

**REACT-JS Cơ bản**

# **Introducing Hooks**

Hooks trong React được thêm vào từ phiên bản React 16.8. Chúng giúp chúng ta có thể sử dụng được state và các tính năng khác mà không cần phải viết lại class.

Hooks là một hàm trong javascript, nhưng chúng yêu cầu bắt buộc tuân theo 2 rule sau:

* Hooks chỉ được gọi ở mức 1. Không được gọi Hooks bên trong vòng lặp, trong lệnh điều kiện, hoặc lồng bên trong một hàm.
* Chỉ được gọi Hooks bên trong function components. Không được gọi Hook bên trong hàm của javascript thông thường.

import React, { useState } from 'react';

function Example() {

// Declare a new state variable, which we'll call "count"

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

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

}

## **State Hook**

Sử dụng useState:

import React, { useState } from 'react';

function Example() {

// Declare a new state variable, which we'll call "count"

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

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

}

## **Effect Hook**

Effect hook tương đương với componentDidMount, componentDidUpdate, và componentWillUnmount trong react class.

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

function Example() {

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

// Similar to componentDidMount and componentDidUpdate:

useEffect(() => {

// Update the document title using the browser API

document.title = `You clicked ${count} times`;

});

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

}

sử dụng Effect với “clean up”

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

function FriendStatus(props) {

const [isOnline, setIsOnline] = useState(null);

function handleStatusChange(status) {

setIsOnline(status.isOnline);

}

useEffect(() => {

ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);

return () => {

ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);

};

});

if (isOnline === null) {

return 'Loading...';

}

return isOnline ? 'Online' : 'Offline';

}

Chúng ta có thể sử dụng nhiều State, và Effect Hooks trong cùng một component:

function Form() {

// 1. Use the name state variable

const [name, setName] = useState('Mary');

// 2. Use an effect for persisting the form

useEffect(function persistForm() {

localStorage.setItem('formData', name);

});

// 3. Use the surname state variable

const [surname, setSurname] = useState('Poppins');

// 4. Use an effect for updating the title

useEffect(function updateTitle() {

document.title = name + ' ' + surname;

});

// ...

}

## **Custom Hook**

Custom hook là một hàm javascript có tên hàm bắt đầu với từ “use" và nó sẽ gọi hàm hook khác. Ví dụ hàm “useFriendStatus” sau:

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

function useFriendStatus(friendID) {

const [isOnline, setIsOnline] = useState(null);

useEffect(() => {

function handleStatusChange(status) {

setIsOnline(status.isOnline);

}

ChatAPI.subscribeToFriendStatus(friendID, handleStatusChange);

return () => {

ChatAPI.unsubscribeFromFriendStatus(friendID, handleStatusChange);

};

});

return isOnline;

}

sử dụng custom hook:

function FriendStatus(props) {

const isOnline = useFriendStatus(props.friend.id);

if (isOnline === null) {

return 'Loading...';

}

return isOnline ? 'Online' : 'Offline';

}

hoặc

function FriendListItem(props) {

const isOnline = useFriendStatus(props.friend.id);

return (

<li style={{ color: isOnline ? 'green' : 'black' }}>

{props.friend.name}

</li>

);

}

Vậy các câu hỏi đặt ra:

1. Tôi có phải đặt tên cho custom hook bắt đầu bằng từ “use" không?
2. Có thể cho 2 components khác nhau sử dụng chung một state được không?
3. Làm thế nào mà custom hook có được sự độc lập về state?

### 

### **Information Between Hooks**

const friendList = [

{ id: 1, name: 'Phoebe' },

{ id: 2, name: 'Rachel' },

{ id: 3, name: 'Ross' },

];

function ChatRecipientPicker() {

const [recipientID, setRecipientID] = useState(1);

const isRecipientOnline = useFriendStatus(recipientID);

return (

<>

<Circle color={isRecipientOnline ? 'green' : 'red'} />

<select

value={recipientID}

onChange={e => setRecipientID(Number(e.target.value))}

>

{friendList.map(friend => (

<option key={friend.id} value={friend.id}>

{friend.name}

</option>

))}

</select>

</>

);

