

ReactJs

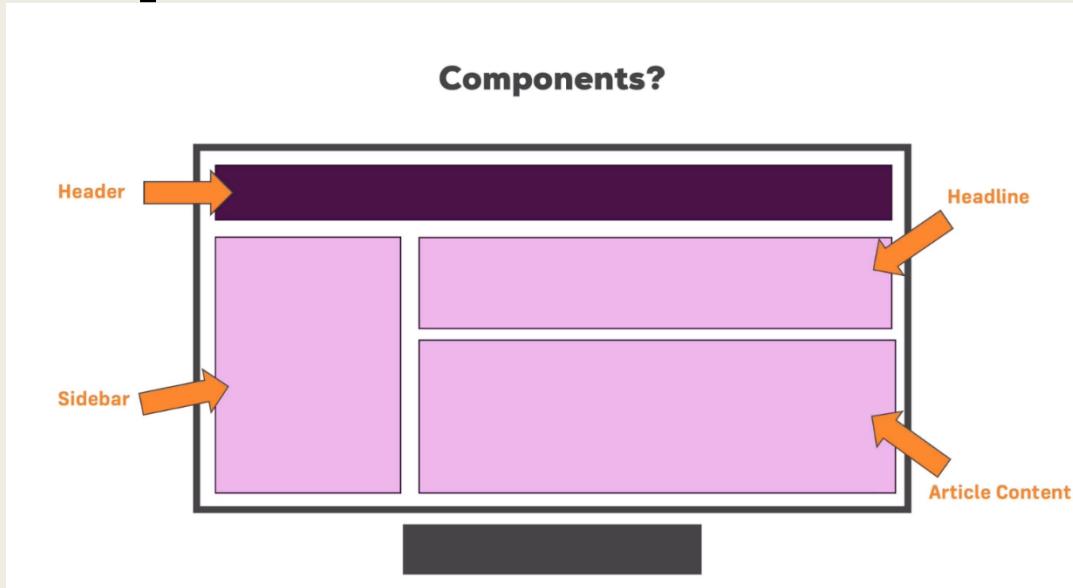
# Tổng quan về reactjs

---

# Giới thiệu về Reactjs

## □ *React là gì?*

- ❖ React là một thư viện javascript dùng để xây dựng giao diện người dùng 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-a

- ❑ *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 with the title bar "Welcome - react - Visual Studio Code". The left sidebar has icons for file operations like "New file", "Open folder...", and "Add workspace folder...". Below it is a "Recent" section with a link to "D:\CYBERSOFT\cybersoft\slideFrontEnd\react". The main area shows a terminal window with the following output:

```
Microsoft Windows [Version 10.0.17134.471]
(c) 2018 Microsoft Corporation. All rights reserved.

d:\CYBERSOFT\cybersoft\slideFrontEnd\react>create-react-app demo

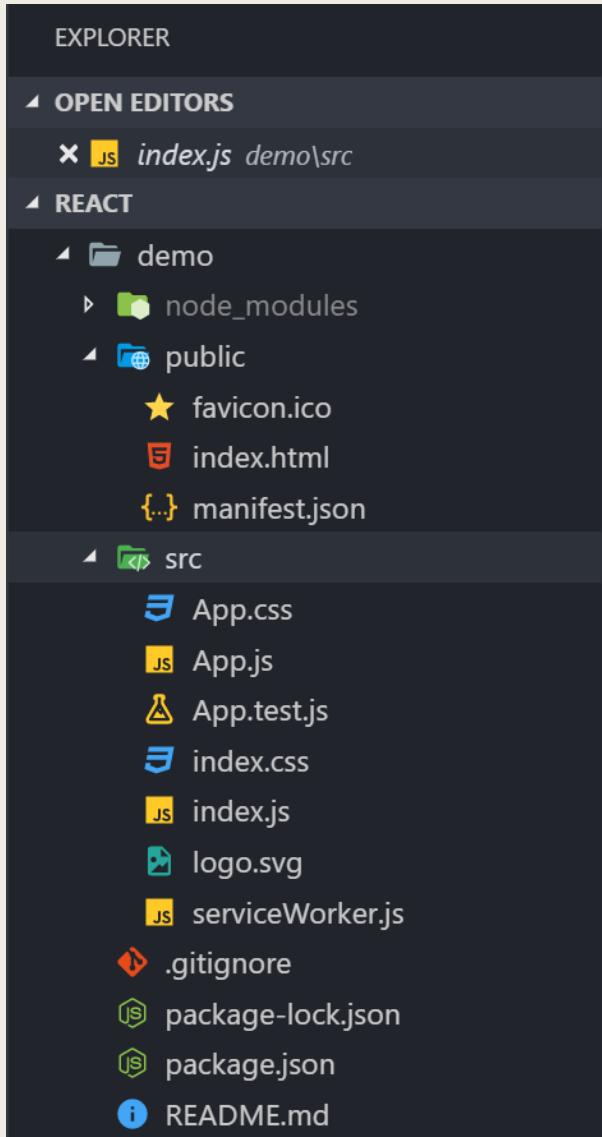
Creating a new React app in d:\CYBERSOFT\cybersoft\slideFrontEnd\react\demo.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...
```

A red box highlights the terminal output: "Installing packages. This might take a couple of minutes. Installing react, react-dom, and react-scripts...". To the right of this box, a red annotation in Vietnamese reads: "create-react-app sẽ tự động cài đặt các package cần thiết để tạo nên react app". The bottom status bar shows a progress bar and the text "[.....] - extract:postcss: sill extract postcss@^7.0.0 extracted to d:\CYBERSOFT\cybersoft\slideFrontEnd\react\demo".

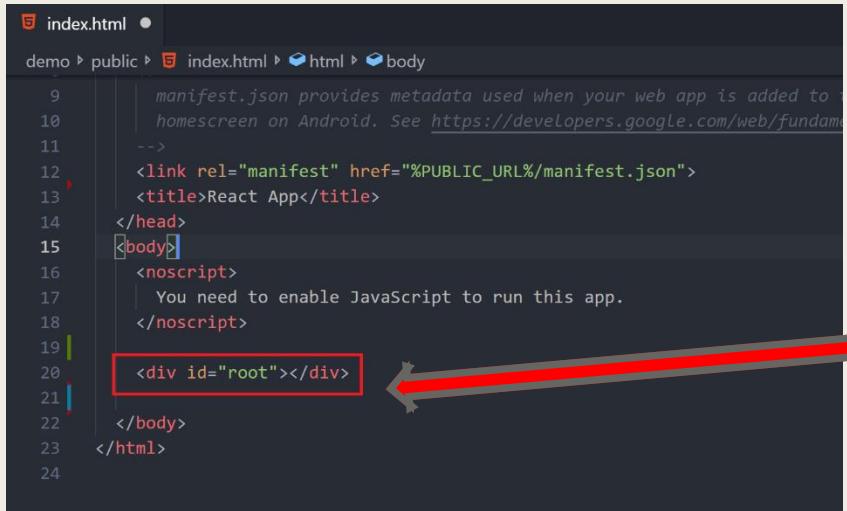
- ❑ Chạy project: **npm start**

# Cấu trúc thư mục



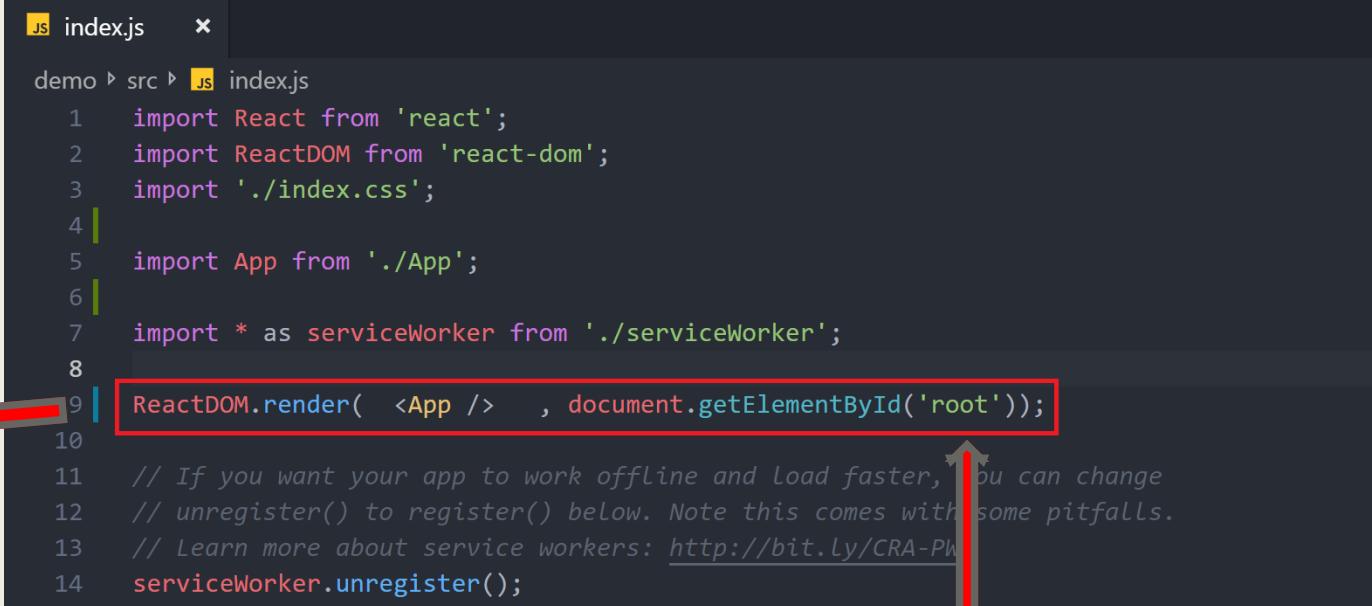
- ❑ ***node\_modules***: chứa các module được cài vào project
- ❑ ***Public***: chứa file index.html và hình ảnh
- ❑ ***Src***: chứa các component của ứng dụng
- ❑ ***Package.json***: lưu lại các thông tin của project và các thư viện được cài vào.

# Luồng đi của ứng dụng



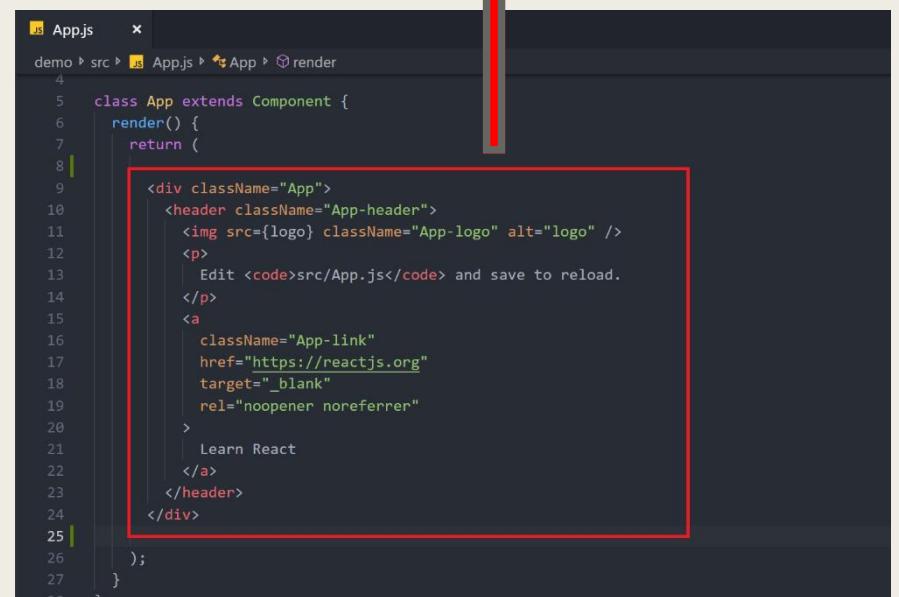
A screenshot of a code editor showing the content of index.html. The file contains an HTML structure with a body element containing a noscript block and a div element with id="root". A red box highlights the div element, and a red arrow points from it to the ReactDOM.render call in index.js.

```
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 |
20 |   <div id="root"></div>
21 |
22 | </body>
23 | </html>
24 |
```



A screenshot of a code editor showing the content of index.js. The file imports React, ReactDOM, and App, and then uses ReactDOM.render to render the App component into the div with id="root". A red box highlights the ReactDOM.render call, and a red arrow points from it to the corresponding div element in index.html.

```
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-PW
14 serviceWorker.unregister();
```



A screenshot of a code editor showing the content of App.js. The file defines a class App extending Component, which has a render method returning an App component. The App component contains an header element with a logo and a link to reactjs.org. A red box highlights the entire App component structure, and a red arrow points from it to the ReactDOM.render call in index.js.

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

- ❑ Đầu tiên, browser sẽ đọc được trang index.html
- ❑ Tiếp theo sẽ đọc tới file index.js, trong file này, ReactDOM được sử dụng để render nội dung của app component ra div root ở ngoài HTML
- ❑ App.js chính là component gốc của toàn ứng dụng

# Component

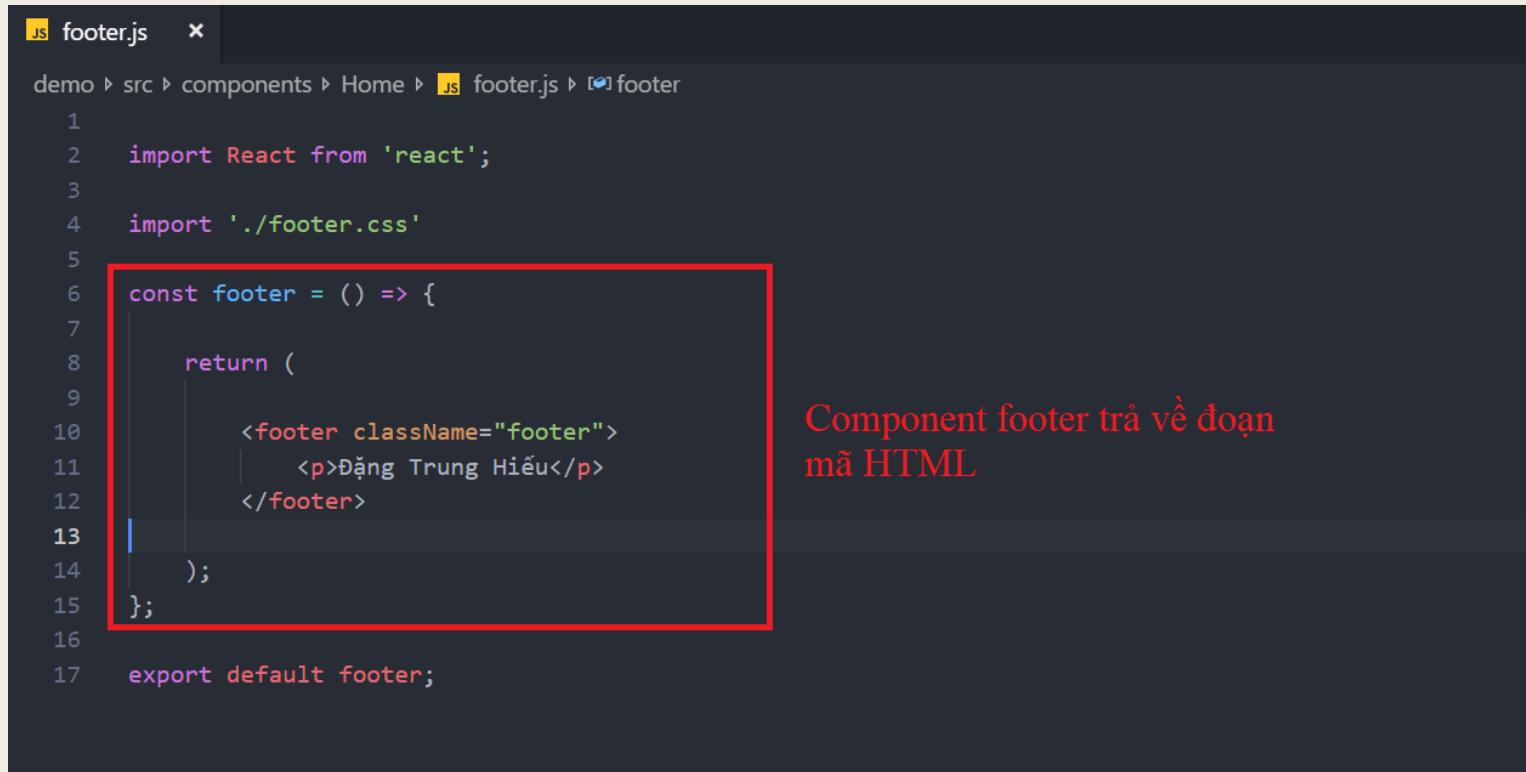
- ❑ Component biểu diễn giao diện UI (file.html).
- ❑ Nói 1 cách đơn giản 1 component là 1 thẻ do mình định nghĩa trong thẻ đó chứa các nội dung html do mình biên soạn.
- ❑ Cấu trúc 1 component bao gồm:
- ❑ Có 2 loại component:
  - ❑ Stateless component (functional component)
  - ❑ Stateful component (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
16                                         tiên và return và một đoạn mã html. Đây chính là
                                         giao diện được hiển thị lên browser
```

The screenshot shows a code editor with a dark theme. An `App.js` file is open. The code defines a class-based component named `App` that extends `Component` from the `react` library. It includes a `render` method that returns a block of placeholder text indicating where the component's content will be rendered. The file also imports CSS from `./App.css`. Annotations with colored boxes explain the code: a red box highlights the import of `react`, a green box highlights the import of `App.css`, a blue box highlights the class definition, and a large blue box highlights the content returned by the `render` method. Red text at the bottom right provides additional context about how components are rendered.

# 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**

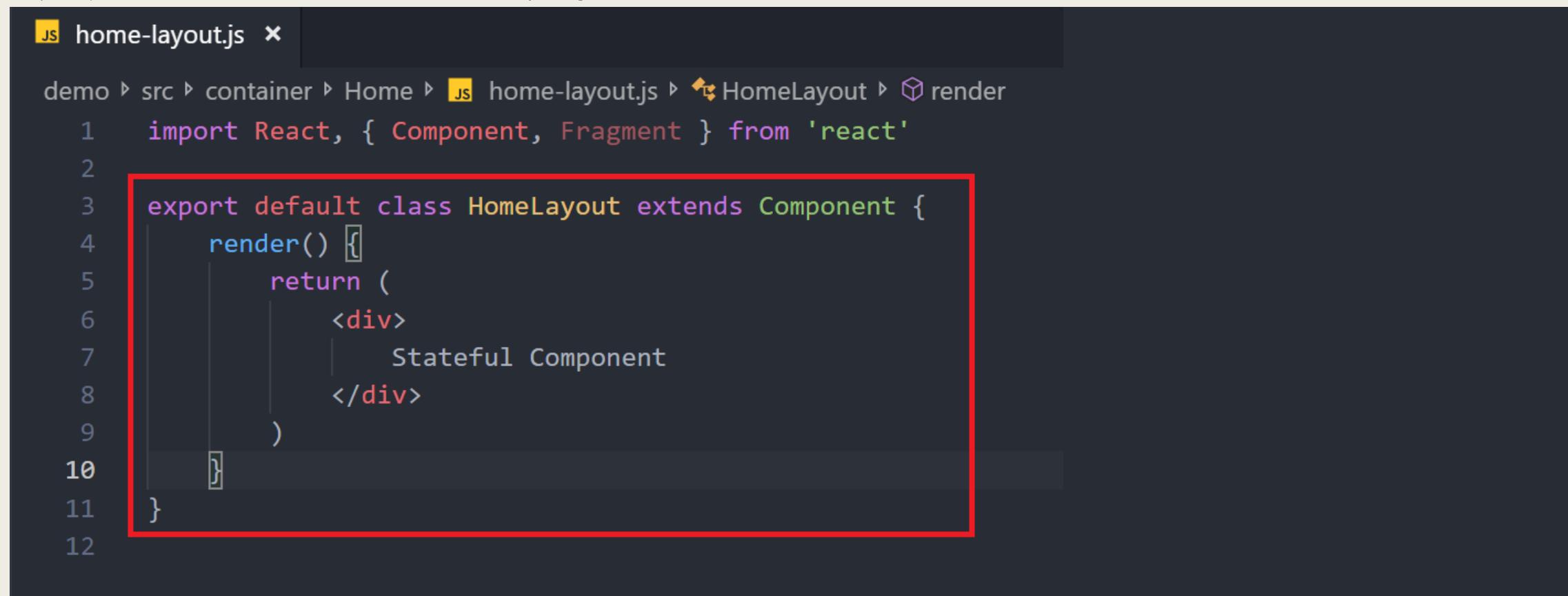


