**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÀI TẬP LỚN**

**BỘ MÔN NỀN TẢNG PHÁT TRIỂN WEB**

**Đề Tài:**

**JSX & VIRTUAL DOM**

|  |  |
| --- | --- |
| **Nhóm thực hiện:** | **Nhóm 20** |
| **Các thành viên:** | 1. **Trương Tuấn Hải** 2. **Nguyễn Minh Đức** 3. **Nguyễn Đình Tuấn Sơn** |

**I. JSX**

**1. Khái niệm**

- **JSX** (JavaScript XML) là phần mở rộng cú pháp của JavaScript.

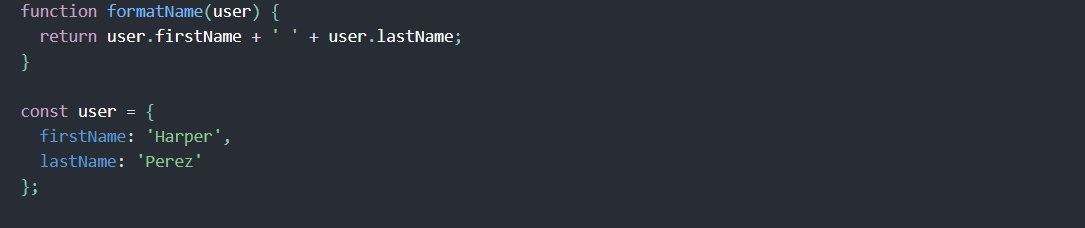
- JSX có thể gợi cho chúng ta nhớ đến HTML, nhưng nó đi kèm với toàn bộ sức mạnh của JavaScript.

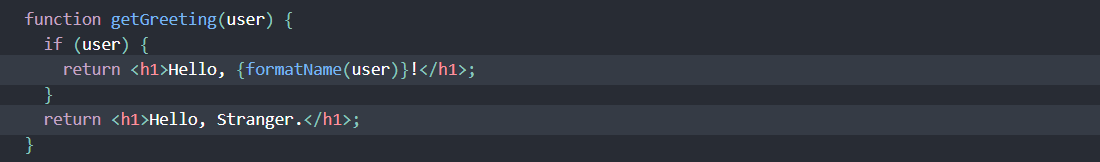
- Trong React chúng ta có thể sử dụng JSX để tạo template thay vì JavaScript thông thường giúp bạn viết ứng dụng dễ dàng hơn.

- Ví dụ sử dụng JSX:

., Nhúng 1 Javascript expression vào trong JSX, ta sử dụng 1 cặp {}  


., Sử dụng JSX như một expression





., Cú pháp để chỉ định thuộc tính trong JSX

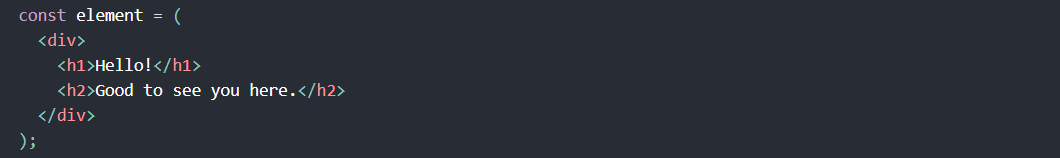
Nếu là 1 xâu thì sử dụng 1 cặp “”



Nếu là 1 javascript expression thì sử dụng 1 cặp {}



., JSX cũng có thể chứa các con như HTML



., Nếu không chứa con thì có thể đóng bằng cú pháp />  


**2. Lợi ích**

- JSX trông rất gần với cú pháp HTML quen thuộc nên giúp lập trình viên dễ đọc và viết giao diện hơn.

- JSX cho phép nhúng JavaScript ngay trong giao diện, giúp xử lý điều kiện, lặp, hay tính toán dữ liệu dễ dàng ngay trong phần hiển thị.

- JSX giúp cấu trúc component React rõ ràng, kết hợp logic và giao diện một cách mạch lạc.

- JSX không phải thuần HTML mà phải qua bước biên dịch, giúp phát hiện lỗi cú pháp hoặc lỗi trong biểu thức JavaScript sớm thông qua các trình biên dịch như Babel, TypeScript….

- Có thể sử dụng mọi tính năng của JavaScript trong JSX

- JSX là cú pháp giúp viết React Element thuận tiện, những element này được **React** dùng trong **Virtual DOM** để tối ưu cập nhật giao diện.

**3. Hạn chế**

- JSX không phải JavaScript thuần, nên cần công cụ như **Babel** để biên dịch → tăng bước build, gây khó khăn cho người mới.

