A diagram of a company

Description automatically generated

Có bốn bước để hiển thị các thành phần trên màn hình

Bước đầu tiên, "Render được kích hoạt", xảy ra khi cập nhật trạng thái ở đâu đó. Sau đó, trong Pha Render, React gọi các hàm thành phần và tìm ra cách DOM nên được cập nhật. Tiếp theo, trong pha commit, React thực sự viết vào DOM, cập nhật, chèn và xóa các phần tử. Cuối cùng, có pha vẽ của trình duyệt, nơi tất cả việc hiển thị xảy ra.

Bạn phải nhớ rõ tuyên bố này: "Trong React, Render KHÔNG phải là cập nhật DOM hoặc hiển thị các phần tử trên màn hình. Render chỉ xảy ra nội bộ bên trong React; nó không tạo ra thay đổi trực quan."

Bây giờ, hãy đi sâu vào từng giai đoạn để hiểu rõ hơn các khái niệm:

**1. RENDER IS TRIGGERED**

Trong hai tình huống khi render được kích hoạt:

* Render ban đầu của ứng dụng
* Trạng thái được cập nhật trong một hoặc nhiều trường hợp Component Quá trình render được kích hoạt cho toàn bộ ứng dụng. Chúng ta phải nhớ rõ rằng Render không được kích hoạt ngay lập tức mà được lên lịch khi JS Engine có thời gian rảnh để thực thi chúng. Ngoài ra còn có việc nhóm nhiều cuộc gọi setState lại với nhau trong trình xử lý sự kiện.

Vậy sau khi trigger là gì?

A screenshot of a diagram

Description automatically generated

Sẽ có một cây DOM ảo mới được tạo ra.

Các trường hợp thành phần kích hoạt render lại sẽ trả về một React Element và chèn vào DOM, New Virtual DOM tree.

**2. RENDER PHASE**

A diagram of a flowchart

Description automatically generated

Bước thứ hai là giai đoạn Render. Giai đoạn Render được phát triển bởi một số bước. Trước hết, React Element được cập nhật tạo ra NEW VIRTUAL DOM.

**Virtual DOM là gì?**

React hoạt động trên Virtual DOM. React giữ một đại diện nhẹ của DOM thực trong bộ nhớ được gọi là DOM ảo. Thay vì thao tác trực tiếp với DOM của trình duyệt, React tạo, cập nhật và xóa các node DOM ảo. Khi trạng thái thay đổi trong một thành phần React, hàm render của thành phần đó được gọi và DOM ảo được cập nhật với cây JSX mới.

Việc render một thành phần sẽ khiến tất cả các thành phần con của nó cũng được render lại, cho dù prop của chúng có thay đổi hay không. Điều này là cần thiết vì React không biết liệu các con có bị ảnh hưởng hay không. Có một thay đổi trạng thái trong thành phần, vì vậy tất cả các con trên Parent cũng sẽ render lại.

**Reconciliation**

Reconciliation là một quá trình mà React quyết định những phần tử DOM nào cần được phản ánh trong những thay đổi trạng thái mới nhất.

Quá trình reconciliation bao gồm các bước chính sau:

* React tạo ra một DOM ảo, là bản sao nhẹ của DOM thực ở dạng đối tượng.
* Với mỗi thay đổi trong trạng thái hoặc props của thành phần, React tạo ra một DOM ảo mới.
* React sử dụng thuật toán diffing duyệt qua cây để so sánh/phân tích các thay đổi giữa DOM ảo mới và DOM ảo cũ.
* Bước trên giúp xác định thành phần nào cần được render lại trên DOM thực.
* Cuối cùng, các thay đổi được thực hiện trên DOM thực tế/thực.

Reconciliation được tiến hành bởi Reconciler và Renderer

[Reconciler: Là một thuật toán được sử dụng bởi React để hiệu quả hơn trong việc cập nhật DOM. Nó chia nhỏ công việc cập nhật thành các phần nhỏ hơn, cho phép React thực hiện các công việc khác trong khi chờ đợi các tác vụ I/O hoàn thành.

Fiber tree: Là một cấu trúc dữ liệu được sử dụng bởi Reconciler để quản lý các công việc cập nhật. Nó cho phép React ưu tiên các công việc quan trọng và tạm dừng hoặc tiếp tục các công việc khác tùy thuộc vào khả năng xử lý của trình duyệt.]