```
js footer.js x
demo › src › components › Home › js footer.js › footer
1
2 import React from 'react';
3
4 import './footer.css'
5
6 const footer = () => {
7
8     return (
9
10         <footer className="footer">
11             |   <p>Đặng Trung Hiếu</p>
12         </footer>
13     );
14 };
15
16 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, thoạt 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: <img />
  - ❑ For => *htmlFor* . Ví dụ <label htmlFor=“dangnhap”></label>



```
App.js x
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}
16
17 export default App;
18
```

# 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 user interface for converting HTML to JSX. On the left, under 'HTML/SVG', there is a code editor containing the following SVG code:

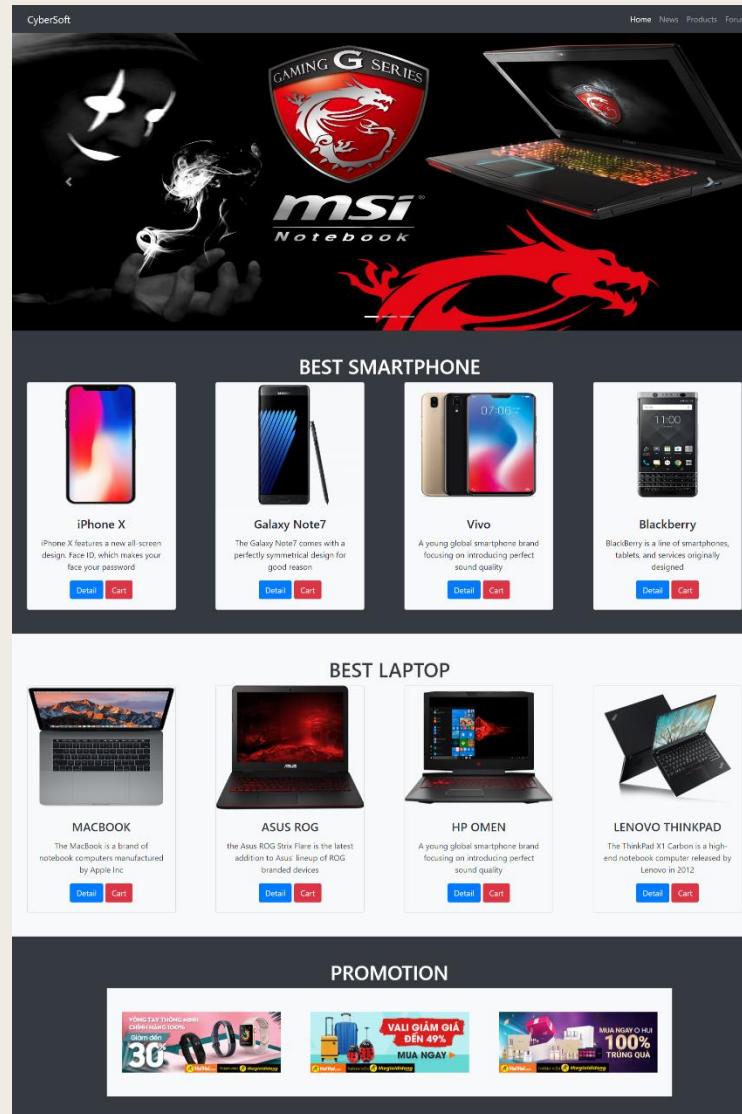
```
1 <svg xmlns="http://www.w3.org/2000/svg" width="200" height="100"
2   <rect width="200" height="100" stroke="black" stroke-width="6"
3     fill="green"/>
</svg>
```

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

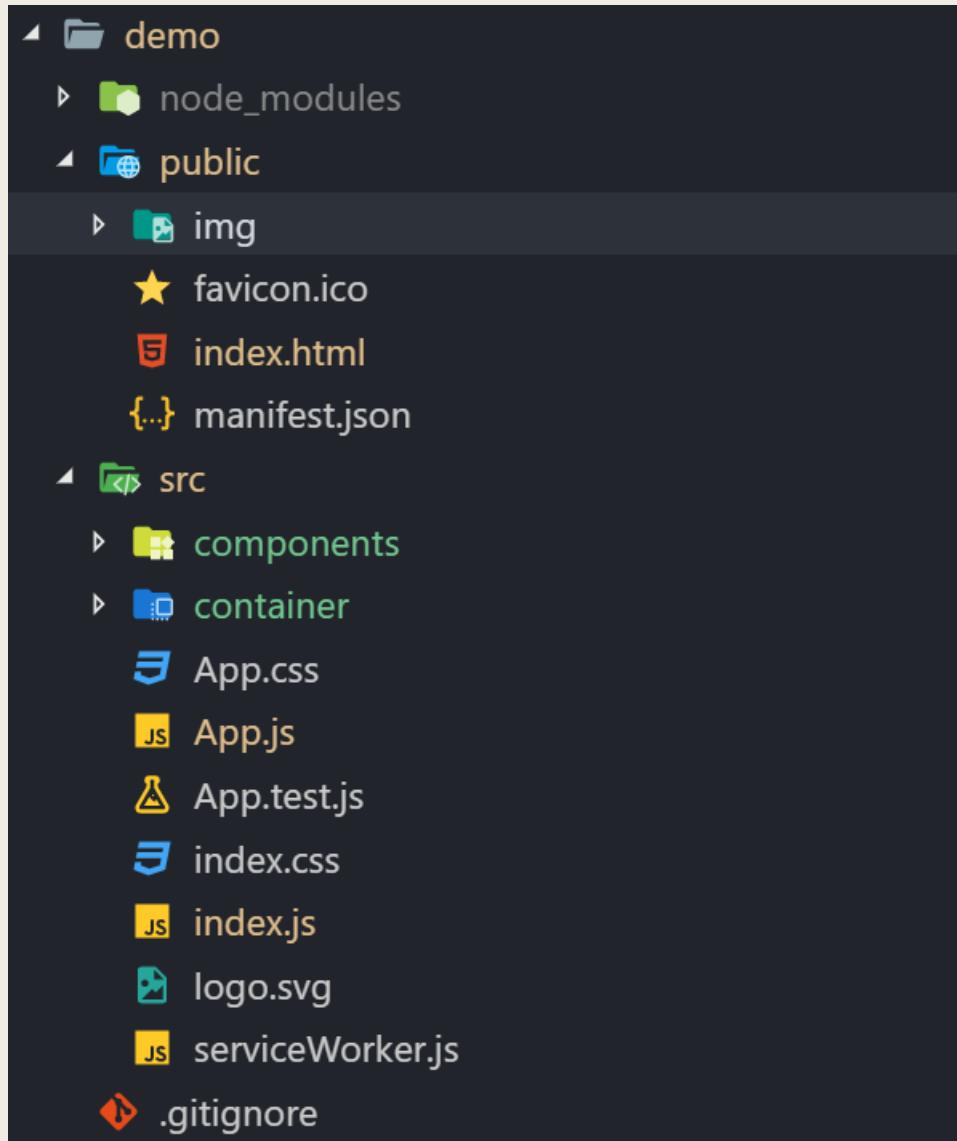
```
;<svg width={200} height={100} version="1.1">
  <rect
    width={200}
    height={100}
    stroke="#000"
    strokeWidth={6}
    fill="green"
  />
</svg>
```

There is a checkbox labeled 'Optimize SVG' and a blue 'Copy' button at the top right. A small circular icon with an upward arrow is located at the bottom center.

# Bài tập: Tổ chức layout sau theo cấu trúc component



# Hướng dẫn : Tổ chức thư mục



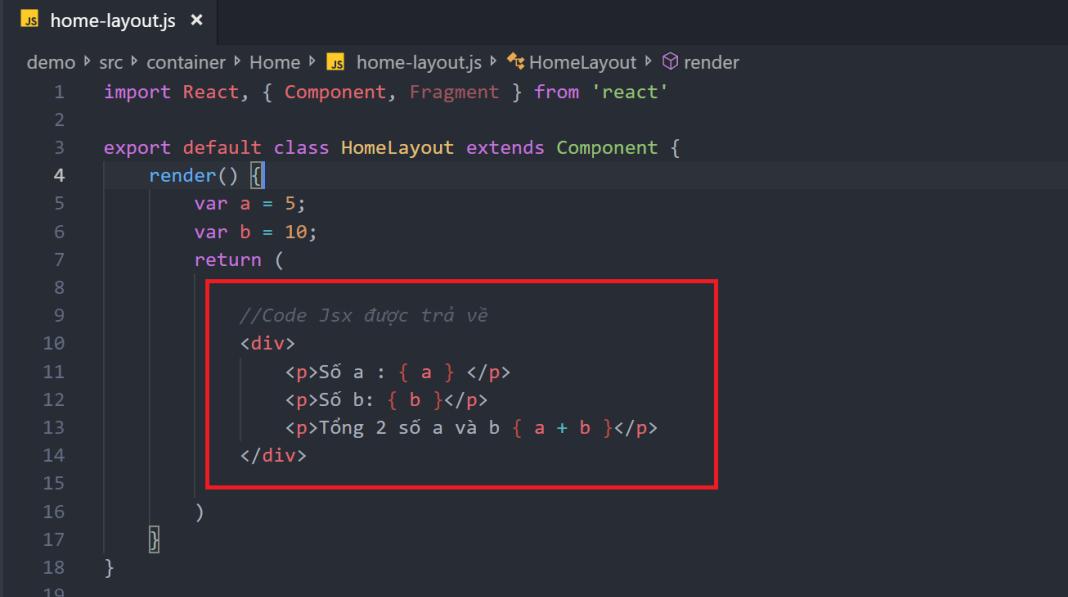
- ❑ *Thư mục hình, ta sẽ bỏ trong folder public*
- ❑ *Ở folder src, ta tạo thêm 2 folder,*
  - ❑ *Components: để chứa các component nhỏ lẻ cấu thành nên 1 trang: header, footer, phim...*
  - ❑ *Containers: chứa các component ở mức độ trang*

# Databinding reactjs

---

# Databinding trong reactjs

- Jsx cho phép ta lồng javascript vào HTML thông qua dấu { a }
- a ở trên có thể là một biến, một hàm, toán tử cộng trừ, so sánh, if else....
- Javascript sẽ được parse và hiển thị ra chung với html



```
home-layout.js x
demo › src › container › Home › home-layout.js › HomeLayout › render
1 import React, { Component, Fragment } from 'react'
2
3 export default class HomeLayout extends Component {
4   render() {
5     var a = 5;
6     var b = 10;
7     return (
8       //Code Jsx được trả về
9       <div>
10         <p>Số a : { a } </p>
11         <p>Số b: { b }</p>
12         <p>Tổng 2 số a và b { a + b }</p>
13       </div>
14     )
15   }
16 }
17
18 }
19
```

# Event binding trong Reactjs

- *Đối với function không có tham số đầu vào:*

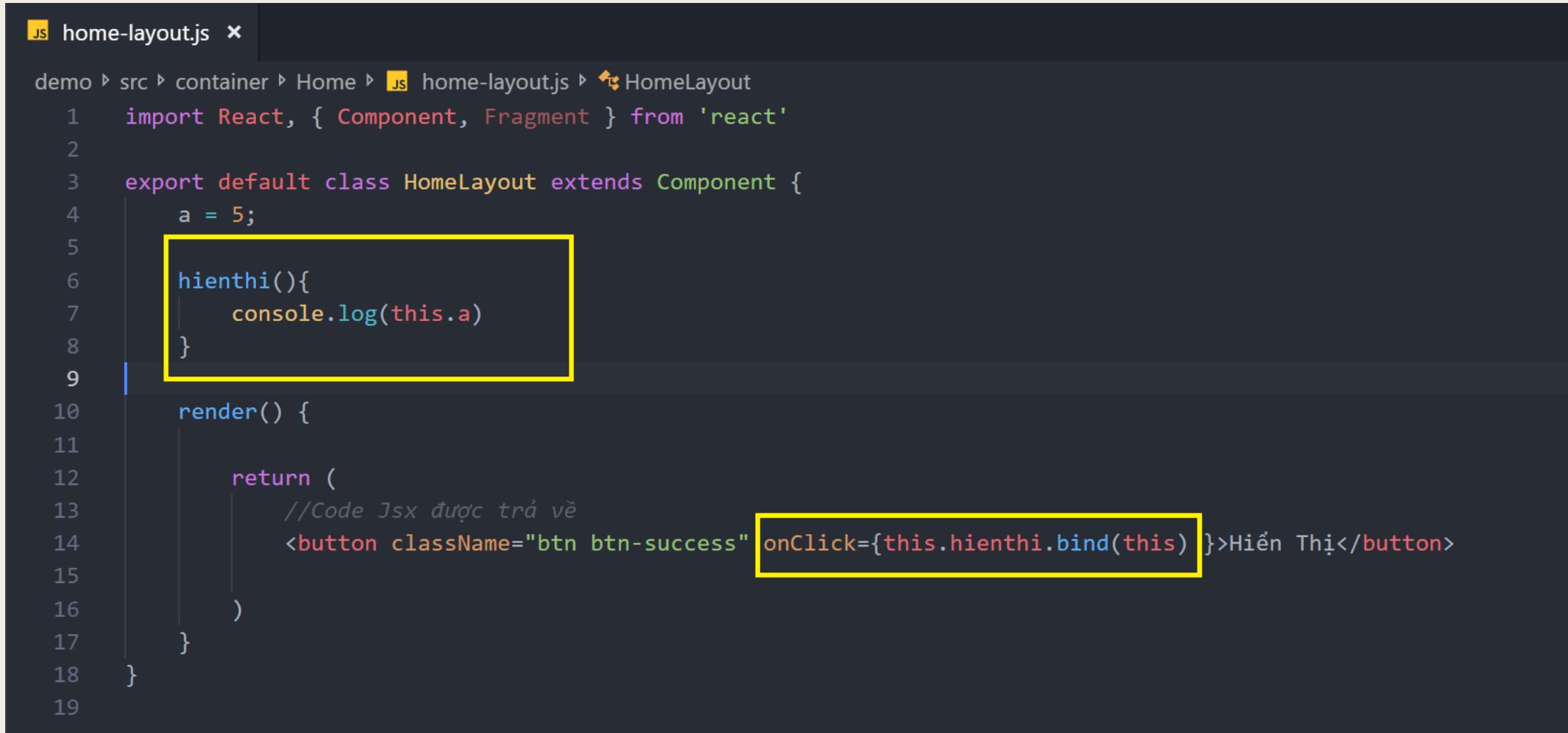


```
JS home-layout.js ●
demo › src › container › Home › JS home-layout.js › ...
1 import React, { Component, Fragment } from 'react'
2
3 export default class HomeLayout extends Component {
4     a = 5;
5
6     hienthi(){
7         console.log('hieu dep trai')
8     }
9
10    render() {
11
12        return (
13            //Code Jsx được trả về
14            <button className="btn btn-success" onClick={this.hienthi}>Hiển Thị</button>
15        )
16    }
17}
18
19
```

- Tên sự kiện: onClick, onBlur, onChange...
- Có từ khóa `this` vì `hienthi` là một phương thức của class
- Không có `cặp ngoặc ()`, vì như v`e` sẽ gọi hàm chạy ngay lập tức khi render

# Event binding trong Reactjs

- Đổi với function có tồn tại từ khóa `this` bên trong:



```
JS home-layout.js x
demo › src › container › Home › JS home-layout.js › HomeLayout
1 import React, { Component, Fragment } from 'react'
2
3 export default class HomeLayout extends Component {
4     a = 5;
5
6     hienthi(){
7         console.log(this.a)
8     }
9
10    render() {
11
12        return (
13            //Code Jsx được trả về
14            <button className="btn btn-success" onClick={this.hienthi.bind(this)}>Hiển Thị</button>
15        )
16    }
17}
18
19
```

# Event binding trong Reactjs

- Đối với function có tham số đầu vào:

The screenshot shows a code editor with a dark theme. The file is named 'home-layout.js'. The code defines a class 'HomeLayout' that extends 'Component'. It contains a method 'hienthi' and a 'render' method. In the 'render' method, there is a button that logs 'name' and 'age' to the console when clicked. A yellow box highlights the 'hienthi' method and another yellow box highlights the button's onClick event handler.

```
JS home-layout.js x
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     a = 5;
5
6     hienthi(name, age) {
7         console.log(name, age)
8     }
9
10    render() {
11
12        return (
13            //Code JSX được trả về
14            <button
15                className="btn btn-success"
16                onClick={this.hienthi.bind(this, 'Dang Trung Hieu', 12)}
17                >Hiển Thị</button>
18
19        )
20    }
21}
22
```

# Directive trong reactjs

---

# Ân hiện phần tử trong jsx

- Trong react không có khái niệm tương đương ngIf, nên ta sử dụng cú pháp của jsx để tự tạo cho mình một ngIf trong reactjs

```
import React, { Component, Fragment } from 'react'
export default class HomeLayout extends Component {
  status = true;
  onToggleText = () => {
    this.status = !this.status;
    console.log(this.status)
  }
  render() {
    return (
      <div>
        <button className="btn btn-success" onClick={this.onToggleText.bind(this)}>Toggle</button>
        <p>
          {this.status === true ? 'Đây là một đoạn text cần được ẩn hiện' : ''}
        </p>
      </div>
    )
  }
}
```

Hàm thay đổi thuộc tính status của class

Đưa ra điều kiện, nếu thuộc tính status là true, thì hiện ra đoạn text, còn nếu không thì hiện ra chuỗi rỗng

- Và kết quả ? Không hề có chuyện gì xảy ra. Vậy lý do tại sao
  - => Hàm render() chỉ chạy một lần duy nhất thì component được khởi tạo, do đó dù ta có thay đổi thuộc tính status nhưng hàm render() không chạy lại thì vẫn vô nghĩa
  - => cần sử dụng một thuộc tính quan trọng của component : **State**

# State là gì ?

- State là một thuộc tính mặc định của class để quản lý trạng thái của component
- Mỗi khi state thay đổi, thì hàm render sẽ được gọi lại

```
import React, { Component, Fragment } from 'react'
export default class HomeLayout extends Component {
  state = {
    status: true
  }
  onToogleText = () => {
    this.setState({
      status : !this.state.status
    })
  }
  render() {
    return (
      <div>
        <button className="btn btn-success" onClick={this.onToogleText.bind(this)}>Toogle</button>
        <p>
          {this.state.status === true ? 'Đây là một đoạn text cần được ẩn hiện' : ''}
        </p>
      </div>
    )
  }
}
```

