REACTJS

Hà Nội, ngày 29 tháng 12 năm 2023

I. React Hook

- React Hooks là một tính năng của React từ phiên bản 16.8 trở lên, giúp bạn sử dụng các tính năng của React như state và lifecycle trong các hàm component(functional components) thay vì chỉ trong các class component.
- Hooks giúp làm cho việc quản lý trạng thái và các side effect trở nên dễ dàng hơn và giúp bạn viết code tái sử dụng.

I. React Hook

Đặc điểm của React hook

Cho Phép Sử Dụng State và Lifecycle Trong Functional Components

Giảm Sự Phức Tạp của Class Components

Dễ Kiểm Soát Side Effects

Các Điều Cần Biết

- Hook nên được gọi tại cấp độ top
- Các Rules of Hooks
- Custom Hooks

I. React Hook

Các loại React Hook

- useState
- useEffect
- useRef
- useMemo
- useCallback
- useContext
- useReducer

I. React Hook

I.1. useState

là một trong những React Hook quan trọng nhất, được sử dụng để quản lý trạng thái trong functional components của React.

Syntax:

```
const [state, setState] = useState(initialState);
```

- state: Giá trị trạng thái hiện tại.
- setState: Hàm để cập nhật giá trị trạng thái.
- initialState: Giá trị khởi tạo (Number, String, Boolean…)

I. React Hook

I.2. useEffect

useEffect là một trong những hooks quan trọng trong thư viện React để thực hiện các side effects trong các components React. Side effects có thể bao gồm các tác vụ như:

- Call API,
- Thay đổi DOM trực tiếp,
- Đăng ký và hủy đăng ký các sự kiện, và nhiều công việc khác.

I. React Hook

```
import React, { useEffect } from 'react';

function ExampleComponent() {
        useEffect(() => {
                // Các side effect ở đây
                // Cleanup function (nếu cần)
                return () => {
                        // Thực hiện các tác vụ cleanup, hủy đăng ký sự kiện, v.v.
                };
        }, [/* dependencies */]);
        return <div>Example Component</div>;
}
```

I. React Hook

I.2. useEffect

Sử dụng:

- **Fetch Data**: Khi bạn muốn gọi API để lấy dữ liệu khi component được render.
- **Subscribe to Something**: Khi bạn muốn đăng ký sự kiện, như theo dõi thay đổi trong state hoặc props.
- **Manipulate DOM**: Khi bạn cần thay đổi trực tiếp DOM sau khi component được render.
- **LifeCycle:** xử lý lifecycle useEffect

I. React Hook

I.2. useEffect

Cần chú ý khi sử dụng useEffect:

- **Cleanup Function**: Nếu useEffect trả về một hàm, hàm đó sẽ được chạy khi component bị unmount hoặc khi dependencies thay đổi và useEffect chạy lại.
- **Dependencies Array**: Chú ý các cách sử dụng với dependencies.
- **Vòng lặp vô tận (Infinite Loop)**: Hãy tránh tình trạng khi useEffect tạo ra một vòng lặp vô tận.

I. React Hook

I.3. useRef

useRef là một trong những hooks của React được sử dụng để tạo và giữ một tham chiếu (reference) đến một phần tử DOM hoặc một giá trị giữ nguyên giữa các render của component. Nó cung cấp một cách để tương tác với các phần tử DOM mà không gây re-render cho component.

Khi nào sử dụng useRef:

- **Thao tác với Phần Tử DOM**: Khi bạn cần thao tác trực tiếp với một phần tử DOM, như là gọi hàm focus(), scroll, hoặc thay đổi một thuộc tính không ảnh hưởng đến việc render.
- **Lưu Trữ Dữ Liệu Giữa Các Lần Render**

I. React Hook

Cần chú ý khi sử dụng useRef:

- Không Gây Re-render:
- Thay Đổi Giá Trị Ref Không Kích Hoạt Render:
- Sử Dụng Khi Cần Thiết:

```
import React, { useRef, useEffect } from 'react';

function ExampleComponent() {
        const myRef = useRef(null);
        useEffect(() => {
          myRef.current.focus();   // Sử dụng ref để thao tác với phần tử DOM
        }, []);
        return <input ref={myRef} />;
}
```

THANK YOU

FOR

WATCHING