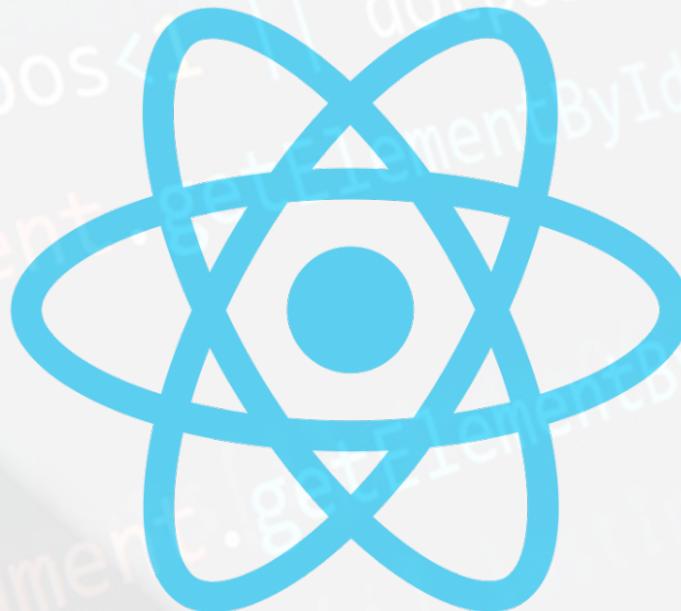
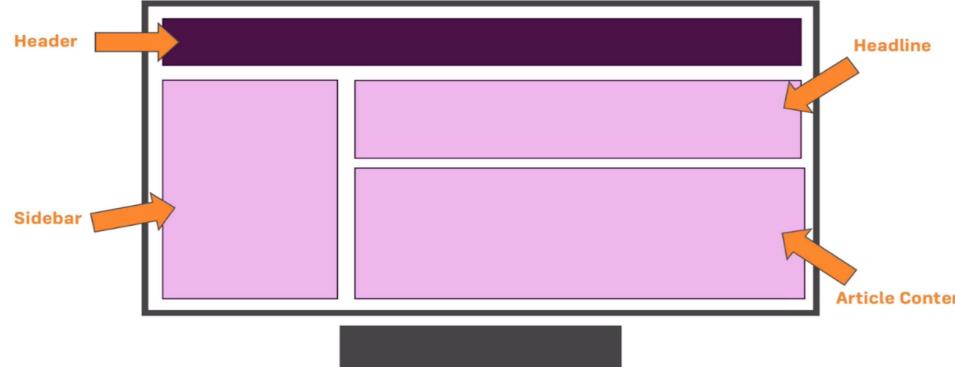


# React js



# Giới thiệu

## Components?



- React xây dựng giao diện ứng dụng theo kiến trúc **component**.
- React hỗ trợ xây dựng SPAs (Single page application).

A screenshot of the TFL Tube Tracker application. On the left, there is a sidebar with a yellow border containing several purple cards, each with a line icon and a 'Stations' button. The main content area has a red border and contains two green cards labeled 'Station Name' and 'Line'. Each card has a title and a subtitle. Below them are two blue cards labeled 'Platform 1' and 'Platform 2', each containing a green 'DepartureBoard' card with a table of train departure information.

A screenshot of a Salesforce Opportunity page for 'Burlington Textiles Weaving Plant Generator'. The page features a 'HIGHLIGHTS PANEL' at the top with account details. A 'SALES PATH BAR' is shown below it. The main content area is divided into several sections: 'ACTIVITY' (containing tabs for 'DETAILS', 'CHATTER', and 'ACTIVITY'), 'TABS' (highlighted with a red box), and 'RELATED LISTS' (containing 'Products', 'Notes & Attachments', and 'Contact Roles'). The 'TABS' section is described as containing three other Lightning Components: DETAILS, CHATTER, and ACTIVITY.

