**Components trong ReactJS**

Trong bài viết này chúng ta sẽ cùng nhau đi **tìm hiểu về components trong ReactJS**. Trong bài viết đầu tiên giới thiệu về ReactJS mình cũng đã đề cập đến **component**, đây là một phần không thể thiếu trong dự án **ReactJS**. Bài này chúng ta sẽ cùng nhau đi tìm hiểu kĩ hơn về nó.

**Mục lục**

* [1. Component trong ReactJS là gì?](https://freetuts.net/components-trong-reactjs-2348.html#goto-h2-0)
* [2. Khởi tạo một React Component](https://freetuts.net/components-trong-reactjs-2348.html#goto-h2-1)
  + [Functional Component](https://freetuts.net/components-trong-reactjs-2348.html#goto-h3-0)
  + [Class Component](https://freetuts.net/components-trong-reactjs-2348.html#goto-h3-1)

## 1. Component trong ReactJS là gì?

**Components** giúp phân chia các UI (giao diện người dùng) thành các phân nhỏ để dễ dàng quản lý và tái sử dụng. Giả sử mình có một website gồm nhiều phần bố cục khác nhau và mình muốn chia nhỏ các phần ra để dễ quản lý.

Ở hình ảnh bên trên chúng ta có thể chia nó thành 2 components, đó là phần "khóa học nổi bật" và "xu hướng kiếm tiền". Mỗi **components** sẽ đảm nhiệm phần hiển thị khác nhau. Khi bạn muốn làm một trang hoàn chỉnh chỉ ghép các components này lại với nhau.

Trong mỗi React App đều có thể chứa rất nhiều components, mỗi **components** trong đó thường nhận về các props và trả về React elements từ đó hiển thị ra cho UI. **Components trong React** thường được viết theo 2 loại chính đó là functional component và class components. Bên dưới là một functional components:

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

|  |  |
| --- | --- |
| 1 | const App = () =>  <h1>Hello Freetuts.net</h1>; |

**Components** bên trên được viết theo cú pháp ES6, không nhận bất cứ props nào và trả về một react element.

## 2. Khởi tạo một React Component

Ở đây mình đề cập 2 cách để khởi tạo một **components**, mỗi cách có ưu và nhược điểm riêng và bạn có thể lựa chọn theo từng trường hợp sử dụng. Trước khi thực hiện viết components, chúng ta nên khởi tạo một thư mục có tên components trong thư mục src để chứa tất cả các component trong dự án. Cấu trúc thư mục của dự án lúc này sẽ là:

public/

node\_modules/

src/

-----components/

---------------Components sẽ viết ở trong thư mục này

-----App.js

-----index.js

---- vv....

packages.json

packages-lock.json

### Functional Component

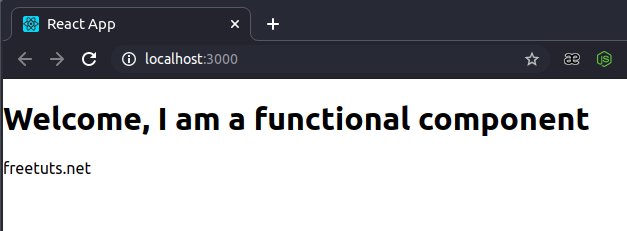
Đây là cách viết phổ biến và được sử dụng nhiều nhất trong các dự án mà mình viết, bởi tính nhanh gọn và không quá phức tạp như các viết class component. Tiến hành tạo một file có tên Welcome.js trong thư mục src/components:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | //Import react vào trong dự án  import React from "react";    const Welcome = function(props) {    return (      <div>        <h1>Welcome ! I am a functional component </h1>      </div>    )  }    export default Welcome; |

Bạn cũng có thể nhận các props được truyền vào component bằng cách nhận một tham số trong funciton. Chúng ta sẽ return các react element bằng cách sử dụng cú pháp return() như ví dụ bên trên, cuối file bạn cần phải export component này để file khác có thể lấy để sử dụng. Bây giờ chúng ta sẽ tiến hành import file Welcome.js vào trong file App.js và hiển thị nó ra :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import React from 'react';  import Welcome from './components/Welcome'  function App() {    return (      <div>          <Welcome />          <p>freetuts.net</p>      </div>    );  }    export default App; |

Tiến hành khởi chạy dự án chúng ta sẽ có kết quả như hình :



Vậy khi nào ban cần sử dụng**funcitonal component**? Mình hay sử dụng functional component nhất bởi nó rất gọn nhẹ và không quá nhiều dòng code dài dòng.

Có một vấn đề là trongfunctional component thì sẽ không có các khái niệm như state, life cycles, events,...nhưng trong phiên bản gần đây thì React cũng đã hỗ trợ thêm React Hooks cho phép bạn làm việc với state, lifecycles dễ dàng hơn trong funcitonal components.

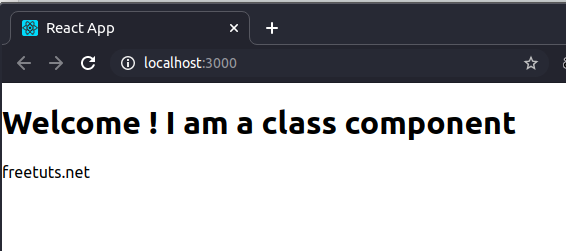
Bởi vậy, bây giờ hầu như **funcitonal component** có hầu hết chức năng giống như **class component**nên bạn có thể cân nhắc sử dụng nó nhiều hơn.

### Class Component

Cách viết này là cách viết đầy của của một component, khi bạn viết một **class component** bạn sẽ sử dụng được hầu hết các chức năng của component như state, props, lifecycle,..Chúng ta tiếp tục với ví dụ trên, chúng ta chỉ sửa đổi file src/Welcome.js :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import React, { Component } from "react";  class Welcome extends Component {    render() {      return (        <div>          <h1>Welcome ! I am a class component </h1>        </div>      );    }  }  export default Welcome; |

Khởi chạy mà bạn sẽ nhận được kết quả giống như ví dụ bên trên.



Khi bạn muốn làm việc với với các chức năng của component như events, state, lifce cycles hay tổ chức các đoạn code theo cấu trúc theo mô hình OOP thì bạn có thể cân nhắc sử dụng **class components**. Ngược lại, bạn có thể sử dụng functional component để thay thế.

Trong bài này chúng ta đã cùng nhau đi tìm hiểu về components 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.