**Higher-Order Components trong ReactJS**

**Trong bài viết này ta sẽ tìm hiểu về Higner-Order Components trong React JS, đây là tính năng rất hay trong việc tái sử dụng tài nguyên.**

Trong một dự án ReactJS việc tái sử dụng các component cũng như phân chia các component sao cho hợp lý là điều tất yếu. Khi các component được nhóm với nhau thành các phần riêng biệt, tái sử dụng các component sẽ giúp quá trình duy trì dự án được dễ dàng hơn sau này. Chúng ta có phương pháp đó là **Higher-Order Components** cho phép làm những việc này một cách đơn giản.

## 1. Higher-Order Components là gì ?

**Higher-Order Components(HOC)** theo định nghĩa nó là một funciton nhận vào một component và trả về một component. **Higher-Order Components** không phải là một tính năng trong React hoặc bất kỳ ngôn ngữ lập trình nào khác, mà là một phương pháp phát triển component.

Để hiểu rõ hơn tại sao phải sử dụng nó thì ta sẽ làm một ví dụ đơn giản như sau:

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

Ở đây mình xây dựng 1 component có chức năng hiển thị ra một ảnh và mỗi khi di con trỏ chuột vào thì ảnh sẽ bị mờ đi.

|  |  |
| --- | --- |
| 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 | import React from "react";    const Image = (props) => {    return <img src="<https://freetuts.net/public/logo/logo.png>" alt="freetuts" />;  };    export default class HoverComponent extends React.Component {    constructor(props) {      super(props);      this.state = {        opacity: 1,      };      //bind this      this.onMouseLeave = this.onMouseLeave.bind(this);      this.onMouseEnter = this.onMouseEnter.bind(this);    }    //Được gọi khi chuột được di vào    onMouseEnter() {      this.setState({        opacity: 0.5,      });    }    //Được gọi khi chuột được rời đi    onMouseLeave() {      this.setState({        opacity: 1,      });    }    render() {      return (        <div          style={{ opacity: this.state.opacity }}          onMouseEnter={this.onMouseEnter}          onMouseLeave={this.onMouseLeave}        >          <Image />        </div>      );    }  } |

Trong chương trình trên, khi muốn hiển thị nhiều ảnh thì phải xây dựng rất nhiều HoverComponent chỉ nhằm 1 mục đích làm mờ ảnh. Điều này là không cần thiết. Vậy làm sao để làm mờ nhiều ảnh mà không cần phải viết lại HoverComponent cho mỗi ảnh. Chúng ta sẽ đi tìm cách giải quyết cho bài toán này ở phần ví dụ nhé.

## 2. Triển khai ví dụ

Ta sẽ đi giải quyết bài toán ở đầu bài cho, lúc này sẽ cần sử dụng **Higher Order Componets(HOC)** có chức năng làm mờ ảnh. Cùng bắt đầu xây dưng một HOC có tên withHoverOpacity :

|  |  |
| --- | --- |
| 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 | //Đây được gọi là một HOC, nó nhận vào 1 component  //và trả ra một component  const withHoverOpacity = (ImageComponent) => {    return class extends React.Component {      constructor(props) {        super(props);        this.state = {          opacity: 1,        };        //bind this        this.onMouseLeave = this.onMouseLeave.bind(this);        this.onMouseEnter = this.onMouseEnter.bind(this);      }      //Được gọi khi chuột được di vào      onMouseEnter() {        this.setState({          opacity: 0.5,        });      }      //Được gọi khi chuột được rời đi      onMouseLeave() {        this.setState({          opacity: 1,        });      }      render() {        return (          <div            style={{ opacity: this.state.opacity }}            onMouseEnter={this.onMouseEnter}            onMouseLeave={this.onMouseLeave}          >            <ImageComponent />          </div>        );      }    };  }; |

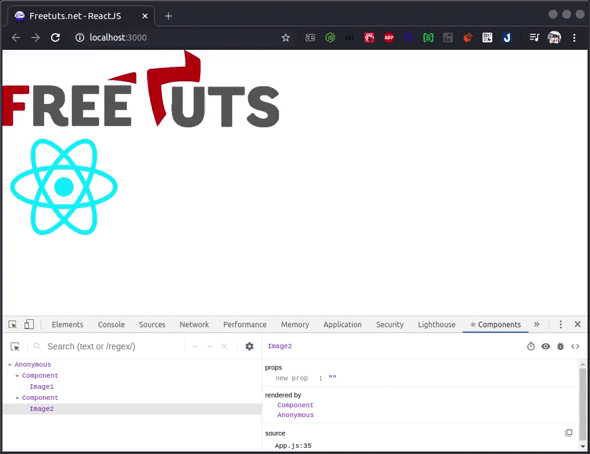
Một HOC nhận vào một component và trả về một component, bên trên mình đã xây dựng thành công một HOC. Tiếp theo, để sử dụng nó bạn chỉ cần gọi nó.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | //Các component là các ảnh cần Hover  const Image1 = (props) => {    return <img src="<https://freetuts.net/public/logo/logo.png>" alt="freetuts" />;  };  const Image2 = (props) => {    return (      <img        src="<https://facebookbrand.com/wp-content/uploads/2019/04/f_logo_RGB-Hex-Blue_512.png?w=512&h=512>"        alt="facebook"      />    );  };    //Lúc này mình truyền vào HOC một component  //và mình sẽ nhận vào một component mới    //Ở đây mình có thể hiển thị rất nhiều ảnh  // mà không cần phải xây dựng component hỗ trợ việc làm  //mờ ảnh quá nhiều  const ImageWithHoverOpacity1 = withHoverOpacity(Image1);  const ImageWithHoverOpacity2 = withHoverOpacity(Image2); |

Lúc này component ImageWithHoverOpacity1, ImageWithHoverOpacity2 là 2 component đã được thêm hiệu ứng làm mờ ảnh bằng HOC. Bây giờ, ta chỉ cần render nó ra:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | //Hiển thi component  export default function () {    return (      <>        <ImageWithHoverOpacity1 />        <ImageWithHoverOpacity2 />      </>    );  } |

Lúc này ta sẽ nhận được kết quả gần giống như đầu bài, nhưng các đọan mã sẽ được tái sử dụng nhiều hơn. Khởi chạy ứng dụng và đây là kết quả:



Như vậy là mình đã giới thiệu xong Higher-Order Components (HOC) trong ReactJS. 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.