

ReactJs

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

CYBERSOFT

Tổng quan về reactjs

<https://cybersoft.edu.vn>

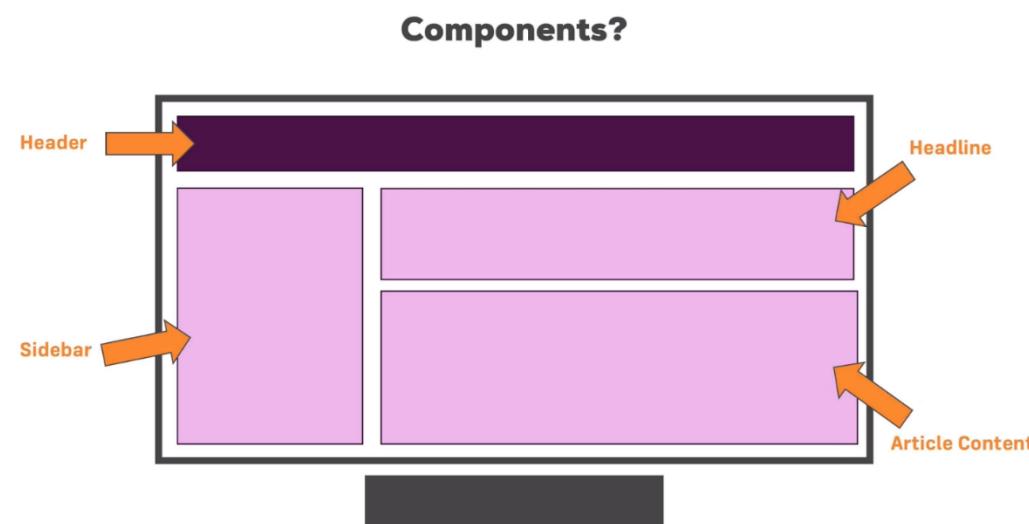
<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Giới thiệu về Reactjs

☐ **React là gì?**

- ❖ React là một thư viện javascript dùng để xây dựng giao diện sinh viên theo kiến trúc **component**



- ❖ React hỗ trợ xây dựng SPAs (Single page application)
- ❖ React sử dụng javascript chuẩn ES6 (giống với typescript những không có kiểu dữ liệu)

Cài đặt create-react-app

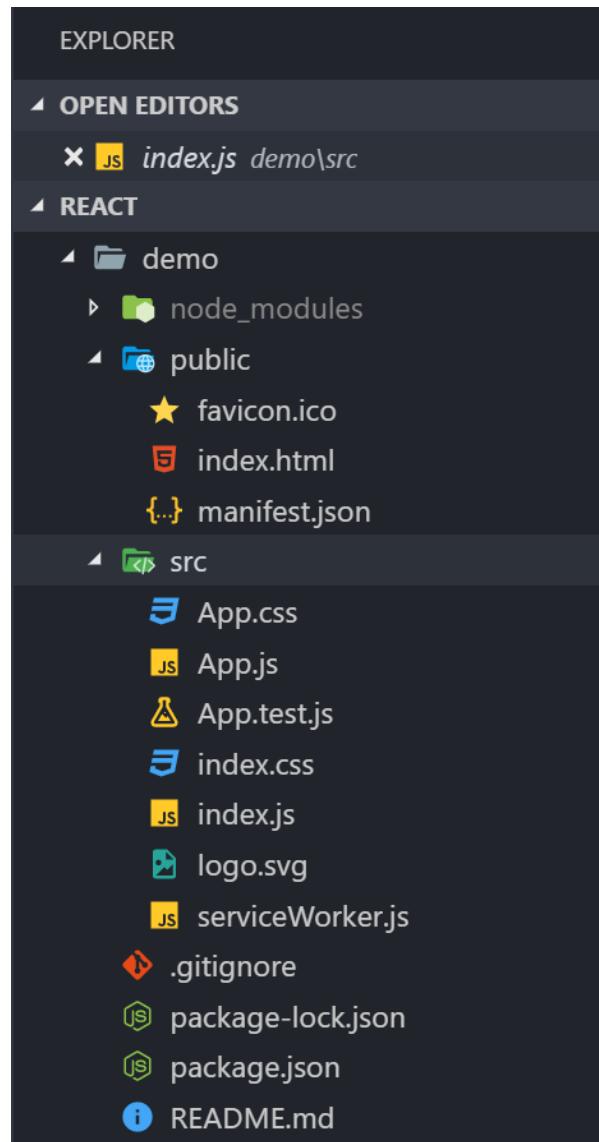
- **Create-react-app** là bộ công cụ tương tự angular-cli giúp ta tạo ra cấu trúc cho project
- Cài đặt **Create-react-app**: `npm install create-react-app -g`
- Tạo project react mới: `create-react-app tenproject`

The screenshot shows the Visual Studio Code interface. The title bar says "Welcome - react - Visual Studio Code". The left sidebar has icons for file operations like "New file", "Open folder...", and "Add workspace folder...". Below that is a "Recent" section with a link to "Quản lý Nhóm View - Tайл lên ChiaCode". The main area has tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL". The "TERMINAL" tab shows the command line output:
Microsoft Windows [Version 10.0.17134.471]
(c) 2018 Microsoft Corporation. All rights reserved.
d:\CYBERSOFT\cybersoft\slideFrontEnd\react>create-react-app demo
Creating a new React app in d:\CYBERSOFT\cybersoft\slideFrontEnd\react\demo.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...
[██████████] - extract:postcss: sill extract postcss@^7.0.0 extracted to d:\CYBERSOFT\cybersoft\slideFrontEnd\react\demo

A red box highlights the line "Installing packages. This might take a couple of minutes." in the terminal output.

On the right side of the terminal, there is a red annotation text: "create-react-app sẽ tự động cài đặt các package cần thiết để tạo nên react app".

Cấu trúc thư mục



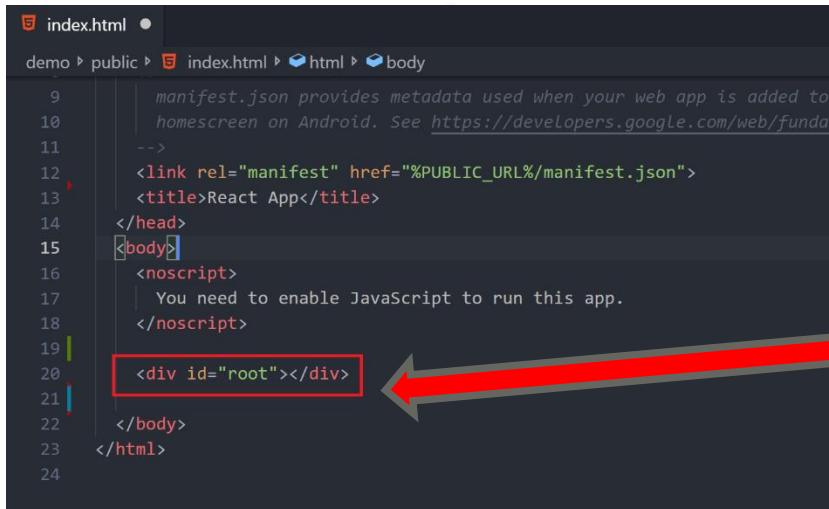
- ❑ **node_modules:** chứa các module được cài vào project
- ❑ **Public:** chứa file index.html và hình ảnh
- ❑ **Src:** chứa các component của ứng dụng
- ❑ **Package.json:** lưu lại các thông tin của project và các thư viện được cài vào.

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Luồng đi của ứng dụng

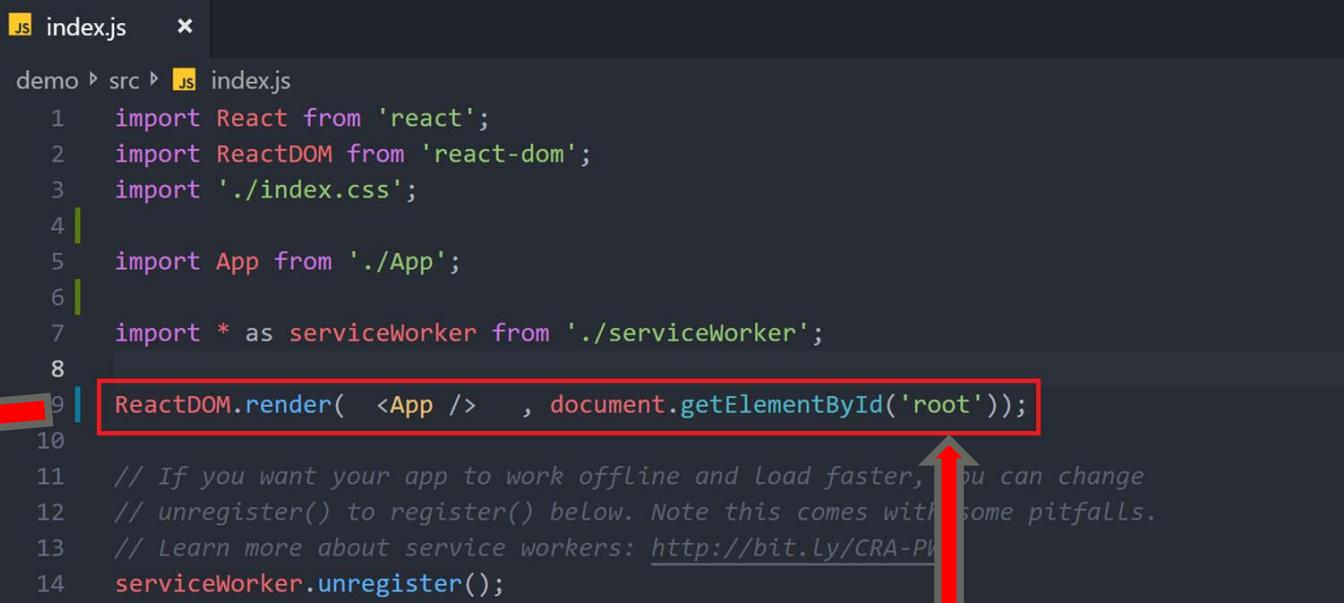
https://cybersoft.edu.vn



```

index.html
demo > public > index.html > html > body
9   manifest.json provides metadata used when your web app is added to
10  homescreen on Android. See https://developers.google.com/web/fundam...
11  -->
12  <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
13  <title>React App</title>
14  </head>
15  <body>
16  <noscript>
17      You need to enable JavaScript to run this app.
18  </noscript>
19  <div id="root"></div>
20  </body>
21  </html>
22
23
24
    
```

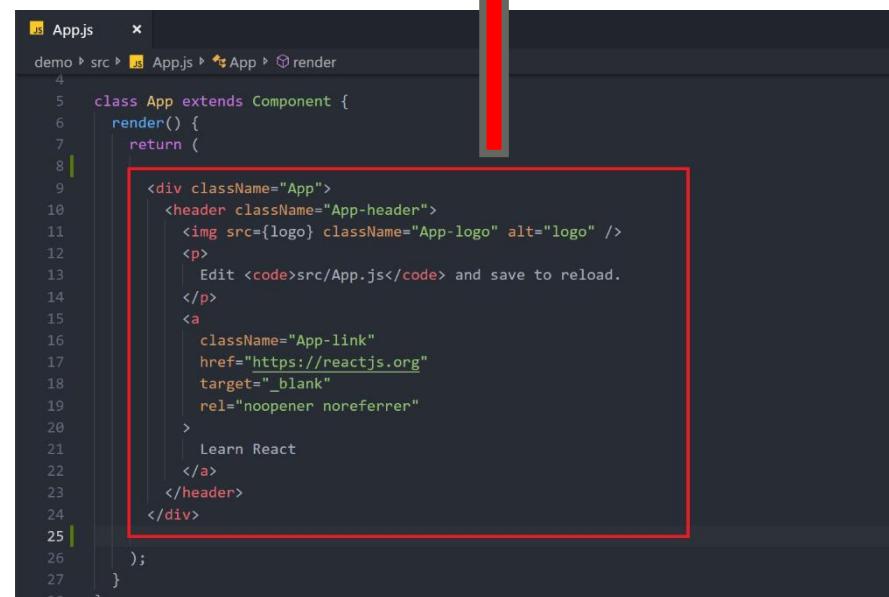
https:



```

index.js
demo > src > index.js
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4
5 import App from './App';
6
7 import * as serviceWorker from './serviceWorker';
8
9 ReactDOM.render( <App /> , document.getElementById('root'));
10
11 // If you want your app to work offline and load faster, you can change
12 // unregister() to register() below. Note this comes with some pitfalls.
13 // Learn more about service workers: http://bit.ly/CRA-PWA
14 serviceWorker.unregister();
    
```

https://cybersoft.edu.vn



```

App.js
demo > src > App.js > App > render
4
5 class App extends Component {
6     render() {
7         return (
8             <div className="App">
9                 <header className="App-header">
10                     <img src={logo} className="App-logo" alt="logo" />
11                     <p>
12                         Edit <code>src/App.js</code> and save to reload.
13                     </p>
14                     <a
15                         className="App-link"
16                         href="https://reactjs.org"
17                         target="_blank"
18                         rel="noopener noreferrer"
19                     >
20                         Learn React
21                     </a>
22                 </header>
23             </div>
24         );
25     }
26 }
27
    
```

- Đầu tiên, browser sẽ đọc được trang index.html

https://cybersoft.edu.vn

- Tiếp theo sẽ đọc tới file index.js, trong file này,

ReactDOM được sử dụng để render nội dung của của app component ra div root ở ngoài HTML

- App.js chính là component gốc của toàn ứng dụng

Component

- Component biểu diễn giao diện UI (file.html).
- Nói 1 cách đơn giản 1 component là 1 thẻ do mình định nghĩa trong thẻ đó chứa các nội dung html do mình biên soạn.
- Cấu trúc 1 component bao gồm:
- Có 2 loại component:
 - Stateless component (functional component)
 - Stateful component (class component)

```

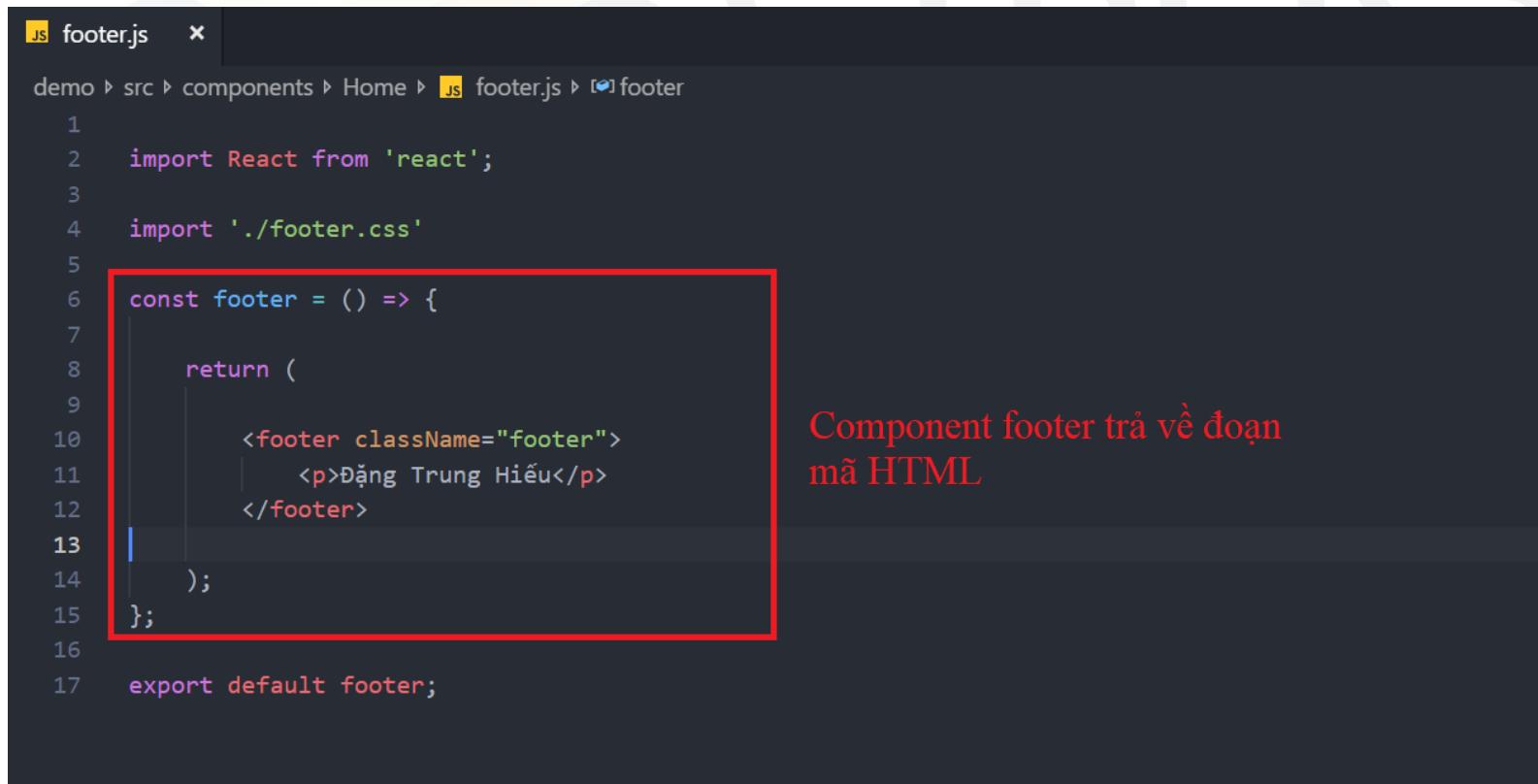
JS App.js
demo > src > JS App.js > App > render
1 import React, { Component } from 'react';
2 import './App.css'; import vào file css dùng để định dạng
3                                         cho component
4 class App extends Component { component thực chất chỉ là một class
5                                         được kế thừa từ Component của react
6   render() {
7     return (
8       Nội dung giao diện html sẽ được viết tại đây
9     );
10    }
11  }
12 }
13 }
14
15 export default App;
  
```

□ Khi component được load, hàm render sẽ chạy đầu tiên và return và một đoạn mã html.

Đây chính là giao diện được hiển thị lên browser

Thực hành tạo stateless component

- ☐ Stateless component (functional component) thực chất chỉ là một function ,return về một đoạn mã HTML để hiển thị ra giao diện
- ☐ Stateless component không thể sử dụng được **state** và **component lifecycle**

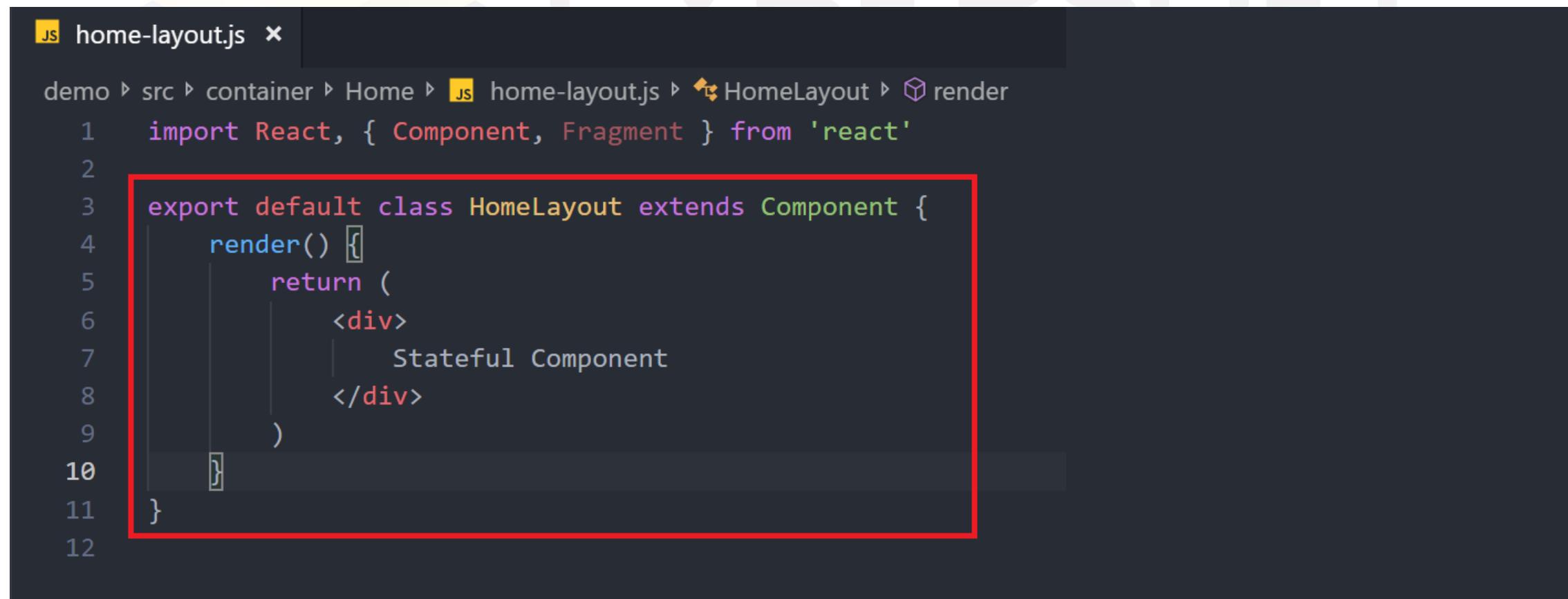


```
footer.js  x
demo › src › components › Home › footer.js  footer
1
2 import React from 'react';
3
4 import './footer.css'
5
6 const footer = () => {
7
8   return (
9
10     <footer className="footer">
11       <p>Đặng Trung Hiếu</p>
12     </footer>
13   );
14 };
15
16 export default footer;
```

Component footer trả về đoạn mã HTML

Thực hành tạo stateful component

- ☐ Stateful component (class component) thực chất chỉ là một class , có phương thức là **render()**
- ☐ Khi component được gọi, **render()** sẽ chạy và trả về đoạn mã HTML
- ☐ Stateful component có thể sử dụng được **state** và **component lifecycle**



JS home-layout.js ×

demo › src › container › Home › JS home-layout.js › HomeLayout › render

```
1 import React, { Component, Fragment } from 'react'
2
3 export default class HomeLayout extends Component {
4   render() {
5     return (
6       <div>
7         |   Stateful Component
8       </div>
9     )
10    }
11  }
12 }
```

Cấu trúc jsx

- **Đoạn mã được return trong component, thoát nhìn sẽ giống HTML , nhưng thực chất đó là jsx, cho phép chúng ra kết hợp HTML và js trên một source.**
- Một số điểm cần chú ý trong jsx:
 - **Class => className**
 - Các thẻ khuyết đóng phải đúng cú pháp, bắt buộc phải có "/". VD: ****
 - **For => htmlFor .Ví dụ <label htmlFor="dangnhap"></label>**

```
App.js
demo > src > App.js > ...
1 import React, { Component } from 'react';
2
3 import './App.css';
4
5 class App extends Component {
6   render() {
7     return (
8       <div className="app">
9         <h1>Cybersoft academy</h1>
10        </div>
11      );
12    }
13  }
14
15  export default App;
16
17
18
```

<https://cybersoft.edu.vn>

Cấu trúc jsx

- Khi lấy source code HTML từ nguồn khác, để sử dụng với react, ta cần convert ra thành jsx.
- Trang web hỗ trợ convert: <https://transform.now.sh/html-to-jsx/>
- *Copy code HTML vào ô bên trái, jsx được convert ra nằm ở ô bên phải*

The screenshot shows a comparison between two code editors. On the left, under 'HTML/SVG', is the following XML code:

```
<svg xmlns="http://www.w3.org/2000/svg" width="200" height="100" version="1.1">
  <rect width="200" height="100" stroke="black" stroke-width="6" fill="green"/>
</svg>
```

On the right, under 'JSX', is the converted JSX code:

```
;<svg width={200} height={100} version="1.1">
  <rect
    width={200}
    height={100}
    stroke="#000"
    strokeWidth={6}
    fill="green"
  />
</svg>
```

A blue 'Copy' button is visible at the top right of the JSX editor.