Tạo ra thuộc tính state để quản lý status  
state là thuộc tính mặc định của class, nên phải viết đúng

Ta không thể trực tiếp thay đổi state, mà phải dùng hàm setState() để làm  
Mỗi khi hàm setState() chạy, thì hàm render() sẽ  
được gọi lại. Một giao diện mới sẽ được  
render() lại ra màn hình

Xét điều kiện ẩn hiện theo this.state.state

# Bài tập: quay lại layout bán hàng, áp dụng state

- ❖ Yêu cầu, ở component phimItem, xây dựng hàm cho nút detail để  
hiển hiện đoạn mô tả

## BEST SMARTPHONE



**iPhone X**

iPhone X features a new all-screen design. Face ID, which makes your face your password

[Detail](#) [Cart](#)



**Galaxy Note7**

The Galaxy Note7 comes with a perfectly symmetrical design for good reason

[Detail](#) [Cart](#)



**Vivo**

A young global smartphone brand focusing on introducing perfect sound quality

[Detail](#) [Cart](#)



**Blackberry**

BlackBerry is a line of smartphones, tablets, and services originally designed

[Detail](#) [Cart](#)

# Làm việc với list component

- ☐ List component, hay quen thuộc hơn là khái niệm ngFor trong Angular
- ☐ Ở react không hỗ trợ chúng ta làm điều này, nên ta sẽ tự viết hàm để lặp component như sau:
  - ❖ Tạo component *SinhVien* và component *DanhSachNhanVien*

## ❖ SinhVien Component

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

export default class sinhvien extends Component {
  render() {
    let {HoTen,Lop ,MaSV} = this.props.sinhVien;

    return (
      <div className="card w-25 m-2 p-3">
        <p className="lead">Tên sinh viên :{HoTen} </p>
        <p className="lead">Mã sinh viên: {MaSV}</p>
        <p className="lead">Lớp:{Lop} </p>
      </div>
    )
  }
}
```

## ❖ DanhSachSinhVien Component

```
import React, { Component, Fragment } from 'react'
import SinhVien from './sinhvien';

export default class DanhSachSinhVien extends Component {
  danhSachSinhVien = [
    {HoTen: 'Dang Trung Hieu',MaSV:'0164',Lop:'VLH1'},
    {HoTen: 'Phan Dinh Phung',MaSV:'0162',Lop:'VLH2'},
    {HoTen: 'Nguyen Tuan Canh',MaSV:'0165',Lop:'VLH3'}
  ]
  renderSinhVien = () => {
    return this.danhSachSinhVien.map((sv,index) => {
      return <SinhVien key={index} sinhVien={sv} />
    })
  }
  render() {
    return (
      <div>
        {this.renderSinhVien()}
      </div>
    )
  }
}
```

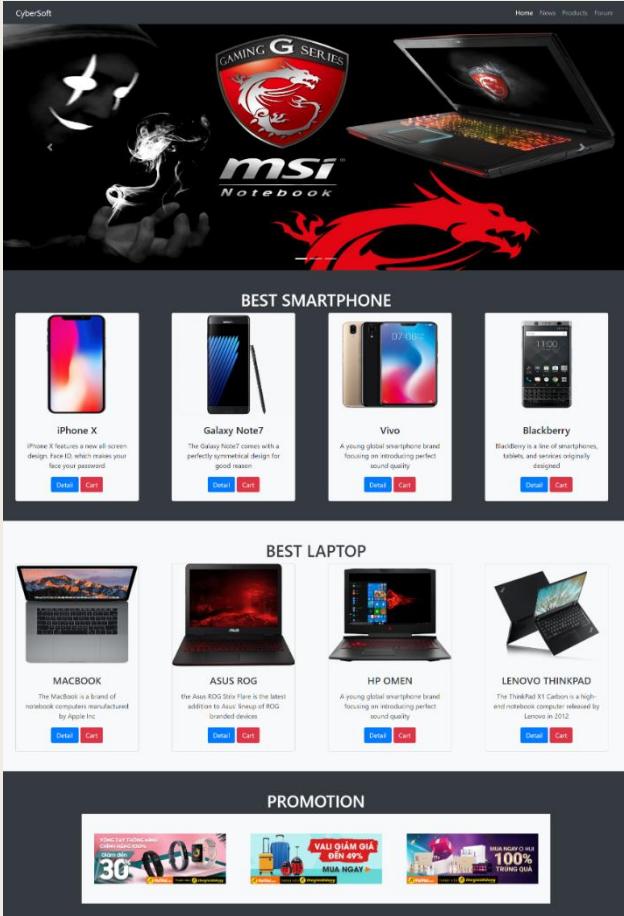
1.Trong *DanhSachSinhVien* component, ta có mảng danhSachSinhVien để tiến hành lặp component *SinhVien*

2.Hàm *renderSinhVien*, ta tiến hành duyệt mảng với hàm map, mỗi lần duyệt lấy ra một đối tượng trong mảng là sv và số chạy là index và trả về 1 component *SinhVien* với key (bắt buộc) khác nhau và truyền đối sv vào *SinhVien* Component thông qua props

3. Ở hàm *render()*, ta gọi hàm *renderSinhVien()*, hàm này sẽ trả về cho ta danh sách component *SinhVien*

# Bài tập: quay lại layout bán hàng, áp dụng list component

- ❖ Tiến hành duyệt theo danh sách điện thoại và laptop đã cho để render ra các component phoneItem và laptopItem tương ứng



## ❖ Danh sách điện thoại:

```
[ { id: 'sp_1', name: 'iPhoneX', price: '30.000.000 VNĐ', screen: 'screen_1', backCamera: 'backCamera_1', frontCamera: 'frontCamera_1', img: './img/sp_iphoneX.png', desc: 'iPhone X features a new all-screen design. Face ID, which makes your face your password' }, { id: 'sp_2', name: 'Note 7', price: '20.000.000 VNĐ', screen: 'screen_2', backCamera: 'backCamera_2', frontCamera: 'frontCamera_2', img: './img/sp_note7.png', desc: 'The Galaxy Note7 comes with a perfectly symmetrical design for good reason' }, { id: 'sp_3', name: 'Vivo', price: '10.000.000 VNĐ', screen: 'screen_3', backCamera: 'backCamera_3', frontCamera: 'frontCamera_3', img: './img/sp_vivo850.png', desc: 'A young global smartphone brand focusing on introducing perfect sound quality' }, { id: 'sp_4', name: 'Blacberry', price: '15.000.000 VNĐ', screen: 'screen_4', backCamera: 'backCamera_4', frontCamera: 'frontCamera_4', img: './img/sp_blackberry.png', desc: 'BlackBerry is a line of smartphones, tablets, and services originally designed' } ]
```

## ❖ Danh sách laptop:

```
[ { id: 'sp_1', name: 'Macbook', price: '30.000.000 VNĐ', img: './img/lt_macbook.png', desc: 'iPhone X features a new all-screen design. Face ID, which makes your face your password' }, { id: 'sp_2', name: 'Asus Rog', price: '20.000.000 VNĐ', img: './img/lt_rog.png', desc: 'The Galaxy Note7 comes with a perfectly symmetrical design for good reason' }, { id: 'sp_3', name: 'HP Book', price: '10.000.000 VNĐ', img: './img/lt_hp.png', desc: 'A young global smartphone brand focusing on introducing perfect sound quality' }, { id: 'sp_4', name: 'Lenovo Thinkpad', price: '15.000.000 VNĐ', img: './img/lt_lenovo.png', desc: 'BlackBerry is a line of smartphones, tablets, and services originally designed' } ]
```

# Truyền dữ liệu trong reactjs

---

# Props (Truyền dữ liệu từ component cha sang con)

- ❑ *Props là thuộc tính mặc định của component để chứa các dữ liệu được truyền vào từ component cha, tương tự như kỹ thuật input ở Angular*
- ❑ *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

## □ **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>
    </footer>
  );
};

export default footer;
```

Truyền props vào dưới dạng tham số

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

Export để có thể gọi ở các component khác

## □ **Homelayout component (cha)**

```
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>
    )
  }
}
```

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

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

# 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 }
```

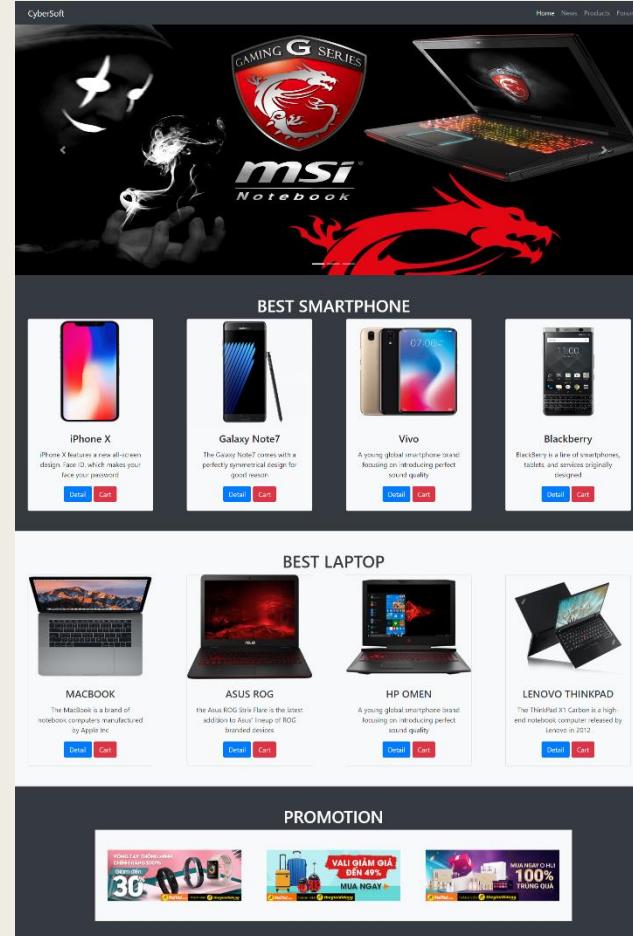
## □ ***Homelayout component (cha)***

```
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



# ❖ Output

- ☐ Ở reactjs không có khái niệm output, nên để truyền dữ liệu component con ra cha, ta phải xây dựng hàm xử lý

## ☐ *SinhVien component (con)*

```
1 import React, { Component } from 'react'
2
3 export default class sinhvien extends Component {
4   name = "Đặng Trung Hiếu";
5   render() {
6     return (
7       <button onClick={this.props.getNameFromSinhVien.bind(this,this.name)}>
8         Truyền status ra danhsachsinhvien component
9       </button>
10    )
11  }
12}
13
```

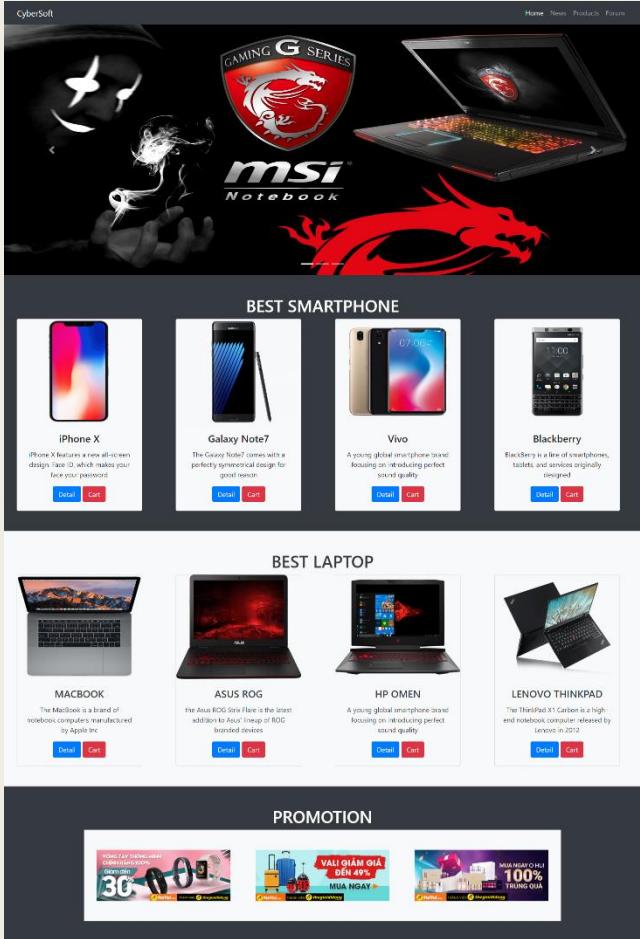
Ở component con, khi click vào button, ta sẽ gọi hàm getNameFromSinhVien được truyền vào từ component cha chạy, truyền vào tham số là thuộc tính name, lúc đó hàm getNameFromSinhVien sẽ chạy hàm getNameFromSinhVien và nhận được tham số name từ con truyền ra

## ☐ *DanhSachSinhVien component (cha)*

```
demo › src › container › JS danhsachsinhvien.js › danhsachsinhvien
1 import React, { Component } from 'react'
2 import Sinhvien from './sinhvien';
3
4 export default class danhsachsinhvien extends Component {
5   name;
6   getNameFromSinhVien(name){
7     this.name = name;
8     console.log(this.name);
9   }
10  render() {
11    return (
12      <Sinhvien getNameFromSinhVien={this.getNameFromSinhVien} />
13    )
14  }
15}
16
```

Ở component cha, ta xây dựng hàm getNameFromSinhVien với tham số là name, sau đó truyền hàm này vào component con dưới dạng props

**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**

- Component modal nằm ở App Component, do đó phải lấy app component làm trung gian, truyền đối tượng phone từ phoneItem ra App, từ app truyền vào modal dưới dạng props
- Để có thể render lại giao diện mỗi lần chọn sản phẩm, ta phải tạo ra state ở app để quản lý đối tượng phone trả ra từ component PhoneItem, mỗi lần component con trả ra đối tượng Phone, setState lại để render ra giao diện mới

# Bài tập: Xây dựng giỏ hàng

Mua Hàng Online

Oppo R1

Samsung Galaxy Note 9

Iphone XS

Cart

Cart

Cart

Chào mừng tới shopping online!

Giỏ Hàng

Hình Ảnh	Sản Phẩm	Giá	Số Lượng	Tổng Cộng
	Iphone XS	600\$	<input type="button" value="-"/> <input type="button" value="+"/>	600\$

Tổng Tiền 600\$

[Thanh Toán](#)

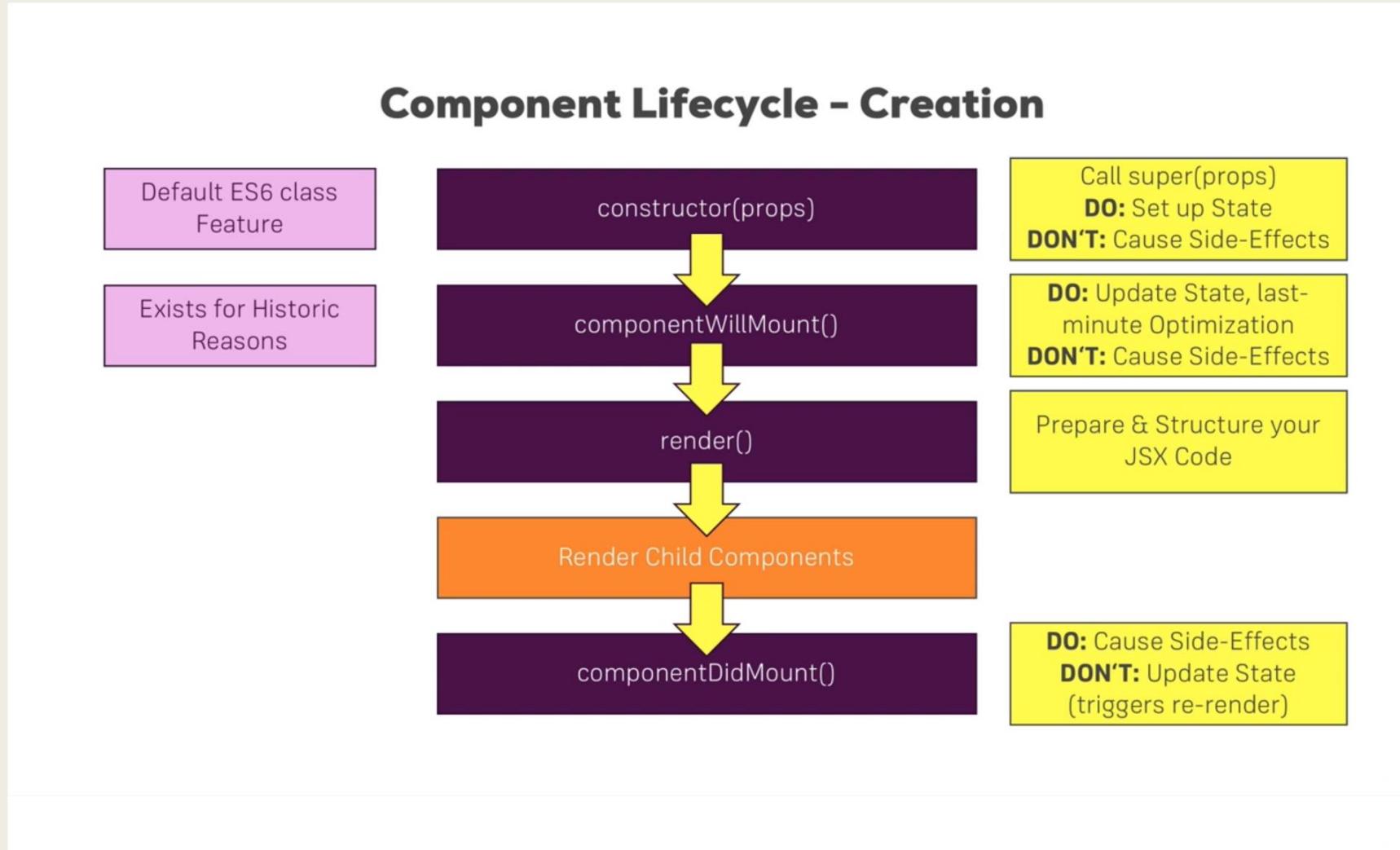
- Nội dung : Xây dựng các tính năng cho giỏ hàng
  - Hiển thị danh sách sản phẩm
  - Put sản phẩm vào giỏ hàng
  - Tăng giảm số lượng sản phẩm trong giỏ hàng
  - Xóa sản phẩm khỏi giỏ hàng
  - Tính tổng tiền từng sản phẩm
  - Tính tổng tiền tất cả sản phẩm
  - Thanh toán giỏ hàng

