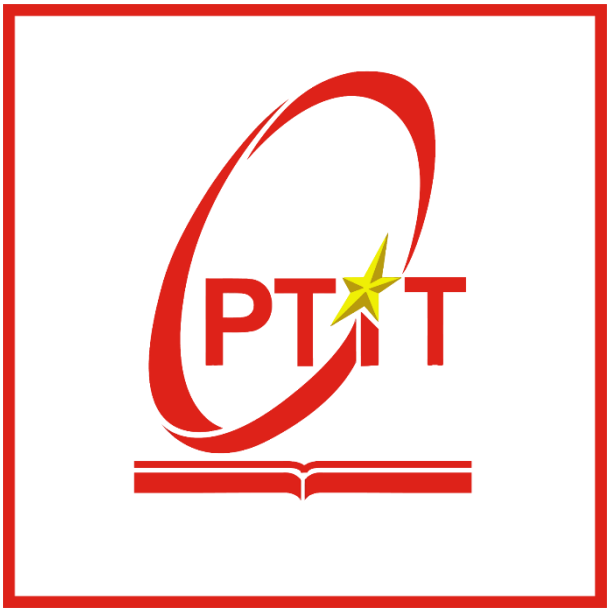
BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 4

TÌM HIỂU VỀ REACT JS

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

Lê Quang Thanh – B22DCVT509

MỤC LỤC

I. REACT JS là gì?

- React (ReactJS) là một thư viện JavaScript mã nguồn mở, được dùng để xây dựng giao diện người dùng (frontend) cho web. React chỉ tập trung vào phần hiển thị giao diện (view), chứ không can thiệp vào cách sắp xếp logic nghiệp vụ hoặc cấu trúc ứng dụng.

- Nguyên lý xây dựng của React dựa trên components (component-based approach), có thể tái sử dụng và phù hợp với ứng dụng 1 trang (Single Page Application – SPA). React giúp lập trình viên xây dựng giao diện người dùng dựa trên JSX (môt cú pháp mở rộng của JavaScript), tạo ra các DOM ảo (virtual DOM) để tối ưu việc render 1 trang web.

II. REACT hoạt động thế nào?

1. Component-Based Architecture

- React sử dụng kiến trúc dựa trên component, trong đó giao diện người dùng được chia thành các thành phần nhỏ, độc lập và có thể tái sử dụng. Mỗi component trong React là một lớp hoặc một hàm JavaScript, chứa logic và giao diện riêng biệt. Các component có thể lồng vào nhau để tạo ra cấu trúc UI phức tạp.

Ví dụ về một component đơn giản trong React:

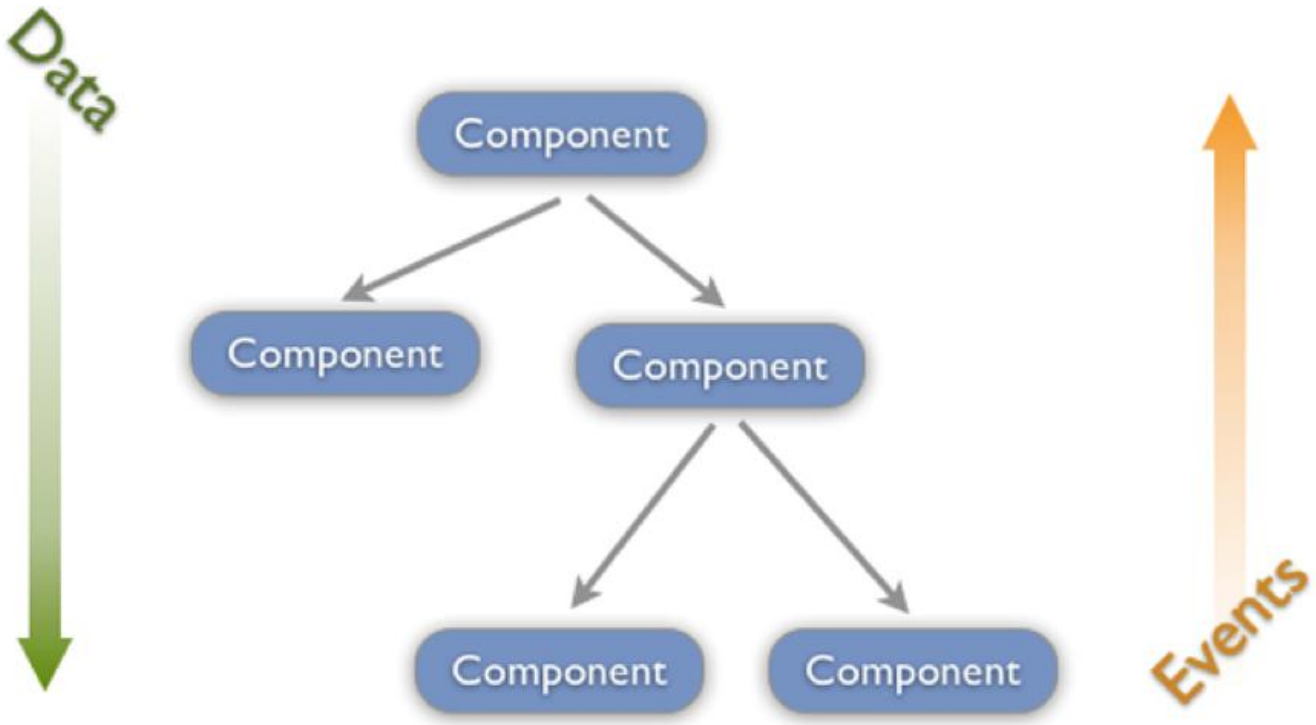
```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}

function App() {
  return (
    <div>
      <Greeting name="Alice" />
      <Greeting name="Bob" />
      <Greeting name="Charlie" />
    </div>
  );
}
```

2. JSX (JavaScript XML)

- Trong React, thay vì thường xuyên sử dụng JavaScript để thiết kế bố cục trang web thì sẽ dùng JSX. JSX được đánh giá là sử dụng đơn giản hơn JavaScript

và cho phép trích dẫn HTML cũng như việc sử dụng các cú pháp thẻ HTML để render các subcomponent. JSX tối ưu hóa code khi biên soạn, vì vậy nó chạy nhanh hơn so với code JavaScript tương đương.

