**useEffect trong React Hooks**

Trong bài viết này chúng ta sẽ đi tìm hiểu về **useEffect trong React Hooks**. Ở các bài trước chúng ta đã cùng nhau đi tìm hiểu về useState hooks. Bài viết này sẽ tiếp nối series bằng một hook cũng khá quan trọng đó là **useEffect**, nó cho phép chúng ta làm việc với life cycle trong functional component*.*

**Mục lục**

* [1. useEffect trong React Hooks](https://freetuts.net/useeffect-trong-react-hooks-2578.html#goto-h2-0)
* [2. Sử dụng useEffect](https://freetuts.net/useeffect-trong-react-hooks-2578.html#goto-h2-1)
  + [Sử dụng useEffect như componentDidMount](https://freetuts.net/useeffect-trong-react-hooks-2578.html#goto-h3-0)
  + [Sử dụng useEffect như componentDidUpdate](https://freetuts.net/useeffect-trong-react-hooks-2578.html#goto-h3-1)
  + [Sử dụng useEffect như componentUnWillMount](https://freetuts.net/useeffect-trong-react-hooks-2578.html#goto-h3-2)

## 1. useEffect trong React Hooks

useEffect là một hook trong React Hooks cho phép chúng ta làm việc với các [life cycle](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html)ở functional component. Có thể hiểu đơn giản rằng useEffect Hook là của 3 phương thức componentDidMount**,**componentDidUpdate**,**vàcomponentWillUnmount kết hợp lại với nhau.

Lifecyle là một phần rất quan trọng trong một component. Trong một vài trường hợp chúng ta cần phải fetch data từ API khi component đã được render, hay thực hiện hành động nào đó khi một component được update. Bởi vậy có thể thấy rằng phương thức quan trọng và hay được sử dụng nhất trong lifce cycle đó là componentDidMount, componentDidUpdate.

Nhưng trong một functional component không thể làm việc với các life cycle này bằng cách thông thường, bởi vậy useEffect Hooks sinh ra để làm điều này.

*Bài viết này được đăng tại [free tuts .net]*

**useEffect** cho phép chúng ta xử lý các logic trong các vòng đời của component và được gọi mỗi khi có bất cứ sự thay đổi nào trong một componnet.

## 2. Sử dụng useEffect

Chúng ta sẽ đi vào tìm hiểu sâu hơn về **useEffect** bằng cách đi qua các ví dụ cụ thể. Để sử dụng useEffect bạn cần phải import nó vào trong component cần sử dụng:

|  |  |
| --- | --- |
| 1 | import React, { useEffect } from 'react' |

và sử dụng nó bằng cú pháp:

|  |  |
| --- | --- |
| 1 | useEffect(effectFunction, arrayDependencies) |

trong đó arrayDependencies là một mảng các giá trị, khi giá trị trong mảng này thay đổi thì hàm effectFunction sẽ được gọi. Trước khi đi tiếp vào phần sau bạn có thể tham khảo thêm bài viết về [Tìm hiểu về Life cycle trong ReactJS](https://freetuts.net/component-life-cycle-trong-reactjs-2387.html).

### Sử dụng useEffect như componentDidMount

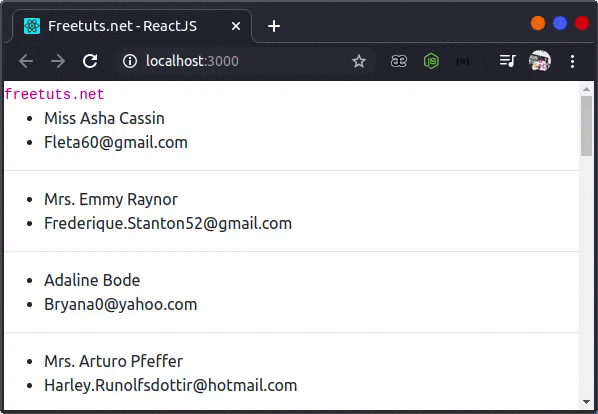
Chúng ta có thể bắt sự kiện componentDidMount trong một functional component bằng cách sử dụng useEffect và chỉ định arrayDependencies là một mảng rỗng:

|  |  |
| --- | --- |
| 1 | useEffect(effectFunction, []) |

Nó thường được sử dụng để gọi API khi compoent đã được render. Ở đây mình có một ví dụ về gọi API.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | //Gọi React và useState  import React, { useState, useEffect } from "react";  import axios from "axios"; //Sử dụng axios    //Component hiển thị danh sách người dùng  const ShowUser = (props) => {    //Lấy giá tri của props listUser    const { listUser } = props;      // Render ra list user    // React.Fragment cho phép bọc JSX lại.    // List Keys :  chỉ định key, giúp loại bỏ cảnh báo.    return (      <div>        {listUser.map((user, index) => {          return (            <React.Fragment key={user.id}>              <ul>                <li>{user.name}</li>                <li>{user.email}</li>              </ul>              <hr />            </React.Fragment>          );        })}      </div>    );  };    export default function App(props) {    //Khai báo state, sử dụng hook: useState    const [listUser, setListUser] = useState([]);      //Sử dụng useEffect hook như componentDidMount    useEffect(() => {      const getUserAPI = '<https://5df8a4c6e9f79e0014b6a587.mockapi.io/freetuts/users>'        //Gọi API bằng axios      axios.get(getUserAPI).then((res) => {        // Cập nhật lại listUser bằng        // Bạn có thể xem lại bài viết về useState()        setListUser(res.data);      }).catch((err) => {        //Trường hợp xảy ra lỗi        console.log(err);        alert("Xảy ra lỗi");      })      }, [])      return (      <>        <code>freetuts.net</code> <br />          <ShowUser listUser={listUser} />      </>    );  } |

