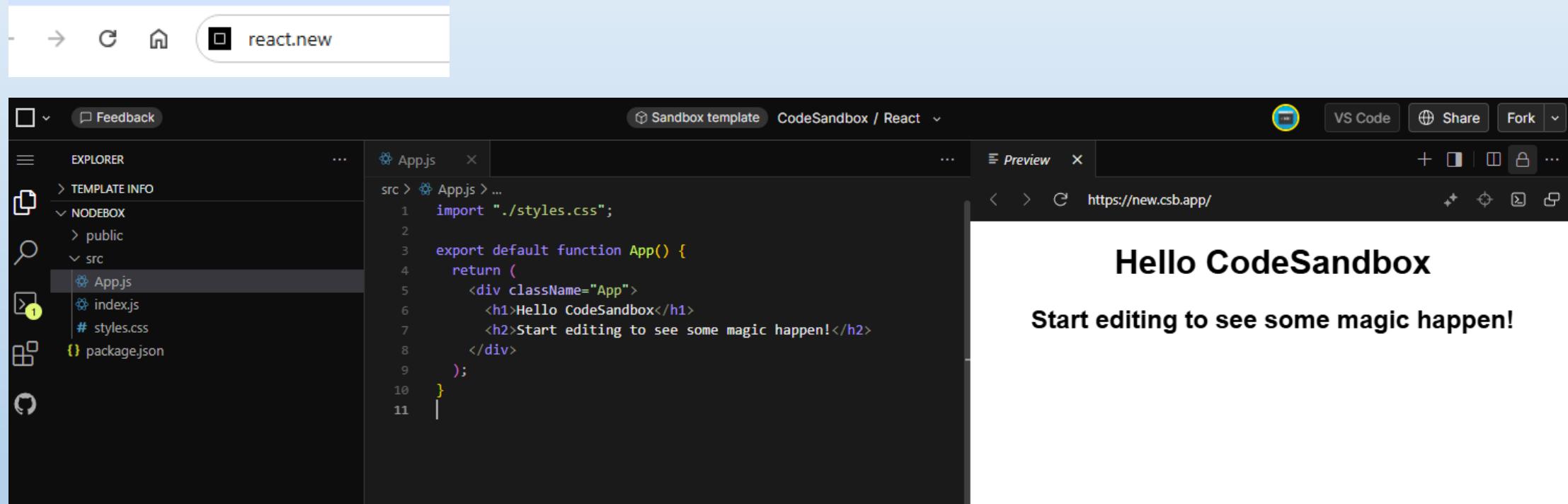
CodeSandbox
vs Local Project



1

CodeSandbox

- ✓ *Tạo một dự án React mới trên CodeSandbox – Đăng ký tài khoản free :*

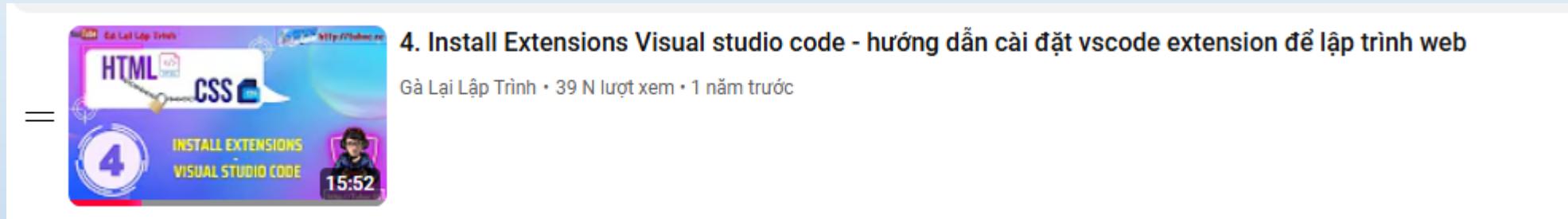


- ❑ *CodeSandbox cung cấp môi trường lập trình React trực tuyến:*
 - ✓ *Không cần cài đặt phần mềm hay công cụ nào.*
 - ✓ *Các bạn có thể thực hành code react trực tiếp trên trình duyệt*

2

Some extension in Vscode

Cài đủ các tiện ích ở bài 4 - HTML-CSS-Bootstrap 5: <http://web.tuhoc.cc>



❖ *Eslint : Hỗ trợ phát hiện lỗi trong quá trình soạn thảo mã*

eslin

ESLint 39.7M Integrates ESLint JavaScri... Microsoft Install

Prettier ESLint 3M A Visual Studio Extension ...

ESLint

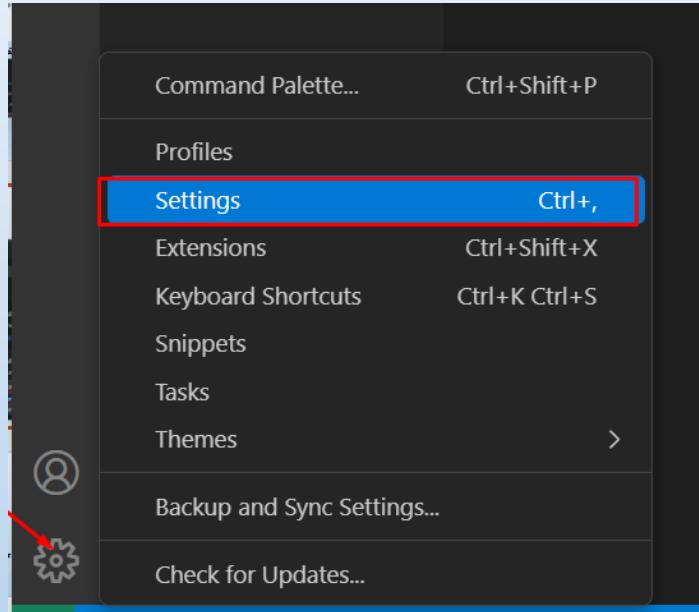
Microsoft [microsoft.com](#) 39,732,715 ★★★★★ (240)

Integrates ESLint JavaScript into VS Code.

Install Auto Update This extension is recommended based on the files you recently opened.

2

Some extension in Vscode



A screenshot of the VS Code Settings sidebar. The 'eslint run' setting is selected and highlighted with a red box. The sidebar shows sections for 'User' and 'Workspace'. Under 'User', there are collapsed sections for 'Features (1)', 'Terminal (1)', 'Extensions (7)', and 'ESLint (7)'. Under 'Workspace', there are sections for 'Eslint: Runtime' and 'Eslint: Run'. The 'Eslint: Run' section has a dropdown menu with options: 'onType', 'onSave', and 'onType' again. 'onSave' is highlighted with a red box. A blue box highlights the 'default' option at the bottom of the list.

3

React với Vscode

1. Cài node.js :



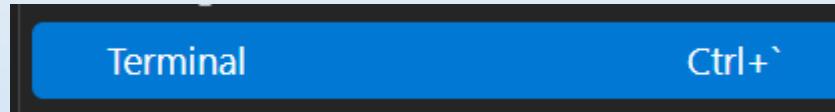
The screenshot shows the official Node.js download page. At the top, there's a navigation bar with links for Learn, About, Download, Blog, Docs, Contribute, and Certification. A red arrow points to the 'Download' button. Below the navigation, there's a section titled 'Download Node.js®' with the sub-instruction 'Download Node.js the way you want.' Underneath, there are tabs for Package Manager, Prebuilt Installer (which is highlighted with a green border and a red arrow), Prebuilt Binaries, and Source Code. Further down, there's a dropdown menu for selecting a version, which is set to 'v22.12.0 (LTS)' (highlighted with a red arrow). To the right of the dropdown are dropdown menus for 'Windows' (set to 'running') and 'x64'. At the bottom of this section is a large green 'Download Node.js v22.12.0' button, which is also highlighted with a red arrow. Below this button, there's some descriptive text about Node.js including npm (10.9.0) and links to changelog, blog post, and verification pages.

```
C:\Users\tuhoc.cc>node -v  
v22.12.0
```

3

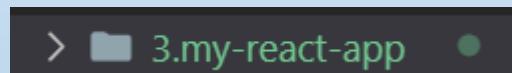
React với Vscode

2. Tạo dự án với vite :



a. *Gõ lệnh sau và làm theo hướng dẫn để tạo mới dự án react, ở đây đặt tên là 3.my-react-app:*

- npm create vite@latest
- Set-ExecutionPolicy RemoteSigned nếu bị lỗi như trong video thì mới cần dùng lệnh này



b. *Di chuyển vào project cd <tên project>:*

```
\1.code-bai-hoc-tuhoc.cc\8.reactjs-tuhoc.cc> cd 3.my-react-app
\1.code-bai-hoc-tuhoc.cc\8.reactjs-tuhoc.cc\3.my-react-app>
```

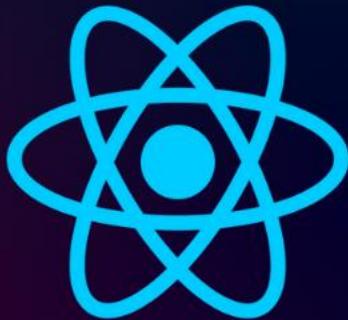
c. *Thực hiện các lệnh sau :*

- npm install
- npm run dev

```
VITE v6.0.3 ready in 206 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

Khi tắt máy chủ, lần sau chạy lại cần run lại

- npm run dev



REACT JS

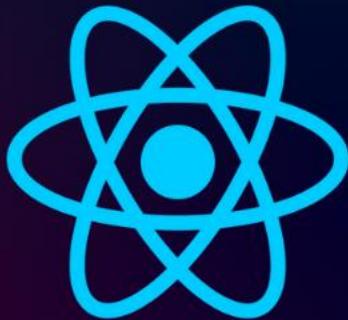


BÀI 4

JS Function Review



**Ôn tập
Function trong JavaScript**



REACT JS

BÀI 5

JS Function Review

Arrow Function



1

Arrow function

Tổng quan:

khi thân hàm có nhiều hơn 1 dòng lệnh
cần thêm khối {}

```
let ten_bien = (parameters) =>{  
    // Thân hàm  
    // Return something  
}
```

```
// Arrow function  
let multiplyAndAddArrow = (a, b) => {  
    let product = a * b;  
    let sum = a + b;  
    return product + sum;  
};
```

```
// Gọi hàm  
console.log(multiplyAndAdd(3, 5)); // 23  
console.log(multiplyAndAddArrow(3, 5)); // 23
```

2. Function expression

```
let multiplyAndAdd = function (a, b) {  
    let product = a * b;  
    let sum = a + b;  
    return product + sum;  
};
```

1

Arrow function

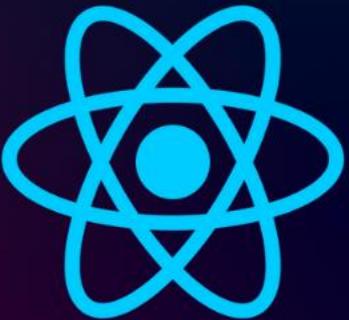
- ✓ *Arrow Function (anonymous function - hàm ẩn danh) là một cách khai báo, giản lược hơn so với cách khai báo truyền thống.*
- ✓ *Nó sử dụng phô biến trong react*

```
// Arrow function (hàm mũi tên)
let multiplyArrow = (a, b) => a * b;
```

*Với các biểu thức đơn giản,
Cú pháp sẽ ngắn gọn hơn nhiều mà không
cần phải dùng keyword return*

2. Function expression

```
// Function Expression
let multiply = function (a, b) {
  return a * b;
};
```



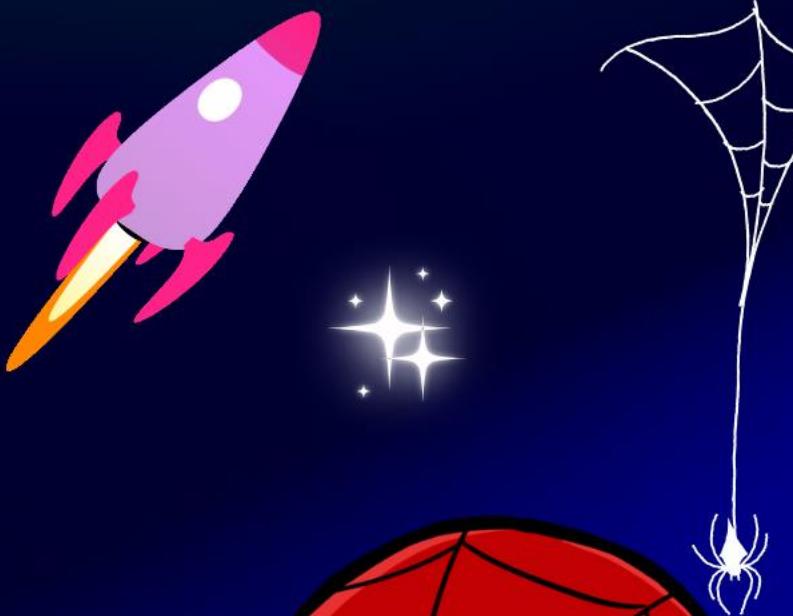
REACT JS

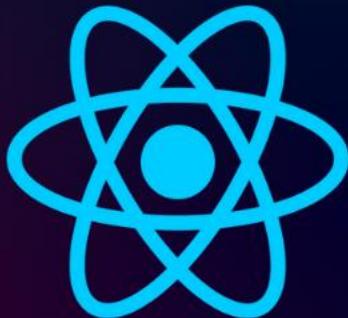
BÀI 6

JS OOP Review



Objects và Classes JS



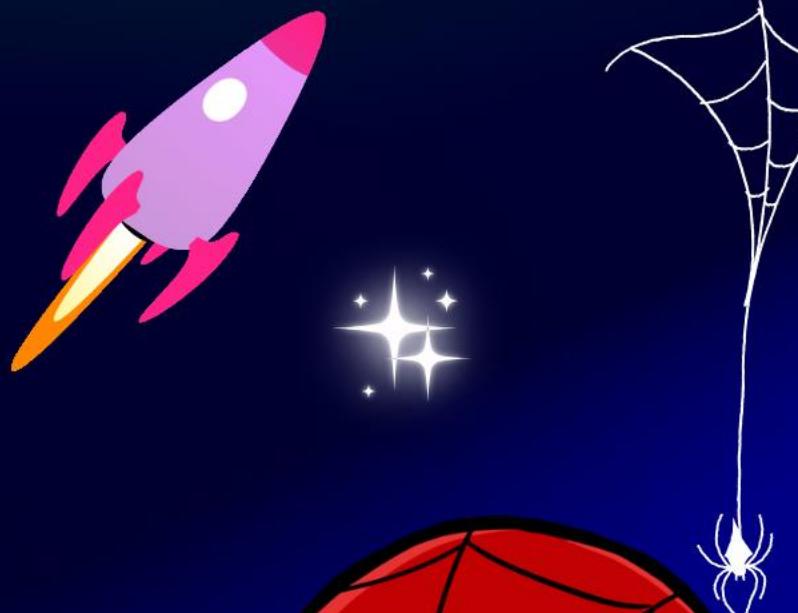


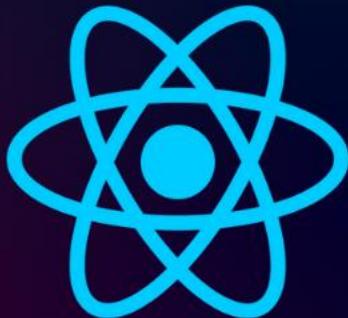
REACT JS

BÀI 7

JS Array Review

Ôn tập mảng





REACT JS



BÀI 8

Destructuring JS

Array & Object



1

Deconstructing là gì ?

- ✓ *Deconstructing là một tính năng quan trọng, cho phép bạn trích xuất dữ liệu từ arrays hoặc objects một cách dễ dàng và ngắn gọn.*
 - *Với Array: Trích xuất dữ liệu dựa trên vị trí (index).*
 - *Với Object: Trích xuất dữ liệu dựa trên tên thuộc tính (property name).*

2

Destructuring Arrays

```
//giả sử chúng ta có một mảng như sau:  
const userName = ["Toàn", "Hà", "Huy"];
```

//Nếu muốn lấy ra các phần tử trong mảng trên, chúng ta sẽ làm như sau:

```
const user1 = userName[0];  
const user2 = userName[1];  
const user3 = userName[2];  
console.log(user1, user2, user3); //Toàn Hà Huy
```

- ✓ 2.1 *Nhưng nếu sử dụng destructuring, chúng ta sẽ viết ngắn gọn hơn như sau:*

```
const [user4, user5, user6] = userName;  
console.log(user4, user5, user6); //Toàn Hà Huy
```

2

Destructuring Arrays

2.2 Có thể bỏ qua phần tử không cần thiết bằng cách để dấu phẩy:

```
const [user7, , user8] = userName;
console.log(user7); //Toàn
console.log(user8); //Huy
```

2.3 Destructuring với giá trị mặc định:

```
const [user9, user10, user11, user12 = "Huy"] = userName;
```

2.4 Destructuring với rest parameter:

```
const [user13, ...rest] = userName;
//rest sẽ chứa các phần tử còn lại sau phần tử đầu tiên
console.log(user13); //Toàn
console.log(rest); //["Hà", "Huy"]
```

3

Destructuring Objects

```
//Giả sử chúng ta có một object như sau:  
const user = {  
    name: "Toàn",  
    age: 20,  
};  
  
//Nếu muốn lấy ra các thuộc tính trong object trên, chúng ta sẽ làm như sau:  
const ten = user.name;  
const tuoi = user.age;  
console.log(ten); //Toàn  
console.log(tuoi); //20
```

3.1 *Nhưng nếu sử dụng **destructuring**, chúng ta sẽ viết ngắn gọn hơn như sau:*

*Lưu ý: **Tên biến phải trùng với tên thuộc tính trong object***

```
const {name, age} = user;  
console.log(name); //Toàn  
console.log(age); //20
```

3

Destructuring Objects

3.2 Đổi tên biến khi destructuring Obj (Dùng alias – tên bí danh) :

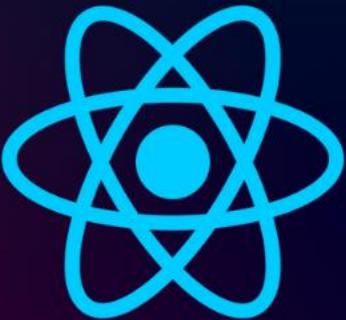
```
const {name: ten1, age: tuoi1} = user;
console.log(ten1); //Toàn
console.log(tuoi1); //20
```

3.3 Destructuring với giá trị mặc định:

```
const {name: ten2, age: tuoi2, gioiTinh = "Nam"} = user;
console.log(ten2); //Toàn
console.log(tuoi2); //20
console.log(gioiTinh); //Nam
```

3.4 Destructuring với rest parameter:

```
const {name: ten3, ...rest1} = user;
console.log(ten3); //Toàn
console.log(rest1); //{"age": 20}
```



REACT JS



BÀI 9

Destructuring JS



Ứng dụng Destructuring với đối tượng trong hàm

4

Destructuring in Function

- ✓ *Destructuring còn có thể được sử dụng trực tiếp trong tham số (parameter) của hàm.*
Điều giúp tránh việc lặp lại truy cập thuộc tính bên trong thân hàm.

```
//giả sử có 1 đối tượng đơn hàng
const donHang = {
    maDonHang: 101,
    tienTe: "VND",
    tongTien: 500000,
};
```

4

Destructuring in Function

```
//giả sử có 1 đối tượng đơn hàng
const donHang = {
    maDonHang: 101,
    tienTe: "VND",
    tongTien: 500000,
};
```

```
//1. Không sử dụng Destructuring
function xuLyDonHang(donHang) {
    const maDonHang = donHang.maDonHang;
    const tienTe = donHang.tienTe;
    const tongTien = donHang.tongTien;

    console.log(`Mã đơn hàng: ${maDonHang}`);
    console.log(`Tiền tệ: ${tienTe}`);
    console.log(`Tổng tiền: ${tongTien}`);
}

//gọi hàm:
xuLyDonHang(donHang);
```

```
//2. Sử dụng Destructuring
console.warn("Sử dụng Destructuring:");
function xuLyDonHang2({maDonHang, tienTe, tongTien}) {
    console.log(`Mã đơn hàng: ${maDonHang}`);
    console.log(`Tiền tệ: ${tienTe}`);
    console.log(`Tổng tiền: ${tongTien}`);
}

xuLyDonHang2(donHang);
```

*Ngắn gọn và dễ đọc hơn:
Không cần gọi donHang.property nhiều lần.*

4

Destructuring in Function

3. Sử dụng Destructuring với mặc định

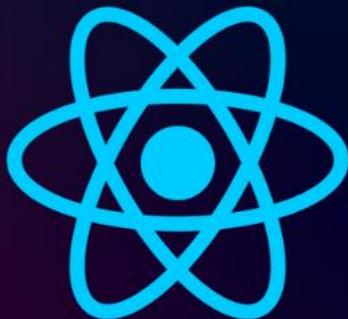
```
function xuLyDonHang({maDonHang = 0, tienTe = "VND", tongTien = 0}) {  
    console.log(`Mã đơn hàng: ${maDonHang}`);  
    console.log(`Tiền tệ: ${tienTe}`);  
    console.log(`Tổng tiền: ${tongTien}`);  
}  
  
//Gọi hàm với dữ liệu thiếu:  
xuLyDonHang({maDonHang: 102}); //Tiền tệ và tổng tiền sẽ lấy giá trị mặc định  
  
/*  
Mã đơn hàng: 102  
Tiền tệ: VND  
Tổng tiền: 0  
*/
```

4

Destructuring in Function

4. Sử dụng Rest Parameter

```
function xuLyDonHang({maDonHang, ...rest}) {  
    console.log(`Mã đơn hàng: ${maDonHang}`);  
    console.log(rest);  
}  
  
//Gọi hàm:  
xuLyDonHang({  
    maDonHang: 103,  
    tienTe: "USD",  
    tongTien: 1000,  
    khachHang: "Nguyen Van A",  
});  
//Kết quả:  
/*  
Mã đơn hàng: 103  
{tienTe: 'USD', tongTien: 1000, khachHang: 'Nguyen Van A'}  
*/
```



REACT JS

BÀI 10



Spread Operator



1

Toán tử Spread

- ✓ *Spread Operator (...).*
- ✓ *Tác dụng :*
 - "Trải" các phần tử của một mảng ra thành các phần tử riêng lẻ.
 - "Trải" các cặp key-value trong một đối tượng để kết hợp hoặc sao chép.
- ✓ *Thường được sử dụng để:*
 - Gộp (merge) dữ liệu.
 - Sao chép mảng hoặc đối tượng.
 - Thêm dữ liệu mới một cách linh hoạt.

2

spread operator với mảng

```
// Danh sách sở thích hiện tại
let soThich = ["đọc sách", "nghe nhạc"];

// Sở thích mới muốn thêm
const SoThichMoi = "đi du lịch";

// Cập nhật danh sách sở thích
soThich = [...soThich, SoThichMoi];

console.log("Danh sách sở thích sau khi thêm:", soThich);
```

```
/* Kết quả:
Danh sách sở thích sau khi thêm:
[
  'đọc sách',
  'nghe nhạc',
  'đi du lịch'
]*/
*/
```

2

spread operator với object

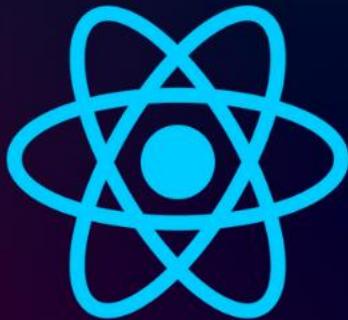
```
// Thông tin người dùng ban đầu
let user = {
    id: 1,
    name: "Nguyễn Văn A",
    email: "vana@gmail.com",
};

// Thông tin cần cập nhật
const updates = {
    address: "123 Đường ABC",
    role: "admin",
};

// Cập nhật thông tin người dùng
user = {...user, ...updates};

console.log("Thông tin người dùng sau khi cập nhật:", user);
```

```
/* Kết quả:
Thông tin người dùng sau khi cập nhật:
{
    id: 1,
    name: 'Nguyễn Văn A',
    email: 'vana@gmail.com',
    address: '123 Đường ABC',
    role: 'admin'
}
*/
```

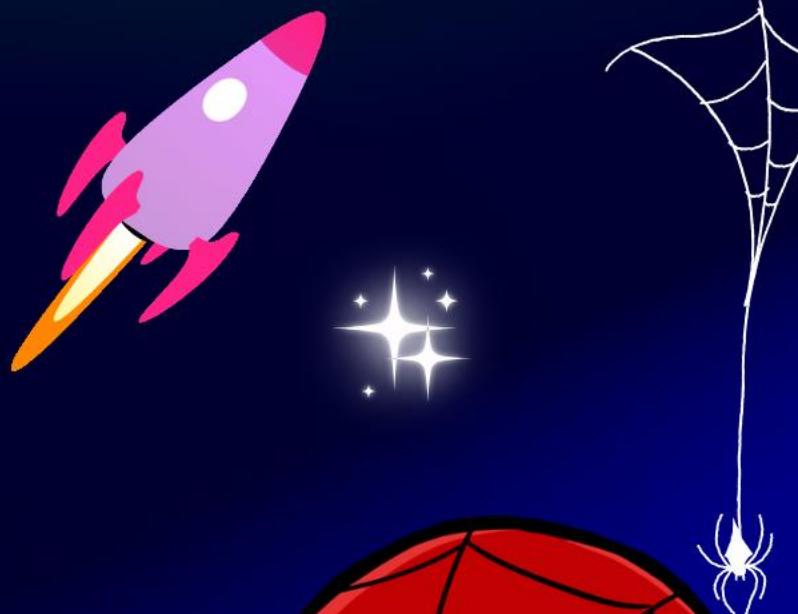


REACT JS

BÀI 11

JSX

Cách Sử Dụng JSX
Trong React



1 Component trong React

- ✓ **Component trong React:** Component là các khối chứa **HTML, CSS, và JavaScript** liên quan. Chúng thường có cùng kiểu dáng và cùng logic JavaScript.



- ✓ **Code thông thường:**
 - Dễ dẫn đến việc lặp code
 - Trong các dự án lớn, phải ghi chú từng thành phần, khi sửa đổi dễ bị bỏ sót mã.

- ✓ **Tái sử dụng:** Một Component có thể được dùng nhiều lần, giúp giảm thiểu việc lặp lại code.
- ✓ **Quản lý dễ dàng:** Chia nhỏ giao diện phức tạp thành các phần nhỏ hơn.
- ✓ **Tập trung mã liên quan:** HTML, CSS, và JavaScript liên quan được gói gọn trong từng Component.

2

Tệp JSX là gì?

- ✓ Trong file **index.html**, bạn sẽ thấy một phần tử duy nhất với ***id="root"***
- ✓ Toàn bộ nội dung giao diện sẽ được **React render** lên đây.

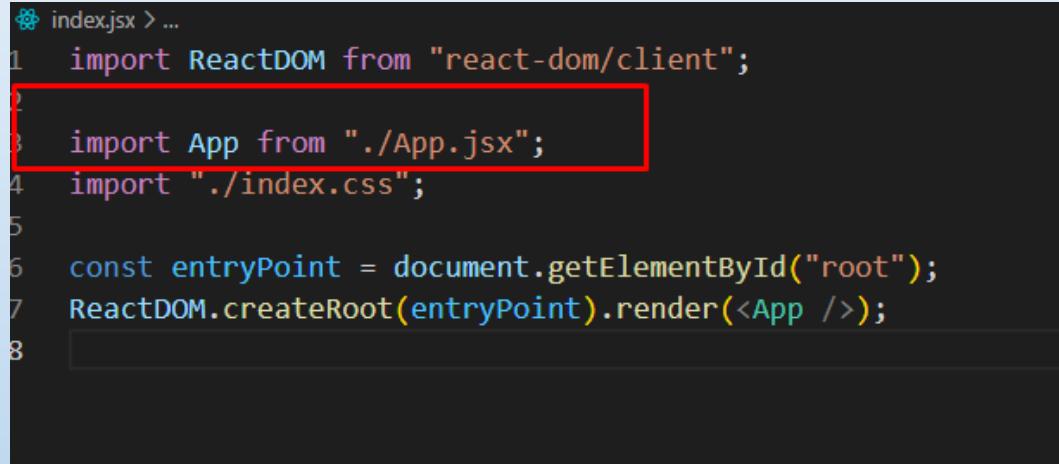
```
</head>
<body>
  <div id="root"></div>
  <script type="module" src="/src/index.jsx"></script>
</body>
</html>
```

- ✓ Nhìn vào thân body chúng ta thấy, có 1 tệp **jsx** được nhúng
- ✓ Phần mở rộng **jsx** khá kỳ lạ với các bạn bây giờ. Nó không phải mã javascript thuần, chúng ta hãy click vào nó để khám phá

2

Giải thích JSX

- ✓ Chúng ta vẫn chưa nhìn thấy bất cứ hình ảnh hay text nào hiển thị trên trình duyệt