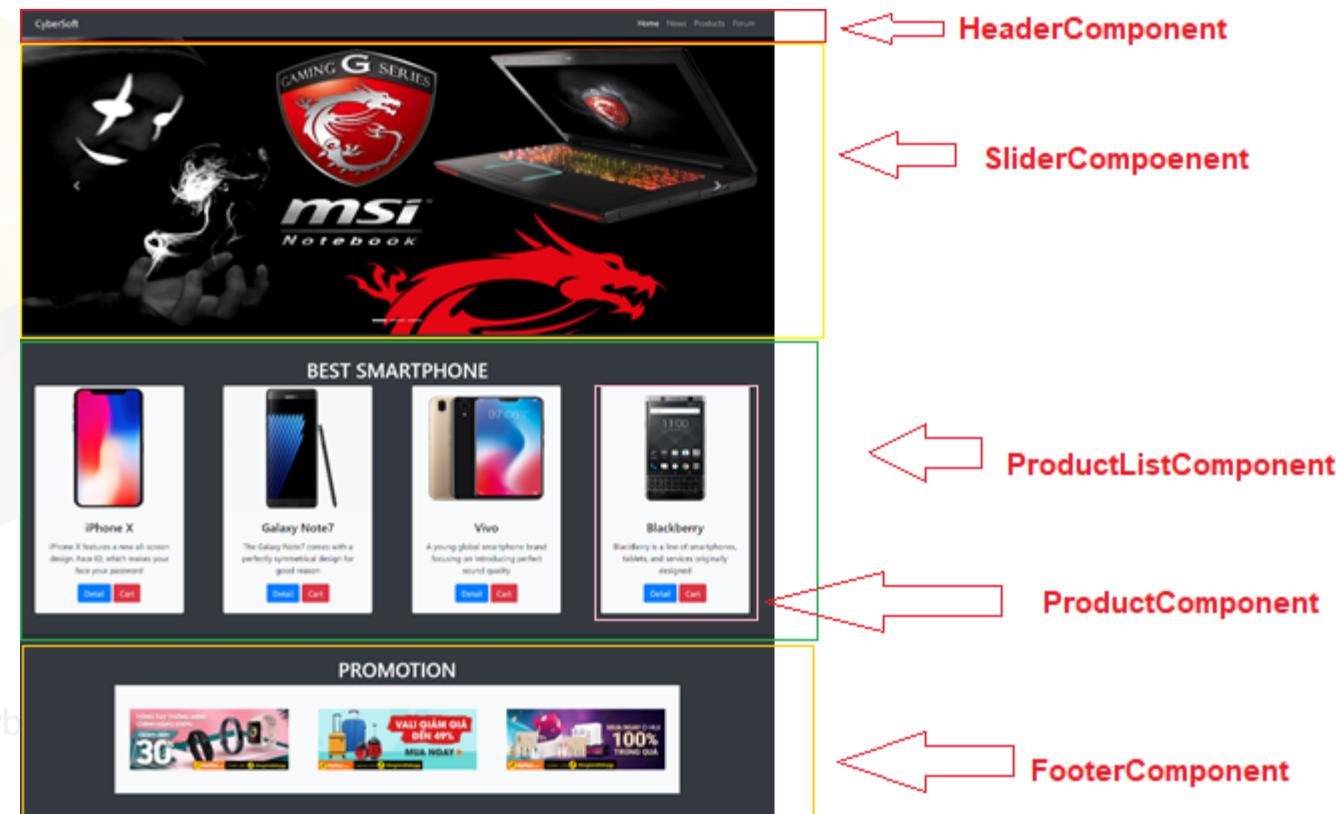
https://cybersoft.edu.vn

Có thể cài extention html to jsx : Để chuyển đổi từ code html thuần sang => code jsx

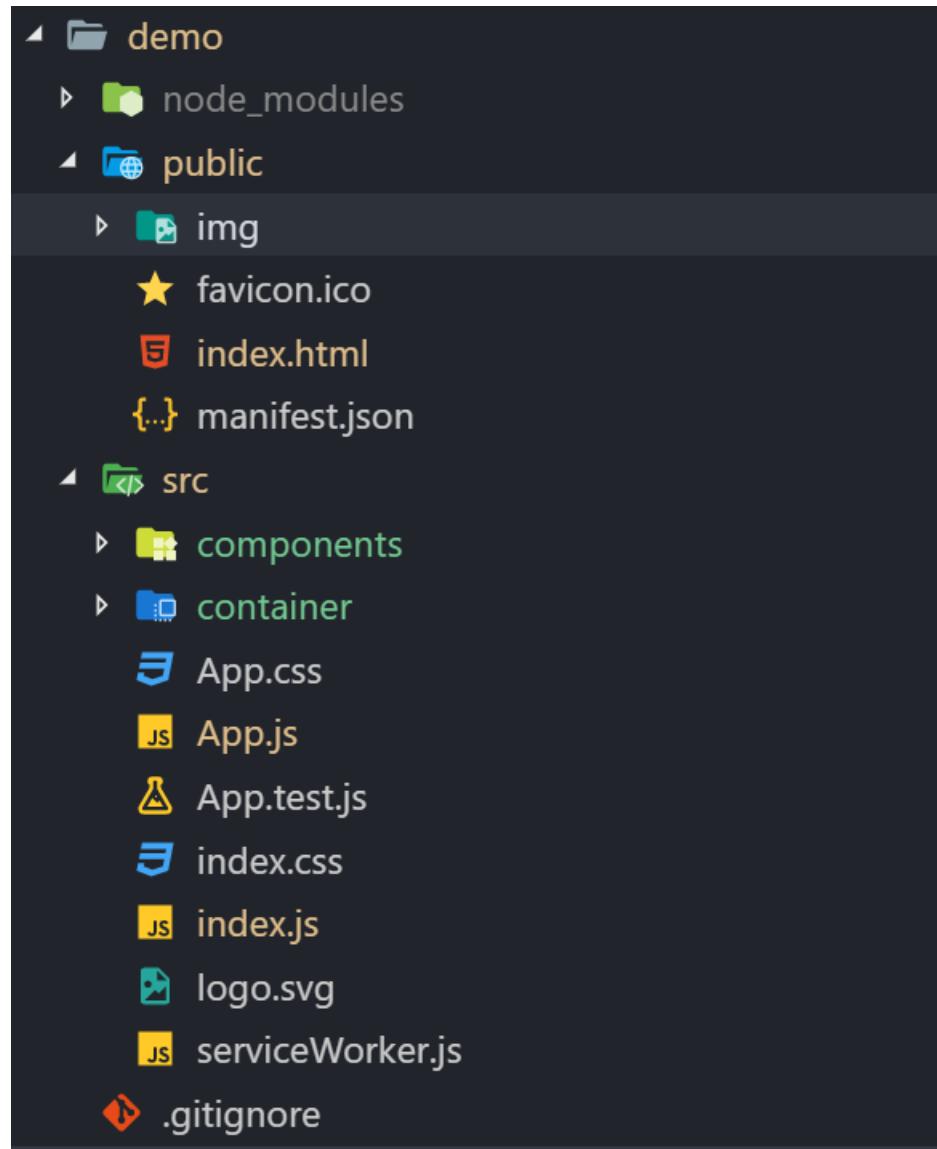
Bài tập: Tổ chức layout sau theo cấu trúc component

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>



Hướng dẫn : Tổ chức thư mục



<https://cybersoft.edu.vn>

- Thư mục hình, ta sẽ bỏ trong folder public**

<https://cybersoft.edu.vn>

- Ở folder src, ta tạo thêm 2 folder,**

**Components: để chứa các component nhỏ
lẻ cấu thành nên 1 trang: header, footer,
phim...**

- Containers: chứa các component ở mức
độ trang**

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Databinding reactjs

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Databinding trong reactjs

- Jsx cho phép ta lồng javascript vào HTML thông qua dấu { a }
- *title, product name, renderProduct... jsx cho phép chúng ta có thể render biến chuỗi hàm, ... tại phần nội dung miễn kết quả trả về là 1 đoạn jsx (cách viết HTML và js kết hợp)*
- *Javascript sẽ được parse và hiển thị ra chung với html*

JS App.js Product.jsx x

```

reactslide > src > components > Product.jsx > Product
1 import React, { Component } from 'react'
2
3 export default class Product extends Component {
4
5   render() {
6     let title = 'CYBERSOFT';
7     let productName = `<p><b>FrontEnd XXX</b></p>`;
8     return (
9
10       //Code jsx trả về
11       <div className="Product">
12         {title}
13         {productName}
14       </div>
15     )
16   }
17 }
18 
```

JS App.js Product.jsx x

```

reactslide > src > components > Product.jsx > ...
1 import React, { Component } from 'react'
2
3 export default class Product extends Component {
4
5   //Tạo ra 1 phương thức trả về 1 đoạn jsx
6   renderProduct = () => {
7     let title = 'CYBERSOFT';
8     let productName = `<p><b>FrontEnd XXX</b></p>`;
9
10    return (
11      <div>
12        {title}
13        {productName}
14      </div>
15    )
16  }
17
18  render() {
19
20    //Code jsx trả về
21    <div className="Product">
22      {this.renderProduct()} /* <= Phương thức render sẽ được gọi ở đây this.tenPhuongThuc() */
23    </div>
24  }
25
26 }

```

Xử lý sự kiện trong react

- ❑ Các sự kiện `onClick`, `onChange`, `onSubmit` ... trong javascript đều có thể sử dụng trong react. Tuy nhiên sẽ có khác biệt về cú pháp => Tham khảo ví dụ bên dưới
- ❑ Cú pháp: `sukien={callbackfunction}` => sự kiện là các sự kiện nêu trên, callback function là 1 function để xử lý cho sự kiện đó lưu ý: callback function gán vào không có 2 dấu () .

```
3  export default class App extends Component {  
4    handleClick = () => { //Phương thức xử lý sự kiện click cho button  
5      console.log()  
6    }  
7    render() {  
8      return (  
9        <div>  
10          /*Khi gán sự kiện Lưu ý ta truyền dưới dạng callback  
11          function */  
12          <button onClick={this.handleClick}>Show message</button>  
13        </div>  
14      )  
15    }  
16  }
```

Xử lý sự kiện trong react

- Câu hỏi đặt ra vậy nếu như ta muốn truyền 1 callback function xử lý sự kiện có tham số thì sẽ viết như thế nào
- Cú pháp `sukien={()=> callbackfunction(param)}` ta sẽ viết dưới dạng truyền 1 callbackfuntion **nặc danh (function không tên)** và function đó sẽ trả về 1 function có tham số khi thực thi => Khi gọi function đó

```
export default class App extends Component {  
  
    //Phương thức xử lý sự kiện click cho button  
    handleOnclik = (message) => {  
        console.log(message)  
    }  
  
    render() {  
        return (  
            <div>  
                /*Khi gán sự kiện Lưu ý ta truyền dưới dạng callback  
                function */  
                <button onClick={() => this.handleOnclik('hi khai')}>Show  
                    message</button>  
            </div>  
        )  
    }  
}
```

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

CYBERSOFT

Làm việc với lệnh điều kiện và vòng lặp – thuộc tính state (react stateful component)

<https://cybersoft.edu.vn>

Lệnh điều kiện trong jsx

- ☐ Kết hợp if else và hàm để xác định nội dung cần hiển thị trong react.
- ☐ Bằng cách xây dựng 1 hàm bên trong sử dụng lệnh if else để tùy biến nội dung jsx được render ra giao diện

```
1 import React, { Component } from 'react'
2
3 export default class App extends Component {
4
5   //Thuộc tính
6   isLoggedIn = true;
7   userName = 'khải';
8   //Phương thức checkLogin
9   renderContent = () => {
10     if(this.isLoggedIn){ //Dựa vào thuộc tính isLoggedIn để xác định hiển thị html là hello khải hoặc nút đăng nhập
11       return <b>Hello {this.userName}</b>
12     }
13     return <button>Đăng nhập</button>;
14   }
15   //Phương thức render
16   render() {
17     //Return về jsx hiển thị ra giao diện
18     return (
19       <div>
20         {this.renderContent()} /*Gọi phương thức renderContent để hiển thị nội dung ra giao diện */
21       </div>
22     )
23   }
24 }
```

Thuộc tính của class App

Dùng if else để hiển thị nội dung jsx

Gọi phương thức renderContent hiển thị nội dung cần render giao diện

Lệnh điều kiện tử trong jsx

- ❑ Ngoài ra mình cũng có thể sử dụng toán tử 3 ngôi để xác định phần nội dung hiển thị trực tiếp trên hàm render.

```
1 import React, { Component } from 'react'
2
3 export default class App extends Component {
4
5   //Thuộc tính
6   isLoggedIn = true;
7   userName = 'khải';
8
9   //Phương thức render
10  render() {
11    //Return về jsx hiển thị ra giao diện
12    return (
13      

14        {this.isLoggedIn ? <b>Hello {this.userName}</b> : ''}
15      </div>
16    )
17  }
18 }


```



Nếu giá trị isLoggedIn === true thì render ra ** ...**: biểu thức trước dấu : nếu isLoggedIn === false render ra: ''

Re-render với State trong react

- State là một thuộc tính mặc định của class kế thừa từ class component để quản lý trạng thái của component
- Mỗi khi state thay đổi, thì hàm render sẽ được gọi chạy lại. Lưu ý: Muốn component render lại ta phải thay đổi state thông qua phương thức setState chứ không được gán trực tiếp.
- Phương thức setState là 1 phương thức bất đồng bộ, có 2 tham số
 - + Tham số 1: giá trị state mới
 - + Tham số 2: callback thực thi ngay sau khi state thay đổi

Lưu ý:

Không được set lại giá trị state theo cách này: `this.state.thuocTinh = [giá trị]`

Ta set giá trị của state thông qua phương thức setState:

```
this.setState({
```

<https://cybersoft.edu.vn> `thuocTinh: [giá trị mới]`

```
)}
```

<https://cybersoft.edu.vn>

Sau khi gọi phương thức `setState()` giao diện sẽ được render lại

```
2 export default class App extends Component {  
3   //Phương thức khởi tạo của app Constructor  
4   constructor(props) {  
5     super(props); //Thuộc tính this.props có sẵn từ class Component  
6     this.state = { //Thuộc tính this.state có sẵn từ class Component  
7       isLoggedIn:false //Định nghĩa các biến làm thay đổi giao diện tại đây  
8     }  
9   }  
10 }  
11 //Phương thức Login  
12 login = () => {  
13   //Phương thức this.setState() là phương thức làm thay đổi giá trị state và gọi lại phương thức render  
14   this.setState({  
15     isLoggedIn: true  
16   });  
17   //Lưu ý: Không được gán this.state.isLoggedIn = false => gán như vậy hàm render sẽ không được gọi giao diện sẽ không được render lại  
18 }  
19 //Thuộc tính  
20 userName = 'khải';  
21 //Phương thức render  
22 render() {  
23   //Return về jsx hiển thị ra giao diện  
24   return (  
25     <div>  
26       {this.state.isLoggedIn ? <b>Hello {this.userName}</b> : <button onClick={this.login}>Login</button>}  
27     </div>  
28   )  
29 }  
30 }
```

Bài tập:

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Please choose your favorite about car's color

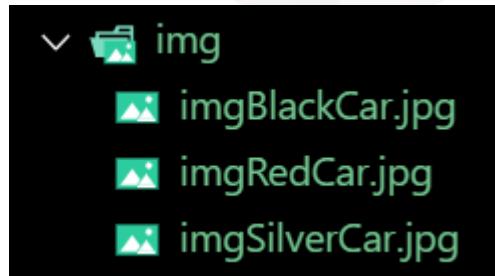


Change color

Red Color

Silver Color

Black Color



❖ Hướng dẫn:

Tạo 1 thư mục bài tập bên trong tạo 1 component
baitap1.jsx

Sử dụng state lưu trữ thuộc tính img của hình
Tạo các button tương ứng xử lý state thay đổi

Render giao diện với mảng dữ liệu dùng vòng lặp

- ❖ Để tạo ra các tag html (jsx trong react) thì ta có rất nhiều cách thực hiện, dùng các hàm như for, foreach, map ... tạo nội dung.
- ❖ Ví dụ bên dưới ta có 1 mảng các sản phẩm => yêu cầu render ra table sản phẩm đó

```
1 import React, { Component } from 'react';
2
3 export default class BaiTapVongLap extends Component {
4
5   constructor(props) {
6     super(props);
7     this.state = {
8       mangSanPham: [
9         { maSP: 1, tenSP: 'Iphone X', gia: 1000 },
10        { maSP: 2, tenSP: 'Huawei mate 20 pro', gia: 2000 },
11        { maSP: 3, tenSP: 'Samsung galaxy note 10', gia: 3000 }
12      ]
13    }
14 }
```

❖ Viết 1 hàm render kết quả trả về là 1 mảng các tag jsx là thẻ `<tr>`.

❖ Lưu ý: tag được tạo ra từ vòng lặp luôn phải có 1 thuộc tính key không trùng nhau

```
renderTable = () => {
  let contentTable = [];
  for (let i = 0; i < this.state.mangSanPham.length; i++) {
    let sp = this.state.mangSanPham[i]; //Lấy ra sản phẩm
    contentTable.push( //Tạo thẻ tr jsx push vào mảng kết quả
      <tr key={i}>
        <td>{sp.maSP}</td>
        <td>{sp.tenSP}</td>
        <td>{sp.gia}</td>
      </tr>
    )
  }
  return contentTable; //Return về mảng kết quả là 1 mảng các tag <tr></tr>
}
```

<https://cybersoft.edu.vn>

❖ Gọi hàm tại thẻ body để giao diện
render ra các thẻ `<tr>` tại đó

```
31   render() {
32     return (
33       <div className="container">
34         <table className="table">
35           <thead>
36             <tr>
37               <th>Mã SP</th><th>Tên SP</th><th>Giá</th>
38             </tr>
39           </thead>
40           <tbody>
41             {this.renderTable()}
42           </tbody>
43         </table>
44       </div>
45     )
46   }
```

- ❖ Ngoài ra chúng ta có thể viết phương thức `renderTable` dưới dạng hàm `map()`

```
renderTable = () => {
  return this.state.mangSanPham.map((sp, index) => {
    return (
      <tr key={index}>
        <td>{sp.maSP}</td>
        <td>{sp.tenSP}</td>
        <td>{sp.gia}</td>
      </tr>
    )
  })
}
```

edu.vn

<https://cybersoft.edu.vn>

- ❖ Hoặc viết trực tiếp trên giao diện code render trên giao diện

```
37
38
39
40
41
42
43
44
45
46
47
48
<tbody>
  { //Định nghĩa trực tiếp dưới phương thức render
    this.state.mangSanPham.map((sp, index) => {
      return (
        <tr key={index}>
          <td>{sp.maSP}</td>
          <td>{sp.tenSP}</td>
          <td>{sp.gia}</td>
        </tr>
      )
    })
  }
</tbody>
```

<https://cybersoft.edu.vn>

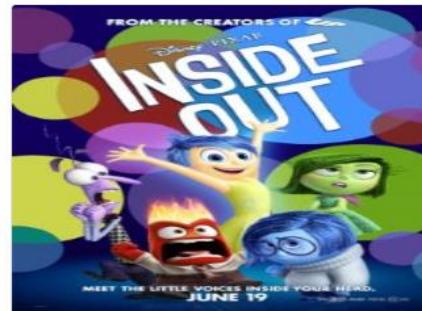
Bài tập: Sử dụng map hoặc for duyệt danh sách phim và in ra kết quả như hình bên dưới

Danh sách phim



Trainwreck

Having thought that monogamy was never possible, a commitment-phobic career woman may have to face h ...



Inside Out

After young Riley is uprooted from her Midwest life and moved to San Francisco, her emotions - Joy - ...



Home

Oh, an alien on the run from his own people, lands on Earth and makes friends with the adventurous T ...



Batman v Superman: Dawn of Justice

Fearing the actions of a god-like Super Hero left unchecked, Gotham City's own formidable, forceful ...



Ant-Man

Armed with a super-suit with the astonishing ability to shrink in scale but increase in strength, ca ...



Jurassic World

A new theme park is built on the original site of Jurassic Park. Everything is going well until the ...

Hướng dẫn:

- Trong thư mục bài tập tạo tiếp 1 component baitap2.jsx
- Tạo thêm 1 folder data chứa file json cung cấp import vào baitap2 component
- Sử dụng data đó dùng hàm map để load dữ liệu như hình bên trái

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

CYBERSOFT

Truyền dữ liệu trong reactjs

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Props (Truyền dữ liệu từ component cha sang con)

Props là gì ?

- Props là thuộc tính của thẻ (Ta có thể hiểu prop là property của thẻ). Ví dụ thẻ input bên dưới ta có các **props** **ClassName**, **type**, **placeholder**

```
<input className="form-control" type="text" placeholder="Nhập họ tên"/>
```

Props đối với component

- Props là thuộc tính mặc định của component để nhận dữ liệu từ các giá trị component cha truyền vào => Để binding dữ liệu ra component con tại html tương ứng
- Props của component chỉ nhận các thuộc tính được truyền vào từ component cha của nó và không thể bị chỉnh sửa bên trong component
- Đối với stateful và stateless component có các cách sử dụng props khác nhau.

Props trong stateless component

<https://cybersoft.edu.vn>

□ **Footer component (con)**

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

```
import './footer.css'    Truyền props vào dưới dạng  
                        tham số
```

```
const footer = (props) => {
```

```
    return (
```

```
        <footer className="footer">  
            <p>{props.name}</p>
```

sử dụng props, trong đó name là thuộc tính được truyền vào từ component cha, ở đây sử dụng footer component

```
    );
```

```
export default footer;
```

Export để có thể gọi ở các component khác

<https://cybersoft.edu.vn>

□ **Homelayout component (cha)**

<https://cybersoft.edu.vn>

```
import React, { Component, Fragment } from 'react'
```

```
import Footer from '../../components/Home/footer';
```

```
export default class HomeLayout extends Component {
```

```
    name = "dang trung hieu";
```

```
    render() {
```

```
        return (
```

```
            //Code Jsx được trả về
```

```
            <Footer name={'hieu'}></Footer>
```

Gọi footer component ra như một thẻ HTML thông thường

Trong đó name là thuộc tính ta truyền vào footerComponent với giá trị là chuỗi 'hieu'. Thay vì truyền chuỗi, ta có thể truyền vào biến name

Dó là ví dụ vì sao props của footer component lại có thuộc tính name

import footer component để có thể gọi nó ra trong homelayout component.

Vì bên footer component ta export default nên bên này có thể đặt lại tên cho nó tùy ý và không cần dấu

<https://cybersoft.edu.vn>

Props trong statefull component

- Đối với stateful component, thì props là thuộc tính mặc định của class, nên không cần truyền tham số. Chỉ cần gọi **this.props**

□ **Header component (con)**

```
demo › src › components › Home › header.js › default
1  import React, { Component } from 'react'
2
3  export default class extends Component {
4      render() {
5          return (
6              <header>
7                  <p> {this.props.name} </p>
8              </header>
9          )
10     }
11 }
12 }
```

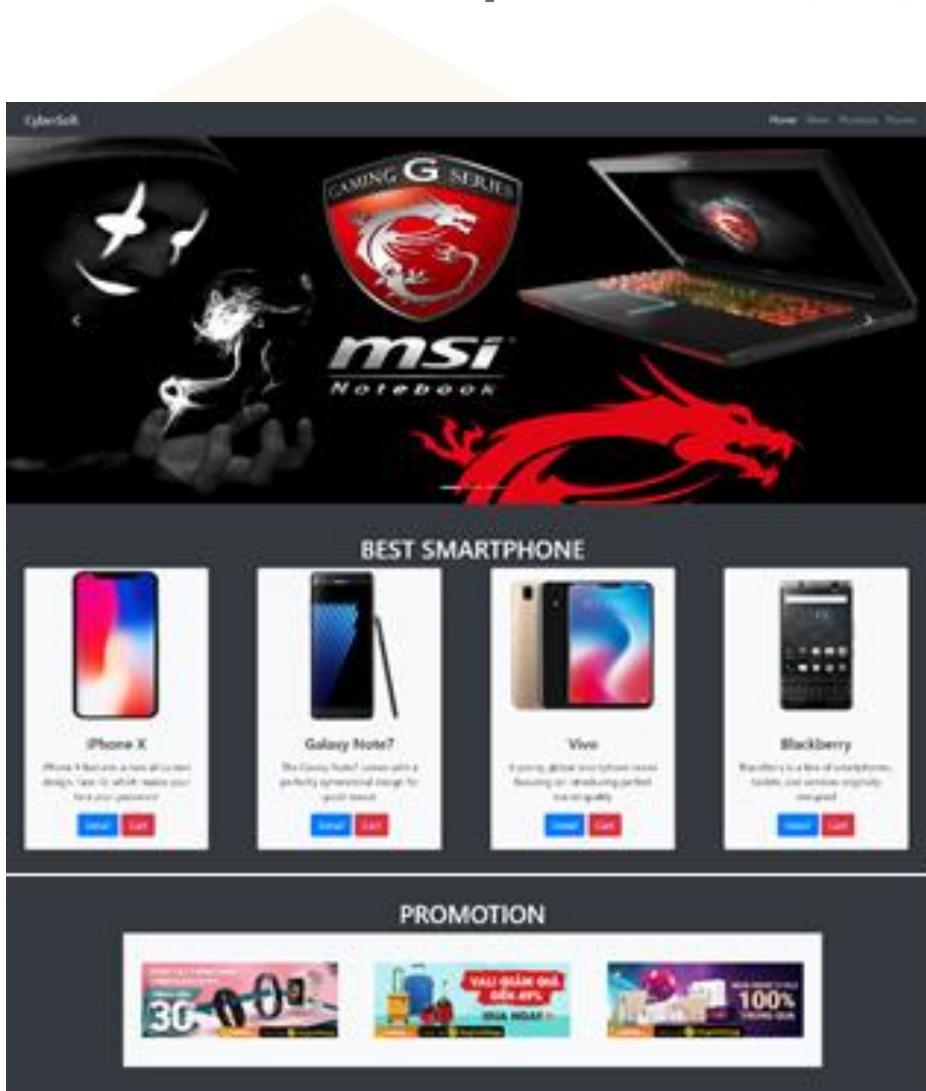
□ **Homelayout component (cha)**

```
import React, { Component, Fragment } from 'react'

import Header from '../../components/Home/header';

export default class HomeLayout extends Component {
    name = "dang trung hieu";
    render() {
        return [
            //Code Java được trả về
            <Header name={this.name}></Header>
        ]
    }
}
```

Bài tập: quay lại layout bán hàng, dùng props để tạo ra các item sản phẩm có dữ liệu khác nhau



<https://cybersoft.edu.vn>

□ Bài tập:

- Sử dụng hàm map để load component product.
- Trong lúc render ta tiến hành truyền props. Để hiển thị dữ liệu tương ứng. Với dữ liệu bên dưới

```
[  
  { maSP: 1, tenSP: 'Black Berry',hinhAnh:'./img/sp_blackberry.png', gia: 1000 },  
  { maSP: 2, tenSP: 'Iphone X',hinhAnh:'./img/sp_iphoneX.png', gia: 2000 },  
  { maSP: 3, tenSP: 'Note 7',hinhAnh:'./img/sp_note7.png', gia: 3000 },  
  { maSP: 3, tenSP: 'Vivo 850',hinhAnh:'./img/vivo850.png', gia: 3000 }  
]
```