chúng ta sẽ nhận được kết quả như hình:



### Sử dụng useEffect như componentDidUpdate

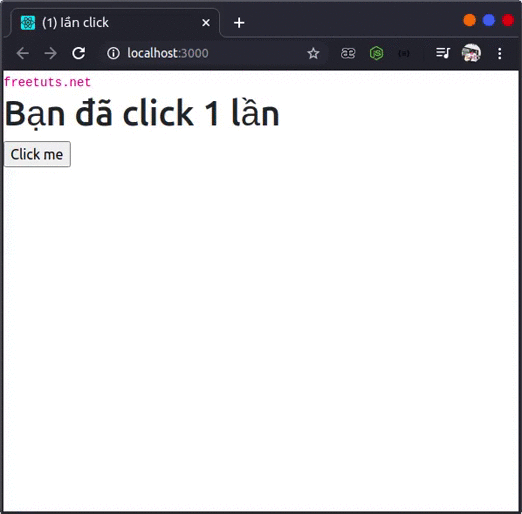
Để sử dụng componentDidUpdate trong functional component chúng ta sẽ chỉ định giả trị của tham số arrayDependencies là null.

|  |  |
| --- | --- |
| 1 | useEffect(effectFunction) |

Ở đây chúng ta sẽ xây dựng ví dụ về thay đổi tiêu đề mỗi khi component thực hiện update.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | //Gọi React và useState  import React, { useState, useEffect } from "react";      export default function App() {    const [count, setCount] = useState(1);    //Sử dụng useEffect() như componentDidUpdate    useEffect(() => {      document.title = `(${count}) lần click`    })    return (      <>        <code>freetuts.net</code> <br />        <h1>Bạn đã click {count} lần</h1>        <button onClick = {() => {            setCount(count+1)        }}>Click me</button>      </>    );  } |

Và đây là kết quả.

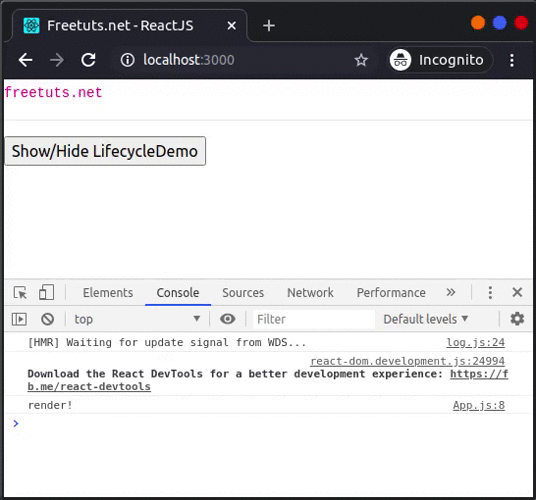


### Sử dụng useEffect như componentUnWillMount

Lifecycle thứ 3 mà chúng ta có thể sử dụng được với useEffect là componentUnWillMount, lifecycle này được khởi chạy khi mỗi khi component chuẩn bị được xóa khỏi tree dom. Để sử dụng nó bằng useEffect chúng ta chỉ cần return về 1 function trong effectFunction, function được return đó sẽ đóng vai trò như là componentUnWillMount. Cùng xem xét ví dụ bên duới:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | import React, { useEffect, useState } from 'react';      function LifecycleDemo() {      useEffect(() => {      //Được gọi khi component render      console.log('render!');      // useEffect trả về một hàm ,      // hàm trả về đó là đóng vai trò như      // là componentWillUnmount      return () => console.log('unmounting...');    })      return (<code>freetuts.net</code>);  }    export default function App() {    const [mounted, setMounted] = useState(true);    const toggle = () => setMounted(!mounted);    return (      <>        {mounted && <LifecycleDemo/>} <hr />        <button onClick={toggle}>Show/Hide LifecycleDemo</button>     </>    );  } |

mở **Dev Tools -> Console** lên và bạn sẽ thấy sự thay đổi ở mỗi lần component bị xóa bỏ khỏi tree dom.



Trên đây chúng ta đã cùng nhau đi tìm hiểu về useEffect trong React Hooks. Đây là kiến thức rất cơ bản về nó nhưng cũng hết sức quan trọng trong quá trình làm việc với ReactJS sau này. Mong rằng bài viết sẽ giúp ích cho bạn.