REACTJS

1) Tìm hiểu thư viện reactjs, lợi thế, lý do sử dụng, v.v... (1/2 ngày )

2) Tìm hiểu các thuộc tính đặc thù của reactjs: jsx, component, prop, state, react hook, .. (1.5 ngày)

3) Tìm hiểu life cycle của 1 component trong reactjs (1/2 ngày)

4) Tìm hiểu mô hình redux: lý do sử dụng, các thành phần, vai trò cụ thể từng thành phần, tương tác giữa các thành phần với nhau (1.5 ngày)

**Bài Làm**

1) Thư viện ReactJS

**Khái Niệm**:

+ Reactjs là một thư viện Javascript mã nguồn mở

+ Hỗ trợ xây dựng giao diện người dùng

**Ưu điểm**:

+ React.js chia nhỏ các phần của trang thành từng phần riêng biệt để xử lý, gọi là component, component giống như một hàm (function) Javascript giúp dễ quản lý, dễ sử dụng lại ở nhiều nơi.

+ Mỗi khi dữ liệu được cập nhật mới, thay vì thay đổi nguyên trang, thì React.js sẽ giúp thay đổi chỉ component liên quan, việc này sẽ tối ưu rất nhiều thời gian làm mới dữ liệu.

+ Đặc biệt là hầu hết các component thường dùng đã được phát triển và chia sẻ, chỉ cần cài đặt và sử dụng

+ cho phép nhúng HTML vào javascript thông qua JSX (thay vì nhúng javascript vào code HTML thông qua attribute như AngularJS)

+ Một trong những đặc trưng duy nhất của Reactjs là việc render dữ liệu không những có thể thực hiện ở tầng server mà còn ở tầng client.

**Nhược điểm:**

Reactjs chỉ phục vụ cho tầng View. React chỉ là View Library nó không phải là một MVC framework như những framework khác. Đây chỉ là thư viện của Facebook giúp render ra phần view. Vì thế React sẽ không có phần Model và Controller, mà phải kết hợp với các thư viện khác. React cũng sẽ không có 2-way binding hay là Ajax

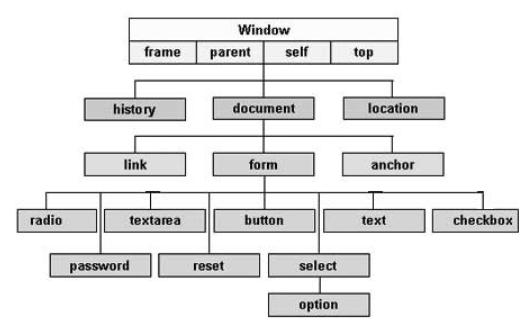
Tích hợp Reactjs vào các framework MVC truyền thống yêu cầu cần phải cấu hình lại.

React khá nặng nếu so với các framework khác React có kích thước tương tương với Angular. Trong khi đó Angular là một framework hoàn chỉnh

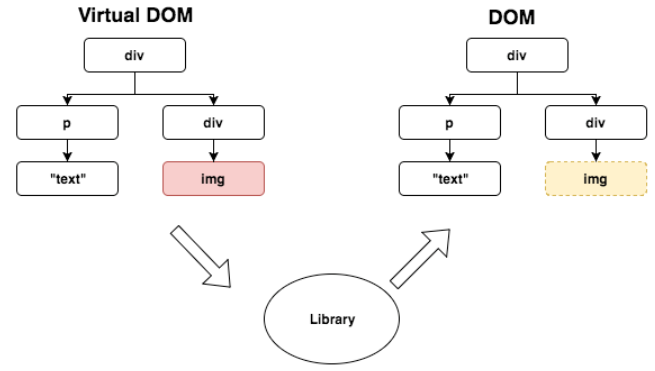
Khó tiếp cận cho người mới học Web

**Lý Do Sử Dụng React**

- DOM ảo: DOM (Document Object Model) là nơi **lưu trữ** tất cả các **đối tượng trên trang web** theo cấu trúc giống như cây.