Standard Lightning Components on an Opportunity Page

HIGHLIGHTS PANEL

SALES PATH BAR

ACTIVITY TABS

RELATED LISTS

TABS (TABS is a Lightning Component containing 3 other Lightning Components inside of it: DETAILS, CHATTER and ACTIVITY)

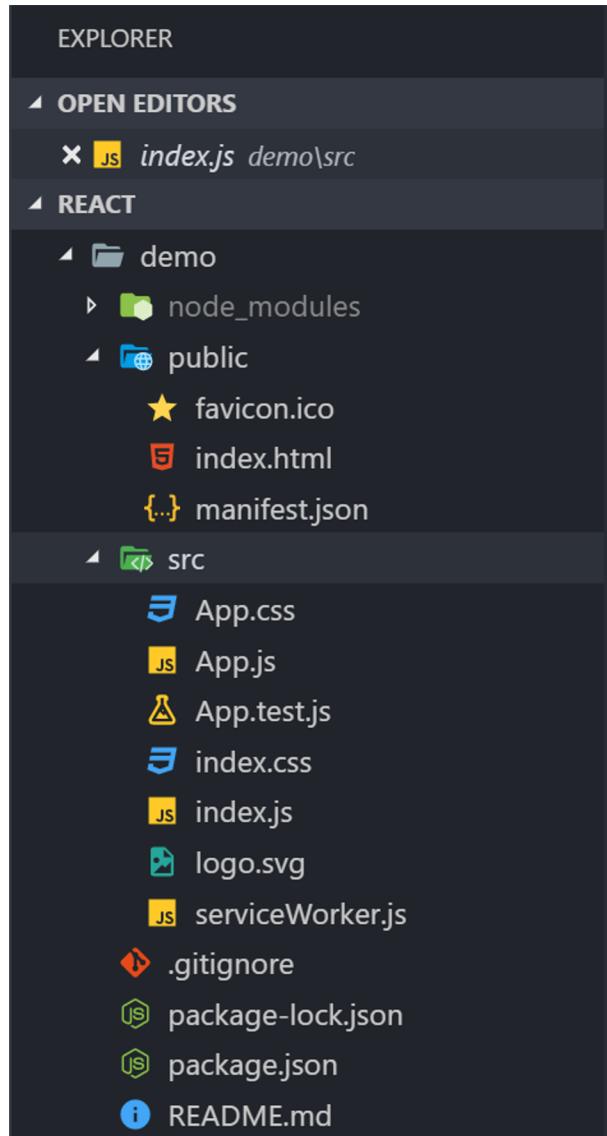
# Cài đặt

- ❑ *Create-react-app* là bộ công cụ tương tự angular-cli giúp ta tạo ra cấu trúc cho project
  - ❑ Cài đặt *tạo dự án với create react app* : `npx create-react-app [tenDuAn]`

```
REACTUI  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL node + ▾ ▷  ^ X  
  
complete:13: command not found: compdef  
macbook@macbooks-MacBook-Pro CreateReactApp % npx create-react-app reactbootcampfe  
npx: installed 67 in 4.15s  
  
Creating a new React app in /Users/macbook/Desktop/CreateReactApp/reactbootcampfe.  
  
Installing packages. This might take a couple of minutes.  
Installing react, react-dom, and react-scripts with cra-template...  
  
(( [██████████] )) :: fetchMetadata: sill resolveWithNewModule workbox-webpack-plugin@6.5.3 checking installable status
```

