MỤC LỤC

[15 CÂU HỎI PHỎNG VẤN REACT PHỔ BIẾN 2](#_Toc75254136)

[1. DOM ảo là gì? 2](#_Toc75254137)

[2. JSX là gì? 2](#_Toc75254138)

[3. Sự khác nhau giữa class component và functional component? 2](#_Toc75254139)

[4. Keys được dùng để làm gì? 3](#_Toc75254140)

[5. Sự khác nhau giữa state và props 3](#_Toc75254141)

[6. Tại sao phải goi setState thay vì trực tiếp thay đổi state? 4](#_Toc75254142)

[7. Làm thế nào để bạn giới hạn kiểu giá trị cho props được truyền vào hay làm nó luôn cần required? 4](#_Toc75254143)

[8. Props drilling là gì? Cách để phòng tránh nó? 4](#_Toc75254144)

[9. React context là gì? 5](#_Toc75254145)

[10. Redux là gì? 5](#_Toc75254146)

[11. Cách tiếp cận phổ biến để style ứng dụng React? 5](#_Toc75254147)

[12. Sự khác nhau controlled và uncontrolled component? 5](#_Toc75254148)

[13. Các phương thức vòng đời là gì? 6](#_Toc75254149)

[14. React hook là gì? 6](#_Toc75254150)

[15. Các ưu điểm của React hooks? 6](#_Toc75254151)

# 15 CÂU HỎI PHỎNG VẤN REACT PHỔ BIẾN

## 1. DOM ảo là gì?

Trả lời:

DOM ảo (virtual DOM) là một đại diện được nằm trong bộ nhớ cho một thành phần HTML thật mà cấu thành nên giao diện cho chương trình. Khi một component được thông dịch lại (re-render), DOM ảo sẽ so sánh sự thay đổi với mô hình của DOM thật để tạo một danh sách cập nhật sẽ được thực hiện. Lợi ích chính của việc này là giúp tăng hiệu năng, chỉ tập trung vào các thay đổi nhỏ và thực sự cần thiết với DOM thật hơn là phải re-render lại một tập component lớn.

## 2. JSX là gì?

Trả lời:

JSX là một phần mở rộng của cú pháp JavaScript cho phép viết code trông giống như HTML. Nó sẽ biên dịch thành các lời gọi hàm Javascript thông thường, cung cấp một cách tiếp cận dễ hiểu hơn để tạo các markup cho component của bạn.

Ta có JSX:

<div className="sidebar" />

Khi được dịch sang Javascript sẽ là:

React.createElement(

'div',

{className: 'sidebar'}

)

## 3. Sự khác nhau giữa class component và functional component?

Trả lời:

Trước phiên bản React 16.8 ( trước khi giới thiệu hooks ), component dựa theo class thường được sử dụng để tạo component, với mục đích để lưu giữ trạng thái bên trong hay tận dụng các phương thức vòng đời ( VD: componentDidMount hay shouldComponentUpdate ). Một component dựa theo class là một class ES6, nó mở rộng class React Component và với tối thiểu phải thực hiện phương thức render()

Class component:

class Welcome extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

Functional component là không có trạng thái (stateless) ( được dùng ở phiên bản React trước 16.8 ) và trả về output mà cần được render. Chúng phù hợp với việc render UI chỉ phụ thuộc vào props, thông thường chúng đơn giản hơn và nhanh hơn so với component dựa theo class.

Functional component:

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

## 4. Keys được dùng để làm gì?

Trả lời:

Khi thực hiên render một tập hợp trong React, việc thêm một key cho mỗi thành phần được lặp lại là cần thiết để giúp React theo dấu mối liên kết giữa các thành phần và dữ liệu. Key nên là một ID duy nhất, lý tưởng nhất nên là một UUID hay một chuỗi duy nhất khác từ tập hợp phần tử, hoặc cũng có thể là index của array.

<ul>

{todos.map((todo) =>

<li key={todo.id}>

{todo.text}

</li>

)};

</ul>

Không dùng key có thể dẫn đến một vài sự thay đổi lạ khi thêm và xoá các phần tử từ tập hợp.

## 5. Sự khác nhau giữa state và props

Trả lời:

Props là dữ liệu được truyền vào trong một component từ cha của nó. Chúng không nên bị thay đổi, và chỉ dùng để hiển thị hay tính toán các giá trị khác. State là dữ liệu bên trong của một component, nó có thể được thay đổi trong vòng đời của component và được duy trì giữa các lần re-render

## 6. Tại sao phải goi setState thay vì trực tiếp thay đổi state?

Trả lời:

Nếu bạn cố gắng thay đổi một state của component trực tiếp, React sẽ không thể biết được khi nào nó cần phải re-render component. Bằng cách sử dụng phương thức setState(), React có thể cập nhật component của UI.