Hầu hết các thư viện và frameworks JavaScript sử dụng cái được gọi là DOM thực. vấn đề với DOM thực là toàn bộ nội dung phải cập nhật mỗi khi có một thay đổi trong trạng thái UI điều này làm mọi thứ chậm lại một chút và có thể khá tốn kém, đây là lúc DOM ảo xuất hiện. Khi DOM ảo cập nhật, nó so sánh phiên bản cập nhật với phiên bản trước của DOM ảo. React sau đó so sánh cả hai phiên bản của DOM ảo để xem những gì đã thay đổi thông qua một quá trình được gọi là khác biệt và chỉ áp dụng những thay đổi đó cho DOM thực.



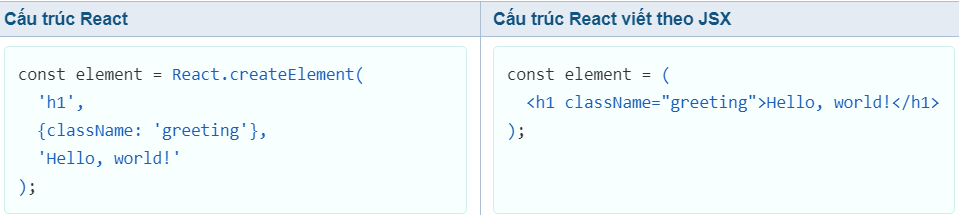
Bằng cách này, thay vì đi qua toàn bộ cây và cập nhật từng đối tượng có trong nó, chúng ta chỉ phải cập nhật các phần thay đổi.

- JSX:

+ JSX là một cú pháp được sử dụng trong React. Dùng để viết HTML trong JavaScript

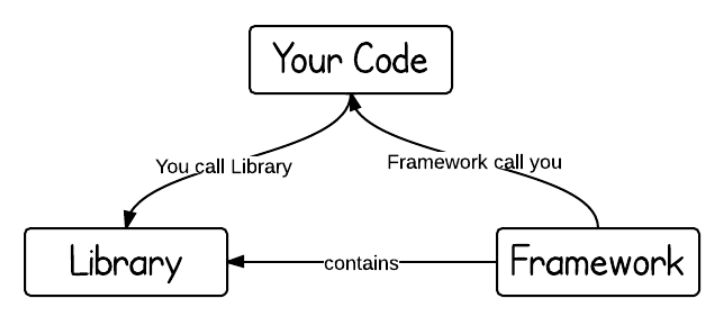
+ JSX kết hợp các yếu tố của cả Javascript và XML (HTML cũng được viết theo cấu trúc XML)

+ JSX cũng có thể được sử dụng trong các function, các câu lệnh, vòng lặp, biến và bất cứ nơi nào bạn có thể sử dụng JavaScript



3) React is a Library (React là một thư viện):

React được gọi là JavaScript framework, nhưng thực tế nó là một thư viện chứ không phải framework. sự khác biệt giữa framework và thư viện là cấu trúc framework đã được xác định trước khi bạn bắt đầu trong khi thư viện thì không. Điều này có nghĩa là một thư viện cung cấp cho bạn sự linh hoạt và tùy biến cao hơn nhiều so với một framework. một thư viện về cơ bản cho phép bạn kiểm soát dòng chảy của ứng dụng. Khi bạn sử dụng thư viện, bạn là người gọi thư viện nhưng khi bạn sử dụng framework, nó sẽ gọi cho bạn. Điều này được gọi là đảo ngược của kiểm soát.

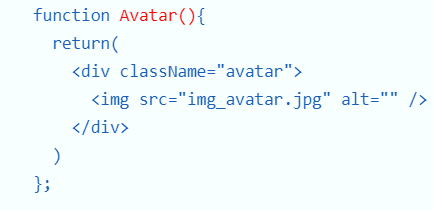


4) Một ý tưởng khác làm cho ReactJS trở nên độc đáo là nó cho phép Developer phá vỡ những cấu tạo UI phức tạp thành những component độc lập. Dev sẽ không phải lo lắng về tổng thể ứng dụng web, giờ đây Developer dễ dàng chia nhỏ các cấu trúc UI/UX phức tạp thành từng component đơn giản hơn.