# Vòng đời của component

---

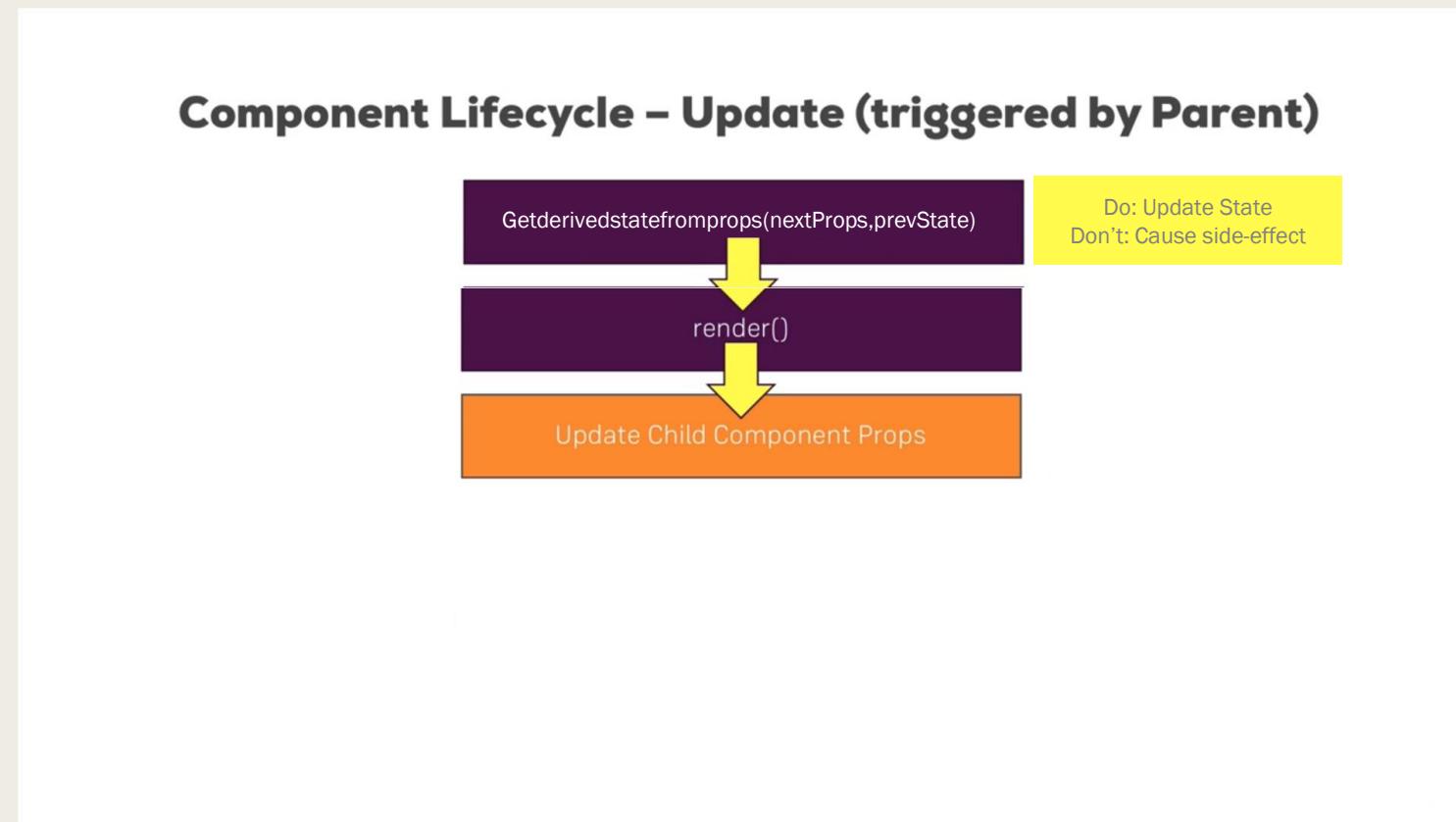
# Vòng đời component - Creation

- ❑ Các phương thức bên dưới sẽ được khởi chạy lần lượt khi component được khởi tạo



# *Vòng đời component - Update*

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



# PureComponent

- ❑ 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

- 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

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

export default class test extends PureComponent {

  render() {

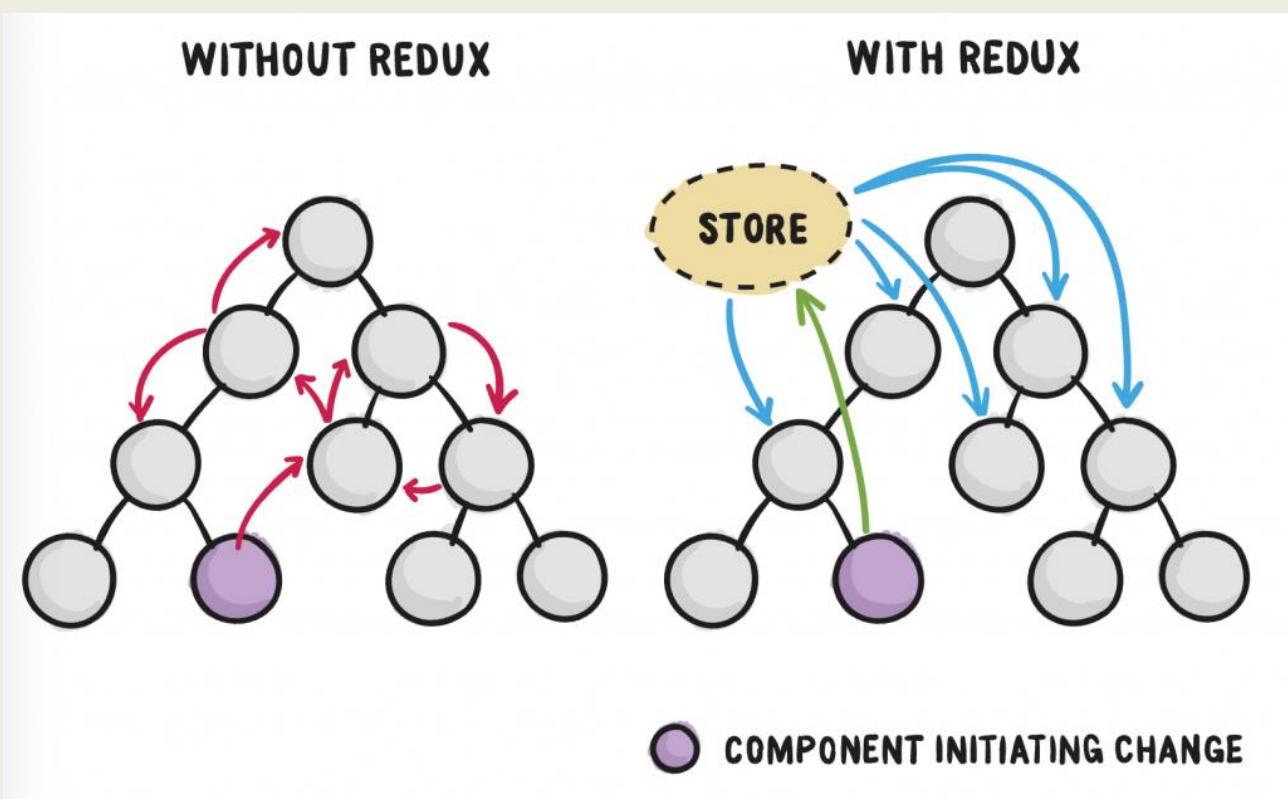
    return (
      <div>
        aaa
      </div>
    )
  }
}
```

# Redux

---

# Giới thiệu về Redux

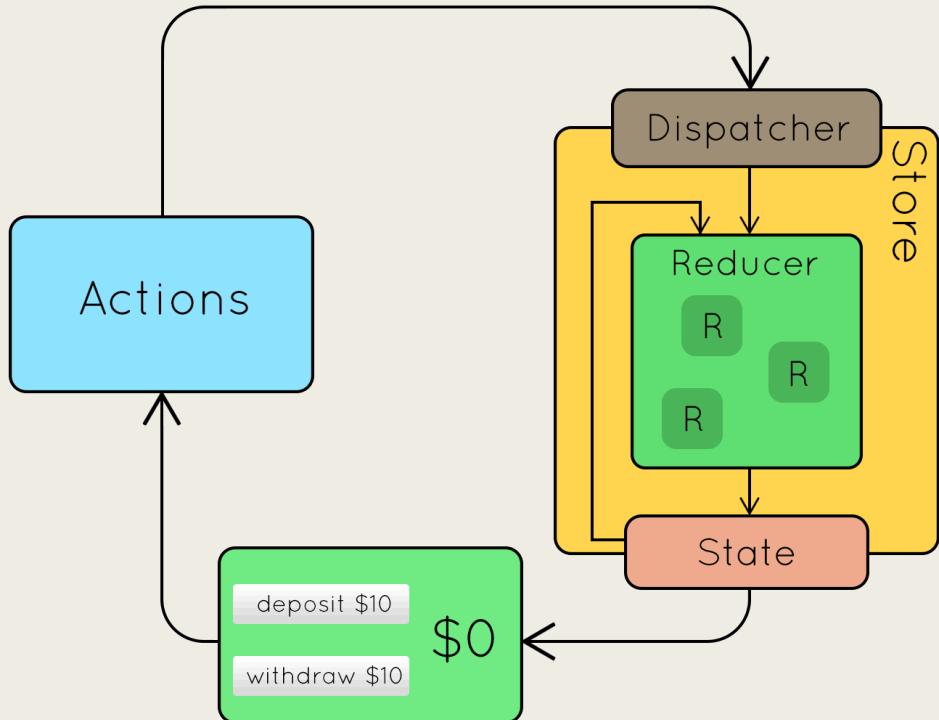
- ❑ Bản chất làm việc với React là việc truyền dữ liệu giữa các component và thay đổi state để re-render lại giao diện component
- ❑ Redux là thư viện cung cấp cho ta một store trung tâm, lưu trữ tất cả các state, từ component muốn thay đổi state chỉ cần truy cập tới store để thay đổi
- ❑ **Ví dụ:**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



# *Cấu trúc của redux*

- ❑ 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)

# Sơ đồ hoạt động của Redux



- ❑ 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

# Bài tập thực hành

- ❑ Thực hành thông qua bài tập quản lý người dùng.
- ❑ Với các chức năng thêm xóa sửa cập nhật người dùng

Quản lý Sinh Viên				
Mã Sinh Viên	Họ Tên	Tuổi	Email	Hành Động
123	Dang Trung Hieu	123	123@gmail.com	<button>Delete</button> <button>Edit</button>
45	Phan Dinh Phung	43	dangtrunghieu147@gmail.com	<button>Delete</button> <button>Edit</button>

# Các bước thực hiện

- ❑ Bước 1 : phân chia component
  - ❑ Tổng cộng có 3 component: DanhSachSinhVien, SinhVien và FormNhap

```
import React, { Component } from 'react'
import Sinhvien from './sinhvien';
export default class DSSV extends Component {
  render() {
    return (
      <div>
        <h1 class="display-4 text-center">Quản lý Người Dùng</h1>
        <div class="container-fluid">
          <button
            class="btn btn-outline-success mb-3"
            data-toggle="modal"
            data-target="#modalForm"
          >Thêm Người Dùng</button>
          <table class="table">
            <thead>
              <tr>
                <th>Mã Sinh Viên</th>
                <th>Họ Tên</th>
                <th>Tuổi</th>
                <th>Email</th>
                <th>Hành Động</th>
              </tr>
            </thead>
            <tbody>
              <Sinhvien />
            </tbody>
          </table>
        </div>
      )
    }
  }
}
```

```
import React, { Component } from 'react'
export default class FormNhap extends Component {
  render() {
    return (
      <div class="modal fade" id="modalForm" tabindex="-1" role="dialog">
        <div class="modal-dialog" role="document">
          <div class="modal-content">
            <h5 class="modal-title">Thêm Người Dùng</h5>
            </div>
            <div class="modal-body">
              <form action="">
                <div class="form-group">
                  <label>Mã SV:</label>
                  <input type="text" class="form-control" />
                </div>
                <div class="form-group">
                  <label>Họ Tên:</label>
                  <input type="text" class="form-control" />
                </div>
                <div class="form-group">
                  <label>Tuổi:</label>
                  <input type="text" class="form-control" />
                </div>
                <div class="form-group">
                  <label>Email:</label>
                  <input type="text" class="form-control" />
                </div>
                <div class="form-group">
                  <button class="btn btn-outline-danger">Thêm Người Dùng</button>
                </div>
              </form>
            </div>
          </div>
        </div>
      )
    }
  }
}
```

```
import React from 'react';
const sinhvien = () => {
  return (
    <tr>
      <td>1</td>
      <td>Đặng Trung Hiếu</td>
      <td>12</td>
      <td>dangtrunghieu147@gmail.com</td>
      <td>
        <button class="btn btn-outline-success mr-2">Delete</button>
        <button class="btn btn-outline-danger">Edit</button>
      </td>
    </tr>
  );
};

export default sinhvien;
```

# Các bước thực hiện

- ❑ Bước 2 : Hiển thị danh sách sinh viên ra app.js để hiển thị ra màn hình.
- ❑ Bước 3: Cài đặt Redux: **npm install redux –save**
- ❑ Bước 4: Cài đặt React-redux (thư viện giúp kết nối ứng dụng react vs store của redux): **npm install react-redux –save**
- ❑ Tại trang gốc index.js, tiến hành khai báo và khởi tạo store

```
import React from 'react';
import ReactDOM from 'react-dom';

import App from './App';
import * as serviceWorker from './serviceWorker';

import { createStore } from 'redux';
import { Provider } from 'react-redux'

const store = createStore();

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();
```

1. Import và khởi tạo store, tham số của hàm createStore  
sẽ là Root Reducer, lát nữa sẽ tạo và bổ sung sau
2. Import và sử dụng Component Provider để kết nối  
ứng dụng react của mình với store của redux

# Các bước thực hiện

## ❑ Bước 4 : Khởi tạo reducer

The screenshot shows the VS Code interface with the following details:

- OPEN EDITORS:** Shows files like App.js, dssvjsx, form-nhap, index.js (src), index.js (src\redux\reducers), and sinhvienjsx.
- SHOPPING-CART:** Shows the project structure including node\_modules, public, favicon.ico, index.html, manifest.json, SRC, components, and redux.
- File Explorer:** Shows the same structure as the sidebar.
- Content Area:** Displays the code for index.js in the redux/reducers folder:

```
1 import { combineReducers } from 'redux';
2
3 const rootReducer = combineReducers(
4
5   {
6     // State mà store đang
7     // lưu trữ
8   }
9 )
10
11 export default rootReducer;
```

A yellow box highlights the state definition block: `{ State mà store đang  
lưu trữ }`.

**Cáu trúc thư mục, trong đó index.js  
chính là reducer root**

## ❑ Bước 5: Quay lại trang index.js gốc, tiến hành import và truyền reducer vào store.

The screenshot shows the VS Code interface with the following details:

- OPEN EDITORS:** Shows files like index.js, App.js, dssvjsx, form-nhap, index.js (src), index.js (src\redux\reducers), and sinhvienjsx.
- SHOPPING-CART:** Shows the project structure including node\_modules, public, favicon.ico, index.html, manifest.json, SRC, components, and redux.
- File Explorer:** Shows the same structure as the sidebar.
- Content Area:** Displays the code for index.js (the main entry point):

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3
4 import App from './App';
5 import * as serviceWorker from './serviceWorker';
6
7 import { createStore } from 'redux';
8
9 import { Provider } from 'react-redux';
10
11 import rootReducer from '../redux/reducers/index';
12
13 const store = createStore(rootReducer);
14
15 ReactDOM.render(
16   <Provider store={store}>
17     <App />
18   </Provider>
19   ,
20   document.getElementById('root')
21 );
22
23 // If you want your app to work offline and load faster, you can change
24 // unregister() to register() below. Note this comes with some pitfalls.
25 // Learn more about service workers: http://bit.ly/CRA-PWA
26 serviceWorker.unregister();
```

A yellow box highlights the import statement: `import rootReducer from '../redux/reducers/index';`

# *React Form*

## *Các bước xử lý form trong ReactJs*

- ❑ Bước 1: Tạo state, với các thuộc tính tương ứng với đối tượng

```
2
3   class formNhap extends Component {
4
5     constructor(props) {
6       super(props);
7       this.state = {
8         MaSV: '',
9         HoTen: '',
10        Tuoi: 0,
11        Email: ''
12      }
13    }
14  }
```

# React Form

## Các bước xử lý form trong ReactJs

- ❑ Bước 2: đặt cho các ô input trong form thuộc tính name tương ứng với thuộc tính của state và sự kiện onChange(khi nội dung input thay đổi thì hàm chạy)

```
<div className="modal-body">
  <form onSubmit={this.handleSubmit}>
    <div className="form-group">
      <label>Mã SV: </label>
      <input
        type="text" className="form-control" name="MaSV" onChange={this.handleChange}>
    </div>
    <div className="form-group">
      <label>Họ Tên: </label>
      <input
        type="text" className="form-control" name="HoTen" onChange={this.handleChange}>
    </div>
    <div className="form-group">
      <label>Tuổi: </label>
      <input
        type="text" className="form-control" name="Tuoi" onChange={this.handleChange}>
    </div>
    <div className="form-group">
      <label>Email: </label>
      <input
        type="email" className="form-control" name="Email" onChange={this.handleChange}>
    </div>
    <div className="form-group">
      <button className="btn btn-outline-danger">Thêm Người Dùng</button>
    </div>
  </form>
</div>
```

# React Form

## Các bước xử lý form trong ReactJs

### ❑ Bước 3:Xây dựng phương thức **handleOnchange**

- ❖ Phương thức nhận vào một tham số mặc định, ở đây đặt tên là evt, trong có có đối tượng target chứa các thông tin của input
- ❖ Ở đây, mỗi lần input, ta sẽ cập nhật giá trị input đó cho state tương ứng
- ❖ Ví dụ, khi ta thay đổi input có name= “HoTen”. Thì hàm handleChange sẽ cập nhật lại thuộc tính HoTen trong state, ở đây , evt.target.name chính là tên thuộc tính (name của input), evt.target.value là giá trị của input

```
handleChange = (evt) => {  
    this.setState({  
        [evt.target.name]: evt.target.value  
    })  
}
```

# React Form

## Các bước xử lý form trong ReactJs

- ❑ Bước 4: Gắn sự kiện **onSubmit** cho form và xây dựng phương thức **handleOnSubmit**
  - ❑ Trong đó, **evt.preventDefault()** là để ngăn sự kiện click submit reload lại trang
  - ❑ **this.state** lúc này sẽ là đối tượng , chứa thông tin người dùng nhập vào, vì khi ta thay đổi input, phương thức **handleOnchange** đã chạy là cập nhật lại state rồi, nên khi ta submit, state đã có dc thông tin người dùng nhập vào

```
<form onSubmit={this.handleOnSubmit}>  
  <div className="form-group">  
    <label>Mã SV: </label>  
    <input  
      type="text" className="form-control" name="MaSV" onChange={this.handleChange}>  
  </div>  
  <div className="form-group">  
    <label>Họ Tên: </label>  
    <input  
      type="text" className="form-control" name="HoTen" onChange={this.handleChange}>  
  </div>
```

```
handleOnSubmit = (evt) => {  
  evt.preventDefault();  
  console.log(this.state);  
}
```

# Xây dựng chức năng Thêm Sinh Viên

## Các bước thực hiện

- ❖ Bước 1: xử lý form và lấy được đối tượng user nhập vào (như cũ)
- ❖ Bước 2: khai báo để store lưu trữ DSSV vào state
  - ❖ Bước 2.1: đầu tiên, ta cần có 1 child reducer để xử lý việc thay đổi thuộc tính DSSV. Tiến hành tạo **sinhVienReducer**
    - ❖ **Thực chất 1 reducer chỉ là một function, nhận vào 2 tham số, giá trị ban đầu của thuộc tính (ở đây DSSV ban đầu chưa có gì nên sẽ là mảng rỗng) và action từ component gửi lên để thay đổi DSSV**

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar titled 'OPEN EDITORS' containing files like 'form-nhap.jsx', 'index.js', 'sinhvien.js', 'dssv.jsx', and 'sinhvien.jsx'. Below it is a 'SHOPPING-CART' section with 'components', 'cart', 'CRUD' (containing 'dssv.jsx', 'form-nhap.jsx', and 'sinhvien.jsx'), 'products', 'redux', and 'reducers' (containing 'index.js' and 'sinhvien.js'). The main area shows the content of 'sinhvien.js':

```
src > redux > reducers > js sinhvien.js > ...
1
2 let DSSV = [];
3
4 const sinhVienReducer = (state = DSSV, action) => {
5
6 }
7 export default sinhVienReducer;
8
9
10
11
12
```

A yellow rectangular box highlights the first two lines of code: 'let DSSV = [];'.