<https://cybersoft.edu.vn>

❖ Truyền sự kiện – nhận dữ liệu các cấp component

<https://cybersoft.edu.vn>



Vấn đề đặt ra:
Ta có 2 component là `DanhSachSanPham`, `SanPham`

+ `DanhSachSanPham` chứa: 2 thuộc tính là

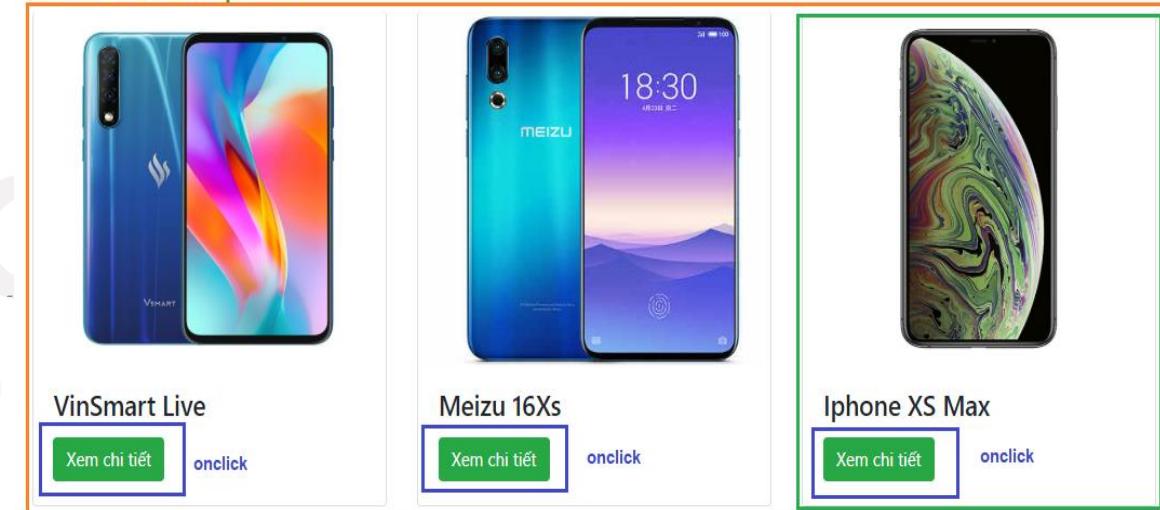
- ◆ `mangSanPham`
- ◆ `state.sanPhamChiTiet`

Yêu cầu:

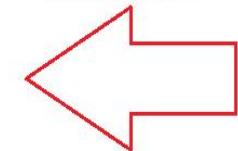
- Khi click vào nút xem chi tiết thì thông tin của sản phẩm được click sẽ được cập nhật vào `this.state.sanPhamChiTiet` làm cho nội dung bên dưới thay đổi
- Điều này thật đơn giản khi ta binding tất cả html chỉ trên 1 component duy nhất `DanhSachSanPham`.
- Tuy nhiên ta đã tách các `sanPham` ra thành riêng 1 component nên bên trong class của component `sanPham` không chứa `state.sanPhamChiTiet` nữa nên ta không thể cập nhật được.
- Do vậy tại component cha bắt buộc ta phải viết 1 hàm để truyền vào sự kiện click của component con để sau khi click thì nó sẽ lấy được dữ liệu tại hàm ta xây dựng ở component cha từ đó cập nhật lại `state.sanPhamChiTiet` => làm cho giao diện thay đổi

<https://cybersoft.edu.vn>

Danh sách sản phẩm



`this.mangSanPham`

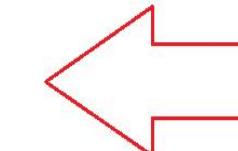


VinSmart Live

Thông số kỹ thuật

Màn hình	AMOLED, 6.2", Full HD+
Hệ điều hành	Android 9.0 (Pie)
Camera trước	20 MP
Camera sau	Chính 48 MP & Phụ 8 MP, 5 MP
RAM	4 GB
ROM	64 GB

`this.state.sanPhamChiTiet`



❖ Truyền sự kiện – nhận dữ liệu các cấp component

- ❑ Code: <https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❑ DanhSachSanPham component:

```
4  export default class DanhSachSanPham extends Component {  
5  
6      constructor(props) {  
7          super(props);  
8          this.state = {  
9              sanPhamChiTiet: { maSP: 1, tenSP: 'VinSmart Live', manHinh: 'AMOLED, 6.2", Full HD+', heDieuHanh: 'Android 9.0 (Pie)',  
10                 cameraTruoc: '20 MP', cameraSau: 'Chính 48 MP & Phụ 8 MP, 5 MP', ram: '4 GB', rom: '64 GB', giaBan: 5700000, hinhAnh: './img/  
11                 vsphone.jpg' },  
12             }  
13         }  
14         mangDienThoai = []  
15         { maSP: 1, tenSP: 'VinSmart Live', manHinh: 'AMOLED, 6.2", Full HD+', heDieuHanh: 'Android 9.0 (Pie)', cameraTruoc: '20 MP',  
16             cameraSau: 'Chính 48 MP & Phụ 8 MP, 5 MP', ram: '4 GB', rom: '64 GB', giaBan: 5700000, hinhAnh: './img/vsphone.jpg' },  
17         { maSP: 2, tenSP: 'Meizu 16Xs', manHinh: 'AMOLED, FHD+ 2232 x 1080 pixels', heDieuHanh: 'Android 9.0 (Pie); Flyme', cameraTruoc:  
18             '20 MP', cameraSau: 'Chính 48 MP & Phụ 8 MP, 5 MP', ram: '4 GB', rom: '64 GB', giaBan: 7600000, hinhAnh: './img/meizuphone.jpg' },  
19         { maSP: 3, tenSP: 'Iphone XS Max', manHinh: 'OLED, 6.5", 1242 x 2688 Pixels', heDieuHanh: 'iOS 12', cameraSau: 'Chính 12 MP & Phụ  
12 MP', cameraTruoc: '7 MP', ram: '4 GB', rom: '64 GB', giaBan: 27000000, hinhAnh: './img/applephone.jpg' }  
20     ]
```

❖ Truyền sự kiện – nhận dữ liệu các cấp component

- ❑ Code: <https://cybersoft.edu.vn> <https://cybersoft.edu.vn>
- ❑ DanhSachSanPham component: <https://cybersoft.edu.vn>

```
xemChiTiet = (sanPham) => {
  this.setState({
    sanPhamChiTiet: sanPham
  })
}

render() {
  let {sanPhamChiTiet} = this.state;      state.sanPhamChiTiet
  return (
    <div className="container">
      <h3 className="text-success">Danh sách sản phẩm</h3> Binding mang SanPham
      <div className="row">
        {
          this.mangDienThoai.map((sanPham, index) => {
            return (
              <div className="col-md-4" key={index}>
                <SanPham xemChiTiet={this.xemChiTiet} sanPham={sanPham} />
              </div>
            )
          })
        }
      </div>
      <div className="clearfix"></div>
      <hr />
    </div>
  )
}
```

❖ Truyền sự kiện – nhận dữ liệu các cấp component

- ❑ Code: <https://cybersoft.edu.vn>
- ❑ Binding dữ liệu từ state => table chi tiết

```
<div className="row">
  <div className="col-md-4">
    <br />
    <h3 className="text-center">{sanPhamChiTiet.tenSP}</h3>
    <img className="card-img-top" src={sanPhamChiTiet.hinhAnh} width={170} height={300} alt={sanPhamChiTiet.tenSP} />
  </div>
  <div className="col-md-8">
    <table className="table">
      <thead>
        <tr>
          <td colSpan="2"><h3>Thông số kỹ thuật</h3></td>
        </tr>
        <tr>
          <td>Màn hình</td>
          <td>{sanPhamChiTiet.manHinh}</td>
        </tr>
        <tr>
          <td>Hệ điều hành</td>
          <td>{sanPhamChiTiet.heDieuHanh}</td>
        </tr>
        <tr>
          <td>Camera trước</td>
          <td>{sanPhamChiTiet.cameraTruoc}</td>
        </tr>
        <tr>
          <td>Camera sau</td>
          <td>{sanPhamChiTiet.cameraSau}</td>
        </tr>
        <tr>
          <td>RAM</td>
          <td>{sanPhamChiTiet.ram}</td>
        </tr>
        <tr>
          <td>ROM</td>
          <td>{sanPhamChiTiet.rom}</td>
        </tr>
      </thead>
      <tbody>
        </tbody>
    </table>
  </div>
</div>
```

❖ Truyền sự kiện – nhận dữ liệu các cấp component

❑ Code: <https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

DemoCallback.jsx ● SanPham.jsx X

demo > src > components > Demo > SanPham.jsx > ...

```
1 import React, { Component } from 'react'
2
3 export default class SanPham extends Component {
4
5     render() {
6         let {sanPham} = this.props;
7         let {xemChiTiet} = this.props;
8         //Hoặc có thể khai báo
9         // let {sanPham,xemChitiet} = this.props;
10        return (
11            <div className="card text-left">
12                <img className="card-img-top" src={sanPham.hinhAnh} width={170} height={300} alt={'true'} />
13                <div className="card-body">
14                    <h4 className="card-title">{sanPham.tenSP}</h4>
15                    <button className="btn btn-success" onClick={() => xemChiTiet(sanPham)}>Xem chi tiết</button>
16                </div>
17            </div>
18        )
19    }
20 }
```

Prop và sự kiện nhận từ component cha

Sử dụng hàm từ component cha để truyền vào sự kiện click



Bài tập: Xây dựng chức năng tiếp theo cho trang bán hàng online: Khi click vào nút card, hiện ra modal hiển thị thông tin của sản phẩm



- Gợi ý: Sử dụng bài tập bán hàng đã dàn layout ở phần 1**
- Dùng danh sách sản phẩm được cho render tự động**
- Tạo thêm component <Modal> với các cấp**
- <app>**
- <BaiTap1>**
- <Header />**
- <Slider />**
- <ProductList>**
- <Product />**
-**
- <Product />**
- </ProductList>**
- <Modal />**
- </BaiTap1>**
- </app>**
- Thực hiện chức năng click vào nút xem chi tiết hiển thị thông tin sản phẩm lên modal**

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Bài tập: Xây dựng chức năng giỏ hàng như hình bên dưới

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

The screenshot shows a mobile application interface. At the top, there is a navigation bar with icons for home, search, and account. Below it, a large banner displays a smartphone. The main content area has a header "Bài tập giỏ hàng". A modal window titled "Giỏ hàng" is open, showing a single item: "VinSmart Live" with a quantity of 1, a price of 5700000, and a total of 5700000. There are buttons to increase or decrease the quantity and a red "Xóa" (Delete) button. To the right of the modal, a small image of an iPhone XS Max is visible. Below the modal, there is a section for "Thông số kỹ thuật" (Technical specifications) for the VinSmart Live, listing details like screen type (AMOLED, 6.2", Full HD+), operating system (Android 9.0 (Pie)), camera (20 MP front, 48 MP + 8 MP + 5 MP rear), RAM (4 GB), and ROM (64 GB). At the bottom, there are three cards for other products: "VinSmart Live", "Meizu 16Xs", and "Iphone XS Max", each with "Xem chi tiết" and "Thêm giỏ hàng" buttons.

- Gợi ý: Sử dụng lại ví dụ truyền sự kiện**
- Dùng lại source bài truyền dữ liệu**
- Xây dựng chức năng cho nút thêm giỏ hàng**
- <app>
- <BaiTap>
- <Modal />
- <DanhSachSanPham>
- <SanPham />
- <SanPham />
- <SanPham />
- </DanhSachSanPham>
- </BaiTap>
- </app>
- Xác định thành phần dữ liệu của giỏ hàng có thay đổi hay không ? Nếu có thì dùng gì để quản lý nguồn dữ liệu đó và nên đặt dữ liệu đó ở component nào?**
- Gợi ý: Sử dụng callback function**

Redux



<https://cybersoft.edu.vn>

CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Giới thiệu về Redux

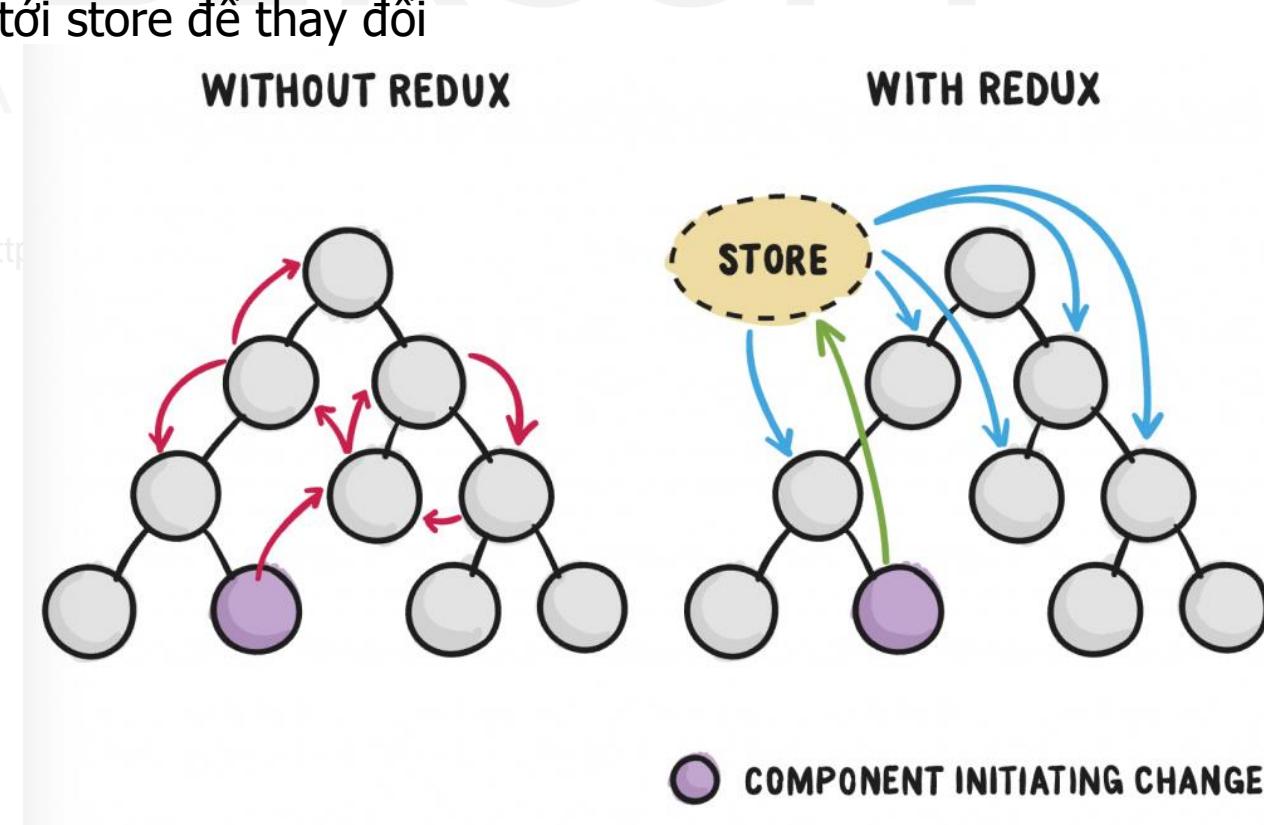
<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- Bản chất làm việc với React là việc truyền dữ liệu giữa các component và thay đổi state để re-render lại giao diện component
- Redux là thư viện cung cấp cho ta một store trung tâm, lưu trữ tất cả các state, từ component muốn thay đổi state chỉ cần truy cập tới store để thay đổi

□ Ví dụ:

- Ta có 2 component là DSSV và SinhVien
- Thay vì lưu trữ danh sách tại DSSV component, mỗi lần ta muốn xóa sinh viên hay cập nhật lại sinh viên thì phải truyền dữ liệu từ component SinhVien ra DSSV .
- Với redux , ta sẽ để danh sách ở store trung tâm, từ bất kỳ component nào muốn thay đổi danh sách sinh viên (CRUD), chỉ cần đi trực tiếp tới store đổi, không phải truyền qua lại giữa các



Cấu trúc của redux

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- Redux bao gồm 3 thành phần chính

- Action:Là nơi mang các thông tin dùng để gửi từ ứng dụng đến Store. Các thông tin này là 1 object mô tả những gì đã xảy ra . Nói dễ hiểu, từ 1 component, ta muốn thay đổi state trên store, ta phải gửi action , là một object để miêu tả muốn làm gì.

- Reducer: nơi tiếp nhận action và thay đổi state.Gồm 2 loại:

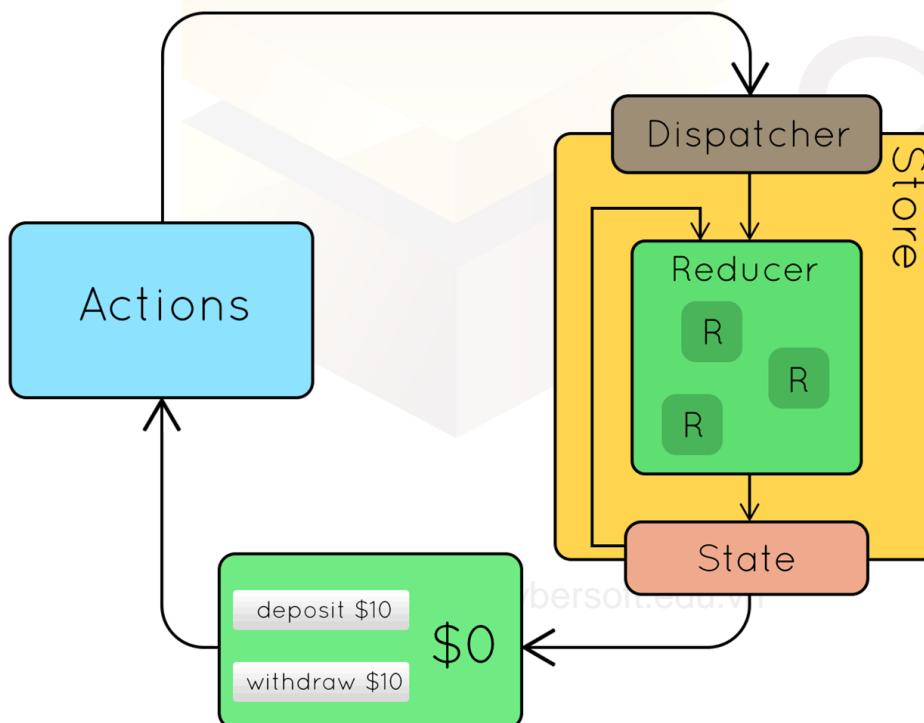
- Root Reducer: là Boss, quản lý tất cả reducer con
 - Child Reducer: như đã biết về state, state là một object có nhiều thuộc tính, mỗi child reducer chịu trách nhiệm thay đổi 1 thuộc tính trong state.
 - Store:Nơi lưu trữ và quản lý state (Chính là Big Boss)

<https://cybersoft.edu.vn>

Sơ đồ hoạt động của Redux

<https://cybersoft.edu.vn>

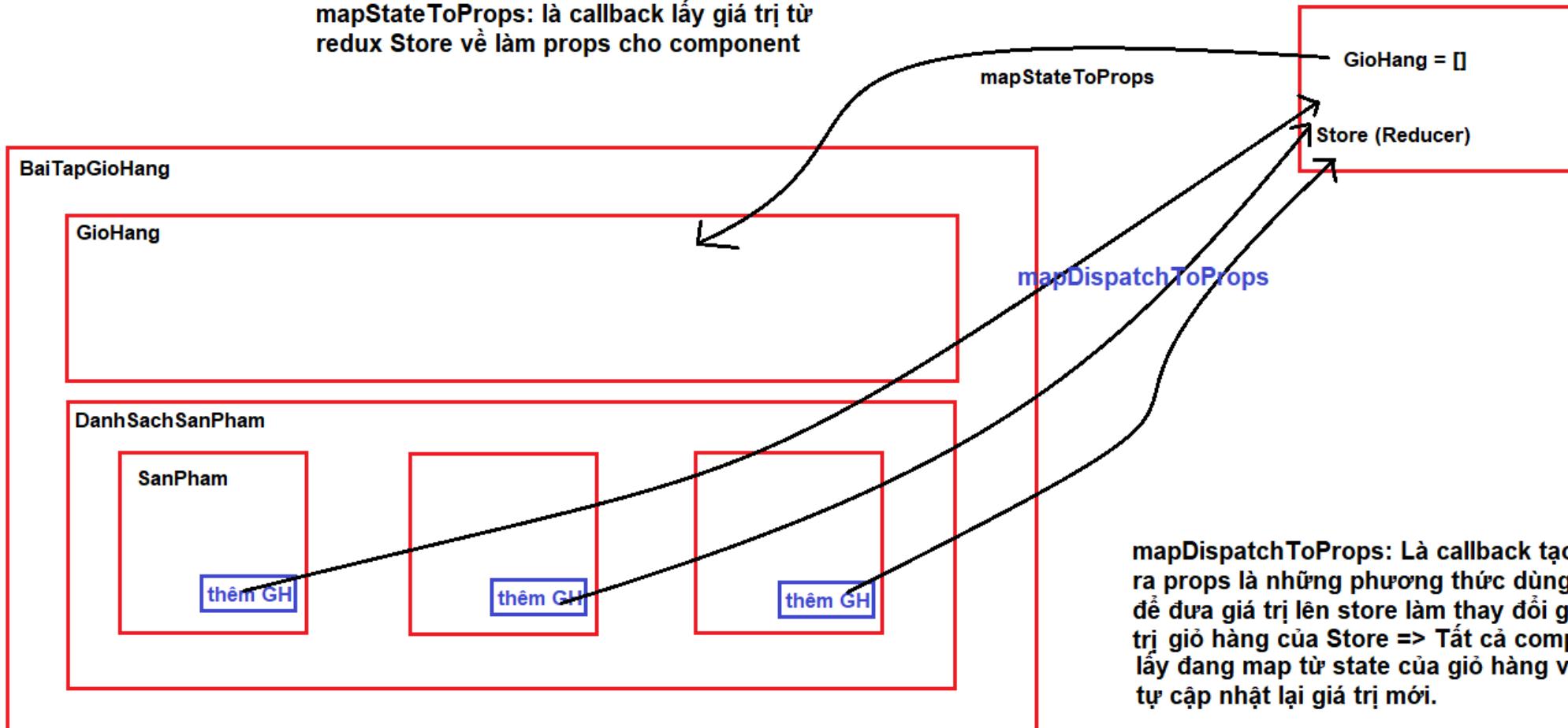
<https://cybersoft.edu.vn>



- ❑ Ví dụ ta có 2 component Form và component DanhSachSinhVien
- ❑ Ta muốn từ component Form, sẽ thêm một sinh viên vào danh sách.
- ❑ Đầu tiên, ta sẽ để cho store lưu trữ state với thuộc tính là dssv state={dssv: []}
- ❑ Từ component Form, ta sẽ gửi một action, là một object miêu tả điều chúng ta muốn làm .Ví dụ: action { type:'ADD_USER', user: user}
- ❑ Reducer sẽ tiếp nhận và xử lý action. Tại reducer, ta sẽ có 1 Root reducer để tổng hợp, và 1 child reducer để xử lý từng thuộc tính, trong trường hợp này là **dssv**
- ❑ Tại child reducer sẽ kiểm tra action, cập nhật lại và trả ra **dssv** mới
- ❑ Tại Component DanhSachSinhVien, ta truy cập lên store để lấy **dssv** về và hiển thị ra màn hình

Ứng dụng redux làm ví dụ sau:

mapStateToProps: là callback lấy giá trị từ redux Store về làm props cho component



mapDispatchToProps: Là callback tạo ra props là những phương thức dùng để đưa giá trị lên store làm thay đổi giá trị giỏ hàng của Store => Tất cả component lấy đang map từ state của giỏ hàng về sẽ tự cập nhật lại giá trị mới.

Cài đặt redux

❖ npm i redux --save

<https://cybersoft.edu.vn>

❖ npm I react-redux --save

<https://cybersoft.edu.vn>

❑ Cấu hình file index.js

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import * as serviceWorker from './serviceWorker';
6
7 //Root reducer là file mình tạo ra nó là store tổng của toàn ứng dụng
8 import {rootReducer} from './redux/reducers/rootReducer';
9 //Provider là component kết nối redux store với component react
10 import {Provider} from 'react-redux';
11 import {createStore} from 'redux';
12 const store = createStore(rootReducer);
13
14 ReactDOM.render(
15   <Provider store = {store}>
16     <App />
17   </Provider>
18   ,
19   document.getElementById('root'));
```

.edu.vn

Cấu hình file gioHangReducer.jsx

GioHangReducer.jsx X

```
demo > src > redux > reducers > GioHangReducer.jsx > ...
1  const stateGioHang = {
2    |   gioHang: [] //Giá trị mặc định ban đầu của giỏ hàng
3  }
4
5  export const GioHangReducer = (state = stateGioHang, action) => {
6    |   return {...state}
7 }
```

s://cybersoft.edu.vn



ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Cấu hình file rootReducer.jsx

```
1  import {combineReducers} from 'redux';
2  import {GioHangReducer} from './GioHangReducer';
3  export const rootReducer = combineReducers({
4    |   //Nơi sẽ chứa các store theo từng nghiệp vụ
5    |   //GioHangReducer:GioHangReducer có thể viết theo cách này
6    |   GioHangReducer //Hoặc viết theo kiểu rút gọn
7  });
```

Thực hiện phương thức mapStateToProps lấy dữ liệu từ store về

GioHangReducer

```

1
2 const stateGioHang = {
3     gioHang:[
4         {maSP:1,hinhAnh:'./img/vsphone.jpg',tenSP:'VinSmart Live',soLuong:1,
5          giaBan:5700000}
6     ] //Giá trị mặc định ban đầu của giỏ hàng
7 }
8 export const GioHangReducer = (state = stateGioHang, action)=>{
9     return {...state}
10 }

```

rootReducer

```

1 import {combineReducers} from 'redux';
2
3 import {GioHangReducer} from './GioHangReducer';
4
5 export const rootReducer = combineReducers({
6     //Nơi sẽ chứa các store theo từng nghiệp vụ
7     //GioHangReducer:GioHangReducer có thể viết theo cách này
8     GioHangReducer //Hoặc viết theo kiểu rút gọn
9 });

```

gioHangComponent. Lấy giá trị từ store về.