5) Không chỉ vậy, thông qua Props, các component có thể kết nối với nhau bằng cách sử dụng data tùy biến. Điều này giúp chuyển các dữ liệu đã được tùy biến đến một UI component cụ thể.

6) Tính năng State khi sử dụng ReactJS, giúp điều chỉnh trạng thái cho các component (child) nhận data từ một component nguồn (parent). Developer có thể thay đổi trạng thái cho nhiều component (child) trên ứng dụng nhưng không ảnh hưởng tới component gốc (parent) đang ở trạng thái Stateful. Việc điều chỉnh các ứng dụng cũng trở nên dễ dàng hơn.

**Component**: là các thành phần nhỏ riêng biệt được tách ra từ một trang lớn vd: header, footer, sidebar, navigation, itemList,… Nó giống như hàm javascript dễ quản lý, tái sử dụng. Component thì được viết dưới dạng function hoặc Class. Dạng function luôn được return. Bên trong return sẽ có ( ) => để bao tất cả nội dung return. Ta thường sử dụng React.DOM để render một component. Function Render phải có cấu trúc XML <tag/> hoặc <tag></tag>



**Props và state:**

Props là các biến, data được truyền từ component cha và có thể truy cập được ở cả các component con.

State được khởi tạo và chỉnh sửa và chỉ có thể truy bởi chính bản thân component chứa nó.

**Props:** phương thức để truyền giá trị vào các component. Chúng được React xử lý và trả về kết quả trên trình duyệt.

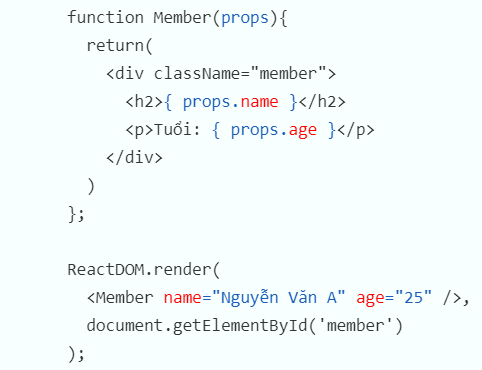
VD: thay vì truyền tên với tuổi cố định vào function member như vậy





Ta sẽ sử dụng Prop để truyền vào như vậy

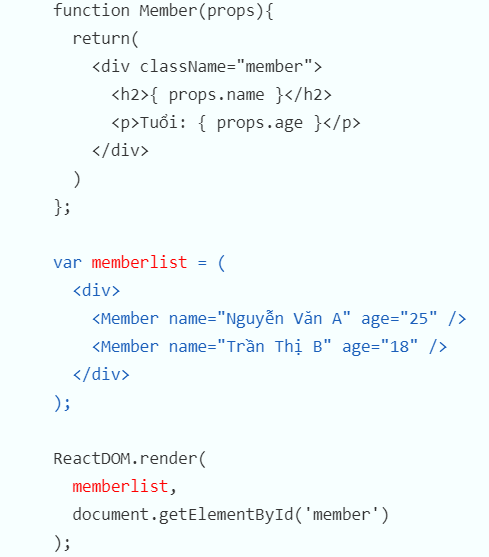




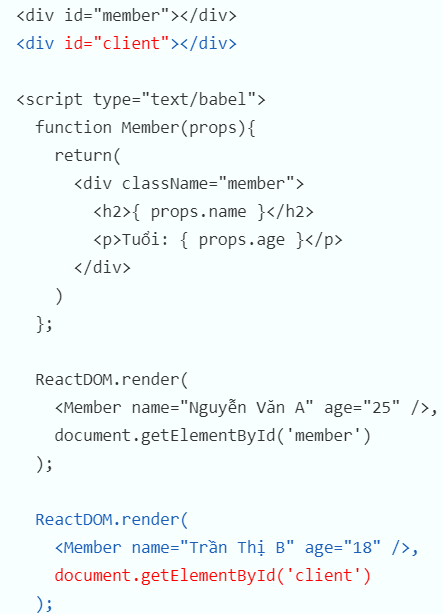
**Ý nghĩa**: Ở component Member ta đã sử dụng props, giống như cách báo cho hệ thống React biết nơi nào xuất hiện props thì nơi đó sẽ được liên kết với các biến (name, age), khi render thì các tham số (Nguyễn Văn A, 25) sẽ được truyền vào các biến này.

