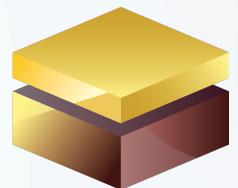
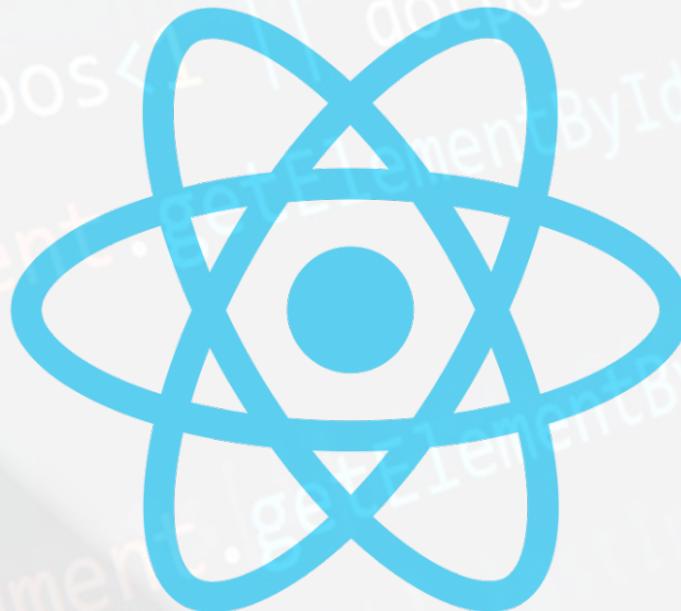


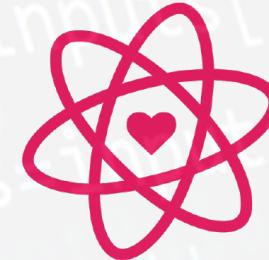
# React js



**CYBERLEARN**  
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Hướng dẫn : Trương Tấn Khải

# Giới thiệu



# React JS

## Binding dữ liệu trong react

&&

## state and updating state

# Databinding (hiển thị dữ liệu lên html)

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

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

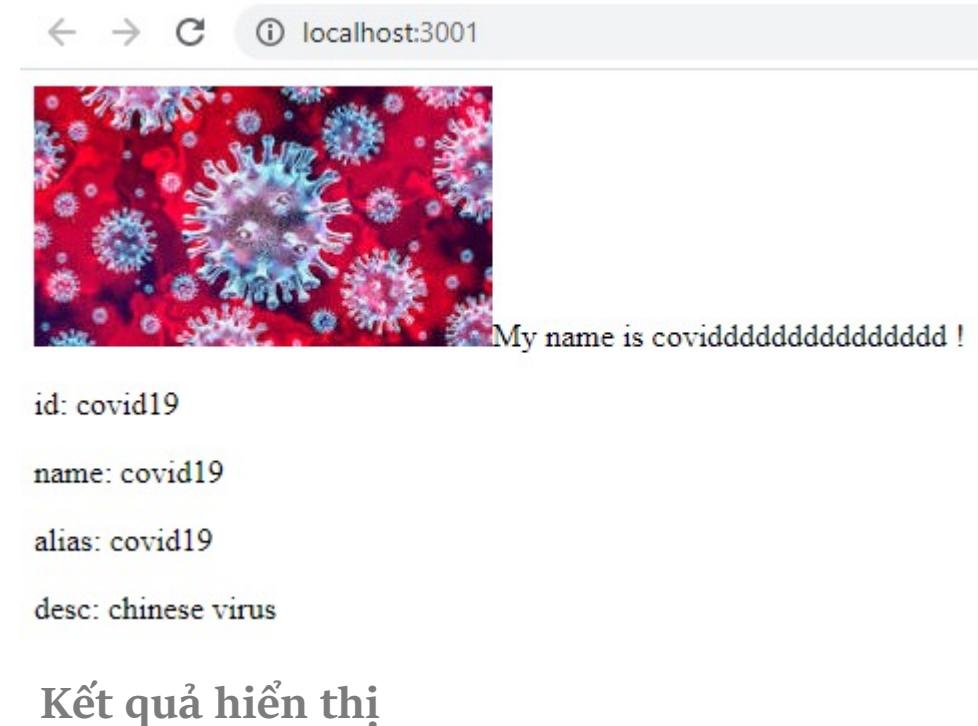
```
JS App.js JS Product.jsx x
reactslide > src > components > Product.jsx > ...
1 import React, { Component } from 'react'
2
3 export default class Product extends Component {
4
5   //Tạo ra 1 phương thức trả về 1 đoạn jsx
6   renderProduct = () => {
7     let title = 'CYBERSOFT';
8     let productName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

# Databinding (hiển thị dữ liệu lên html)

✓ React còn cho phép chúng ta sử dụng gọi hàm để binding dữ liệu `{tenham()}`

➔ Lưu ý: Kết quả của hàm trả về bắt buộc là chuỗi hoặc 1 đoạn jsx (React component)