- JSX kết hợp nhiều component → file build ra có thể nặng nếu không dùng các kỹ thuật như **tree shaking** hay **code splitting**.

- Do JSX là phần mở rộng của JavaScript, một số cú pháp khác biệt với HTML thuần, một số thuộc tính/thẻ HTML phải viết lại cho đúng cú pháp JSX dễ gây nhầm lẫn(ví dụ thuộc tính class của HTML là className trong JSX)

**4. Cơ chế chuyển JSX**

- Trình duyệt không hiểu JSX ngay lập tức, vì vậy React dựa vào trình biên dịch như Babel hoặc TypeScript để chuyển đổi mã JSX thành JavaScript thông thường . Nhiều bộ công cụ được cấu hình sẵn như Create React App hoặc Next.js có chức năng chuyển đổi JSX.

- Trong các phiên bản trước react 17: 1 JSX sẽ được chuyển thành 1 hàm React.createElement() trong build. Để gọi hàm này cần import thư viện React qua cú pháp “**import React from 'react';**”

-Cú pháp chuyển đổi tương ứng:

., JSX: **const** element = <**Type Properties = {}** > **…Children** </ **Type**>;

., JavaScript:

**var** element = **React.createElement**(

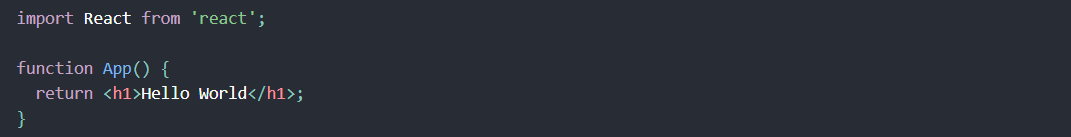
**Type,** //kiểu phần tử html hoặc kiểu phần tử tự định nghĩa

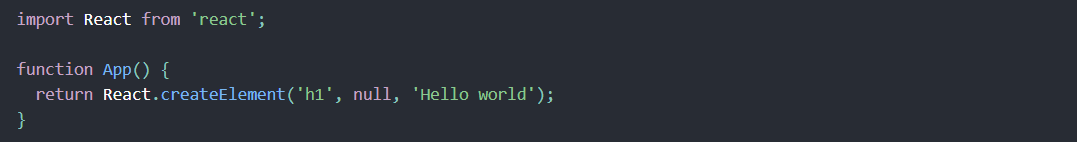
**{Properties = {}},** //thuộc tính

**…Children** //Nội dung hoặc các các React Element con

);

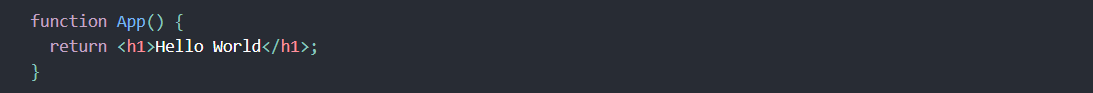
- Ví dụ:

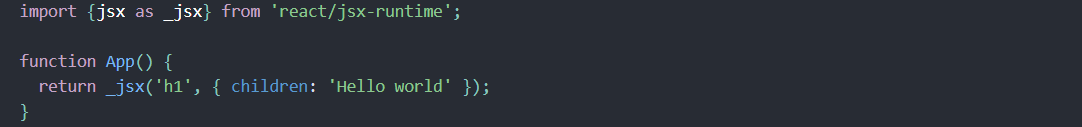




- Trong các phiên bản React 17 trở lên, cú pháp đã được thay đổi, không cần phải **import React** để sử dụng JSX mà sẽ tự động sử dụng 1 module trong react: **react/jsx-runtime** để biên dịch JSX

- Ví dụ:



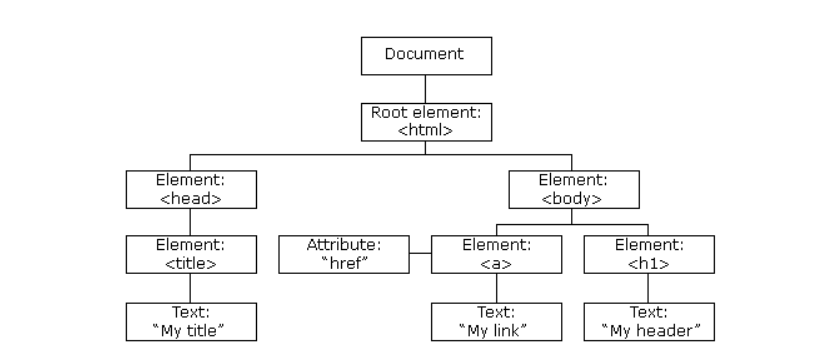


**II. Virtual DOM**

**1. Khái niệm**

- **DOM (Document Object Model)** là một mô hình dạng **cây (tree)** biểu diễn cấu trúc của một trang HTML. Mỗi phần tử trong HTML (thẻ, nội dung, thuộc tính...) sẽ được trình duyệt **chuyển thành các đối tượng (object)** trong cây DOM.