```

63 //state chính là rootReducer
64 const mapStateToProps = (state) => {
65     //state.GioHangReducer => rootReducer.GioHangReducer
66     return { gioHang: state.GioHangReducer.gioHang }
67
68 };
69
70 //Sử dụng hàm connect để kết nối Component và Redux Store
71 export default connect(mapStateToProps, null)(GioHangModal)

```

↑ **Hàm kết nối giữa redux store và reactComponent**
Chuyển các state của redux => các props của component

gioHangComponent. Sau khi lấy kết quả lấy từ store về thông qua props ta binding dữ liệu lên giao diện dựa vào hàm map.

```
<div className="modal-body">
  <table className="table">
    <tr>
      <th>Mã sản phẩm</th>
      <th>Hình ảnh</th>
      <th>Tên sản phẩm</th>
      <th>Số lượng</th>
      <th>Đơn giá</th>
      <th>Thành tiền</th>
      <th></th>
    </tr>
    {
      this.props.gioHang.map((spGioHang, index) => {
        return <tr key={index}>
          <td>{spGioHang.maSP}</td>
          <td><img src={spGioHang.hinhAnh} width={50} height={50} /></td>
          <td>{spGioHang.tenSP}</td>
          <td><button className="btn btn-primary">+</button>{spGioHang.soLuong}<button className="btn btn-primary">-</button></td>
          <td>{spGioHang.giaBan}</td>
          <td>{spGioHang.soLuong * spGioHang.giaBan}</td>
          <th><button className="btn btn-danger">Xóa</button></th>
        </tr>
      })
    }
  </table>
</div>
<div className="modal-footer">
  <button type="button" className="btn btn-secondary" data-dismiss="modal">Đóng</button>
</div>
```

```
6 //Lưu ý bỏ export default của component
7 class GioHangModal extends Component {
8   constructor(props) {
9     super(props);
10 }
```

DFT
TRÌNH

edu.vn

Lưu ý : tại component bỏ export default ta chuyển export default xuống dưới hàm connect

```

4  class SanPham extends Component {
5
6    render() {
7      let { sanPham } = this.props;
8      let { xemChiTiet } = this.props;
9      //Hoặc có thể khai báo
10     // let {sanPham,xemChitiet} = this.props;
11     return (
12       <div className="card text-left">
13         <img className="card-img-top" src={sanPham.hinhAnh} width={170} height={300} alt={'true'} />
14         <div className="card-body">
15           <h4 className="card-title">{sanPham.tenSP}</h4>
16           <button className="btn btn-success" onClick={() => xemChiTiet(sanPham)}>Xem chi tiết</button>
17           &nbsp; <button className="btn btn-danger" onClick={() => this.props.themGioHang(sanPham)}>Thêm giỏ hàng</button>
18
19         </div>
20       </div>
21     )
22   }
23 }
24 }
```

Gắn phương thức
themGioHang vừa định nghĩa
vào sự kiện của nút button
thêm giỏ hàng



<https://cybersoft.edu.vn>

```

26 const mapDispatchToProps = (dispatch) => {
27   return {
28     themGioHang: (sanPham) => {
29       let sanPhamGioHang = {...sanPham, soLuong:1}; //Sản phẩm giỏ hàng cần thêm
30       dispatch({
31         type: 'THEM_GIO_HANG',
32         sanPhamGioHang
33       });
34     }
35   }
36 }
37
38 export default connect(null,mapDispatchToProps)(SanPham)
```



Tương tự props là dữ liệu lấy
về từ store về thì ở đây hàm
dispatch cho ta định nghĩa 1
props là dữ 1 phương thức
đưa dữ liệu lên store



Thực hiện phương thức mapDispatchToProps tạo phương thức đưa dữ liệu lên Redux store

Tại **gioHangReducer** ta nhận được dữ liệu từ nút thêm giỏ hàng sau phương thức dispatch đưa lên store qua biến **action**. Ta dùng thuộc tính type để phân biệt xử lý và dùng thuộc tính giá trị(ở đây là thuộc tính **sanPhamGioHang**) để thay đổi state của Reducer gioHang (state.gioHang)

```
2  const stateGioHang = {  
3      gioHang: [  
4          { maSP: 1, hinhAnh: './img/vsphone.jpg', tenSP: 'VinSmart Live', soLuong: 1, giaBan: 5700000 }  
5      ] //Giá trị mặc định ban đầu của giỏ hàng  
6  }  
7  
8 export const GioHangReducer = (state = stateGioHang, action) => {  
9  
10    switch (action.type) {  
11        case 'THEM GIO HANG': {  
12            let gioHang = [...state.gioHang];  
13            const index = gioHang.findIndex(spGH => spGH.maSP === action.sanPhamGioHang.maSP);  
14            if (index === -1) {  
15                //Kiểm tra nếu sản phẩm chưa có trong giỏ hàng thì thêm sản phẩm đó vào giỏ hàng  
16                gioHang.push(action.sanPhamGioHang)  
17            } else {  
18                //Ngược lại nếu đã có trong giỏ hàng rồi thì tăng số lượng  
19                gioHang[index].soLuong += 1;  
20            }  
21            //Cập nhật lại state  
22            state.gioHang = gioHang;  
23            return { ...state }  
24        }  
25    }  
26  
27    return { ...state }  
28 }
```

action gửi lên 2 thuộc tính là type, và dữ liệu => dựa vào type để xác định xử lý tương đương cho action.
type: là thuộc tính bắt buộc

dữ liệu gửi lên từ dispatch

Tại **gioHangReducer** ta nhận được dữ liệu từ nút thêm giỏ hàng sau phương thức dispatch đưa lên store qua biến **action**. Ta dùng thuộc tính **type** để phân biệt xử lý và dùng thuộc tính **giá trị**(ở đây là thuộc tính **sanPhamGioHang**) để thay đổi state của Reducer gioHang (**state.gioHang**). Những nới **mapStateToProps** từ gioHangReducer khi thay đổi sẽ tự động cập nhật lại.

```
2  const stateGioHang = {
3      gioHang: [
4          { maSP: 1, hinhAnh: './img/vsphone.jpg', tenSP: 'VinSmart Live', soLuong: 1, giaBan: 5700000 }
5      ] //Giá trị mặc định ban đầu của giỏ hàng
6  }
7
8  export const GioHangReducer = (state = stateGioHang, action) => {
9
10     switch (action.type) {
11         case 'THEM GIO HANG': {
12             let gioHang = [...state.gioHang];
13             const index = gioHang.findIndex(spGH => spGH.maSP === action.sanPhamGioHang.maSP);
14             if (index === -1) {
15                 //Kiểm tra nếu sản phẩm chưa có trong giỏ hàng thì thêm sản phẩm đó vào giỏ hàng
16                 gioHang.push(action.sanPhamGioHang)
17             } else {
18                 //Ngược lại nếu đã có trong giỏ hàng rồi thì tăng số lượng
19                 gioHang[index].soLuong += 1;
20             }
21             //Cập nhật lại state
22             state.gioHang = gioHang;
23             return { ...state }
24         }
25     }
26
27     return { ...state }
28 }
```

action gửi lên 2 thuộc tính là type, và dữ liệu => dựa vào type để xác định xử lý tương đương cho action.
type: là thuộc tính bắt buộc

dữ liệu gửi lên từ dispatch

CYBERSOFT

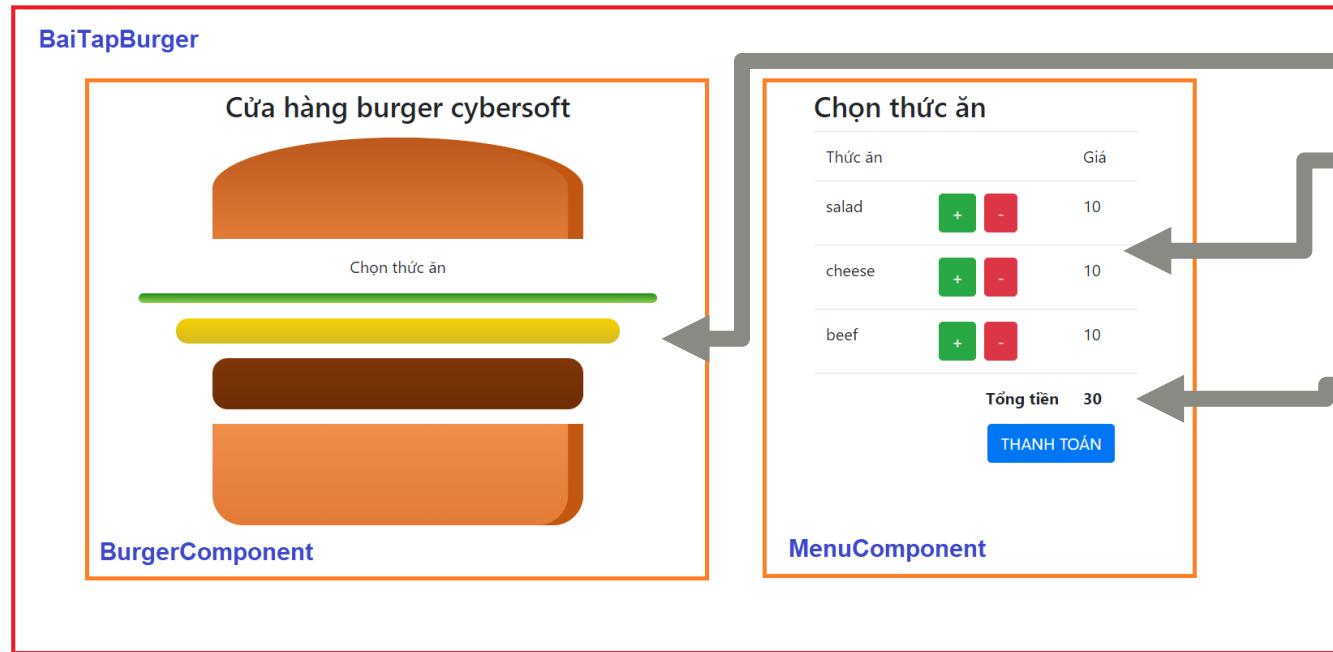
AP TRÌNH

ersoft.edu.vn

Bài tập: Thực hiện bài tập sau

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>



Burger Reducer

```
2 ↴ const burgerState = {  
3   burger: { salad: 1, cheese: 1, beef: 1 },  
4  
5 ↴   menu:  
6   {  
7     salad: 10,  
8     cheese: 10,  
9     beef: 10  
10    },  
11  
12   total: 30  
13  
14  
15 ↴ export const BurgerReducer = (state = burgerState, action) => {  
16   |   return {...state}  
17 }
```

Xây dựng chức năng khi người dùng click vào nút cộng trừ thì burger bên phải sẽ hiển thị như tương ứng với các thức ăn đã chọn bên phải tăng hoặc giảm.

Xây dựng component như hình trên

Sử dụng redux lưu trữ các dữ liệu (state) thay đổi trên giao diện

React Form - validation

<https://cybersoft.edu.vn><https://cybersoft.edu.vn>

- Khác với javascript ta thường sử dụng để lấy giá trị thông qua 1 tag `<input>` ta phải thực hiện dom thông qua các **selector** sau đó truy xuất đến thuộc tính **value** để lấy giá trị ... mệt khá nhiều thao tác. Tuy nhiên:
- Trong react chúng ta sẽ sử dụng thuộc tính **state** của component kết hợp cùng các sự kiện của control input để thực hiện việc lấy dữ liệu từ form dễ dàng hơn qua ví dụ sau:

Thông tin sinh viên

Mã SV

Họ tên

Số điện thoại

Email

Thêm sinh viên

Mã SV

Họ tên

Số điện thoại

Email

1

Nguyễn Văn A

0938111111

nguyenvana@gmail.com

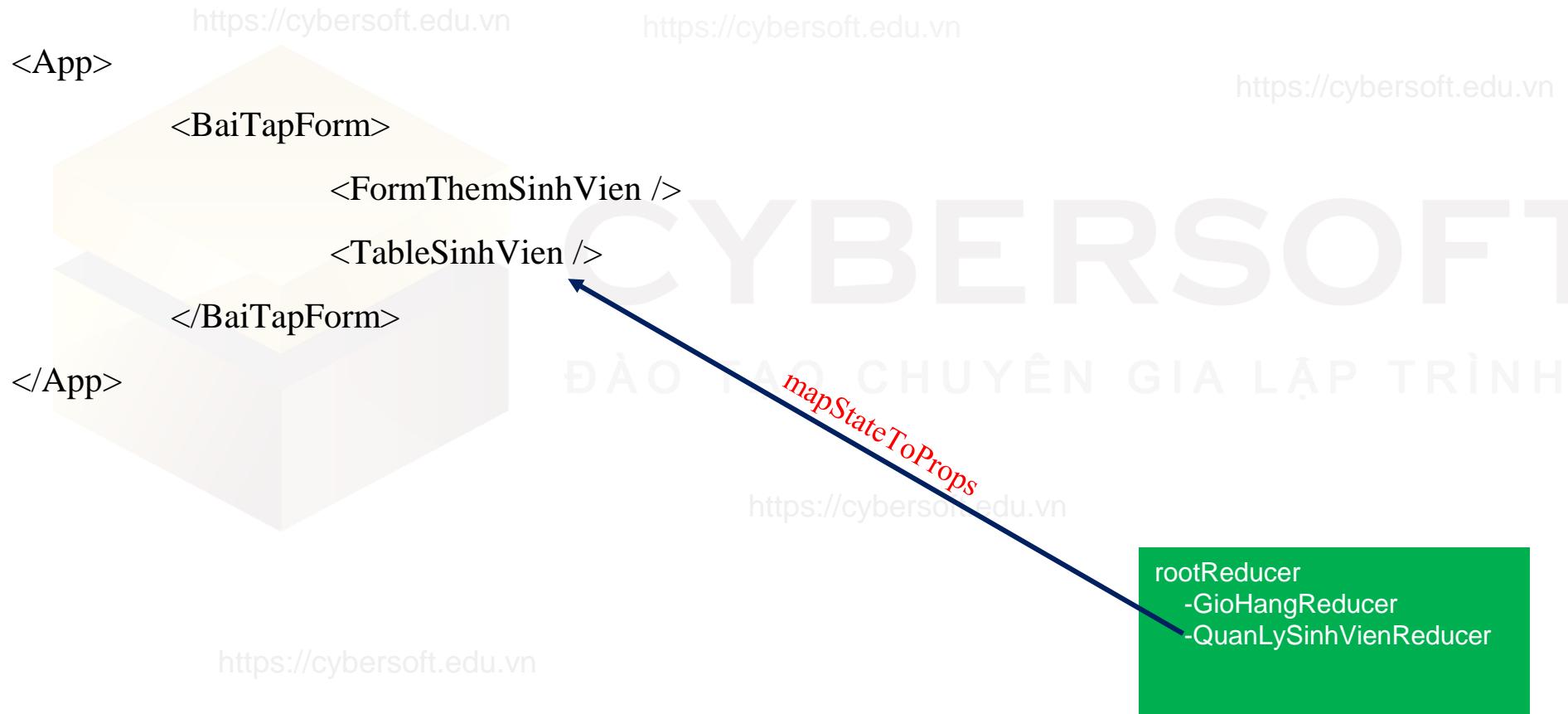
2

Nguyễn Văn B

093822232232

nguyenvanb@gmail.com

Tạo component theo cấu trúc



Cấu trúc reducer cho bài học

Cấu trúc reducer cho nghiệp vụ quản lý sinh viên

```
3 const stateDefault = {  
4   mangSinhVien: []  
5 }  
6  
7 export const QuanLySinhVienReducer = (state = stateDefault, action) => {  
8  
9   switch (action.type) {  
10  
11     case 'THEM_SINH_VIEN':  
12     {  
13       state.mangSinhVien = [...state.mangSinhVien,action.sinhVien];  
14       return {...state}  
15     }  
16     default:  
17       return {...state}  
18   }  
19 }
```

https://cybersoft.edu.vn

https://cybersoft.edu.vn

YIASS CYBERSOFT
HUYÊN GIA LẬP TRÌNH

Khai báo QuanLySinhVienReducer trong rootReducer

```
1 import { combineReducers } from 'redux';  
2  
3 import { GioHangReducer } from './GioHangReducer';  
4  
5 import { QuanLySinhVienReducer } from './QuanLySinhVienReducer';  
6  
7 export const rootReducer = combineReducers({  
8   //Nơi sẽ chứa các store theo từng nghiệp vụ  
9   //GioHangReducer:GioHangReducer có thể viết theo cách này  
10  GioHangReducer, //Hoặc viết theo kiểu rút gọn  
11  
12  QuanLySinhVienReducer  
13  
14});
```

https://cybersoft.edu.vn

Cấu trúc component

Tại component <TableSinhVien> kết nối dữ liệu với redux lấy mangSinhVien về binding ra giao diện

```
3  class TableSinhVien extends Component {  
4      render = () => (  
5          <div className="container">  
6              <table className="table table-hover">  
7                  <thead className="thead-dark ">  
8                      <tr>  
9                          <th>Mã SV</th>  
10                         <th>Họ tên</th>  
11                         <th>Số điện thoại</th>  
12                         <th>Email</th>  
13                     </tr>  
14                 </thead>  
15                 <tbody>  
16                     {this.props.mangSinhVien.map((sv, index) => {  
17                         return (  
18                             <tr key={index}>  
19                                 <td>{sv.masV}</td>  
20                                 <td>{sv.hoTen}</td>  
21                                 <td>{sv.soDienThoai}</td>  
22                                 <td>{sv.email}</td>  
23                             </tr>  
24                         )  
25                     })}  
26                 </tbody>  
27             </table>  
28         </div>  
29     )  
30 }  
  
//Kết nối với reducer redux lấy dữ liệu mảng sinh viên về => this.props.mangSinhVien  
35 const mapStateToProps = state => {  
36     return {  
37         mangSinhVien: state.QuanLySinhVienReducer.mangSinhVien  
38     }  
39 }  
40 }  
41  
42 export default connect(mapStateToProps, null)(TableSinhVien);
```

<https://cybersoft.edu.vn>

CYBERSOFT
LẬP TRÌNH

cybersoft.edu.vn

```

render() {
  return (
    <form className="container" onSubmit={this.handleSubmit}>
      <h3 className="text-left bg-dark text-white p-3">Thông tin sinh viên</h3>
      <div className="form-group row">
        <div className="col-6">
          <span>Mã SV</span>
          <input name="maSV" className="form-control" value={this.state.maSV} onChange={this.handleChange} />
        </div>
        <div className="col-6">
          <span>Họ tên:</span>
          <input name="hoTen" className="form-control" value={this.state.hoTen} onChange={this.handleChange} />
        </div>
      </div>
      <div className="form-group row">
        <div className="col-6">
          <span>Số điện thoại:</span>
          <input name="soDienThoai" className="form-control" value={this.state.soDienThoai} onChange={this.handleChange} />
        </div>
        <div className="col-6">
          <span>Email:</span>
          <input name="email" className="form-control" value={this.state.email} onChange={this.handleChange} />
        </div>
      </div>
      <div className="form-group">
        <button className="btn btn-success" type="submit">Thêm sinh viên</button>
      </div>
    </form>
  )
}

```

Đặt tên các thuộc tính **name** của **input** đặt **trùng tên với state** để mượn tính năng object literal (tính năng của es6 cho phép ta động về mặt thuộc tính) ta có thể **setState** ứng với các dữ liệu người dùng nhập vào tự động set giá trị cho state.

RSOFT
NGÀNH GIÁO DỤC

```

4  class FormSinhVien extends Component {
5
6    constructor(props) {
7      super(props);
8      //Đặt tên thuộc tính là tên name các control input
9      this.state = {
10        maSV: '',
11        hoTen: '',
12        soDienThoai: '',
13        email: ''
14      }
15    }

```

```

17    handleSubmit = (e) => {
18      e.preventDefault();
19      //Gọi this.props.themSinhVien() từ mapDispatchToProps redux
20      this.props.themSinhVien(this.state);
21
22    }
23
24
25    //sự kiện xử lý mỗi lần người dùng nhập dữ liệu (onchange)
26    //thì sẽ dựa vào name để lấy value của thẻ input đó
27    handleChange = (e) => {
28      const { name, value } = e.target;
29      //set state tương ứng với name của reducer (tính năng object literal động về thuộc tính)
30      this.setState({ [name]: value });
31    }

```

Validation input trong react

Trong javascript thuần để kiểm tra ngoại lệ của 1 control input ta cũng phải thực hiện dom đến control đó qua các selector sau đó so sánh các giá trị ... sau đó lại dom đến các thẻ thông báo lỗi để gán các giá trị đó bên dưới. Tuy nhiên react với state giúp ta tiết kiệm được nhiều thời gian cho việc này hơn. Cú pháp ngắn gọn hơn rất nhiều. Thông qua ví dụ sau:

<https://cybersoft.edu.vn>

Tại **<FormSinhVien>** thay vì ta các các thuộc tính state là các control name của input ta thay đổi lại như sau:

```
4  class FormSinhVien extends Component {  
5  
6      constructor(props) {  
7          super(props);  
8          //Đặt tên thuộc tính values là tên name các control input  
9          //Thuộc tính errors là tên các lỗi ứng với các thuộc tính values  
10         this.state = {  
11             values: {  
12                 maSV: '',  
13                 hoTen: '',  
14                 soDienThoai: '',  
15                 email: ''  
16             },  
17             errors: {  
18                 maSV: '',  
19                 hoTen: '',  
20                 soDienThoai: '',  
21                 email: ''  
22             }  
23         }  
24     }
```

SOFT
LẬP TRÌNH

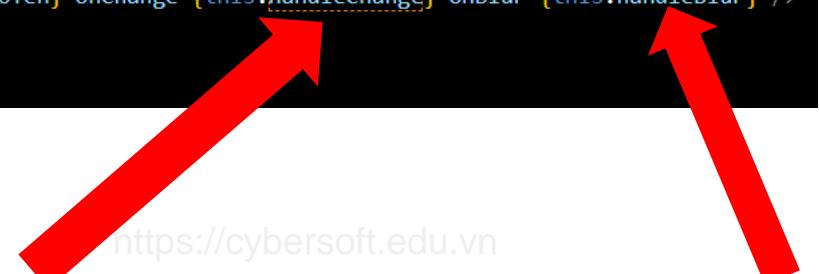
[/cybersoft.edu.vn](https://cybersoft.edu.vn)

Vì vậy tại các control <input> thay vì ta set **value bằng this.state.thuocTinh** thì ta set **value = this.state.values.thuocTinh**
 Và ta sẽ cho thêm 1 div phía dưới **set errors** từ **this.state.errors.thuocTinh**

```

92 renderErrorMessage = (errorMessage) => {
93   if (errorMessage !== '') {
94     return <div className="alert alert-danger">{errorMessage}</div>;
95   }
96   return '';
97 }
98
99 render() {
100   return (
101     <form className="container" onSubmit={this.handleSubmit}>
102       <h3 className="text-left bg-dark text-white p-3">Thông tin sinh viên</h3>
103       <div className="form-group row">
104         <div className="col-6">
105           <span>Mã SV</span>
106           <input name="maSV" className="form-control" value={this.state.values.maSV} onChange={this.handleChange} onBlur={this.handleBlur} />
107           {this.renderErrorMessage(this.state.errors.maSV)}
108         </div>
109         <div className="col-6">
110           <span>Họ tên</span>
111           <input name="hoTen" className="form-control" value={this.state.values.hoTen} onChange={this.handleChange} onBlur={this.handleBlur} />
112           {this.renderErrorMessage(this.state.errors.hoTen)}
113         </div>
114       </div>

```



<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Sự kiện cập nhật values

Sự kiện cập nhật errors

```

//sự kiện xử lý mỗi lần người dùng nhập dữ liệu (onchange)
//thì sẽ dựa vào name để lấy value của thẻ input đó
handleChange = (e) => {
  const { name, value } = e.target;
  //set state tương ứng với name của reducer (tính năng object literal động về thuộc tính)
  this.setState({ values: { ...this.state.values, [name]: value } });
}

```

```

handleBlur = (e) => {
  const { name, value } = e.target;
  //Hàm này check validation được định nghĩa bên dưới
  const errorMessage = this.validateInput(name, value);
  this.setState({ errors: { ...this.state.errors, [name]: errorMessage } });
}

```