```
index.jsx > ...
1 import ReactDOM from "react-dom/client";
2
3 import App from "./App.jsx"; // Line 3 is highlighted with a red box
4 import "./index.css";
5
6 const entryPoint = document.getElementById("root");
7 ReactDOM.createRoot(entryPoint).render(<App />);
8
```

- ✓ Chúng ta thấy có phần import từ file *App.jsx* → Và trong file này mới là nơi chứa nội dung hiển thị trên màn hình mà các bạn đang thấy
- ✓ Tuy nhiên bên trong nó không phải js thuần, nó là 1 hàm js nhưng lại return về 1 phần tử HTML
- ✓ Và cũng chính vì lý do đó, nó có phần mở rộng là jsx, không phải .js như chúng ta đã biết về javascript

2

Giải thích JSX

JavaScript Syntax eXtension

JSX

```
function App() {
  return (
    <>
      <h3>Welcome</h3>
      <p>You are logged in!</p>
    </>
  );
}
```

- ✓ *File .jsx là JavaScript Syntax Extension – cú pháp mở rộng từ JavaScript.*
- ✓ *Trình duyệt không thể hiểu trực tiếp file .jsx, mà cần một công cụ để chuyển đổi sang JavaScript thuần.*
- ✓ *Trong khóa học này : Vite đóng vai trò chuyển đổi jsx → js.*
- ✓ *Để chạy được vite chúng ta cần cài node js (môi trường để chạy vite)*

3

Fragment

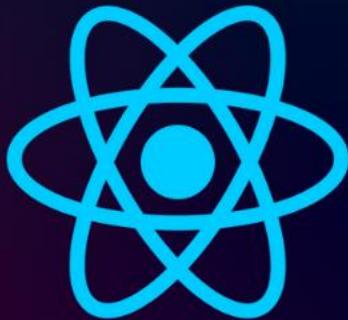
- ✓ *Cách viết sử dụng cặp thẻ <> và </> là cú pháp Fragment trong React. Đây là một tính năng giúp nhóm nhiều phần tử con lại với nhau mà không cần thêm một thẻ HTML bao bọc nào (như div, span, v.v.).*

<https://react.dev/reference/react/Fragment>

```
function App() {
  return (
    <>
      <h3>Welcome</h3>
      <p>You are logged in!</p>
    </>
  );
}
```

- ✓ *Lý do sử dụng Fragment*

- Không thêm thẻ dư thừa vào DOM: Khi bạn chỉ cần nhóm các phần tử lại mà không muốn thêm thẻ HTML bao bọc, Fragment giúp tránh việc làm "bẩn" cấu trúc DOM.*
- Giữ cho mã sạch sẽ: Giảm thiểu việc sử dụng các thẻ không cần thiết chỉ để bọc nội dung.*

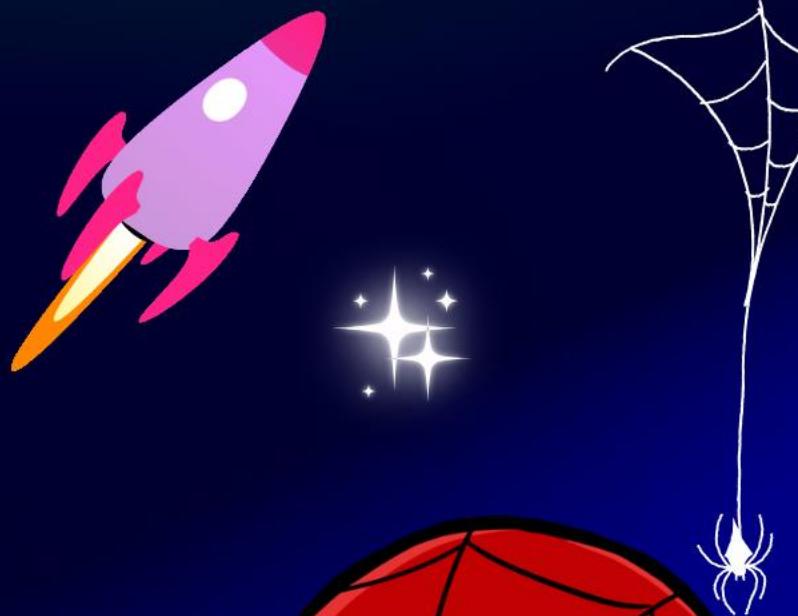


REACT JS

BÀI 12

Thực hành

Tạo ứng dụng React
cơ bản



1

Quy ước Function trong ReactJS

- ✓ Như bạn thấy trong react, component thật ra chỉ là 1 hàm js, tuy nhiên nó phải tuân theo quy tắc
1. Tên function viết hoa chữ cái đầu
 2. Và nó phải return về giá trị có thể hiển thị được, và là 1 phần tử duy nhất

```
function App() {
  const [count, setCount] = useState(0);

  return (
    <>
    <div>...
    </div>
    <h1>Vite + React </h1>
    <div className="card">...
    </div>
    <p className="read-the-docs">
      Click on the Vite and React logos to learn more
    </p>
    </>
  );
}
```

2

Tạo ứng dụng React cơ bản

```
import "./App.css";

function Header() {
  return (
    <>
      <h2>Chào mừng bạn đến với thế giới React!</h2>
      <p>
        Hôm nay là <strong>22/1/2024</strong>. Thời gian hiện tại:<" ">
        <strong>19:00</strong>.
      </p>
    </>
  );
}

function App() {
  return (
    <>
      <Header />
    </>
  );
}

export default App;
```

3

React >< Vanilla Javascript

- ✓ *Vanilla JavaScript* là thuật ngữ dùng để chỉ mã JavaScript được viết mà không cần bất kỳ thư viện hoặc khuôn khổ bên ngoài nào

```
<body>
  <div id="root"></div>
  <script>
    // Tạo thẻ <div> để chứa nội dung
    const root = document.getElementById("root");

    // Tạo tiêu đề h1
    const title = document.createElement("h1");
    title.textContent = "Danh sách công việc của tôi";

    // Tạo danh sách không thứ tự (ul)
    const ul = document.createElement("ul");

    // Tạo danh sách công việc (li)
    const tasks = ["Học bài React", "Hoàn thành bài tập lập trình", "Dọn dẹp bàn làm việc"];
    tasks.forEach(task => {
      const li = document.createElement("li");
      li.textContent = task;
      ul.appendChild(li);
    });

    // Tạo thông điệp chào mừng
    const message = document.createElement("p");
    message.textContent = "Chúc bạn hoàn thành công việc tốt nhất hôm nay!";

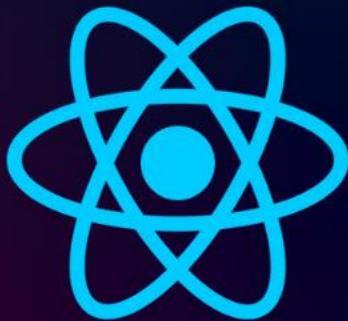
    // Gắn tất cả các phần tử vào thẻ root
    root.appendChild(title);
    root.appendChild(ul);
    root.appendChild(message);
  </script>
</body>
```

```
function MainContent() {
  return (
    <>
      <h1>Danh sách công việc của tôi</h1>
      <ul>
        <li>Học bài React</li>
        <li>Hoàn thành bài tập lập trình</li>
        <li>Dọn dẹp bàn làm việc</li>
      </ul>
      <p>Chúc bạn hoàn thành công việc tốt nhất hôm nay!</p>
    </>
  );
}
```

Danh sách công việc của tôi

- Học bài React
- Hoàn thành bài tập lập trình
- Dọn dẹp bàn làm việc

Chúc bạn hoàn thành công việc tốt nhất hôm nay!



REACT JS

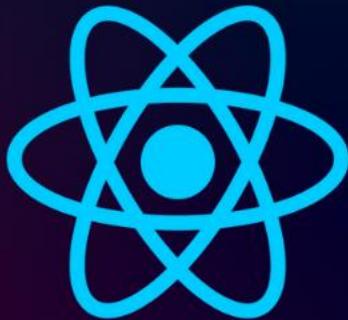


BÀI 13-1

Thực hành

Render hình ảnh, dữ
liệu động





REACT JS

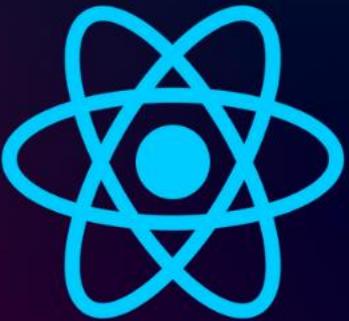


BÀI 13-2

Thực hành

Render hình ảnh, dữ
liệu động





REACT JS



BÀI 14-1

Props

Props Trong React Là Gì ?



1

Mở đầu

- ✓ *Tái sử dụng Component*



Leo Harnent Keorn
Project Manager



Leo Harnent Keorn
Project Manager



Leo Harnent Keorn
Project Manager



Leo Harnent Keorn
Project Manager

- ✓ *Trong các buổi học trước, chúng ta biết rằng, có thể gọi 1 component bao nhiêu lần tùy thích*
- ✓ *Bằng cách sử dụng props: Bạn có thể tùy chỉnh nội dung hoặc hành vi của mỗi lần gọi component*

2

Ví dụ vận dụng

```

1  import pic1 from "./assets/pic1.png";
2  import pic2 from "./assets/pic2.png";
3  import pic3 from "./assets/pic3.png";
4  import pic4 from "./assets/pic4.png";
5
6  function MainContent(props) {
7    return (
8      <li>
9        <img src={props.image} alt={props.title} />
10       <h2>{props.title}</h2>
11       <p>{props.desc}</p>
12     </li>
13   );
14 }
  
```

```

<MainContent
  image={pic1}
  desc="Khối xây dựng giao diện cơ bản - kết hợp nhiều thành phần để tạo nên ứng dụng."
  title="Components"
/>

<MainContent
  image={pic2}
  desc="Kết hợp HTML và JavaScript để tạo giao diện động và mạnh mẽ."
  title="JSX"
/>
  
```

Khái niệm chính trong React



Components

Khối xây dựng giao diện cơ bản - kết hợp nhiều thành phần để tạo nên ứng dụng.



JSX

Kết hợp HTML và JavaScript để tạo giao diện động và mạnh mẽ.



Props

Truyền dữ liệu vào thành phần để làm nó linh hoạt và tái sử dụng.



State

Dữ liệu được React quản lý, khi thay đổi sẽ tự động làm mới giao diện.

3

Xử lý cảnh báo lỗi eslint

```
function MainContent(props) {
  console.log(props);
  return (
    <li>
      <img src={props.image} />
      <h2>{props.title}</h2>
      <p>{props.desc}</p>
    </li>
  );
}
```

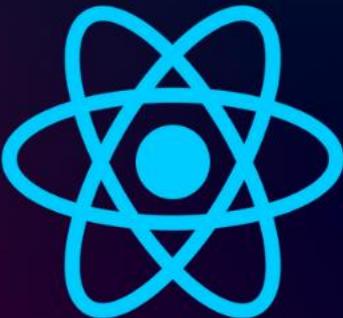
'title' is missing in props validation eslint([react/prop-types](#))
any
View Problem (Alt+F8) Quick Fix... (Ctrl+.)

Lỗi do chưa định nghĩa prop types (**kiểu dữ liệu và yêu cầu** của các props)

- npm install prop-types

```
import PropTypes from "prop-types";
```

```
MainContent.propTypes = {
  image: PropTypes.string.isRequired, // Kiểu chuỗi, bắt buộc phải có
  title: PropTypes.string.isRequired, // Kiểu chuỗi, bắt buộc phải có
  desc: PropTypes.string, // Kiểu chuỗi, không bắt buộc phải có
};
```



REACT JS



BÀI 14-2

Props

Tối ưu hoá code



4

Mở đầu

Cách viết của bài hôm trước vẫn chưa được tối ưu, mặc dù chúng ta đã đạt được mục đích hiển thị dữ liệu khác nhau trong quá trình tái sử dụng component, nhưng vẫn bị lặp code khá nhiều

```
<MainContent
  image={pic1}
  desc="Khối xây dựng giao diện cơ bản - kết hợp nhiều thành phần để tạo nên ứng dụng."
  title="Components"
/>

<MainContent
  image={pic2}
  desc="Kết hợp HTML và JavaScript để tạo giao diện động và mạnh mẽ."
  title="JSX"
/>

<MainContent
  image={pic3}
  desc="Truyền dữ liệu vào thành phần để làm nó linh hoạt và tái sử dụng."
  title="Props"
/>

<MainContent
  image={pic4}
  desc="Dữ liệu được React quản lý, khi thay đổi sẽ tự động làm mới giao diện."
  title="State"
/>
```

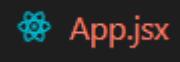
5

Giải quyết vấn đề lặp code

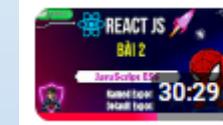
- 1) **Bước 1** : Chuyển data sang mảng → Sau đó chúng ta có thể dễ dàng thay thế nội dung bằng các phần tử của mảng kèm chỉ số index ([data.js](#))

```
export const myData
```

- 2) **Bước 2** : import data



```
import {myData} from "../data.js";
```



2. JavaScript ES6 - Modules
Tim hiểu Named Export,...
Gà Lại Lập Trình

- 3) **Bước 3** : Sử dụng chỉ số index của mảng để truyền dữ liệu

```
<MainContent
  image={pic1}
  desc="Khối xây dựng giao diện cơ bản - kết hợp nhiều thành phần để tạo nên ứng dụng."
  title="Components"
/>
```

```
<MainContent
  image={myData[0].image}
  desc={myData[0].desc}
  title={myData[0].title}
/>
```

6

Tối ưu hóa code



10. Review Javascript Spread Operator - Làm việc với mảng...
Gà Lại Lập Trình

1) Bước 1 : Sử dụng spread operator để "trải" (spread) các thuộc tính của một object

```
<MainContent
  image={myData[0].image}
  desc={myData[0].desc}
  title={myData[0].title}
/>
```

```
<MainContent {...myData[0]} />
```

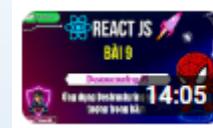
```
console.log({...myData[0]});
```

```
▼ {image: '/src/assets/pic1.png', title: 'Components', desc: 'Khởi xây dựng giao diện cơ bản - kết hợp nhiều thành phần'
  desc: "Khởi xây dựng giao diện cơ bản - kết hợp nhiều thành phần"
  image: "/src/assets/pic1.png"
  title: "Components"
▶ [[Prototype]]: Object
```

```
▼ {image: '/src/assets/pic1.png', title: 'Components', desc: 'Khởi xây dựng giao diện cơ bản - kết hợp nhiều thành phần'
  desc: "Khởi xây dựng giao diện cơ bản - kết hợp nhiều thành phần"
  image: "/src/assets/pic1.png"
  title: "Components"
▶ [[Prototype]]: Object
```

6

Tối ưu hoá code

9. destructuring function - Ứng
dụng Destructuring với đối...

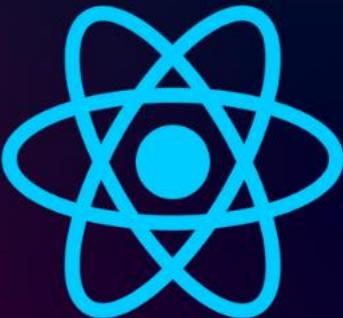
Gà Lại Lập Trình

2) *Bước 2 : Sử dụng **Destructuring** để giảm thiểu gọi : props.*

```
function MainContent(props) {  
  return (  
    <li>  
      <img src={props.image} alt={props.title} />  
      <h2>{props.title}</h2>  
      <p>{props.desc}</p>  
    </li>  
  );  
}
```



```
function MainContent({image, title, desc}) {  
  return (  
    <li>  
      <img src={image} alt={title} />  
      <h2>{title}</h2>  
      <p>{desc}</p>  
    </li>  
  );  
}
```



REACT JS



BÀI 14-3

Props

Tách component



7

Chia nhỏ tệp jsx

*Trong buổi học trước đó, chúng ta đang viết các Component vào chung file App.jsx
Điều này là không nên, vì sau này dự án mở rộng, file App.jsx sẽ có quá nhiều code → khiến nó dài và khó bảo trì*

```
> function MainContent({image, title, desc}) { ...
}

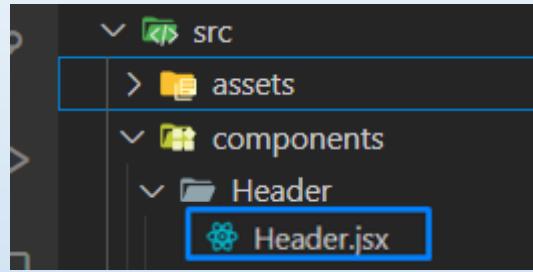
> MainContent.propTypes = { ...
};

function Header() {
>   return ( ...
);
}

function App() {
  return (
    <div>
      <Header />
>      <section id="core-concepts">...
      </section>
    </div>
  );
}
```

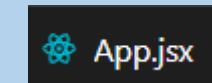
8

Xử lý Header.jsx



The code editor shows the 'Header.jsx' component. The import statement at the top is highlighted with a blue box. The component itself is a simple function returning a blank space.

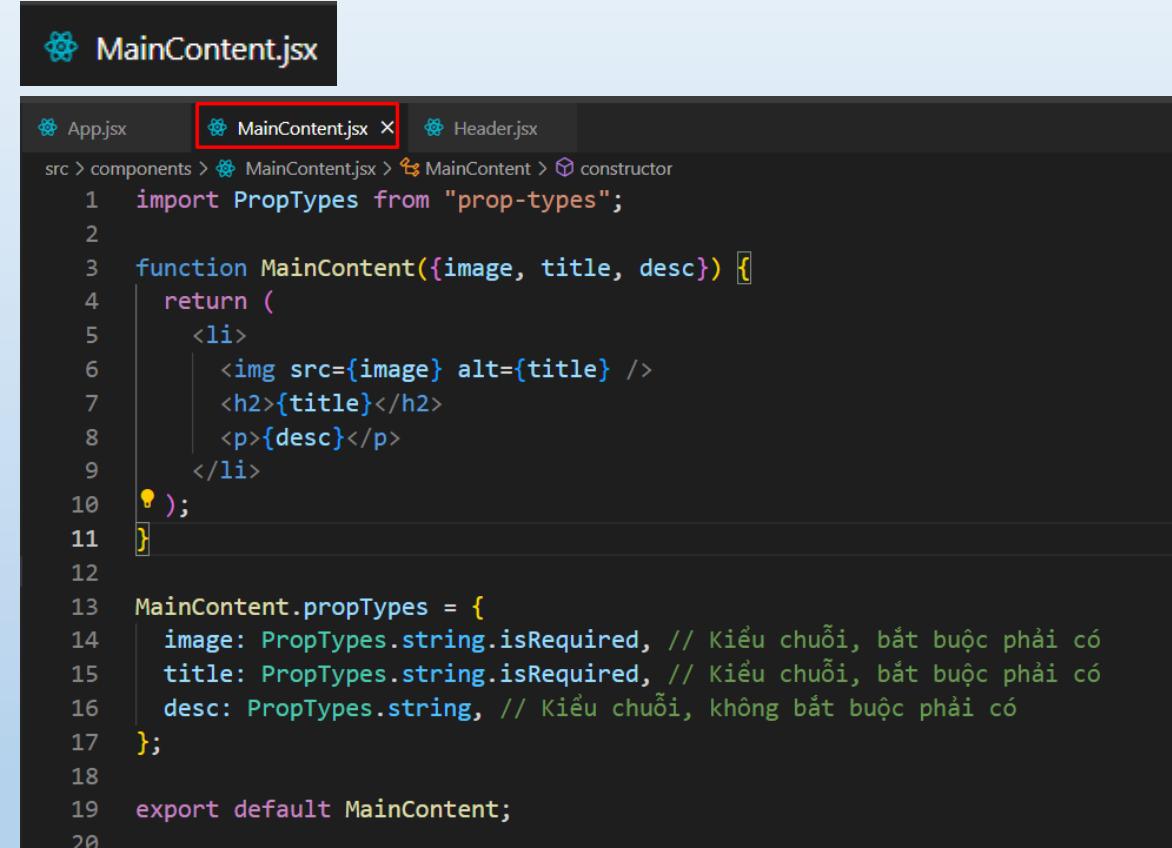
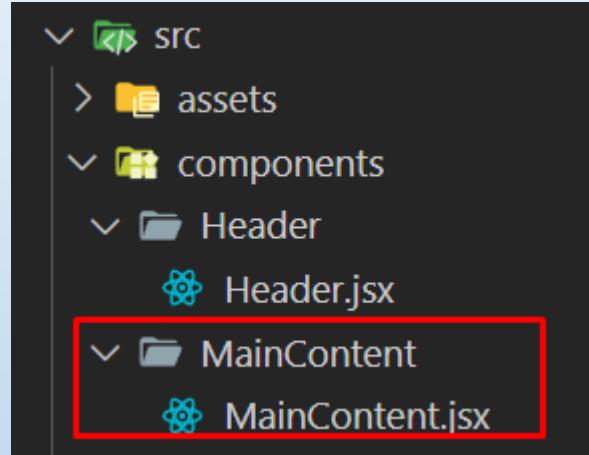
```
1 import logo from "../../assets/logo-tuhoc.png";
2
3
4 function Header() {
5 >   return (
6     );
7 }
8
9 export default Header;
10
```



```
import Header from "./components/Header/Header.jsx";
```

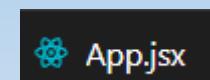
9

Xử lý MainContent.jsx

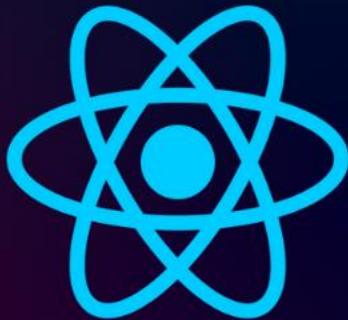


MainContent.jsx

```
App.jsx MainContent.jsx Header.jsx
src > components > MainContent > MainContent > constructor
1 import PropTypes from "prop-types";
2
3 function MainContent({image, title, desc}) {
4     return (
5         <li>
6             <img src={image} alt={title} />
7             <h2>{title}</h2>
8             <p>{desc}</p>
9         </li>
10    );
11 }
12
13 MainContent.propTypes = {
14     image: PropTypes.string.isRequired, // Kiểu chuỗi, bắt buộc phải có
15     title: PropTypes.string.isRequired, // Kiểu chuỗi, bắt buộc phải có
16     desc: PropTypes.string, // Kiểu chuỗi, không bắt buộc phải có
17 };
18
19 export default MainContent;
20
```



```
import MainContent from "./components/MainContent/MainContent.jsx";
```



REACT JS



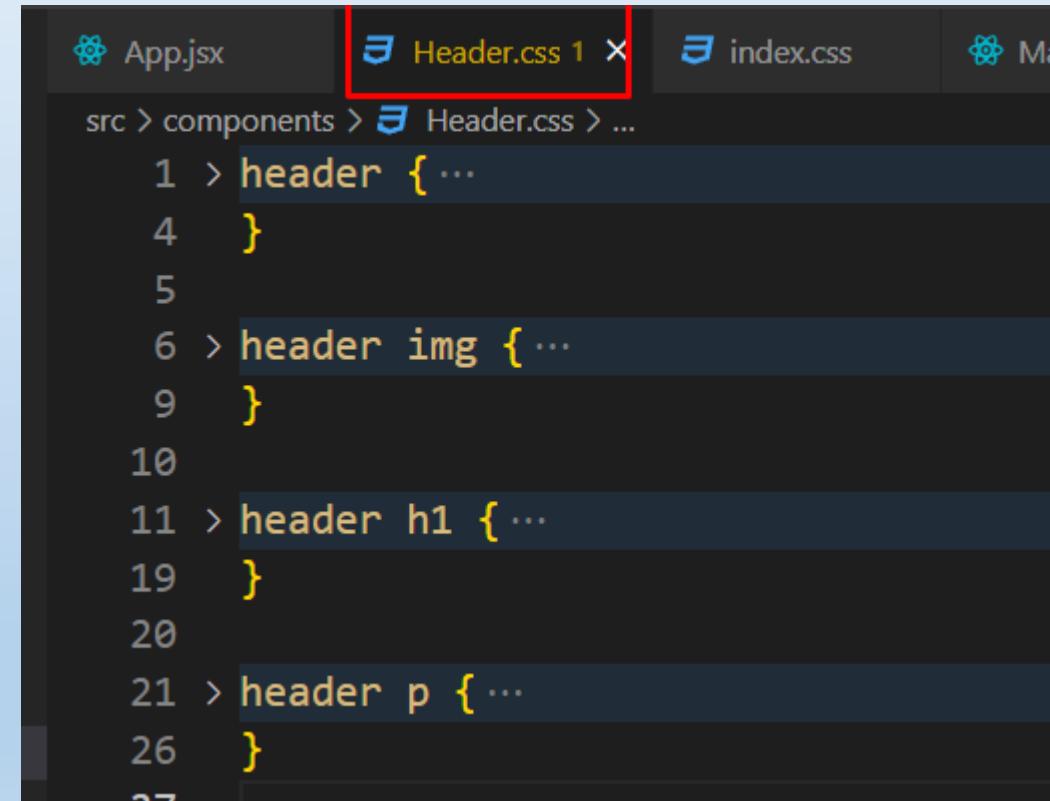
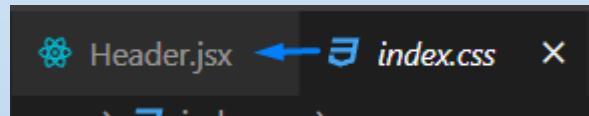
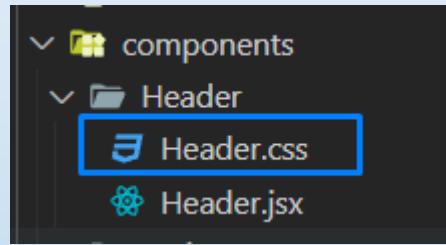
BÀI 14-4

Props

Tối ưu Component



10

Chia nhỏ tệp *Header.css*1) Chia nhỏ tệp *Header.css* từ *index.css*

A screenshot of a code editor showing the content of 'Header.css'. The file contains CSS rules for the 'header' element and its child 'img' and 'h1' elements. The file path 'src > components > Header.css' is visible at the top of the editor.

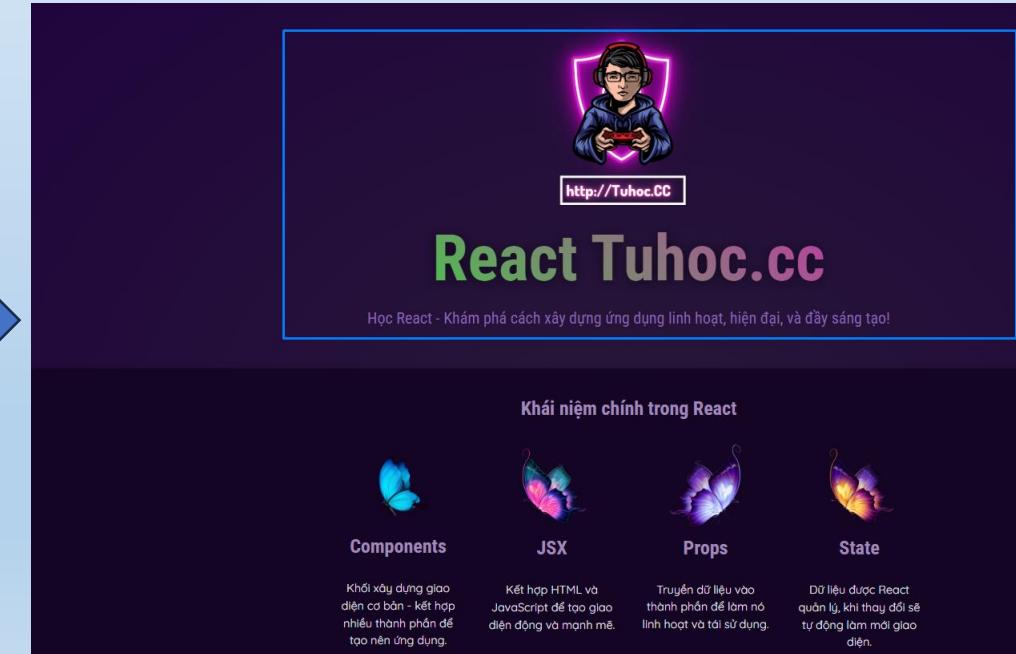
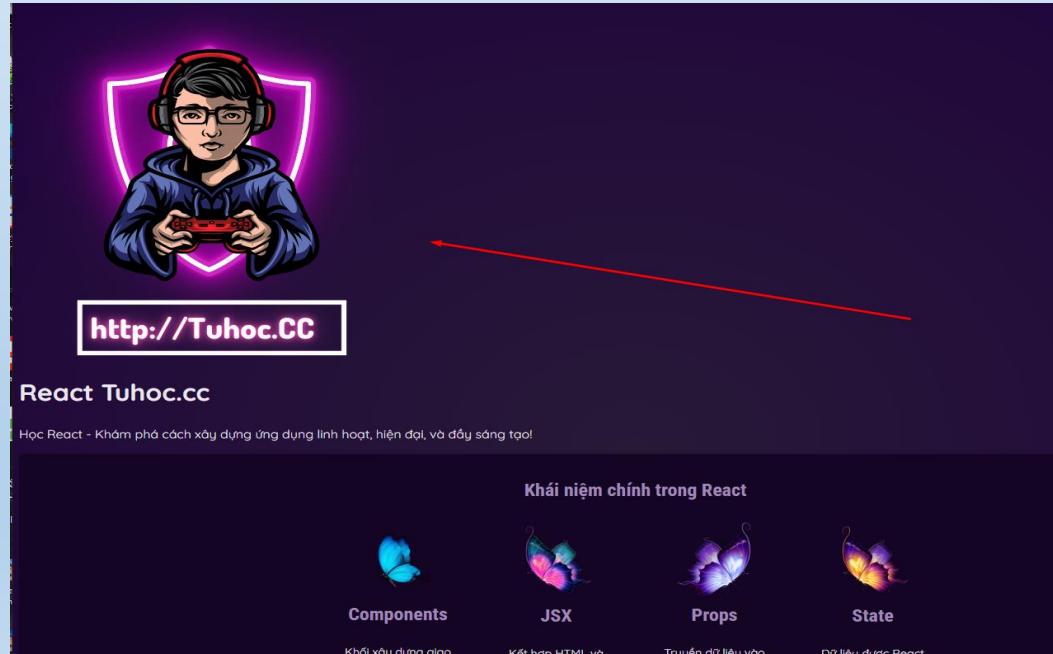
```
1 > header { ...
4   }
5
6 > header img { ...
9   }
10
11 > header h1 { ...
19   }
20
21 > header p { ...
26   }
```

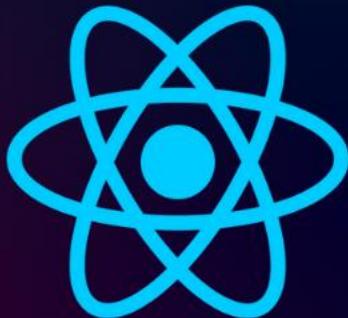
11

Chia nhỏ tệp *Header.css*

2) Import *Header.css* vào file *Header.jsx*

