REACTJS

Hà Nội, ngày 29 tháng 12 năm 2023

I. Conditional Rendering

Conditional rendering trong ReactJS là quá trình hiển thị nội dung khác nhau trên giao diện người dùng dựa vào điều kiện cụ thể.

Sử Dụng If Statements

```
import React from 'react';

const MyComponent = ({ condition }) => {
    if (condition) {
        return <div>This content is rendered if the condition is true.</div>;
    } else {
        return <div>This content is rendered if the condition is false.</div>;
    }
};
};
```

I. Conditional Rendering

Sử Dụng Toán Tử &&:

Toán tử && có thể được sử dụng để kiểm tra điều kiện và render JSX một cách ngắn gọn:

```
import React from 'react';

const MyComponent = ({ condition }) => {
        return (
                <div>
                        {condition && <p>This content is rendered if the condition is
                    true.</p>}
                </div>
        );
};
```

I. Conditional Rendering

Sử Dụng Toán Tử ? : (Ternary Operator):

Toán tử ? : còn được gọi là ternary operator, giúp viết điều kiện và render JSX

```
import React from 'react';

const MyComponent = ({ condition }) => {
        return (
                <div>
                        {condition
                         ? <p>This content is rendered if the condition is true.</p>
                         : <p>This content is rendered if the condition is false.</p>}
                </div>
        );
};
```

I. Conditional Rendering

Lưu Ý:

1. **Người Dùng && để Tránh Rerender Không Cần Thiết:**

   Khi sử dụng &&, chỉ có một phần của JSX sẽ được render nếu điều kiện là false. Điều này giúp tránh rerender không cần thiết và cải thiện hiệu suất.

2. **Kiểm Tra Đúng null và undefined:**

   Nếu điều kiện là null hoặc undefined, React sẽ không hiển thị gì cả.

3. **Cân Nhắc Sử Dụng Toán Tử ? : Cho Các Biểu Thức Phức Tạp:**

   Khi điều kiện và phần thì của bạn phức tạp, sử dụng toán tử ? : có thể làm cho mã nguồn dễ đọc hơn.

II. Lists and Keys

II.1. Lists trong React:

Để render một danh sách trong React, bạn có thể sử dụng hàm `map` để chuyển đổi mỗi phần tử trong mảng thành một phần tử JSX.

```
//Render danh sách
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) => <li key={number}>{number}</li>);
```

```
//Render Components trong danh sách
const items = ['item1', 'item2', 'item3'];
const listComponents = items.map((item) => <MyComponent key={item} value={item} />);
```

II. Lists and Keys

II.1. Keys trong React:

- **Key là gì?**

  Key là một thuộc tính đặc biệt được sử dụng trong React để giúp React xác định những phần tử nào đã thay đổi, thêm mới, hoặc bị xóa. Key phải là duy nhất trong cùng một cấp.

- **Tại sao cần sử dụng Key?**

  Keys giúp React hiểu rõ về sự thay đổi trong danh sách, giúp tối ưu hóa việc render lại và cập nhật UI một cách hiệu quả

II. Lists and Keys

Lưu Ý:

- Keys chỉ cần là duy nhất trong phạm vi cấp cha.
- Keys không được truyền vào component qua props, mà được sử dụng bởi React nội bộ để theo dõi các phần tử.
- Keys không có nghĩa là phải là số, có thể là bất kỳ kiểu dữ liệu nào có thể được sử dụng làm định danh duy nhất.
- Keys khi sử dụng với fragment

THANK YOU

FOR

WATCHING