**I. REACT JS LÀ GÌ?**

**Lịch sử ra đời?**

-Trước thời điểm năm 2013 việc xây dựng UI cho Facebook hoàn toàn dựa trên framework AngularJS của Google, tuy nhiên sau thời điểm năm 2013 khi lượng người dùng trở nên quá lớn và dữ liệu Facebook lưu trữ ngày càng nhiều, kéo theo đó việc xử lý dữ liệu ở ngoài UI trở nên quá khó khăn vì AngularJS chậm và khá nặng nề. Facebook không thể tìm kiếm được 1 framework nào khác đủ khả năng thay thế cho AngularJS, vậy nên các lập trình viên của Facebook đã tự tạo ra 1 công nghệ mới với tên gọi React và nó bao gồm 2 phần:

-ReactJS (dùng cho web app) và React Native (dùng cho mobile app).

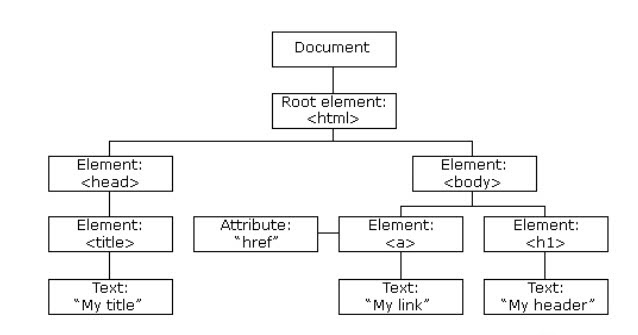
-Đến năm 2015 thì Facebook quyết định open source mã nguồn của React cho cộng đồng và đến nay cộng đồng dev trên toàn thế giới đã góp 1 phần công sức không nhỏ giúp cho React ngày 1 hoàn thiện hơn.

**ReactJS là gì?**

-ReactJS là 1 library javascript (hiện tại nó chưa đủ khả năng để trở thành 1 framework nhưng mình nghĩ ngày đó không còn xa), ReactJS sử dụng để xây dựng UI cho web theo đúng xu hướng Single Page Applications.

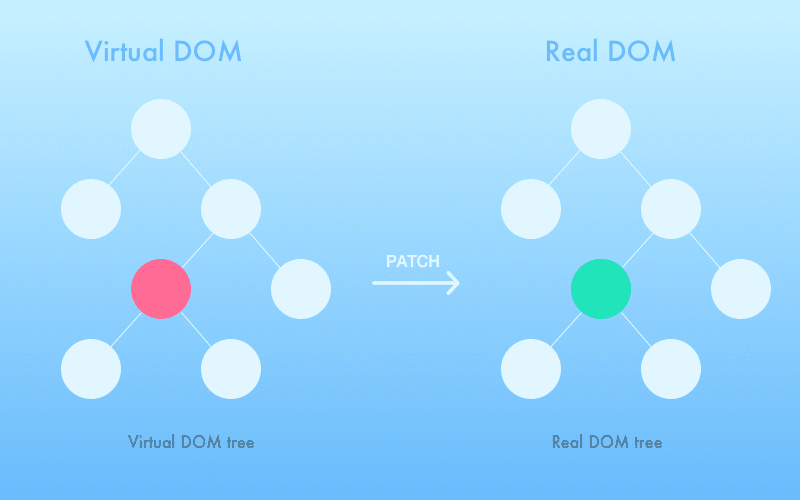
**Virtual DOM (DOM ảo)**

DOM (Document Object Model)

**

-Như thấy thì node parent đầu tiên sẽ là thẻ document, tiếp là thẻ html, rồi đến các node child head và body trong đó lại có nhiều node child khác, vậy khi có 1 sự kiện yêu cầu tác động đến một thành phần trong trang web, thì phải tác động vào node document, tức là trang web sẽ chạy lại từ đầu để hiển thị dữ liệu của phần yêu cầu thay đổi, mặc dù các thành phần khác không có thay đổi gì, điều này dẫn đến việc nếu trang web phức tạp ra sẽ khiến hiệu năng của web giảm đi rất nhiều, việc web chạy chậm là dễ hiểu.

