Xử lý memory leak trong React

1

0



[Ken](https://nextlint.com/@khanhhaquang)

10:25 10/07/2021đăng trongCông nghệ

Hầu như với những người mới làm quen với React ai cũng từng gặp qua chuyện **console** hiện ra **warning** như thế này

Đọc sơ qua mình cũng hiểu được là đang gặp một rắc rối nhỏ với React **setState** api. State của một component vừa được gọi update trong khi nó đã **unmount** rồi. Thông thường **warning** này sẽ diễn ra trong 2 hoàn cảnh mà bản thân mình thấy hay gặp :

* Gọi một function trả về promise và sau đó thực thi setState ngay bên trong cái api như là **.then** , **.catch** hoặc **.finally** , nhưng **component** chứa function này lại bị unmount trước khi promise chạy xong và thực thi code.
* Sử dụng library handle routing của react như react-router, thực hiện chuyển route nhưng vẫn có một hàm bất đồng bộ chưa chạy xong được thực thi ở route trước đó và gọi **setState**.

Để mình làm một ví dụ nhỏ để mô phỏng lại trường hợp này

Đầu tiên mình tạo một function trả về một **promise** mẫu

**const fetchSomething = () => new Promise((resolve) => {**

**setTimeout(() => resolve('success'), 5000);**

**});**

Mình tạo một component **Parent** chứa component **Child**

**const Child = () => {**

**const [fetchStatus, setFetchStatus] = React.useState('start');**

**React.useEffect(() => {**

**setFetchStatus('pending');**

**fetchSomething().then((result) => {**

**console.log('result: ', result);**

**setFetchStatus(result);**

**});**

**}, []);**

**return Fetch status: {fetchStatus};**

**};**

**const Parent = () => {**

**const [isVisible, setVisible] = React.useState(true);**

**return (**

**// A fetching example**

**);**

**};**

Bên trong Child mình sẽ call **fetchSomething** trong **useEffect** lúc này đồng nghĩa với **componentDidMount**. Trong Parent các bạn để ý sẽ có một button để toggle ẩn **Child** đi, hiện giờ **fetchSomething** sau khi chờ 5 giây sẽ gọi **setFetchStatus**, nhưng trong thời gian chờ đó mình click toggle button, lúc này component Child unmount và khi xong 5 giây warning sẽ xuất hiện.

Vậy lỗi này có nguy hiểm không, thực sự thì tuỳ, nếu các bạn để nó xảy ra trên rất rất nhiều component mà thì việc tràn bộ nhớ mức độ nghiêm trọng sẽ tăng theo.

Ở đây mình sẽ có vài cách mình hay dùng để giải quyết việc này

**1. Nhận biết khi nào component unmount**

Ở đây mình sẽ tạo một **flag** để nhận biết là component đã **unmount** hay chưa, nếu rồi thì mình không setState nữa, mình sẽ mô phỏng lại tiếp theo ví dụ phía trên

**const Child = () => {**

**const [fetchStatus, setFetchStatus] = React.useState('start');**

**React.useEffect(() => {**

**let isUnmounted = false;**

**fetchSomething().then((result) => {**

**if (isUnmounted) return;**

**setFetchStatus(result);**

**});**

**return () => {**

**isUnmounted = true;**

**};**

**}, []);**

**return Fetch status: {fetchStatus};**

**};**

Nếu mà không gọi async function **fetchSomething** trong **useEffect** mà chỉ gọi khi click một button nào đó thì sao ?

Mình có thể chuyển flag **isUnmounted** kia thành **useRef**

**const Child = () => {**

**const [fetchStatus, setFetchStatus] = React.useState('start');**

**const isUnmounted = React.useRef(false);**

**const startFetching = () =&gt; {**

**fetchSomething().then((result) =&gt; {**

**if (isUnmounted.current) return;**

**setFetchStatus(result);**

**});**

**};**

**React.useEffect(**

**() => () => {**

**isUnmounted.current = true;**

**},**

**[],**

**);**

**return (**

**//sample code**

**);**

**};**

**2. Dùng AbortController**

AbortController là một object controller global được support hầu hết trên các browser ( trừ IE :lol: ) dùng để cancel các Web request. Thường thì các bạn dùng **axios** hoặc **fetch** để tạo http request lên server. Mình sẽ ví dụ cách **cancel** khi dùng **fetch**

**const Child = () => {**

**const [fetchStatus, setFetchStatus] = React.useState('start');**

**React.useEffect(() => {**

**const abortController = new AbortController();**

**const { signal } = abortController;**

**window**

**.fetch('https://randomuser.me/api/', { signal })**

**.then((result) => result.json())**

**.then((data) => {**

**setFetchStatus(data);**

**});**

**return () => {**

**abortController.abort();**

**};**

**}, []);**

**return Fetch status: {JSON.stringify(fetchStatus, null, 4)};**

**};**

Ở đây khi gọi fetch mình sẽ gắn signal và options init của request này tương ứng với **abortController** vừa khởi tạo, khi unmount thì mình dùng hàm **abort()** của controller để cancel.

**3. Dùng Redux**

Ở đây nếu các bạn đưa **state** cần update ra global store của **Redux** , lúc này đang wrap toàn bộ Web app thì cũng sẽ tránh được tình trạng này. Việc dùng Redux store cũng sẽ tránh bị memory leak khi chuyển route. Việc này các bạn có thể cân nhắc sử dụng.

!!! Một note lưu ý là khi dùng **React useReducer hook** cũng sẽ giống như useState nên các bạn cũng cần phải handle cancel các **asynchronous** calls nếu xảy ra **warning** như trên.

Cảm ơn các bạn đã đọc bài ! <3