```
3 export default class BindingDemoRCC extends Component {
4   //Xây dựng phương thức binding dữ liệu là component
5   renderImage = () => {
6     //Nội dung trả về 1 component
7     //(Lưu ý là đối tượng <img> jsx gần giống html do react định nghĩa không phải html)
8     return 
9   }
10  //Xây dựng phương thức binding dữ liệu là chuỗi
11  renderText = () => {
12    return 'My name is covidoooooooooooo !'
13  }
14  renderMultiComponent = () => {
15    //Kết hợp xử lý binding đối tượng tại hàm và trả về 1 component
16    const virus = {
17      id: 'covid19',
18      name:'corona',
19      alias: 'SARS-CoV-2',
20      desc: 'chinese virus'
21    }
22    return <div className="container">
23      <p>id: {virus.id}</p>
24      <p>name: {virus.id}</p>
25      <p>alias: {virus.id}</p>
26      <p>desc: {virus.desc}</p>
27    </div>
28  }
29  render() {
30    return (
31      <div>
32        {/*Gọi hàm */}
33        <this.renderImage()>
34        <this.renderText()>
35        <this.renderMultiComponent()>
36      </div>
37    )
38  }
39}
40
```



Kết quả hiển thị

React class component (Ứng với react functional component cũng tương tự). Đề cập slide bên dưới !

# Event handler( Xử lý sự kiện)

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

```
1 import React, { Component } from 'react'
2
3 export default class EventDemoRCC extends Component {
4   //Tạo ra 1 PHƯƠNG THỨC (Để truy xuất đến PHƯƠNG THỨC trong CLASS ta dùng từ khóa this.tenPhuongThuc)
5   handleClick = () => {
6     alert("Hello Khải !!!")
7   }
8
9   render() {
10     return (
11       <div>
12         |   <button onClick={this.handleClick}>Click me</button> <br />
13       </div>
14     )
15   }
16 }
```

```
1 import React from 'react'
2
3 export default function EventDemoRFC() {
4
5   /*Khác với react class react functional vì chỉ là 1 hàm nên bên trong hàm
6    object hay function đều được khai báo dưới dạng biến là let var hay const
7    */
8   const handleClick = () => {
9     alert("Hello Khải !!!")
10 }
11
12
13
14
15
16   /*
17    Khi gọi hàm không cần con trỏ this bởi vì đơn giản component này không phải là
18    lớp đối tượng mà ta sử dụng nó như là 1 function bình thường thôi
19    (Bên trong function thì ta khai báo và sử dụng biến bình thường thôi hehe khai báo this chỉ cho rườm rà nè !)
20    Tuy nhiên vẫn sử dụng được this nhé (nhưng không cần thiết)
21   */
22   <button onClick={handleClick}>Click me</button>
23
24 }
```

# Passing Arguments to Event Handlers

## (Truyền tham số qua sự kiện)