-Vậy ReactJS đã xử lý bài toán này như thế nào?

**

-ReactJS sẽ sử dụng một DOM ảo và nó là một Object Javascript chứa đầy đủ thông tin của trang web giống như DOM thật, khi có sự tác động đến 1 thành phần của trang web thì ReactJS sẽ so sánh DOM ảo với DOM thật để biết được node nào sẽ thay đổi và tạo ra 1 *‘bản vá’* nhằm thay đổi dữ liệu ở node đó trên DOM thật. Với ReactJS khi có sự thay đổi ở 1 thành phần, thì trang web sẽ không load lại từ đầu mà chỉ thay đổi ở thành phần đã yêu cầu, dẫn đến trang web sẽ hoạt động nhanh hơn và phía server cũng đỡ mất công phải query lại các dữ liệu của những thành phần khác.

**JSX**

JSX là viết tắt của Javascript XML, có thể hiểu nó là 1 dạng ngôn ngữ cho phép viết các mã html trong javascript. Việc sử dụng JSX khi xây dựng app với ReactJS không bắt buộc, hoàn toàn có thể viết mã javascript như bình thường, tuy nhiên nên dung JSX với những lý do này.

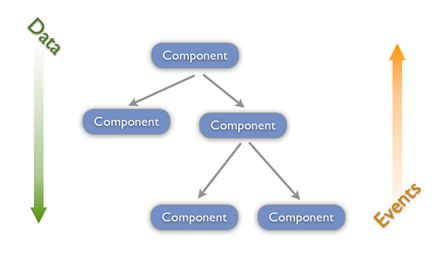
1. Cách viết của JSX khá giống với cách viết mã HTML, việc khai báo các attributes cũng rất đơn giản, khiến ta khi nhìn vào có thể dễ dàng hiểu được đoạn mã JSX thể hiện cho cái gì.

2. Bản chất của JSX vẫn là javascript vậy nên nó không thay đổi ý nghĩa của 1 mã javascript.

3. Vì cách viết JSX khá giống với cách viết mã HTML nên với các developer nó sẽ dễ chịu hơn.

**One way data binding (luồng dữ liệu 1 chiều)**

Dữ liệu trong ReactJS hoàn toàn đi theo 1 chiều và được truyền từ components cha xuống components con thông qua props.

**

Điều này dẫn tới việc sau này trang web sẽ hoạt động 1

**Component based**

-ReactJS được xây dựng xung quanh 1 khái niệm là component.

Cho ví dụ 1 giao diện web bán hàng bạn đang xây dựng sẽ do nhiều thành phần tạo nên:

* Phần menu loại sản phẩm
* Phần menu sản phẩm
* Phần thông tin liên hệ
* Phần tìm kiếm

….

-Những thành phần đó trong ReactJS được gọi chung là các component, nó cũng giống như các file template html, chỉ khác biệt ở chỗ các component là file chứa các mã JSX dùng để cấu thành nên giao diện cho trang web. Việc chia nhỏ giao diện web ra thành từng component sẽ dễ dàng quản lý dữ liệu cho trang web và tái sử dụng lại các component.

Trong React, có hai loại component có thể sử dụng:

* **Functional components**
* và **Class components**

-Một **Functional component** là một hàm JavaScript đơn giản, ví dụ:

function Hello() {

  return <h2>Hello React!</h2>;

}

-Đoạn code trên định nghĩa một component gọi là Hello và component này trả về một phần tử react đơn giản.

-Loại compoents thứ hai có thể sử dụng là **Class components**  
**-Class components** thường được sử dụng khi cần tạo thành phần có tương tác người dùng cao hơn như Forms và Animation...  
-Tất cả các **Class components** cần phải mở rộng lớp **React.Component**  
-Ta có thể viết lại Functional components Hello của dưới dạng một Class components như sau:

// Class Components

class Hello extends React.Component {

  render() {

    return <h2>Hello React!</h2>;

  }

}

-Class components có một phương thức render() để cho biết trang sẽ render ra nội dung gì.