```
48 //Hàm kiểm tra validation
49 validateInput = (name, value) => {
50     let errorMessage = '';
51     if (name === 'maSV') {
52         if (!value) {
53             errorMessage = 'Mã sinh viên không được bỏ trống !';
54         }
55     }
56
57     if (name === 'email') {
58         if (!value) {
59             errorMessage = 'Email không được bỏ trống !';
60         } else if (!/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(value)) {
61             errorMessage = 'Email không hợp lệ !';
62         }
63     }
64
65     if (name === 'hoTen') {
66         if (!value) {
67             errorMessage = 'Họ tên không được bỏ trống !';
68         }
69     }
70
71     if (name === 'soDienThoai') {
72         if (!value) {
73             errorMessage = 'Số điện thoại không được bỏ trống !';
74         }
75     }
76
77     return errorMessage;
78 }
```

<https://cybersoft.edu.vn>

SOFT
HỌA LẬP TRÌNH

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

CYBERSOFT

Vòng đời của component

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

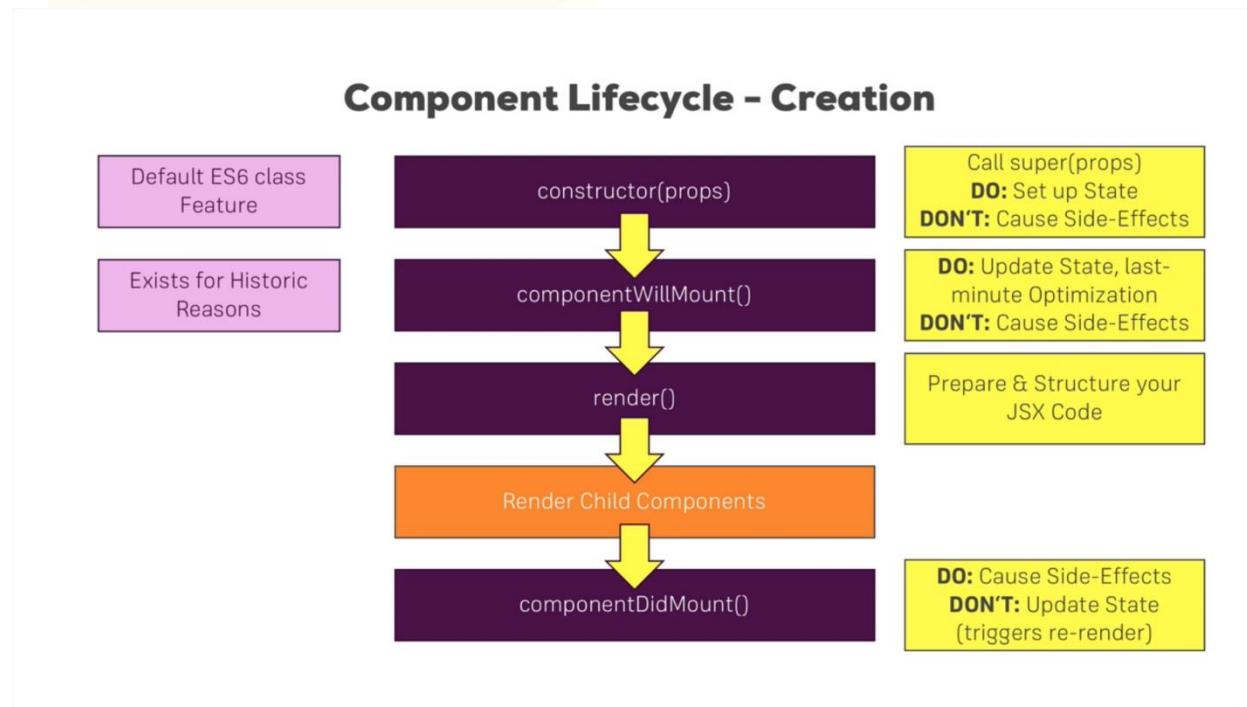
<https://cybersoft.edu.vn>

Vòng đời component - Creation

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❑ Vòng đời Creation bao gồm các phương thức sẽ được khởi chạy lần lượt khi component được khởi tạo và chỉ sử dụng được với statefull



- ❑ Trong đó, componentWillMount sắp bị loại bỏ ở phiên bản reactjs 17 nên ta sẽ ko dùng tới
- ❑ componentDidMount là nơi để ta thực hiện các hành động khi load component lên , ví dụ như gọi API load dữ liệu...

<https://cybersoft.edu.vn>

Vòng đời component - Creation

<https://cybersoft.edu.vn>

```
import React, { Component } from 'react';

class demo extends Component {
  constructor(props){
    super(props);
    console.log('Constructor')
  }
  componentWillMount(){
    console.log('componentWillMount')
  }
  render() {
    console.log('render');
    return (
      <div>
        <h1>DEMO CREATION LIFECYCLE</h1>
      </div>
    );
  }
  componentDidMount(){
    console.log('componentDidMount');
  }
}

export default demo;
```

<https://cybersoft.edu.vn>

Constructor

componentWillMount

render

componentDidMount

 ./src/App.js

Line 1: 'Fragment' is defined but never used
 Line 2: 'DanhSachSinhVien' is defined but never used
 Line 3: 'Test' is defined but never used

<https://cybersoft.edu.vn>
[demo.js:6](#)
[demo.js:9](#)
[demo.js:12](#)
[demo.js:20](#)
[webpackHotDevClient.js:120](#)

no-unused-vars

no-unused-vars

no-unused-vars

<https://cybersoft.edu.vn>

=> Thứ tự console.log đã chứng minh thứ tự chạy của các lifecycle creation

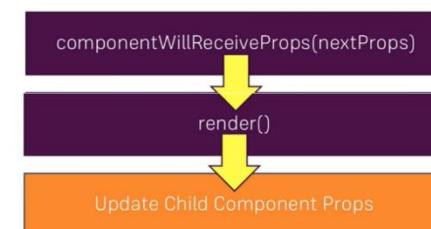
Vòng đời component - Update

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❑ Khi giá trị props bị thay đổi, sẽ dẫn tới việc component được update, các hàm bên dưới sẽ chạy lượt khi component update

Component Lifecycle – Update (triggered by Parent)



YÊN GIA LẬP TRÌNH

edu.vn

<https://cybersoft.edu.vn>

- ❑ Vậy thì khi nào sự thay đổi của component thực sự xảy ra ?

Vòng đời component - Update

<https://cybersoft.edu.vn>
<https://cybersoft.edu.vn>

- ❑ Ví dụ, ta tạo 2 component DemoParent và DemoChild như sau:

```
import React, { Component } from 'react';
import DemoChild from './demo-child';
class DemoParent extends Component {
  state = {
    name: 'Hiếu'
  }
  onChangeName = () => {
    this.setState({
      name: 'Dũng'
    })
  }
  render() {
    return (
      <div>
        <button onClick={this.onChangeName}>Change Name</button>
        <DemoChild name={this.state.name} />
      </div>
    );
  }
}

export default DemoParent;
```

```
import React, { Component } from 'react';

class DemoChildComponent extends Component {
  render() {
    return (
      <div>
        <p>Tên: {this.props.name}</p>
      </div>
    );
  }
}

export default DemoChildComponent;
```

- ❑ Ở component DemoParent, ta truyền props **this.state.name** vào trong DemoChild component
- ❑ Khi nhấn nút, ta thay đổi **this.state.name**, nghĩa là props truyền vào DemoChild component bị thay đổi => DemoChild component sẽ được update => các lifecycle update sẽ chạy

Vòng đời component - Update

<https://cybersoft.edu.vn>
<https://cybersoft.edu.vn>

- ❑ Ví dụ, ta tạo 2 component DemoParent và DemoChild như sau:

```
import React, { Component } from 'react';
import DemoChild from './demo-child';
class DemoParent extends Component {
  state = {
    name: 'Hiếu'
  }
  onChangeName = () => {
    this.setState({
      name: 'Dũng'
    })
  }
  render() {
    return (
      <div>
        <button onClick={this.onChangeName}>Change Name</button>
        <DemoChild name={this.state.name} />
      </div>
    );
  }
}

export default DemoParent;
```

```
import React, { Component } from 'react';

class DemoChildComponent extends Component {
  render() {
    return (
      <div>
        <p>Tên: {this.props.name}</p>
      </div>
    );
  }
}

export default DemoChildComponent;
```

- ❑ Ở component DemoParent, ta truyền props **this.state.name** vào trong DemoChild component
- ❑ Khi nhấn nút, ta thay đổi **this.state.name**, nghĩa là props truyền vào DemoChild component bị thay đổi => DemoChild component sẽ được update => các lifecycle update sẽ chạy

Vòng đời component - Update

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ☐ Tại component DemoChild :

```
import React, { Component } from 'react';

class DemoChildComponent extends Component {
  componentWillMountReceiveProps(nextProps){
    console.log('componentWillReceiveProps',nextProps);
  }
  render() {
    return (
      <div>
        <p>Tên: {this.props.name}</p>
      </div>
    );
  }
  componentDidUpdate(){
    console.log('componentDidUpdate')
  }
}

export default DemoChildComponent;
```

componentWillReceiveProps ► {name: "Dũng"}
componentDidUpdate

nextProps được in ra chính là props mới sau khi được thay đổi

C

cybersoft.edu.vn

<https://cybersoft.edu.vn>

PureComponent

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❑ Trong một vài trường hợp, dù props của component không hề thay đổi nhưng vẫn bị update, dẫn tới ảnh hưởng performance của app
- ❑ Ví dụ :
- ❑ Ở đây ta có thể thấy, khi click vào nút change, state sẽ được set lại, dẫn tới hàm render() sẽ chạy lại và các component con bên trong app cũng được render lại.
- ❑ Vấn đề ở đây là state chỉ ảnh hưởng tới component DanhSachSinhVien, còn component Test vốn ko hề thay đổi, do đó việc render lại nó là ko cần thiết

```
File Edit Selection View Go Debug Terminal Help
demo > src > App.js ...
1 import React, { Component, Fragment } from 'react';
2 import { DanhSachSinhVien } from './container/danh sach sinh vien';
3 import Test from './container/Home/test';
4 class App extends Component {
5   state = {
6     name: 'hieu'
7   }
8   changeName = () => {
9     this.setState({
10       name: 'Dung'
11     })
12 }
13 render() {
14   console.log('a');
15   return (
16     <div>
17       <button onClick={this.changeName}>Change</button>
18       <DanhSachSinhVien name={this.state.name} />
19       <Test />
20     </div>
21   );
22 }
23
24
25
26
27
28
export default App;
```

PureComponent

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❑ Để xử lý vấn đề này, ta có cách giải quyết sau
- ❑ Ở component Test, thay vì class Test extends Component, ta sẽ cho nó extends từ PureComponent, lúc này component Test chỉ render lại khi mà props của nó thật sự thay đổi
- ❑ Lưu ý: chỉ sử dụng PureComponent, ko nên lạm dụng vì có thể dẫn tới lỗi . Bản chất của PureComponent là tự động kiểm tra xem nếu props và state của component đó thay đổi thì sẽ render lại, không thì thôi. Nhưng sự so sánh thay đổi của react là so sánh tham chiếu , nếu như ta truyền một object dưới dạng props, và thay đổi một thuộc tính nào đó thì react ko so sánh đc, vì căn bản là cùng 1 object

```
src > container > Home > test.js > ...
import React, { PureComponent } from 'react'

export default class test extends PureComponent {

  render() {

    return (
      <div>
        | aaa
      </div>
    )
  }
}
```

<https://cybersoft.edu.vn>

Bài tập: Áp dụng lifecycle vào bài tập quản lý sinh viên để tăng hiệu suất cho ứng dụng web



CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

HTTP và kết nối API

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

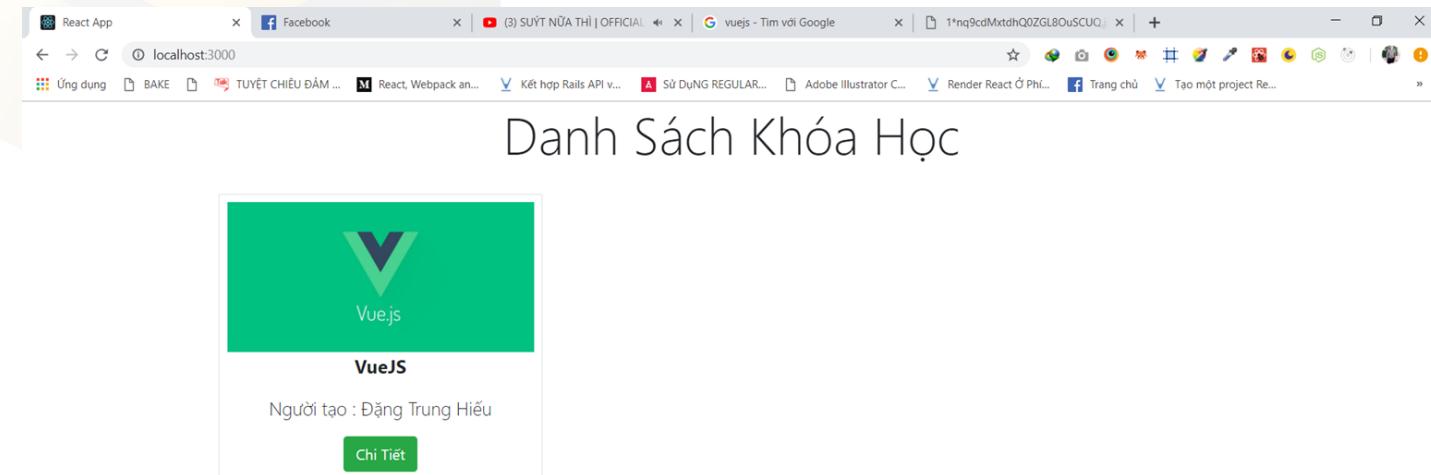
<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Thư viện hỗ trợ : Axios

- ❖ Axios là thư viện hỗ trợ ta gọi API lấy dữ liệu
- ❖ Cách sử dụng tương tự như ajax của jquery
- ❖ Bài tập thực hành : Lấy danh sách khóa học thông qua API



<https://cybersoft.edu.vn>

Bài tập : lấy danh sách khóa học

- ❖ Đầu tiên, tạo ra 2 component : DanhSachKhoaHoc và KhoaHoc
- ❖ Tiếp theo , tại app.js, gọi component DanhSachKhoaHoc để hiện thị ra màn hình

DanhSachKhoaHoc

The screenshot shows a code editor with the following details:

- EXPLORER:** Shows files in the project structure:
 - OPEN EDITORS:** dssv.jsx, sinhvien.jsx, actionTypes.js, sinhvien.js, dskhoahoc.jsx (highlighted with a yellow box), khoahoc.jsx.
 - SHOPPING-CART:** components, cart, CRUD (highlighted with a yellow box),
 - dskhoahoc.jsx (highlighted with a yellow box)
 - dssv.jsx
 - form-nhap.jsx
 - khoaahoc.jsx
 - sinhvien.jsx
 - products
 - redux
 - actions
 - constants
 - actionType.js
- CODE EDITOR:** The code for the `DanhSachKhoaHoc` component is displayed:

```
import React, { Component } from 'react';
import KhoaHoc from './khoaahoc';

class DanhSachKhoaHoc extends Component {
  render() {
    return (
      <div className="container">
        <div className="row">
          <div className="w-100 text-center mb-4">
            <h1 className="display-4">Danh Sách Khóa Học</h1>
            <KhoaHoc />
          </div>
        </div>
      </div>
    );
  }
}

export default DanhSachKhoaHoc;
```

KhoaHoc

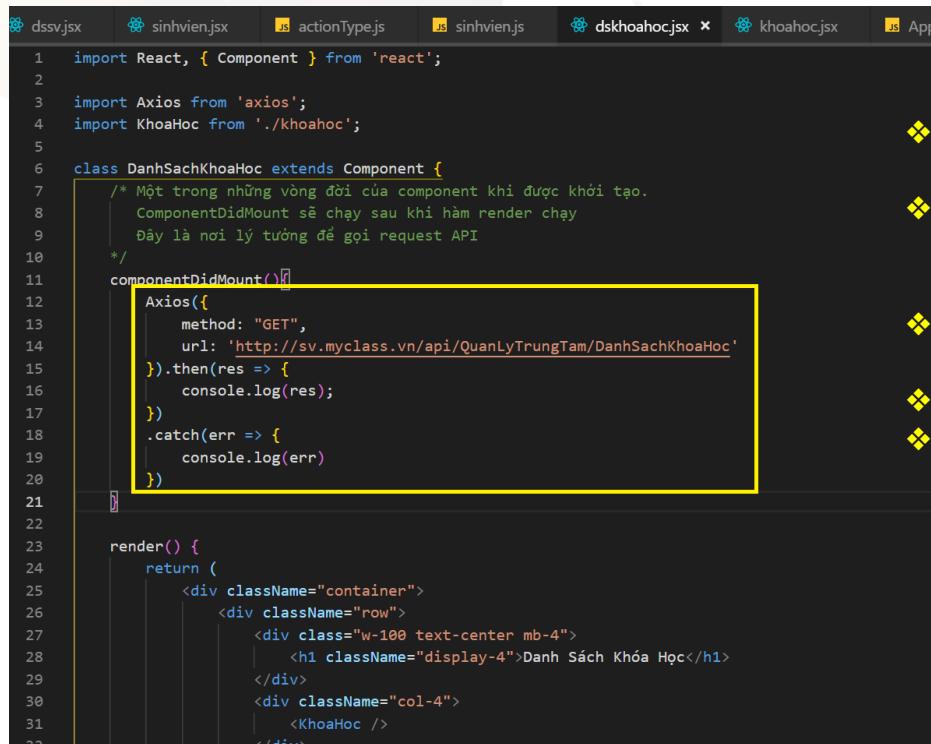
```
import React from 'react';

const khoahoc = () => {
  return (
    <div className="border p-2 text-center">
      
      <p className="lead font-weight-bold">VueJS</p>
      <p className="lead">Người tạo : Đặng Trung Hiếu</p>
      <button className="btn btn-success">Chi Tiết</button>
    </div>
  );
};

export default khoahoc;
```

Bài tập : lấy danh sách khóa học

- ❖ Tiếp theo, tại component DanhSachKhoaHoc, ta sử dụng **axios** gọi lên api lấy danh sách khóa học về và hiển thị ra màn hình
 - ❖ Bước 1 : cài đặt axios **npm install axios –save**
 - ❖ Bước 2 : tại component DanhSachKhoaHoc, gọi và sử dụng axios



```

1 import React, { Component } from 'react';
2
3 import Axios from 'axios';
4 import KhoaHoc from './khoahoc';
5
6 class DanhSachKhoaHoc extends Component {
7   /* Một trong những vòng đời của component khi được khởi tạo.
8    | ComponentDidMount sẽ chạy sau khi hàm render chạy
9    | Đây là nơi lý tưởng để gọi request API
10   */
11   componentDidMount() {
12     Axios({
13       method: "GET",
14       url: 'http://sv.myclass.vn/api/QuanLyTrungTam/DanhSachKhoaHoc'
15     }).then(res => {
16       console.log(res);
17     })
18     .catch(err => {
19       console.log(err)
20     })
21   }
22
23   render() {
24     return (
25       <div className="container">
26         <div className="row">
27           <div className="w-100 text-center mb-4">
28             <h1 className="display-4">Danh Sách Khóa Học</h1>
29           </div>
30           <div className="col-4">
31             <KhoaHoc />
32           </div>
33         </div>
34       </div>
35     );
36   }
37 }
38
39 export default DanhSachKhoaHoc;

```

- ❖ Đầu tiên, ta import phương thức Axios từ package axios đã install
- ❖ Tiếp theo, ta gọi hàm Axios chạy và trả ra một đối tượng promise (hỗ trợ tác vụ bất đồng bộ). Một đối tượng promise sẽ có 2 trạng thái chính ; Thành công hoặc thất bại
- ❖ Nếu thành công thì nhảy vào then với res là kết quả server trả về
- ❖ Nếu thất bại thì nhảy vào catch, với err là lỗi bắt được
- ❖ Kiểm tra cửa sổ console và thấy kết quả. Nếu thành công , server trả về object Response, để lấy cái ta cần là danh sách khóa học,

Bài tập : lấy danh sách khóa học

- ❖ Tuy nhiên, componentDidMount chạy sau render, cho nên dù ta lấy được kết quả từ api, giao diện vẫn không load lại được giao diện
 - ⇒ phải sử dụng state để lưu danh sách khóa học, khi lấy được danh sách về, ta setState lại thì giao diện sẽ render lại
 - ⇒ V state nên lưu ở đâu, nên để tại component hay lưu trên store ?
 - ⇒ Tùy vào yêu cầu sử dụng, nếu ta chỉ sử dụng danh sách lấy được ở component DanhSachKhoaHoc thôi, hay sẽ dùng ở một số component khác nữa
 - ⇒ Ở đây ta sẽ lưu trên store, để có sử dụng ở nhiều component

Bài tập : lấy danh sách khóa học

- ❖ Lưu danh sách khóa học lấy được trên store,
 - ❖ Bước 1: tạo reducer quản lý danh sách khóa học
 - ❖ Bước 2: tại reducers/index.js , ta thêm 1 thuộc tính vào state store đang lưu trữ

The image shows two screenshots of a code editor. The left screenshot displays the 'khoahoc.js' file, which defines a reducer for managing course lists. The right screenshot shows the 'index.js' file, which is the root reducer combining multiple reducers, including one for courses and one for students.

```
let DSKH = [];

const khoaHocReducer = (state = DSKH, action) => {
  switch(action.type){
    default: return [...state];
  }
}

export default khoaHocReducer;
```

```
import { combineReducers } from 'redux';
import SinhVienReducer from './sinhvien';
import KhoaHocReducer from './khoahoc';

const rootReducer = combineReducers({
  DSSV: SinhVienReducer,
  DSKH : KhoaHocReducer
});

export default rootReducer;
```

Bài tập : lấy danh sách khóa học

- ❖ Bước 4: tại component DanhSachKhoaHoc, ta cần dispatch action lên để lưu danh sách, đồng thời cần lấy danh sách đó xuống để sử dụng , do đó cần cả mapStateToProps lẫn mapDispatchToProps

```
2
3 const mapStateToProps = (state) => {
4   return {
5     DSKH : state.DSKH
6   }
7 }
8
9 const mapDispatchToProps = (dispatch) => {
10   return {
11     onSaveDSKH : (danhSachKhoaHoc) =>{
12       dispatch(actGetCourseList(danhSachKhoaHoc))
13     }
14   }
15 }
16 export default connect(mapStateToProps, mapDispatchToProps)(DanhSachKhoaHoc);
```

Bài tập : lấy danh sách khóa học

- ❖ Bước 5: Sau khi lấy được danh sách khóa học từ api về, ta tiến hành dispatch action gửi danh sách đó lên store để lưu trữ

```
componentDidMount(){
  Axios({
    method: "GET",
    url: 'http://sv.myclass.vn/api/QuanLyTrungTam/DanhSachKhoaHoc'
  }).then(res => {
    this.props.onSaveDSKH(res.data);
  })
  .catch(err => {
    console.log(err)
  })
}
```

<https://cybersoft.edu.vn>

SOFT
IA LẬP TRÌNH

<https://cybersoft.edu.vn>

Bài tập : lấy danh sách khóa học

- ❖ Bước 6: Tiến hành lập component KhoaHoc theo danh sách lấy được

```
renderCourse = () => {
  return this.props.DSKH.map((khoaHoc, index) => {
    return (
      <div className="col-4" key={index}>
        <KhoaHoc khoaHoc={khoaHoc} />
      </div>
    )
  })
}

render() {
  return (
    <div className="container">
      <div className="row">
        <div className="w-100 text-center mb-4">
          <h1 className="display-4">Danh Sách Khóa Học</h1>
        </div>
        <div>
          {this.renderCourse()}
        </div>
      </div>
    );
}
```

Bài tập : lấy danh sách khóa học

- ❖ Tại component Danh Sách Khóa Học, khi lấy danh sách từ trên store về, ta sẽ tiến hành dispatch 1 action để đem danh sách đó lưu trên store

- ❖ Bước 1: tạo 1 biến const mới trong constants/actionType.js

```
1  export const DELETE_SV = "DELETE_SV";  
2  
3  
4  export const GET_COURSE_LIST = "GET_COURSE_LIST";
```

- ❖ Bước 2: tạo 1 action creator để tạo ra action sẽ được dispatch trong file actions/khoaHoc.js

```
import {GET_COURSE_LIST} from '../constants/actionType'  
  
export const actGetCourseList = (danhSachKhoaHoc) => {  
  return {  
    type: GET_COURSE_LIST,  
    // danh sách khóa học gửi lên để lưu trên store  
    danhSachKhoaHoc  
  }  
}
```

https://cybersoft.edu.vn

- ❖ Bước 3: tiến hành connect component DanhSachKhoaHoc với store

Nâng cấp: tạo async action với middleware

- ❖ Bước 7: Xây dựng trong reducers/khoaHoc.js để handle action

<https://cybersoft.edu.vn>

```
import { GET_COURSE_LIST } from "../constants/actionType";

let DSKH = [];

const khoaHocReducer = (state = DSKH, action) => {
  switch(action.type){
    case GET_COURSE_LIST :
      var updateState = [...action.danhSachKhoaHoc];
      return updateState;
    default: return [...state];
  }
}
export default khoaHocReducer;
```

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Bài tập : lấy danh sách khóa học

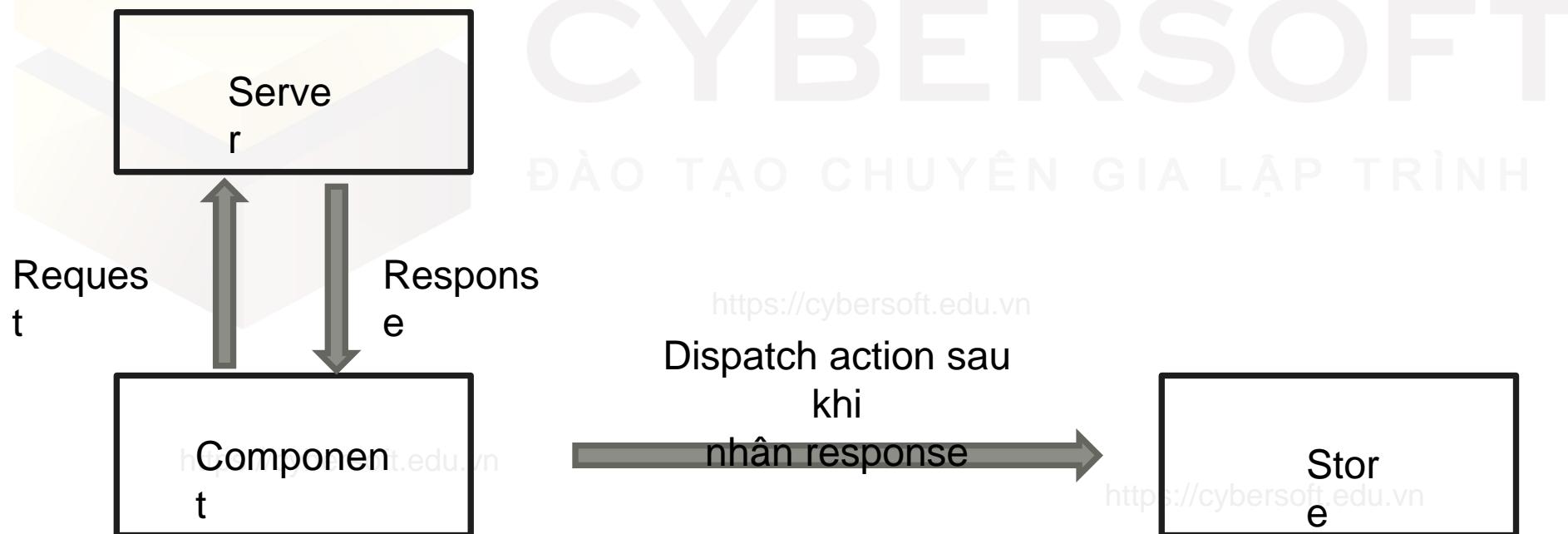
- ❖ Bước 8: Thay đổi trong component KhoaHoc để hiển thị thông tin khóa học tương ứng

<https://cybersoft.edu.vn>