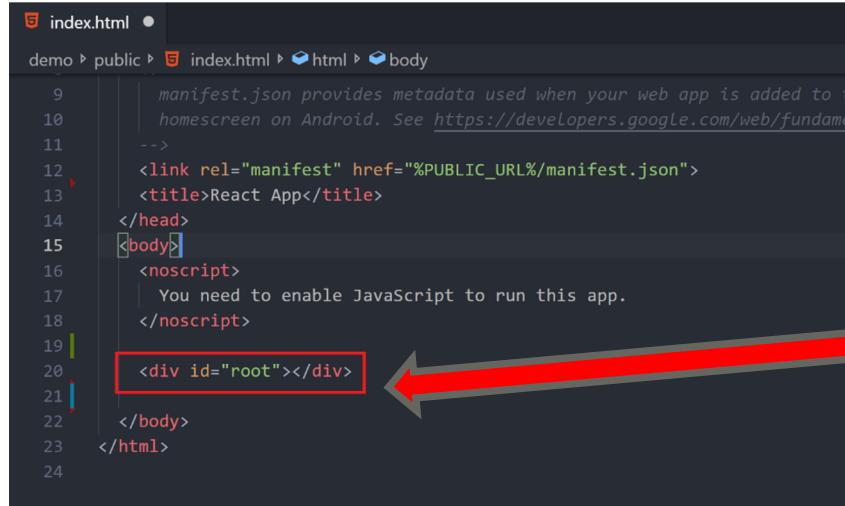
- ## ❑ Chay project: `npm start`

# Cấu trúc thư mục



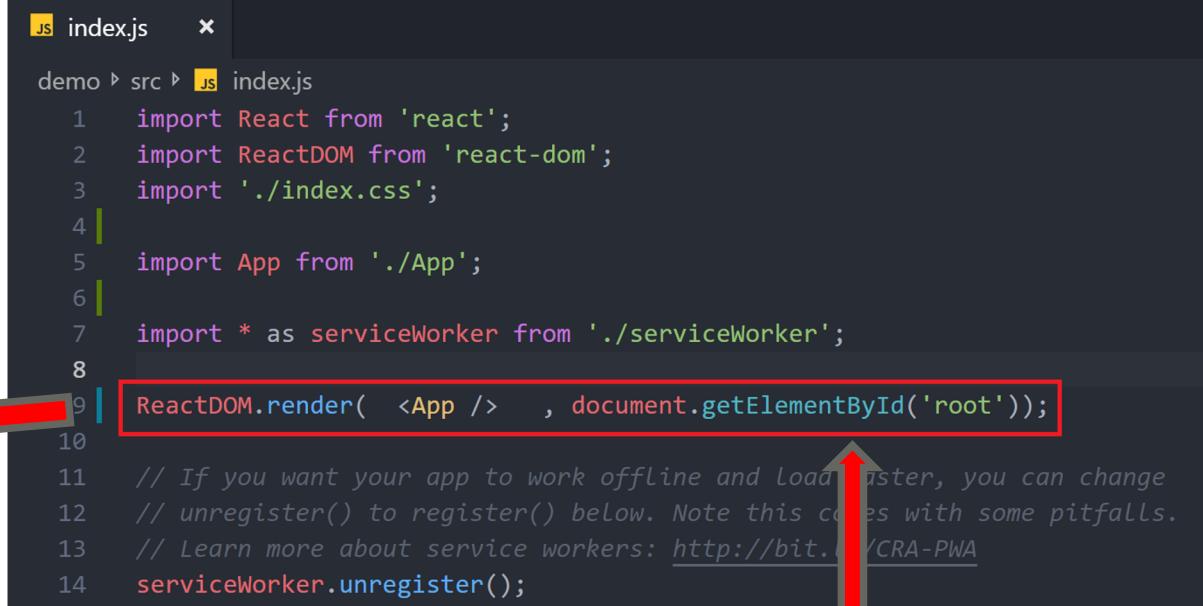
- ❑ ***node\_modules***: chứa các module được cài vào project
- ❑ ***Public***: chứa file index.html và hình ảnh
- ❑ ***Src***: chứa các component của ứng dụng
- ❑ ***Package.json***: lưu lại các thông tin của project và các thư viện được cài vào.

# Luồng đi của ứng dụng



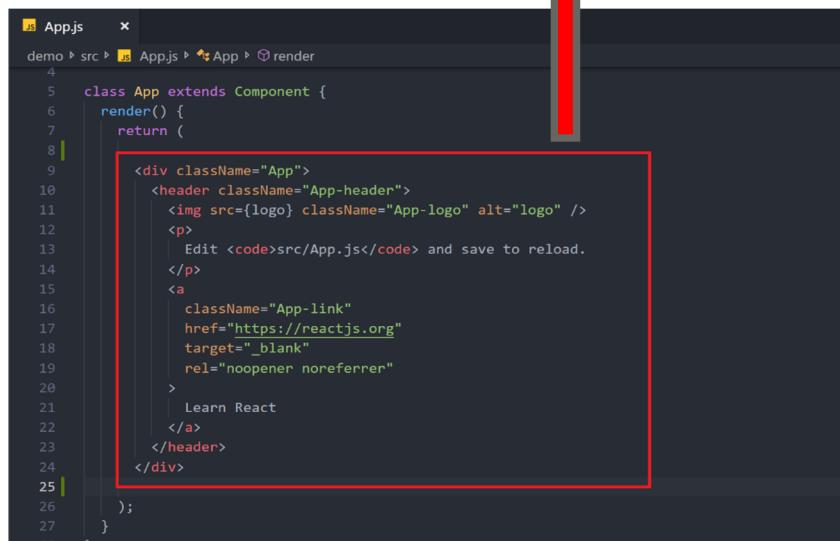
A screenshot of a code editor showing the content of index.html. A red box highlights the div element with the id "root". A red arrow points from this highlighted element in the HTML to the corresponding line of code in index.js.