```
Header.jsx X
src > components > Header.jsx > ...
1 import logo from "../assets/logo-tuhoc.png";
2 import "./Header.css";
3
```





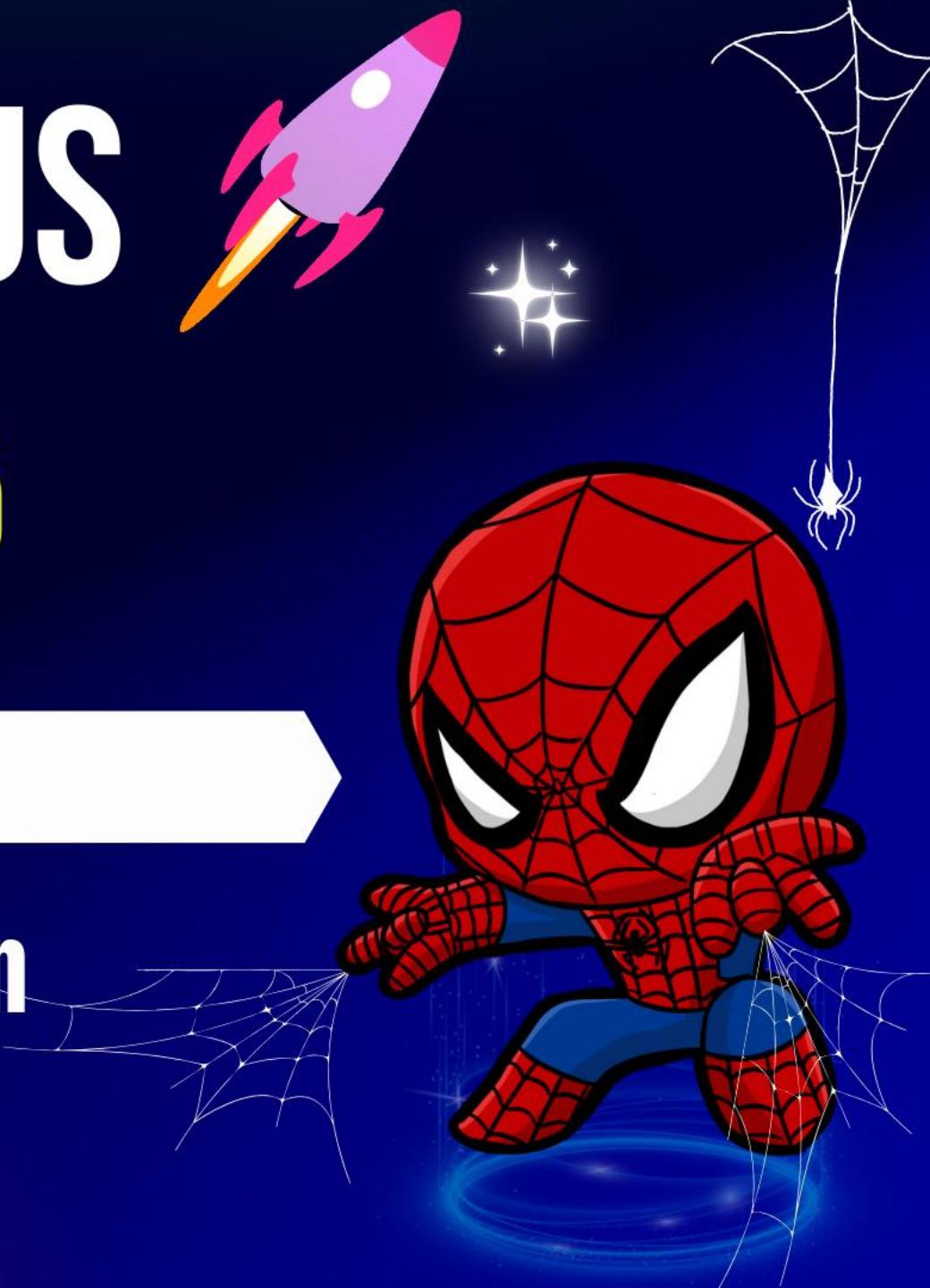
REACT JS



BÀI 14-5

Props

props.children

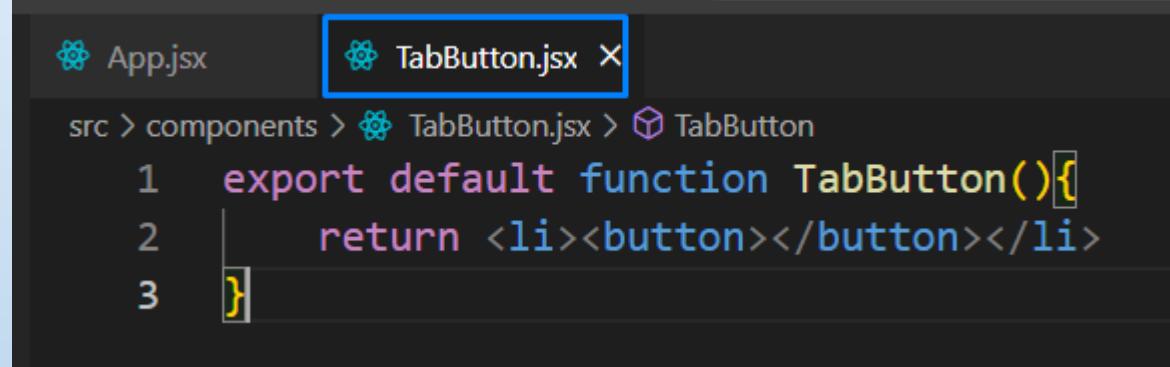
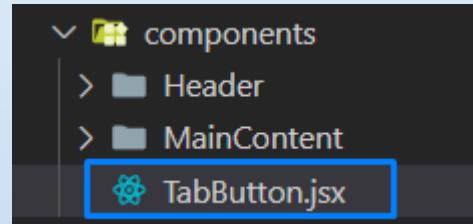


12

children props

```
<li>
| <button>Props</button>
</li>
```

- ✓ *children props*, tự động chứa mọi thứ bên trong cặp thẻ mở và đóng của component.



```
App.jsx TabButton.jsx ×
src > components > TabButton.jsx > TabButton
1 export default function TabButton(){
2   return <li><button></button></li>
3 }
```

```
export default function TabButton(props) {
  return (
    <li>
      <button>{props.children}</button>
    </li>
  );
}
```

```
<menu>
| <TabButton>Button1</TabButton>
</menu>
```

props.children sẽ hiển thị nội dung mà bạn đặt giữa cặp thẻ `<TabButton></TabButton>` khi sử dụng component này.

12

children rops

- ✓ Sử dụng **Destructuring** để giảm thiểu gọi : props.

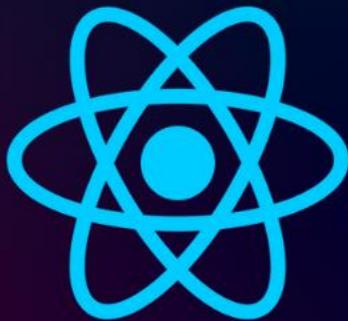
```
export default function TabButton({children}) {
  return (
    <li>
      <button>{children}</button>
    </li>
  );
}
```

```
export default function TabButton({label}) {
  return (
    <li>
      <button>{label}</button>
    </li>
  );
}
```

```
<menu>
  <TabButton>Button1</TabButton>
</menu>
```

```
<menu>
  <TabButton label="button1"></TabButton>
</menu>
```

*React tự động hiểu nội dung giữa
thẻ mở và thẻ đóng là props.children*



REACT JS



BÀI 14-6

React onClick



13

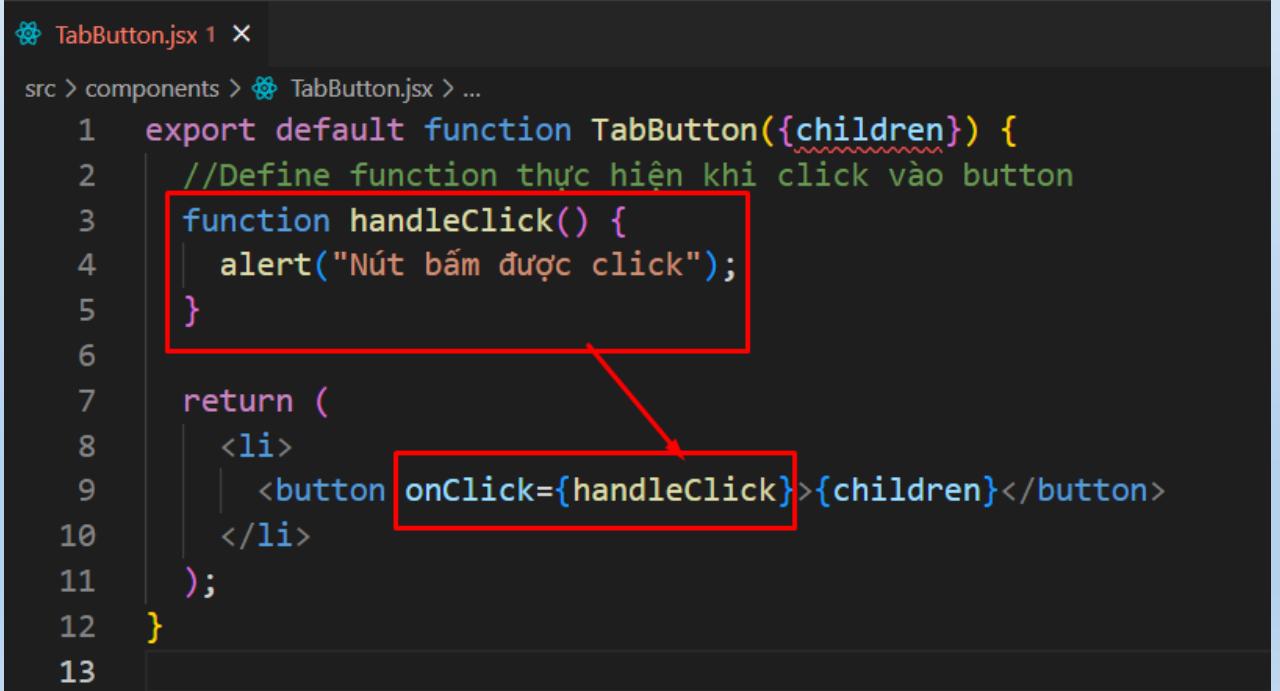
React onClick

```
// Trong js chúng ta làm như sau để lắng nghe sự kiện lên nút, và thực hiện công việc nào đó
const btn = document.querySelector("button");
btn.addEventListener("click", ()=>{
  //làm việc j đó
})
```

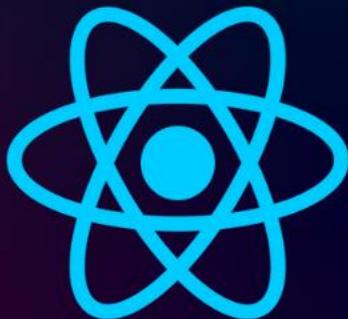
onClick và các event handler khác: Là các props đặc biệt được React hỗ trợ để gắn sự kiện vào

Bạn cần truyền một hàm xử lý sự kiện vào onClick để xác định hành động khi người dùng click.

Lưu ý chúng ta truyền tên hàm handleClick, chứ không truyền hàm thực thi handleClick()



```
TabButton.jsx 1 ×
src > components > TabButton.jsx > ...
1  export default function TabButton({children}) {
2    //Define function thực hiện khi click vào button
3    function handleClick() {
4      alert("Nút bấm được click");
5    }
6
7    return (
8      <li>
9        <button onClick={handleClick}>{children}</button>
10       </li>
11     );
12   }
13 }
```



REACT JS



BÀI 14-7

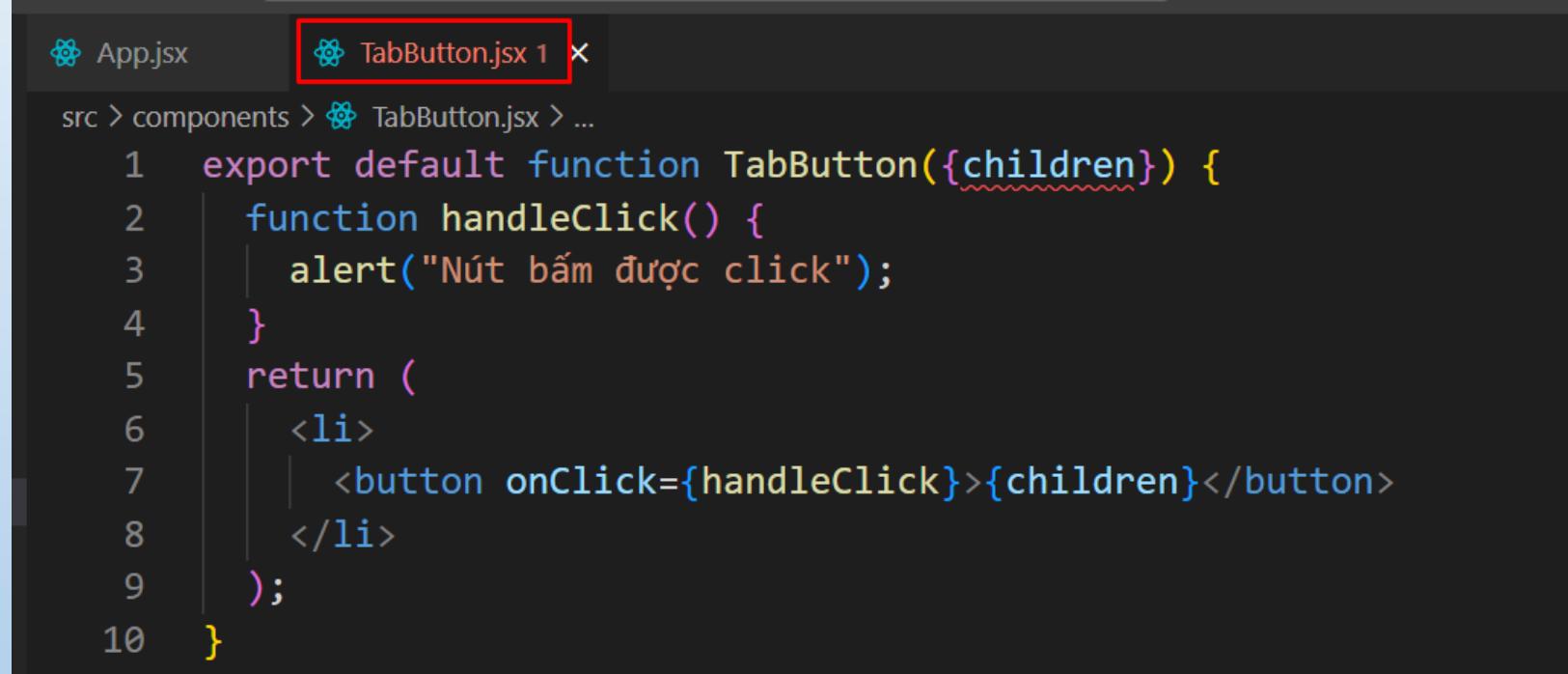
Truyền sự kiện từ Component
cha vào Component con



14

Truyền hàm sự kiện từ component cha vào TabButton

Trong bài trước chúng ta đã tạo ra được 4 nút bấm



```
App.jsx TabButton.jsx 1 x
src > components > TabButton.jsx > ...
1  export default function TabButton({children}) {
2    function handleClick() {
3      alert("Nút bấm được click");
4    }
5    return (
6      <li>
7        <button onClick={handleClick}>{children}</button>
8      </li>
9    );
10 }
```

Hàm **handleClick** không nhận được giá trị hoặc nội dung từ bên ngoài vì nó bị giới hạn trong phạm vi của component TabButton

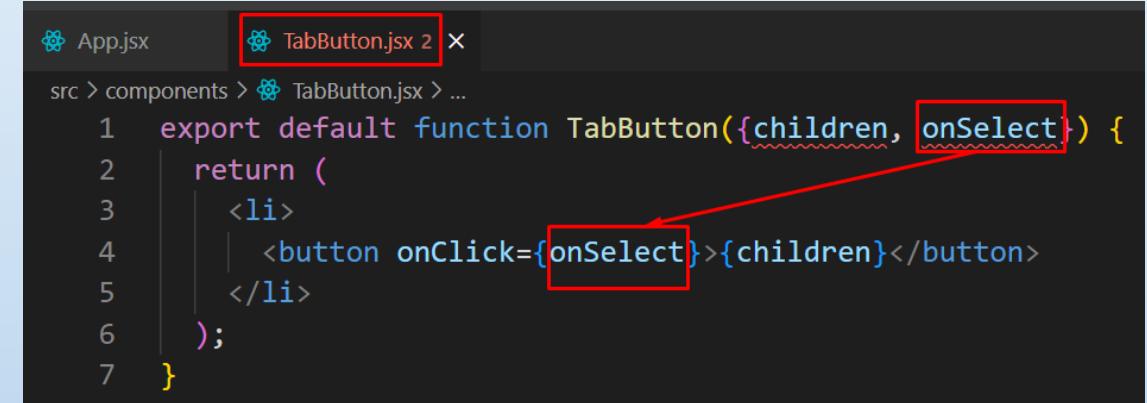
14

Truyền hàm sự kiện từ component cha vào TabButton

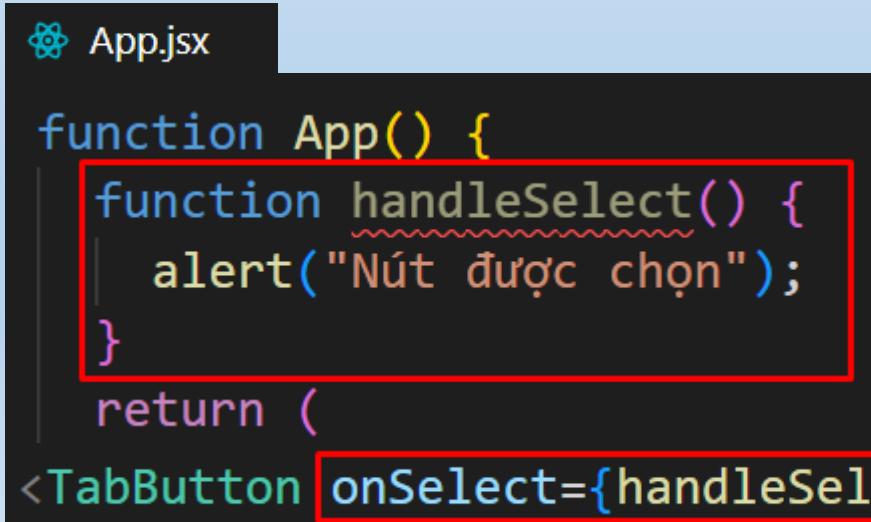
- Định nghĩa hàm xử lý sự kiện tại App.jsx (component cha).
- Truyền hàm xử lý sự kiện đó qua prop vào TabButton



```
App.jsx TabButton.jsx 1
src > components > TabButton.jsx > ...
1  export default function TabButton({children}) {
2    function handleClick() {
3      alert("Nút bấm được click");
4    }
5    return (
6      <li>
7        <button onClick={handleClick}>{children}</button>
8      </li>
9    );
10 }
```

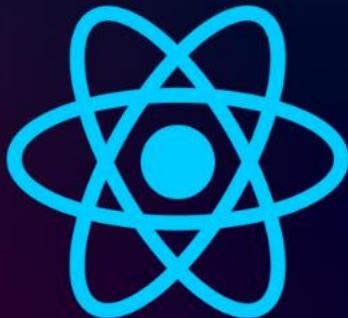


```
App.jsx TabButton.jsx 2
src > components > TabButton.jsx > ...
1  export default function TabButton({children, onSelect}) {
2    return (
3      <li>
4        <button onClick={onSelect}>{children}</button>
5      </li>
6    );
7 }
```



```
App.jsx
function App() {
  function handleSelect() {
    alert("Nút được chọn");
  }
  return (
    <TabButton onSelect={handleSelect}>Components</TabButton>
  );
}
```

Khác với *children* là prop đặc biệt, cần
ghi đúng chính tả, prop tùy chỉnh
onSelect có thể đặt tùy ý



REACT JS



BÀI 14-8

Xử Lý Sự Kiện Trong React
Với Arrow Function



15

Sử Dụng Arrow Function trong Xử Lý Sự Kiện

App.jsx

```
function App() {
  function handleSelect() {
    alert("Nút được chọn");
  }
  return (
    <TabButton onSelect={handleSelect}>Components</TabButton>
  );
}
```

- ✓ Hiện tại, khi click vào mọi nút, đều gọi hàm handleSelect giống nhau
- ✓ Chúng ta mong muốn, khi click vào nút, thì phải biết nút nào được click và thực hiện nhiệm vụ riêng

15

Sử Dụng Arrow Function trong Xử Lý Sự Kiện

{/* prettier-ignore */}

Sử dụng function (ở đây sử dụng arrow fun cho ngắn gọn) để kiểm soát(thêm logic nếu cần – khi thoả mãn mới thực thi hàm)

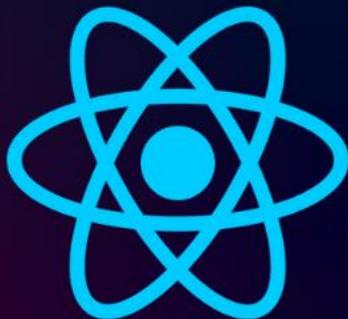
```
function handleSelect() {  
  alert("Nút được chọn");  
}
```

```
function handleSelect(selectedButton) {  
  alert(`#${selectedButton} được chọn`);  
}
```

```
<TabButton onSelect=[handleSelect]>Components</TabButton>
```

```
<TabButton onSelect={()=>{handleSelect('components')}}>Components</TabButton>  
<TabButton onSelect={()=>{handleSelect('jsx')}}>JSX</TabButton>  
<TabButton onSelect={()=>{handleSelect('props')}}>Props</TabButton>  
<TabButton onSelect={()=>{handleSelect('state')}}>State</TabButton>
```

Thông qua ví dụ trên chúng ta có thể truyền dữ liệu động, khi nhấn vào từng nút (dữ liệu sẽ hiển thị khác nhau trên từng nút)



REACT JS



BÀI 14-9

useState react

Tìm hiểu về State và hook trong
react



16

State cơ bản với React

Bây giờ chúng ta giả lập nội dung hiển thị bên dưới nút

Examples

Components JSX Props State

Nội dung được hiển thị

```
function App() {
  let tabContent = "Nội dung được hiển thị"

  function handleSelect(selectedButton) {
    alert(` ${selectedButton} được chọn`);
  }

  return (
    <div>
      <h1>React</h1>
      <p>This is a simple React app.</p>
      <button onClick={handleSelect}>Click me</button>
      <br/>
      {tabContent}
    </div>
  );
}
```

```
<section id="examples">
  <h2>Examples</h2>
  <menu>
    <TabButton onSelect={()=>{handleSelect('components')}}>Components</TabButton>
    <TabButton onSelect={()=>{handleSelect('jsx')}}>JSX</TabButton>
    <TabButton onSelect={()=>{handleSelect('props')}}>Props</TabButton>
    <TabButton onSelect={()=>{handleSelect('state')}}>State</TabButton>
  </menu>
  {tabContent}
</section>
```

16

State cơ bản với React

Trong function: mặc dù chúng ta thay đổi giá trị cho tabContent nhưng chúng vẫn không được update trên giao diện

```
function handleSelect(selectedButton) {
  alert(` ${selectedButton} được chọn`);
  tabContent = selectedButton;
}
```

Lý do là **các hàm chỉ được gọi 1 lần khi chạy chương trình**, khi chúng ta click vào nút, thì hàm **handleSelect** được gọi, **nhưng App()** thì không được gọi lại vì vậy nó không gán lại giá trị cho **tabContent**

```
function App() { X
  let tabContent = "Nội dung được hiển thị"

  function handleSelect(selectedButton) {
    alert(` ${selectedButton} được chọn`);
    tabContent = selectedButton;
  }
}
```

16

State cơ bản với React

Để cập nhật lại 1 thành phần chúng ta sử dụng `useState` - (State có nghĩa là trạng thái, trong ngữ cảnh này, nó có nghĩa là trạng thái của dữ liệu, thay đổi từ A → B)

Tất cả các thành phần bắt đầu bằng tiền tố `use...` được gọi là **react Hook**
Hook bản chất là các hàm tính năng được react thiết kế sẵn

```
App.jsx  x
src > App.jsx > App
1 import {useState} from "react";
2
3 import MainContent from "./components/MainContent/MainContent.jsx"
4 import Header from "./components/Header/Header.jsx";
5 import {myData} from "../data.js";
6 import TabButton from "./components/TabButton.jsx";
7
8 function App() {
9   useState();
10  let tabContent = "Nội dung được hiển thị";
11
12  function handleSelect(selectedButton) {
13    alert(`${selectedButton} được chọn`);
14    tabContent = selectedButton;
15  }
16
17  return (
18    <div>
19      <Header />
20      <MainContent tabContent={tabContent}>
21        <TabButton handleSelect={handleSelect} />
22      </MainContent>
23    </div>
24  );
25}
```

16

State cơ bản với React

1. Khi sử dụng bắt buộc phải gọi tại cấp cao nhất của hàm thành phần

```
8  function App() {  
9    useState(); ✓  
10   let tabContent = "Nội dung được hiển thị";  
11  
12   function handleSelect(selectedButton) {  
13     alert(`#${selectedButton} được chọn`);  
14     tabContent = selectedButton;  
15   }  
16   return ()
```

```
function App() {  
  let tabContent = "Nội dung được hiển thị"  
  
  function handleSelect(selectedButton) {  
    useState(); ✗  
    alert(`#${selectedButton} được chọn`);  
    tabContent = selectedButton;  
  }  
  return ()
```

2. Không được gọi bên trong các câu lệnh if, while.....

