

Khóa học Front-end ATT Lab - ReactJS

Tổng quan 🎯

1. Giới thiệu useRef
 2. Side effect là gì?
 3. Giới thiệu useEffect
 4. Sự khác nhau useEffect vs useLayoutEffect
-

1. Giới thiệu useRef

What? Trong ReactJS, **useRef** là một hook được sử dụng để tham chiếu đến một phần tử DOM hoặc một instance của một component. Nó thường được sử dụng để lưu trữ các tham chiếu tới các phần tử DOM hoặc giá trị không thay đổi khi component re-render. Dưới đây là cách sử dụng **useRef** một cách dễ hiểu.

- Trường hợp tham chiếu vào 1 **phần tử DOM**

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

function MyComponent() {
  const inputRef = useRef(null);

  useEffect(() => {
    // Tự động focus vào input khi component được render
    inputRef.current.focus();
  }, []);

  return (
    <div>
      <input ref={inputRef} type="text" />
      <button onClick={() => inputRef.current.focus()}>Focus
    </div>
  );
}
```

- Trường hợp giữ cho 1 giá trị không thay đổi khi component re-render

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

function PreviousValueComponent() {
  const [count, setCount] = useState(0);
  const prevCountRef = useRef();

  // Lưu trữ giá trị trước đó của count
  useEffect(() => {
    prevCountRef.current = count;
  });

  const handleIncrement = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Current Count: {count}</p>
      <p>Previous Count: {prevCountRef.current}</p>
      <button onClick={handleIncrement}>Increment Count</button>
    </div>
  );
}
```

```
    </div>
  );
}

export default PreviousValueComponent;
```

Bài tập

- <https://codesandbox.io/p/sandbox/quirky-smoke-z5z634>
- <https://codesandbox.io/p/sandbox/serverless-night-thvxpd>

2. Side effect là gì?

Khi xây dựng ứng dụng, chúng ta thường cần đồng bộ hóa với các hệ thống bên ngoài. Điều này có thể bao gồm những thứ như:

2.1. Thực hiện các network requests

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

const UserProfile = ({ userId }) => {
  const [userData, setUserData] = useState(null);

  useEffect(() => {
    const fetchUserData = async () => {
      try {
        const response = await
        axios.get(`https://api.example.com/user/${userId}`);
        setUserData(response.data);
      } catch (error) {
        console.error('Lỗi khi tải dữ liệu người dùng:', error);
      }
    };

    fetchUserData();

    return () => {
      // Cleanup function
    };
  }, [userId]);

  return (
    <div>
      {userData ? (
        <div>
          <h2>{userData.name}</h2>
          <p>{userData.email}</p>
          {/* Hiển thị hồ sơ người dùng */}
        </div>
      ) : (
```

```
        <p>Đang tải...</p>
      )}
    </div>
  );
};

export default UserProfile;
```

2.2. Quản lý timeouts / intervals

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

const TimerComponent = () => {
  useEffect(() => {
    const timeoutId = setTimeout(() => {
      console.log('Timeout finished!');
    }, 3000); // Timeout sau 3 giây

    return () => {
      clearTimeout(timeoutId); // Dọn dẹp timeout khi component bị unmount
    };
  }, []);

  return (
    <div>
      <h3>Timer Component</h3>
      {/* Hiển thị giao diện của component */}
    </div>
  );
};

export default TimerComponent;
```

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

const IntervalComponent = () => {
  useEffect(() => {
    const intervalId = setInterval(() => {
      console.log('Interval tick!');
    }, 1000); // Gọi mỗi 1 giây

    return () => {
      clearInterval(intervalId); // Dọn dẹp interval khi component bị unmount
    };
  }, []);

  return (
    <div>
```

```
    <h3>Interval Component</h3>
    { /* Hiển thị giao diện của component */ }
  </div>
);
};

