**EVENT – FORM – LIST&KEY**

Mục lục:

[1. GIỚI THIỆU 2](#_Toc208385906)

[2. CƠ SỞ LÝ THUYẾT 2](#_Toc208385907)

[2.1. Event trong ReactJS 2](#_Toc208385908)

[2.2. Form trong ReactJS 2](#_Toc208385909)

[2.3. Render có điều kiện 3](#_Toc208385910)

[2.4. List và Key 3](#_Toc208385911)

[3. THIẾT KẾ VÀ MINH HỌA 3](#_Toc208385912)

[3.1. Ví dụ về Event 3](#_Toc208385913)

[3.2. Ví dụ về Form 3](#_Toc208385914)

[3.3. Ví dụ về Render có điều kiện 4](#_Toc208385915)

[3.4. Ví dụ về List & Key 5](#_Toc208385916)

[4. ỨNG DỤNG THỰC TẾ 5](#_Toc208385917)

[5. ĐÁNH GIÁ ƯU – NHƯỢC ĐIỂM 5](#_Toc208385918)

[6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 5](#_Toc208385919)

[7. TÀI LIỆU THAM KHẢO 6](#_Toc208385920)

## 1. GIỚI THIỆU

Trong lập trình giao diện người dùng hiện đại, **ReactJS** là một thư viện JavaScript phổ biến giúp phát triển ứng dụng web động. Ba khái niệm quan trọng cần nắm vững để xây dựng ứng dụng React là: **Event, Form, List & Key**.

* **Event** cho phép ứng dụng phản hồi tương tác của người dùng.
* **Form** cung cấp giao diện nhập liệu.
* **Render có điều kiện** cho phép hiển thị linh hoạt dựa trên trạng thái.
* **List & Key** giúp quản lý và hiển thị danh sách dữ liệu hiệu quả.

Đây là những thành phần nền tảng, xuất hiện trong hầu hết các ứng dụng React thực tế, từ form đăng nhập đơn giản cho đến hệ thống quản lý dữ liệu phức tạp.

## 2. CƠ SỞ LÝ THUYẾT

### 2.1. Event trong ReactJS

* Trong HTML truyền thống, event thường được viết trực tiếp trong thẻ.
* Trong JSX (React), event được viết dưới dạng **camelCase** và gán cho một function.
* Ví dụ:
  + onClick={handleClick} → hàm handleClick sẽ được gọi khi click.
  + Phân biệt onClick={fncName} và onClick={fncName()}.

Các loại event phổ biến:

* **Không có tham số** → gọi trực tiếp function.
* **Có tham số** → dùng arrow function hoặc bind.
* **Sử dụng props/state** để thay đổi giao diện động.
* **Event trên thẻ <a>** → sử dụng e.preventDefault() để ngăn hành vi mặc định.

### 2.2. Form trong ReactJS

Form trong React chia làm 2 loại:

* **Controlled Component**
  + Dữ liệu trong form được quản lý bởi **state**.
  + Mỗi lần thay đổi input sẽ kích hoạt setState.
  + Dữ liệu luôn đồng bộ giữa giao diện và state.
* **Uncontrolled Component**
  + Dữ liệu được quản lý trực tiếp bởi DOM.
  + Sử dụng **ref** để truy cập giá trị input.
  + Thường dùng cho các input đặc biệt (file, media).

### 2.3. Render có điều kiện

* Cho phép hiển thị component tùy theo state.
* Có thể sử dụng:
  + Câu lệnh if thông thường.
  + Toán tử ba ngôi ? :.
  + Biểu thức logic &&.
* Giúp giao diện linh hoạt, tránh hiển thị dư thừa.

### 2.4. List và Key

* Khi render danh sách từ mảng, React yêu cầu **key** cho mỗi phần tử.
* **Key** giúp React xác định phần tử nào thay đổi, thêm hoặc xóa.
* Key cần **duy nhất** trong danh sách.
* Có thể dùng **index** làm key, nhưng không khuyến khích khi dữ liệu thay đổi thường xuyên.

## 3. THIẾT KẾ VÀ MINH HỌA

### 3.1. Ví dụ về Event

function Welcome() {

function handleClick() {

alert("Xin chào!");

}

return <button onClick={handleClick}>Click me</button>;

}

### 3.2. Ví dụ về Form

* **Controlled Component**

function LoginForm() {

const [username, setUsername] = useState("");

const handleChange = (e) => setUsername(e.target.value);

return (

<form>

<input type="text" value={username} onChange={handleChange} />

<p>Xin chào, {username}</p>

</form>

);

}

* **Uncontrolled Component**

function FileUpload() {

const fileInput = useRef(null);

const handleSubmit = (e) => {

e.preventDefault();

alert(`File: ${fileInput.current.files[0].name}`);

};

return (

<form onSubmit={handleSubmit}>

<input type="file" ref={fileInput} />

<button type="submit">Upload</button>

</form>

);

}

### 3.3. Ví dụ về Render có điều kiện

function UserGreeting({ isLoggedIn }) {

return (

<div>

{isLoggedIn ? <h1>Chào mừng bạn quay lại!</h1> : <h1>Vui lòng đăng nhập</h1>}

</div>

);

}

### 3.4. Ví dụ về List & Key

function NumberList({ numbers }) {

return (

<ul>

{numbers.map((num) => (

<li key={num}>{num}</li>

))}

</ul>

);

}

## 4. ỨNG DỤNG THỰC TẾ

* **Event**: nút bấm like/share, điều khiển game, điều hướng menu.
* **Form**: đăng nhập, đăng ký, giỏ hàng thương mại điện tử.
* **Render có điều kiện**: hiển thị thông báo khi chưa đăng nhập, render dữ liệu động.
* **List & Key**: hiển thị danh sách sản phẩm, bảng dữ liệu, tin nhắn trong chat app.

## 5. ĐÁNH GIÁ ƯU – NHƯỢC ĐIỂM

* **Event**
  + Ưu điểm: dễ quản lý, gọn gàng.
  + Nhược điểm: khi nhiều event lồng nhau dễ gây khó bảo trì.
* **Form**
  + Ưu điểm: Controlled component cho phép quản lý chặt chẽ dữ liệu.
  + Nhược điểm: Uncontrolled component khó kiểm soát logic.
* **Render có điều kiện**
  + Ưu điểm: giao diện linh hoạt, thân thiện người dùng.
  + Nhược điểm: nếu quá nhiều điều kiện có thể gây rối.
* **List & Key**
  + Ưu điểm: tối ưu hiệu suất, dễ quản lý.
  + Nhược điểm: nếu key không duy nhất → bug khó phát hiện.

## 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

**Event – Form – Render có điều kiện – List & Key** là nền tảng không thể thiếu khi lập trình với ReactJS. Việc nắm vững giúp:

* Xây dựng ứng dụng dễ mở rộng, dễ bảo trì.
* Tăng hiệu suất xử lý dữ liệu và trải nghiệm người dùng.

**Hướng phát triển**:

* Kết hợp với **Context API**, **Redux** để quản lý state toàn cục.
* Tối ưu hiệu suất render bằng kỹ thuật **memoization**.
* Ứng dụng trong các dự án thực tế như thương mại điện tử, mạng xã hội, hệ thống quản trị.

## 7. TÀI LIỆU THAM KHẢO

1. ReactJS Official Documentation – https://react.dev
2. W3Schools React Tutorial – https://www.w3schools.com/react
3. Slide giảng dạy: Session 03 - Event-Form-ListKey
4. GeeksforGeeks – ReactJS Guide