```
const isOnline = true;  
if(isOnline){  
  useState() ✗  
  //thực thi điều gì đó  
}
```

16

State cơ bản với React

- ✓ *useState nhận một giá trị khởi tạo (initState) làm đối số*
- ✓ *Trả về một mảng gồm hai phần tử: giá trị hiện tại của state và một hàm để cập nhật state đó*

destructuring trong JS

```
const [state, setState] = useState(initState);
```

↑
Giá trị hiện tại

↑
Function để cập nhật state

↑
Giá trị khởi tạo

16

State cơ bản với React

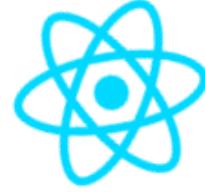
✓ Ví dụ quen thuộc khi khởi tạo vite

```
const [count, setCount] = useState(0)
```

Giá trị hiện tại

Function để cập nhật state

Giá trị khởi tạo



Vite + React

count is 0

```
<button onClick={() => setCount((count) => count + 1)}>  
| count is {count}  
</button>
```

16

State cơ bản với React

App.jsx

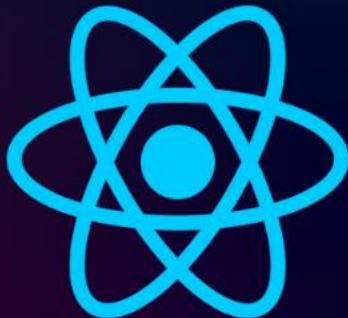
```
function App() {
  const [selectedTopic, setSelectedTopic] = useState("Vui lòng click vào nút");

  function handleSelect(selectedButton) {
    setSelectedTopic(selectedButton);
  }
  return (
    <>
```

TabButton.jsx

```
export default function TabButton({children, onSelect}) {
  return (
    <li>
      <button onClick={onSelect}>{children}</button>
    </li>
  );
}
```

```
<section id="examples">
  <h2>Examples</h2>
  <menu>
    <TabButton onSelect={()=>{handleSelect('components')}}>Components</TabButton>
    <TabButton onSelect={()=>{handleSelect('jsx')}}>JSX</TabButton>
    <TabButton onSelect={()=>{handleSelect('props')}}>Props</TabButton>
    <TabButton onSelect={()=>{handleSelect('state')}}>State</TabButton>
  </menu>
  {selectedTopic}
</section>
```



REACT JS



BÀI 14-10

useState react

Bài tập React 01



17

State – Bài Tập react 01

Hãy viết một ứng dụng React hiển thị lời chào dựa trên thời gian trong ngày.
Sử dụng useState để quản lý trạng thái của lời chào. Ứng dụng cần:

Hiển thị một thông báo mặc định là "Chào bạn!".

Khi người dùng nhấn vào nút "Cập nhật lời chào", lời chào sẽ thay đổi:

"Chào buổi sáng!" nếu thời gian hiện tại là buổi sáng (5h - 12h).

"Chào buổi chiều!" nếu thời gian hiện tại là buổi chiều (12h - 18h).

"Chào buổi tối!" nếu thời gian hiện tại là buổi tối (sau 18h).

```
//const currentHour = new Date().getHours();
const currentHour = 19;
```

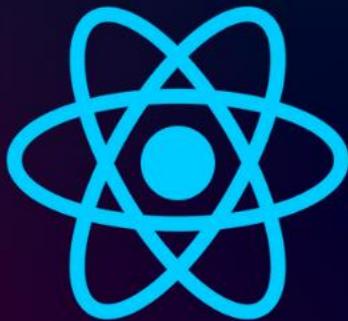
Chào bạn!

Cập nhật lời chào



Chào buổi tối!

Cập nhật lời chào



REACT JS



BÀI 14-11

useState react



**Ứng Dụng State
Để Hiển Thị Dữ Liệu Thực**



18

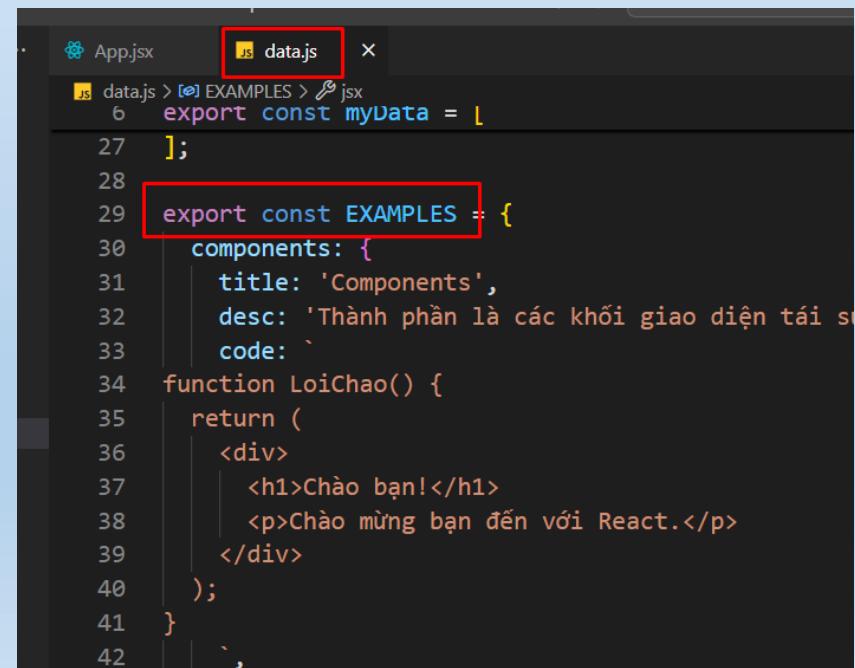
Display Real Dynamic Values

Trong bài này chúng ta sẽ hiển thị dữ liệu thực tế hơn 1 chút so với ví dụ trước đó

```
import {EXAMPLES} from "../data.js";
```

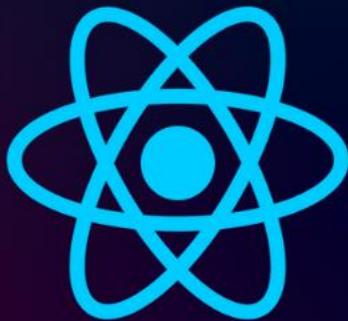
```
const [selectedTopic, setSelectedTopic] = useState('components');
```

```
<div id="tab-content">
  <h3>{EXAMPLES[selectedTopic].title}</h3>
  <p>{EXAMPLES[selectedTopic].desc}</p>
  <pre>
    <code>
      {EXAMPLES[selectedTopic].code}
    </code>
  </pre>
</div>
```



The screenshot shows a code editor with two tabs: "App.jsx" and "data.js". The "data.js" tab is active, displaying the following code:

```
data.js > EXAMPLES > jsx
6 export const myData = [
  {
    title: 'Hello World',
    desc: 'This component displays a simple greeting message.'
  }
];
29 export const EXAMPLES = [
  {
    components: {
      title: 'Components',
      desc: 'Thành phần là các khối giao diện tái sử dụng.',
      code: `function LoiChao() {
      return (
        <div>
          <h1>Chào bạn!</h1>
          <p>Chào mừng bạn đến với React.</p>
        </div>
      );
    }
  }
];
```



REACT JS



BÀI 14-12

useState react

**Hiển Thị Nội Dung
Theo Điều Kiện**



19

Hiển thị dữ liệu mặc định

Đôi khi ta muốn khi người dùng chưa lựa chọn, sẽ có 1 thông tin mặc định nào đó ví dụ đơn giản : *Vui lòng click vào nút để lựa chọn 1 chủ đề*

```
const [selectedTopic, setSelectedTopic] = useState();
```

```
<div id="tab-content">
  <h3>{EXAMPLES[selectedTopic].title}</h3>
  <p>{EXAMPLES[selectedTopic].desc}</p>
  <pre>
    <code>
      {EXAMPLES[selectedTopic].code}
    </code>
  </pre>
</div>
```

```
/* dùng toán tử 3 ngôi */
!selectedTopic ? (
  <p>Vui lòng click vào nút để lựa chọn 1 chủ đề </p>
) : (
  <div id="tab-content">
    <h3>{EXAMPLES[selectedTopic].title}</h3>
    <p>{EXAMPLES[selectedTopic].desc}</p>
    <pre>
      <code>{EXAMPLES[selectedTopic].code}</code>
    </pre>
  </div>
)
```

19

Hiển thị dữ liệu mặc định

Cách 2: *Dùng toán tử AND - &&*

Cấu trúc này có vẻ dễ đọc hơn toán tử 3 ngôi

```
const [selectedTopic, setSelectedTopic] = useState();  
  
/* Cách 2 && */  
{!selectedTopic && <p>Vui lòng click vào nút để lựa chọn 1 chủ đề </p>}  
{selectedTopic && (  
  <div id="tab-content">  
    <h3>{EXAMPLES[selectedTopic].title}</h3>  
    <p>{EXAMPLES[selectedTopic].desc}</p>  
    <pre>  
      <code>{EXAMPLES[selectedTopic].code}</code>  
    </pre>  
  </div>  
)}
```

19

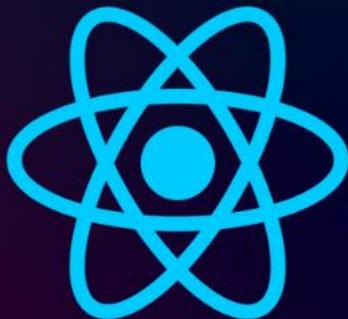
Hiển thị dữ liệu mặc định

Cách 3: *Khai báo biến riêng biệt – giúp cho mã jsx clean hơn !*

```
const [selectedTopic, setSelectedTopic] = useState();
```

```
let tabContent = <p>Vui lòng click vào nút để lựa chọn 1 chủ đề </p>;
if (selectedTopic) {
  tabContent = (
    <div id="tab-content">
      <h3>{EXAMPLES[selectedTopic].title}</h3>
      <p>{EXAMPLES[selectedTopic].desc}</p>
      <pre>
        <code>{EXAMPLES[selectedTopic].code}</code>
      </pre>
    </div>
  );
}
```

```
/* Cách 3: Khai báo biến riêng biệt */
{tabContent}
```



REACT JS



BÀI 14-13

useState react

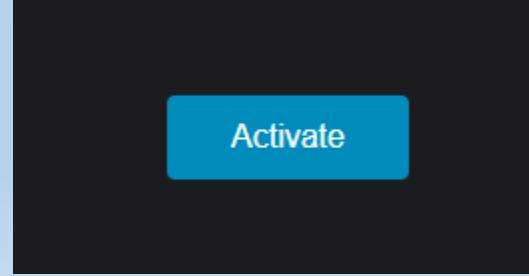
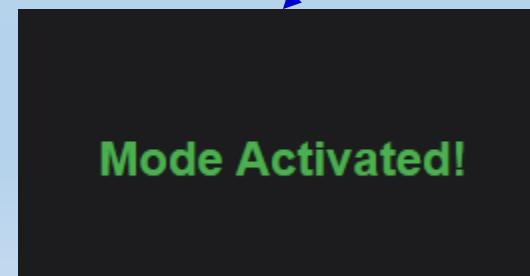
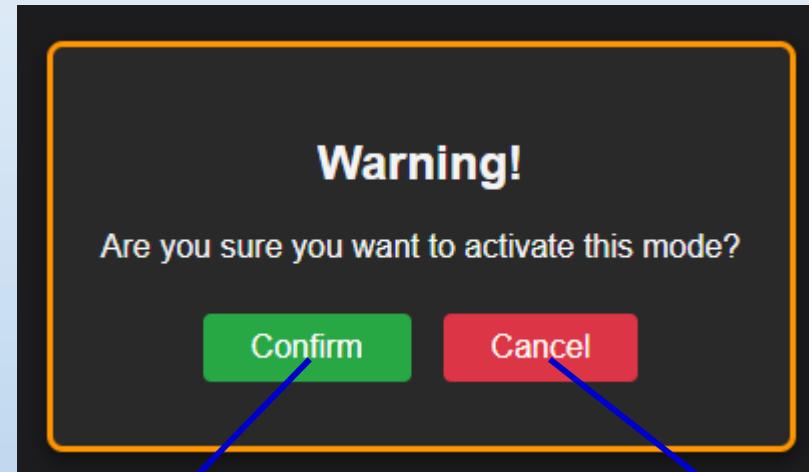
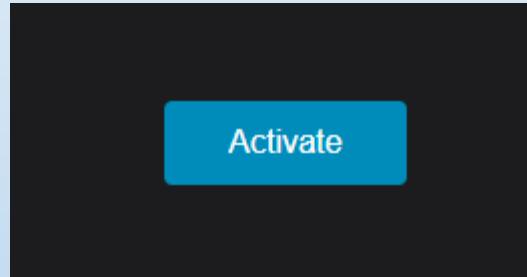


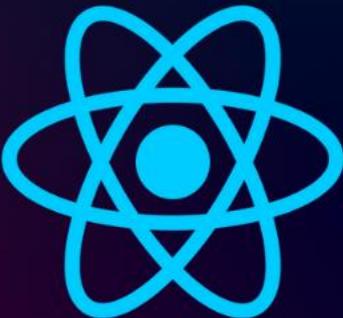
**Bài tập React js 02
Hiển Thị Hộp Cảnh Báo**

20

Bài tập useState react 02

Bài tập react 02 : Xây dựng một phần của ứng dụng web chịu trách nhiệm hiển thị cảnh báo





REACT JS



BÀI 14-14

useState react

**Thêm Class Active
Cho button**



21

Dynamic Styling for button

Chúng ta muốn khi chọn nút, thì nút đó sẽ có style sáng lên, để người dùng biết nút nào đang được chọn (Dựa trên class active đã được viết sẵn css)



```
TabButton.jsx 2 ×  
src > components > TabButton.jsx > TabButton  
1  export default function TabButton({children, onSelect}) {  
2    return (  
3      <li>  
4        <button className="active" onClick={onSelect}>  
5          {children}  
6        </button>  
7      </li>  
8    );  
9  }  
10 }
```



21

Dynamic Styling for button

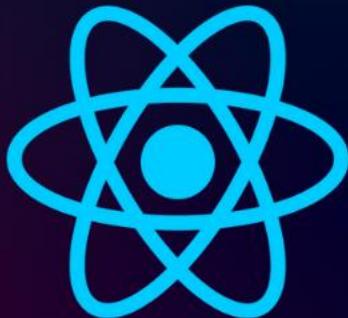
Để có thể thêm được dữ liệu động chúng ta thêm 1 prop - *isSelected*

 TabButton.jsx

```
TabButton.jsx 3 ×  
src > components > TabButton.jsx > ...  
1  export default function TabButton({children, onSelect, isSelected}) {  
2    return (  
3      <li>  
4        <button className={isSelected? 'active' :undefined} onClick={onSelect}>  
5          {children}  
6        </button>  
7      </li>  
8    );  
9  }
```

 App.jsx

```
<menu>  
  <TabButton  
    isSelected={selectedTopic === 'components'}  
    onSelect={()=>{handleSelect('components')}}  
    >Components  
  </TabButton>  
  <TabButton onSelect={()=>{handleSelect('jsx')}}>JSX</TabButton>  
  <TabButton onSelect={()=>{handleSelect('props')}}>Props</TabButton>  
  <TabButton onSelect={()=>{handleSelect('state')}}>State</TabButton>  
</menu>
```



REACT JS



BÀI 14-15

useState react



Bài tập React 03

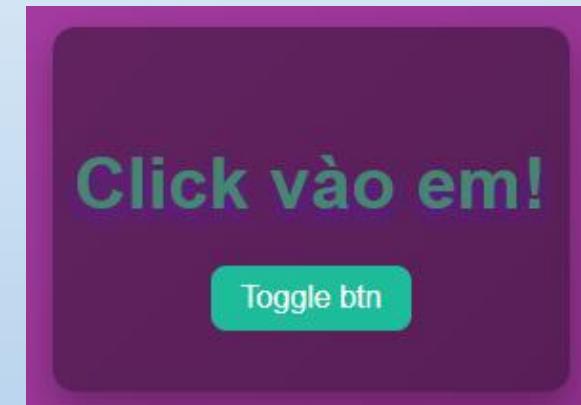
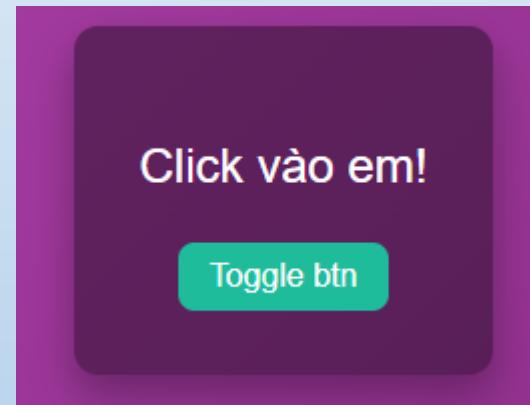
Thêm class active cho phần tử

21

Bài tập useState react 03

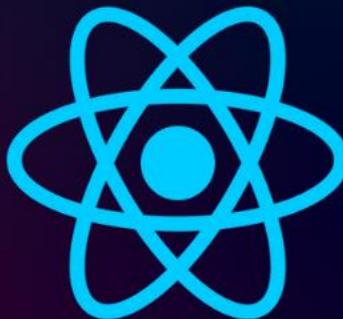
Bài tập react 03 :

Hãy thêm class active vào phần tử `<p>` chứa nội dung "Click vào em!" khi người dùng nhấn nút "Toggle btn". Class sẽ được thêm vào và loại bỏ nếu nhấn lại.



```
▼ <div class="container">
  <p>Click vào em!</p>
  <button>Toggle btn</button> == $0
</div>
```

```
<div class="container">
  <p class="active">Click vào em!</p>
  <button>Toggle btn</button> == $0
</div>
```



REACT JS



BÀI 14-16

map () in react

**Tối ưu hóa mảng
trong React**



21 Hoàn thiện dự án cơ bản

Vẫn có 1 vài phần chúng ta có thể tối ưu, chúng ta đang lấy theo index cứng của mảng, và điều gì sẽ xảy ra nếu số phần tử này của mảng thay đổi ?? :

```
<ul>
  <MainContent {...myData[0]} />
  <MainContent {...myData[1]} />
  <MainContent {...myData[2]} />
  <MainContent {...myData[3]} />
</ul>
```



✖ ► Warning: Failed prop type: The prop `image` is marked as required in `MainContent`, but its value is `undefined`. at MainContent (<http://localhost:5173/src/components/MainContent/MainContent.jsx:18:13>) at App (<http://localhost:5173/src/App.jsx?t=1738295721628:26:45>)

21

Hoàn thiện dự án cơ bản

JSX có khả năng hiển thị mảng

```
{["xin chào", "tạm biệt", "hẹn gặp lại"]}

{[<p>xin chào</p>, <p>tạm biệt</p>, <p>hẹn gặp lại</p>]}
```

```
xin chào
tạm biệt
hẹn gặp lại
```

21

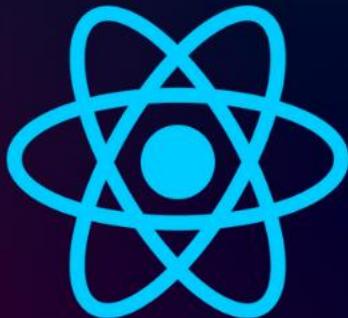
Hoàn thiện dự án cơ bản

Chúng ta có thể dùng phương thức map để giải quyết vấn đề này với mảng data

```
<ul>
  <MainContent {...myData[0]} />
  <MainContent {...myData[1]} />
  <MainContent {...myData[2]} />
  <MainContent {...myData[3]} />
</ul>
```



```
<ul>
  {myData.map(item) =>
    <MainContent key={selectedTopic} {...item} />
  })
</ul>
```



REACT JS



BÀI 14-17

map () in react

Bài tập react 04



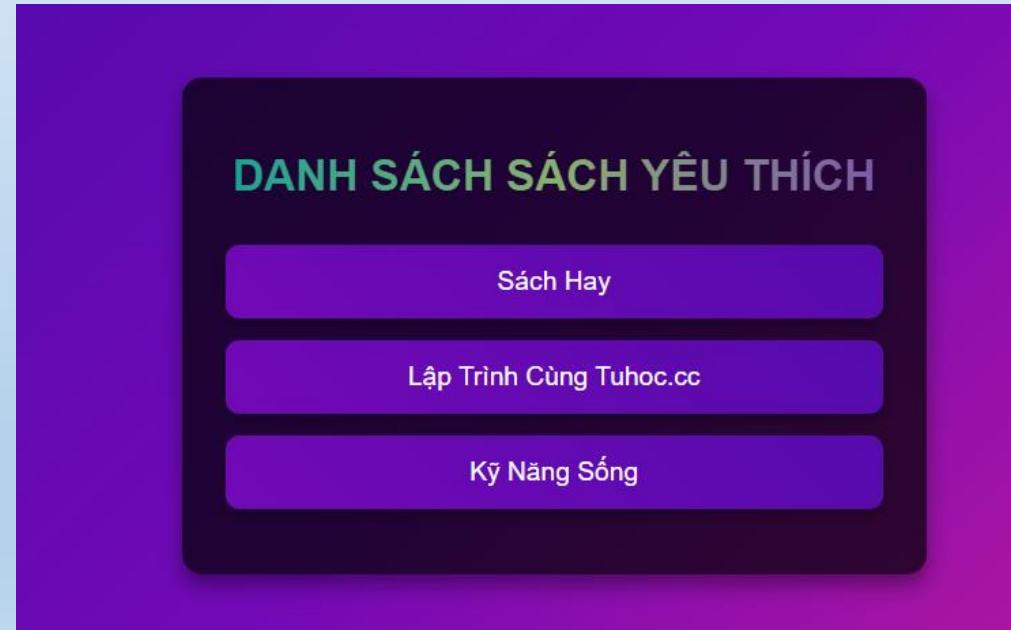
22

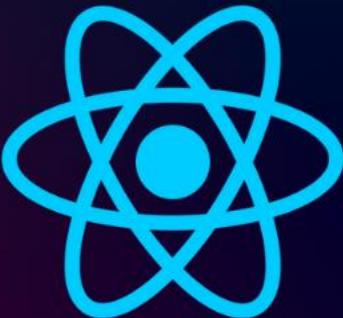
Bài tập react 04 – ôn tập map

Bài tập react 04 :

Tạo một mảng các cuốn sách yêu thích (ví dụ: "Sách Hay", "Lập Trình Cùng Tuhoc.CC", "Kỹ Năng Sống").

Sử dụng phương thức map để duyệt qua mảng và hiển thị mỗi cuốn sách qua component Book





REACT JS



BÀI 14-18

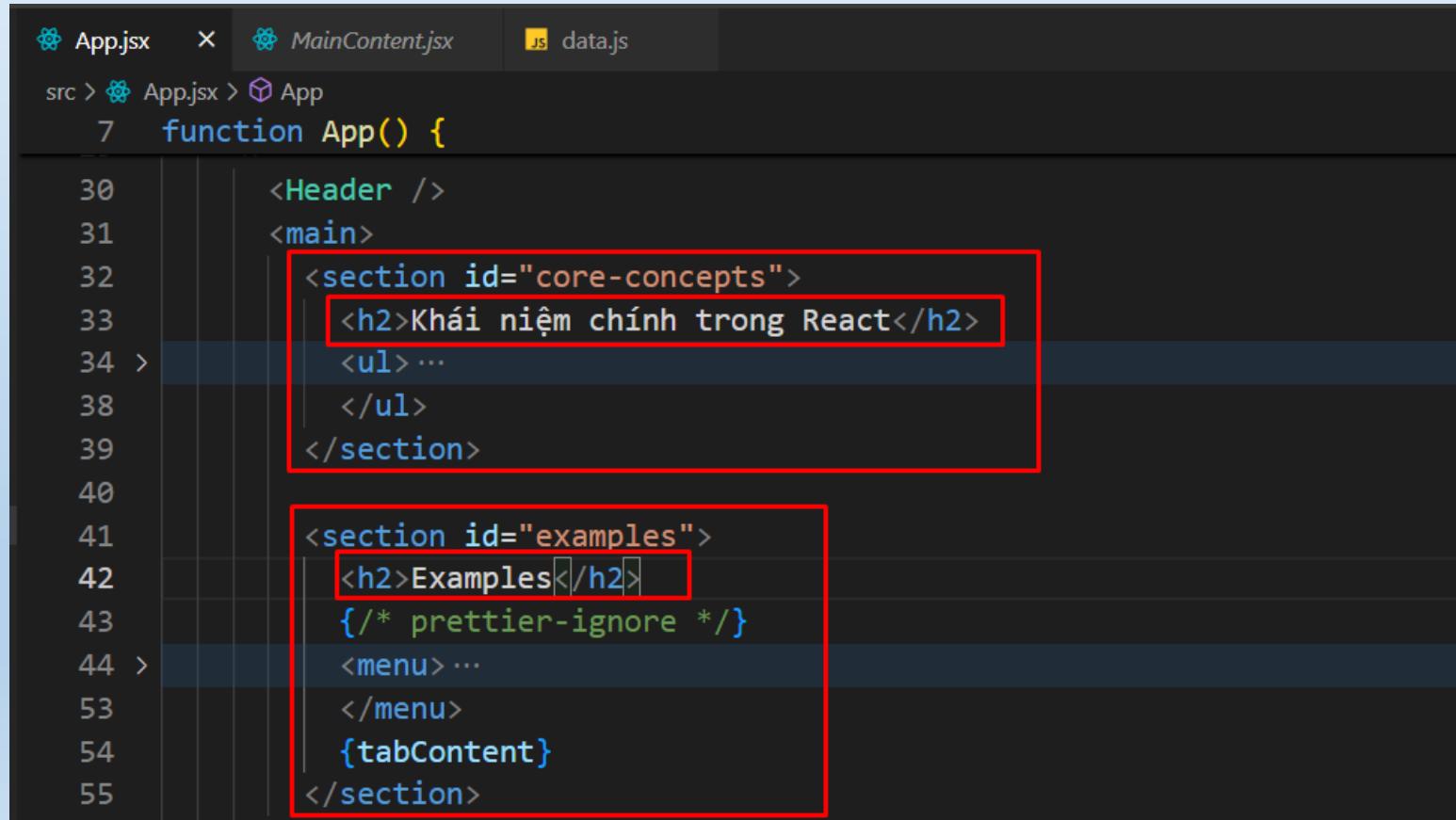
Tối ưu code với Section Component



23

Tối ưu hoá code - Section component

- ✓ Để ý thấy phần **section** đang có cấu trúc lặp lại giữa các phần, và nếu như sau này chúng ta có thêm các phần mục khác cấu trúc tương tự, ta có thể sử dụng **jsx** để tái sử dụng

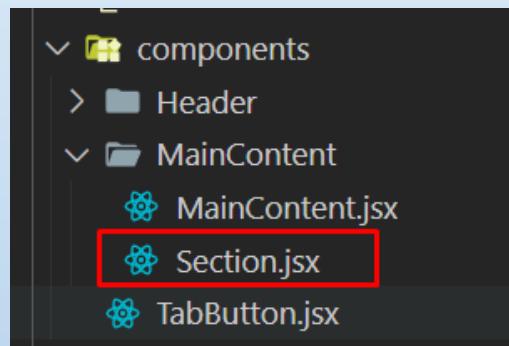


```
App.jsx  X  MainContent.jsx  data.js
src > App.jsx > App
7   function App() {
30     <Header />
31     <main>
32       <section id="core-concepts">
33         <h2>Khái niệm chính trong React</h2>
34       <ul>...
38     </ul>
39   </section>
40
41   <section id="examples">
42     <h2>Examples</h2>
43     {/* prettier-ignore */}
44   <ul>...
53   </ul>
54   {tabContent}
55 </section>
```

23

Tối ưu hoá code - Section component

- ✓ Tạo component Section.jsx



Section.jsx >

```
components / MainContent / Section.jsx / ...  
export default function Section({title, children}) {  
  return (  
    <section>  
      <h2>{title}</h2>  
      {children}  
    </section>  
  );  
}
```

23

Tối ưu hoá code - Section component

- ✓ Examples.jsx sẽ được viết lại như sau

