REACTJS

Hà Nội, ngày 29 tháng 12 năm 2023

BUỔI 2

Tìm hiểu về cấu trúc JSX

- Structure
- Cấu trúc cơ bản của JSX
- Các thuộc tính của phần tử
- Các hàm tạo thành phần tử
- Các biến trong JSX

Tạo và sử dụng component

- Khái niệm về component
- Cách tạo component
- Các loại component
- Cách sử dụng component

I. Tìm hiểu về cấu trúc và JSX

Cấu trúc project reactjs

I. Tìm hiểu về cấu trúc và JSX

I.1. Các đặc điểm của JSX

- JSX (JavaScript XML) là một phần quan trọng trong ReactJS, được sử dụng để định nghĩa cấu trúc giao diện người dùng trong React
- JSX không phải là HTML
- JSX cần được chuyển đổi thành JavaScript
- JSX có thể được sử dụng trong các hàm tạo thành phần tử

I. Tìm hiểu về cấu trúc và JSX

I.2 Cấu trúc JSX

```
const element = <h1>Hello, world!</h1>;
```

I.3 Các thành phần và thuộc tính của JSX

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;
```

```
<h1>The current time is {neDate().toLocaleString()}</h1>
```

II. Tạo và sử dụng component

II.1. Khái Niệm về Component:

- Trong React, một component là một phần độc lập và có thể tái sử dụng của giao diện người dùng.
- Mỗi component đều có trách nhiệm riêng biệt và có thể được kết hợp để tạo thành giao diện người dùng phức tạp hơn.
- Component trong React giúp tách biệt logic và hiển thị, làm cho quá trình phát triển ứng dụng trở nên dễ dàng quản lý và bảo trì hơn.

II. Tạo và sử dụng component

II.2. Cách tạo một component

Functional Components:

- Functional components được viết dưới dạng hàm JavaScript và được sử dụng chủ yếu để render giao diện.
- Chúng không có state và lifecycle methods (trước React 16.8).

```
function Greet (props) {
  return <p>Hello, {props.name}!</p>;
}
```

II. Tạo và sử dụng component

II.2. Cách tạo một component

Class Components:

- Class components là các class JavaScript và được sử dụng khi bạn cần sử dụng state và lifecycle methods.
- Các component class có thể truy cập các thuộc tính và phương thức của lớp React.Component.

```
class Greet extends React.Component {
  render() {
    return <p>Hello, {this.props.name}!</p>;
  }
}
```

II. Tạo và sử dụng component

II.3. Các loại component (Cách tạo)

- Class component
- Function component

II. Tạo và sử dụng component

II.3. Các loại component (Chức năng)

Presentational Components (Dumb Components):

- Presentational components chủ yếu đảm nhiệm việc hiển thị UI và nhận dữ liệu thông qua props.
- Chúng không quản lý state hoặc thực hiện logic nghiệp vụ phức tạp.

Container Components (Smart Components):

- Container components chịu trách nhiệm quản lý state, xử lý logic nghiệp vụ, và truyền dữ liệu xuống các presentational components.
- Chúng thường gọi các actions và là nơi lưu trữ state của ứng dụng.

II. Tạo và sử dụng component

II.4. Cách sử dụng component

Để sử dụng một component trong React, bạn cần tạo một phần tử React và truyền component làm giá trị cho thuộc tính `component` của phần tử đó.

Tạo component (Fc component or Class component

```
// Class Component
class Greet extends React.Component {
  render() {
    return <p>Hello, {this.props.name}!</p>;
  }
}
```

II. Tạo và sử dụng component

II.4. Cách sử dụng component

Sử dụng component

```
// Sử dụng Functional Component
const App = () => {
  return (
    <div>
      <Greet name="John" />
      <Greet name="Jane" />
    </div>
  );
};
```

II. Tạo và sử dụng component

II.4. Cách sử dụng component

- **Self-Closing Tag**
  - Là một cách viết ngắn gọn được gọi là self-closing tag hoặc void element.
  - Đây là cách thường được sử dụng khi bạn không có nội dung bên trong component.

```
<ButtonComponent onClick={handleClick} />
```

- **Opening và Closing Tags**
  - Là cách viết truyền thống với opening tag và closing tag được sử dụng khi bạn có nội dung bên trong component.

```
<ButtonComponent>Click me</ButtonComponent>
```

  - Sử dụng cặp opening và closing tags là quan trọng khi bạn muốn đặt nội dung bên trong component, chẳng hạn như văn bản, các components khác, hoặc bất kỳ nội dung HTML/JSX nào.

II. Tạo và sử dụng component

II.4. Cách sử dụng component

- **Fragment**

  Fragment là một tính năng trong React giúp nhóm nhiều phần tử con mà không tạo thêm một phần tử ngoại vi

```
//Syntax
<>
    <ButtonComponent onClick={handleClick} />
</>
or
<Fragment>
    <ButtonComponent onClick={handleClick} />
</Fragment>
```

II. Tạo và sử dụng component

II.4. Cách sử dụng component

- **Rendering**
  - Rendering trong React là quá trình chuyển đổi các components React thành các phần tử UI hiển thị trên trình duyệt.
  - Khi một component được tạo hoặc có sự thay đổi trong state và props, React sẽ thực hiện quá trình rendering để cập nhật giao diện người dùng.

II.4. Cách sử dụng component

- **Rendering**
  - Phương thức render
  - Cơ Chế Reconciliation
  - Vitual-DOM
  - Re-rendering Components
  - Unidirectional Data Flow
  - Conditional Rendering

THANK YOU

FOR

WATCHING