# Xây dựng chức năng Thêm Sinh Viên

## ❑ Các bước thực hiện

- ❖ Bước 1: xử lý form và lấy được đối tượng user nhập vào (như cũ)
- ❖ Bước 2: khai báo để store lưu trữ DSSV vào state
  - ❖ Bước 2.1: đầu tiên, ta cần có 1 child reducer để xử lý việc thay đổi thuộc tính DSSV. Tiến hành tạo *sinhVienReducer*
    - ❖ *Thực chất 1 reducer chỉ là một function, nhận vào 2 tham số, giá trị ban đầu của thuộc tính (ở đây DSSV ban đầu chưa có gì nên sẽ là mảng rỗng) và action từ component gửi lên, tiến hành thay đổi DSSV và trả ra giá trị mới*

The screenshot shows a code editor interface with a dark theme. On the left, there is a sidebar titled "OPEN EDITORS" showing several files: "form-nhap.jsx", "index.js", "sinhvien.js" (which is the active file), "dssv.jsx", and "sinhvien.jsx". Below this is a "SHOPPING-CART" section with a "redux" folder containing "index.js" and "reducers" folder containing "sinhvien.js". The main workspace shows the content of "sinhvien.js":

```
src > redux > reducers > js sinhvien.js > ...
1 let DSSV = [];
2
3 const sinhVienReducer = (state = DSSV, action) => {
4
5 }
6 export default sinhVienReducer;
```

The code is highlighted with a yellow rectangle around the first six lines. The file path "src > redux > reducers > js sinhvien.js" is also visible at the top of the editor.

# Xây dựng chức năng Thêm Sinh Viên

- ❖ Bước 2: khai báo để store lưu trữ DSSV vào state
  - ❖ Bước 2.2: tống hợp lại ở root reducer, ở đây store đang lưu trữ state với DSSV trong đó, và DSSV do sinhVienReducer chịu trách nhiệm, nhận action, xử lý và trả ra giá trị mới mỗi khi có thay đổi

The screenshot shows a code editor interface with the following details:

- OPEN EDITORS:** 1 UNSAVED
- SHOPPING-CART:** A folder structure:
  - components
  - cart
  - CRUD
    - dssv.jsx
    - form-nhap.jsx
    - sinhvien.jsx
  - products
  - redux
  - reducers
    - index.js
    - sinhvien.js
- src/redux/reducers/index.js:** This file contains the following code:

```
1 import { combineReducers } from 'redux';
2
3 import SinhVienReducer from './sinhvien';
4
5 const rootReducer = combineReducers(
6   {
7     DSSV: SinhVienReducer
8   }
9 )
10
11 )
12 export default rootReducer;
```

# Xây dựng chức năng Thêm Sinh Viên

- ❖ Bước 3: Từ component, tiến hành dispatch action lên để thêm sinh viên mới vào danh sách
  - ❖ Ở component, ta cần import {connect} from ‘react-redux’ để tiến hành kết nối component và store

The screenshot shows a code editor with a file tree on the left and a code editor window on the right.

**File Tree:**

- index.html
- manifest.json
- src
  - components
    - cart
    - CRUD
      - dssv.jsx
      - form-nhap.jsx** (highlighted)
      - sinhvien.jsx
  - products
  - redux
    - reducers
      - index.js
      - sinhvien.js

```
82 const mapDispatchToProps = (dispatch) => {
83   return {
84     onAddSinhVien: (user) => {
85       dispatch(
86         {
87           type: 'ADD_USER',
88           user
89         }
90       )
91     }
92   }
93 }
94 }
95
96 export default connect(null, mapDispatchToProps)(formNhap);
```

- ❖ connect nhận vào 2 tham số, và ở đây tham số thứ 2 là mapDispatchToProps, giúp ta chuyển đổi phương thức dispatch thành một props của component, ở đây component FormNhập sẽ có thêm một props là onAddSinhVien (là 1 hàm) và khi hàm này chạy sẽ tiến hành dispatch một action lên store để xử lý

# Xây dựng chức năng Thêm Sinh Viên

- ❖ Bước 3: Từ component, tiến hành dispatch action lên để thêm sinh viên mới vào danh sách
  - ❖ Ở đây action là 1 object có thuộc tính type (bắt buộc) là một chuỗi string để miêu tả công việc cần phải làm
  - ❖ Ở đây action này muốn thêm một sinh viên vào DSSV trên store ,nên cần phải gửi lên đối tượng sinh viên muốn thêm , sẽ được vào như tham số khi ta gọi hàm onAddUser chạy
- ❖ Bước 4: khi nhấn nút submit

```
handleOnSubmit = (evt) => {
  evt.preventDefault();

  this.props.onAddSinhVien(this.state);

}
```

# Xây dựng chức năng Thêm Sinh Viên

- ❖ Bước 5: Xây dựng sinhVienReducer xử lý action thêm sinh viên
  - ❖ Ở reducer, ta sẽ kiểm tra type của action gửi lên là gì, nếu là 'ADD\_USER' tức làm thêm sinh viên, khi đó ta sẽ xử lý, thêm sinh viên mới vào DSSV và trả ra DSSV mới

The screenshot shows a code editor with a sidebar displaying a file tree. The tree includes files like index.js, sinhvien.js, dssv.jsx, and sinhvien.jsx under the SHOPPING-CART folder, and products, redux, and reducers subfolders. The main editor area contains the following code:

```
1 // ...
2 let DSSV = [];
3
4 const sinhVienReducer = (state = DSSV, action) => [
5   switch (action.type) {
6     case 'ADD_USER':
7       const user = action.user;
8       const updateState = [...state, user];
9       return updateState;
10    default: return [...state];
11  }
12]
13
14 export default sinhVienReducer;
```

# Xây dựng chức năng Thêm Sinh Viên

- ❖ Bước 6: Tại component DanhSachSinhVien, ta sẽ truy cập lên store để lấy DSSV từ trên đó xuống và render ra giao diện
  - ❖ Ta xây dựng phương thức mapStateToProps như là tham số thứ nhất của hàm connect
  - ❖ Phương thức này cho phép chúng ta lấy DSSV từ trên store xuống và chuyển nó thành props để sử dụng
  - ❖ Lúc này component DanhSachSinhVien sẽ có thêm một thuộc tính props là DSSV chính là DSSV đang lưu trên store, ta có thể sử dụng

```
40
41  const mapStateToProps = (state) => {
42    return {
43      DSSV: state.DSSV
44    }
45  }
46  export default connect(mapStateToProps,null)(dssv);
```

# Xây dựng chức năng Thêm Sinh Viên

- ❖ Bước 7: sử dụng dữ liệu lấy từ store, render ra component SinhVien dựa theo danh sách lấy từ store

```
OPEN EDITORS
  ✘ dssv.jsx src\components\CRUD
  ✘ form-nhap.jsx src\components\CRUD
SHOPPING-CART
  node_modules
  public
  src
    components
    cart
    CRUD
      dssv.jsx
      form-nhap.jsx
      sinhvien.jsx
    products
    redux
      App.js
      index.js
      serviceWorker.js
    .gitignore
    package-lock.json
    package.json
    README.md

  1 import React, { Component } from 'react'
  2 import Sinhvien from './sinhvien';
  3 import { connect } from 'react-redux';
  4 class dssv extends Component {
  5   renderSinhVien = () => {
  6     return this.props.DSSV.map((sv,i) => {
  7       return <Sinhvien sv={sv} key={i} />
  8     })
  9   }
 10   render() {
 11
 12     return (
 13       <div>
 14         <h1 className="display-4 text-center">Quản lý Sinh Viên</h1>
 15         <div className="container-fluid">
 16           <button
 17             className="btn btn-outline-success mb-3"
 18             data-toggle="modal"
 19             data-target="#modalForm"
 20           >Thêm Người Dùng</button>
 21           <table className="table">
 22             <thead>
 23               <tr>
 24                 <th>Mã Sinh Viên</th>
 25                 <th>Họ Tên</th>
 26                 <th>Tuổi</th>
 27                 <th>Email</th>
 28                 <th>Hành Động</th>
 29               </tr>
 30             </thead>
 31             <tbody>
 32               <tr>
 33                 {this.renderSinhVien()}
 34               </tr>
 35             </tbody>
 36           </table>
 37         </div>
 38       </div>
 39     )
  }
```

# Xây dựng chức năng Xóa Sinh Viên

Yêu Cầu: khi click vào nút “xóa” sẽ tiến hành xóa sinh viên ở dòng đó

Quản lý Sinh Viên				
Mã Sinh Viên	Họ Tên	Tuổi	Email	Hành Động
123	Dang Trung Hieu	123	123@gmail.com	<button>Delete</button> <button>Edit</button>
45	Phan Dinh Phung	43	dangtrunghieu147@gmail.com	<button>Delete</button> <button>Edit</button>

# Xây dựng chức năng Xóa Sinh Viên

Bước 1: khi click vào nút “xóa”, lấy được mã sinh viên của sinh viên dòng hiện tại

Bước 2: sau khi lấy được mã sinh viên, tiến hành dispatch 1 action lên store để tiến hành xóa người dùng trong danh sách (vì danh sách đang được lưu tại store)

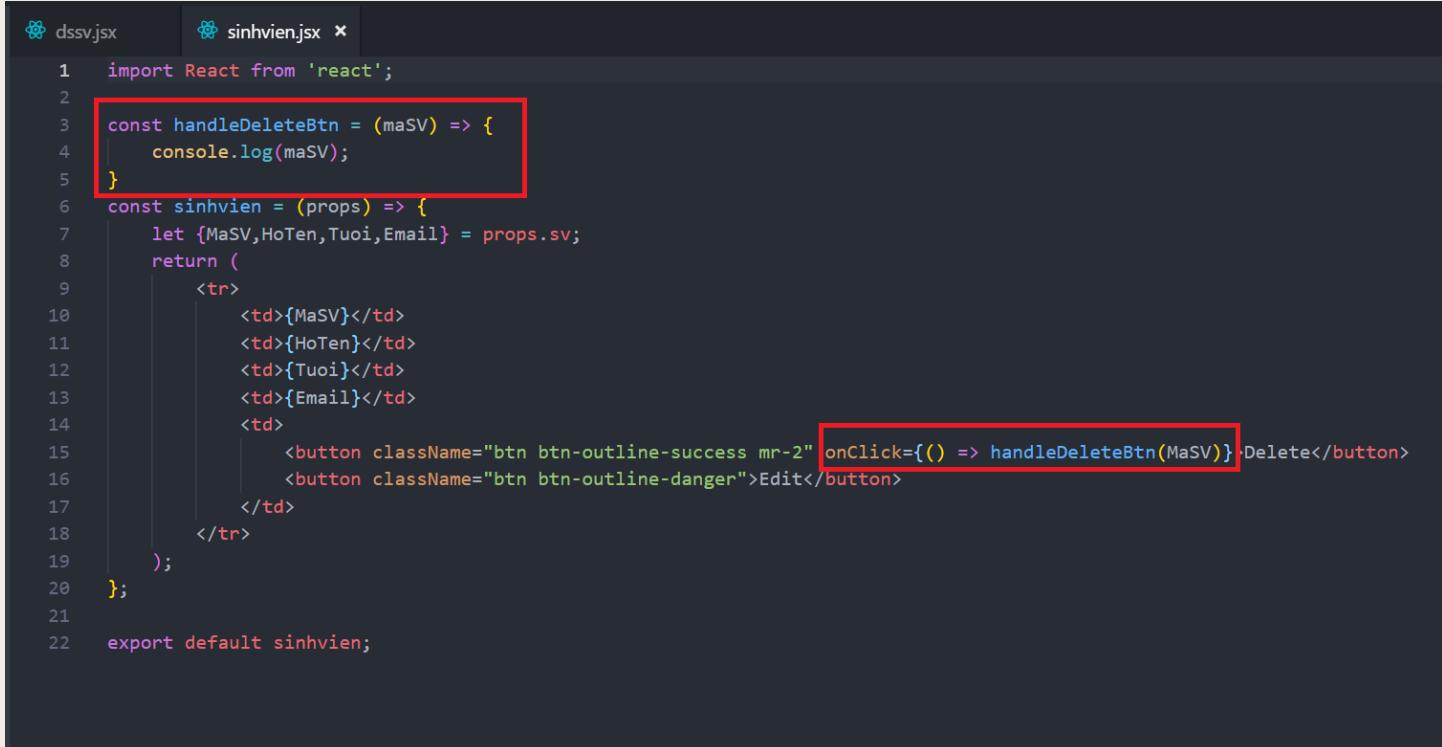
Bước 3: muốn xóa thì ta phải xóa trong DSSV, mà DSSV thì do SinhVienReducer quản lý, nên tại SinhVienReducer, ta xây dựng thêm một case nữa để xử lý action xóa được dispatch lên

Quản lý Sinh Viên				
Mã Sinh Viên	Họ Tên	Tuổi	Email	Hành Động
123	Dang Trung Hieu	123	123@gmail.com	<button>Delete</button> <button>Edit</button>
45	Phan Dinh Phung	43	dangtrunghieu147@gmail.com	<button>Delete</button> <button>Edit</button>

# Xây dựng chức năng Xóa Sinh Viên

Bước 1: khi click vào nút “xóa”, lấy được mã sinh viên của sinh viên dòng hiện tại

\* Bước 1.1 : tại component SinhVien, gắn sự kiện cho nó nút “Delete” , khi click sẽ lấy được MaSV muốn xóa.



```
1 import React from 'react';
2
3 const handleDeleteBtn = (maSV) => {
4     console.log(maSV);
5 }
6 const sinhvien = (props) => {
7     let {MaSV,HoTen,Tuoi,Email} = props.sv;
8     return (
9         <tr>
10            <td>{MaSV}</td>
11            <td>{HoTen}</td>
12            <td>{Tuoi}</td>
13            <td>{Email}</td>
14            <td>
15                <button className="btn btn-outline-success mr-2" onClick={() => handleDeleteBtn(MaSV)}>Delete</button>
16                <button className="btn btn-outline-danger">Edit</button>
17            </td>
18        </tr>
19    );
20 };
21
22 export default sinhvien;
```

# Xây dựng chức năng Xóa Sinh Viên

Bước 2: sau khi lấy được mã sinh viên, tiến hành dispatch 1 action lên store để tiến hành xóa người dùng trong danh sách (vì danh sách đang được lưu tại store)

- ❑ Bước 2.1: Connect component SinhVien với store. Lúc này props của component SinhVien sẽ có một phương thức mới là onDeleteSinhVien, khi phương thức này chạy sẽ dispatch action có type là “Delete\_SV” và maSV muốn xóa lên store
- ❑ Bước 2.2 : khi click vào nút “Delete” , tiến hành gọi phương thức onDeleteSinhVien chạy

```
File Edit Selection View Go Debug Terminal Help
dssv.jsx      sinhvien.jsx - shopping cart - Visual Studio Code
1 import React from 'react';
2 import {connect} from 'react-redux';
3
4 const handleDeleteBtn = (maSV) => {
5 }
6
7 const sinhvien = (props) => {
8 }
9
10
11 const mapDispatchToProps = (dispatch) => {
12   return {
13     onDeleteSinhVien: function(maSV){
14       dispatch({
15         type:"DELETE_SV",
16         maSV
17       })
18     }
19   }
20 }
21
22 export default connect(null,mapDispatchToProps)(sinhvien);
```

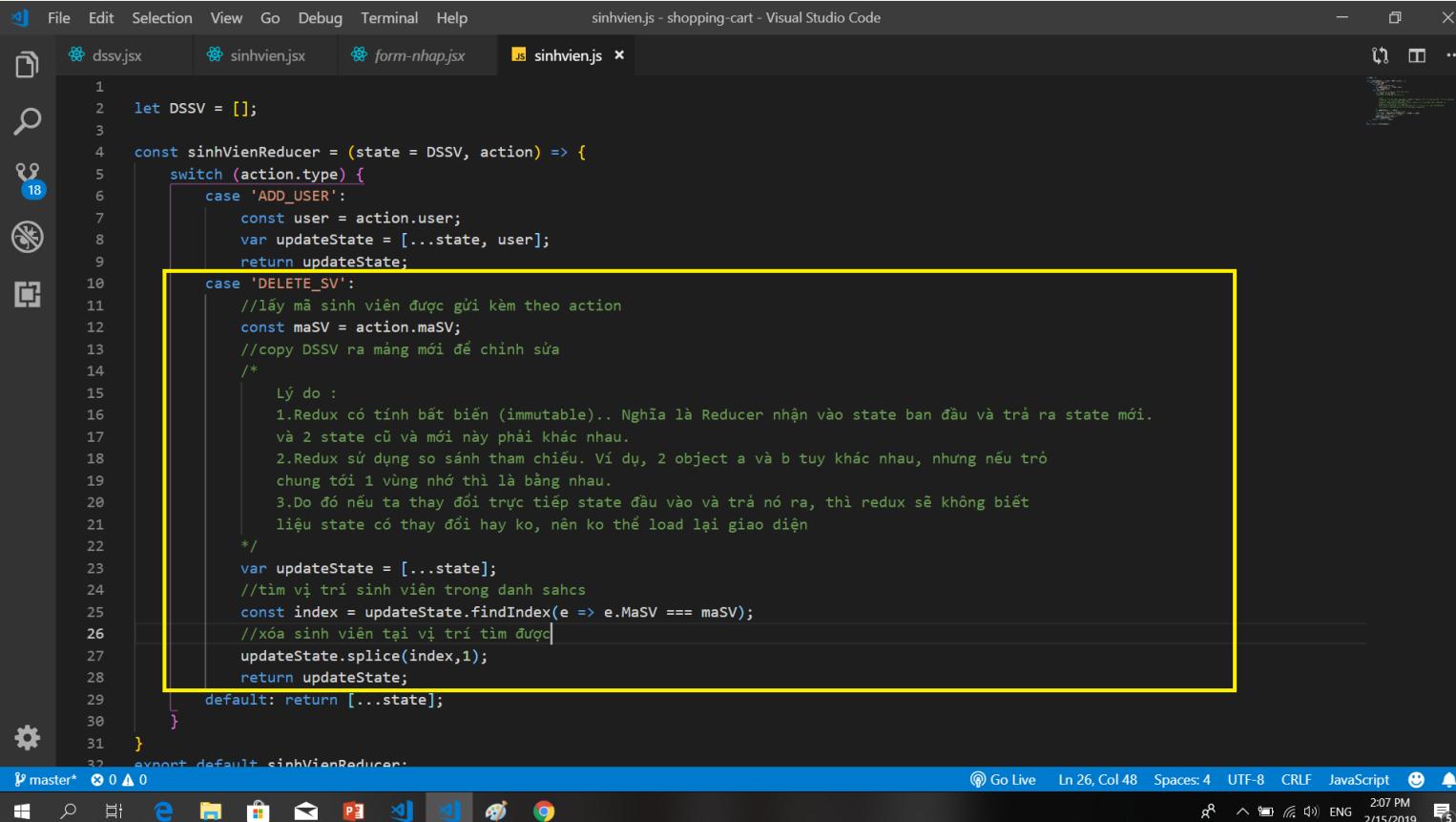
ta cần mapDispatchToProps như là tham số thứ 2 của hàm connect vì ta muốn dispatch action lên store

```
File Edit Selection View Go Debug Terminal Help
dssv.jsx      sinhvien.jsx - shopping cart - Visual Studio Code
1 import React from 'react';
2 import {connect} from 'react-redux';
3
4 const handleDeleteBtn = (maSV,onDeleteSinhVien) => {
5   onDeleteSinhVien(maSV);
6 }
7
8 const sinhvien = (props) => {
9   let {MaSV,HoTen,Tuoi,Email} = props.sv;
10  return (
11    <tr>
12      <td>{MaSV}</td>
13      <td>{HoTen}</td>
14      <td>{Tuoi}</td>
15      <td>{Email}</td>
16      <td>
17        <button className="btn btn-outline-success mr-2" onClick={() => handleDeleteBtn(MaSV,props.onDeleteSinhVien)}>Delete</button>
18        <button className="btn btn-outline-danger">Edit</button>
19      </td>
20    </tr>
21  );
22
23
24 const mapDispatchToProps = (dispatch) => {
25   return {
26     onDeleteSinhVien: function(maSV){
27       dispatch({
28         type:"DELETE_SV",
29         maSV
30       })
31     }
32   }
33 }
```