- Ảnh minh họa:



-  **Virtual DOM (DOM ảo)** là một biểu diễn của giao diện người dung (DOM thật) được lưu trong bộ nhớ dưới dạng cây JavaScript, chứ không hiển thị trực tiếp trên trình duyệt và đồng bộ với DOM thật bằng một thư viện như ReactDOM.

**2. Nguyên lý hoạt động**

- DOM ảo là một cấu trúc dữ liệu mà React dung để quản lý giao diện

- DOM ảo là bản sao trong bộ nhớ của cây DOM thật, dưới dạng cấu trúc dữ liệu JavaScript

**Quá trình reconciliation gồm các bước:**

1. **Tạo Virtual DOM mới:**  
   Khi có thay đổi (state/props), React tạo một Virtual DOM mới.
2. **So sánh với Virtual DOM cũ (diffing):**  
   React so sánh cây mới và cây cũ để tìm ra sự khác biệt.
3. **Xác định các thay đổi (patch):**  
   React xác định những node nào cần cập nhật, thêm, xóa.
4. **Cập nhật DOM thật:**  
   React chỉ cập nhật những phần thực sự thay đổi trên DOM thật mà không cần render toàn bộ DOM thật.

**Qúa trình Diffing:**

1. **So sánh (diff) với Virtual DOM cũ:**  
   React sử dụng thuật toán diffing để so sánh cây mới với cây cũ.
   * Nếu một node khác loại (type), React sẽ xóa node cũ và tạo node mới.
   * Nếu cùng loại, React sẽ so sánh props và children để xác định sự thay đổi.
2. **Tìm sự khác biệt (diff):**
   * React đi từ trên xuống dưới (top-down), so sánh từng node.
   * Với danh sách (arrays), React dùng key để xác định phần tử nào thay đổi, thêm, xóa.

**Một số quy tắc reconciliation của React:**

* Nếu component cùng loại (cùng key và type), React giữ lại DOM cũ và chỉ cập nhật props/conten.
* Nếu khác loại, React xóa node cũ và tạo node mới.
* Với danh sách, React dùng **key** để xác định phần tử nào thay đổi, giúp cập nhật hiệu quả.

**3. So sánh giữa virtual DOM và DOM**

|  |  |  |
| --- | --- | --- |
|  | ***DOM ảo*** | ***DOM thật*** |
| ***Tốc độ thao tác trực tiếp*** | Nhanh, do thao tác trực tiếp trên bộ nhớ máy người dùng | Chậm, do thao tác với cấu trúc của trình duyệt |
| ***Tần suất cập nhật, khả năng tối ưu*** | Tối ưu hóa số lần cập nhật của toàn bộ UI | UI được cập nhật nhiều lần, dễ dàng bị cập nhật thừa, không tối ưu gây lag |
| ***Khả năng phát triển và bảo trì*** | Cấu trúc component rõ ràng, dễ tái sử dụng, tách biệt logic và giao diện trong một đơn vị | Viết code trực tiếp trên DOM phức tạp, dễ lỗi khi UI phứ tạp |
| ***Khả năng mở rộng*** | Dễ mở rộng và phát triển tính năng mới với kiến trúc component (kiến trúc chia nhỏ giao diện thành các thành phần độc lập trong React) | Khó mở rộng, bảo trì khi ứng dụng lớn do thao tác trực tiếp trên DOM dễ gây ra tình trạng spaghetti code |

**III. Kết luận**

**JSX – JavaScript XML**

* Là cú pháp mở rộng cho JavaScript, giúp viết code JS React giống HTML.
* Không phải JS thuần, nên cần biên dịch thành JavaScript (React.createElement hoặc jsx-runtime).
* Giúp code ngắn gọn, trực quan, dễ thấy được giao diện ngay trong JS.
* Cho phép nhúng biểu thức JS với {} bên trong JSX.
* JSX không bắt buộc – chỉ là cú pháp giúp code dễ viết hơn.

**Virtual DOM (DOM ảo)**

* Là bản mô phỏng trong bộ nhớ của DOM thật.
* Khi UI thay đổi, React thuật toán diffing Virtual DOM mới với bản cũ. Sau đó, React đối chiếu (reconciliation) chỉ cập nhật phần DOM thật cần thiết
* Virtual DOM giảm thao tác trực tiếp lên DOM thật, tăng hiệu năng rerender.