**Nhiệm vụ 1: JSX & Virtual DOM**

1.**JSX & Virtual DOM là gì ?**

**1.1 Giới thiệu JSX**

JSX (JavaScript XML) là một cú pháp mở rộng của JavaScript, cho phép chúng ta viết mã HTML ngay trong file JavaScript.

JSX là một tính năng chủ yếu trong React giúp mô tả UI dễ dàng, trực quan hơn.

JSX không phải là HTML, mà là cú pháp được biên dịch thành các lời gọi hàm JavaScript – cụ thể là React.createElement().

Ví dụ JSX:

const element = <h1>Chào bạn!</h1>;

Sau khi được biên dịch bởi Babel:

const element = React.createElement('h1', null, 'Chào bạn!');

JSX cần trình biên dịch, vì trình duyệt không hiểu trực tiếp cú pháp này. Thường dùng Babel để chuyển JSX sang JavaScript thuần.

**1.2 Khái niệm Virtual DOM**

Virtual DOM là một đối tượng JavaScript mô phỏng cây DOM thật của trình duyệt.

Khi trạng thái (state) của một component thay đổi, React tạo ra một Virtual DOM mới.

React sẽ so sánh (diff) giữa Virtual DOM mới và Virtual DOM cũ để tìm sự thay đổi.

Sau đó, React chỉ cập nhật phần bị thay đổi lên DOM thật (reconciliation).

Cách tiếp cận này tối ưu hiệu năng, tránh việc thao tác DOM nhiều lần – vốn là một trong những thao tác chậm trong trình duyệt.

**2.1 Định nghĩa JSX, lợi ích và hạn chế**

Định nghĩa JSX:

Là cú pháp giúp viết các phần tử React giống như HTML nhưng bên trong JavaScript.

Không phải là HTML mà được dịch thành hàm React.createElement.

Lợi ích:

Viết code dễ nhìn, dễ hiểu hơn.

Gần giống HTML nên thân thiện với người dùng mới.

Có thể sử dụng JavaScript trong JSX thông qua {}.

Hỗ trợ tree structure rõ ràng – dễ quản lý layout UI.

Hạn chế:

Phải qua bước biên dịch (compile).

Cần học một số quy tắc cú pháp khác HTML (vd: className thay cho class, htmlFor thay cho for).

Có thể gây khó khăn khi debug với người mới nếu chưa hiểu rõ cách biên dịch.

**2.2 Cơ chế chuyển JSX sang React.createElement()**

JSX sẽ được Babel dịch sang dạng React.createElement(component, props, ...children).

Ví dụ:

const element = <button disabled={true}>Bấm vào</button>;

Sau khi dịch:

const element = React.createElement(

'button',

{ disabled: true },

'Bấm vào'

);

Đây là cách React dựng nên “cây” các phần tử, từ đó tạo ra Virtual DOM.

**2.3 Nguyên lý hoạt động của Virtual DOM**

Diffing:

So sánh hai cây Virtual DOM (mới và cũ).

Tìm ra các điểm khác biệt nhỏ nhất giữa chúng.

Reconciliation:

Dựa vào diffing, React sẽ đưa ra kế hoạch cập nhật DOM thật một cách tối ưu, chỉ cập nhật phần thay đổi.

Quá trình:

Component được render lần đầu → tạo Virtual DOM.

Khi có sự thay đổi (state/props) → tạo Virtual DOM mới.

So sánh với Virtual DOM cũ → tìm khác biệt.

Cập nhật DOM thật tương ứng.

**2.4 Lợi ích của Virtual DOM về hiệu năng**

Giảm thao tác DOM thật: thay vì cập nhật trực tiếp từng thay đổi nhỏ, React gom và tối ưu việc cập nhật.

Tăng tốc độ render: chỉ cập nhật phần thay đổi.

Tránh “giật lag” UI: do thao tác DOM thật thường tốn thời gian.

Cải thiện hiệu suất ứng dụng lớn, nhiều component con.

**3. Demo**

**3.1 JSX Playground**

Một công cụ cho phép gõ JSX và xem kết quả chạy trực tiếp.

Có thể dùng Babel Standalone (qua CDN) để chuyển JSX thành JavaScript trên trình duyệt.

Ví dụ setup đơn giản:

Một textarea để nhập JSX.

Một div để hiển thị kết quả sau khi compile.

**3.2 Mini App hiển thị danh sách**

App hiển thị danh sách (ví dụ: tên sinh viên).

Có nút “Cập nhật” → danh sách thay đổi.

In ra console các bước diff DOM để minh họa hoạt động của Virtual DOM.

Dùng React DevTools để quan sát sự thay đổi component tree.

**4. Nội dung tổng kết**

JSX: giúp lập trình viên viết giao diện dễ dàng, trực quan hơn, nhưng cần biên dịch để trình duyệt hiểu.

Virtual DOM: giúp tăng hiệu suất bằng cách tránh thao tác trực tiếp với DOM thật mỗi khi có thay đổi UI.

Kết hợp cả hai là lý do khiến React trở nên mạnh mẽ và phổ biến.