<https://codestus.com/posts/clean-up-memory-leak-trong-react-useeffect>

# Clean up memory leak trong React useEffect

5.107 lượt xem - Thời gian đọc: 3 phút đọc

Memory leak không còn là thuật ngữ quá xa lạ đối với các bạn khi làm việc với React. Đôi khi bạn sẽ thấy thông báo lỗi về rò rì bộ nhớ, điều này cảnh báo rằng chúng ta phải khắc phục nó ngay để tránh tác hại to lớn mà nó mang lại

Đôi khi chúng ta thấy nó xuất hiện với dòng thông báo lỗi

*Warning: Can't perform a React state update on an unmounted component....*

Tôi sẽ chỉ cho bạn thời điểm cảnh báo trên xuất hiện và cách cleanup nó.

## Tiếp tục cập nhật state sau khi unmount

Để hình dung rõ hơn, thử tạo 1 tác vụ cập nhật trạng thái trong quá trình chúng ta unmounting component đó.

Một ứng dụng hiển thị thông tin về một nhà hàng địa phương. Trang đầu tiên hiển thị danh sách nhân viên (nhân viên phục vụ, nhân viên bếp) và trang thứ hai hiển thị thông tin dạng văn bản.

Danh sách nhân viên được tải bằng yêu cầu tìm nạp **(fetch)**.

import { useState, useEffect } from 'react';

function Employees() {

const [list, setList] = useState(null);

useEffect(() => {

(async () => {

try {

const response = await fetch('/employees/list');

setList(await response.json());

} catch (e) {

// Some fetch error

}

})();

}, []);

return (

<div>

{list === null ? 'Fetching employees...' : ''}

{list?.map(name => <div>{name}</div>)}

</div>

);

}

function About() {

return (

<div>

<p>Our restaurant is located ....</p>

</div>

);

}

Thành phần **<App />** sẽ bao bọc và giúp chúng ta điều hướng qua lại giữa 2 thành phần trên

import { useState } from 'react';

function App() {

const [page, setPage] = useState('employees');

const showEmployeesPage = () => setPage('employees');

const showAboutPage = () => setPage('about');

return (

<div className="App">

<h2>My restaurant</h2>

<a href="#" onClick={showEmployeesPage}>Employees Page</a>

<a href="#" onClick={showAboutPage}>About Page</a>

{page === 'employees' ? <Employees /> : <About />}

</div>

);

}

Mở bản **demo** ứng dụng và trước khi quá trình tìm nạp của nhân viên hoàn tất, hãy nhấp vào liên kết Trang giới thiệu. Sau đó, mở bảng điều khiển và nhận thấy rằng React đã đưa ra một cảnh báo:

Lý do cho cảnh báo này là thành phần **<Employees />** đã được ngắt kết nối, nhưng vẫn còn, khiến quá trình tìm nạp hoàn thành và cập nhật trạng thái của thành phần khi đã và đang trong quá trình unmounting

function Employees() {

const [list, setList] = useState(null);

useEffect(() => {

(async () => {

try {

const response = await fetch('/employees/list');

// Sau khi quá trình tìm nạp hoàn thành

// Cập nhật trạng thái setList vẫn được gọi trong khi thành phần đã unmounting

setList(await response.json());

} catch (e) {

// Some fetch error

}

})();

}, []);

// ...

}

Giải pháp cho vấn đề này là gì? Như cảnh báo cho thấy, bạn cần phải hủy mọi tác vụ không đồng bộ đang hoạt động nếu thành phần ngắt kết nối. Hãy xem cách thực hiện điều đó trong phần tiếp theo.

## Cleanup quá trình gửi yêu cầu tìm nạp

useEffect(callback, dependencies) cho phép bạn cleanup các side-effects. Đó là khi tham số **callback** trong useEffect trả về một hàm **() => {}**, React sẽ gọi nó khi hàm chuyển trạng thái **unmount**:

const MyComponent = () => {

useEffect(() => {

// Side-effect logic...

return () => {

// Side-effect cleanup

};

}, []);

// ...

}

Theo đó, để huỷ bỏ một yêu cầu tìm nạp đang được thực hiện, chúng ta sẽ cần đến WebAPI **AbortController**.

Nào bây giờ hãy dựa trên ý tưởng phía trên và tiến hành khắc phục lỗi mà react đã thông báo

import { useState, useEffect } from 'react';

function Employees() {

const [list, setList] = useState(null);

useEffect(() => {

let controller = new AbortController();

(async () => {

try {

const response = await fetch('/employees/list', {

signal: controller.signal

});

setList(await response.json());

controller = null;

} catch (e) {

// Handle fetch error

}

})();

return () => controller?.abort();

}, []);

return (

<div>

{list === null ? 'Fetching employees...' : ''}

{list?.map(name => <div>{name}</div>)}

</div>

);

}

**let controller = new AbortController()** để chúng ta khởi tạo một đối tượng có quyền hạn huỷ bỏ các tác vụ đang thực hiện. Sau đó thiết lập kết nối giữa abort với yêu cầu tìm nạp của bạn trong fetch **await fetch(..., {signal: controller.signal})**.

Cuối cùng, trong hàm **cleanup** của **useEffect** chúng ta hãy gọi controller?.abort() để huỷ bỏ yêu cầu trong trường hợp nếu component vào trạng thái **unmount**.

Như vậy, khi chúng ta thao tác điều hướng nhanh qua lại giữa 2 component. Sẽ k thấy xuất hiện thông báo lỗi về rò rĩ bộ nhớ nữa.

Để tham khảo chi tiết hơn về giải pháp, bạn có thể tìm hiểu ở bài viết gốc của tác giả [ở đây](https://dmitripavlutin.com/react-cleanup-async-effects/).