Vi props chỉ tồn tại trong function sinhVien, do đó để có thể sử dụng được props.onDeleteSinhVien trong hàm handleDeleteBtn, ta phải truyền props.onHandleDeleteBtn như là tham số để sử dụng

# Xây dựng chức năng Xóa Sinh Viên

Bước 3: muốn xóa thì ta phải xóa trong DSSV, mà DSSV thì do sinhVienReducer quản lý, nên tại SinhVienReducer, ta xây dựng thêm một case nữa để xử lý action xóa được dispatch lên



```
File Edit Selection View Go Debug Terminal Help
dssv.jsx | sinhvien.jsx | form-nhap.jsx | sinhvienjs
1
2 let DSSV = [];
3
4 const sinhVienReducer = (state = DSSV, action) => {
5     switch (action.type) {
6         case 'ADD_USER':
7             const user = action.user;
8             var updateState = [...state, user];
9             return updateState;
10
11        case 'DELETE_SV':
12            //lấy mã sinh viên được gửi kèm theo action
13            const maSV = action.maSV;
14            //copy DSSV ra mảng mới để chỉnh sửa
15            /*
16                Lý do :
17                1.Redux có tính bất biến (immutable).. Nghĩa là Reducer nhận vào state ban đầu và trả ra state mới.
18                và 2 state cũ và mới này phải khác nhau.
19                2.Redux sử dụng so sánh tham chiếu. Ví dụ, 2 object a và b tuy khác nhau, nhưng nếu trả
20                chung tới 1 vùng nhớ thì là bằng nhau.
21                3.Do đó nếu ta thay đổi trực tiếp state đầu vào và trả nó ra, thì redux sẽ không biết
22                liệu state có thay đổi hay ko, nên ko thể load lại giao diện
23            */
24            var updateState = [...state];
25            //tim vị trí sinh viên trong danh sach
26            const index = updateState.findIndex(e => e.MaSV === maSV);
27            //xóa sinh viên tại vị trí tìm được
28            updateState.splice(index,1);
29            return updateState;
30        default: return [...state];
31    }
32
33    export default sinhVienReducer;

```

# *Redux nâng cao : Action creator*

- ❖ Thay vì việc action trực tiếp trong component, giờ ta sẽ nâng cấp ứng dụng của mình bằng cách tách các action ra thành một file riêng và tạo các action creator.
- ❖ Vậy action creator là gì ? Dưới đây là ví dụ:
  - ❖ Bước 1: Trong folder “redux”, tạo thêm một folder mới “actions”. Trong folder “actions”, tạo ra một file để chứa tất cả các action liên quan tới sinh viên

```
EXPLORER          dssv.jsx  sinhvien.jsx  js sinhvien.js x
OPEN EDITORS  1 UNSAVED
  dssv.jsx  src\components\CRUD  U
  sinhvien.jsx  src\components\CRUD  U
  sinhvien.js  src\redux\actions  U
SHOPPING-CART
  src
    components
      cart
        CRUD
          dssv.jsx
          form-nhap.jsx
          sinhvien.jsx  U
      products
    redux
      actions
        sinhvien.js  U
      reducers
        index.js
        sinhvien.js  U
      App.js
      index.js
      serviceWorker.js
.gitignore

1 export const actDeleteSinhVien = (maSV) => {
2   return {
3     type:"DELETE_SV",
4     maSV
5   }
6 }
7 // Hàm trên được gọi là một action creator, nó trả về cho ta một
8 // action có type và maSV đi kèm.|
```

# *Redux nâng cao : Action creator*

- ❖ Tiếp theo, tại component SinhVien , ta sẽ thực hiện một số thay đổi

```
File Edit Selection View Go Debug Terminal Help
sinhvien.jsx - shopping-cart - Visual Studio Code
EXPLORER          dssv.jsx  sinhvien.jsx  sinhvien.js
OPEN EDITORS
  dssv.jsx src\components\CRUD
  ✘ sinhvien.jsx src\components\CRUD
    JS sinhvien.js src\redux\actions
SHOPPING-CART
  node_modules
  public
  src
    components
    cart
    CRUD
      dssv.jsx
      form-nhap.jsx
      sinhvien.jsx
    products
    redux
    actions
      JS sinhvien.js
    reducers
      JS index.js
      JS sinhvien.js
    App.js
    index.js
    serviceWorker.js
    .gitignore
    package-lock.json
OUTLINE

1  import React from 'react';
2  import {connect} from 'react-redux';
3
4  //import hàm actDeleteSinhVien từ sinhvien.js vào đây để sử dụng
5  import {actDeleteSinhVien} from '../../redux/actions/sinhvien';
6
7  const handleDeleteBtn = (maSV,onDeleteSinhVien) => {
8
9  }
10
11 const sinhvien = (props) => {
12
13 };
14
15 const mapDispatchToProps = (dispatch) => {
16   return {
17     onDeleteSinhVien: function(maSV){
18       // Thay vì dispatch trực tiếp action, ta sẽ dispatch hàm actDeleteSinhVien,
19       // khi hàm này chạy cũng return về action tương ứng
20       dispatch(actDeleteSinhVien(maSV))
21     }
22   }
23 }
24
25
26
27
28
29
30
31
32
33
34
35
36 export default connect(null,mapDispatchToProps)(sinhvien);
```

# *Redux nâng cao : Tách các type của action ra một file riêng*

- ❖ Trong folder **redux** , tạo thêm một folder **constants** . Trong folder **constants**, tạo một file **actionType.js**. Tiếp theo, trong **actions/sinhvien.js** , ta sẽ thực hiện một số chỉnh sửa như sau

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar:
  - OPEN EDITORS**: dssv.jsx, sinhvien.jsx, actionTypes.js (highlighted), sinhvien.js
  - SHOPPING-CART**: components, cart, CRUD, dssv.jsx, form-nhap.jsx, sinhvien.jsx, products, redux, actions, constants (highlighted), reducers, index.jsx, sinhvien.js, App.jsx, index.jsx, serviceWorker.js
- actionType.js** editor:

```
1 //import DELETE_SV from './constants/actionType'
2 import {DELETE_SV} from '../constants/actionType'
3 export const actDeleteSinhVien = (maSV) => {
4   return {
5     type: DELETE_SV,
6     maSV
7   }
8 }
9 // Hàm trên được gọi là một action creator, nó trả về cho ta một
10 // action có type và maSV đi kèm.
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: dssv.jsx, sinhvien.jsx, actionTypes.js, sinhvien.js
- sinhvien.js** editor:

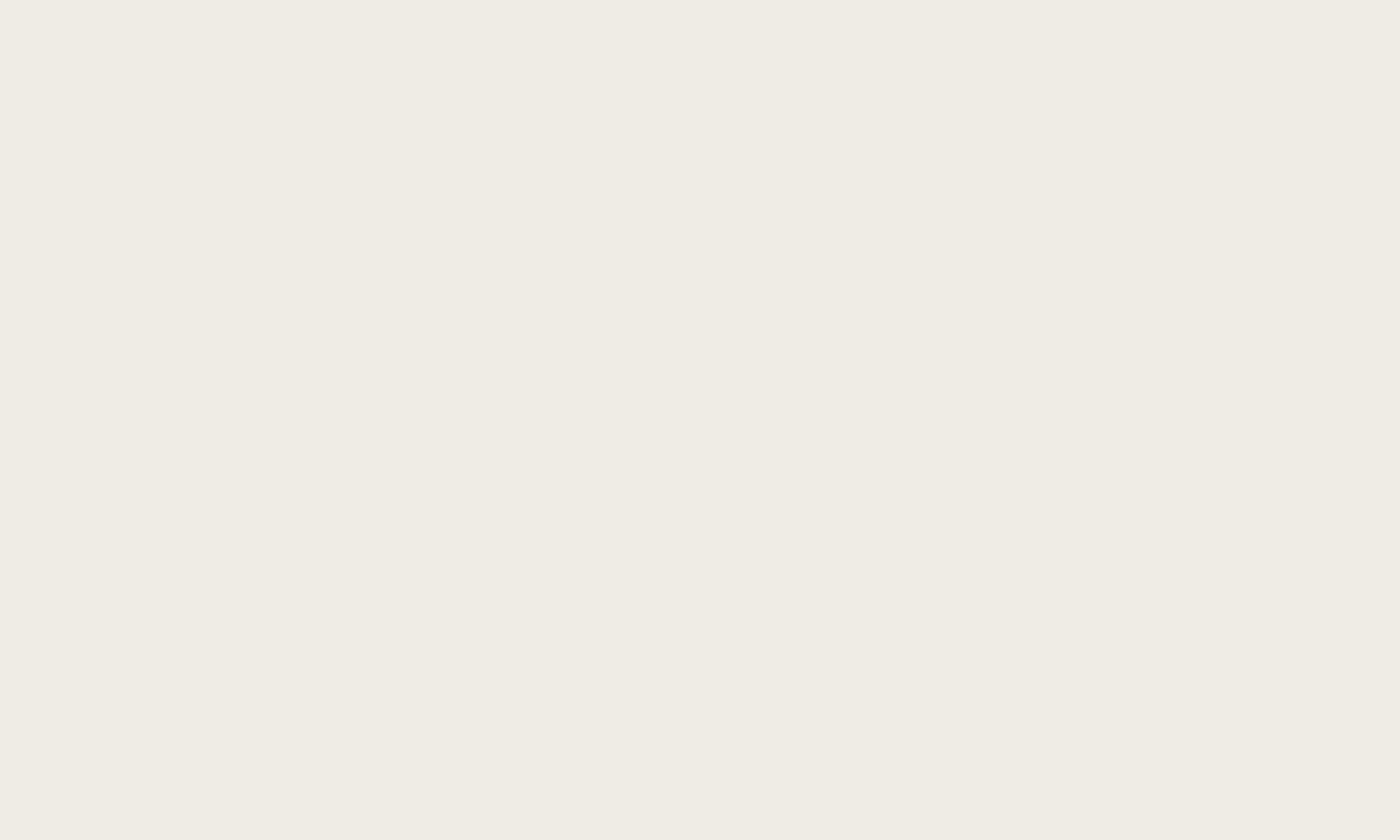
```
1 //import DELETE_SV from './constants/actionType'
2 import {DELETE_SV} from '../constants/actionType'
3 export const actDeleteSinhVien = (maSV) => {
4   return {
5     type: DELETE_SV,
6     maSV
7   }
8 }
9 // Hàm trên được gọi là một action creator, nó trả về cho ta một
10 // action có type và maSV đi kèm.
```

# *React HOOK*

25 -26 tháng 10 2018, một tính mới mang tên Hooks được công bố và đưa vào thử nghiệm ở phiên bản 16.7.0-alpha.

Hooks là một đề xuất tính năng mới cho phép bạn sử dụng state và các tính năng React khác mà không cần viết một Class

## *Form - validation*

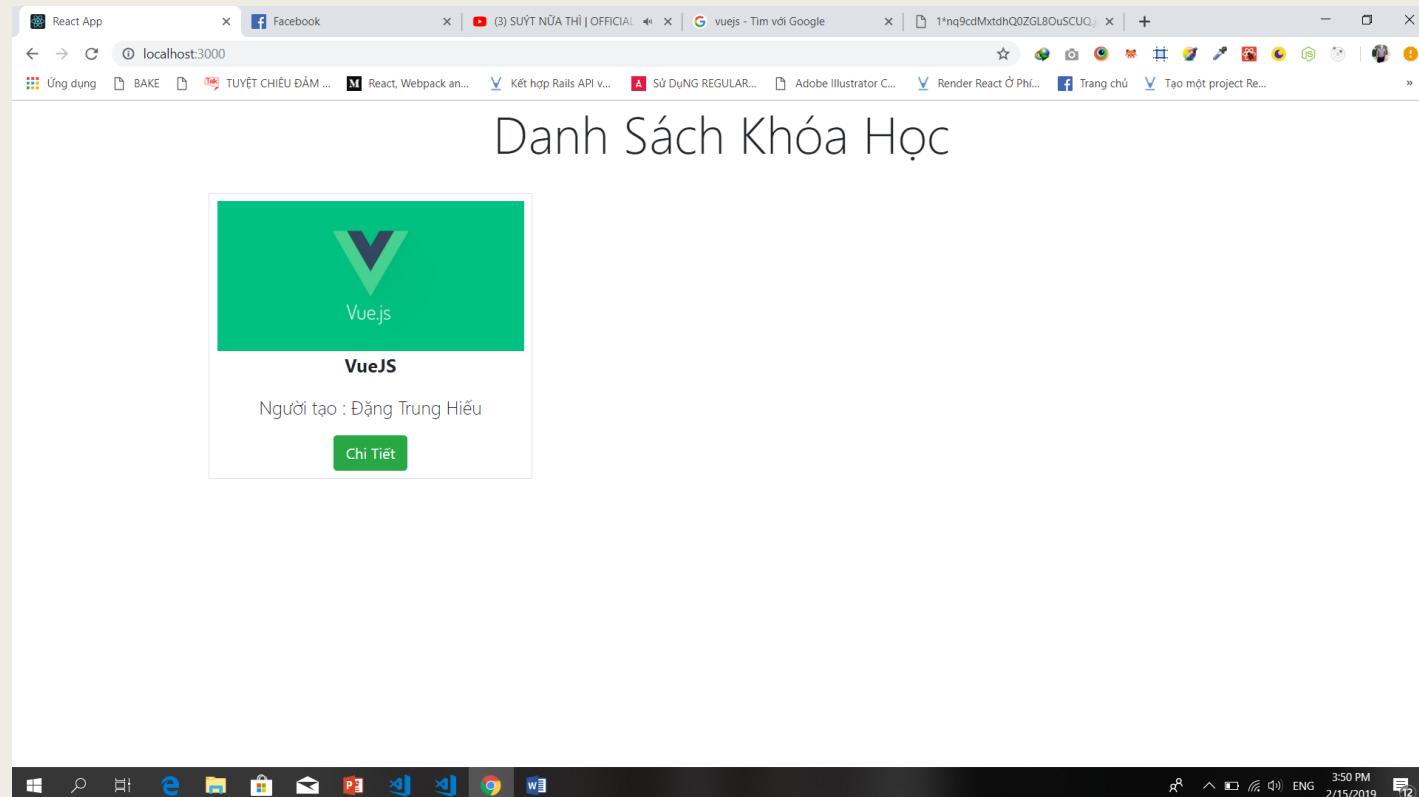


# HTTP và kết nối API

---

# *Thư viện hỗ trợ : Axios*

- ❖ Axios là thư viện hỗ trợ 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



# 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 hiển thị ra màn hình

DanhSachKhoaHoc

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a file tree. In the 'OPEN EDITORS' section, there is a file named 'dskhoahoc.jsx'. This file is highlighted with a yellow box. The code editor shows the following code:

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

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>
          </div>
          <div className="col-4">
            <KhoaHoc />
          </div>
        </div>
      </div>
    );
  }
}

export default DanhSachKhoaHoc;
```

KhoaHoc

The screenshot shows the VS Code interface with the code editor open. The file is named 'khoahoc.jsx'. The code is as follows:

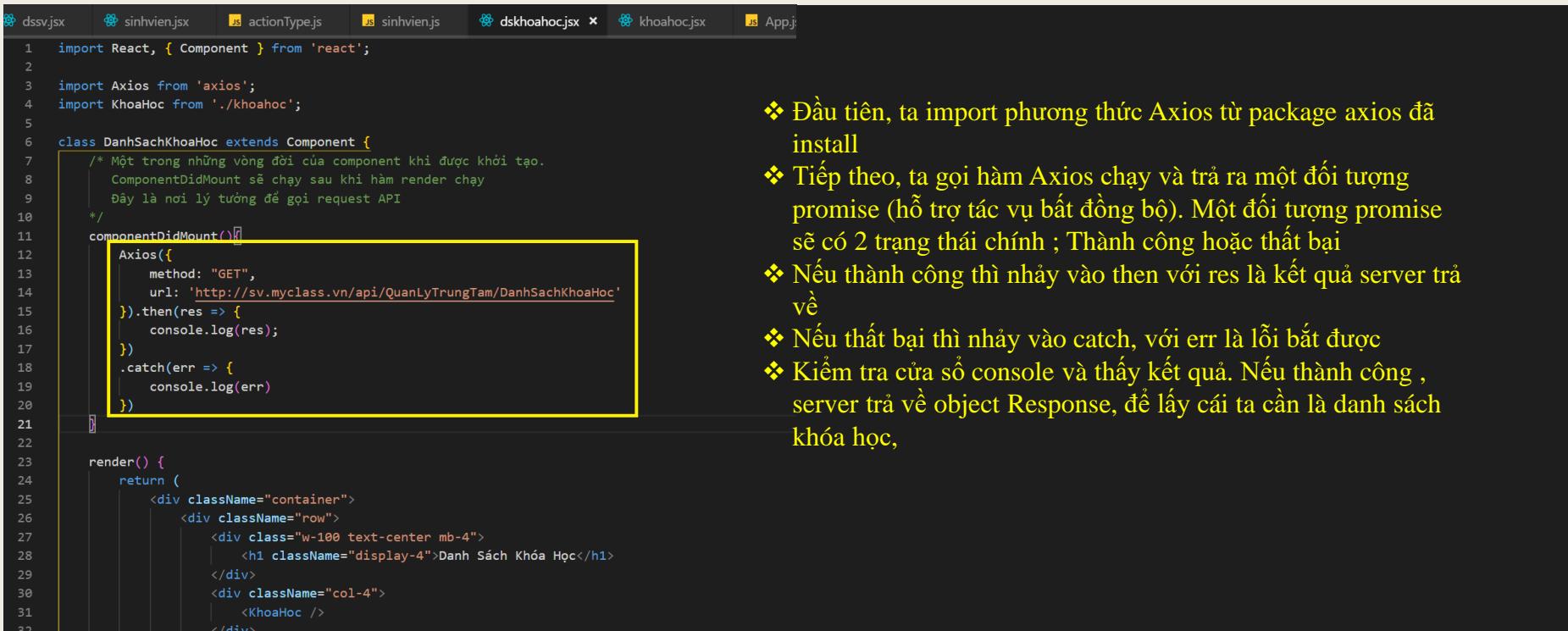
```
import React from 'react';

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}
```

- ❖ 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 code editor panes side-by-side.