**The Reconciler : Fiber tree**

Cây Fiber là một cây nội bộ có một "Fiber" cho mỗi trường hợp thành phần và phần tử DOM.

React Fiber là một bản viết lại hoàn toàn tương thích ngược của reconciler cũ. Thuật toán điều hòa mới này của React được gọi là Fiber Reconciler.

Mục tiêu chính của Fiber reconciler là render tăng dần, render tốt hơn hoặc mượt mà hơn các hiệu ứng chuyển động và cử chỉ của UI, và khả năng đáp ứng các tương tác của người dùng.

Trong việc triển khai reconciler ngăn xếp cũ, duyệt đệ quy và gọi phương thức render của toàn bộ cây được cập nhật xảy ra trong một luồng duy nhất. Điều này có thể dẫn đến việc bỏ khung hình.

Chúng ta gọi nó là một cây, React Fiber tạo ra một danh sách liên kết các node, trong đó mỗi node là một fiber. Và có mối quan hệ giữa cha, con và anh chị em. React sử dụng một khóa trả về để trỏ đến node cha, nơi bất kỳ fiber con nào cũng nên trả về sau khi hoàn thành công việc.

Các fiber không được tạo lại trong mỗi lần Render, nó chỉ đơn giản là đột biến qua lại trong các bước điều hòa trong tương lai.

**Diffing Algorithm**

**[Thuật toán Diffing là một khái niệm cốt lõi trong React, giúp tối ưu hóa việc cập nhật DOM. Đây là quá trình so sánh hai cây DOM ảo để xác định tập hợp nhỏ nhất các thay đổi cần thiết để cập nhật DOM thực tế.]**

React sử dụng thuật toán diffing như một phần của quá trình reconciliation để hiệu quả cập nhật DOM. Khi trạng thái của một thành phần thay đổi, React tạo ra một DOM ảo mới và so sánh nó với DOM hiện tại. Quá trình so sánh này, được gọi là "diffing", cho phép React xác định số lượng tối thiểu các thao tác cần thiết để cập nhật DOM.

Khi diffing hai cây DOM ảo, React đầu tiên so sánh các phần tử gốc. Nếu chúng là các loại khác nhau, cây cũ bị loại bỏ và cây mới được tạo ra. Nếu chúng là cùng một loại, React đệ quy kiểm tra các thuộc tính và phần tử con. Bất kỳ thành phần nào có cùng tham chiếu được giữ nguyên để duy trì trạng thái. Cuối cùng, React cập nhật DOM thực với các thay đổi cần thiết. Điều này giảm thiểu các cập nhật đối với DOM thực, điều quan trọng đối với hiệu suất.

**3. COMMIT PHASE AND BROWSER PAINT**

Trong Pha Render, chúng ta nhận được danh sách các cập nhật DOM, và trong pha commit, nó cập nhật trên DOM. Việc thao tác DOM thực tế của DOM thực xảy ra trong giai đoạn commit.

A diagram of a process

Description automatically generated

Thư viện React không giao tiếp trực tiếp với DOM thực. Nó sử dụng các gói bên thứ ba được gọi là Renderers như React DOM (cho nền tảng web) và React Native (cho nền tảng di động) để xử lý thao tác DOM thực tế. React cung cấp cho chúng ta phương tiện biểu đạt để chúng ta có thể định nghĩa các thành phần, v.v.

Trong các ứng dụng web React, chúng ta thường chỉ nhập gói React DOM một lần, hầu hết là trong tệp index.js, sau đó gọi phương thức render của nó. React giao tiếp với renderer bằng cách sử dụng hooks hoặc một số phương thức vòng đời thành phần khác.

* React viết vào DOM: chèn, xóa và cập nhật.
* committing là đồng bộ: DOM được cập nhật cùng một lúc, nó không thể bị gián đoạn. điều này là cần thiết để DOM không bao giờ hiển thị kết quả một phần, đảm bảo một giao diện người dùng nhất quán (đồng bộ với trạng thái mọi lúc)
* Sau khi giai đoạn commit hoàn thành, cây sợi đang hoạt động trở thành cây hiện tại cho chu kỳ render tiếp theo.

Sau đó, trình duyệt vẽ các thay đổi lên màn hình.