```
1 import React from 'react';
2
3 const khoahoc = (props) => {
4     return (
5         <div class="border p-2 text-center">
6             <img src={props.khoaHoc.HinhAnh} className="w-100" />
7             <p className="lead font-weight-bold">{props.khoaHoc.TenKhoaHoc}</p>
8             <p className="lead">Người tạo : {props.khoaHoc.NguoiTao}</p>
9             <button className="btn btn-success">Chi Tiết</button>
10        </div>
11    );
12}
13
14 export default khoahoc;
```

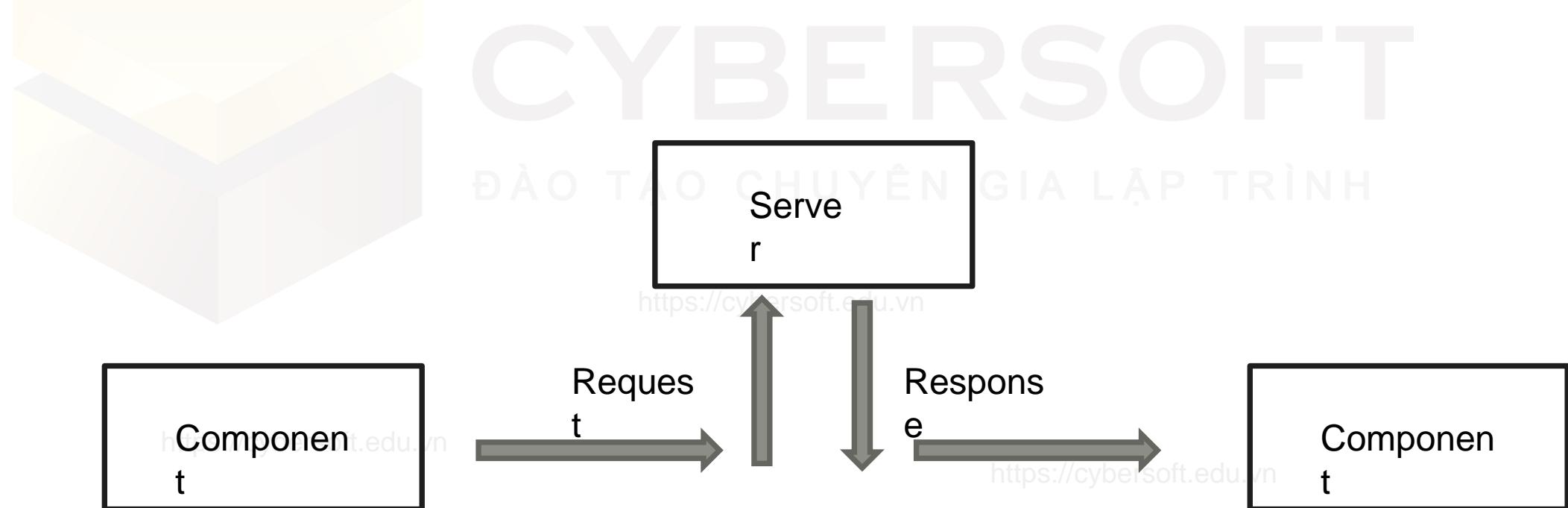
Nâng cấp : tạo async action

- ❖ Để có thể lấy được danh sách khóa học từ api về, lưu trữ trên store và sử dụng, ta phải trả qua khá nhiều công đoạn và thời gian



Nâng cấp : tạo async action

- ❖ Để rút ngắn quãng đường, ta có dispatch 1 action lên store ngay lập tức, trên quãng đường từ component lên tới store, ta sẽ tiến hành gửi request và nhận response



Nâng cấp : tạo async action

- ❖ Bước 1: tại actions/khoaHoc.js , ta sẽ tạo thêm 1 action nữa.

<https://cybersoft.edu.vn>

```
import {GET_COURSE_LIST} from '../constants/actionType'

import Axios from 'axios';

export const saveCourseList = () => {
  return (dispatch) => {
    Axios({
      method: "GET",
      url: 'http://sv.myclass.vn/api/QuanLyTrungTam/DanhSachKhoaHoc'
    }).then(res => {
      dispatch(actGetCourseList(res.data))
    })
    .catch(err => {
      console.log(err)
    })
  }
}

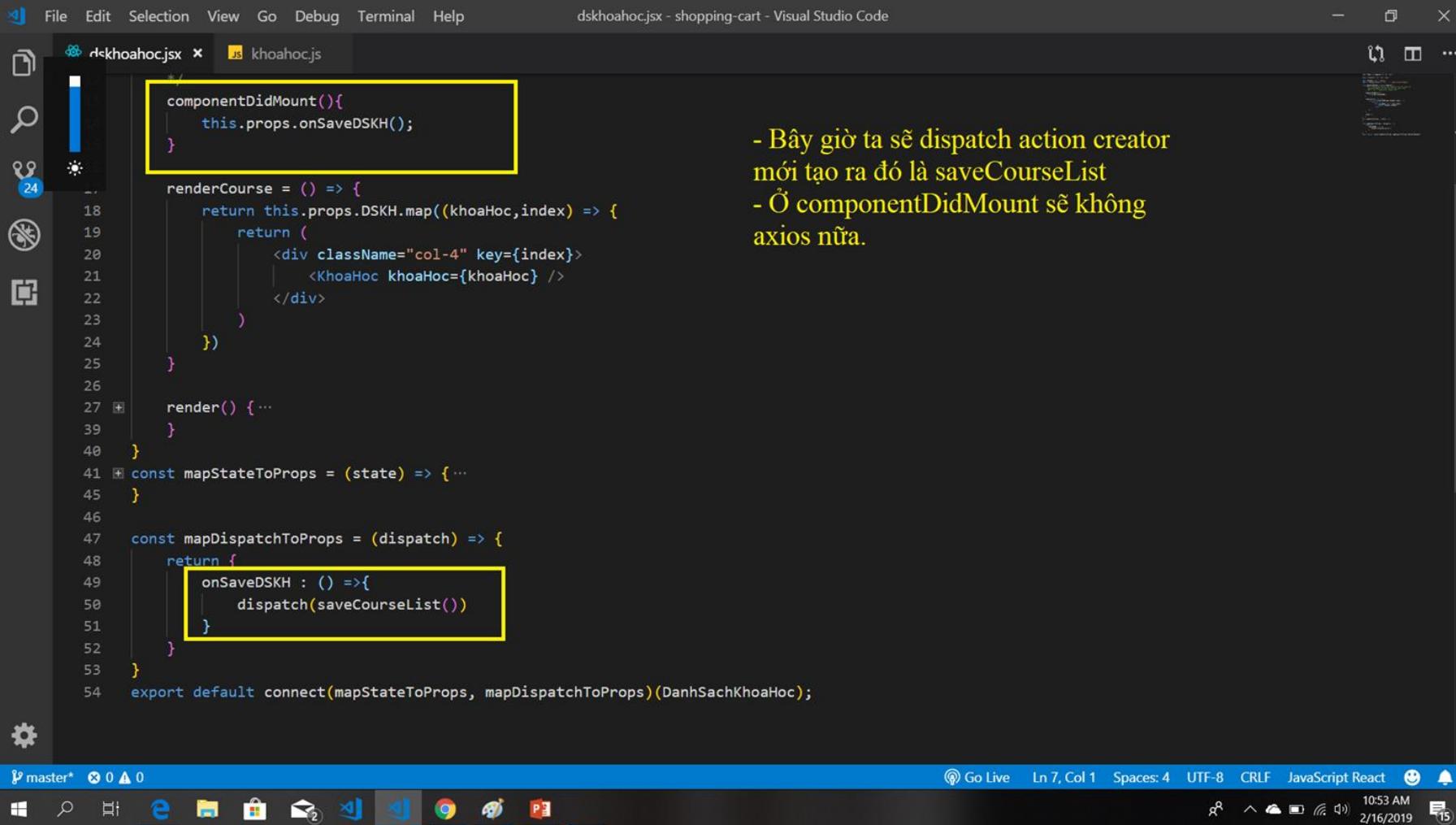
export const actGetCourseList = (danhSachKhoaHoc) => {
  return {
    type: GET_COURSE_LIST,
    //danh sách khóa học gửi lên để lưu trên store
    danhSachKhoaHoc
  }
}
```

Ta sẽ tiến hành dispatch action này lên store, trên đường đi sẽ tiến hành gửi request lên server và nhận response, sau đó dispatch tiếp action GET_COURSE_LIST lên để reducer xử lý

Nâng cấp : tạo async action

- ❖ Bước 2: Tại component DanhSachKhoaHoc, ta sẽ có 1 số thay đổi như sau

<https://cybersoft.edu.vn>



```
File Edit Selection View Go Debug Terminal Help
dskhoahoc.jsx x js khoahoc.js
componentDidMount(){
  this.props.onSaveDSKH();
}

renderCourse = () => {
  return this.props.DSKH.map((khoaHoc,index) => {
    return (
      <div className="col-4" key={index}>
        <KhoaHoc khoahoc={khoaHoc} />
      </div>
    );
  });
}

render() { ... }

const mapStateToProps = (state) => { ... }

const mapDispatchToProps = (dispatch) => {
  return {
    onSaveDSKH : () =>{
      dispatch(saveCourseList());
    }
  }
}

export default connect(mapStateToProps, mapDispatchToProps)(DanhSachKhoaHoc);
```

master* 0 ▲ 0

Go Live Ln 7, Col 1 Spaces: 4 UTF-8 CRLF JavaScript React 1053 AM 2/16/2019

- Nay ta sẽ dispatch action creator mới tạo ra đó là saveCourseList
- Ở componentDidMount sẽ không axios nữa.

Nâng cấp : tạo async action

Tuy nhiên, cách này sẽ có một vấn đề xảy ra. Việc gọi api là bất đồng bộ, cho nên không chắc chắn là khi tới được reducer, response đã trả về hay chưa.

=> Phải sử dụng middleware

Error: Actions must be plain objects. Use custom middleware for async actions.

```
dispatch
D:/CYBERSOFT/cybersoft/FE11/reactjs/Shopping_cart/shopping-cart/node_modules/redux/es/redux.js:192

onSaveDSKH
D:/CYBERSOFT/cybersoft/FE11/reactjs/Shopping_cart/shopping-cart/src/components/CRUD/dskhoahoc.jsx:50

47 | const mapDispatchToProps = (dispatch) => {
48 |   return {
49 |     onSaveDSKH : () =>{
> 50 |       dispatch(saveCourseList())
51 |     }
52 |   }
53 | }
```

[View compiled](#)

```
componentDidMount
D:/CYBERSOFT/cybersoft/FE11/reactjs/Shopping_cart/shopping-cart/src/components/CRUD/dskhoahoc.jsx:14
```

```
11 |   Đây là nơi lý tưởng để gọi request API
12 |   */
13 |   componentDidMount(){
> 14 |     this.props.onSaveDSKH();
15 |   ^ }
16 |
17 |   renderCourse = () => {
```

[View compiled](#)

3 stack frames were collapsed.

This screen is visible only in development. It will not appear if the app crashes in production.
Open your browser's developer console to further inspect this error.

Nâng cấp : Sử dụng middleware

- ❖ Middleware có thể xem như là lớp ngăn cách giữa component và reducer
- ❖ Action được dispatch lên reducer phải đi qua middleware
- ❖ Ta có thể sử dụng middleware để đảm bảo rằng khi tới được reducer, response từ server đã được trả về
- ❖ Có nhiều loại middleware, ở đây ta sử dụng **Redux-thunk**

<https://cybersoft.edu.vn>

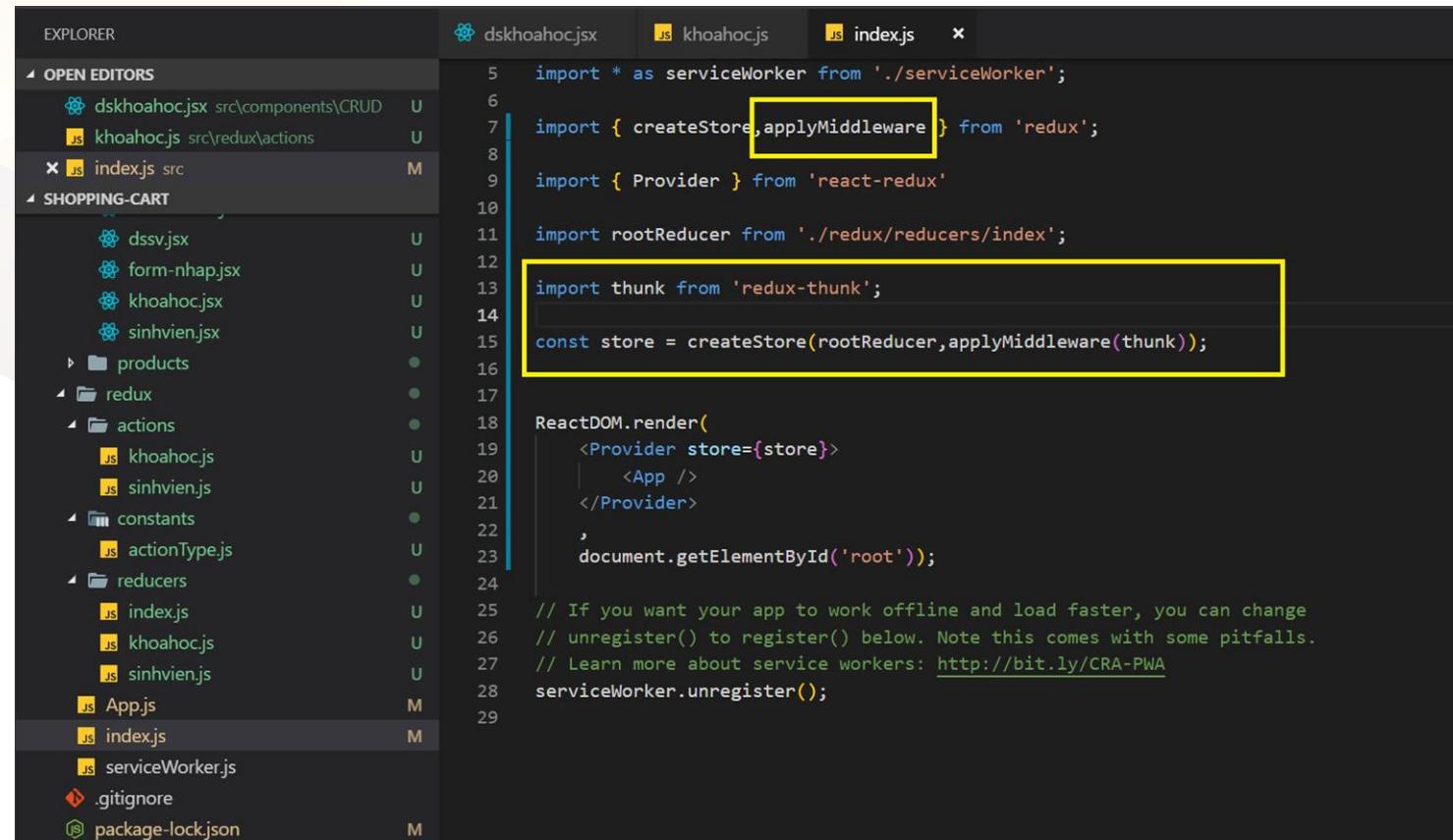
<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Nâng cấp : Sử dụng Redux-thunk

- ❖ Bước 1: cài đặt redux bằng lệnh **npm install --save redux-thunk**
- ❖ Bước 2: tại file index.js, ta sử dụng redux-thunk

<https://cybersoft.edu.vn>



```
import * as serviceWorker from './serviceWorker';
import { createStore, applyMiddleware } from 'redux';
import { Provider } from 'react-redux';

import rootReducer from './redux/reducers/index';

import thunk from 'redux-thunk';

const store = createStore(rootReducer, applyMiddleware(thunk));

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>
,
  document.getElementById('root');

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: http://bit.ly/CRA-PWA
serviceWorker.unregister();

```

Routing

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Routing là gì ?

- ❖ Routing là cơ chế trong single page giúp ta chuyển đổi qua lại giữa các component
- ❖ Để sử dụng được routing với reactjs, ta cần package hỗ trợ đó là React-router-dom
- ❖ Tiến hành cài đặt: **npm install --save react-router-dom**
- ❖ Sử dụng: tại app.js, ta đang có 2 component không thể hiện cùng lúc, đó là danh sách khóa học, và danh sách sinh viên . Do đó ta sẽ dùng routing để quản lý, dựa theo đường dẫn url để hiển thị component tương ứng.
- ❖ Tạo thêm một component home.js nữa để thực hành

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Routing là gì ?

App.js

```
import React, { Component, Fragment } from 'react';
import DanhSachSinhVien from './components/CRUD/dssv';
import FormNhaph from './components/CRUD/form-nhap';
import DanhSachKhoaHoc from './components/CRUD/dskhoahoc';

import { BrowserRouter, Route } from 'react-router-dom'
import Home from './components/CRUD/home';

class App extends Component {
  render() {
    return (
      <BrowserRouter>
        <Fragment>
          <Route path="/sinhvien" component={DanhSachSinhVien} />
          <Route path="/khoaHoc" component={DanhSachKhoaHoc} />
          <Route path="/" component={Home} />
          <FormNhaph />
        </Fragment>
      </BrowserRouter>
    );
  }
}
```

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Home.jsx

<https://cybersoft.edu.vn>

```
dskhoahoc.jsx  js khoahoc.js  js index.js  js App.js  home.jsx  x  js sinhvien.js
1 import React, { Component } from 'react';
2
3 class home extends Component {
4   render() {
5     return (
6       <div>
7         <h1 className="display-4 text-center">Welcome</h1>
8       </div>
9     );
10  }
11
12
13 export default home;
```



<https://cybersoft.edu.vn>

❖ Trong đó :

- ❖ BrowserRouter là component sẽ bao toàn bộ ứng dụng để có thể sử dụng được routing
- ❖ Route hỗ trợ load component dựa theo path tương ứng

Routing là gì ?

- ❖ Ở đây, ta sẽ thấy có một vấn đề xảy ra, đó là khi path là rỗng, thì tất cả các component đều hiện ra
- ❖ Lý do là vì path ở đây ko so sánh toàn bộ mà so sánh theo prefix .Ví dụ , path rỗng là “/” , path sinh viên là “/sinhvien” , cả 2 đều bắt đầu với “/”, do đó đều hợp lệ và đều được hiện lên
- ❖ Có 2 cách fix điều này.

❑ Cách 1 : sử dụng exact (chính xác là path rỗng mới hiện Home)

```
<Route path="/" exact component={Home} />
```

❑ Cách 2: sử dụng component Switch của react-router-dom (tương tự như switch case, chỉ 1 trong các Route phép hiện)

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'
import Home from './components/CRUD/home';

class App extends Component {
  render() {
    return (
      <BrowserRouter>
        <Fragment>
          <Switch>
            <Route path="/sinhvien" component={DanhSachSinhVien} />
            <Route path="/khoaHoc" component={DanhSachKhoaHoc} />
            <Route path="/" component={Home} />
          </Switch>
          <FormNhap />
        </Fragment>
      </BrowserRouter>
    );
  }
}

export default App;
```

```
11   render() {
12     return (
13       <BrowserRouter>
14         <Fragment>
15           <div className="menu">
16             <Link to="/">Home</Link>
17             <Link to="/sinhvien">Sinh Viên</Link>
18             <Link to="/khoaHoc">Khóa Học</Link>
19           </div>
20
21         <Switch>
22           <Route path="/sinhvien" render={({props}) => <DanhSachSinhVien {...props} />} />
23           <Route path="/khoaHoc" render={({props}) => <DanhSachKhoaHoc {...props} />} />
24           <Route path="/" render={({props}) => <Home {...props} />} />
25         </Switch>
26
27         <FormNhap />
28       </Fragment>
29     </BrowserRouter>
30   );
31 }
32 }
```

Cách viết khác

Chuyển đổi component

- ❖ Sử dụng component <Link> để chuyển đổi component
- ❖ Lưu ý : giá trị của thuộc tính **to** phải tương ứng với **path** đã xét với Route

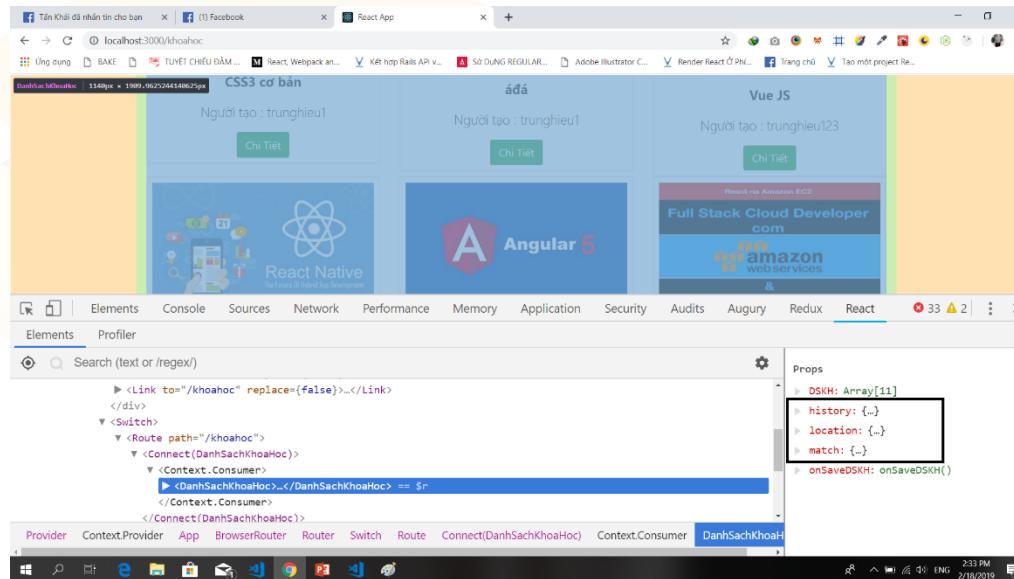
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like dskhoahoc.jsx, khoahoc.jsx, index.jsx, App.jsx, form-nhap.jsx, and home.jsx.
- Code Editor:** Displays the App.jsx code. A yellow box highlights the following section of code:

```
<Switch>
  <Route path="/sinhvien" component={DanhSachSinhVien} />
  <Route path="/khoa" component={DanhSachKhoaHoc} />
  <Route path="/" component={Home} />
```
- Terminal:** Shows the command `./src/components/CRUD/khoa.jsx` and a warning about img elements.
- Status Bar:** Shows the date and time as 2/18/2019 2:28 PM.

Chuyển đổi component

- ❖ Các component được load lên sử dụng Routing sẽ mặc định có thêm 3 đối tượng mới trong props, đó là : history, match , location.



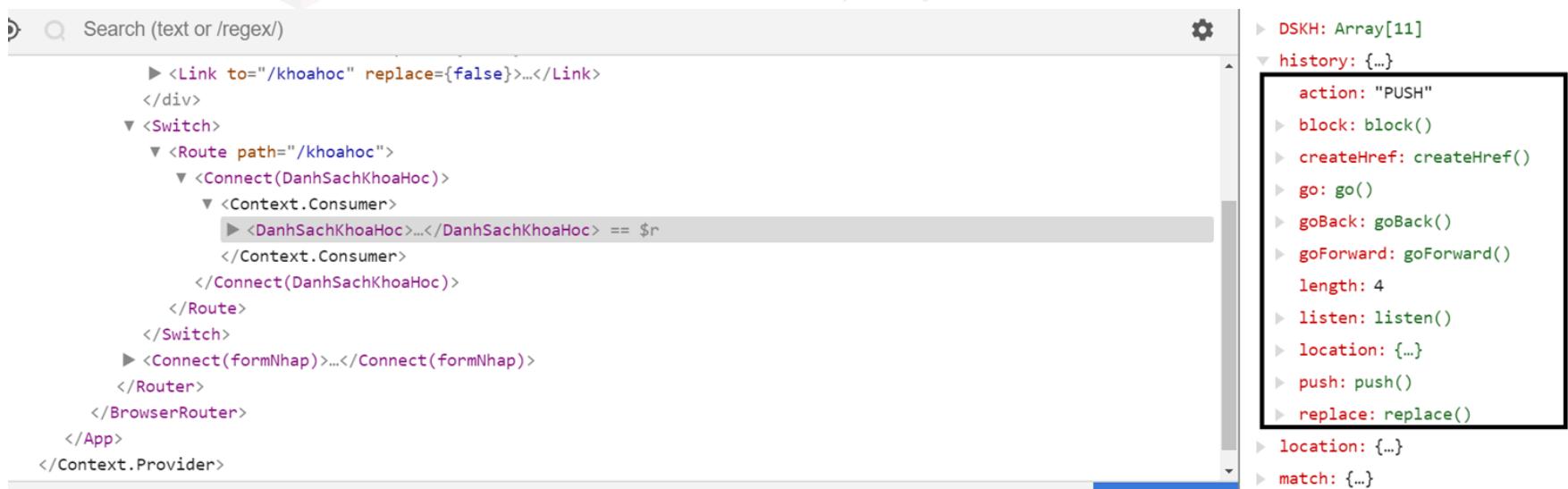
<https://cybersoft.edu.vn>

- ❖ Vậy công dụng của 3 thuộc tính mới này là gì?

