# 3. CÁC VẤN ĐỀ CƠ BẢN CỦA FRONTEND

## 1. Hiển thị UI

## 1.1. Định nghĩa

* Hiển thị UI (viết tắt từ User Interface Display) là các yếu tố trực quan mà người dùng nhìn thấy và tương tác khi sử dụng một sản phẩm kỹ thuật số, chẳng hạn như trang web, ứng dụng di động hoặc phần mềm máy tính. Nó bao gồm tất cả các thành phần đồ họa, văn bản và các yếu tố tương tác mà người dùng sử dụng để thực hiện các tác vụ và điều hướng trong sản phẩm.

## 1.2. Phương thức xử lý

## 1.3. UI rendering trong React

* Component: Mọi thứ trong React đều là một component, bao gồm cả các phần tử HTML cơ bản. Mỗi component có một hàm render() trả về mô tả giao diện người dùng của nó.
* Virtual DOM: React sử dụng một cây DOM ảo để biểu diễn giao diện người dùng. Cây DOM ảo hiệu quả hơn nhiều so với DOM gốc vì nó chỉ cập nhật các phần của giao diện người dùng đã thay đổi thay vì render lại toàn bộ giao diện.
* Diffing: React sử dụng thuật toán diffing để so sánh cây DOM ảo trước đó với cây DOM ảo hiện tại. Thuật toán này xác định các phần của giao diện người dùng đã thay đổi và chỉ cập nhật các phần đó trong DOM gốc.
* Updating the DOM: Sau khi diffing, React cập nhật DOM gốc để phản ánh các thay đổi được xác định trong bước trước.
* React sử dụng một số kỹ thuật để tối ưu hóa quá trình rendering:
  + Batching: React sẽ gom các cập nhật trạng thái liên tiếp thành một lần cập nhật duy nhất để giảm thiểu số lần render.
  + Conditionally rendering: React cho phép bạn render các phần tử giao diện người dùng có điều kiện, nghĩa là chỉ render các phần tử cần thiết cho trạng thái hiện tại của ứng dụng.
  + PureComponent: React cung cấp lớp component PureComponent giúp tối ưu hóa hiệu suất rendering bằng cách chỉ render lại component khi các prop hoặc state của nó thay đổi.
* Một số khái niệm quan trọng khác liên quan đến UI rendering trong React:
  + State: State là dữ liệu cục bộ được lưu trữ trong mỗi component. Khi state của component thay đổi, component sẽ được render lại.
  + Props: Props là dữ liệu được truyền từ component cha sang component con. Props không ảnh hưởng trực tiếp đến việc rendering của component, nhưng chúng có thể được sử dụng trong hàm render() của component để tạo ra giao diện người dùng động.
  + Lifecycle methods: React cung cấp một số lifecycle methods cho phép bạn thực hiện các hành động nhất định tại các thời điểm khác nhau trong vòng đời của component, chẳng hạn như khi component được mounted hoặc unmounted.

## 2. Tương tác UI

### 2.1. Giải quyết vấn đề

### 2.2. Tương tác trong React

* Dưới đây là một số khái niệm quan trọng về tương tác UI trong React:
  + Sự kiện: React sử dụng hệ thống sự kiện để theo dõi và phản hồi các hành động của người dùng. Các sự kiện phổ biến bao gồm nhấp chuột, cuộn, thay đổi kích thước và nhập văn bản.
  + Thành phần: Các thành phần React là các khối xây dựng cơ bản của giao diện người dùng. Mỗi thành phần có thể quản lý trạng thái và xử lý các sự kiện của riêng nó.
  + Hooks: React Hooks là các hàm đặc biệt cho phép bạn "móc" vào trạng thái và vòng đời của component mà không cần viết lớp. Hooks cung cấp một cách linh hoạt và gọn nhẹ để xử lý các tương tác UI.
  + Thư viện: Có nhiều thư viện React của bên thứ ba hỗ trợ việc xây dựng các tương tác UI phức tạp, chẳng hạn như react-router, react-motion và react-spring.
* Để tương tác UI hiệu quả trong React, bạn nên tuân thủ các nguyên tắc sau:
  + Rõ ràng và nhất quán: Thiết kế các tương tác UI rõ ràng, dễ hiểu và nhất quán trong suốt ứng dụng của bạn.
  + Phản hồi: Cung cấp phản hồi trực quan cho các hành động của người dùng để đảm bảo họ biết rằng hành động của họ đã được thực hiện.
  + Hiệu quả: Tối ưu hóa hiệu suất của các tương tác UI để mang lại trải nghiệm mượt mà và nhanh chóng.
  + Khả năng sử dụng: Đảm bảo rằng các tương tác UI của bạn có thể sử dụng được bởi tất cả mọi người, bất kể khả năng của họ.
  + Kiểm thử: Kiểm tra kỹ lưỡng các tương tác UI của bạn với người dùng thực tế để xác định và sửa lỗi.
* Một số ví dụ:
  + Điều hướng: React cung cấp các thư viện như react-router để tạo ra các hệ thống điều hướng mượt mà và dễ sử dụng.
  + Biểu mẫu: React cho phép bạn tạo các biểu mẫu trực quan và dễ tương tác bằng cách sử dụng các thành phần HTML và xử lý sự kiện.
  + Phản hồi động: Bạn có thể sử dụng Hooks và thư viện như react-motion và react-spring để tạo ra các hiệu ứng động và phản hồi mượt mà cho các hành động của người dùng.
  + Khả năng tiếp cận: React hỗ trợ các tính năng trợ năng để đảm bảo rằng ứng dụng của bạn có thể sử dụng được bởi tất cả mọi người, bất kể khả năng của họ.