```
1 import React, { Component } from 'react'
2
3 export default class EventDemoRCC extends Component {
4     //Tạo ra 1 PHƯƠNG THỨC (Để truy xuất đến PHƯƠNG THỨC trong CLASS ta dùng từ khóa this.tenPhuongThuc)
5     handleClick = () => {
6         alert("Hello Khải !!!")
7     }
8
9     //Đối với sự kiện có tham số truyền vào có 2 cách để gán cho sự kiện (ở đây là onClick)
10    //Cách 1 dùng phương thức .bind(this, tenHam) để xử lý khi người dùng click mới gọi hàm này
11    showMessage = (message,tagButton) => {
12        console.log(tagButton.target) //tagButton.target: đại diện cho đối tượng button => tag button là this
13        alert(message); //message: là chuỗi 'cyberlearn chào covid !'
14    }
15
16    //Cách 2 dùng arrowFunction khi người dùng click mới gọi hàm này
17    //Cách này đơn giản hơn dùng khai báo 1 function nặc danh khi click vào sẽ gọi function đó (bên trong function khi thực hiện sẽ gọi hàm này)
18    reply = (content) => {
19        alert(content);
20    }
21
22    render() {
23        return [
24            <div>
25                <button onClick={this.handleClick}>Click me</button> <br />
26
27                /* Khi sử dụng bind truyền
28                    tham số 1: this là chính đối tượng button (mỗi thẻ component là 1 đối tượng)
29                    tham số 2: giá trị khi hàm đó được click sẽ gọi hàm đó xử lý
30                */
31                <button name="btnShowMessage" onClick={this.showMessage.bind(this, 'cyberlearn chào covid !')}>Show Message</button>
32
33                <button name="btnReplyMessage" onClick={() => { this.reply('next !') }}>Reply message!</button>
34
35            </div>
36        ]
37    }
}
```

# Render với điều kiện if (Rendering conditions)

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

```
1 import React, { Component } from 'react'
2
3 export default class Rendering_Conditions extends Component {
4
5   //Thuộc tính
6   login = true;
7   userName = 'Trương Tấn Khải';
8
9   //Phương thức gọi render nội dung
10  renderContent = () => {
11    /*Để render kết quả tùy chọn bởi 1 điều kiện nào đó ta viết hàm render và thực thi logic tại hàm
12     |=> Kết quả miễn là trả về js là được => Bên dưới chỉ việc gọi hàm này khi cần render nội dung
13     */
14    if(this.login)
15    {
16      return <h1>Hello {this.userName}</h1>
17    }
18    return <button> Đăng nhập </button>
19  }
20
21
22  render() {
23    return (
24      <div>
25        {this.renderContent()}
26      </div>
27    )
28  }
29}
```

# Render và rerender với state (state and updating state)

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

Lưu ý:

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

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

```
this.setState({  
    thuocTinh: [giá trị mới]  
})
```

# Render và rerender với state (state and updating state)

```
1 import React, { Component } from 'react'
2
3 export default class StateDemo extends Component {
4
5   userName = 'Trương Tấn Khải';
6
7   //Thuộc tính state : là thuộc tính có sẵn từ class Component
8   state = {
9     login:false //Định nghĩa các biến làm thay đổi giao diện tại đây (Là thuộc tính của state)
10  }
11
12
13  //Phương thức login
14  login = () => {
15    /*Phương thức this.setState(): phương thức làm thay đổi giá trị thuộc tính trong state
16     và đồng thời làm hàm render gọi lại
17     */
18    this.setState({
19      login:true
20    })
21    /*Lưu ý: Không được gán this.state.login = false => gán như vậy hàm render sẽ không được gọi
22     => Giao diện không được render lại
23     */
24  }
25
26  render() {
27    return (
28      <div>
29        {this.state.login ? <b>Hello {this.userName}</b> : <button onClick={this.login}>Login</button>}
30      </div>
31    )
32  }
33 }
```

## ➔ Vậy khi nào nên cần dùng state ?

- ❑ Khi ta cần thực thi 1 sự kiện như là nhấn nút button hay gõ phím ... làm cho các giá trị trên giao diện thay đổi thì ta **đặt giá trị** thuộc tính đó trong state.

# Sử dụng css trong react

## (Styling with Stylesheets - styling with module - working with inline style )

### ✓ Css nhúng (styling with stylesheets)

- Để nhúng css vào component ta dùng cú pháp import → đường dẫn đến file .css từ component đó.
- Lưu ý: css đó sẽ áp dụng cho toàn ứng dụng