Nếu bạn cần cập nhật một state của component dựa theo dữ liệu của state khác hay props khác, chỉ cần truyền một hàm vào setState(), khi đấy bạn truyền state và props như 2 tham số của hàm:

this.setState((state, props) => ({

counter: state.counter + props.increment

}));

## 7. Làm thế nào để bạn giới hạn kiểu giá trị cho props được truyền vào hay làm nó luôn cần required?

Trả lời:

Để kiểm tra kiểu cho một props của component, bạn có thể dùng thư viện prop-types để chỉ rõ kiểu của giá trị mong muốn và yêu cầu prop là cần phải có hay không:

import PropTypes from 'prop-types';

class Greeting extends React.Component {

render() {

return (

<h1>Hello, {this.props.name}</h1>

);

}

}

Greeting.propTypes = {

name: PropTypes.string

};

## 8. Props drilling là gì? Cách để phòng tránh nó?

Trả lời:

Prop drilling là điều xảy ra khi bạn cần truyền dữ liệu từ một component cha xuống một component thấp hơn trong cây component, drilling - khoan vào các component khác mà các component đấy có thể không cần giá trị props, trong khi chỉ một vài component là cần thôi.

Thỉnh thoảng vấn đề prop drilling có thể được phòng tránh bằng việc refactor component, tránh việc chia component thành nhỏ hơn và giữ các state chung vào trong component cha chung gần nhất. Với việc bạn cần chia sẻ state giữa các component mà không phụ thuộc vào vị trí xa hay gần trong cây component, bạn có thể sử dụng React Context API hay thư viện quản lý state tập trung \_ vd như Redux.

## 9. React context là gì?

Trả lời:

Context API được cung cấp bởi React để giải quyết vấn đề chia sẻ state giữa các component trong một ứng dụng. Trước khi context ( bối cảnh ) được giới thiệu, giải pháp duy nhất là sử dụng một thư viện quản lý state, VD như Redux. Tuy nhiên, nhiều nhà phát triển cảm thấy Redux cung cấp nhiều thứ phức tạp không cần thiết, đặc biệt là với ứng dụng nhỏ.

## 10. Redux là gì?

\*Trả lời:

Redux là thư viện quản lý state bên thứ 3 cho React, được tạo trước context API. Nó dựa theo khái niệm của một kho chứa state, hay gọi là store, các component có thể nhận dữ liệu từ các props. Cách duy nhất để update store là dispatch một hành động đến store, và việc này được thực thi qua reducer. Reducer sẽ nhận action và state hiện tại, và trả về một state mới, đồng thời kích hoạt cho các component đăng ký trạng thái re-render lại.

## 11. Cách tiếp cận phổ biến để style ứng dụng React?

\*Trả lời:

Có nhiều cách tiếp cần để style React component, mỗi cách đều có ưu và nhược điểm. Đó là:

Inline styling: rất hữu ích cho việc sửa style, nhưng có nhiều giới hạn ( vd: không style được các class giả )

Class based CSS styles: nhiều tính năng hữu ích hơn inline styling và phù hợp với các nhà phát triển mới

CSS in JS styling: có nhiều thư viện cho phép style được định nghĩa như Javascript trong components, và chúng ta thao tác giống như code JS thông thường

## 12. Sự khác nhau controlled và uncontrolled component?

\*Trả lời:

Trong một tài liệu HTML, có rất nhiều thành phần quản lý trạng thái của chính nó. Một uncontrolled component ( component không tự điều khiển ) coi DOM như nguồn state của input của chúng. Còn trong một controlled component ( component tự điều khiển ), state bên trong thường được sử dụng để theo dấu giá trị của thành phần. Khi giá trị của input thay đổi, React sẽ re-render lại input.

Uncontrolled components có thể hữu ích khi tích hợp với code không React

## 13. Các phương thức vòng đời là gì?

\*Trả lời:

Các component dựa theo class có thể sử dụng các phương thức đặc biệt, được gọi tại các thời điểm cụ thể trong vòng đời của nó, như khi nó được mount ( render vào DOM ) và khi nó bị unmount. Những thời điểm này rất hữu ích cho việc cài đặt và cung cấp thông tin cho một component có thể cần, cài đặt bộ đo thời gian hay gắn một vài sự kiện trình duyệt, ...

Các phương thức vòng đời bên dưới hay được dùng:

componentWillMount: được gọi sau khi component được tạo, nhưng trước khi nó được render vào DOM.

componentDidMount: được gọi sau lần render đầu tiên, thành phần DOM của component đã chuyển sang trạng thái sẵn sàng sử dụng

componentWillReceiveProps: được gọi khi một prop cập nhật

shouldComponentUpdate: khi các prop mới được nhận, phương thức này có thể chặn quá trình re-render để tối ưu hoá hiệu suất

componentWillUpdate: được gọi khi prop mới được nhận và shouldComponentUpdate trả về true

componentDidUpdate: được gọi sau khi component đã cập nhật

componentWillUnmount: được gọi trước khi component được xoá khỏi DOM, cho phép bạn xoá mọi thứ như bộ lắng nghe sự kiện, ...

## 14. React hook là gì?

\*Trả lời:

Hook là nỗ lực của React để mang sự thuận tiện của component dựa theo class sang component dựa theo function ( bao gồm state nội tại và các phương thức vòng đời )

## 15. Các ưu điểm của React hooks?

\*Trả lời:

Có một vài lợi ích khi sử dụng React hook là:

Xoá bỏ sự cần thiết của component dựa theo class, các vòng đời và keyword this

Dễ dàng tái sử dụng logic, bằng cách trừu tượng hoá các chức năng phổ biến thành custom hook

Code dễ đọc, dễ test bằng cách chia sẻ logic giữa các component với nhau