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



A screenshot of a code editor showing the content of index.js. A red box highlights the ReactDOM.render call. A red arrow points from this highlighted line to the corresponding line in App.js.

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



A screenshot of a code editor showing the content of App.js. A red box highlights the render method of the App component. A red arrow points from this highlighted line to the corresponding line in index.js.

```
App.js
demo > src > 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



## ❑ Mục đích của việc chia component

- ✓ Đóng gói chức năng ứng với giao diện ( dễ tổ chức thư mục)
- ✓ Tái sử dụng (rút ngắn thời gian làm dự án)
- ✓ Dễ bảo trì nâng cấp, phát triển ( fixbug, test , thêm | bớt | thay đổi tính năng)
- ✓ Ngoài ra còn giúp cho việc tổ chức mã nguồn dễ dàng hơn (Code ngắn gọn, component nào nằm ở thư mục đó dễ tìm kiếm – thay thế)
- ✓ Tái sử dụng trên nhiều dự án khác nhau và còn nhiều lợi ích khác nữa.

Example Admin

<List>

actions

filters

<Datagrid>

#	Post Title	Author	Date	Action	Action	Action
1	Post 1	User 1	2023-10-01			
2	Post 2	User 2	2023-10-02			
3	Post 3	User 3	2023-10-03			
4	Post 4	User 4	2023-10-04			
5	Post 5	User 5	2023-10-05			
6	Post 6	User 6	2023-10-06			
7	Post 7	User 7	2023-10-07			
8	Post 8	User 8	2023-10-08			
9	Post 9	User 9	2023-10-09			
10	Post 10	User 10	2023-10-10			

pagination

# 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 (function component)
  - ❑ Stateful component (class component)

```
JS App.js
demo › src › JS App.js › App › render

1 import React, { Component } from 'react';
2
3 import './App.css'; import vào file css dùng để định dạng
4
5 class App extends Component { component thực chất chỉ là một class
6   render() { được kế thừa từ Component của react
7     return (
8       Nội dung giao diện html sẽ được viết tại đây
9     );
10    }
11  }
12 }
13 }
14
15 export default App;
16
```

import React, { Component } from 'react';  
import './App.css'; import vào file css dùng để định dạng  
cho component  
class App extends Component { component thực chất chỉ là một class  
được kế thừa từ Component của react  
render() {  
 return (  
 Nội dung giao diện html sẽ được viết tại đây  
 );  
}  
}  
export default App;

# Thực hành tạo stateful component (rcc)

## ❖ class component

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3
4 class Hello extends React.Component {
5   render() {
6     const {greeting, firstName} = this.props;
7     return (
8       <div>
9         {greeting} {firstName}
10      </div>
11    )
12  }
13}
14
15 export default Hello;
```