```
Styling.css
...
JS Styling.js
...
JS Child.js
```

Đoạn css được nhúng vào component

Dùng import để nhúng css vào

Đoạn text thuộc component Child

Đoạn text thuộc component Styling

```
1 import React, { Component } from 'react' 8.3K (gzipped: 3.3K)
2
3 //Nhúng file css
4 import './Styling.css';
5 import Child from './Child';
6 export default class Styling extends Component {
7   render() {
8     return (
9       <div>
10      <Child />
11      <p className='txt'>Đoạn text thuộc component Styling</p>
12    </div>
13  }
14}
15
16
```

```
1 import React, { Component } from 'react' 8.3K (gzipped: 3.3K)
2
3 export default class Child extends Component {
4   render() {
5     return (
6       <div className="txt">
7         Đoạn text thuộc component Child
8       </div>
9     )
10   }
11}
12
```

- Kết quả:

localhost:3000

Đoạn text thuộc component Child  
Đoạn text thuộc component Styling

# Sử dụng css trong react

## (Styling with Stylesheets - styling with module - working with inline style )

### ✓ styling with module

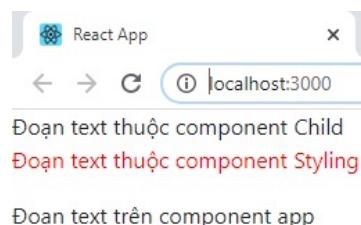
- Là phương pháp đóng gói css theo component, sử dụng dưới dạng import đối tượng giống như component.
- Lưu ý: Khi áp dụng css dưới dạng styling module này, css chỉ được áp dụng cho tại component import các component khác không bị ảnh hưởng.

The screenshot shows a code editor with three files:

- Styling.module.css**: Contains a single rule: `.txt { color: red; }`.
- Styling.js**: Imports the module and defines a component `Styling` that contains a child component `Child` and a paragraph with class `style.txt`.
- Child.js**: Defines a component `Child` that renders a paragraph with class `txt`.

Annotations in pink text and arrows explain the behavior:

- An arrow points from the `style.txt` class in **Styling.js** to the `txt` class in **Child.js**, with the text: "Khi import import dạng đối tượng tương tự như component".
- A callout box points to the `style.txt` class in **Styling.js** with the text: "Khi cần nhúng css ta sử dụng tên đối tượng .class => Lưu ý: Khi nhúng kiểu này css chỉ áp dụng cho component này, các component khác không bị ảnh hưởng".
- A callout box points to the `txt` class in **Child.js** with the text: "Đoạn text thuộc component Child".
- A callout box points to the `txt` class in **Child.js** with the text: "Đoạn text thuộc component Styling".
- A callout box points to the `txt` class in **Child.js** with the text: "css txt không bị ảnh hưởng khi ta sử dụng import dưới dạng file css.module".



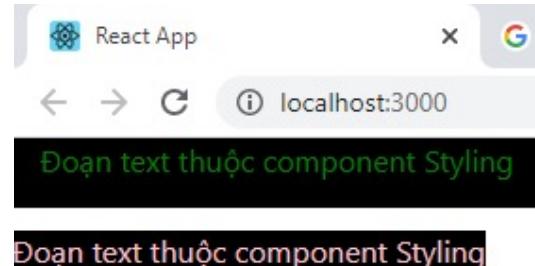
# Sử dụng css trong react

## (Styling with Stylesheets - styling with module - working with inline style )

### ✓ working with inline style