VD: nếu ta muốn render thêm một đối tượng member nữa. Thì ta làm như sao





**Tái sử dụng component:** ở VD trên ta sử dụng component Member render 1 nơi có id là member. Tuy nhiên ta có thể render các component ra nhiều nơi khác nhau



Thay vì viết dữ liệu trực tiếp bên trong phần render (như hình trên), ta sẽ tách riêng phần dữ liệu ra một dữ liệu của một đối tượng (data Object).

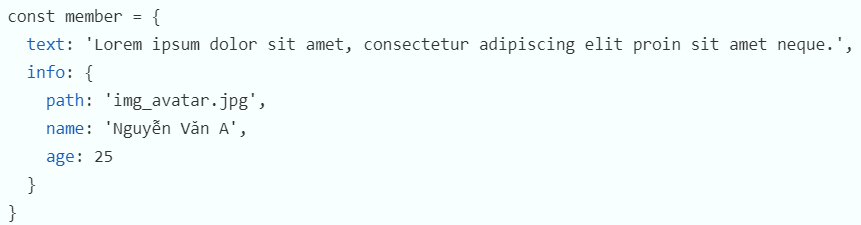


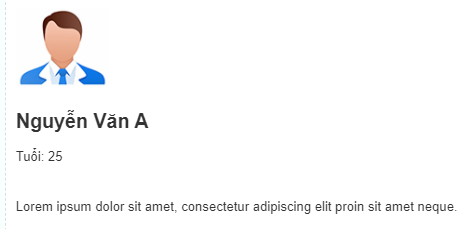
const info khai báo giá trị cho một object.

info.name lấy giá trị name của object.

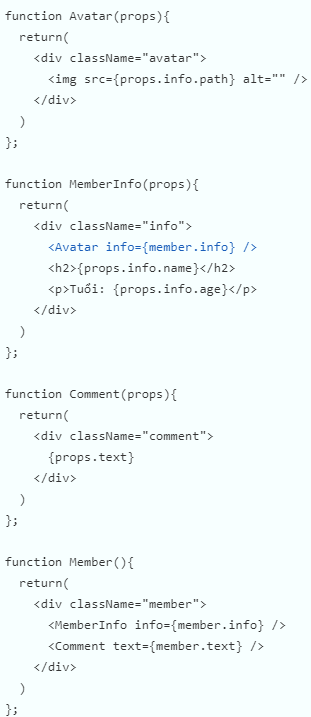
info.age lấy giá trị age của object.

**Xử lý dữ liệu khi các component lồng vào nhau (tái sử dụng component)**



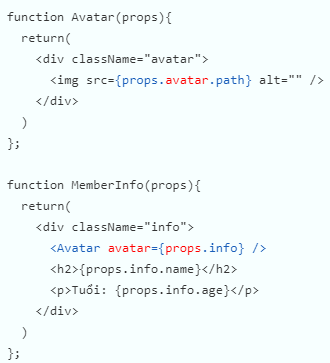


Để hiểu rõ hơn việc truyền dữ liệu giữa các component, ta tiến hành lồng function Avatar(props) vào bên trong MemberInfo(props), ta làm như sau (nhớ xóa bỏ function Avatar(props) bên trong Member()):



Việc di chuyển <Avatar info={member.info} /> từ function Member() vào bên trong MemberInfo(props). Khi này kết quả vẫn không thay đổi.