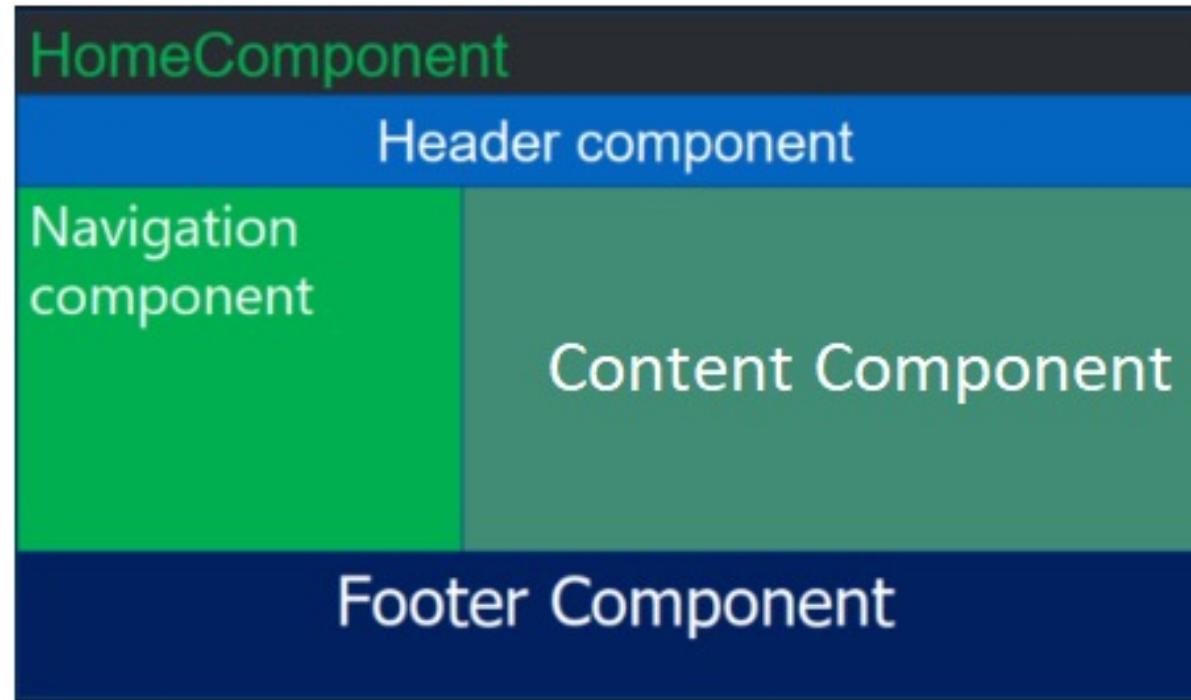
## ❖ functional component

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3
4 function Hello({greeting, firstName}) {
5   return (
6     <div>
7       {greeting} {firstName}
8     </div>
9   )
10 }
11
12 export default Hello;
```

- ✓ Trước khi **react class component** và **react functional component** có nhiều khác biệt do react **class component** có nhiều thuộc tính, phương thức hỗ trợ việc render dữ liệu. Tuy nhiên thời điểm hiện tại React **functional component** cũng đã có các thư viện hooks hỗ trợ các tính năng mà react class có thể làm được.

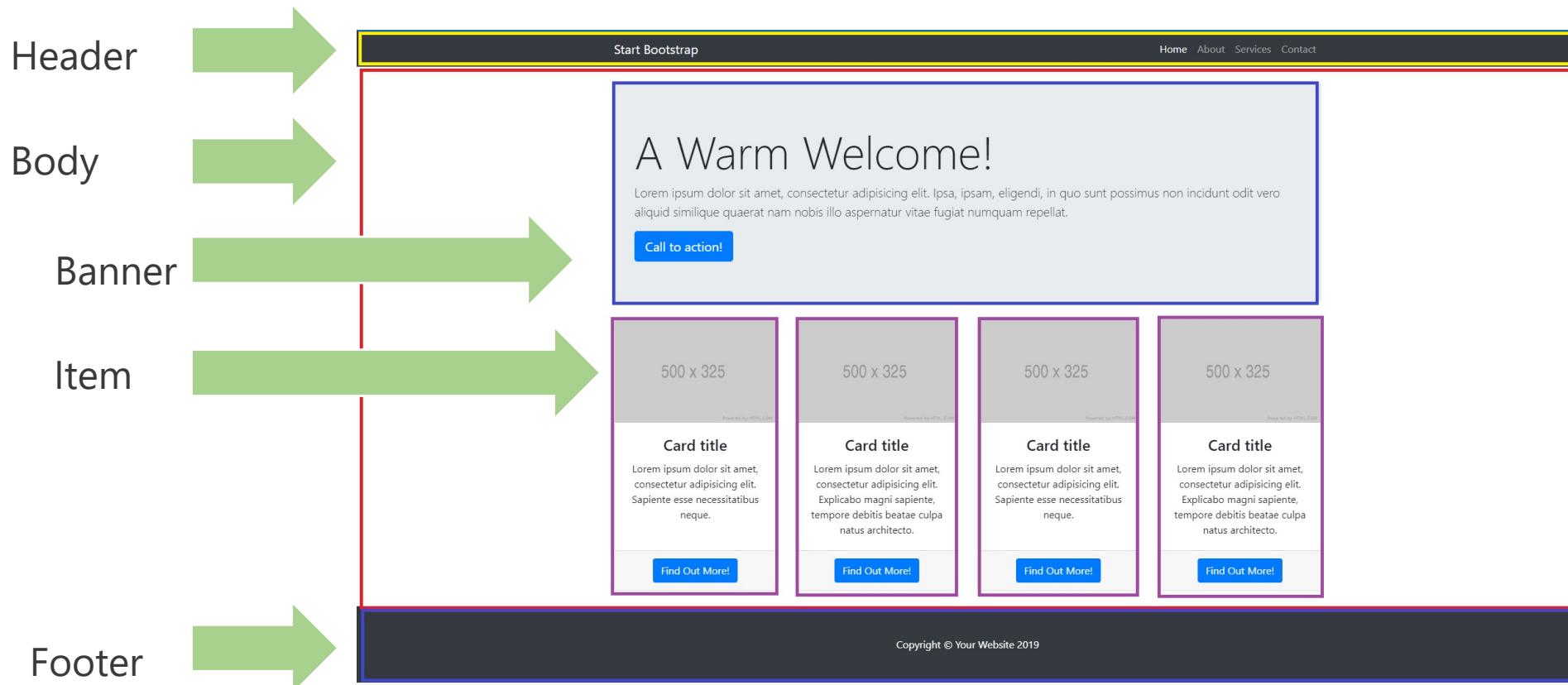
# THỰC HÀNH

- ✓ Sử dụng cdn bootstrap
- ✓ Thực hành 1:
  - ✓ Tạo 1 component **Header** với react class component chứa tiêu đề là cyberlearn.vn
  - ✓ Tạo 1 component **Product** với react functional component nội dung chứa card bootstrap
- ✓ Thực hành 2:
  - ✓ Sử dụng bootstrap tạo các component như hình bên dưới



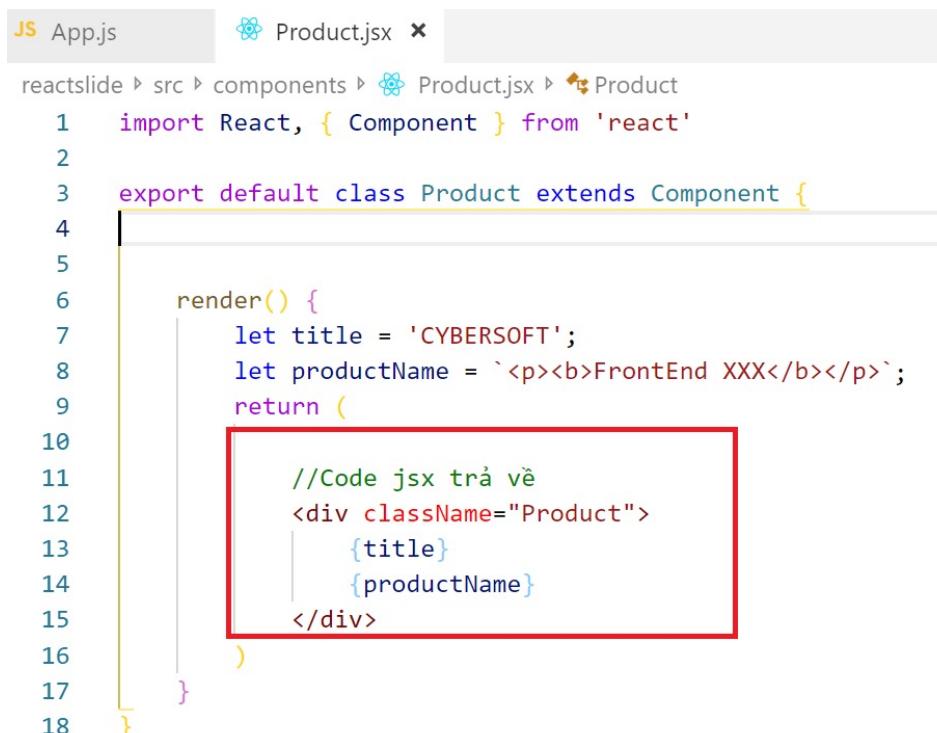
# BÀI TẬP

- ✓ Sử dụng cdn bootstrap
- ✓ Link layout: <https://startbootstrap.com/previews/heroic-features/>
- ✓ Dựa vào layout có sẵn các bạn hãy tạo 1 folder BaiTapLayoutComponent
- ✓ Yêu cầu tạo 1 component tên <BaiTapThucHanhLayout> Bên trong tổ chức nội dung các component như hình bên dưới



# BINDING DỮ LIỆU TRONG REACT

- 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 }
```