```
JS Styling.js ×
src > Styling > JS Styling.js > ...
1 import React, { Component } from 'react' 8.3K (gzipped: 3.3K)
2 export default class Styling extends Component {
3   render() {
4     const obStyle = {
5       color:'green',
6       backgroundColor:'black',
7       padding:'15px',
8     }
9
10    return (
11      <div>
12        {/*Cách 1: Truyền object Style */}
13        <span style={obStyle}>Đoạn text thuộc component Styling</span>
14
15        <br />
16        <br />
17
18        {/*Cách 2: Truyền trực tiếp object {} */}
19        <span style={{color:'pink',background:'black'}}>Đoạn text thuộc component Styling</span>
20
21      </div>
22
23    )
24  }
25}
```

### ■ Kết quả:



- React cho phép chúng ta style trực tiếp trên thẻ - tuy nhiên với thuộc tính css tương tự dom.style trong js
- Lưu ý: Truyền thông qua style đối với các thuộc tính css có dấu – thì chuyển sang ký tự thứ đầu tiên từ thứ 2 viết hoa.
- Ví dụ:
  - background-color → backgroundColor
  - padding-top → paddingTop
  - border-radius → borderRadius

# Thực hành

The screenshot shows a car configuration interface. On the left, there is a 3D rendering of a black Honda Civic LX. A circular button labeled "360° view" is overlaid on the front-left side of the car. Below the car, the text "CYBERLEARN ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH" is visible. To the right of the car, there is a table with the title "See More LX Features". The table lists the following specifications:

Color	Black
Price	\$19,550
Engine Type	In-Line 4-Cylinder
Displacement	1996 cc
Horsepower (SAE net)	158 @ 6500 rpm
Torque (SAE net)	138 lb-ft @ 4200 rpm
Redline	6700 rpm

On the far right, there is a sidebar titled "Exterior Color" which lists four color options: Crystal Black Pearl, Modern Steel Metallic, Lunar Silver Metallic, and Rallye Red Metallic. Below this is another section titled "Wheels".

## Yêu cầu:

Tạo 1 thư mục bài tập bên trong tạo 1 component BaiTapChonXe.js

Sử dụng state lưu trữ thuộc tính img của hình

Tạo các button chọn màu bên phải tương ứng xử lý state thay đổi

Download resource tại đây

<https://drive.google.com/open?id=1us>

[u44aCL3o-](#)

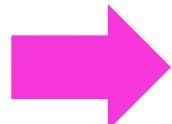
[Lnh9hbXYVwOZMS6cN3Dm](#)

# Render mảng component (Render component with array)

- Đôi lúc thành phần nội dung là mảng component cần render lặp đi lặp lại nhiều lần, reactjs cho phép chúng ta sử dụng vòng lặp để render mảng dữ liệu.
- Tuy nhiên các phần dữ liệu trong mảng khi được render phải được xác định dưới thuộc tính **key** cùng xem ví dụ bên dưới.

## ❖ Từ mảng dữ liệu render ra mảng component <tr>

```
productList = [  
    { id: 1, name: 'black car', price: 'black car', img: './carbasic/products/black-car.jpg' },  
    { id: 2, name: 'red car', price: 'red car', img: './carbasic/products/red-car.jpg' },  
    { id: 3, name: 'silver car', price: 'silver car', img: './carbasic/products/silver-car.jpg' },  
    { id: 4, name: 'steel car', price: 'steel car', img: './carbasic/products/steel-car.jpg' }  
]
```



LIST OF FASHION CARS

id	name	price	img
1	black car	black car	
2	red car	red car	
3	silver car	silver car	
4	steel car	steel car	

# Thực hành

- ☐ Ví dụ cụ thể bên dưới ta có 1 mảng là các đối tượng sản phẩm ta cần render mảng dưới dạng table (các thẻ `<tr>` `</tr>`)
- ☐ Tại phần cần render ta gọi hàm, bên trong hàm xử lý dùng mảng dữ liệu tạo ra các component `<tr>` sau đó tạo 1 mảng đưa các component `<tr>` này vào lưu ý phải có thuộc tính **key**. Cuối cùng return về mảng component.