**Left Editor (File: `khoaahoc.js`)**

```
1 let DSKH = [];
2
3 const khoaHocReducer = (state = DSKH, action) => {
4     switch(action.type){
5         case 'ADD_KHOA_HOC':
6             return [...state, action.payload];
7         default: return [...state];
8     }
9 }
export default khoaHocReducer;
```

**Right Editor (File: `index.js`)**

```
1 import { combineReducers } from 'redux';
2
3 import SinhVienReducer from './sinhvien';
4 import KhoaHocReducer from './khoaahoc';
5
6 const rootReducer = combineReducers({
7     DSSV: SinhVienReducer,
8     DSKH : KhoaHocReducer
9 })
10
11
12
13
14
15
16 export default rootReducer;
```

# *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
2  export const DELETE_SV = "DELETE_SV";
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
  }
}
```

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

# *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)
  })
}
```

# 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>
        {this.renderCourse()}
      </div>
    );
}
```

# Nâng cấp: tạo async action với middleware

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

```
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;
```

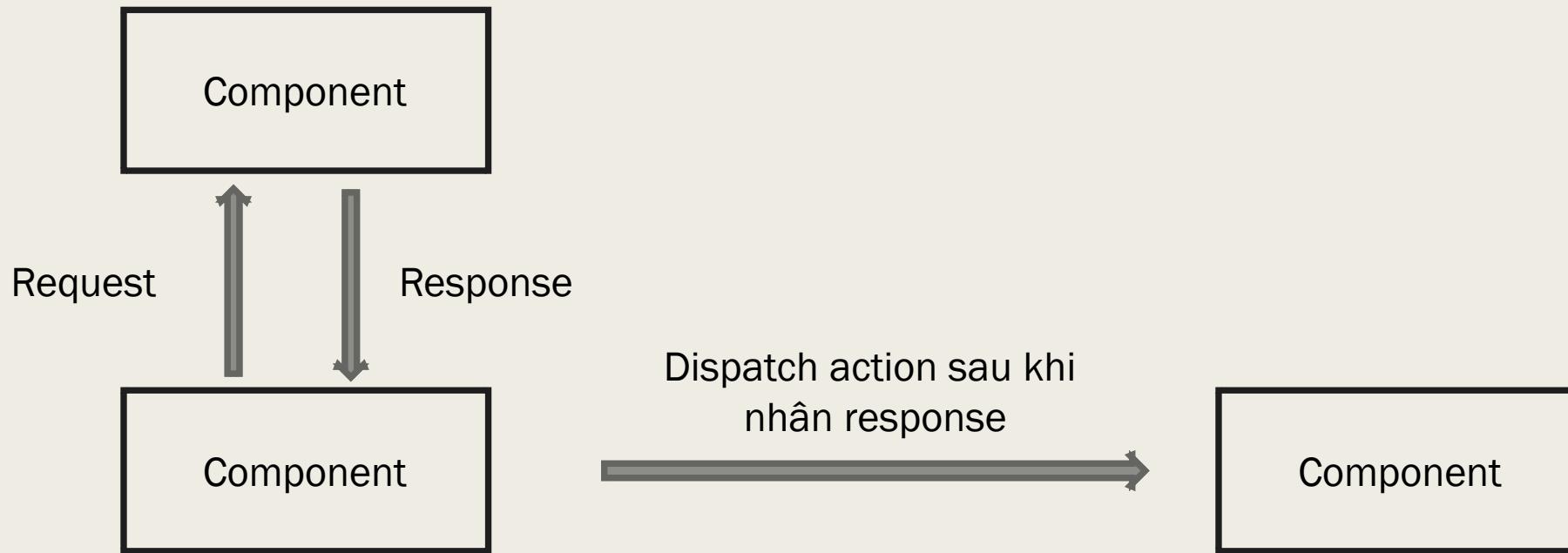
# *Bài tập : lấy danh sách khóa học*

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

```
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;
```

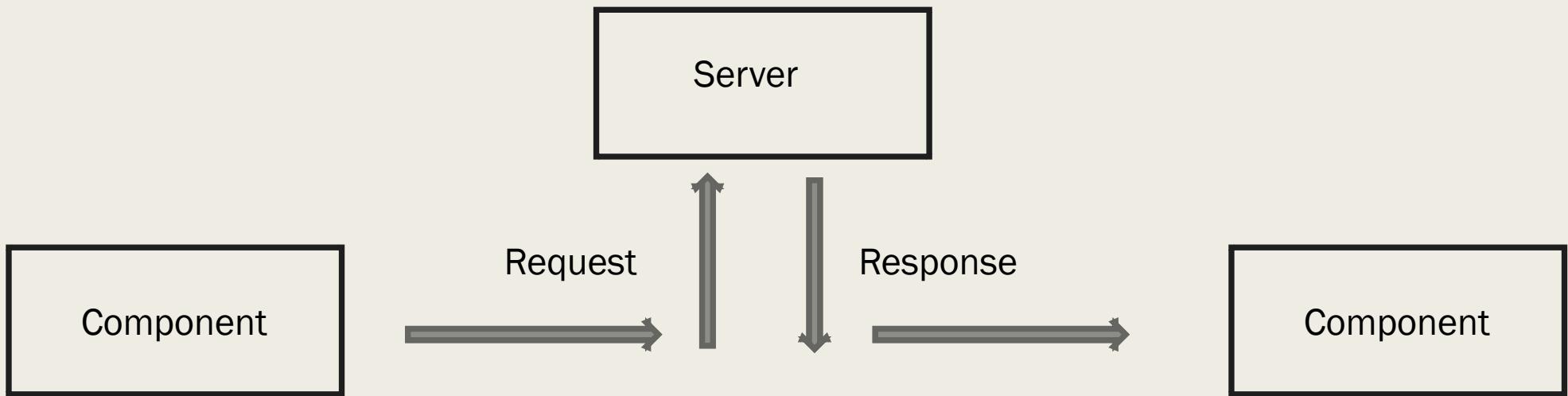
## Nâng cấp : tạo async action

- ❖ 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.

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

import Axios from 'axios';

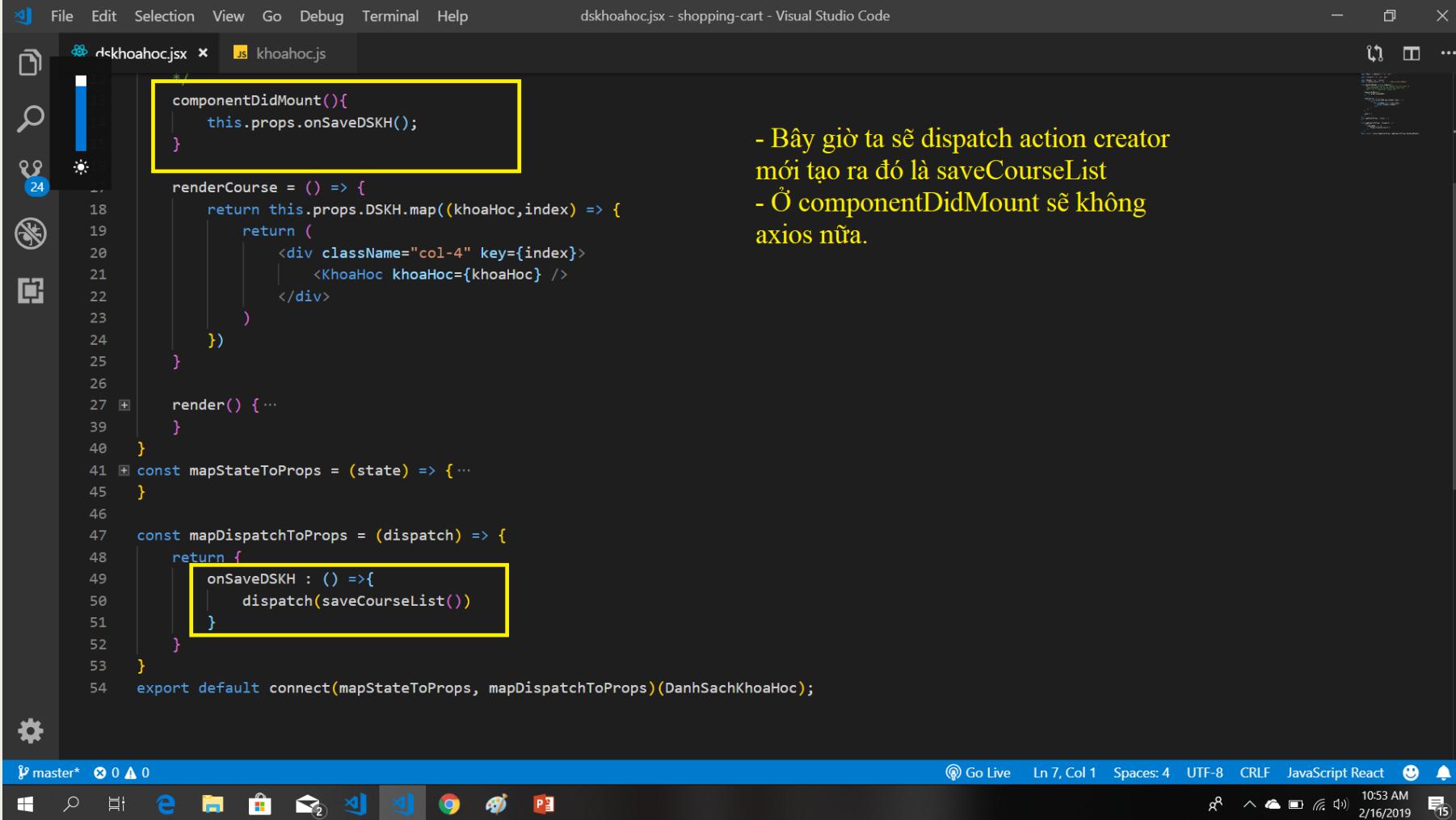
export const saveCourseList = () => {
  return (dispatch) => {
    Axios({
      method: "GET",
      url: 'http://sv.myclass.vn/api/QuanLyTrungTam/DanhSachKhoaHoc'
    }).then(res => {
      dispatch(actGetCourseList(res.data))
    })
    .catch(err => {
      console.log(err)
    })
  }
}

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



```
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);
```

- Bây giờ 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**

# 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

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure with files like dskhoahoc.jsx, khoahoc.js, index.js, dssv.jsx, form-nhap.jsx, khoahoc.jsx, sinhvien.jsx, products, redux/actions/khoahoc.js, redux/actions/sinhvien.js, redux/constants/actionType.js, redux/reducers/index.js, App.js, index.js, serviceWorker.js, .gitignore, and package-lock.json.
- CODE EDITOR:** The file `index.js` is open. The code is as follows:

```
5 import * as serviceWorker from './serviceWorker';
6
7 import { createStore, applyMiddleware } from 'redux';
8
9 import { Provider } from 'react-redux'
10
11 import rootReducer from './redux/reducers/index';
12
13 import thunk from 'redux-thunk';
14
15 const store = createStore(rootReducer, applyMiddleware(thunk));
16
17 ReactDOM.render(
18   <Provider store={store}>
19     <App />
20   </Provider>
21   ,
22   document.getElementById('root')
23 )
24
25 // If you want your app to work offline and load faster, you can change
26 // unregister() to register() below. Note this comes with some pitfalls.
27 // Learn more about service workers: http://bit.ly/CRA-PWA
28 serviceWorker.unregister();
29
```

Two specific sections of the code are highlighted with yellow boxes:

- The import statement for `applyMiddleware` from `'redux'`.
- The import statement for `thunk` from `'redux-thunk'`, and the subsequent line where `thunk` is passed to `applyMiddleware`.

# Routing

---

# *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 người dùng . 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

# 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>
    );
  }
}
```

## Home.jsx

```
dskhoahoc.jsx  js khoahoc.js  js index.js  js App.js  tsx 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;
```

❖ 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>
    );
  }
}

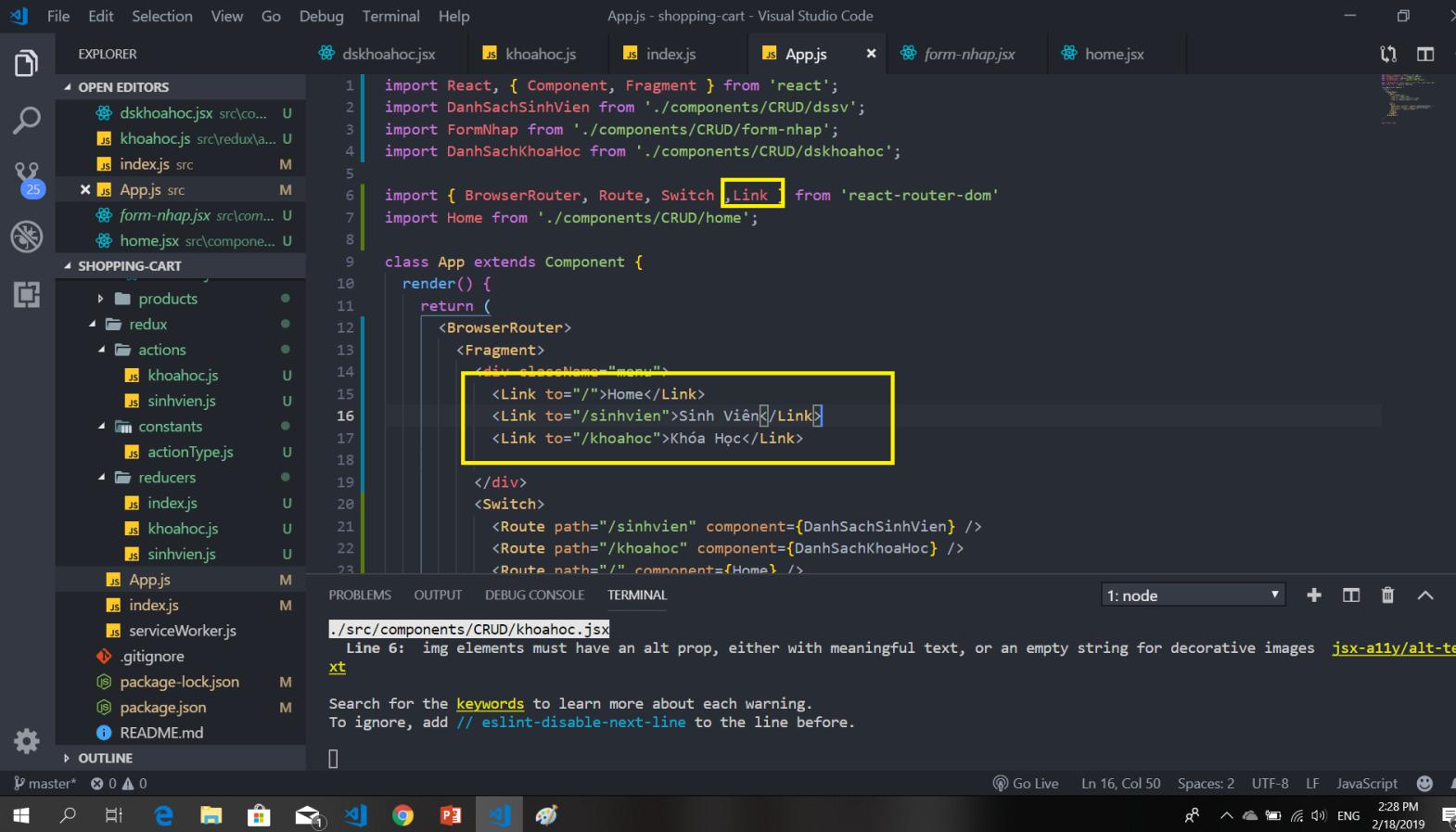
export default App;
```

```
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 under the "SHOPPING-CART" folder, including "products", "redux", "constants", and "reducers".
- Code Editor:** The "App.js" file is open, displaying the following code:

```
1 import React, { Component, Fragment } from 'react';
2 import DanhSachSinhVien from './components/CRUD/dssv';
3 import FormNhaph from './components/CRUD/form-nhap';
4 import DanhSachKhoaHoc from './components/CRUD/dskhahoc';
5
6 import { BrowserRouter, Route, Switch, Link } from 'react-router-dom'
7 import Home from './components/CRUD/home';
8
9 class App extends Component {
10   render() {
11     return (
12       <BrowserRouter>
13         <Fragment>
14           <div className="menu">
15             <Link to="/">Home</Link>
16             <Link to="/sinhvien">Sinh Viên</Link>
17             <Link to="/khoahoc">Khóa Học</Link>
18           </div>
19           <Switch>
20             <Route path="/sinhvien" component={DanhSachSinhVien} />
21             <Route path="/khoahoc" component={DanhSachKhoaHoc} />
22             <Route path="/" component={Home} />
23         </Fragment>
24     )
25   }
26 }
27
28 export default App;
```

The code uses the `Link` component from `react-router-dom` to create navigation links. The `to` prop is used to specify the URL paths: `/`, `/sinhvien`, and `/khoahoc`. These paths correspond to the `path` props defined in the `Route` components further down in the code.
- Terminal:** Shows a warning message about `img` elements missing `alt` attributes.
- Status Bar:** Displays the current file is "master", line 16, column 50, and the date/time as 2/18/2019 2:28 PM.

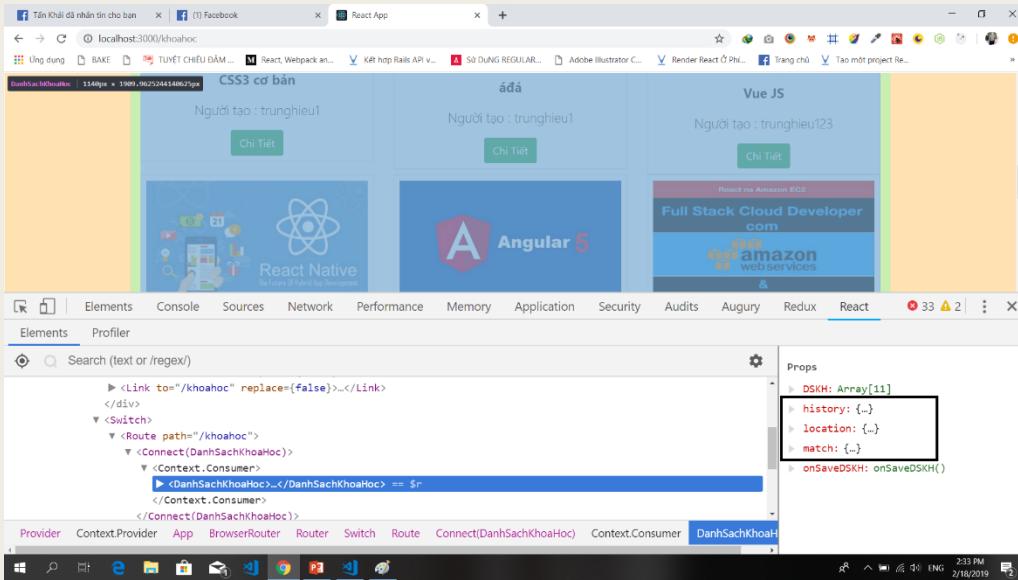
# *Chuyển đổi component*

- ❖ Sử dụng component <NavLink> để chuyển đổi component

```
<NavLink exact  
    activeStyle={{ backgroundColor : 'white', color : 'red' }} to="/"  
    className="my-link">Trang Chu  
</NavLink>
```

# 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.

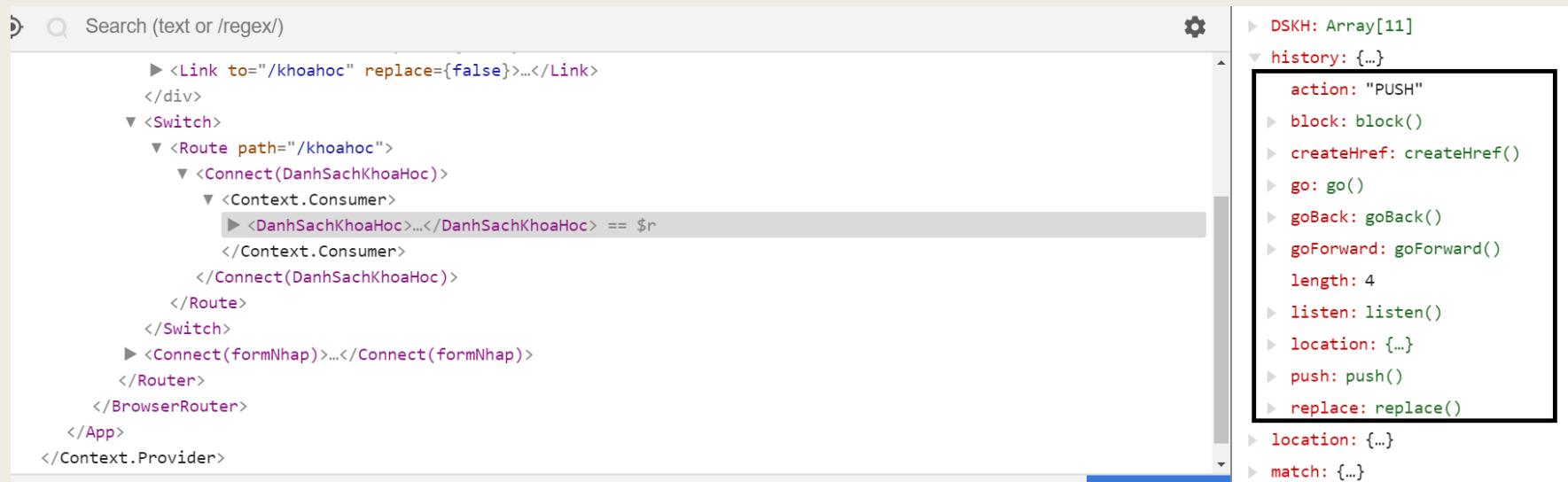


- ❖ Vậy công dụng của 3 thuộc tính mới này là gì?