<https://cybersoft.edu.vn>

History

- ❖ Quay lại component Link, ta sử dụng nó để khi click vào thẻ có thể chuyển route để hiện component tương ứng
- ❖ Vậy trong trường hợp ta có một chức năng đăng nhập, ta chờ sau khi đăng nhập thành công, mới chuyển từ component Đăng Nhập sang component Trang Chủ , ta không thể dùng Link được vì đây là code javascript
=> đối tượng history được thêm vào để giải quyết điều này, nó cung cấp các phương thức để chuyển đổi route.

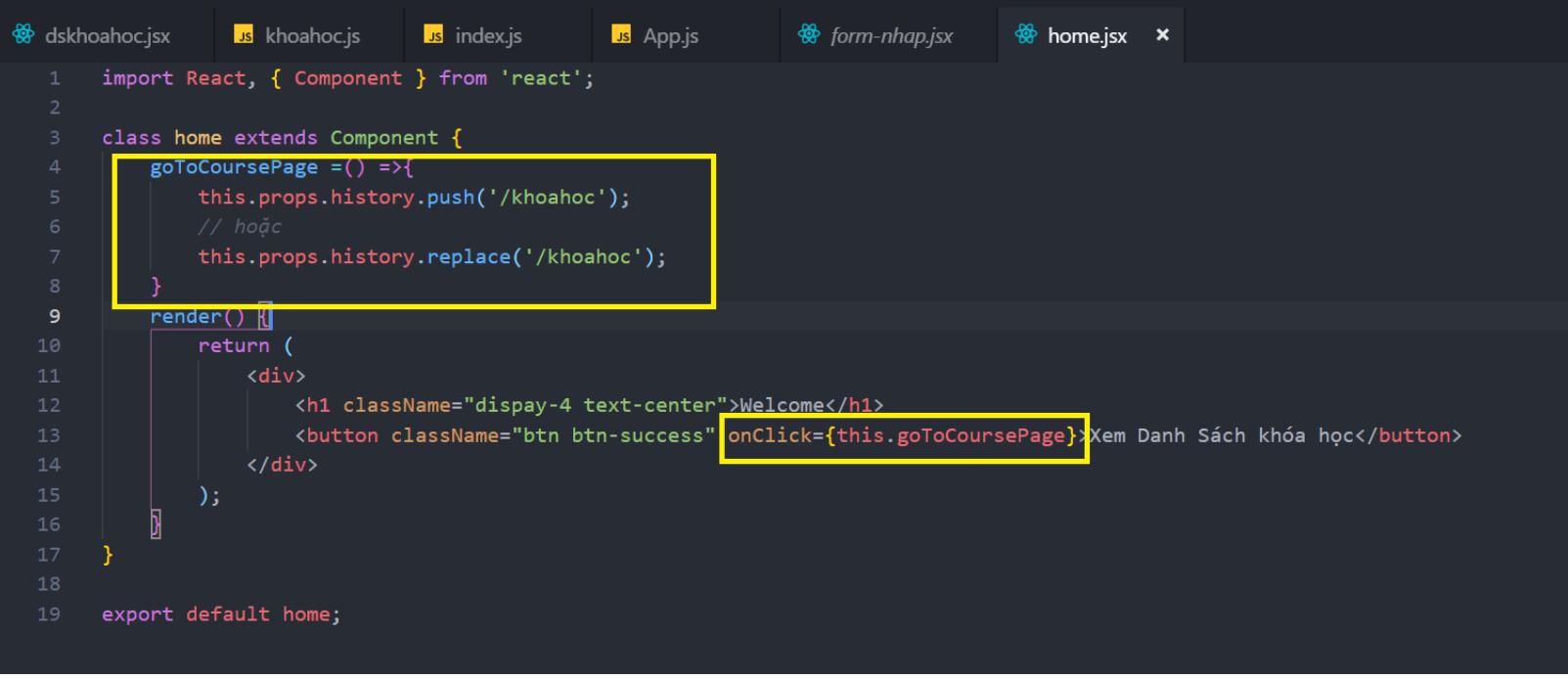


The screenshot shows the browser's developer tools with the element inspector open. On the left, the DOM tree is visible, showing components like `<Link>`, `<Switch>`, `<Route path="/khoahoc">`, and `<Context.Consumer>`. On the right, the `history` object is expanded, showing its methods and properties:

```
▶ DSKH: Array[11]
▼ history: {...}
  action: "PUSH"
  ▶ block: block()
  ▶ createHref: createHref()
  ▶ go: go()
  ▶ goBack: goBack()
  ▶ goForward: goForward()
  length: 4
  ▶ listen: listen()
  ▶ location: {...}
  ▶ push: push()
  ▶ replace: replace()
  ▶ location: {...}
  ▶ match: {...}
```

History

- ❖ Ví dụ, thay vì sử dụng component Link, ta có thể dùng history như sau
- ❖ Sự khác biệt giữa push và replace:
 - ❖ Push: đẩy ta qua component mới và đưa nó vào dòng thời gian, có thể back lại component trước đó bằng nút back của browser
 - ❖ Replace: thay thế component cũ bằng component mới, không thể back lại được



```
dskhoaohoc.jsx | js khoahoc.js | js index.js | js App.js | form-nhap.jsx | home.jsx ×  
1 import React, { Component } from 'react';  
2  
3 class home extends Component {  
4   goToCoursePage =() =>{  
5     this.props.history.push('/khoaohoc');  
6     // hoặc  
7     this.props.history.replace('/khoaohoc');  
8   }  
9   render() {  
10     return (  
11       <div>  
12         <h1 className="display-4 text-center">Welcome</h1>  
13         <button className="btn btn-success" onClick={this.goToCoursePage}>Xem Danh Sách khóa học</button>  
14       </div>  
15     );  
16   }  
17 }  
18  
19 export default home;
```

Match

- ❖ Đối tượng match cung cấp một số thuộc tính hỗ trợ: path hiện tại, tham số được truyền qua url...
- ❖ Ở đây ta sẽ sử dụng thuộc tính params để thực hiện chức năng xem chi tiết khóa học

The screenshot shows a code editor with a hierarchical tree view on the left and a detailed object viewer on the right.

Hierarchical Tree View:

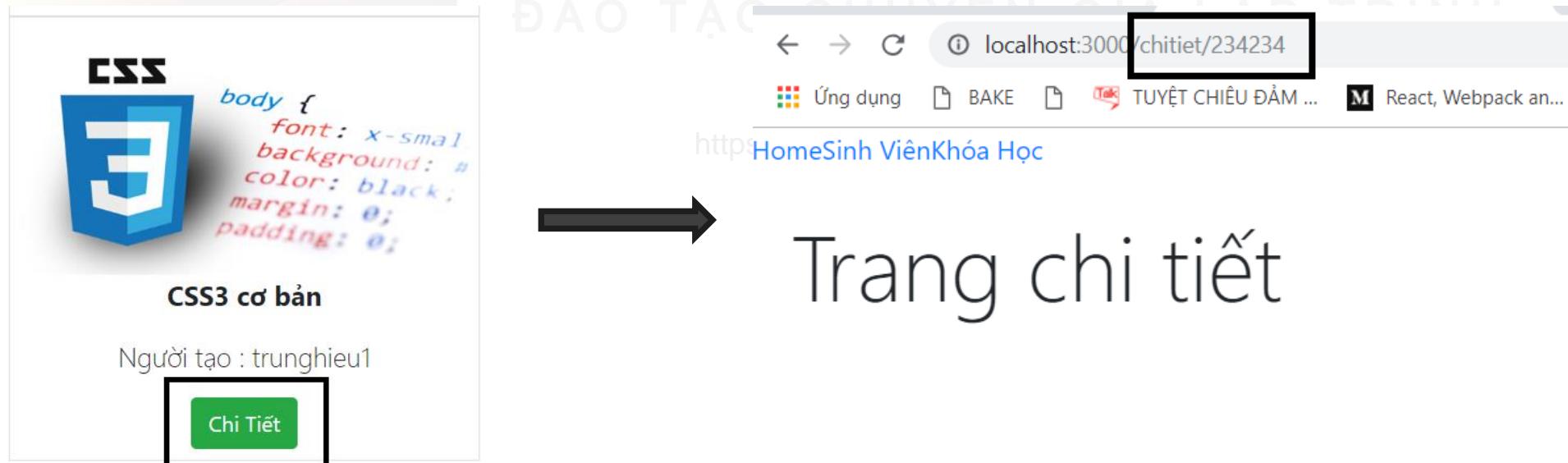
- <Provider>
- <Context.Provider>
- <App>
- <BrowserRouter>
- <Router>
- <div className="menu">
- <Link to="/" replace={false}>...</Link>
- <Link to="/sinhvien" replace={false}>...</Link>
- <Link to="/khoa hoc" replace={false}>...</Link>
- </div>
- <Switch>
- <Route path="/khoa hoc">
- <Connect(Danh Sach Khoa Hoc)>
- <Context.Consumer>
- <Danh Sach Khoa Hoc>...</Danh Sach Khoa Hoc> == \$r
- </Context.Consumer>

Detailed Object Viewer (match object):

- DSKH: Array[11]
- history: {...}
- location: {...}
- match: {...} (highlighted with a red box)
 - isExact: true (checkbox checked)
 - params: {...} (empty object)
 - path: "/khoa hoc"
 - url: "/khoa hoc"
- onSaveDSKH: onSaveDSKH()

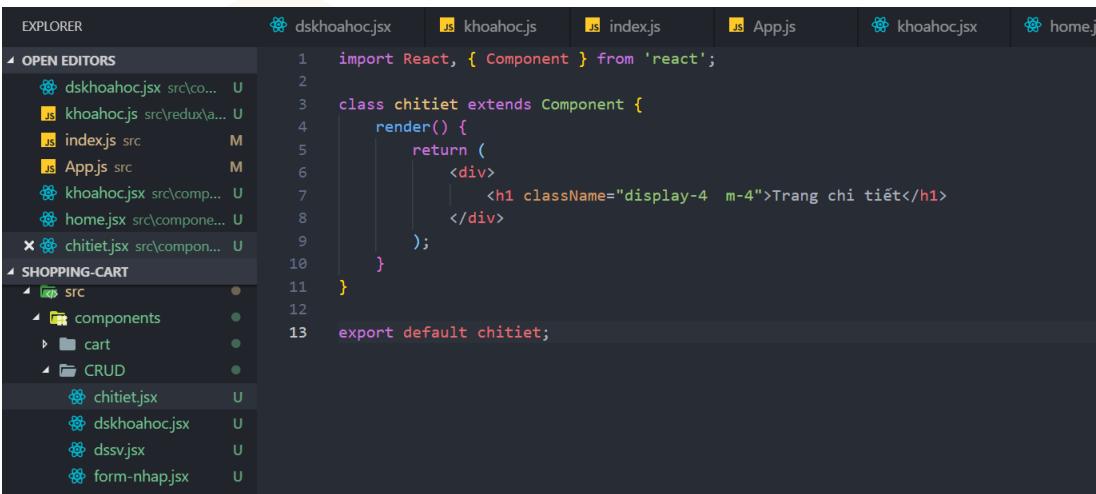
Match

- ❖ Chức năng xem chi tiết khóa học : khi click vào nút xem chi tiết, ta sẽ tiến hành chuyển trang trang chi tiết, hiển thị thông tin của khóa học đó ra màn hình. Vấn đề làm sao để biết ở component DanhSachKhoaHoc , người ta đã chọn khóa nào để xem ?
=> tiến hành truyền mã khóa học qua trang chi tiết thông qua url, ở trang chi tiết, ta sẽ lấy mã khóa học từ url xuống.



Match

- ❖ Bước 1: tạo component ChiTietKhoaHoc và gắn cho nó một path tương ứng

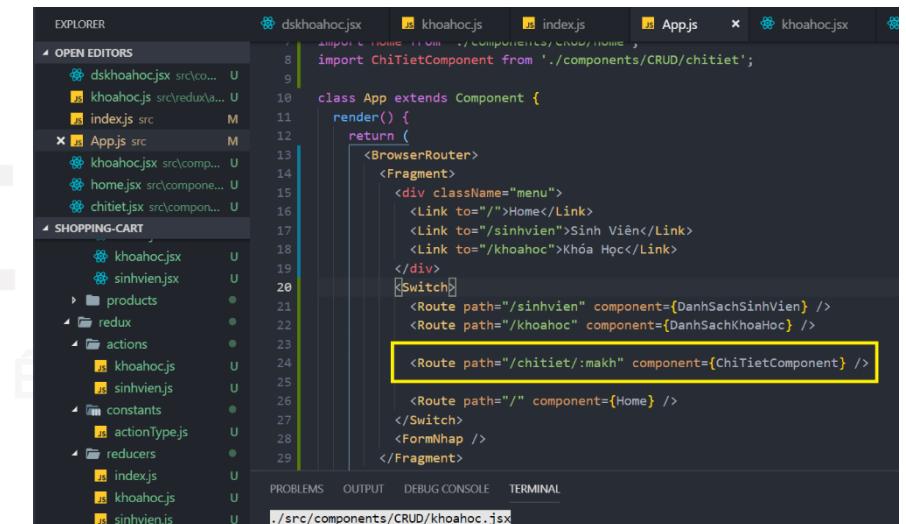


```

EXPLORER
dskhoahoc.jsx  khoahoc.js  index.js  App.js  khoahoc.jsx  home.jsx
OPEN EDITORS
dskhoahoc.jsx src\co... U
khoa... src\redux\... U
index.js src M
App.js src M
khoa... src\comp... U
home.jsx src\compone... U
x chiti... src\compon... U
SHOPPING-CART
SRC
components
cart
CRUD
chiti...jsx U
dskhoahoc.jsx U
dssv.jsx U
form-nhap.jsx U
    
```

```

1 import React, { Component } from 'react';
2
3 class chiti... extends Component {
4     render() {
5         return (
6             <div>
7                 <h1 className="display-4 m-4">Trang chi tiết</h1>
8             </div>
9         );
10    }
11 }
12
13 export default chiti...;
    
```



```

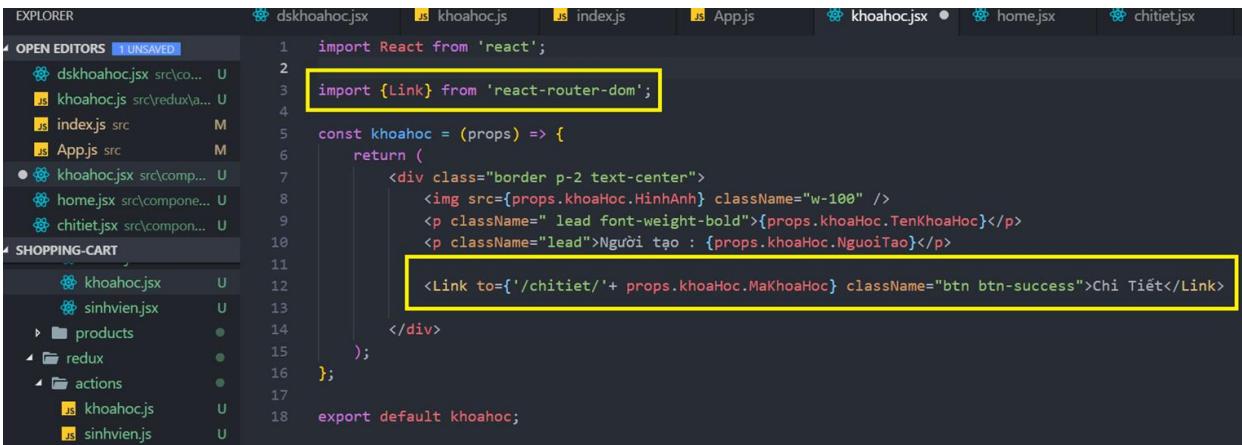
EXPLORER
dskhoahoc.jsx  khoahoc.js  index.js  App.js  khoahoc.jsx
OPEN EDITORS
dskhoahoc.jsx src\co... U
khoa... src\redux\... U
index.js src M
x App.js src M
khoa... src\comp... U
home.jsx src\compone... U
chiti...jsx src\compon... U
SHOPPING-CART
SRC
products
redux
actions
khoa...jsx U
sinhvien.jsx U
constants
actionType.js U
reducers
index.js U
khoa...jsx U
sinhvien.jsx U
    
```

```

8 import { Home, SinhVien, KhoaHoc } from './components/CRUD/home';
9
10 class App extends Component {
11     render() {
12         return (
13             <BrowserRouter>
14                 <Fragment>
15                     <div className="menu">
16                         <Link to="/">Home</Link>
17                         <Link to="/sinhvien">Sinh Viên</Link>
18                         <Link to="/khoaHoc">Khóa Học</Link>
19                     </div>
20                 <Switch>
21                     <Route path="/sinhvien" component={DanhSachSinhVien} />
22                     <Route path="/khoaHoc" component={DanhSachKhoaHoc} />
23                     <Route path="/chiti.../:makh" component={Chiti...Component} />
24
25                     <Route path="/" component={Home} />
26                 </Switch>
27             </Fragment>
28         );
29     }
30 }
31
32 export default App;
    
```

https://cybersoft.edu.vn

- ❖ Bước 2: ở component Khoa học, thay đổi button xem chi tiết thành component Link để chuyển.



```

EXPLORER
dskhoahoc.jsx  khoahoc.js  index.js  App.js  khoahoc.jsx  home.jsx  chiti...jsx
OPEN EDITORS
dskhoahoc.jsx src\co... U
khoa... src\redux\... U
index.js src M
App.js src M
khoa... src\comp... U
home.jsx src\compone... U
chiti...src\compon... U
SHOPPING-CART
SRC
khoa...jsx U
sinhvien.jsx U
products
redux
actions
khoa...jsx U
sinhvien.jsx U
    
```

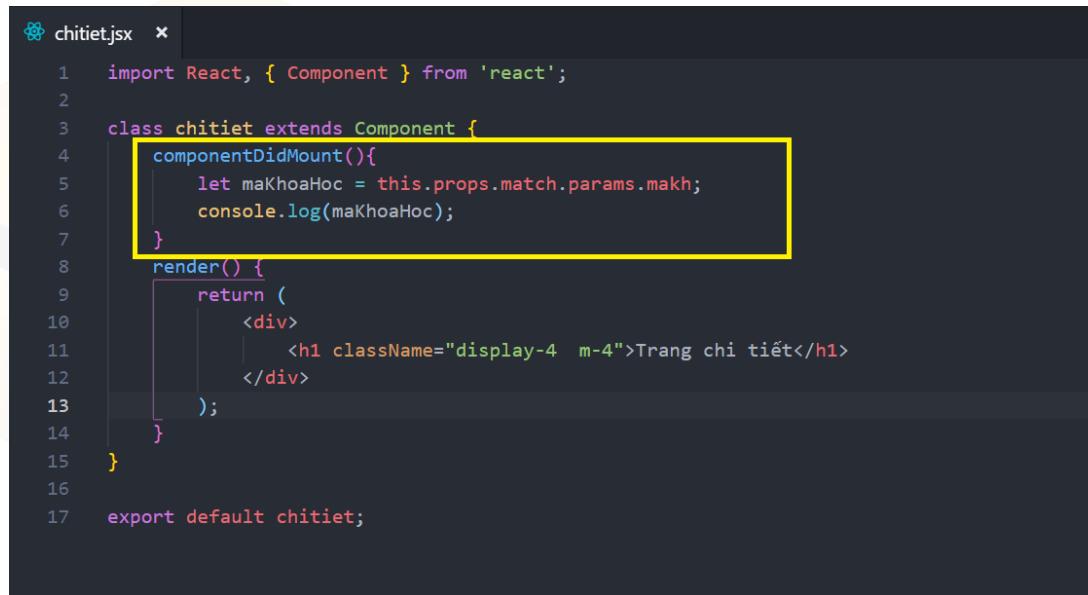
```

1 import React from 'react';
2
3 import {Link} from 'react-router-dom';
4
5 const khoahoc = (props) => {
6     return (
7         <div className="border p-2 text-center">
8             <img src={props.khoaHoc.HinhAnh} className="w-100" />
9             <p className="lead font-weight-bold">{props.khoaHoc.TenKhoaHoc}</p>
10            <p className="lead">Người tạo : {props.khoaHoc.NguoiTao}</p>
11
12            <Link to={'/chiti.../' + props.khoaHoc.MaKhoaHoc} className="btn btn-success">Chi Tiết</Link>
13        </div>
14    );
15 }
16
17 export default khoahoc;
    
```

https://cybersoft.edu.vn

Match

- ❖ Bước 3 : tại component ChiTietKhoaHoc , tiến hành lấy tham số từ trên url xuống với đối tượng match.



```
chitiets.jsx
1 import React, { Component } from 'react';
2
3 class chitiets extends Component {
4     componentDidMount(){
5         let maKhoaHoc = this.props.match.params.makh;
6         console.log(maKhoaHoc);
7     }
8     render() {
9         return (
10             <div>
11                 <h1 className="display-4 m-4">Trang chi tiết</h1>
12             </div>
13         );
14     }
15 }
16
17 export default chitiets;
```

<https://cybersoft.edu.vn>

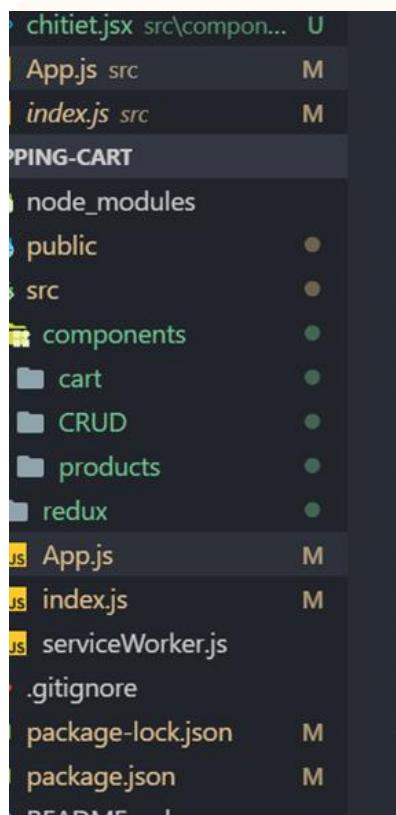
- ❖ Bước 4 : sử dụng mã khóa học lấy được , gửi request lên server lấy về chi tiết phim (tự làm)

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Redirect

- ❖ Component Redirect được cung cấp bởi react-router-dom hỗ trợ ta điều hướng từ path này tới path khác
- ❖ Ví dụ :



```
15      <div className="menu">
16        <Link to="/">Home</Link>
17        <Link to="/sinhvien">Sinh Viên</Link>
18        <Link to="/khoahoc">Khóa Học</Link>
19      </div>
20
21      <Switch>
22        <Route path="/sinhvien" render={({props}) => <DanhSachSinhVien {...props} />} />
23        <Route path="/khoahoc" render={({props}) => <DanhSachKhoaHoc {...props} />} />
24        <Route path="/home" render={({props}) => <Home {...props} />} />
25
26        // Nếu path được nhập vào không giống ở trên, thì Redirect sẽ chuyển hướng về
27        // path home
28        <Redirect to="/home" />
29
30      </Switch>
31
32      <FormNhap />
33    </Fragment>
34  </BrowserRouter>
35
36);
```

The code snippet shows a portion of the 'index.js' file from a React application. It includes a navigation menu with links to 'Home', 'Sinh Viên', and 'Khóa Học'. Below the menu is a 'Switch' component that handles three specific routes: '/sinhvien', '/khoahoc', and '/home'. A yellow box highlights the code for the default route: '`<Redirect to="/home" />`'. This indicates that if a path is entered that does not match one of the defined routes, the application will redirect the user to the '/home' page.

