React.lazy: Chuc nang nay cho phep ban render mot import dong nhe mot component binh thuong

VD: Neu import binh thuong thi nhu sau

import OtherComponent from './OtherComponent';

con neu dung React.lazy thi la

const OtherComponent = React.lazy(() => import('./OtherComponent'));

const OtherComponent = React.lazy(() => import('./OtherComponent'));

function MyComponent() {

return (

<div>

<Suspense fallback={<div>Loading...</div>}>

<OtherComponent />

</Suspense>

</div>

);

}

Trong khi chờ đợi load lazy component thì sẽ chạy fallback

Nếu Othercomponent không thể tải lên, nó sẽ kích hoạt lỗi. Bạn có thể tạo Error Boudary, bạn sử dụng nó bất kì nơi nào bên trên lazy components để hiển thị thông báo lỗi.

Route trong react.

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import React, { Suspense, lazy } from 'react';

const Home = lazy(() => import('./routes/Home'));

const About = lazy(() => import('./routes/About'));

const App = () => (

<Router>

<Suspense fallback={<div>Loading...</div>}>

<Switch>

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

<Route path="/about" component={About}/>

</Switch>

</Suspense>

</Router>

);

React.lazy hiện tại chỉ hỗ trợ defaul export. Nếu bạn muốn import sử dụng named export, tạo một module trung gian và sau đó export dưới dạng export default.

Context: Thông thường một ứng dụng react, dât được truỳên từ component cha đến component con, tuy nhiên nếu dùng Context thì chia sẻ data khi chúng là “global data” của toàn bộ ứng dụng React, chẳng hạn như thông tin về user đang đăng nhập

ErrorBoundary: thông báo khi bị lỗi

class MyComponent extends React.Component {

constructor(props) {

super(props);

this.state = { error: null };

this.handleClick = this.handleClick.bind(this);

}

handleClick() {

try {

// Do something that could throw

} catch (error) {

this.setState({ error });

}

}

render() {

if (this.state.error) {

return <h1>Caught an error.</h1>

}

return <div onClick={this.handleClick}>Click Me</div>

}

}

Sử dụng như sau

<ErrorBoundary>

<MyWidget />

</ErrorBoundary>

Higher-Order Components: là một hàm lấy một component và trả về một commponent.

Kĩ thuật này sẽ xem lạ sau.

**Integrating with Other Libraries: tích hợp với các thư viện khác.**

Ref: dùng để tham chiếu tời một phần tử. Về căn bản, nên tránh sử dụng nó trong hầu hết các trường hợp.

Không dùng ref

export default class index extends React.Component {

constructor() {

super();

this.state = {sayings: "" };

}

update(e) {

this.setState({sayings: e.target.value});

}

render() {

return (

<div>

Bob says <input type="text" onChange={this.update.bind(this)} />

{this.state.sayings}

</div>

);

}

}

Nếu dùng ref thì

export default class index extends React.Component {

constructor() {

super();

this.state = {sayings: "" };

}

update(e) {

this.setState({sayings: this.refs.btalks.value});

}

render() {

return (

<div>

Bob says <input type="text" ref="btalks" onChange={this.update.bind(this)} />

{this.state.sayings}

</div>

);

}

}

Như bạn thấy, chúng ta đang sử dụng react.Fragment làm thẻ bao cho 3 divs đó

class ThreeDivsFragments extends React.Component {

render() {

return (

<React.Fragment>

<div>First</div>

<div>Second</div>

<div>Third</div>

</React.Fragment>

);

}

}

Nếu để như sau

class ThreeDivsOrdinary extends React.Component {

// Báo lỗi ngay vì không có element cha

render() {

return (

<div>First</div>

<div>Second</div>

<div>Third</div>

);

}

}

Thì sẽ bào lỗi vì trông có element cha, nếu không muốn thêm một thẻ div cha thì nên sử dụng <React.Fragment>.

Cú pháp ngắn cho Fragments là <>, vì vậy component Columns cũng có thể viết như sau:

Sử dụng một mảng list thì ta có

function Glossary(props) {

return (

<dl>

{props.items.map(item => (

// Nếu không có `key`, React sẽ báo warning về key

<React.Fragment key={item.id}>

<dt>{item.term}</dt>

<dd>{item.description}</dd>

</React.Fragment>

))}

</dl>

);

}

HOC: Higher-Order Components nhận một componet và trả về một component.

JSX In Depth: thêm component vào trong code js.

Cú pháp

React.createElement(

MyButton,

{color: 'blue', shadowSize: 2},

'Click Me'

)

# Optimizing Performance trong ReactJS

Kĩ thuật nhằm tổi thiểu số lượng các DOM được thực thi để update UI. Một lát nữa tìm hiểu thêm.

**componentWillMount** đây là method sẽ được thực thi trước khi 1 component được render trên cả server side và client side.

**componentDidMount** method này được thực thi khi 1 component được render trên client side. Đây là nơi các hàm AJAX requests, DOM or update state được thực thi. Method này cũng đucợ sử dụng để kết nối với các JS Framework khác và các function với delayed execution như setTimeout or setInterval.

**componentWillReceiveProps** sẽ được thực thi ngay khi thuộc tính props (tìm hiểu [props](http://5minuteshack.blogspot.com/2018/03/bai-5-reactjs-props-overview.html) là gì?) được update và trước khi component được render lại. Trong ví dụ dưới, ta sẽ sử dung method này vùng với setNewNumber để update state.

**shouldComponentUpdate** sẽ trả về kết quả true or false. Phương thức này sẽ xác định 1 component có được update hay không. Mặc định giá trị này là true. Nếu bạn không muốn component render lại sau khi update state hay props thì return giá trị thành false. Các bạn xem ví dụ dưới để hiểu rõ.

**componentWillUpdate** được gọi khi chúng ta update state của component trước khi nó render lại.

**componentDidUpdate** sau khi componentWillUpdate ở trên được gọi xong thì đến lượt thằng này được goi.

**componentWillUnmount** được gọi khi chúng ta unmout 1 component kiểu như xóa nó khỏi react.

# Strict Mode

Sẽ tìm hiểu quá trình sử dụng sau.