Hiểu về hook useRef của React như thế nào cho đúng

Bài viết được sự cho phép của tác giả Lưu Bình An

Câu nói chào hàng của `useState` vẫn thường được nghe: thêm state vào trong function component.

> **Làm animate siêu đơn giản với hook khi react component mount và unmount**

> **React hook là gì và lợi ích mà React hook đem lại**

```
const [value, setValue] = React.useState("init value");
```

Giả dụ tình huống là thế này, bạn làm gì đó mà nó ko liên quan đến UI, **không cần re-render**, nhưng vẫn muốn giá trị này cố định giữa các lần render? `useState` có thể cố định giá trị, nhưng ngặt nỗi nó sẽ trigger re-render nếu bị thay đổi

```
function usePersistentValue(initValue) {
  return React.useState({
    current: initialValue,
  })[0];
}
```

Vì chúng ta không muốn trigger re-render, nên chỉ trả về giá trị của state (phần tử đầu tiên trong mảng), không trả về hàm để cập nhập nó.

Vẫn còn chưa rõ ràng lắm nhỉ, thí dụ trong trong ứng dụng chúng ta muốn có một giá trị counter tăng lên 1 từng giây, một button để stop việc đó.

```
function Counter() {
  const [count, setCount] = React.useState(0);

  let id;

  const clear = () => {
    window.clearInterval(id);
  };

  React.useEffect(() => {
    id = window.setInterval(() => {
      setCount((c) => c + 1);
    }, 1000);

    return clear;
  }, []);

  return (
    <div>
      <h1>{count}</h1>
      <button onClick={clear}>Stop</button>
    </div>
  );
}
```

Code này chạy không? *Không*, lý do? bạn có để ý biến `id` giữa các lần chạy (render) là khác nhau, nói cách khác bạn không clear được cái interval đã setup.

Bạn sẽ phải viết lại sử dụng cách `usePersistentValue` ở trên

```
function usePersistentValue(initialValue) {
  return React.useState({
    current: initialValue,
  })[0];
}

function Counter() {
  const [count, setCount] = React.useState(0);
  const id = usePersistentValue(null);

  const clearInterval = () => {
    window.clearInterval(id.current);
  };

  React.useEffect(() => {
    id.current = window.setInterval(() => {
      setCount((c) => c + 1);
    }, 1000);

    return clearInterval;
  }, []);

  return (
    <div>
      <h1>{count}</h1>
      <button onClick={clearInterval}>Stop</button>
    </div>
  );
}
```

Nói có cảm giác hơi *sai trái* khi hack như vậy, nhưng nó chạy được.

Tuy nhiên không khuyến khích bạn tự viết như vậy, vì việc cố định giá trị giữa các lần render là nhu cầu khá *bình thường* nên bạn sẽ được team React làm sẵn cho một API mà xài: **useRef**

Vẫn là đoạn ứng dụng trên nhưng giờ chúng ta viết lại nó bằng `useRef`

```
function Counter() {
  const [count, setCount] = React.useState(0);
  const id = React.useRef(null);

  const clearInterval = () => {
    window.clearInterval(id.current);
  };

  React.useEffect(() => {
    id.current = window.setInterval(() => {
      setCount((c) => c + 1);
    }, 1000);

    return clearInterval;
  }, []);

  return (
    <div>
      <h1>{count}</h1>
      <button onClick={clearInterval}>Stop</button>
    </div>
  );
}
```

Công dụng của `useRef` như đã đề cập, cố định dữ liệu giữa các lần re-render, truy xuất giá trị đó qua thuộc tính `current`

Một ứng dụng rất phổ biến của `useRef` là truy xuất đến DOM node. Thí dụ để set focus của input

```
function Form() {
  const nameRef = React.useRef();
  const emailRef = React.useRef();
  const passwordRef = React.useRef();

  const handleSubmit = (e) => {
    e.preventDefault();

    const name = nameRef.current.value;
    const email = emailRef.current.value;
    const password = passwordRef.current.value;

    console.log(name, email, password);
  };

  return (
    <React.Fragment>
      <label>
        Name:
        <input placeholder="name" type="text" ref={nameRef} />
      </label>
      <label>
        Email:
        <input placeholder="email" type="text" ref={emailRef} />
      </label>
      <label>
        Password:
        <input placeholder="password" type="text" ref={passwordRef} />
      </label>

      <hr />

      <button onClick={() => nameRef.current.focus()}>Focus Name Input</button>
      <button onClick={() => emailRef.current.focus()}>
        Focus Email Input
```

```
        </button>
        <button onClick={() => passwordRef.current.focus()}>
          Focus Password Input
        </button>

        <hr />

        <button onClick={handleSubmit}>Submit</button>
      </React.Fragment>
    );
}
```

Bài viết gốc được đăng tải tại Vui Lập Trình

Có thể bạn quan tâm:

- React hook là gì và lợi ích mà React hook đem lại
- Sử dụng CSS content như thế nào cho đúng
- Tìm hiểu sâu hơn về useEffect từ a tới z

Xem thêm các việc làm Developer hấp dẫn tại TopDev

TopDev

Ban Biên Tập Blog TopDev. Nice to meet you