-Cách khoa học và dễ dàng duy trì hay năng cấp tính năng. Với đặc điểm này thì ReactJS dễ dàng cho việc xây dựng các trang web với lượng người dùng cũng như dữ liệu lớn.

**Props trong React**

-Trong React, Functional components có thể chấp nhận các đối số, tương tự như các hàm JavaScript. Các đối số này được gọi là props và đại diện cho một đối tượng.

Ví dụ: Chúng ta có thể sử dụng props trong để nhận đối số trong Functional components Hello của chúng ta như sau:

// Function Components

function Hello(props) {

  return <h2>Hello {props.hi}!</h2>

}

Sau đó, tạo phần tử với JSX và truyền giá trị cho name.

<Hello hi = "hello" />;

**State trong React**

-Nhiều ứng dụng web cần các component để thay đổi dữ liệu, chẳng hạn như sau khi người dùng tương tác (nhấp vào nút, gửi biểu mẫu, v.v.).

-Tuy nhiên, với props thì không thể thay đổi nó.

-Để cho phép các component quản lý và thay đổi dữ liệu của chúng, React cung cấp một tính năng gọi là state (trạng thái)

-State là một đối tượng được thêm vào như một thuộc tính trong các compnent của Class.

// Class Components

class Hello extends React.Component {

  // state

  state = {

    name: "Nguyễn văn A"

  }

  render() {

    return <h1>Xin chào {this.state.name}.</h1>;

  }

}

-Như đã nói, có thể sử dụng state để mong muốn thay đổi dữ liệu.

-Tuy nhiên, không nên để state có thể sửa đổi trực tiếp.

-Mà thay vào đó, React cung cấp một phương thức có tên là **setState(),** có thể được sử dụng để sửa đổi state.

// Phương thức thay đổi state

this.setState({

  name: "Nguyễn văn B",

  age: 18

});

- Cần truyền một đối tượng có cặp key:value mới vào trong phương thức setState()

- Nhưng vì sao nên sử dụng setState(), thay vì chỉ thay đổi trực tiếp các giá trị của các thuộc tính đối tượng?

- Vì khi setState() được gọi, React sẽ tự động hiển thị lại component bị ảnh hưởng với state mới!.

**Tóm lại, qua hai phần props và state bạn cần nắm được:**

* Có thể sử dụng props để truyền dữ liệu cho các component.
* Các component sử dụng state để quản lý dữ liệu của chúng.
* Props ở chế độ chỉ đọc và không thể sửa đổi.
* State thì có thể được sửa đổi bởi component của nó bằng phương thức setState().
* Phương thức setState() giúp ta thay đổi sate và kích hoạt hành động render lại nội dung đã thay đổi.

**Lifecycle Methods**

Mỗi lifecycle component được phân loại thành bốn giai đoạn:

* initialization
* mounting
* updating
* unmounting

**Initialization**

-Đây là giai đoạn mà component sẽ bắt đầu hành trình của nó bằng cách thiết lập state và props. Điều này thường được thực hiệp bên trong constructor method.

**Mounting**

-Mounting là giai đoạn mà React component của ta sẽ được mount on the DOM.

-Giai đoạn này được thực hiện sau khi giai đoạn initialization được hoàn thành. Trong giai đoạn này, component sẽ được khởi động lần đầu tiên.

-Các method trong giai đoạn này:

**1. ComponentWillMount()**

Method này sẽ được gọi ngay trước khi component được mount on the DOM hoặc render method được gọi.

**2. ComponentDidMount()**

Method này sẽ được gọi sau khi component được mount on the DOM. Giống như componentWillMount, nó sẽ được gọi một lần trong một lifecycle. Trước khi thực thi method này, thì render method sẽ được gọi.