export default IntervalComponent;
```

2.3. Đọc/ghi từ localStorage

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

const UserProfileSettings = () => {
  const [darkMode, setDarkMode] = useState(false);

  useEffect(() => {
    const storedDarkMode = localStorage.getItem('darkMode');
    if (storedDarkMode) {
      setDarkMode(JSON.parse(storedDarkMode));
    }
  }, []);

  useEffect(() => {
    localStorage.setItem('darkMode', JSON.stringify(darkMode));
  }, [darkMode]);

  const toggleDarkMode = () => {
    setDarkMode(!darkMode);
  };

  return (
    <div>
      <label>
        Chế Độ Tối:
        <input type="checkbox" checked={darkMode} onChange=
{toggleDarkMode} />
      </label>
    </div>
  );
};

export default UserProfileSettings;
```

2.4. Lắng nghe các global events

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

const GlobalEventListener = () => {
  useEffect(() => {
```

```

const handleKeyPress = (event) => {
  console.log('Global key pressed:', event.key);
};

window.addEventListener('keypress', handleKeyPress);

return () => {
  window.removeEventListener('keypress', handleKeyPress); // Dọn dẹp
  // lắng nghe khi component bị unmount
};
}, []);

return (
  <div>
    <h3>Global Event Listener</h3>
    {/* Hiển thị giao diện của component */}
  </div>
);
};

export default GlobalEventListener;

```

3. Giới thiệu useEffect

3.1. Giới thiệu Lifecycle

3.2. useEffect với dependencies rỗng

Khi useEffect có một mảng dependencies rỗng [], nó sẽ chỉ chạy một lần sau khi component được mount lần đầu tiên và sẽ không chạy lại khi component cập nhật.

```

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

function EmptyDependenciesDemo() {
  useEffect(() => {
    console.log('This runs only once after the first render');

    return () => {
      console.log('Clean up when component unmounts');
    };
  }, []);

  return (
    <div>
      <p>Check the console to see the effect</p>
    </div>
  );
}

```

```
export default EmptyDependenciesDemo;
```

3.3. useEffect với dependencies

Khi bạn cung cấp một mảng dependencies cho useEffect, hiệu ứng sẽ chạy lại mỗi khi một trong các dependencies thay đổi. Điều này hữu ích khi bạn cần thực hiện các side effect dựa trên các giá trị thay đổi.

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

function DependenciesDemo() {
  const [count, setCount] = useState(0);
  const [text, setText] = useState('');

  useEffect(() => {
    console.log('Effect runs when count or text changes');
  }, [count, text]);

  return (
    <div>
      <p>Count: {count}</p>
      <p>Text: {text}</p>
      <button onClick={() => setCount(count + 1)}>Increment Count</button>
      <input type="text" value={text} onChange={(e) =>
setText(e.target.value)} />
    </div>
  );
}

export default DependenciesDemo;
```

3.6. useEffect with timer functions

Bạn có thể sử dụng useEffect để thiết lập và dọn dẹp các hàm timer như setTimeout hoặc setInterval. Điều này giúp tránh các lỗi bộ nhớ khi component bị unmount.

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

function TimerDemo() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setCount(prevCount => prevCount + 1);
    }, 1000);

    // Cleanup interval on component unmount
    return () => {
      clearInterval(interval);
    };
  }, []);
}
```


```
    };  
    }, []);  
  
    return (  
      <div>  
        <p>Count: {count}</p>  
      </div>  
    );  
  }  
  
  export default TimerDemo;
```

- Demo flow: <https://julesblom.com/writing/react-hook-component-timeline>

Bài tập

- <https://codesandbox.io/p/sandbox/buoi06-bt1-kxx6hq>
- <https://codesandbox.io/p/sandbox/dark-snow-xy596n>

4. Sự khác nhau useEffect vs useLayoutEffect

 alt text

```
import React, { useState, useEffect } from 'react';  
  
function TimerDemo() {  
  const [count, setCount] = useState(0);  
  
  useEffect(() => {  
    if(count === 3) {  
      setCount(0)  
    }  
  }, [count]);  
  
  return (  
    <div>  
      <p>Count: {count}</p>  
    </div>  
  );  
}  
  
export default TimerDemo;
```