```
30 render() {  
31   return [  
32     <div className="container">  
33       <h3 className="text-center">LIST OF FASHION CARS</h3>  
34       <table className="table">  
35         <thead>  
36           <tr>  
37             <th>id</th>  
38             <th>name</th>  
39             <th>price</th>  
40             <th>img</th>  
41             <th></th>  
42           </tr>  
43         </thead>  
44         <tbody>  
45           <div>{this.renderTable()}</div>  
46         </tbody>  
47       </table>  
48     </div>  
49   ]  
50 }  
51  
52 }
```

```
1 import React, { Component } from 'react' 8.3K (gzipped: 3.3K)  
2  
3 export default class RenderWithMap extends Component [  
4   productList = [  
5     { id: 1, name: 'black car', price: 'black car', img: './carbasic/products/black-car.jpg' },  
6     { id: 2, name: 'red car', price: 'red car', img: './carbasic/products/red-car.jpg' },  
7     { id: 3, name: 'silver car', price: 'silver car', img: './carbasic/products/silver-car.jpg' },  
8     { id: 4, name: 'steel car', price: 'steel car', img: './carbasic/products/steel-car.jpg' },  
9   ]  
10  
11   renderTable = () => {  
12     let contentTable = [];  
13     for (let index = 0; index < this.productList.length; index++) {  
14       //Mỗi lần duyệt lấy ra 1 sản phẩm  
15       let product = this.productList[index];  
16       //Từ dữ liệu sản phẩm tạo ra 1 tag component <tr> chứa thông tin sản phẩm  
17       let trProduct = <tr key={index}>  
18         <td>{product.id}</td>  
19         <td>{product.name}</td>  
20         <td>{product.price}</td>  
21         <td><img src={product.img} alt={product.name} width={100} height={50} /></td>  
22         <td></td>  
23     </tr>;  
24     //  
25     contentTable.push(trProduct)  
26   }  
27   return contentTable;  
28 }
```

# Thực hành

- ❖ Trong ES6 có 1 phương thức xử lý mảng cho phép ta tạo ra 1 mảng khác từ mảng đó. Ví dụ ta có 1 mảng dữ liệu cần tạo ra 1 mảng component <tr /> Ta có thể ứng dụng nó trong trường hợp này. Cú pháp sẽ ngắn gọn và đơn giản hơn rất nhiều.

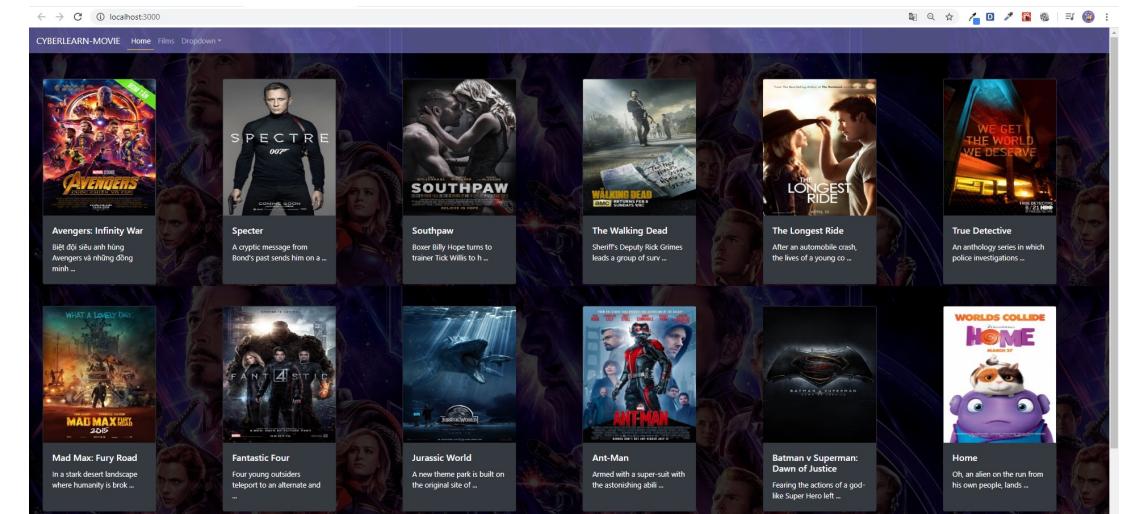
```
11   renderTable = () => {
12     let contentTable = this.productList.map((product, index) => {
13       //Từ dữ liệu sản phẩm tạo ra 1 tag component <tr> chứa thông tin sản phẩm
14       return <tr key={index}>
15         <td>{product.id}</td>
16         <td>{product.name}</td>
17         <td>{product.price}</td>
18         <td><img src={product.img} alt={product.name} width={100} height={50} /></td>
19         <td></td>
20       </tr>;
21     });
22     return contentTable;
23   }
```

# Thực hành