# 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.



# 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

```
 1 import React, { Component } from 'react';
 2
 3 class home extends Component {
 4   goToCoursePage =() =>{
 5     this.props.history.push('/khoa hoc');
 6     // hoặc
 7     this.props.history.replace('/khoa hoc');
 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 the browser's developer tools with the element inspector open. On the left, the component tree is displayed:

```
<Provider>
  <Context.Provider>
    <App>
      <BrowserRouter>
        <Router>
          <div className="menu">
            <Link to="/" replace={false}>...</Link>
            <Link to="/sinhvien" replace={false}>...</Link>
            <Link to="/khoaHoc" replace={false}>...</Link>
          </div>
          <Switch>
            <Route path="/khoaHoc">
              <Connect(DanhSachKhoaHoc)>
                <Context.Consumer>
                  <DanhSachKhoaHoc>...</DanhSachKhoaHoc> == $r
                </Context.Consumer>
              </Connect>
            </Route>
          </Switch>
        </Router>
      </BrowserRouter>
    </App>
  </Context.Provider>
</Provider>
```

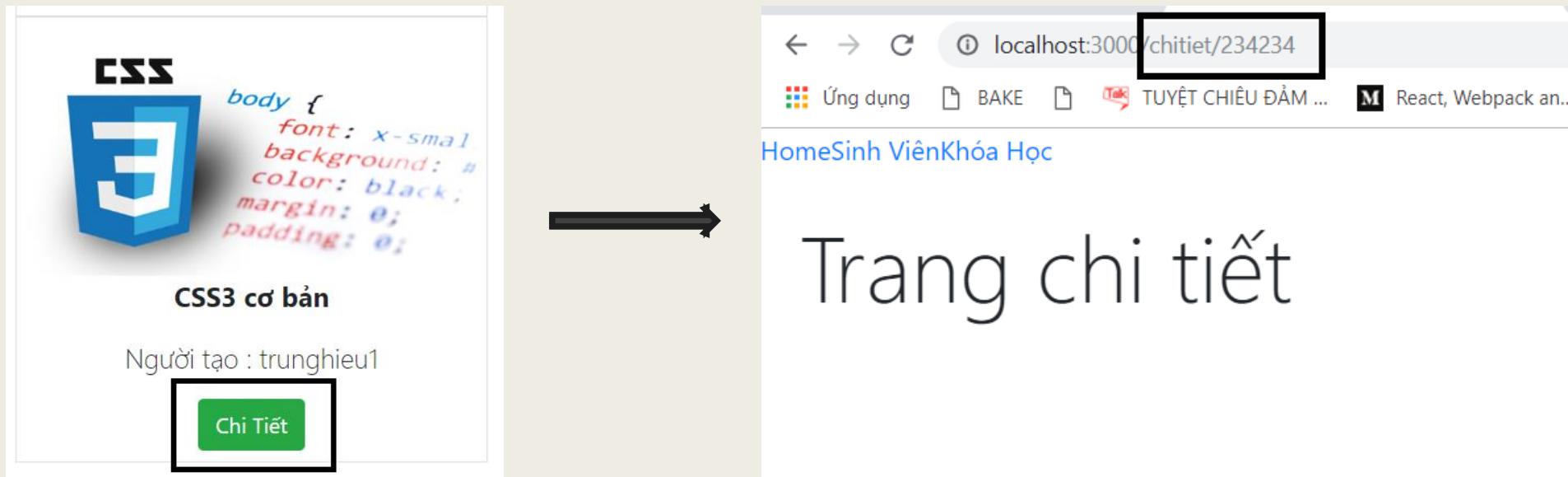
On the right, a detailed view of the 'match' object is shown, highlighted with a black border:

```
match: {...}
  isChecked: true
  params: {...}
  Empty object
  path: "/khoaHoc"
  url: "/khoaHoc"
  onSaveDSKH: onSaveDSKH()
```

The 'DanhSachKhoaHoc' tab is selected at the bottom of the inspector.

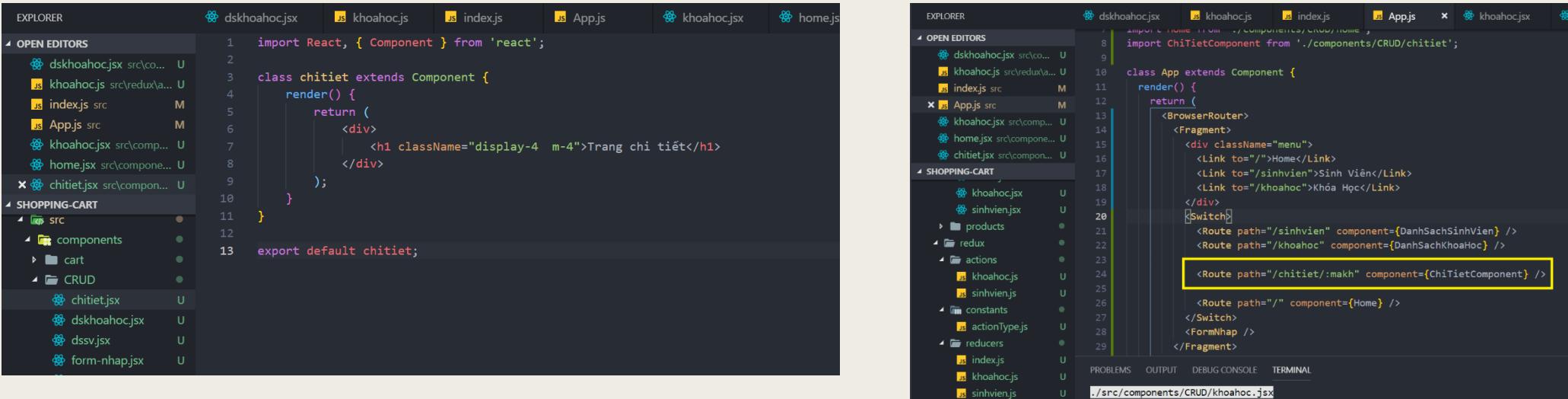
# Match

- ❖ 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



The image shows two side-by-side code editors in a development environment.

**Left Editor:** The file `chitiet.jsx` contains the following code:

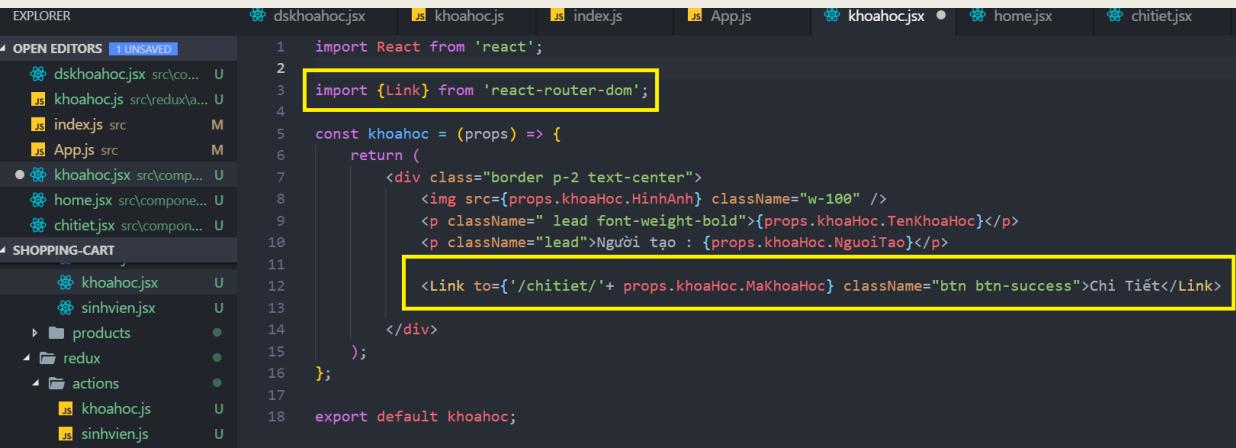
```
1 import React, { Component } from 'react';
2
3 class chitiet extends Component {
4     render() {
5         return (
6             <div>
7                 <h1 className="display-4 m-4">Trang chi tiết</h1>
8             </div>
9         );
10    }
11 }
12
13 export default chitiet;
```

**Right Editor:** The file `App.jsx` contains the following code, with the relevant part highlighted:

```
8 import Home 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="/chitiet/:makh" component={ChiTietComponent} />
24                     </Switch>
25                 </Fragment>
26             </BrowserRouter>
27         );
28     }
29 }
30
31 export default App;
```

The line `<Route path="/chitiet/:makh" component={ChiTietComponent} />` is highlighted with a yellow box.

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



The image shows a code editor with the file `khoaHoc.jsx` open.

The code includes the following imports:

```
1 import React from 'react';
2 import {Link} from 'react-router-dom';
```

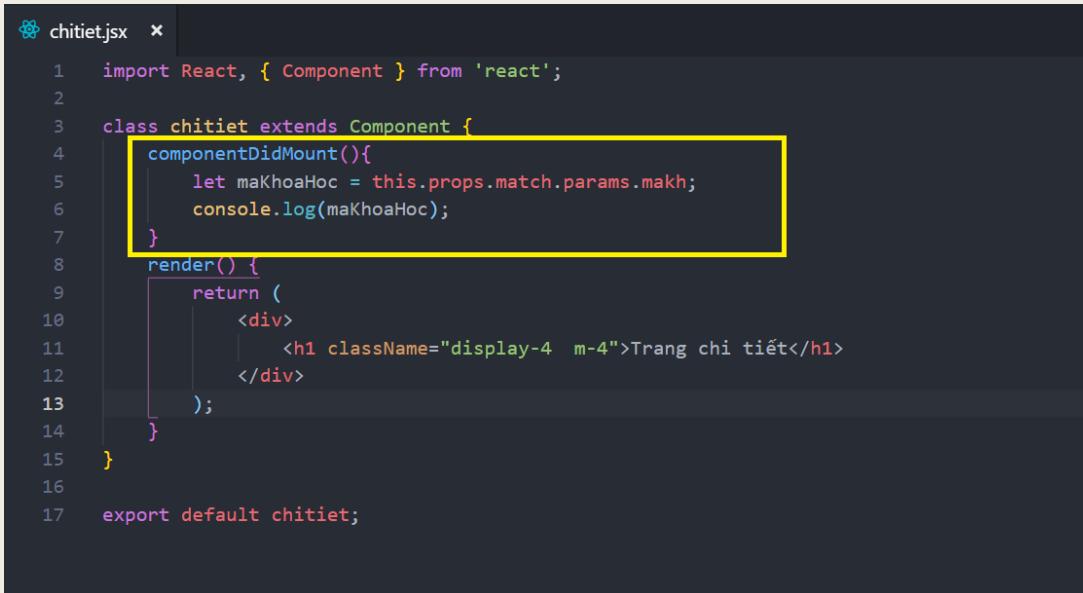
And the component definition:

```
3 const khoahoc = (props) => {
4     return (
5         <div className="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
10             <Link to={'/chitiet/' + props.khoaHoc.MaKhoaHoc} className="btn btn-success">Chi Tiết</Link>
11         </div>
12     );
13 }
14
15
16
17
18 export default khoahoc;
```

The line `<Link to={'/chitiet/' + props.khoaHoc.MaKhoaHoc} className="btn btn-success">Chi Tiết</Link>` is highlighted with a yellow box.

# Match

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

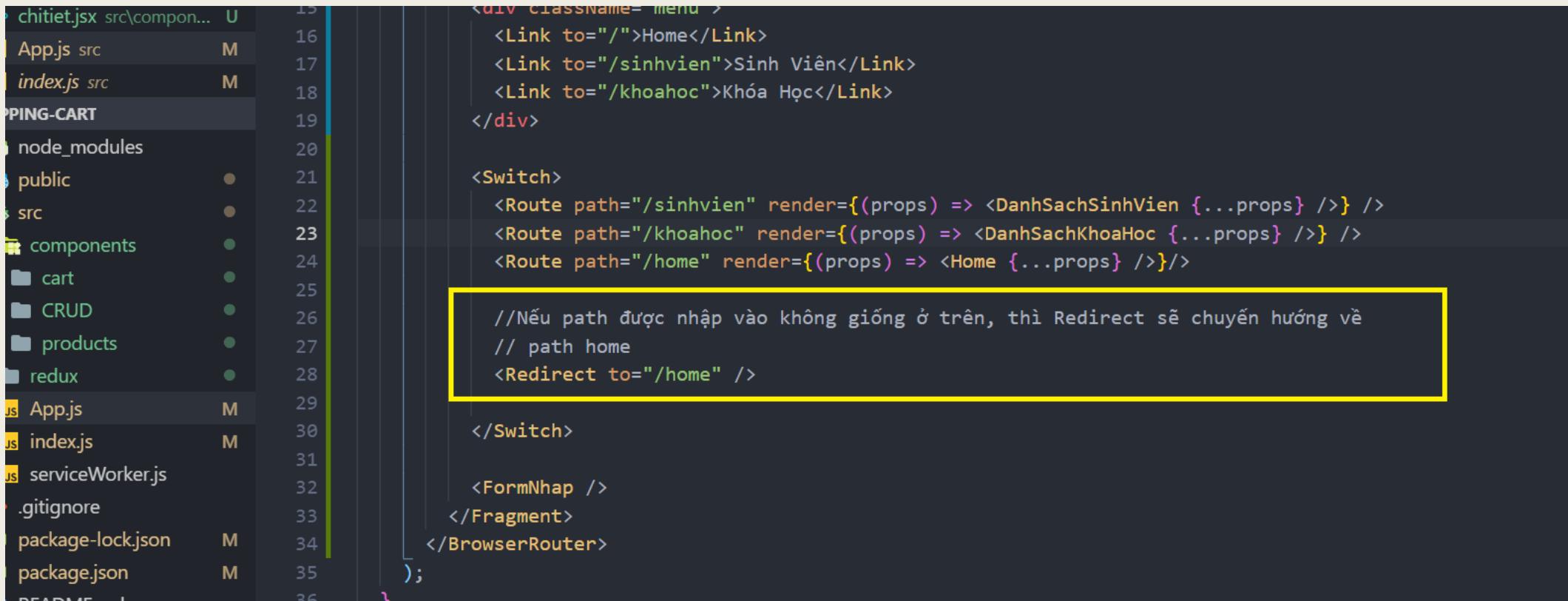


```
chitiet.jsx
1 import React, { Component } from 'react';
2
3 class chitiet extends Component {
4     componentDidMount(){
5         let maKhoaHoc = this.props.match.params.makh;
6         console.log(maKhoaHoc);
7     }
8     render() {
9         return (
10             <div>
11                 <h1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

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

# 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 `Switch` component with three `Route` components for "/sinhvien", "/khoaHoc", and "/home". Below these, there is a `Redirect` component with the attribute `to="/home"`. This indicates that if a path is entered that does not match any of the defined routes, the application will redirect the user to the "/home" page. The code is highlighted with a yellow box.

# Promt

- ❖ Component Promt cho phép người dùng xác định trạng thái cho phép người dùng rời khỏi trang hay không.
- ❖ Thuộc tính:
  - + when: trả về giá trị true hộp thoại hiển thị lên ngược lại false không hiển thị.
  - + message: Hộp thoại chứa nội dung hiển thị

```
import React, { Component } from 'react';
import {Prompt} from 'react-router-dom';
export default class login extends Component {
    state = {
        isSaveForm:true,
        TaiKhoan:'',
        MatKhau:''
    };
    Login = (e) => {
        e.preventDefault();
        console.log(1);
    }
    render() {
        return (
            <div className="container">
                <h3 className="display-4">Đăng nhập</h3>
                <form onSubmit={this.Login}>
                    <div className="form-group">
                        <label htmlFor="exampleInputEmail1">Tài khoản</label>
                        <input type="email" className="form-control" name="TaiKhoan" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Tài khoản" />
                    </div>
                    <div className="form-group">
                        <label htmlFor="exampleInputPassword1">Password</label>
                        <input type="password" className="form-control" name="MatKhau" id="exampleInputPassword1" placeholder="Password" />
                    </div>
                    <button type="submit" className="btn btn-primary">Đăng nhập</button>
                </form>
                <Prompt
                    when = {this.state.isSaveForm}
                    message={location => ('Bạn có chắc muốn rời khỏi trang này ?')}
                />
            </div>
        )
    }
}
```

# Cơ chế Guard (bảo vệ Route trong React)



- ❖ Để bảo vệ Route tránh cho người dùng truy cập vào trường hợp không đủ điều kiện.
- ❖ Ví dụ: khi tài khoản là người dùng nhưng lại muốn truy cập vào route admin,... hoặc người dùng 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

# 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;
```

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;
```

- ❖ 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
- ❖ Nếu dưới localStorage có user rồi , nghĩa là đã có người đăng nhập thì, thì ra render ra Component, (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

# *Bài tập tổng hợp*

- ❖ Thực hiện các kiểm tra trước khi vào admin phải đăng nhập, nhập liệu xong chưa nhấn submit sẽ hiển thị hộp thoại. Xây dựng tính năng đăng ký đăng nhập
- ❖ Tạo trang quản lý khóa học, tại admin template xây dựng tính năng quản lý khóa học (thêm, xóa, sửa) sử dụng redux, axios để xây dựng các service.