}

# 

# **Hooks API**

Các hàm hook:

* [Basic Hooks](https://reactjs.org/docs/hooks-reference.html#basic-hooks)
  + [useState](https://reactjs.org/docs/hooks-reference.html#usestate)
  + [useEffect](https://reactjs.org/docs/hooks-reference.html#useeffect)
  + [useContext](https://reactjs.org/docs/hooks-reference.html#usecontext)
* [Additional Hooks](https://reactjs.org/docs/hooks-reference.html#additional-hooks)
  + [useReducer](https://reactjs.org/docs/hooks-reference.html#usereducer)
  + [useCallback](https://reactjs.org/docs/hooks-reference.html#usecallback)
  + [useMemo](https://reactjs.org/docs/hooks-reference.html#usememo)
  + [useRef](https://reactjs.org/docs/hooks-reference.html#useref)
  + [useImperativeHandle](https://reactjs.org/docs/hooks-reference.html#useimperativehandle)
  + [useLayoutEffect](https://reactjs.org/docs/hooks-reference.html#uselayouteffect)
  + [useDebugValue](https://reactjs.org/docs/hooks-reference.html#usedebugvalue)

### **useContext**

const value = useContext(MyContext);

nhận vào giá trị object của một context được lấy từ hàm React.createContext và trả về giá trị hiện tại của context đó. Giá trị của context hiện tại được xác định bởi giá trị props của một component <MyContext.Provider> gần nhất trong cấu trúc cây.

Một component gọi useContext luộn re-render khi giá trị của context đó thay đổi.

Ví dụ sử dụng:

const themes = {

light: {

foreground: "#000000",

background: "#eeeeee"

},

dark: {

foreground: "#ffffff",

background: "#222222"

}

};

const ThemeContext = React.createContext(themes.light);

function App() {

return (

<ThemeContext.Provider value={themes.dark}>

<Toolbar />

</ThemeContext.Provider>

);

}

function Toolbar(props) {

return (

<div>

<ThemedButton />

</div>

);

}

function ThemedButton() {

const theme = useContext(ThemeContext);

return (

<button style={{ background: theme.background, color: theme.foreground }}>

I am styled by theme context!

</button>

);

}

### 

### **useReducer**

const [state, dispatch] = useReducer(reducer, initialArg, init);

Ví dụ:

const initialState = {count: 0};

function reducer(state, action) {

switch (action.type) {

case 'increment':

return {count: state.count + 1};

case 'decrement':

return {count: state.count - 1};

default:

throw new Error();

}

}

function Counter() {

const [state, dispatch] = useReducer(reducer, initialState);

return (

<>

Count: {state.count}

<button onClick={() => dispatch({type: 'decrement'})}>-</button>

<button onClick={() => dispatch({type: 'increment'})}>+</button>

</>

);

}

### 

### **useCallback**

const memoizedCallback = useCallback(

() => {

doSomething(a, b);

},

[a, b],

);

### **useMemo**

const memoizedValue = useMemo(

() => computeExpensiveValue(a, b),

[a, b]

);

### **useRef**

const refContainer = useRef(initialValue);

ví dụ:

function TextInputWithFocusButton() {

const inputEl = useRef(null);

const onButtonClick = () => {

// `current` points to the mounted text input element

inputEl.current.focus();

};

return (

<>

<input ref={inputEl} type="text" />

<button onClick={onButtonClick}>Focus the input</button>

</>

);

}

### **useImperativeHandle**

useImperativeHandle(ref, createHandle, [deps])

Ví dụ:

function FancyInput(props, ref) {

const inputRef = useRef();

useImperativeHandle(ref, () => ({

focus: () => {

inputRef.current.focus();

}

}));

return <input ref={inputRef} ... />;

}

FancyInput = forwardRef(FancyInput);

### **useLayoutEffect**

Giống với useEffect hook nhưng chỉ chạy sau khi toàn bộ DOM đã được render, tương đương với 2 hàm componentDidMount và componentDidUpdate.

**Bài tập:**

**Yêu cầu viết lại form demo có sử dụng React Hooks (viết các class component thành function component)**