```
let data = [
  {
    "maPhim": 1314,
    "tenPhim": "avarta",
    "biDanh": "avarta",
    "trailer": "https://www.youtube.com/watch?v=EiJ7EsHOC64",
    "hinhAnh": "https://movie0706.cybersoft.edu.vn/hinhanh/avarta_gp01.png",
    "moTa": "hay",
    "maNhom": "GP01",
    "ngayKhoiChieu": "2000-02-11T00:00:00",
    "danhGia": 0
  },
  {
    "maPhim": 1329,
    "tenPhim": "Dao Kinh Hoang",
    "biDanh": "dao-kinh-hoang",
    "trailer": "https://www.youtube.com/embed/IHNzOHi8sJs",
    "hinhAnh": "http://movie0706.cybersoft.edu.vn/hinhanh/dao-kinh-hoang_gp01.jpg",
    "moTa": "Người càng xinh đẹp, càng dễ lừa dối người khác",
    "maNhom": "GP01",
    "ngayKhoiChieu": "2020-11-30T00:00:00",
    "danhGia": 9
  }
]
```

Yêu cầu: Sử dụng mảng dữ liệu và hình ảnh background xây dựng giao diện như hình

<https://drive.google.com/drive/folders/1HV2byzZVIf-WmXsKMHeZ0cKwqgoBjS3m?usp=sharing>



# Tổng hợp kiến thức

## ✓ React binding dữ liệu như thế nào

- Trong react để hiển thị dữ liệu là biến hoặc hàm lên thành phần giao diện (UI) thì ta sử dụng ký tự {} bên trong là lệnh gọi hàm hoặc giá trị biến. Hoặc 1 câu lệnh nào đó trả về 1 giá trị là chuỗi số hoặc jsx (khi trả về phải được chứa bởi duy nhất 1 thẻ bao phủ).
- Ví dụ: {showMessage()}, {hocVien.tenHocvien}, {status === true ? <p>Hello ! Khởi !</p> : <button>Đăng nhập</button>}

## ✓ Trong react để định nghĩa xử lý sự kiện ? Có mấy cách định nghĩa sự kiện (React Handle Event )

- ✓ Tương tự html react định nghĩa sự kiện trực tiếp trên thẻ là các sự kiện như: onClick, onChange, onBlur, onkeyup, onKeyDown ...
- ✓ Có 2 cách định nghĩa sự kiện chính trong react
  - + 1 là định nghĩa trực tiếp bằng arrow function ngay tại thuộc tính sự kiện (Ví dụ: onClick = ()=>{gọi hàm ()};
  - + 2 là dùng callback function để định nghĩa: onClick = {this.tenHamKhiClick}

## ✓ Trong react component có mấy cách nhúng CSS ? Và cách thức nhúng css như thế nào ? (Styling with Componennt)

- ✓ Trong react có 3 cách nhúng css:

- + 1 là import trực tiếp đường dẫn vào thẻ ( Cách này sẽ ảnh hưởng tất cả component nếu cùng selector)
- + 2 là cách import thông qua đối tượng. Ví dụ : import styleObject from 'linkName.module'. Đối với cách này css chỉ ảnh hưởng trực tiếp đến các thẻ được gắn object style.
- + 3 là cách định nghĩa object style rồi trực tiếp truyền vào thuộc tính style của component.

Ví dụ:

```
let objectStyle = {color:'red'};  
<input style={objectStyle} />  
hoặc  
<input style={{color:'red'}} />;
```