Thay vì sử dụng member.info của function <Avatar />, ta sẽ liên kết <Avatar /> với <MemberInfor /> bằng một props.avatar, với avatar là một tên bất kỳ được sử dụng riêng bên trong function <Avatar />, ta viết như sau:

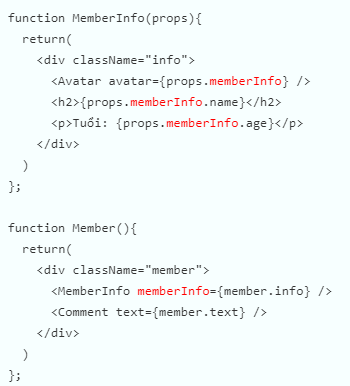


avatar ta có thể đặt tên bất kỳ, miễn sao ở function <MemberInfor /> ta gọi kết quả trả về là props.info là được. Điều này tiện lợi trong việc xử lý các giá trị props ở các component mà không cần quan tâm tới các giá trị props ở các component khác.

Code hoàn chỉnh khi này ta có:



Ta cũng có thể làm tương tự cho component MemberInfo(props) và Member(), ta tiếp tục viết lại code như sau:



Với memberInfo là tên bất kỳ bên trong component MemberInfo(props).

**Xử lý dữ liệu tại phần render**

Hiện tại giá trị dữ liệu object const member được gọi trực tiếp bên trong Member(), ta có thể gọi giá trị này tại phần render, mục đích để tách biệt rõ các component chỉ nên xử lý các thao tác logic, phần dữ liệu nên đặt tại render.

Ta viết lại component Member() và phần render như sau:



=> Kết luận giá trị của Props là không thể thay đổi, chỉ truyển từ component này sang component khác. Để thay đổi dữ liệu trong quá trình truyền giửa các component ta sẽ dùng **State.**

**Tìm hiểu Props – arrow function**

+ ES6 arrow function, viết hàm dạng mũi tên =>

Ưu điểm: ngắn gọn, dễ hiểu

Function khác Arrow Function

Arrow Function:

Arrow function là một tính năng mới của ES6, giúp viết code ngắn gọn hơn.

Arrow function sử dụng trong các TH dùng: map, filter, reducer,...

Arrow function không có bind (bind dùng để định nghĩa lại this). this sẽ tương ứng với ngữ cảnh gần nhất của nó.

Arrow function không phù hợp là method của object.

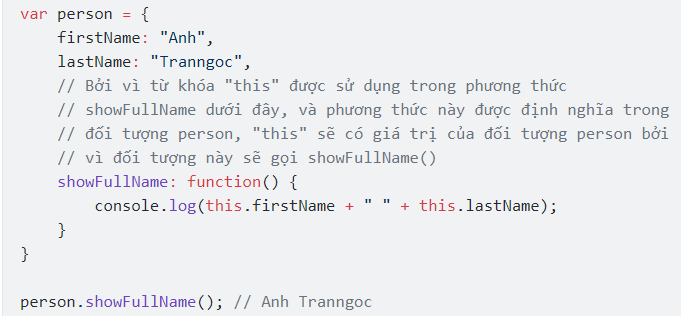
Function:

Cần chú ý thêm về con trỏ được trỏ tới để định nghĩa lại this bằng 'bind'.

This là gì?

This là một từ khoá khá quen thuộc trong khá nhiều ngôn ngữ, nó dùng để trỏ tới chính object gọi hàm đó và javasrcipt cũng thế.

VD



**So Sánh Cấu Trúc**

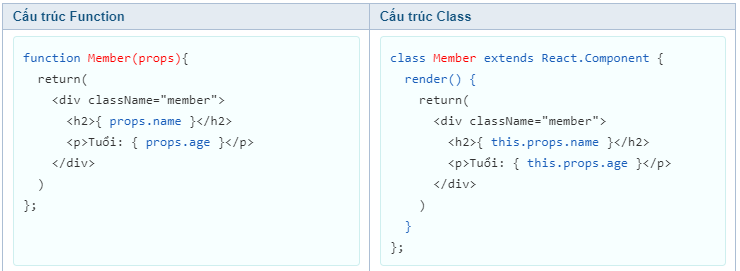
|  |  |
| --- | --- |
| Hàm Function thường | Arrow Function |
| function hello(){  console.log('hello')} | hello = () => {  console.log('hello')  } |