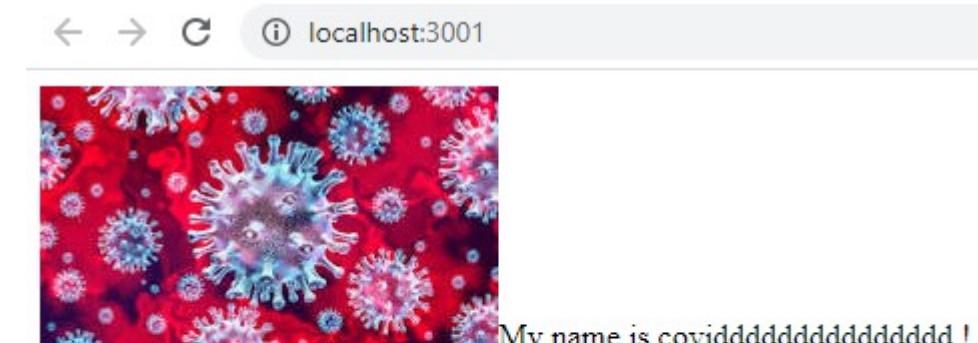


```
JS App.js          JS Product.jsx x
reactslide > src > components > Product.jsx > ...
1 import React, { Component } from 'react'
2
3 export default class Product extends Component {
4
5   //Tạo ra 1 phương thức trả về 1 đoạn jsx
6   renderProduct = () => {
7     let title = 'CYBERSOFT';
8     let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

✓ 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 covidddddddddd !'
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}
```



id: covid19  
name: covid19  
alias: covid19  
desc: chinese virus

Kết quả hiển thị

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

# XỬ LÝ SỰ KIỆN (HANDLE EVENT)

- ❑ 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 () .

## React class component

```
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 }
```

## React functional component

```
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 object hay function đều được khai báo dưới dạng biến là let var hay const*/
6   const handleClick = () => {
7     alert("Hello Khải !!!")
8   }
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24 }
```

/\*
 Khi gọi hàm không cần con trỏ this bởi vì đơn giản component này không phải là lớp đối tượng mà ta sử dụng nó như là 1 function bình thường thôi
 (Bên trong function thì ta khai báo và sử dụng biến bình thường thôi hehe khai báo this chỉ cho rườm rà nè !)
 Tuy nhiên vẫn sử dụng được this nhé (nhưng không cần thiết)
\*/

# TRUYỀN THAM SỐ QUA SỰ KIỆN (Passing Arguments to Event Handlers)

```
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    }
```

# TỔNG HỢP KIẾN THỨC

- ✓ Component có 2 loại:
  - ✓ RFC (REACT FUNCTION COMPONENT)
  - ✓ RCC (REACT CLASS COMPONENT)
- ✓ Tạo và sử dụng component
  - ✓ Tạo file viết hoa chữ cái đầu tiên
  - ✓ Sử dụng như thẻ html : <Demo /> hoặc <Demo> </Demo>
- ✓ Tạo và sử dụng component
  - ✓ Tạo file viết hoa chữ cái đầu tiên
  - ✓ Sử dụng như thẻ html : <Demo /> hoặc <Demo> </Demo>
- ✓ Chuyển đổi nội dung từ html -> jsx :
  - ✓ Tô đen chọn convert html to jsx
  - ✓ Nội dung component phải được bao phủ bởi 1 thẻ. Thường dùng thẻ <div> hoặc <Fragment />
- ✓ Databinding
  - ✓ Binding dữ liệu xuống jsx thông qua {}
  - ✓ Có thể binding : biến và hàm hoặc thuộc tính hay phương thức