<https://cybersoft.edu.vn>

Cơ chế Guard (bảo vệ Route trong React)

- ❖ Để bảo vệ Route tránh cho sinh viên truy cập vào trường hợp không đủ điều kiện.
- ❖ Ví dụ: khi tài khoản là sinh viên nhưng lại muốn truy cập vào route admin,... hoặc sinh viên chưa có tài khoản hoặc chưa đăng nhập thì không được truy cập vào một số Route...
- ❖ Các bước thực hiện:
 - ❖ 1. Tạo ra một component tên là **Auth**. Auth được gọi là 1 HOC (High Order Component), loại component này không dùng để render ra giao diện mà chỉ dùng để bao 1 component khác, trong trường hợp này là dùng để bao component Route

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

Cơ chế Guard (bảo vệ Route trong React)

Tại component Auth.js

```
import React from 'react';
import { Route, Redirect } from 'react-router-dom';
const auth = ({ path, Component }) => [
  return (
    <Route path={path} render={(routeProps) => {
      if (localStorage.getItem('loginUser')) {
        return <Component {...routeProps} />;
      }
      alert('Vui lòng đăng nhập');
      return <Redirect to="/login" />
    }} />
  );
];
export default auth;
```

<https://cybersoft.edu.vn>

Tại component App.js

```
import { BrowserRouter, Switch } from 'react-router-dom';
class App extends Component {
  render() {
    return (
      <BrowserRouter>
        <Switch>
          /* DÙNG GUARD BẢO VỆ ROUTE (YÊU CẦU ĐĂNG NHẬP MỚI ĐƯỢC TRUY CẬP) */
          <Auth
            path="/admin"
            Component={AdminLayout}
          />
          <Auth
            path="/admin"
            Component={Login}
          />
        </Switch>
      </BrowserRouter>
    );
  }
}
export default App;
```

<https://cybersoft.edu.vn>

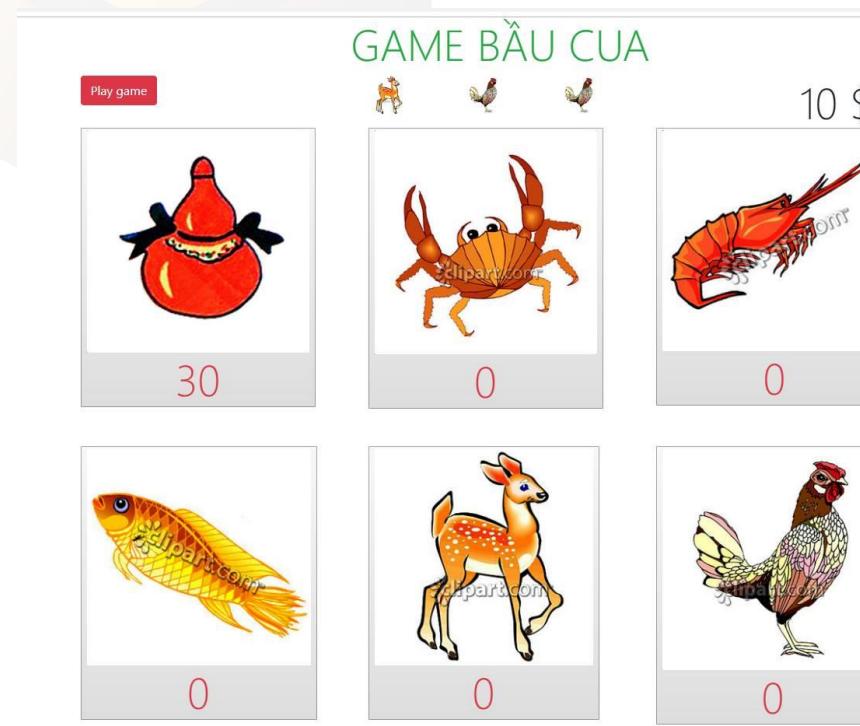
- ❖ Bây giờ khi load app component, ta sẽ không trực tiếp Route nữa, mà ta sẽ load component Auth thay thế, ta sẽ truyền path và Component ra muộn render vào trong Auth dưới dạng Props.
- ❖ Khi component Auth được load, nó cũng sẽ trả ra 1 Route với path tương ứng là truyền vào, tuy nhiên lúc này sẽ không render ra component ngay mà sẽ check trước
- ❖ Nói chung (routeProps ở đây chính là 3 props History, Match, Location) truyền vào trong component được render ra. Ngược lại sẽ render ra component Redirect được import từ react-router-dom với nhiệm vụ chuyển hướng Route về lại trang login

❑ React hook: dự án Lập Sòng Bầu Cua

❑ Trong phần này ta sẽ cùng tìm hiểu các vấn đề sau đây:

- ❑ React hook là gì ?
- ❑ Tại sao lại cho ra đời react hook ?
- ❑ Nên sử dụng hook hay class component thông thường
- ❑ Áp dụng hook vào dự án nhỏ Lắc Bầu Cua , kết hợp redux.

<https://cybersoft.edu.vn>



YÊN GIA LẬP TRÌNH

edu.vn

<https://cybersoft.edu.vn>

➤ React hook: dự án shopping cart

○ React hook là gì?

- Như ta đã biết ở các bài trước, state và lifecycle chỉ có thể sử dụng được trong class component
- Hook ra đời, cho phép ta có thể sử dụng 2 khái niệm này trong 1 functional component, giúp code ngắn và clean hơn, đó chính là react hook.

○ Tại sao lại cho ra đời React hook?

- Hướng đối tượng luôn luôn là vấn đề khó khi học lập trình. con trỏ this, thuộc tính , phương thức... đó là rất nhiều logic và gây khó với một số bạn...
- React hook ra đời, cho phép tận dụng được các tính năng của class component trong functional component

○ Nên xài hook hay class component ?

- Không có sự chênh lệch giữa performance giữa 2 loại này, cũng không cần loại bỏ hoàn toàn kiến thức về class component trước đây ta đã học, hook ra đời chỉ là thêm cho chúng ta một option để chọn khi làm mà thôi. Nắm rõ loại nào thì ta sử dụng loại đó

➤ React hook: Lý thuyết cơ bản

- Các hook cơ bản và phổ biến được cung cấp bởi react mà ta sẽ tìm hiểu bao gồm:
 - useState
 - useEffect
 - useCallback
 - useMemo

<https://cybersoft.edu.vn>

CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

➤ React hook: useState

- Ví dụ đơn giản: nhấn button và tăng number lên 1
- Để number tăng, đồng thời render lại giao diện, ta cần sử dụng state, useState cho phép ta làm điều đó, cho phép sử dụng state ở functional component

increase number

Number: 0

```
import React, { useState } from "react";

function ChildComponent(props) {
  const [number, setNumber] = useState(0);
  const increaseNumber = () => {
    setNumber(number + 1);
  };
  return (
    <React.Fragment>
      <button onClick={increaseNumber}>increase number</button>
      <h1>Number: {number}</h1>
    </React.Fragment>
  );
}

export default React.memo(ChildComponent);
```

1. useState phải được import từ react
2. useState trả về một mảng như hình [number, setNumber], trong đó number là state của component, khi number thay đổi thì component render lại, setNumber là phương thức dùng để set lại number. Cả number lẫn setNumber đều là tên tự đặt tùy ý, chỉ cần đúng thứ tự
3. Một component có thể sử dụng nhiều useState()

➤ React hook: useEffect

- Ví dụ đơn giản: vẫn là nhấn button và tăng number lên 1
- useEffect giúp sử dụng được lifecycle trong functional component.
- useEffect tương ứng với 3 lifecycle đã học trong class component: componentDidMount, componentDidUpdate và componentWillUnmount

```
import React, { useState, useEffect } from "react";

function ChildComponent(props) {
  const [number, setNumber] = useState(0);

  useEffect(() => {
    console.log(`luôn chạy khi component did mount,
    did update và unmount`);
  });

  useEffect(() => {
    console.log(`khi component update, sẽ check thử nếu
    thực sự number
    `);
  }, [number]);

  useEffect(() => {
    console.log(`chạy khi component did mount,
    và chỉ chạy một lần duy nhất`);
  }, []);

  const increaseNumber = () => {
    setNumber(number + 1);
  };

  return (
    <React.Fragment>
      <button onClick={increaseNumber}>increase number</button>
      <h1>Number: {number}</h1>
    </React.Fragment>
  );
}
```

1. Một component có thể sử dụng nhiều useEffect

1. useEffect có tham số thứ 2 là 1 mảng các giá trị, khi component được update, nếu 1 trong số các biến trong mảng này thay đổi, mới chạy lại useEffect, ngược lại sẽ không chạy
2. Mảng rỗng đồng nghĩa với việc useEffect đó chỉ chạy 1 lần duy nhất khi component được khởi tạo

- Trước khi thảo luận tiếp về 2 hook còn lại là useCallback và useEffect, ta hãy cùng điểm qua một khái niệm mới , gọi là Memo (tương ứng với PureComponent trong class component)
- Đầu tiên, tạo ra 2 component: DemoHook và DemoChild

ChildComponent

```
import React from "react";

function ChildComponent(props) {
  console.log("Child render");
  return <React.Fragment>Demo Hook Child Component</React.Fragment>;
}

export default ChildComponent;
```

DemoComponent

```
import React, { useState } from "react";
import DemoChild from "./child";

function ChildComponent(props) {
  const [number, setNumber] = useState(0);

  const increaseNumber = () => {
    setNumber(number + 1);
  };

  return [
    <React.Fragment>
      <button onClick={increaseNumber}>increase number</button>
      <h1>Number: {number}</h1>
      <DemoChild />
    </React.Fragment>
  ];
}

export default React.memo(ChildComponent);
```

- Ở đây, khi ta nhấn nút increase number, ta thay đổi number, kéo theo **DemoComponent** render lại, dẫn tới là component con là **ChildComponent** cũng render lại theo, dù ko hề có sự thay đổi, do đó, react hỗ trợ **Memo**, giống với **PureComponent** ở class component, giúp component chỉ render lại khi props hoặc state của nó thực sự thay đổi
- Cách dùng: như vậy, childComponent chỉ thực sự re-render khi props hoặc state của nó thay đổi

```
import React, { memo } from "react";

function ChildComponent(props) {
  console.log("Child render");
  return <React.Fragment>Demo Hook Child Component</React.Fragment>;
}

export default memo(ChildComponent);
```

<https://cybersoft.edu.vn>

- Quay lại về **useCallback**, ta sẽ xét tới trường hợp dưới đây
- Ở component cha, ta có một hàm là **showNumber** với chức năng đơn giản là console log **number** ra, sau đó truyền vào component con **ChildComponent**.
- Về cơ bản, ở **ChildComponent** ta đã sử dụng **memo**, mà **showNumber** lại là hàm, nên đương nhiên nó sẽ ko đổi, dẫn tới component **ChildComponent** sẽ không được render lại dù component cha có render .
- Kiểm chứng kết quả, ta thấy **ChildComponent** vẫn bị render lại

```
import React, { useState } from "react";
import DemoChild from "./child";
function ParentComponent(props) {
  const [number, setNumber] = useState(0);

  const increaseNumber = () => {
    setNumber(number + 1);
  };

  const showNumber = () => {
    console.log(number);
  };

  return (
    <React.Fragment>
      <button onClick={increaseNumber}>increase number</button>
      <h1>Number: {number}</h1>
      <DemoChild showNumber={showNumber} />
    </React.Fragment>
  );
}

export default ParentComponent;
```

CHUYÊN GIA LẬP TRÌNH

/cybersoft.edu.vn

https://cybersoft.edu.vn

- Lý do: khi component Cha render, nó sẽ render lại tất cả trong function ParentComponent như ta thấy ở dưới, kéo theo **showNumber** sẽ được khai báo lại một lần nữa, tức là nó bị thay đổi, làm cho ChildComponent bị re-render.
- Do đó ở đây ta cần **useCallback**, **showNumberCallback** ở đây là một bản snapshot của **showNumber**, và nó chỉ được khai báo lại khi **number** thay đổi, nếu để mảng rỗng, có nghĩa là sẽ khai báo 1 lần duy nhất.
- Dẫn tới , khi component cha thay đổi state và render lại, nếu number không đổi, thì **showNumberCallback** sẽ không đổi, dẫn tới component con sẽ ko render lại

```
import React, { useState, useCallback } from "react";
import DemoChild from "./child";
function ParentComponent(props) {
  const [number, setNumber] = useState(0);

  const increaseNumber = () => {
    setNumber(number + 1);
  };

  const showNumber = () => {
    console.log(number);
  };

  const showNumberCallback = useCallback(showNumber, [number]);

  return (
    <React.Fragment>
      <button onClick={increaseNumber}>increase number</button>
      <h1>Number: {number}</h1>
      <DemoChild showNumber={showNumberCallback} />
    </React.Fragment>
  );
}
```

https://cybersoft.edu.vn

https://cybersoft.edu.vn

- Tiếp theo về **useMemo**, nó giống như **useCallback**, điểm khác là **useCallback** trả về một hàm, còn **useMemo** trả về một giá trị
- Xét ví dụ dưới đây: ở đây ta có hàm **numberUp** sẽ tiến hành tính toán cái gì đó và trả ra một giá trị như ví dụ, điều đặc biệt ta có thể nhận thấy, là ta chỉ cần tính một lần thôi, vì cảng bản không có gì thay đổi cả. Nhưng khi click button để đổi state, component render lại, sẽ gọi lại **numberUp** chạy một lần nữa, tính toán và trả ra giá trị, đây là điều không cần thiết.

```
import React, { useState } from "react";

function ParentComponent(props) {
  const [number, setNumber] = useState(0);

  const increaseNumber = () => {
    setNumber(number + 1);
  };

  const numberUp = () => {
    let i = 0;
    while (i < 1000) i++;
    console.log(i);
    return i;
  };

  return (
    <React.Fragment>
      <button onClick={increaseNumber}>increase number</button>
      <h1>Number: {numberUp()}</h1>
    </React.Fragment>
  );
}
```

CHUYÊN GIA LẬP TRÌNH

<https://cybersoft.edu.vn><https://cybersoft.edu.vn>

- **useMemo** sẽ giúp ta cache lại giá trị của **numberUp**, là chỉ thực hiện tính toán lại khi có sự thay đổi. Vì đây là điểm khác biệt, **useCallback** trả ra một bản snapshot của 1 hàm, còn **useMemo** chỉ trả ra 1 giá trị thôi.
- **memoizedNumber** chính là giá trị được return từ **numberUp()**, và nó chỉ được tính toán một lần, vì ở đây là mảng rỗng, nếu muốn **memoizedNumber** được tính toán lại khi **number** thay đổi thì thêm **number** vào mảng là được

```
const increaseNumber = () => {
  setNumber(number + 1);
};

const numberUp = () => {
  let i = 0;
  while (i < 1000) i++;
  console.log(i);
  return i;
};

const memoizedNumber = useMemo(() => numberUp[], []);

return (
  <React.Fragment>
    <button onClick={increaseNumber}>increase number</button>
    <h1>Number: {memoizedNumber}</h1>
  </React.Fragment>
);

export default ParentComponent;
```

CHUYÊN GIA LẬP TRÌNH

https://cybersoft.edu.vn

https://cybersoft.edu.vn

Làm quen với thư viện Material UI

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❖ Material UI là thư viện số 1 hiện tại kết hợp với react, tích hợp sẵn các component như : button, table, tab...
- ❖ Material UI khá hay vì nó cho phép ta custom và tạo ra một bộ theme cho chính trang web của mình
- ❖ Sau đây ta sẽ đi tìm hiểu và material UI và cách sử dụng
- ❖ Website: <https://material-ui.com/>
- ❖ Để sử dụng , tiến hành cài đặt material ui vào project : [*npm i @material-ui/core*](#)

<https://cybersoft.edu.vn>

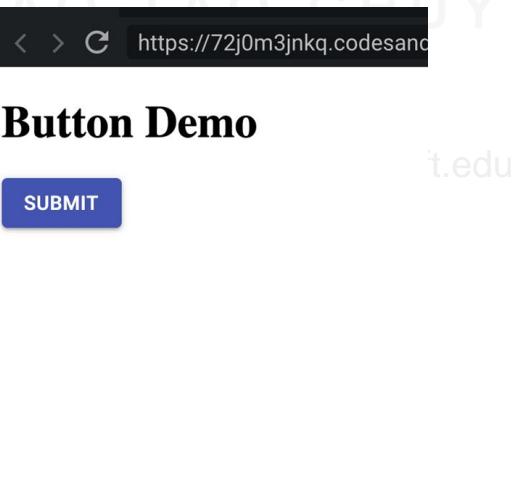
<https://cybersoft.edu.vn>

Làm quen với thư viện Material UI

<https://cybersoft.edu.vn><https://cybersoft.edu.vn>

- ❖ Sử dụng component có sẵn : mỗi component có một cách sử dụng khác nhau và được mô tả chi tiết trên trang của material ui, dưới đây là một ví dụ đơn giản sử dụng component Button

```
import React from "react";
import { Button } from "@material-ui/core";
const app = () => {
  return (
    <div>
      <h1>Button Demo</h1>
      <Button variant="contained" color="primary">
        Submit
      </Button>
    </div>
  );
};
export default app;
```

<https://cybersoft.edu.vn>

Làm quen với thư viện Material UI

<https://cybersoft.edu.vn> <https://cybersoft.edu.vn>

- ❖ Tiếp theo, ta sẽ tìm hiểu về khái niệm JSS (CSS-in-JS)
 - ❖ Trước giờ, ta style component bằng sass hoặc css, có một số ưu điểm là dễ dùng, quen thuộc.. tuy nhiên nó mắc phải một số vấn đề: không linh hoạt(ví dụ , ta muốn truyền một biến vào để xét chiều rộng cho div) thì làm không được và không có tính đóng gói (encapsulation) trong component.
 - ❖ JSS được sinh ra để giải quyết vấn đề này, xây dựng css như một object javascript.
 - ❖ Các hệ thống lớn hiện tại sử dụng JSS rất nhiều, tất nhiên không ngoại trừ material-ui

Làm quen với thư viện Material UI

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❖ Sử dụng JSS trong Material UI với withStyle
- ❖ Tạo một file style.js để định nghĩa jss cho component, mỗi component sẽ có một file style riêng.
- ❖ Ở đây theme đơn giản là một object để quy định nguyên bộ theme của material (màu, font chữ...)
- ❖ (2) là cách viết nếu muốn gọi lồng cấp, trong class content gọi ra span để chỉnh

```
import { makeStyles } from "@material-ui/core";

const useStyles = makeStyles(theme => ({
  content: {
    backgroundColor: "red",
    color: "#ffffff",
    "& span": {
      fontSize: 15
    }
  },
  title: {
    fontSize: 50
  }
));
export default useStyles;
```

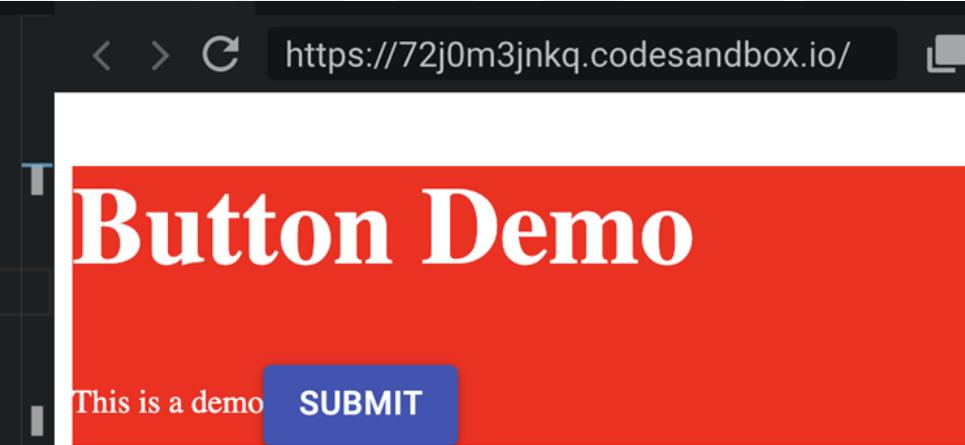
Làm quen với thư viện Material UI

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❖ Ở component , ta sử dụng như sau

```
import React from "react";
import { Button } from "@material-ui/core";
import useStyles from "./style";
const App = () => {
  const classes = useStyles();
  return (
    <div className={classes.content}>
      <h1 className={classes.title}>Button Demo</h1>
      <span>This is a demo</span>
      <Button variant="contained" color="primary">
        Submit
      </Button>
    </div>
  );
}
export default App;
```



Làm quen với thư viện Material UI

<https://cybersoft.edu.vn><https://cybersoft.edu.vn>

- ❖ Tiếp theo , ta sẽ tìm hiểu về Typography trong material-ui, component này cho phép chúng ta thống nhất toàn bộ định dạng text trong trang web

```
App.js  x  JS style.js      JS index.js
1  import React from "react";
2  import { Button, [Typography] } from "@material-ui/core";
3  import useStyles from "./style";
4  const App = () => {
5    const classes = useStyles();
6    return (
7      <div className={classes.content}>
8        <Typography color="textPrimary" variant="h2" component="h1">
9          | Button Demo
10         </Typography>
11        <Typography variant="body1" component="span" color="primary">
12          | This is demo
13         </Typography>
14
15        <Button variant="contained" color="primary">
16          | Submit
17        </Button>
18      </div>
19    );
20  };
21  export default App;
```

Variant: định dạng text

color: màu text

component: tag html sẽ được render ra giao diện

=> toàn bộ định dạng text, màu, đều được định nghĩa sẵn trong theme của material. Ví dụ, nếu variant là h2, thì sẽ có font size bao nhiêu, font-weight bao nhiêu...color="primary" là có màu xanh, còn color="textPrimary" là có màu đen, tất cả những cái này đều định nghĩa sẵn trong theme.

Làm quen với thư viện Material UI

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

- ❖ Trong trường hợp, ta không muốn sử dụng theme của material ui, mà muốn tự custome thì như thế nào ?
- ❖ Tạo ra một file theme để định nghĩa lại, trong đó, palette là màu, typography là định dạng text

❖ 2 thuộc tính fontSize và fontFamily nằm cuối và

riêng biệt có nghĩa là, chỉ cần sử dụng

<Typography> thì text sẽ mặc định

có fontSize: 14 và fontFamily như v

```
import { createMuiTheme } from '@material-ui/core/styles';
User, 2 months ago • UI buyer list + navigation
export const theme = createMuiTheme({
  palette: {
    common: {
      black: 'rgba(47, 46, 46, 1)',
      white: 'rgba(255, 255, 255, 1)'
    },
    background: {
      paper: 'rgba(255, 255, 255, 1)',
      default: 'rgba(251, 251, 251, 1)'
    },
    primary: {
      light: 'rgba(255, 255, 255, 1)',
      main: 'rgba(227, 141, 3, 1)',
      dark: 'rgba(226, 112, 1, 1)',
      contrastText: 'rgba(255, 255, 255, 1)'
    },
    secondary: {
      light: 'rgba(47, 46, 46, 1)',
      main: 'rgba(86, 85, 84, 1)',
      dark: 'rgba(47, 46, 46, 1)',
      contrastText: '#fff'
    },
    text: {
      primary: 'rgba(47, 46, 46, 1)',
      secondary: 'rgba(86, 85, 84, 1)',
      disabled: 'rgba(155, 155, 155, 1)',
      hint: 'rgba(184, 184, 184, 0.38)'
    }
  },
  typography: {
    h5: {
      fontWeight: 600
    },
    h6: {
      fontSize: 18,
      fontWeight: 600
    },
    body1: {
      fontSize: 14,
      fontWeight: 600
    },
    body2: {
      fontWeight: 400
    },
    fontSize: 14,
    fontFamily: 'Open Sans', sans-serif'
  }
});
```

Làm quen với thư viện Material UI

<https://cybersoft.edu.vn>

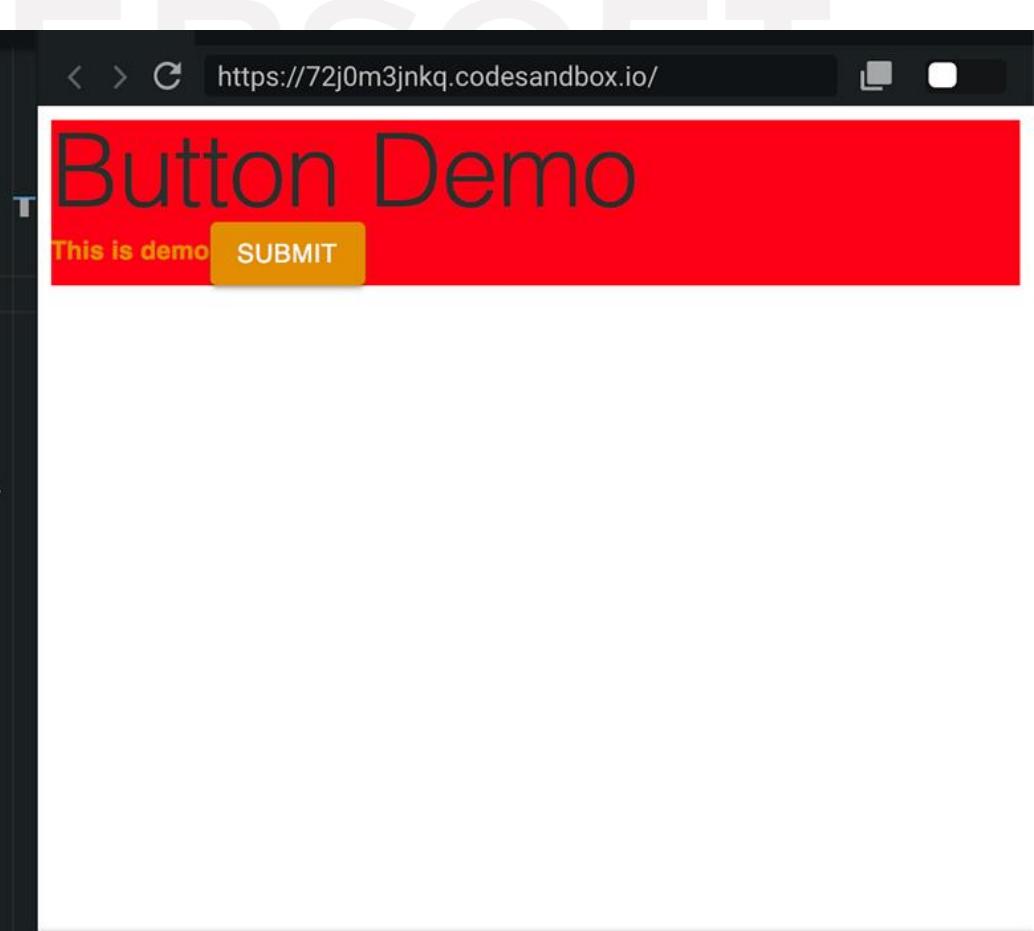
<https://cybersoft.edu.vn>

- ❖ Để bộ theme này có thể sử dụng trong toàn ứng dụng, ta sẽ bọc component lớn nhất lại, trường hợp này là app component

```
import useStyle from "./style";
import { ThemeProvider } from "@material-ui/styles";
import { theme } from "./theme"
const App = () => {
  const classes = useStyle();
  return (
    <ThemeProvider theme={theme}>
      <div className={classes.content}>
        <Typography color="textPrimary" variant="h2" component="h1">
          Button Demo
        </Typography>
        <Typography variant="body1" component="span" color="primary">
          This is demo
        </Typography>

        <Button variant="contained" color="primary">
          Submit
        </Button>
      </div>
    </ThemeProvider>
  );
};

export default App;
```



HOC (High order component)

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>



CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>

<https://cybersoft.edu.vn>