```
<Section title="Examples" id="examples">
  /* prettier-ignore */
  <menu>
    <TabButton
      isSelected={selectedTopic === "components"}
      onSelect={() => handleSelect('components')}>
      Components
    </TabButton>
    <TabButton isSelected={selectedTopic === "jsx"} onSelect={() => handleSel
    <TabButton isSelected={selectedTopic === "props"} onSelect={() => handleS
    <TabButton isSelected={selectedTopic === "state"} onSelect={() => handleS
  </menu>
  {tabContent}
</Section>
```

23

Tối ưu hoá code - Section component

- ✓ *Nhưng có 1 vấn đề , thuộc tính id chúng ta khai báo ở trên gấp lõi, react không biết id đó là gì - bạn có thể kiểm tra phần tử, id không hề tồn tại ở đó, chính vì vậy giao diện xuất hiện lỗi*

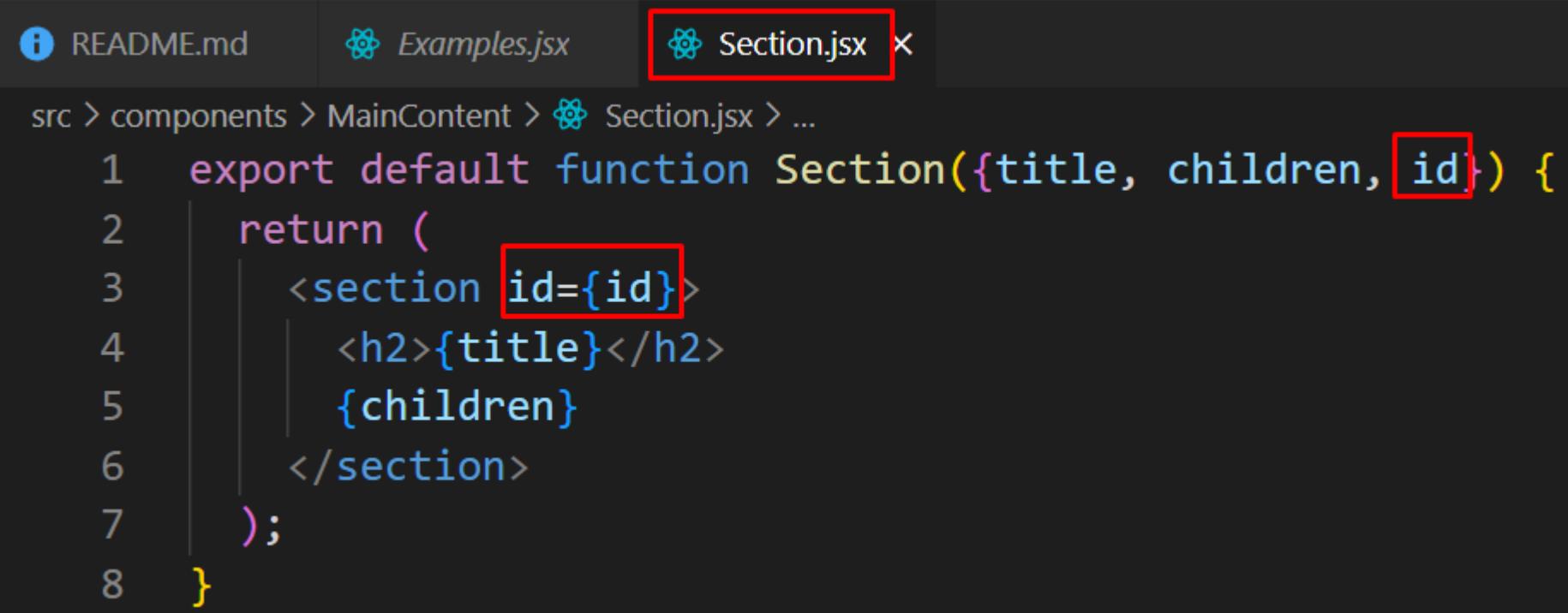


```
<Section title="Examples" id="examples">
  /* prettier-ignore */
  <menu>
    <TabButton
      isSelected={selectedTopic === "components"}
      onSelect={()=>{handleSelect('components')}}>
      Components
    </TabButton>
    <TabButton isSelected={selectedTopic === "jsx"} onSelect={()=>{handleSelect('jsx')}}>
    <TabButton isSelected={selectedTopic === "props"} onSelect={()=>{handleSelect('props')}}>
    <TabButton isSelected={selectedTopic === "state"} onSelect={()=>{handleSelect('state')}}>
  </menu>
  {tabContent}
</Section>
```

23

Tối ưu hoá code - Section component

- ✓ Để xử lý lỗi này, chúng ta cần thêm thuộc tính id, vào trong section để truyền từ giao diện

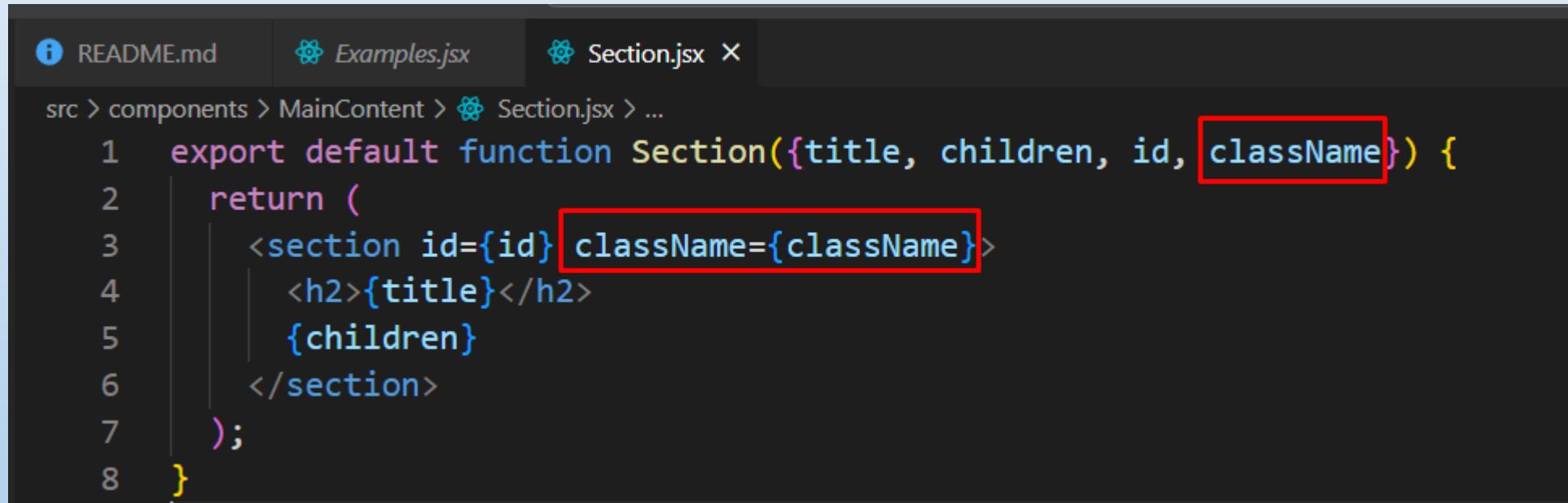


```
1  export default function Section({title, children, id}) {
2    return (
3      <section id={id}>
4        <h2>{title}</h2>
5        {children}
6      </section>
7    );
8  }
```

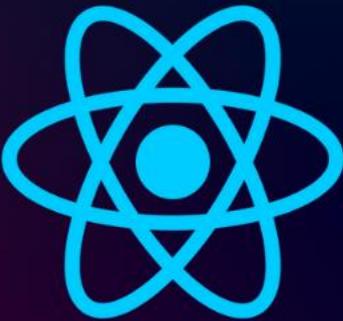
23

Tối ưu hoá code - Section component

- ✓ *Tuy nhiên, nếu cứ mỗi lần truyền thêm thuộc tính: id, className... chúng ta lại truyền thủ công thì rất mất thời gian, chúng ta sẽ xử lý vấn đề này ở clip tiếp theo*



```
1  export default function Section({title, children, id, className}) {
2    return (
3      <section id={id} className={className}>
4        <h2>{title}</h2>
5        {children}
6      </section>
7    );
8  }
```



REACT JS



BÀI 14-19

Forwarding Props

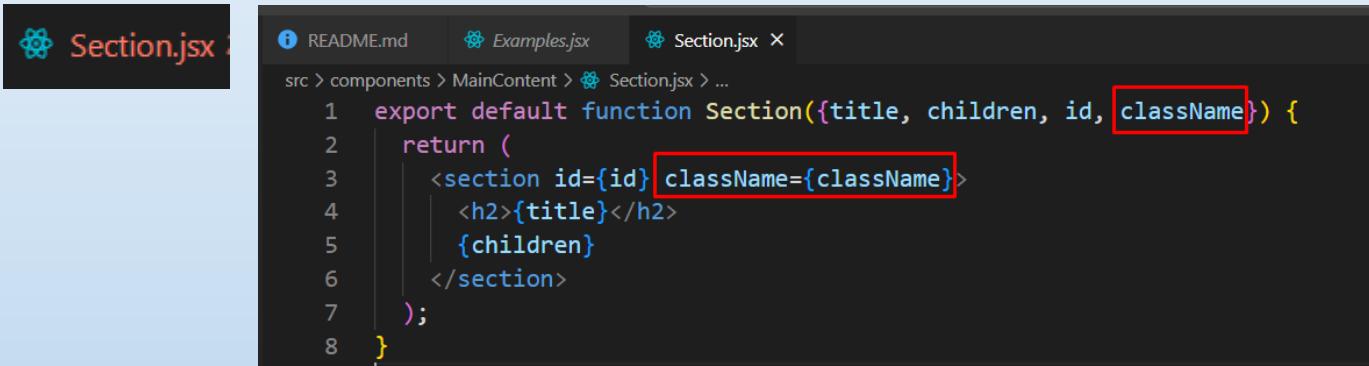
Cách sử dụng
...props



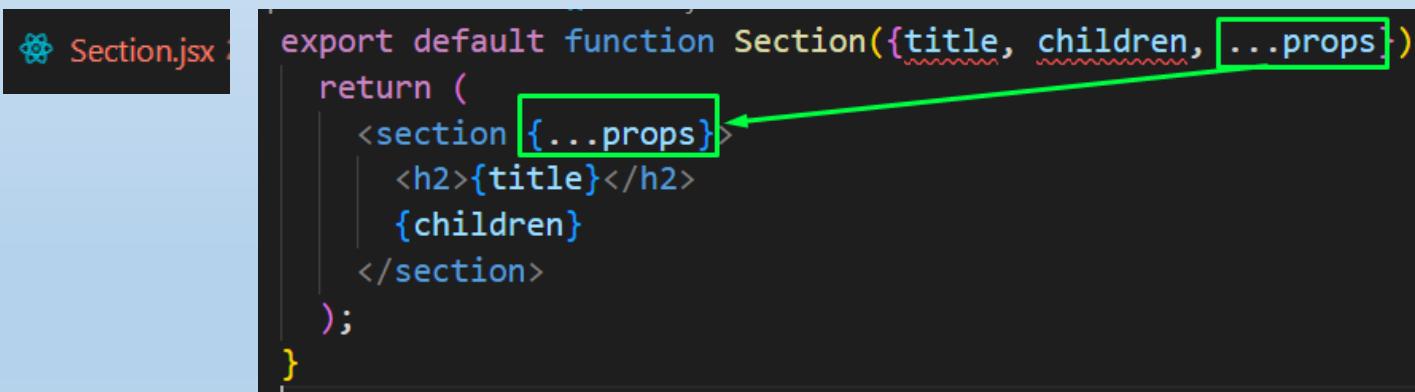
24

Forwarding Props

- ✓ *Cứ mỗi lần truyền thêm thuộc tính: id, className... chúng ta lại truyền thủ công thì rất mất thời gian → Sử dụng*



```
 1 export default function Section({title, children, id, className}) {
 2   return (
 3     <section id={id} className={className}>
 4       <h2>{title}</h2>
 5       {children}
 6     </section>
 7   );
 8 }
```



```
export default function Section({title, children, ...props}) {
  return (
    <section {...props}>
      <h2>{title}</h2>
      {children}
    </section>
  );
}
```

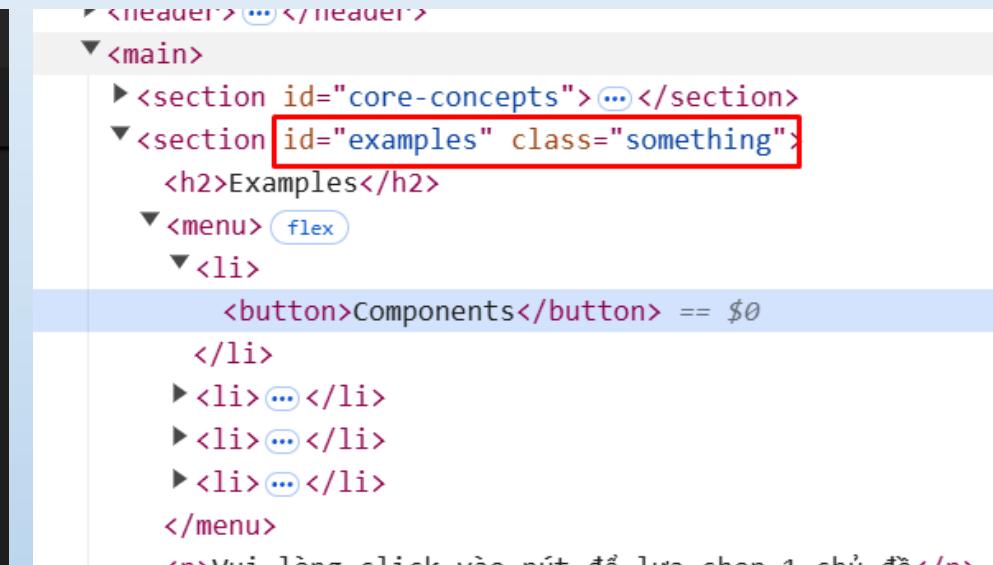
- ✓ *Khi đó chúng ta có truyền mọi thuộc tính 1 cách tự động, Spread Operator: ...props sẽ tự động trải các thuộc tính này và áp dụng*

24 Forwarding Props

Examples.jsx

```
ADME.md  Section.jsx 2  Examples.jsx X
components > MainContent > Examples.jsx > ...
5  export function Examples() {
6    }
7    return (
8      <>
9        <Section title="Examples" id="examples" className="something">
10          /* prettier-ignore */
11          <menu>
12            <TabButton
13              isSelected={selectedTopic === "components"}
14              onClick={()=>{handleSelect('components')}}>
15              Components
16            </TabButton>
17            <TabButton isSelected={selectedTopic === "jsx"} onClick={()=>{ha...}}
18            <TabButton isSelected={selectedTopic === "props"} onClick={()=>{...}}
19            <TabButton isSelected={selectedTopic === "state"} onClick={()=>{...}}
20            </menu>
21            {tabContent}
22          </Section>
23        </>
24      
```

kiểm tra trên Dev Tools

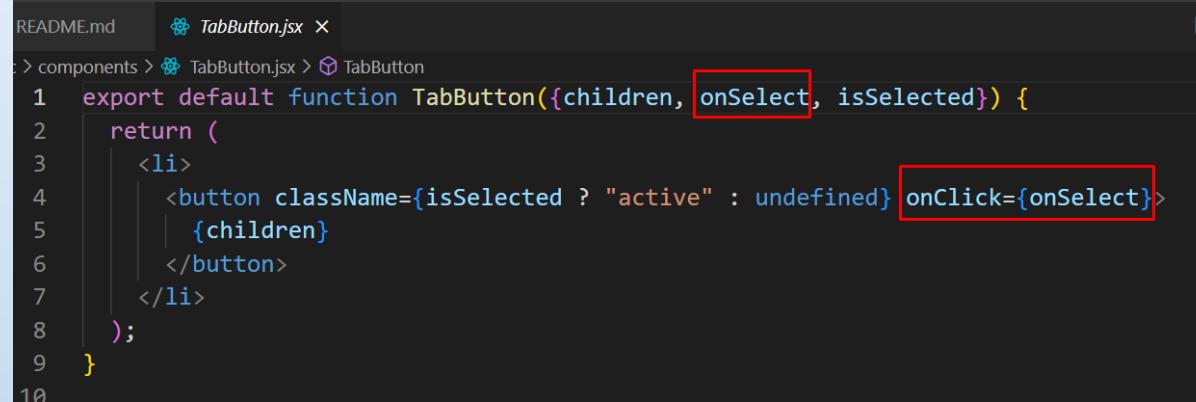


- ✓ Đây là 1 cách hữu ích khi xây dựng các component dạng bao bọc, như Section ở trên

24

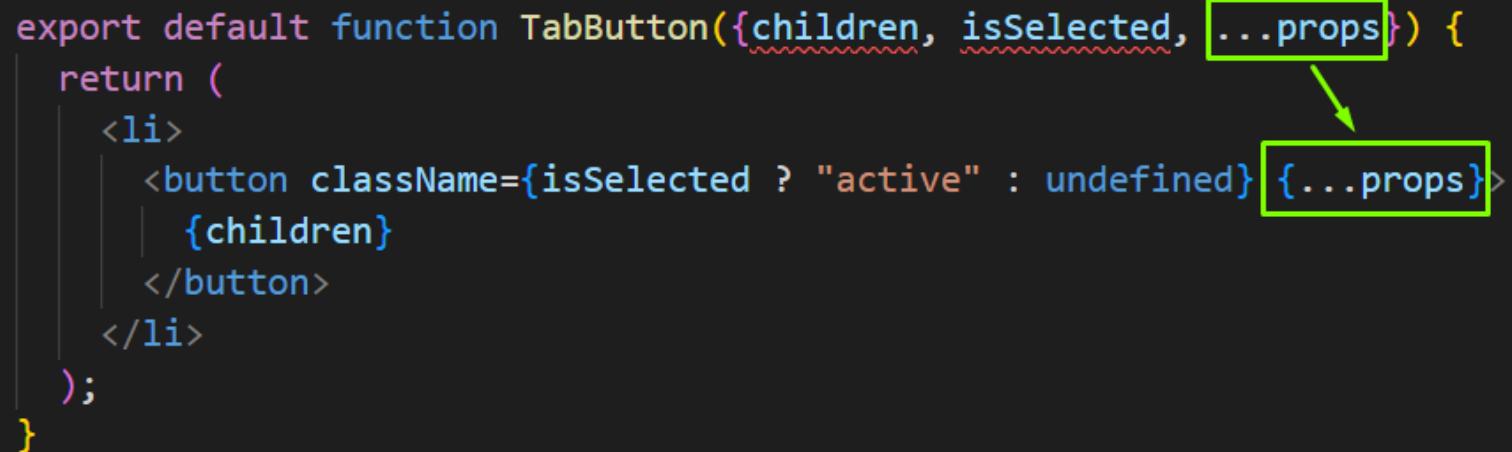
Forwarding Props - TabButton.jsx

- ✓ *Tương tự, chúng ta thấy rằng TabButton.jsx, thuộc tính onSelect cũng được truyền thủ công*



```
README.md  TabButton.jsx X
: > components > TabButton.jsx > TabButton
1  export default function TabButton({children, onSelect, isSelected}) {
2    return (
3      <li>
4        <button className={isSelected ? "active" : undefined} onClick={onSelect}>
5          {children}
6        </button>
7      </li>
8    );
9  }
10
```

- ✓ *Có thể áp dụng ...props như sau*



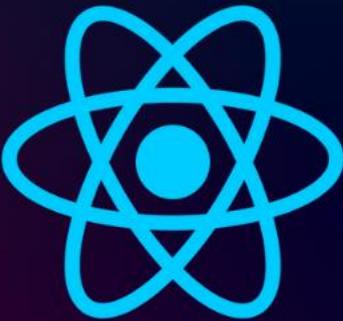
```
export default function TabButton({children, isSelected, ...props}) {
  return (
    <li>
      <button className={isSelected ? "active" : undefined} {...props}>
        {children}
      </button>
    </li>
  );
}
```

24

Forwarding Props - TabButton.jsx

- ✓ Khi đó chúng ta có thể sử dụng mọi props cho Tabbutton

```
<Section title="Examples" id="examples" className="something">
  {/* prettier-ignore */}
  <menu>
    <TabButton
      isSelected={selectedTopic === "components"}
      onClick={()=>{handleSelect('components')}}>
      Components
    </TabButton>
    <TabButton isSelected={selectedTopic === "jsx"} onClick={()=>{handleSelect('jsx')}}>JSX</TabButton>
    <TabButton isSelected={selectedTopic === "props"} onClick={()=>{handleSelect('props')}}>Props</TabButton>
    <TabButton isSelected={selectedTopic === "state"} onClick={()=>{handleSelect('state')}}>State</TabButton>
  </menu>
  {tabContent}
</Section>
```



REACT JS



BÀI 14-20

Tabs Menu

Tăng hiệu quả và tái sử
dụng trong ReactJS



25

Tạo component cho phần tabs menu

✓ Chúng ta còn phần menu cũng có thể xây dựng theo cấu trúc tương tự, để có thể tái sử dụng

 App.jsx

```
<Section title="Examples" id="examples" className="something">
  /* prettier-ignore */
  <menu>
    <TabButton
      isSelected={selectedTopic === "components"}
      onClick={() => handleSelect('components')}
    >Components</TabButton>
    <TabButton isSelected={selectedTopic === "jsx"} onClick={() => handleSelect('jsx')}>JSX</TabButton>
    <TabButton isSelected={selectedTopic === "props"} onClick={() => handleSelect('props')}>Props</TabButton>
    <TabButton isSelected={selectedTopic === "state"} onClick={() => handleSelect('state')}>State</TabButton>
  </menu>
  {tabContent}
</Section>
```

⌄ MainContent
 MainContent.jsx
 Section.jsx
  Tabs.jsx

 Tabs.jsx

```
src > components > MainContent >  Tabs
1  export default function Tabs({children, button}) {
2    return (
3      <>
4        <menu>{button}</menu>
5        {children}
6      </>
7    );
8  }
9
```

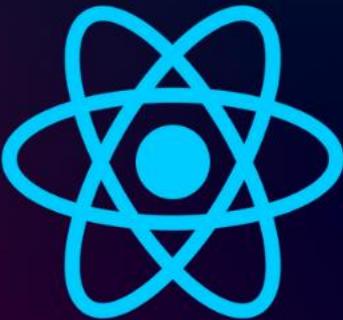
25

Tạo component cho phần tabs menu



```
<Section title="Examples" id="examples" className="something">
  /* prettier-ignore */
  <Tabs>
    <button onClick={()=>handleSelect('components')}>
      Components
    </button>
    <TabButton isSelected={selectedTopic === "jsx"} onClick={()=>handleSelect('jsx')}>JSX</TabButton>
    <TabButton isSelected={selectedTopic === "props"} onClick={()=>handleSelect('props')}>Props</TabButton>
    <TabButton isSelected={selectedTopic === "state"} onClick={()=>handleSelect('state')}>State</TabButton>
  </Tabs>
  {tabContent}
</Section>
```

- ✓ *Bằng cách này, nếu dự án mở rộng, ta có thể thêm nhiều thành phần hơn nữa vào trong Tabs*



REACT JS



BÀI 14-21

Tabs Menu

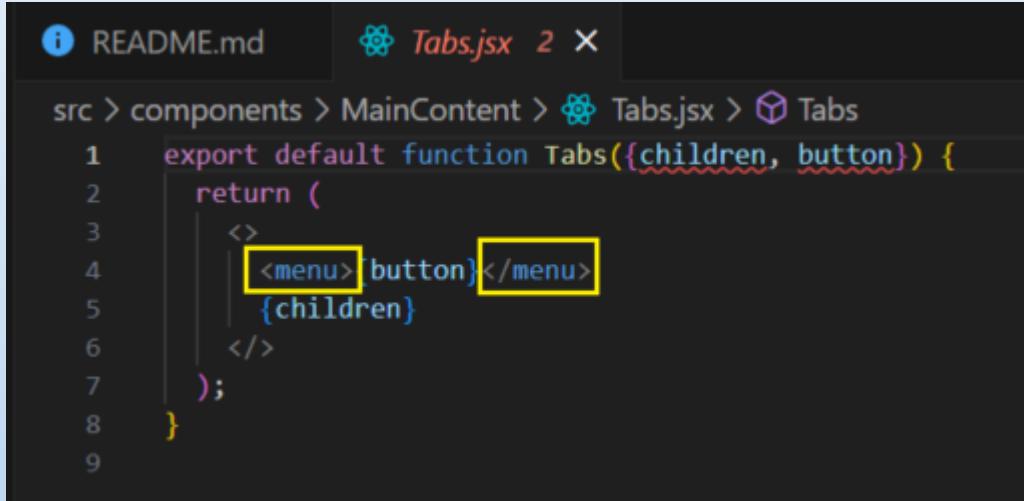
Tối ưu Tabs Menu cho
ứng dụng lớn



26

Cải tiến hơn nữa phần tabs menu

- ✓ *Với các ứng dụng lớn hơn, chúng ta có thể thiết kế để component Tabs trở nên linh hoạt hơn nữa*



```
src > components > MainContent > Tabs.js > Tabs
1  export default function Tabs({children, button}) {
2    return (
3      <>
4        <menu> button</menu>
5        {children}
6      </>
7    );
8  }
9
```

A screenshot of a code editor showing the file structure and content of Tabs.js. The code defines a functional component 'Tabs' that takes 'children' and 'button' props. It returns a JSX fragment (<></>) containing a 'menu' element which contains the 'button' prop and the 'children' prop. Lines 4 and 5 are highlighted with a yellow box.

- ✓ *Chúng ta có thể thiết kế để linh hoạt sử dụng thẻ bọc bên ngoài, thay vì fix cứng trong thẻ menu thẻ này*

26

Cải tiến hơn nữa phần tabs menu

 Tabs.jsx

```
src > components > MainContent > Tabs.jsx > ...
1  export default function Tabs({children, buttons, buttonsContainer}) {
2    return (
3      <>
4        <buttonsContainer>{buttons}</buttonsContainer>
5        {children}
6      </>
7    );
8  }
9
10 |
```

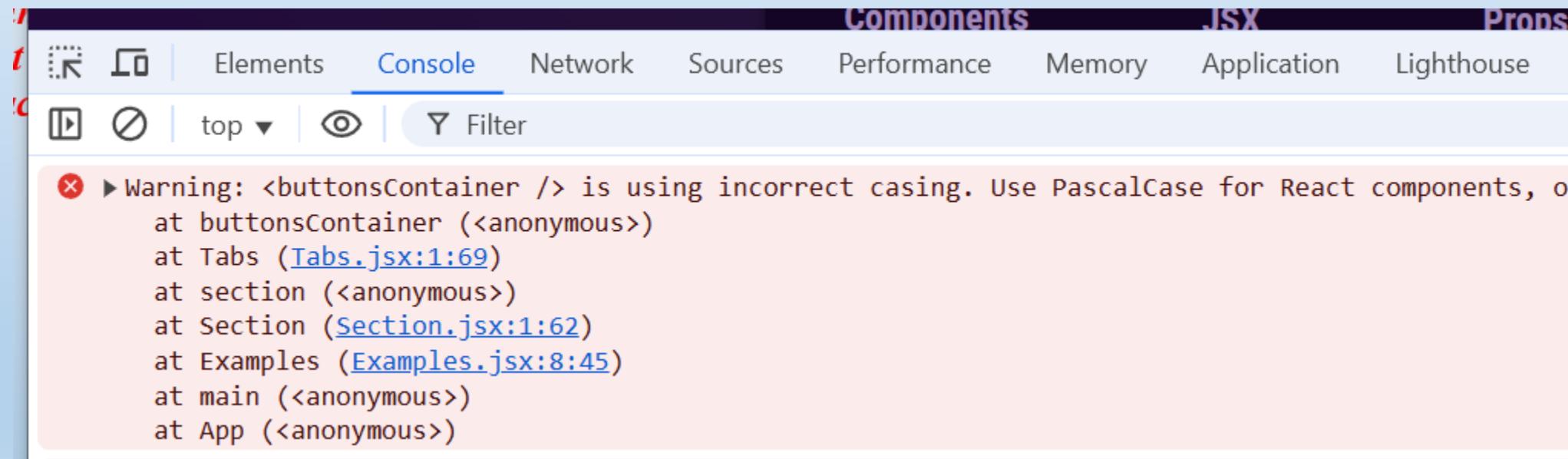
 App.jsx