**Tìm hiểu về class component và function component**

|  |  |
| --- | --- |
| Functional (stateless) | Class (statefull) |
| + Stateless component là các component chỉ chứa props, các component loại này chỉ dùng để render() thì sẽ hiệu quả hơn.  + một function Javascript / ES6 function  + phải trả về 1 React element  + nhận props làm tham số nếu cần | + Stateful Component là các component chứa cả props và state, các component này được dùng xử lý data, phản hỏi yêu cầu người dùng, phù hợp cho mô hình client server…  + là một class ES6, nó sẽ là một component khi nó "kế thừa" React component.  + có thể nhận props (trong hàm khởi tạo) nếu cần.  + có thể maintain data của nó với state  + phải có 1 method render() trả về 1 React element (JSX), or null |

So sánh cấu trúc giữa function và Class ta được như sau:



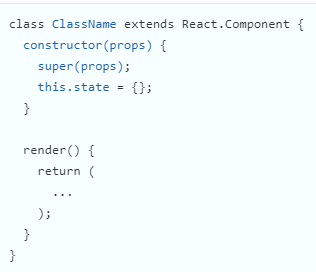
Nhìn vào so sánh cấu trúc bên trên, ta thấy dễ dàng chuyển đổi một function sang Class:

**function Member(props)** thành **class Member extends React.Component**.

**Thêm render() {}.**

Ở mỗi props thêm this.

**Cấu trúc chung của một component sử dụng với State được viết dưới dạng Classes (hoặc arrow function) như bên dưới.**



*ClassName* tên Class.

*extends React.Component* phương thức kế thừa component (thực ra là kế thừa class đã tồn tại).

*constructor(props)* hàm khởi tạo đối tượng cho một class, mỗi class chỉ chứa một hàm khởi tạo duy nhất.

*super(props)* gọi lại constructor trong React.Component, khi này ta mới có thể sử dụng phương thức this được.

*this.state* phải là một Object.

*render* giống như cách dùng của props.

**Tìm hiểu về state**

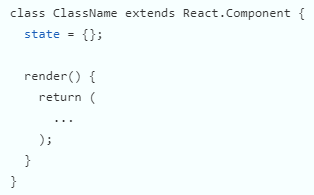


chỉ là khai báo State với constructor, gán giá trị ban đầu cho State là this.state, sau đó render nội dung đã được khai báo.

Ví dụ trên là cách làm việc đơn giản nhất của State.

**Cấu trúc đơn giản của State**

Ta có thể viết một state đơn giản như sau:





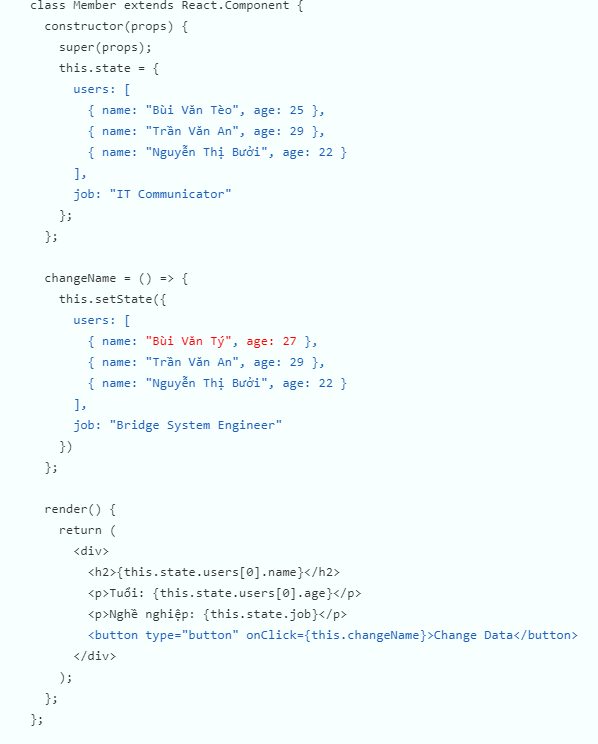
+ Ta thấy nội dung trên rất đơn giản để khai báo một State.