3. Single-way data flow (Luồng dữ liệu một chiều)



- ReactJS không có những module chuyên dụng để xử lý data, vì vậy ReactJS chia nhỏ view thành các component nhỏ có mỗi quan hệ chặt chẽ với nhau.
- Luồng truyền dữ liệu trong ReactJS: Luồng dữ liệu một chiều từ cha xuống con. Việc ReactJS sử dụng one-way data flow có thể gây ra một chút khó khăn cho những người muốn tìm hiểu và ứng dụng vào trong các dự án. Tuy nhiên, cơ chế này sẽ phát huy được ưu điểm của mình khi cấu trúc cũng như chức năng của view trở nên phức tạp thì ReactJS sẽ phát huy được vai trò của mình.

4. Virtual DOM

- Những Framework sử dụng Virtual-DOM như ReactJS khi Virtual-DOM thay đổi, chúng ta không cần thao tác trực tiếp với DOM trên View mà vẫn phản ánh được sự thay đổi đó. Do Virtual-DOM vừa đóng vai trò là Model, vừa đóng vai trò là View nên mọi sự thay đổi trên Model đã kéo theo sự thay đổi trên View và ngược lại. Có nghĩa là mặc dù chúng ta không tác động trực tiếp vào các phần tử DOM ở View nhưng vẫn thực hiện được cơ chế Data-binding.

Điều này làm cho tốc độ ứng dụng tăng lên đáng kể – môt lợi thế không thể tuyệt vời hơn khi sử dụng Virtula-DOM.

5. State và Props

- State: State là một đối tượng quản lý dữ liệu động trong component. Khi state thay đổi, React sẽ tự động cập nhật giao diện để phản ánh các thay đổi này. State thường được sử dụng trong các component stateful, nghĩa là các component có trạng thái thay đổi theo thời gian.
- Props: Props (viết tắt của properties) là các giá trị được truyền từ component cha xuống component con. Props giúp truyền dữ liệu và các hàm giữa các component và không thể thay đổi trong component con (component stateless).

Ví dụ về sử dụng state và props:

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  increment = () => {
    this.setState({ count: this.state.count + 1 });
  };

  render() {
    return (
      <div>
        <h1>Count: {this.state.count}</h1>
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}

function App() {
  return (
    <div>
      <Counter />
    </div>
  );
}
```

6. Lifecycle Methods

- Các component trong React có các phương thức vòng đời (lifecycle methods) cho phép lập trình viên can thiệp vào các giai đoạn khác nhau của vòng đời component. Một số phương thức vòng đời phổ biến:

- componentDidMount: Được gọi sau khi component được render lần đầu tiên. Thường dùng để thực hiện các tác vụ như gọi API.
- componentDidUpdate: Được gọi sau khi component được cập nhật (state hoặc props thay đổi).
- componentWillUnmount: Được gọi ngay trước khi component bị gỡ bỏ khỏi DOM. Thường dùng để dọn dẹp các tài nguyên như bộ đếm thời gian hoặc kết nối mạng.

7. Handling Events

- React xử lý các sự kiện bằng cách sử dụng cú pháp camelCase và truyền hàm sự kiện trực tiếp trong JSX. Điều này tương tự như cách làm việc với các sự kiện trong DOM, nhưng với React, không cần phải sử dụng addEventListener.

III. Những điểm nổi bật của REACT

1. Linh hoạt trong Thiết kế kiến trúc

- React tập trung vào việc hiển thị giao diện người dùng và cho phép lập trình viên tự do quyết định cách sắp xếp logic nghiệp vụ. Chính sự linh hoạt này làm cho React trở nên phổ biến, vì nó phù hợp với nhiều loại dự án và phong cách phát triển khác nhau.
- Khác với các framework có kiến trúc cố định như Angular, React không ép buộc người dùng vào một mô hình cụ thể, khiến nó linh hoạt cho nhiều dự án khác nhau. Mặc dù điều này trong React đem lại lợi ích cho lập trình viên có kinh nghiệm, nhưng lại gây khó khăn cho người mới bắt đầu vì thiếu sự hướng dẫn cụ thể.
- Trong khi đó Angular bắt buộc phải tách biệt logic thêm/xóa công việc vào service (TodoService) và component chỉ để hiển thị danh sách công việc. Bạn không thể viết mọi thứ trong cùng một nơi như React, Angular quy định rõ ràng cách thức tổ chức ứng dụng.

2. Kiến trúc Component đơn giản và nhẹ

- React được xây dựng dựa trên kiến trúc component, nơi mỗi thành phần có thể được tái sử dụng, giúp ứng dụng dễ mở rộng và duy trì. Các component trong React rất nhẹ và có thể chỉ là các hàm đơn giản trả về JSX.

- Điều này giúp các lập trình viên dễ dàng phát triển các ứng dụng lớn hơn bằng cách chia nhỏ thành các thành phần độc lập, có thể tái sử dụng mà không cần phải lo về việc tích hợp phức tạp.