Ghi nho: Bien trong react thi se nam trong {}

Cau truc cua mot component

class Clock extends React.Component {

constructor(props) {

super(props);

this.state = {date: new Date()};

}

render() {

return (

<div>

<h1>Hello, world!</h1>

<h2>It is {this.state.date.toLocaleTimeString()}.</h2>

</div>

);

}

}

Cấu trúc tương tự như angular, phần code sẽ nằm trong src

Có những bài học sau:

1. Hello world.

Như vậy sẽ có một nới đặt hàm element và một file đặt reactDOM.render( );

Trong file App.tsx sẽ đặt thẻ element với cú pháp là

**import** React **from 'react'**;  
**import** logo **from './logo.svg'**;  
**import './App.css'**;  
  
**const** *App*: React.FC = () => {  
 **return** (  
 <**h1**>Hello, world!</**h1**>  
 );  
}  
  
**export default** *App*;

và trong file index.tsx sẽ render ra màn hình đó với code là

**import** React **from 'react'**;  
**import** ReactDOM **from 'react-dom'**;  
**import './index.css'**;  
**import** App **from './App'**;  
**import** \* **as** serviceWorker **from './serviceWorker'**;  
  
ReactDOM.render(<**App** />, document.getElementById(**'root'**));%dịch nghĩa là lấy từ trong App phần tử hiền thị trên màn hình là ‘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: https://bit.ly/CRA-PWA*serviceWorker.unregister();

1. Introducing JSX. Nơi code để hiện thị lên giao diện.
2. Rendering Elements. Element hiển thị trong screens.

*setInterval*(function,time): Thời gian hiển thị element

**function** *tick*() {  
 **const** element = (  
 <**div**>  
 <**h1**>Hello, world!</**h1**>  
 <**h2**>It is {**new *Date***().toLocaleTimeString()}.</**h2**>  
 </**div**>  
 );  
 ReactDOM.render(element, ***document***.getElementById(**'root'**));  
}  
  
*setInterval*(*tick*, 2000);

Hiểu rằng thời gian đê render ra element là 2000 ms.

1. Components and Props. Chia screens thành nhiều screen nhỏ hơn.
2. State and lifecycle: trang thái và vòng đời của trạng thái.

Khiu refresh lai trang thi ham componentWillMount se duoc goi truoc khi render, sau do se chay ham render, ham componentDidMount se duoc goi sau khi render compenent, noi nay thuc hien cac ham AJAX, update state duoc thuc thi ta day,day cung la noi ket noi voi database hoac cac method ben phia server.

Nhu vay, React se follow theo trinh tu sau day

* Khoi taoj class ke thua tu component
* Khoi tao gia tri mac dinh cho Props
* Khoi tao gia tri mac dinh cho State
* Goi ham componentWillMount()
* Goi ham render()
* Goi ham componentDidMount

class Clock extends React.Component {

constructor(props) {

super(props);

this.state = {date: new Date()};

}

componentDidMount() {

this.timerID = setInterval(

() => this.tick(),

1000

);

}

componentWillUnmount() {

clearInterval(this.timerID);

}

tick() {  
 **this**.setState({  
 **date**: **new *Date***()  
 });  
}

render() {

return (

<div>

<h1>Hello, world!</h1>

<h2>It is {this.state.date.toLocaleTimeString()}.</h2>

</div>

);

}

}

1. Handling Evens.
2. Conditionnal Rendering. Điều kiện để rendering màn hình.
3. **class** LoginControl **extends** React.Component {  
    constructor(props) {  
    **super**(props);  
    **this**.handleLoginClick = **this**.handleLoginClick.bind(**this**);  
    **this**.handleLogoutClick = **this**.handleLogoutClick.bind(**this**);  
    **this**.**state** = {**isLoggedIn**: **false**};  
    }  
     
    handleLoginClick() {  
    **this**.setState({**isLoggedIn**: **true**});  
    }  
     
    handleLogoutClick() {  
    **this**.setState({**isLoggedIn**: **false**});  
    }  
     
    render() {  
    **const** isLoggedIn = **this**.**state**.**isLoggedIn**;  
    **let** button;  
     
    **if** (isLoggedIn) {  
    button = <**LogoutButton onClick=**{**this**.handleLogoutClick} />;  
    } **else** {  
    button = <**LoginButton onClick=**{**this**.handleLoginClick} />;  
    }  
     
    **return** (  
    <**div**>  
    <**Greeting isLoggedIn=**{isLoggedIn} />  
    {button}  
    </**div**>  
    );  
    }  
   }  
     
   **function** *UserGreeting*(props) {  
    **return** <**h1**>Welcome back!</**h1**>;  
   }  
     
   **function** *GuestGreeting*(props) {  
    **return** <**h1**>Please sign up.</**h1**>;  
   }  
     
   **function** *Greeting*(props) {  
    **const** isLoggedIn = props.**isLoggedIn**;  
    **if** (isLoggedIn) {  
    **return** <**UserGreeting** />;  
    }  
    **return** <**GuestGreeting** />;  
   }  
     
   **function** *LoginButton*(props) {  
    **return** (  
    <**button onClick=**{props.**onClick**}>  
    Login  
    </**button**>  
    );  
   }  
     
   **function** *LogoutButton*(props) {  
    **return** (  
    <**button onClick=**{props.**onClick**}>  
    Logout  
    </**button**>  
    );  
   }  
     
   ReactDOM.render(  
    <**LoginControl** />,  
    ***document***.getElementById(**'root'**)  
   );
4. Lists and Keys.

function NumberList(props) {

const numbers = props.numbers;

const listItems = numbers.map((number) =>

<li>{number}</li>

);

return (

<ul>{listItems}</ul>

);

}

const numbers = [1, 2, 3, 4, 5];

ReactDOM.render(

<NumberList numbers={numbers} />,

document.getElementById('root')

);

1. Forms.
2. Lifting State Up.
3. Compostion và Inheritance.
4. Thinking in React.