```
<Tabs
  buttonsContainer="menu"
  buttons={<>
    <TabButton
      isSelected={selectedTopic === "components"}
      onClick={()=>{handleSelect('components')}}>
      Components
    </TabButton>
    <TabButton isSelected={selectedTopic === "jsx"} onClick={
    <TabButton isSelected={selectedTopic === "props"} onClick=
    <TabButton isSelected={selectedTopic === "state"} onClick
    </>}
  >
```

26

Cải tiến hơn nữa phần tabs menu

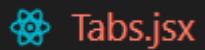
- ✓ Trong JSX, nếu một tag bắt đầu bằng chữ thường (ví dụ: <div>,), React sẽ hiểu đó là một thẻ HTML gốc.
- ✓ Nếu một tag bắt đầu bằng chữ hoa (ví dụ: <MyComponent>), React sẽ hiểu đó là một component React.



26

Cải tiến hơn nữa phần tabs menu

✓ Đã khắc phục, cách 1:



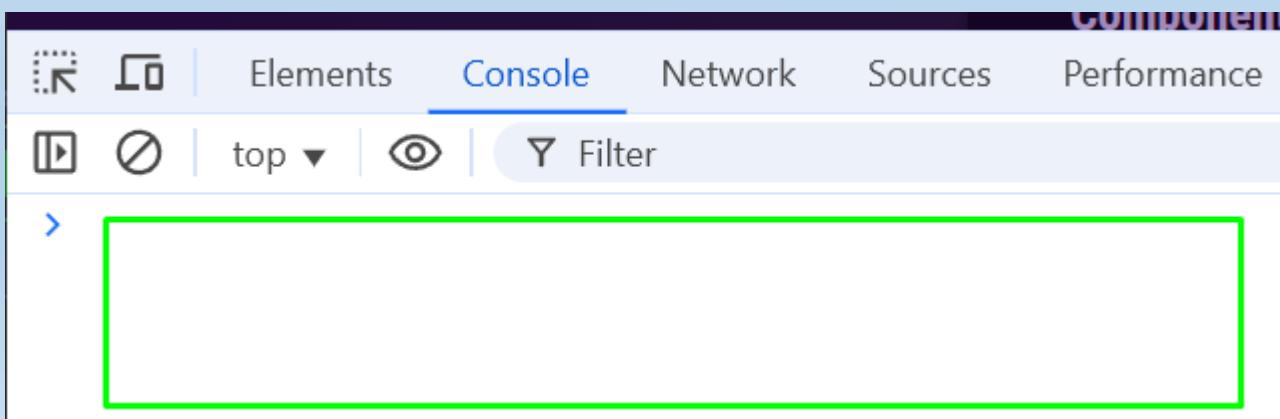
Components > MainContent > Tabs.jsx > Tabs

```
export default function Tabs({children, buttons, buttonsContainer}) {
  const ButtonsContainer = buttonsContainer;
  return (
    <>
      <ButtonsContainer>{buttons}</ButtonsContainer>
      {children}
    </>
  );
}
```



<Tabs

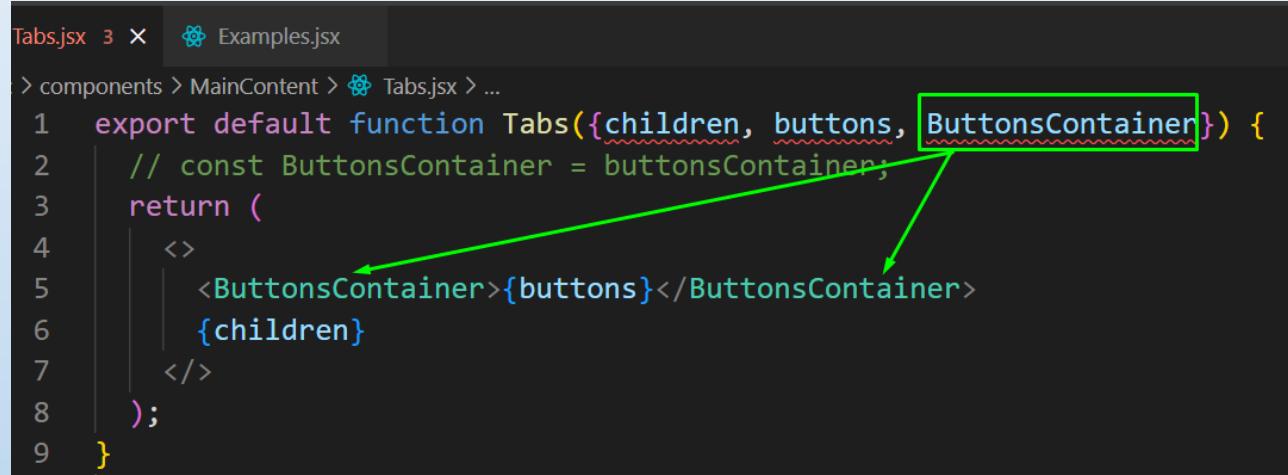
```
  buttonsContainer="menu"
  buttons={<>
    <TabButton
      isSelected={selectedTopic === "components"}
      onClick={() => handleSelect('components')}
      Components
    </TabButton>
    <TabButton isSelected={selectedTopic === "jsx"} onClick={>
      <TabButton isSelected={selectedTopic === "props"} onClick={>
        <TabButton isSelected={selectedTopic === "state"} onClick={>
          </>
        </>
      </>
    </>
  </>
}
```



26

Cải tiến hơn nữa phần tabs menu

✓ Đã khắc phục, cách 2 :

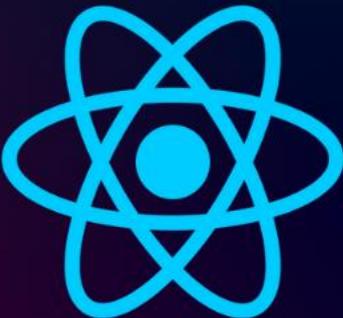


```
Tabs.jsx 3 × Examples.jsx
> components > MainContent > Tabs.jsx > ...
1  export default function Tabs({children, buttons, ButtonsContainer}) {
2    // const ButtonsContainer = buttonsContainer;
3    return (
4      <>
5        <ButtonsContainer>{buttons}</ButtonsContainer>
6        {children}
7      </>
8    );
9 }
```

 App.jsx

```
<Tabs
  ButtonsContainer="menu"
  buttons={<>
    <TabButton
      isSelected={selectedTopic === "components"}
      onClick={() => handleSelect('components')}
    >
      Components
    </TabButton>
    <TabButton isSelected={selectedTopic === "jsx"} onClick={() => handleSelect('jsx')}>
      JSX
    </TabButton>
    <TabButton isSelected={selectedTopic === "props"} onClick={() => handleSelect('props')}>
      Props
    </TabButton>
    <TabButton isSelected={selectedTopic === "state"} onClick={() => handleSelect('state')}>
      State
    </>
  </>
}
```

✓ Quan trọng là các bạn cần hiểu, chúng ta cần dùng **chữ cái đầu viết hoa** để react hiểu đó là thẻ tùy chỉnh, không phải là 1 thẻ gốc đến từ html. Vì nếu nó hiểu là thẻ html thì nó sẽ hiển thị ngay thẻ đó lên trên trình duyệt



REACT JS

BÀI 14-22



Default Props

**Thiết Lập Giá Trị Mặc Định
Cho Props**



27

Default Prop Values

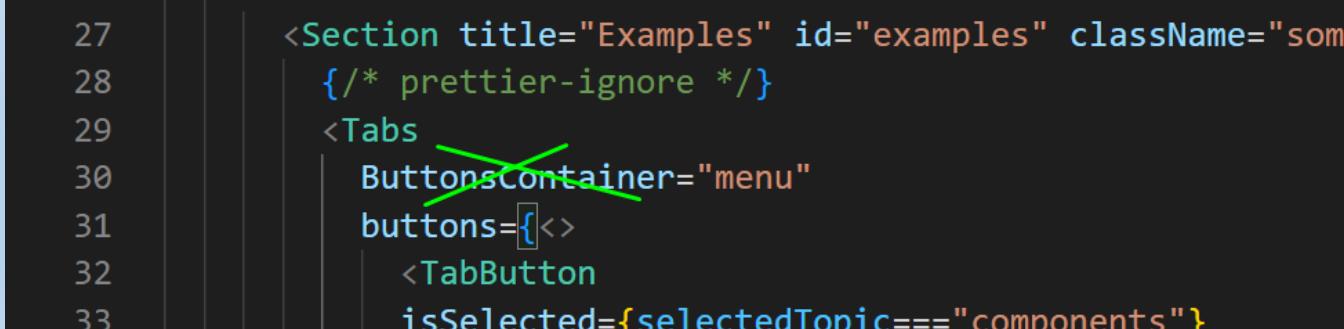
- ✓ Chúng ta muốn trong trường hợp không truyền props, nó sẽ nhận giá trị mặc định



```

  1  export default function Tabs({children, buttons, ButtonsContainer = ["menu"]}){
  2    return (
  3      <>
  4        <ButtonsContainer>{buttons}</ButtonsContainer>
  5        {children}
  6      </>
  7    );
  8  }

```



```

  27  <Section title="Examples" id="examples" className="some-class">
  28    /* prettier-ignore */
  29    <Tabs
  30      ButtonsContainer="menu"
  31      buttons={<>
  32        <TabButton
  33          isSelected={selectedTopic === "components"}>

```

kiểm tra trong developer tool chúng ta vẫn thấy thẻ menu hoạt động như trước

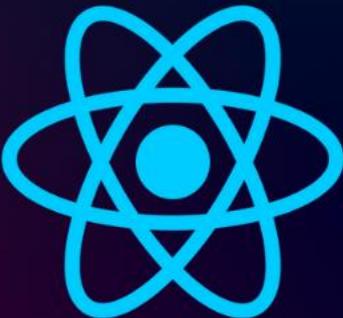


27

Default Prop Values

- ✓ *Bằng cách này, nếu ta không truyền props, nó sẽ nhận value mặc định*
- ✓ *Hoặc có thể truyền bất kỳ thẻ nào div, ul, hoặc cả thành phần tùy chỉnh Section*

```
ButtonsContainer = {Section}
```



REACT JS



BÀI 14-23

Tối ưu hóa App.jsx

Xây dựng App.jsx tối giản
trong React



28

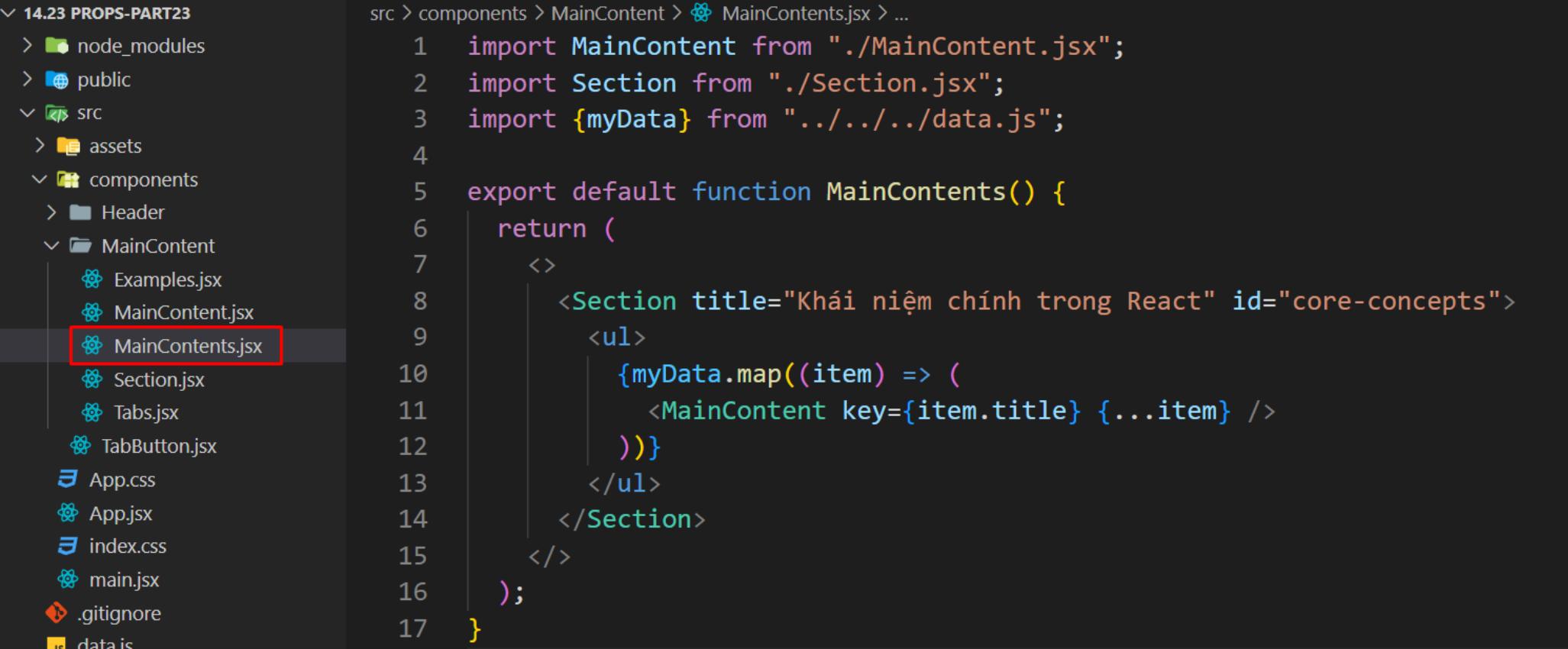
Tách nhỏ component trong App.jsx

- ✓ *File App.jsx hiện tại đang có quá nhiều code khó quản lý, chúng ta có thể tách nhỏ thành các component riêng*

```
1 import Header from "./components/Header/Header.jsx";
2 import MainContents from "./components/MainContent/MainContents.jsx";
3 import Examples from "./components/MainContent/Examples.jsx";
4
5 function App() {
6   return (
7     <>
8       <Header />
9       <main>
10         <MainContents />
11         <Examples />
12       </main>
13     </>
14   );
15 }
```

28

Tách nhỏ component trong App.jsx



The screenshot shows a code editor with a sidebar displaying the project structure:

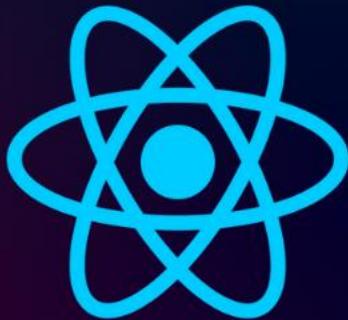
- 14.23 PROPS-PART23
 - node_modules
 - public
 - src
 - assets
 - components
 - Header
 - MainContent
 - Examples.jsx
 - MainContent.jsx
 - MainContents.jsx (highlighted with a red box)
 - Section.jsx
 - Tabs.jsx
 - TabButton.jsx
 - App.css
 - App.jsx
 - index.css
 - main.jsx
 - .gitignore
 - data.js

28

Tách nhỏ component trong App.jsx

 Examples.jsx

```
1 import {useState} from "react";
2 import {EXAMPLES} from "../../data.js";
3 import TabButton from "../TabButton.jsx";
4 import Section from "./Section.jsx";
5 import Tabs from "./Tabs.jsx";
6
7 export default function Examples() {
8     const [selectedTopic, setSelectedTopic] = useState();
9     let tabContent = <p>Vui lòng click vào nút để lựa chọn 1 chủ đề</p>;
10    if (selectedTopic) { ...
11    }
12    function handleSelect(selectedButton) { ...
13    }
14    return (
15        <Section title="Examples" id="examples" className="demo_class" moi="moi">
16            {/* prettier-ignore */}
17            <Tabs
18                // ButtonsContainer= {Section}
19                button={
20                    <>
21                        <TabButton
22                            isSelected={selectedTopic === "components"}
23                            onClick={()=>{handleSelect('components')}}>
24                            Components
25                        </TabButton>
26                        <TabButton isSelected={selectedTopic === "jsx"} onClick={()=>{handleSelect('jsx')}}>JSX</TabButton>
27                        <TabButton isSelected={selectedTopic === "props"} onClick={()=>{handleSelect('props')}}>Props</TabButton>
28                        <TabButton isSelected={selectedTopic === "state"} onClick={()=>{handleSelect('state')}}>State</TabButton></>
29                    </>
30                    {tabContent}
31                </Tabs>
32            </Section>
33        );
34    }
35 }
```



REACT JS



BÀI 15

useState



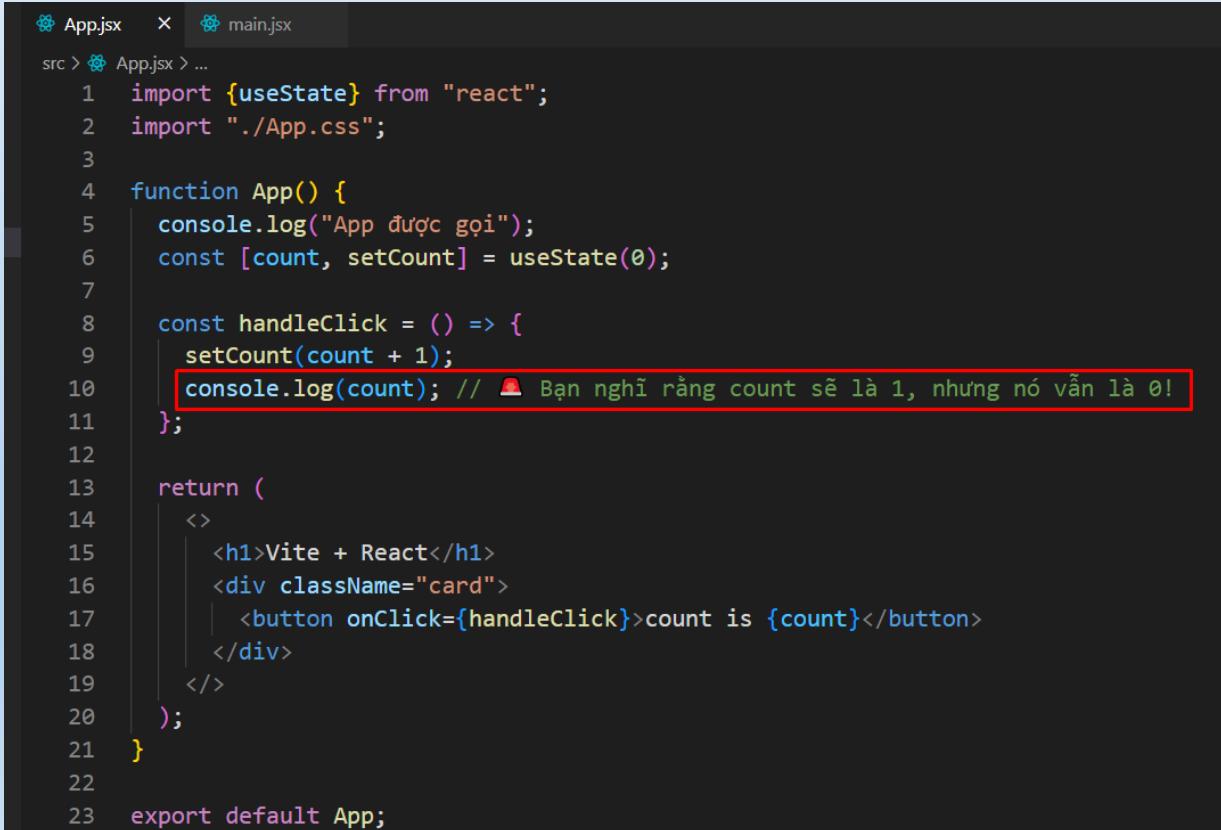
**Sai Lầm Khi sử dụng
useState Trong React**



1

Hiểu Sai Về Cập Nhật State Trong React

- ✓ Một trong những lỗi phổ biến nhất khi làm việc với state trong React là **nghĩ rằng state thay đổi ngay lập tức sau khi gọi setState**. Tuy nhiên, React **không cập nhật state ngay lập tức** mà chỉ đánh dấu state sẽ thay đổi trong lần render tiếp theo.



```
App.jsx  X  main.jsx
src > App.jsx > ...
1 import {useState} from "react";
2 import "./App.css";
3
4 function App() {
5   console.log("App được gọi");
6   const [count, setCount] = useState(0);
7
8   const handleClick = () => {
9     setCount(count + 1);
10    console.log(count); // 🚨 Bạn nghĩ rằng count sẽ là 1, nhưng nó vẫn là 0!
11  };
12
13  return (
14    <>
15      <h1>Vite + React</h1>
16      <div className="card">
17        <button onClick={handleClick}>count is {count}</button>
18      </div>
19    </>
20  );
21}
22
23 export default App;
```

- **setCount(count + 1)** chỉ đánh dấu rằng state sẽ thay đổi, nhưng React không cập nhật ngay lập tức.
- Giá trị **count** trong **console.log(count)** vẫn là giá trị cũ trong render hiện tại.

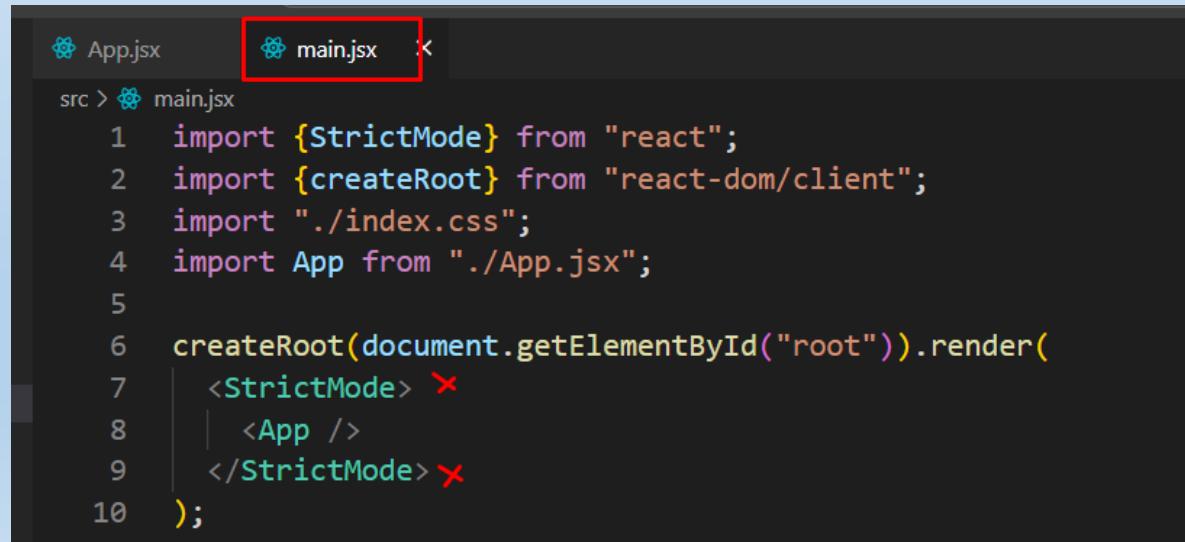
2

React Hoạt Động Với State Như Thế Nào?

☐ Khi gọi `setState`, React thực hiện các bước sau:

- ✓ Đánh dấu state sẽ thay đổi.
- ✓ Chờ render tiếp theo.
- ✓ Cập nhật state mới khi render lại component.

→ Lưu ý: Trong chế độ `StrictMode`, React sẽ render hai lần trong môi trường development để phát hiện lỗi tiềm ẩn trong các effect hoặc lifecycle methods. Nhưng trong production, React chỉ render một lần.



```
App.jsx          main.jsx <
src > main.jsx
1  import {StrictMode} from "react";
2  import {createRoot} from "react-dom/client";
3  import "./index.css";
4  import App from "./App.jsx";
5
6  createRoot(document.getElementById("root")).render(
7    <StrictMode> ✘
8    |  <App />
9    </StrictMode> ✘
10   );
11 
```

3

Sai lầm dễ mắc phải

- ❑ *Sai Lầm Khi Thay Đổi State Nhiều Lần Trong Cùng Một Event:*
 - ✓ *Giả sử bạn muốn tăng count 3 lần trong một sự kiện:*

```
const [count, setCount] = useState(0);

const incrementThreeTimes = () => {
    setCount(count + 1);
    setCount(count + 1);
    setCount(count + 1);

    console.log(count); // ⚠️ Không tăng lên 3 như bạn nghĩ!
};
```

4

Callback Trong setState

- Để giải quyết bài toán trên: React cung cấp cách cập nhật state dựa trên giá trị trước đó bằng cách truyền vào một hàm callback

```
const [count, setCount] = useState(0);

const incrementThreeTimes = () => {
    setCount(count + 1);
    setCount(count + 1);
    setCount(count + 1);

    console.log(count); // 🚨 Không tăng lên 3 như bạn nghĩ!
};
```

Cách dùng sai

```
//dùng hàm call back
const [count, setCount] = useState(0);

const incrementThreeTimes = () => {
    setCount((prevCount) => prevCount + 1);
    setCount((prevCount) => prevCount + 1);
    setCount((prevCount) => prevCount + 1);
};
```

Cách dùng đúng (sử dụng hàm callback)

5

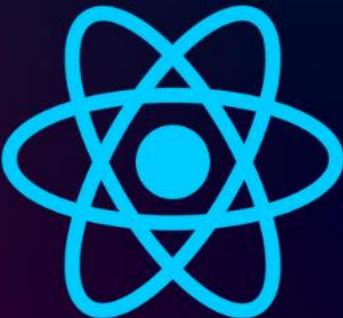
Tóm tắt bài học

- setState không cập nhật ngay lập tức, React chỉ đánh dấu cho lần render tiếp theo.*
- Nếu thay đổi state dựa trên giá trị hiện tại, luôn dùng callback function:*

`setState((prevState) => prevState + 1);`

Callback

- StrictMode trong môi trường development khiến React render 2 lần – không phải lỗi.*



REACT JS



BÀI 16

Two - Way Binding

Hiểu Rõ One-Way Binding
& Two-Way Binding



1

Giới thiệu

- ✓ Theo mặc định **Bất cứ khi nào state thay đổi, giao diện (UI) sau khi được React re-render thì giá trị mới sẽ được hiển thị**. Đây chính là cơ chế của **One-Way Binding** trong React.
(state → UI)

□ **Two-Way Binding là gì ?**

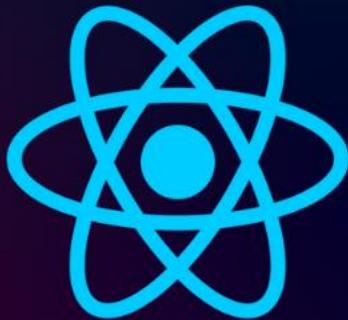
(state ⇌ UI) là cơ chế giúp đồng bộ dữ liệu giữa giao diện người dùng (UI) và state trong React.

- ✓ Khi dữ liệu trong UI thay đổi → state sẽ cập nhật ngay lập tức. Khi state thay đổi → UI sẽ tự động cập nhật theo.
- ✓ Điều này giúp việc quản lý dữ liệu trở nên dễ dàng hơn, đặc biệt là trong các form nhập liệu.
- ✓ Two-Way Binding trong React không chỉ dùng với input mà có thể áp dụng với nhiều thành phần UI khác như checkbox, radio, select, range, toggle, v.v.

2

Ví dụ 1: Two-Way Binding đơn giản với input

```
src > App.jsx > ...
1 import {useState} from "react";
2 import "./App.css";
3
4 function App() {
5   // Khai báo state để lưu giá trị của input
6   const [playerName, setPlayerName] = useState("Player 1");
7   // Hàm xử lý khi người dùng nhập liệu
8   const handleChange = (event) => {
9     setPlayerName(event.target.value); // Cập nhật state với giá trị mới
10  };
11  return (
12    <>
13      <h2>Nhập tên người chơi:</h2>
14      <input type="text" value={playerName} onChange={handleChange} />
15      <p>Bạn đã nhập: {playerName}</p>
16    </>
17  );
18}
19
20 export default App;
```



REACT JS



BÀI 16.2

Two - Way Binding

Two-Way Binding với
nhiều trường input



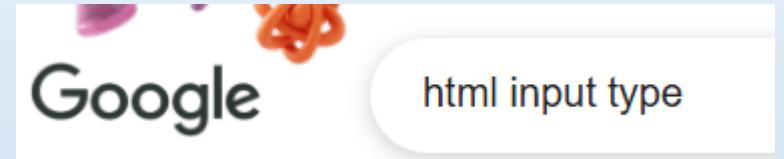
3

Ví dụ 2: Two-Way Binding với nhiều trường input

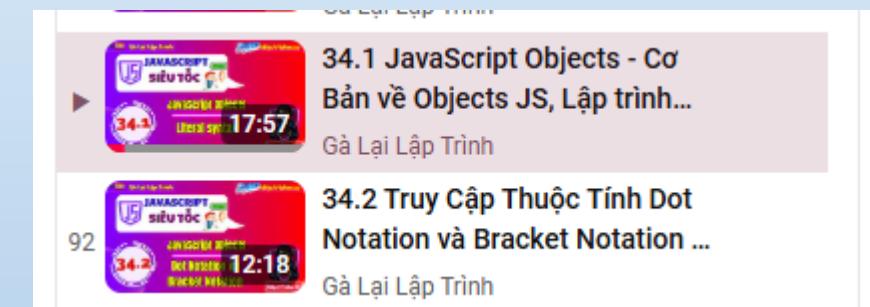
```
// Khởi tạo state dưới dạng object để quản lý nhiều trường dữ liệu
const [user, setUser] = useState({fullName: "", email: ""});
// Hàm xử lý thay đổi dữ liệu nhập vào input
const handleChange = (event) => {
    const {name, value} = event.target;
    setUser((prevUser) => ({
        ...prevUser, // Giữ nguyên các dữ liệu khác
        [name]: value, // Cập nhật dữ liệu theo name của input
    }));
};

return [
    <div>
        <h2>Cập nhật thông tin cá nhân</h2>
        <label>
            Họ và Tên:
            <input type="text" name="fullName" value={user.fullName} onChange={handleChange} />
        </label>
        <br />
        <label>
            Email:
            <input type="email" name="email" value={user.email} onChange={handleChange} />
        </label>
        <br />
        <h3>Thông tin đã nhập:</h3>
        <p>Họ và Tên: {user.fullName}</p>
        <p>Email: {user.email}</p>
    </div>
];
```