**Updating**

-Đây là giai đoạn thứ ba trong mỗi lifecycle. Sau giai đoạn mounting (nơi mà các component được tạo ra), đây là giai đoạn updating bắt đầu thực hiện. Đây là nơi state của component thay đổi, và việc re-rendering được diễn ra.

-Trong giai đoạn này, dữ liệu của mỗi component(state & props) sẽ được update khi user thực hiện các hành động như click, typing… Kết quả sẽ re-rendering lại component. Các method có trong giai đoạn này:

**1. ShouldComponentUpdate()**

-Method này sẽ xác định rằng component có được update hay là không. Mặc định, sẽ trả về là true. Nhưng đến một lúc nào đó, nếu ta muốn re-render lại component bằng một số ràng buộc điều kiện, thì viết các điều kiện đó trong method này.

**2.ComponentWillUpdate()**

- Nó sẽ được gọi trước khi component update và sẽ được gọi mỗi lần sau method shouldComponentUpdate.

**3. ComponentDidUpdate()**

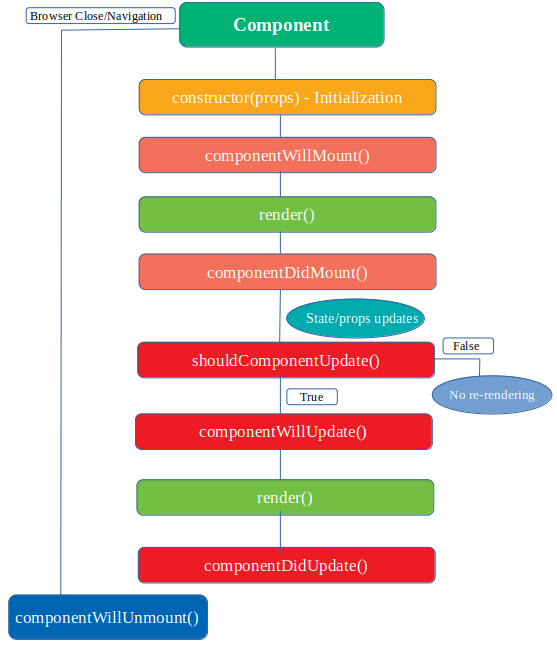
-Method này sẽ chỉ được gọi sau khi re-render component. Sau mỗi lần update mới, component sẽ được update vào Dom. Method componentDidUpdate sẽ được thực hiện. Method này sẽ nhận vào các đối số như prevProps và prevState.

**Unmounting**

Đây là giai đoạn cuối cùng của một component lifecycle. Như cái tên của nó sẽ unmounted DOM trong giai đoạn này. method có trong giai đoạn này:

**ComponentWillUnmount()**

Method này sẽ được gọi trước khi unmmount component. Trước khi loại bỏ thành phần khỏi DOM, componentWillUnmount sẽ được thực thi. Method này sẽ biểu thị sự kết thúc của một lifecycle.



### **useEffect()**

Với useEffect() hooks này cho phép thay thế componentDidMount, componentDidUpdate tùy theo từng trường hợp.

import React, { useState, useEffect } from 'react';

function Example() {

const [count, setCount] = useState(0);

useEffect(() => {

document.title = `You clicked ${count} times`;

}, [count]);

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

}

-Với useEffect() ở trên chúng ta sẽ truyền vào 2 tham số, đầu tiên chính là function thực hiện effect, và khi truyền vào tham số thứ 2 chính là tham số mà useEffect() phụ thuộc vào.

-Tức là mỗi khi có thay đổi ở count thì function này sẽ được thực thi và re-render lại Component. Như vậy nó giống với componentDidUpdate ở trong ClassComponent.

useEffect(() => {

document.title = `You clicked ${count} times`;

}, []);

-Còn khi không truyền vào tham số thứ 2 như thế này thì việc render sẽ không xảy ra khi có thay đổi vì tham số là mảng rỗng.

-Như vậy nó giống với việc sử dụng componentDidMount.