**1. Cách ReactJS tạo form, xử lý form và triển khai validation cho form**

**a. Tạo form trong ReactJS**

Trong React, form được tạo bằng cách sử dụng các thẻ HTML như <input>, <textarea>, <select> kết hợp với state để quản lý dữ liệu đầu vào của người dùng. React sử dụng các **Controlled Components** để kiểm soát giá trị của các phần tử form thông qua state.

**b. Xử lý form trong ReactJS**

React có hai cách chính để xử lý form:

1. **Controlled Components**: Các thành phần của form được điều khiển hoàn toàn bởi state. Khi người dùng nhập dữ liệu, sự kiện onChange cập nhật state, giúp React kiểm soát toàn bộ dữ liệu của form.
2. **Uncontrolled Components**: Dữ liệu của form không được lưu trữ trong state, mà được lấy trực tiếp từ DOM thông qua ref.

**c. Triển khai validation cho form trong ReactJS**

Validation trong React có thể được triển khai theo hai cách:

1. **Validation thủ công**: Kiểm tra dữ liệu đầu vào bằng cách sử dụng state và điều kiện trong sự kiện onSubmit.
2. **Validation bằng thư viện**: Sử dụng các thư viện như **React Hook Form** kết hợp với **Yup** để kiểm tra và hiển thị lỗi một cách tối ưu.

**2. React Router là gì?**

React Router là một thư viện quản lý điều hướng trong React. Nó giúp tạo các đường dẫn khác nhau trong ứng dụng mà không cần tải lại trang, giúp xây dựng **Single Page Application (SPA)**.

Các thành phần chính của React Router bao gồm:

* BrowserRouter: Cung cấp môi trường để quản lý routing bằng lịch sử trình duyệt.
* Routes và Route: Xác định các đường dẫn và component tương ứng.
* Link và NavLink: Tạo liên kết điều hướng giữa các trang.
* useNavigate và useParams: Hỗ trợ điều hướng và lấy tham số từ URL.

**3. Tại sao nên sử dụng Router trong React?**

Sử dụng Router trong React mang lại nhiều lợi ích:

1. **Tạo SPA (Single Page Application)**: Giúp điều hướng trang nhanh mà không cần tải lại.
2. **Quản lý URL hiệu quả**: Cho phép cập nhật nội dung động theo đường dẫn mà không ảnh hưởng đến hiệu suất ứng dụng.
3. **Cải thiện trải nghiệm người dùng**: Điều hướng mượt mà, giảm độ trễ so với việc tải lại toàn bộ trang.
4. **Dễ bảo trì và mở rộng**: Tách biệt rõ ràng giữa các trang và component, giúp code dễ quản lý.
5. **Hỗ trợ Nested Routes & Lazy Loading**: Cho phép tạo các tuyến đường lồng nhau và tải trang theo nhu cầu để tối ưu hiệu suất.