```
import {useState} from "react";
import "./App.css";
```

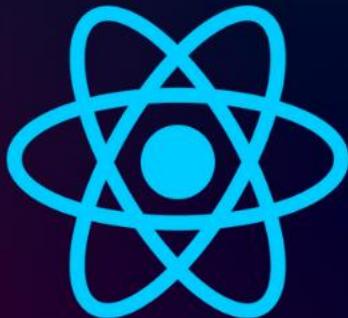


<http://js.tuhoc.cc>



<http://react.tuhoc.cc>





REACT JS



BÀI 17.1

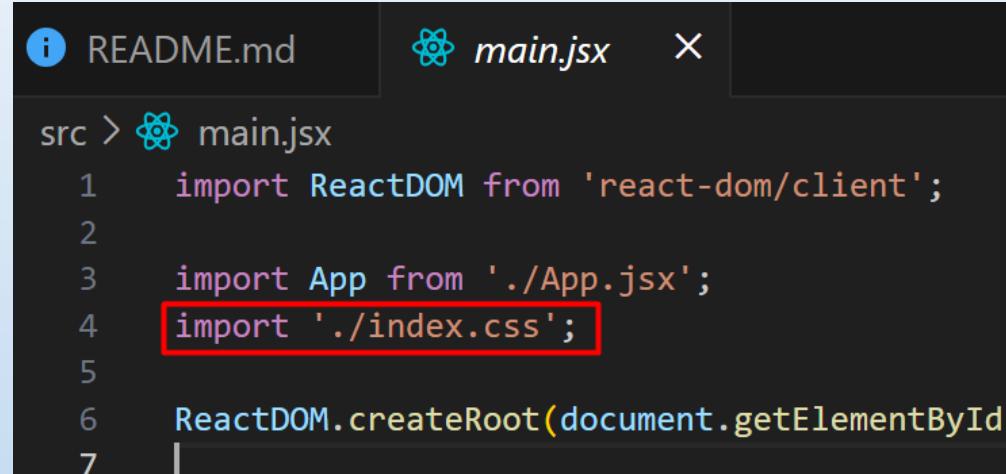
React CSS

Import CSS



1

Import CSS



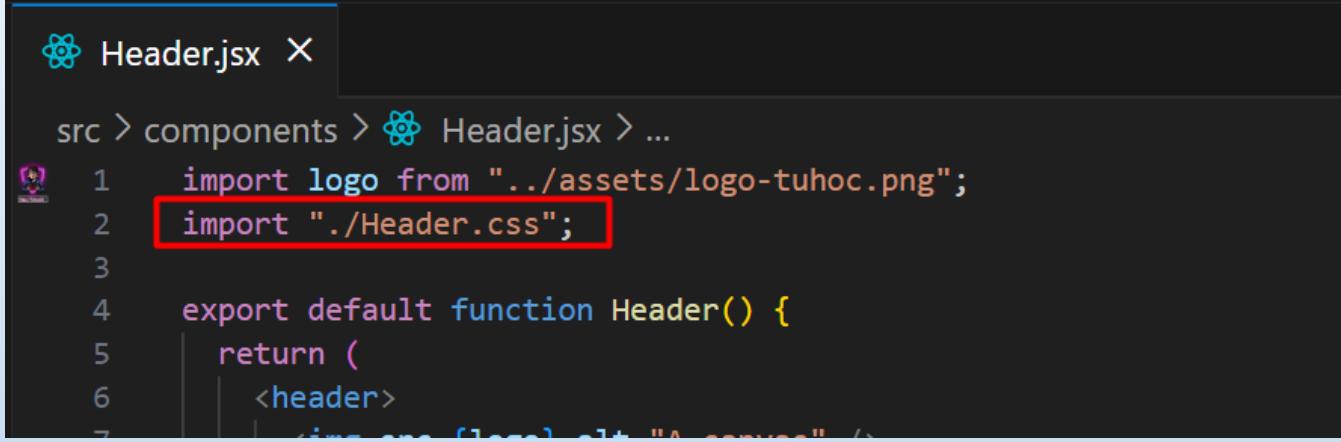
```
src > main.jsx
1 import ReactDOM from 'react-dom/client';
2
3 import App from './App.jsx';
4 import './index.css';
5
6 ReactDOM.createRoot(document.getElementById(
7 |
```

Sau khi import, Vite sẽ xử lý file CSS và nhúng nó vào trang web



2

Tách css theo component

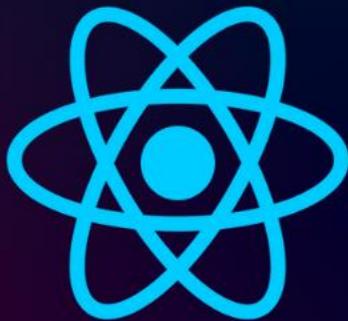


```
Header.jsx X
src > components > Header.jsx > ...
1 import logo from "../assets/logo-tuhoc.png";
2 import "./Header.css";
3
4 export default function Header() {
5   return (
6     <header>
7       <img alt="A canvas" />
```

Điểm quan trọng: mặc dù chúng ta đưa CSS vào trong component, nhưng khi render, toàn bộ CSS được đưa ra ngoài toàn cục như phía dưới



```
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
▶<style type="text/css" data-vite-dev-id="D:/0.lap trinh/3.3-react-js/0..slide bài giảng kiên/1.code-full-baigian.css start project/src/components/Header.css">...</style>
▶<style type="text/css" data-vite-dev-id="D:/0.lap trinh/3.3-react-js/0..slide bài giảng kiên/1.code-full-baigian.css start project/src/index.css">...</style>
</head>
```



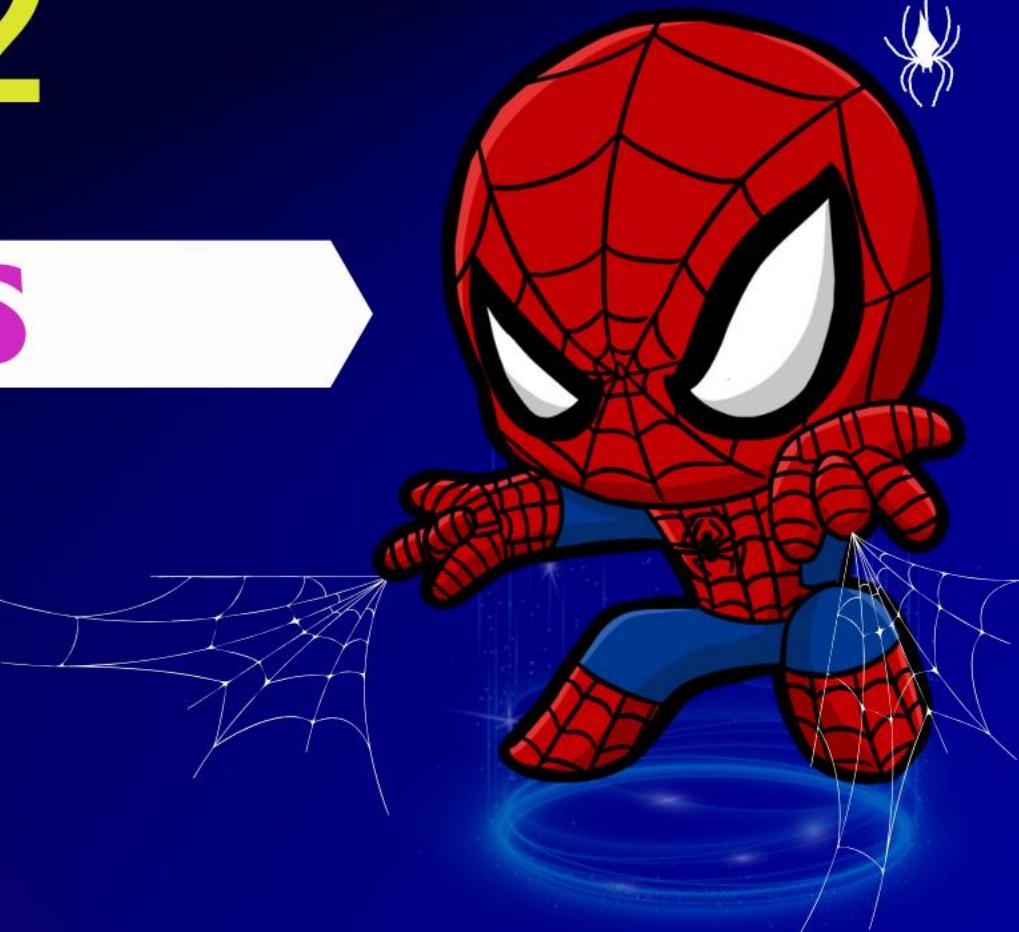
REACT JS



BÀI 17.2

React CSS

Import CSS
Buổi 2



3

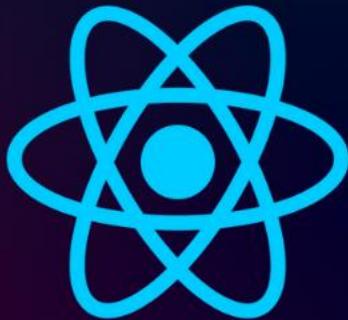
Nâng cao

Vậy nếu bây giờ chúng ta muốn thêm điều kiện , để 1 phần tử có đồng thời nhiều class thì làm thế nào ví dụ label có đồng thời class label và tự động thêm class invalid nếu nó không hợp lệ

```
AuthInputs.jsx x
src > components > AuthInputs.jsx > AuthInputs
  3  export default function AuthInputs() {
  4    <div id="auth-inputs">
  5      <div className="controls">
  6        <p>
  7          <label className={label ${emailNotValid ? "invalid" : ""}}>Email</label>
  8          <input
  9            type="email"
 10            className={emailNotValid ? "invalid" : undefined}
 11            onChange={(event) => handleInputChange("email", event.target.value)}
 12          />
 13        </p>
 14      </div>
 15    </div>
 16  </div>
```



```
<main>
  <div id="auth-inputs">
    <div className="controls"> flex
      <p>
        <label className="label invalid">Email</label> == $0
        <input type="email" className="invalid">
      </p>
      <p>...</p>
    </div>
```



REACT JS



BÀI 17.3

React CSS

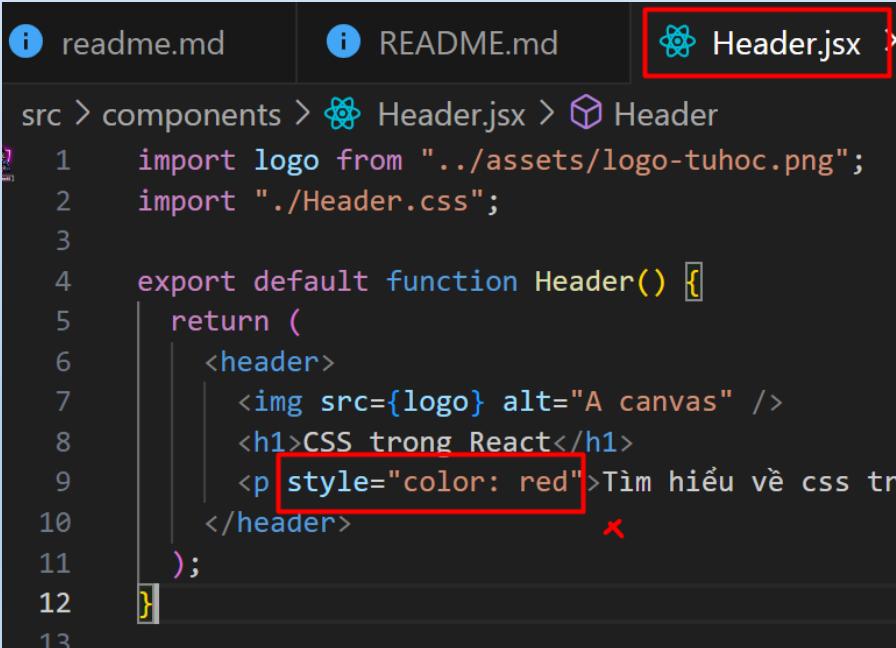
Inline styling in
Reactjs



1

Lời dẫn

- ✓ Trong buổi học hôm trước, khi chúng ta dùng **css thuận**, và import vào component, thì mã CSS của chúng ta vẫn áp dụng cho toàn bộ (global). Nếu chúng ta lại muốn nó chỉ áp dụng nội bộ trong component thì làm thế nào? Bây giờ là lúc chúng ta tìm hiểu về **css inline**
- ✓ Chúng ta sẽ không thêm như cách sử dụng HTML thông thường, vì nó sẽ gặp lỗi



```
readme.md README.md Header.jsx
src > components > Header.jsx > Header
1 import logo from "../assets/logo-tuhoc.png";
2 import "./Header.css";
3
4 export default function Header() {
5   return (
6     <header>
7       <img src={logo} alt="A canvas" />
8       <h1>CSS trong React</h1>
9       <p style="color: red">Tìm hiểu về css tro
10      </header>
11    );
12  }
13
```

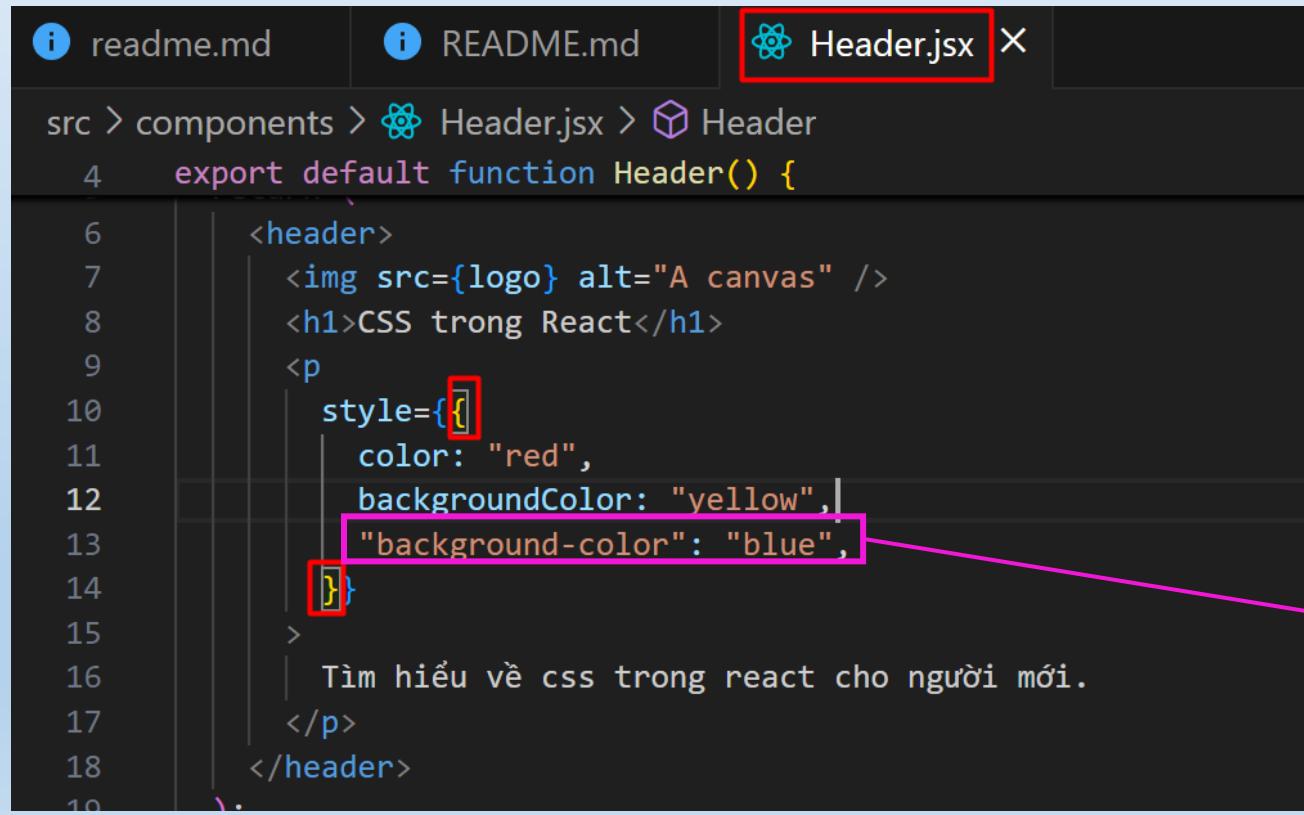


```
Uncaught Error: The `style` prop expects a mapping from style property names to values...
at setValueForStyles (react-dom-client.development.js:2614:15)
at setProp (react-dom-client.development.js:16926:11)
at setInitialProperties (react-dom-client.development.js:17711:13)
at completeWork (react-dom-client.development.js:13775:18)
at runWithFiberInDEV (react-dom-client.development.js:543:16)
at completeUnitOfWork (react-dom-client.development.js:15179:19)
at performUnitOfWork (react-dom-client.development.js:15061:11)
at workLoopSync (react-dom-client.development.js:14870:41)
at renderRootSync (react-dom-client.development.js:14850:11)
at performWorkOnRoot (react-dom-client.development.js:14384:44)
```

2

Inline styling in Reactjs

- ✓ *Thay vào đó chúng ta truyền giá trị động trong jsx, sử dụng đối tượng (object) để truyền dưới dạng cặp key và value*



```
readme.md README.md Header.jsx
src > components > Header.jsx > Header
4  export default function Header() {
5
6      <header>
7          <img src={logo} alt="A canvas" />
8          <h1>CSS trong React</h1>
9
10         <p
11             style={}
12             color: "red",
13             backgroundColor: "yellow",
14             "background-color": "blue",
15         >
16             Tìm hiểu về css trong react cho người mới.
17         </p>
18     </header>
19 
```

Lưu ý : 1 số thuộc tính phải tuân theo quy tắc js, ví dụ
`background-color` → `backgroundColor`
`text-align` → `textAlign` ...
chúng ta đã học ở series JS

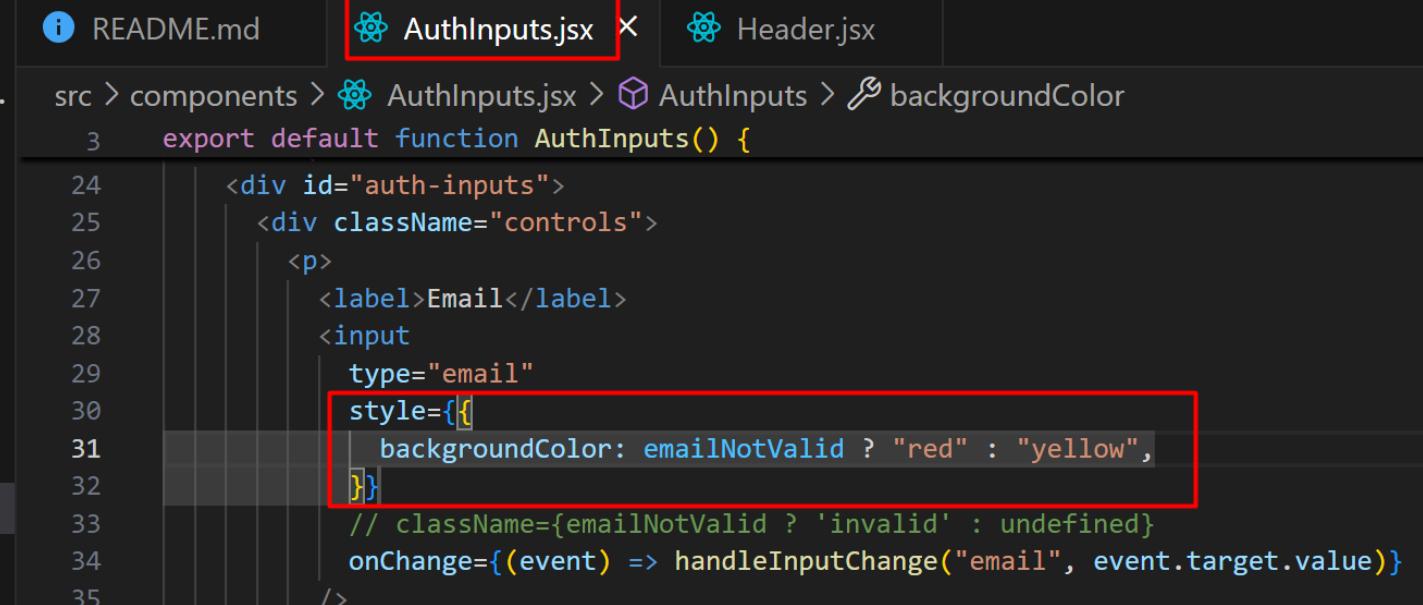
✖ ► Unsupported style property `background-color`.
at p (<anonymous>)
at header (<anonymous>)
at Header (<anonymous>)
at App (<anonymous>)

3

Ưu/ Nhược điểm

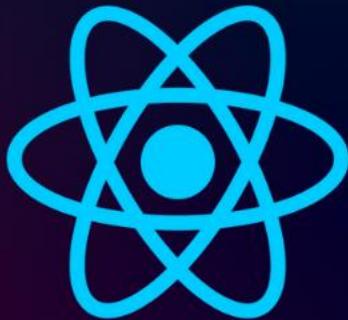
✓ *Ưu điểm:*

- *Inline css thêm vào mã jsx, sẽ chỉ áp dụng lên thành phần đó*
- *Chúng ta có thể sử dụng inline css trong 1 số trường hợp cụ thể :*



```
1 README.md          2 AuthInputs.jsx X  3 Header.jsx
src > components > AuthInputs.jsx > AuthInputs > backgroundColor
3   export default function AuthInputs() {
24     <div id="auth-inputs">
25       <div className="controls">
26         <p>
27           <label>Email</label>
28           <input
29             type="email"
30             style={{
31               backgroundColor: emailNotValid ? "red" : "yellow",
32             }}
33             // className={emailNotValid ? 'invalid' : undefined}
34             onChange={(event) => handleInputChange("email", event.target.value)}
35         />
```

- ✓ *Nhược điểm: Phải thêm mã vào từng thẻ html riêng lẻ, và mã bị trộn lẫn vào jsx không có sự tách biệt css, và jsx*
 - *Không tái sử dụng được*



REACT JS



BÀI 17.4

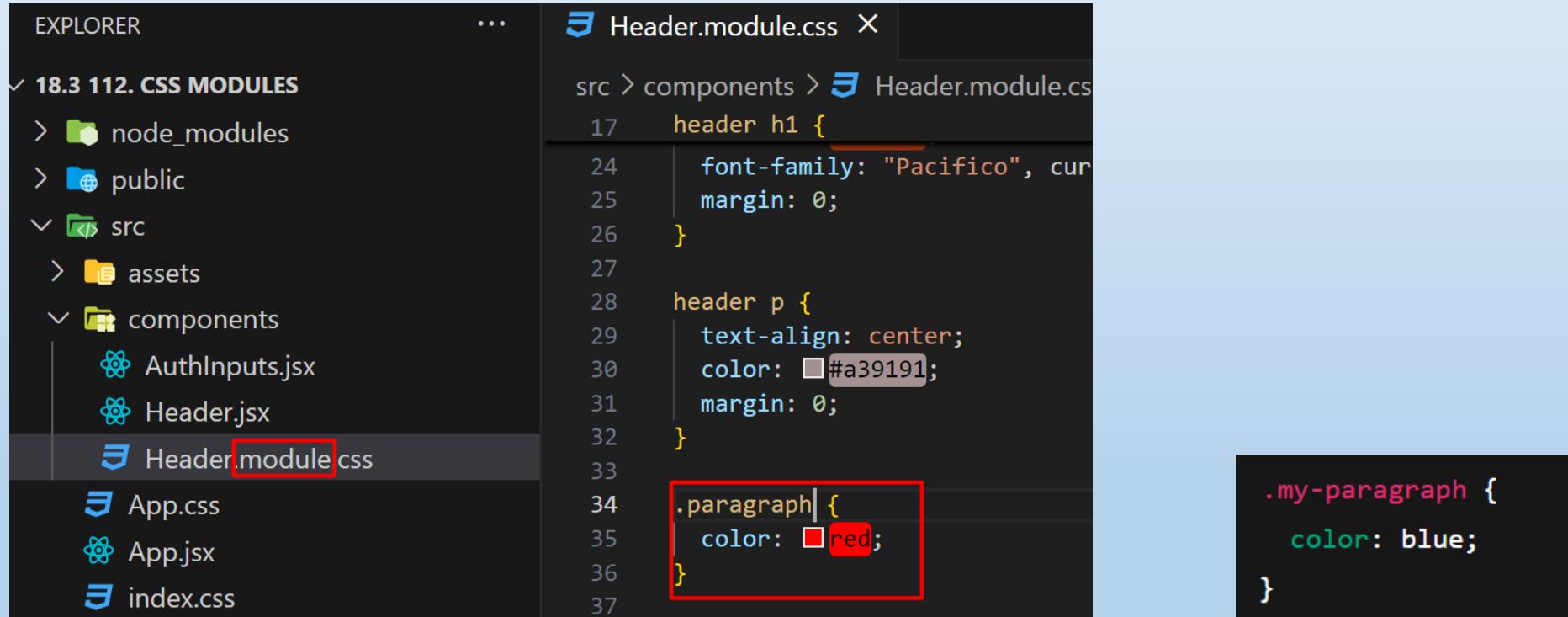
React CSS

CSS Module



1 CSS module

- ✓ Sau khi nhúng, React sẽ tự động tạo class tự động áp dụng riêng cho component
- ✓ Bước 1: Tạo module CSS



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a project structure under '18.3 112. CSS MODULES'. It includes 'node_modules', 'public', and a 'src' folder containing 'assets', 'components' (with files 'AuthInputs.jsx' and 'Header.jsx'), and 'Header.module.css'. The 'Header.module.css' file is currently selected and highlighted with a red border. In the main editor area, the file 'Header.module.css' is open, showing the following CSS code:

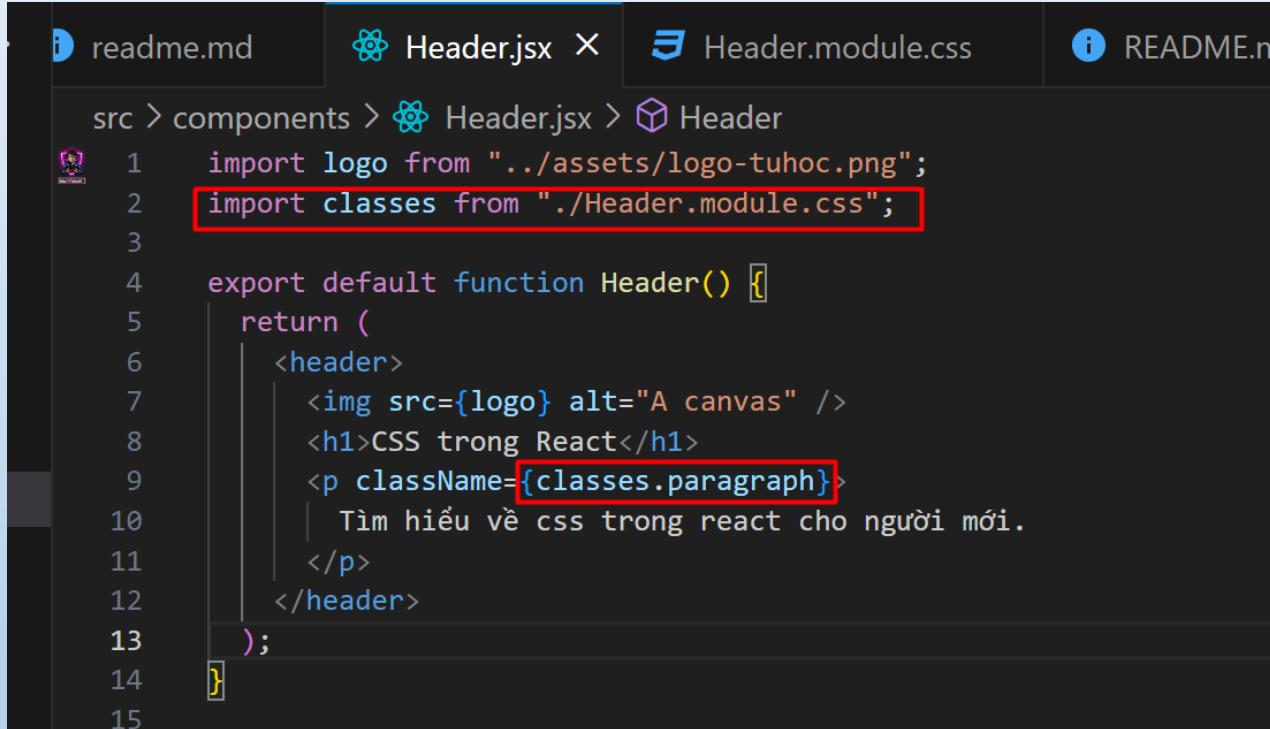
```
src > components > Header.module.css
17 header h1 {
24   font-family: "Pacifico", cur
25   margin: 0;
26 }
27
28 header p {
29   text-align: center;
30   color: #a39191;
31   margin: 0;
32 }
33
34 .paragraph {
35   color: red;
36 }
37
```

A red rectangular box highlights the class definition for '.paragraph' at line 34. To the right of the editor, there is a separate code block showing a standard CSS class definition:

```
.my-paragraph {
  color: blue;
}
```

1

CSS module

✓ *Bước 2: import module css để sử dụng*

```
1 import logo from "../assets/logo-tuhoc.png";
2 import classes from "./Header.module.css";
3
4 export default function Header() {
5   return (
6     <header>
7       <img src={logo} alt="A canvas" />
8       <h1>CSS trong React</h1>
9       <p className={classes.paragraph}>
10        Tìm hiểu về css trong react cho người mới.
11      </p>
12    </header>
13  );
14}
15
```

```
.paragraph {
  color: red;
}
```

```
.my-paragraph {
  color: blue;
}
```

Khi import vào React, cần dùng `classes["my-paragraph"]` thay vì `classes.my-paragraph`

1 CSS module

- ✓ *Kết quả : react sẽ thêm class tự động để đảm bảo nó duy nhất, tránh đụng độ giữa các class cùng tên trong nhiều component khác nhau.*

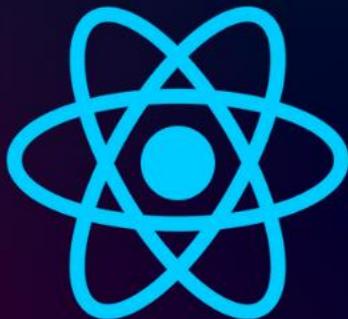
```
<html lang="en"> scroll
  <object id="jnogbabnnbdhjompaagfbjjiimplamll" width="0" height="0" style="display: none;">
    <head> ...
    <body class="p-8">
      <div id="root">
        <header> flex == $0
          
          <h1>CSS trong React</h1>
          <p class="_paragraph_rmkeq_65">Tìm hiểu về css trong react cho người mới.</p>
        </header>
      <main> ...
      </div>
      <script type="module" src="/src/main.js"></script>
    </body>
  </html>
```

- ✓ *Ưu điểm:*

- style áp dụng trên phạm vi component
- Mã CSS có thể tái sử dụng được qua các component mà không lo xung đột

- ✓ *Nhược điểm:*

- Tên class được tạo ra ngẫu nhiên, nên debug sẽ hơi khó khăn hơn bình thường



REACT JS

BÀI 17.5

React CSS

Styled Components



1

Styled components

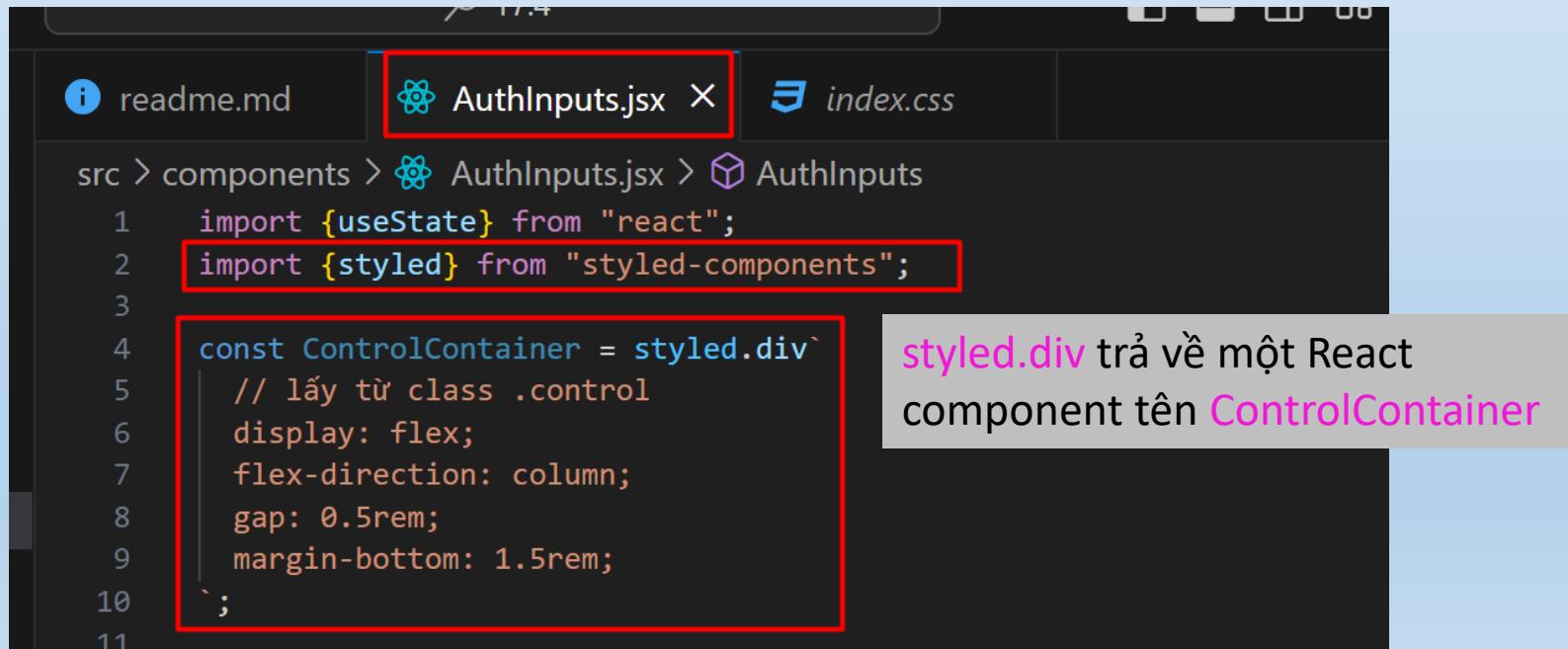
<https://styled-components.com/>

✓ Là một thư viện phổ biến thuộc nhóm **CSS-in-JS**

□ Step 1: Cài đặt thư viện:

npm install styled-components

□ Step 2: Import để sử dụng, khai báo Component chứa mã css :



```
src > components > AuthInputs.jsx > AuthInputs
1 import {useState} from "react";
2 import {styled} from "styled-components";
3
4 const ControlContainer = styled.div` // lấy từ class .control
5   display: flex;
6   flex-direction: column;
7   gap: 0.5rem;
8   margin-bottom: 1.5rem;
9
10`;
11
```

styled.div trả về một React component tên ControlContainer

1

Styled components

<https://styled-components.com/>

- Step 3: Sử dụng component kèm css đã khai báo trước đó

```
<div className="controls">
  <p> ...
  </p>
  <p> ...
  </p>
</div>
```

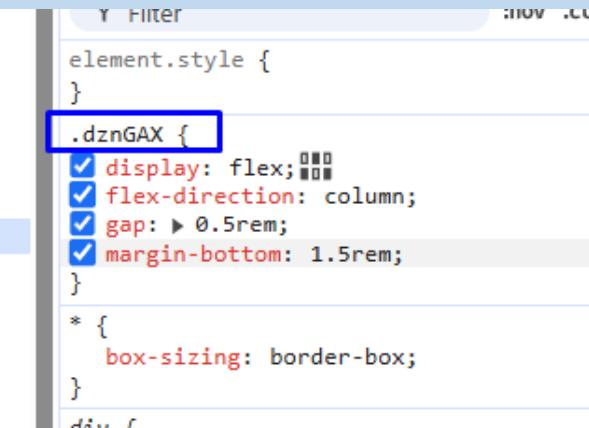


```
<ControlContainer>
  <p> ...
  </p>
  <p> ...
  </p>
</ControlContainer>
```

- Kết quả :



```
> <head> ...
  <body class="p-8">
    <div id="root">
      <header> ...
      <main>
        <div id="auth-inputs">
          <div class="sc-VHjGu dznGAX" ...> ...
          <div class="actions"> ...
        </div>
      </main>
    </div>
    <script type="module" src="/src/main.jsx"></script>
```



Filter: element.style { } .dznGAX { display: flex; flex-direction: column; gap: 0.5rem; margin-bottom: 1.5rem; }

* { box-sizing: border-box; }

1

Styled components

<https://styled-components.com/>

- ✓ Về cơ bản, cách này cũng giúp tạo class ngẫu nhiên không bị trùng lặp giữa các component Và mã css cũng được đưa vào đầu chương trình

```
<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
▶<style type="text/css" data-vite-dev-id="D:/0.lap trinh/3.3-react-js/0..slide bài giảng kiên/1.
  17.5 CSS Styled components/src/components/Header.module.css">@@</style>
▶<style type="text/css" data-vite-dev-id="D:/0.lap trinh/3.3-react-js/0..slide bài giảng kiên/1.
  17.5 CSS Styled components/src/index.css">@@</style>
<style data-styled="active" data-styled-version="6.1.16">.dznGAX{display:flex;
  direction:column;gap:0.5rem;margin-bottom:1.5rem;}</style>
</head>
▼<body class="n-8">
```

Styled-components hoạt động theo component scope, không ảnh hưởng global

1

Styled components

<https://styled-components.com/>

2. Tương tự chúng ta cũng tạo kiểu cho label

AuthInputs.jsx

```
const Label = styled.label`  
  display: block;  
  margin-bottom: 0.5rem;  
  font-size: 0.75rem;  
  font-weight: 700;  
  letter-spacing: 0.1em;  
  text-transform: uppercase;  
  color: #6b7280;  
`;
```

```
<label className={`${label ${emailNotValid ? "invalid" : ""}}>Email</label>
```



```
<Label className={`${label ${emailNotValid ? "invalid" : ""}}>Email</Label>
```

```
<label class="sc-gn0vAp bFowto label ">Password</label> == $0
```

1 Styled components

3. Tương tự với thẻ input

AuthInputs.jsx

```
const Input = styled.input`  
width: 100%;  
padding: 0.75rem 1rem;  
line-height: 1.5;  
background-color: #d1d5db;  
color: #04265c;  
border: 1px solid transparent;  
border-radius: 0.25rem;  
box-shadow: 0 1px 3px 0 rgba(0,  
`;
```

<https://styled-components.com/>

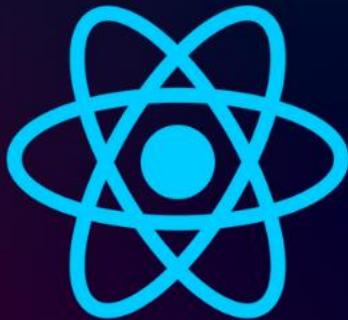
```
<Label className="label">  
  <input  
    type="email" ...  
    onChange={(event) => hand  
  />
```



```
<Label className="label">  
  <Input  
    type="email" ...  
    onChange={(event) => hand  
  />
```

```
  <p>  
    <label class="sc-gn0vAp bFowto label ">Email</label>  
    <input class="sc-dQkurY gHcMuD" type="email" value="S0  
  </p>  
  <p>  
    <label class="sc-gn0vAp bFowto label ">Password</label>  
    <input class="sc-dQkurY gHcMuD" type="password">  
  </p>
```

```
.gHcMuD {  
  width: 100%;  
  padding: 0.75rem 1rem;  
  line-height: 1.5;  
  background-color: #d1d5db;  
  color: #04265c;  
  border: 1px solid transparent;  
  border-radius: 0.25rem;  
  box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.1),  
            0 1px 2px 0 rgba(0, 0, 0, 0.06);  
}
```



REACT JS



BÀI 17.6

React CSS

Styled Components
Dynamic & Conditional
Styling



2

Styled components Điều Kiện

- ✓ Trong bài trước chúng ta đã có thể đưa CSS vào mã jsx sử dụng như với 1 component
- ✓ Chúng ta sẽ không bị lặp code, vì tại nơi sử dụng chỉ cần gọi component

```
return (
  <div id="auth-inputs">
    <ControlContainer>
      <p>
        <Label className={`${label ${emailNotValid ? "invalid" : ""}}>Email</Label>
        <Input
          type="email"
          className={emailNotValid ? "invalid" : undefined}
          onChange={(event) => handleInputChange("email", event.target.value)}
        />
      </p>
      <p>
        <Label className={`${label ${passwordNotValid ? "invalid" : ""}}>Password</Label>
        <Input
          type="password"
          className={passwordNotValid ? "invalid" : undefined}
          onChange={(event) => handleInputChange("password", event.target.value)}
        />
      </p>
    </ControlContainer>
  </div>
)
```

2

Styled components Điều Kiện

- ✓ *Nhưng còn việc thiết lập kiểu css động (Dynamic & Conditional Styling) thì sao ?*
- ✓ *Hiện chúng ta áp dụng css trên Label bằng thuộc tính className, và phụ thuộc vào điều kiện*

```
return (
  <div id="auth-inputs">
    <ControlContainer>
      <p>
        <Label className={`${label ${emailNotValid ? "invalid" : ""}}`}>Email</Label>
        <Input
          type="email"
          className={emailNotValid ? "invalid" : undefined}
          onChange={(event) => handleInputChange("email", event.target.value)}
        />
      </p>
      <p>
        <Label className={`${label ${passwordNotValid ? "invalid" : ""}}`}>Password</Label>
        <Input
          type="password"
          className={passwordNotValid ? "invalid" : undefined}
          onChange={(event) => handleInputChange("password", event.target.value)}
        />
      </p>
    </ControlContainer>
  </div>
```

```
<label class="sc-gn0vAp bFowto label invalid">Email</label> == $0
<input class="sc-dQkurY gHcMuD invalid" type="email">
```

2

Styled components Điều Kiện

- ✓ Thông thường chúng ta sẽ không chèn chúng vào trong jsx (mặc dù nó hoạt động ổn), chúng ta sử dụng props

 AuthInputs.jsx :

```
<div id="auth-inputs">
  <ControlContainer>
    <p>
      <Label invalid= {emailNotValid}>Email</Label>
      <Input
        type="email"
        className={emailNotValid ? "invalid" : undefined}
      >
    </p>
  </ControlContainer>
</div>
```

const Label = styled.label`
display: block;
margin-bottom: 0.5rem;
font-size: 0.75rem;
font-weight: 700;
letter-spacing: 0.1em;
text-transform: uppercase;
// color: #6b7280;
//color: \${props => (props.invalid ? "#f87171" : "#6b7280")};
color: \${({invalid}) => (invalid ? "#f87171" : "#6b7280")};
`;

```
const emailNotValid = submitted && !enteredEmail.includes("@");
```

emailNotValid có thể nhận giá trị là **true** hoặc **false**, và sau đó nó sẽ quyết định giá trị của invalid

Bằng cách này chúng ta có thể gán css theo điều kiện

2

Styled components Điều Kiện

- ✓ *Tương tự với Input*

 AuthInputs.jsx :

Hãy thử tự mình hoàn thành công việc này

```
<ControlContainer>
  <p>
    <Label invalid={emailNotValid}>Email</Label>
    <Input
      type="email"
      className={emailNotValid ? "invalid" : undefined}
      onChange={(event) => handleInputChange("email", event.target.value)}
    />
  </p>
  <p>
    <Label className={`label ${passwordNotValid ? "invalid" : ""}`}>Password</Label>
    <Input
      type="password"
      className={passwordNotValid ? "invalid" : undefined}
      onChange={(event) => handleInputChange("password", event.target.value)}
    />
  </p>
```

 index.css

```
input.invalid {
  color: #ef4444;
  border-color: #f73f3f;
  background-color: #fed2d2;
}
```

2

Styled components Điều Kiện

- ✓ *Tương tự với Input*

 AuthInputs.jsx :

Hướng dẫn

```
<p>
  <Label invalid={emailNotValid}>Email</Label>
  <Input
    type="email"
    //className={emailNotValid ? "invalid" : undefined}
    invalid={emailNotValid}
    onChange={(event) => handleInputChange("email", event.target.value)}
  />
</p>
<p>
  <Label invalid={passwordNotValid}>Password</Label>
  <Input
    type="password"
    //className={passwordNotValid ? "invalid" : undefined}
    invalid={passwordNotValid}
    onChange={(event) => handleInputChange("password", event.target.value)}
  />
</p>
```

2

Styled components Điều Kiện

- ✓ Tương tự với Input

 AuthInputs.jsx :

```
const Input = styled.input`  
  width: 100%;  
  padding: 0.75rem 1rem;  
  line-height: 1.5;  
  // background-color: #d1d5db;  
  background-color: ${({invalid}) => (invalid ? "#fed2d2" : "#d1d5db")};  
  //color: #04265c;  
  color: ${({invalid}) => (invalid ? "#ef4444" : "#04265c")};  
  //border: 1px solid transparent;  
  border: 1px solid ${({invalid}) => (invalid ? "#f73f3f" : "transparent")};  
  border-radius: 0.25rem;  
  box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.1), 0 1px 2px 0 rgba(0, 0, 0, 0.06);  
`;
```

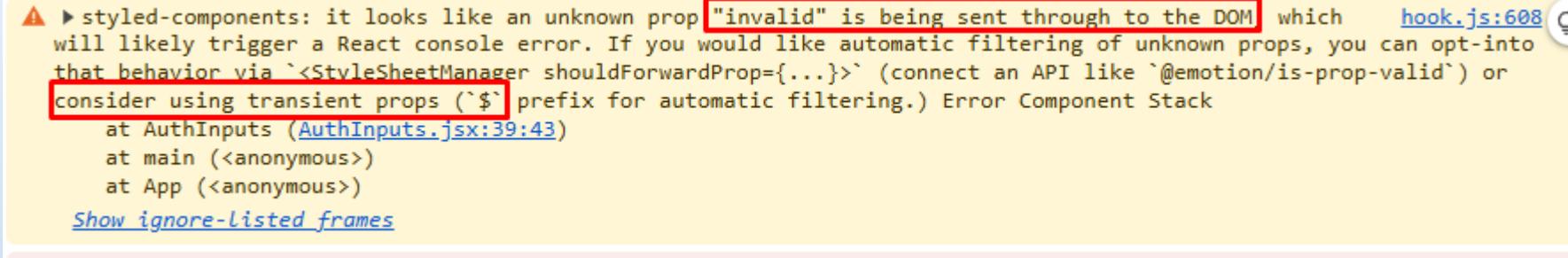
 index.css

Hướng dẫn

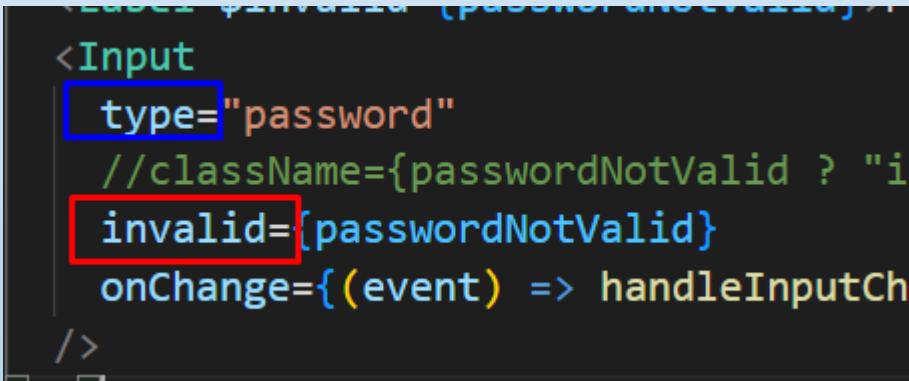
```
input.invalid {  
  color: #ef4444;  
  border-color: #f73f3f;  
  background-color: #fed2d2;  
}
```

2

Styled components Điều Kiện



```
⚠️ ► styled-components: it looks like an unknown prop "invalid" is being sent through to the DOM, which hook.js:608
will likely trigger a React console error. If you would like automatic filtering of unknown props, you can opt-into
that behavior via `<StyleSheetManager shouldForwardProp={...}>` (connect an API like `@emotion/is-prop-valid`) or
consider using transient props (`$` prefix for automatic filtering.) Error Component Stack
  at AuthInputs (AuthInputs.jsx:39:43)
  at main (<anonymous>)
  at App (<anonymous>)
Show ignore-listed frames
```



```
<Label invalid={passwordNotValid}>
  <Input
    type="password"
    //className={passwordNotValid ? "is-invalid" : "is-valid"}
    invalid={passwordNotValid}
    onChange={(event) => handleInputChange(event)}
  />
```

- ✓ *Cách viết này, invalid (là 1 prop tùy chỉnh coder tự đặt) sẽ :*
 1. *Được truyền vào trong component styled.input*
 2. *Nhưng nó cũng bị truyền tiếp xuống DOM thật như sau:* `<input invalid="true" />`
- ✓ *invalid không phải là một thuộc tính HTML hợp lệ (HTML không có attribute tên là invalid)*
- ✓ *React sẽ cảnh báo: "Received 'false' for a non-boolean attribute"*
- ✓ *Trình duyệt có thể bỏ qua hoặc xử lý sai khi thấy những attribute lạ như vậy*

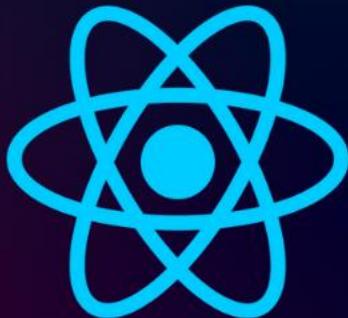
2

Styled components Điều Kiện

- ✓ "'*transient props*' trong styled-components chính là cách styled-components xử lý các props tùy chỉnh nội bộ (*không truyền xuống DOM*) một cách an toàn.

```
const Label = styled.label`  
  display: block;  
  margin-bottom: 0.5rem;  
  font-size: 0.75rem;  
  font-weight: 700;  
  letter-spacing: 0.1em;  
  text-transform: uppercase;  
  // color: #6b7280;  
  //color: ${props => (props.$invalid ? "#f87171" : "#fff")};  
  color: ${($invalid) => ($invalid ? "#f87171" : "#fff")};  
    
    
const Input = styled.input`  
  width: 100%;  
  padding: 0.75rem 1rem;  
  line-height: 1.5;  
  // background-color: #d1d5db;  
  background-color: ${($invalid) => ($invalid ? "#ef4444" : "#fff")};  
  //color: #04265c;  
  color: ${($invalid) => ($invalid ? "#ef4444" : "#04265c")};  
  //border: 1px solid transparent;
```

Cần thêm tiền tố \$ để invalid không bị truyền vào DOM thật mà chỉ truyền vào component styled



REACT JS



BÀI 17.7

React CSS

Pseudo Selectors,
Nested Rules &
Media Queries



3

Styled components - Pseudo Selectors, Nested Rules & Media Queries

 Header.module.css

```
header {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  justify-content: center;  
  margin-top: 2rem;  
  margin-bottom: 2rem;  
}  
  
header img {  
  object-fit: contain;  
  margin-bottom: 2rem;  
  width: 20rem;  
}
```

```
header h1 {  
  font-size: 1.5rem;  
  font-weight: 600;  
  letter-spacing: 0.4em;  
  text-align: center;  
  text-transform: uppercase;  
  color: #902e0d;  
  font-family: "Pacifico", cursive;  
  margin: 0;  
}
```

```
header p {  
  text-align: center;  
  color: #8b1717;  
  margin: 0;  
}
```

```
.my-paragraph {  
  color: rgb(0, 255, 55);  
}
```

```
@media (min-width: 768px) {  
  header {  
    margin-bottom: 4rem;  
  }  
  
  header h1 {  
    font-size: 2.25rem;  
  }  
}
```

 AuthInputs.jsx

```
</button>  
  <button className="button" onClick={ha  
  | Sign In  
  </button>
```

 index.css

```
.button:hover {  
  background-color: #f0920e;  
}
```

3

Styled components - Pseudo Selectors, Nested Rules & Media Queries

```
Header.jsx ×  
src > components > Header.jsx > Header  
1 import logo from "../assets/logo-tuhoc.png";  
2 // import "./Header.css";  
3 //import classes from "./Header.module.css";  
4 import {styled} from "styled-components";  
5  
6 const StyledHeader = styled.header`  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
    justify-content: center;  
    margin-top: 2rem;  
    margin-bottom: 2rem;  
`;
```

```
<header>  
    <img src={logo} alt="A canvas"  
    <h1>CSS trong React</h1>  
    <p>Tìm hiểu về css trong react  
    </header>  
`.
```



```
<StyledHeader>  
    <img src={logo} alt="A canvas" />  
    <h1>CSS trong React</h1>  
    <p>Tìm hiểu về css trong react cho ng  
    </StyledHeader>
```

3

Styled components - Pseudo Selectors, Nested Rules & Media Queries

```
header {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  justify-content: center;  
  margin-top: 2rem;  
  margin-bottom: 2rem;  
}  
  
header img {  
  object-fit: contain;  
  margin-bottom: 2rem;  
  width: 20rem;  
}  
  
header h1 {  
  font-size: 1.5rem;  
  font-weight: 600;  
  letter-spacing: 0.4em;  
  text-align: center;  
  text-transform: uppercase;  
  color: #902e0d;  
  font-family: "Pacifico", cursive;  
  margin: 0;  
}
```



Chúng ta gặp lỗi vì các mã css gắn liền với header trước đó

3

Styled components - Pseudo Selectors, Nested Rules & Media Queries

```
header img {  
    object-fit: contain;  
    margin-bottom: 2rem;  
    width: 20rem;  
}  
  
header h1 {  
    font-size: 1.5rem;  
    font-weight: 600;  
    letter-spacing: 0.4em;  
    text-align: center;  
    text-transform: uppercase;  
    color: #902e0d;  
    font-family: "Pacifico", cursive;  
    margin: 0;  
}  
  
header p {  
    text-align: center;  
    color: #8b1717;  
    margin: 0;  
}
```



```
& img {  
    object-fit: contain;  
    margin-bottom: 2rem;  
    width: 20rem;  
}  
  
& h1 {  
    font-size: 1.5rem;  
    font-weight: 600;  
    letter-spacing: 0.4em;  
    text-align: center;  
    text-transform: uppercase;  
    color: #902e0d;  
    font-family: "Pacifico", cursive;  
    margin: 0;  
}  
  
& p {  
    text-align: center;  
    color: #8b1717;  
    margin: 0;  
}
```

3

Styled components - Pseudo Selectors, Nested Rules & Media Queries

```
@media (min-width: 768px) {  
  header {  
    margin-bottom: 4rem;  
  }  
  
  header h1 {  
    font-size: 2.25rem;  
  }  
}
```



```
@media (min-width: 768px) {  
  & {  
    margin-bottom: 4rem;  
  }  
  
  & h1 {  
    font-size: 2.25rem;  
  }  
}
```

```
@media (min-width: 768px) {  
  margin-bottom: 4rem;  
  
  & h1 {  
    font-size: 2.25rem;  
  }  
}
```

3

Styled components - Pseudo Selectors, Nested Rules & Media Queries

- ✓ Tương tự với thẻ button , css phần hover lên nút

AuthInputs.jsx :

```
const Button = styled.button`  
padding: 1rem 2rem;  
font-weight: 600;  
text-transform: uppercase;  
border-radius: 0.25rem;  
color: #1f2937;  
background-color: #f0b322;  
border-radius: 6px;  
border: none;  
  
//lưu ý viết liền ←  
&:hover {  
background-color: #f0920e;  
}  
`;
```

```
<button type="button" className="text-button">  
| Create a new account  
</button>  
<button [className="button"] onClick={handleLogin}>  
| Sign In  
</button>
```

- ✓ Tổng kết:

- Dùng `&` để viết nested selector (img, h1... bên trong component).
- Dùng `&:hover`, `&:focus` cho pseudo selectors.
- Dùng media queries.



```
<button type="button" className="t  
| Create a new account  
</button>  
<Button onClick={handleLogin}>  
| Sign In  
</Button>
```