Vậy tại sao không sử dụng State đơn giản như trên cho gọn, cần gì phải dùng constructor cho rắc rối? câu trả lời là constructor là phương thức (method) luôn luôn được gọi đầu tiên mỗi khi khởi tạo một Classes, nó đảm bảo mọi thứ đã sẵn sàng trước khi bắt đầu một Classes. Tuy nhiên, tùy vào cách sử dụng mà bạn có thể dùng constructor hay không nhé.

**Thay đổi dữ liệu bên trong component với State**

Muốn thay đổi hay cập nhật nội dung của một State, chúng ta sử dụng this.setState,





**Component Lifecycle là gì?**

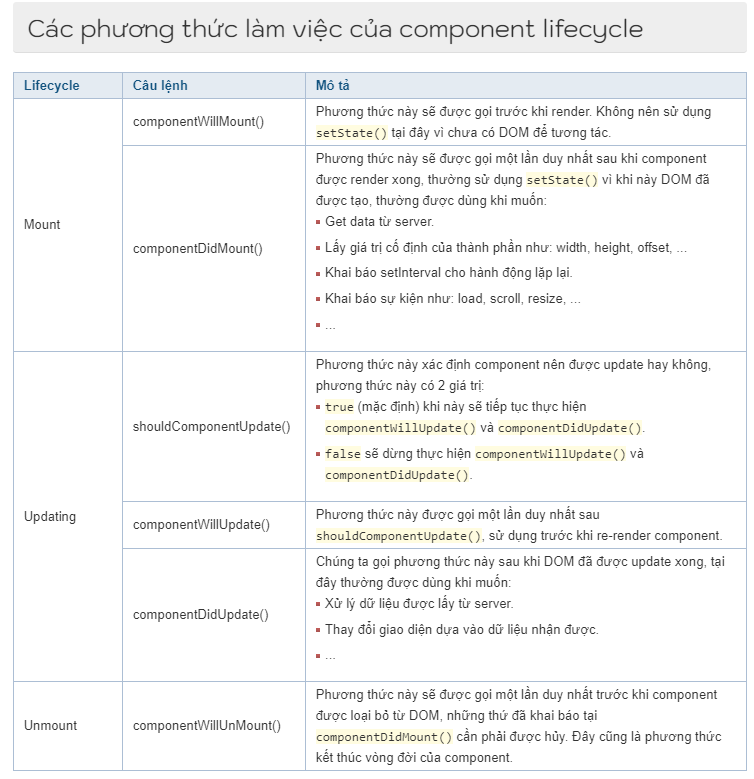
Component có thể được viết theo kiểu của vòng đời này như sau:

Tạo vòng đời (Mount trên DOM)

Cập nhật vòng đời (Updating)

Kết thúc vòng đời (Unmount từ DOM)

Quá trình trên được gọi là Component lifecycle (vòng đời của một component).



**Xem ví dụ về component lifecycle**

Ta tạo một vòng đời (lifecycle) cho một bộ đếm (counter) tăng từ 0 cho đến 3, khi đạt đến giá trị 3 thì sẽ xóa bộ đếm để kết thúc vòng đời, ta lần lượt thực hiện các bước sau:

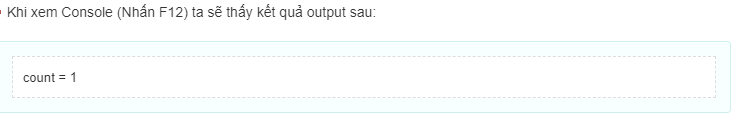
Khai báo ban đầu, render giá trị đầu tiên.

Cho giá trị bắt đầu tăng sau mỗi 1 giây (cập nhật State).

Loại bỏ component ra khỏi DOM khi giá trị đạt đến 3 (Xóa component để xem kết thúc vòng đời).

Ta xét file index.html sau:





**Cho giá trị bắt đầu tăng sau mỗi 1 giây**

Tạo function count để cập nhật giá trị với setState.

Cần setInterval để giá trị tự động cập nhật function count.

Do component đã được render xong, nên setInterval ta đặt bên trong componentDidMount (như đã nói ở table bên trên).



* Giá trị sẽ được tăng 1 đơn vị trên giây, Do chưa có điều kiện kết thúc nên giá trị sẽ được tăng vô hạn.

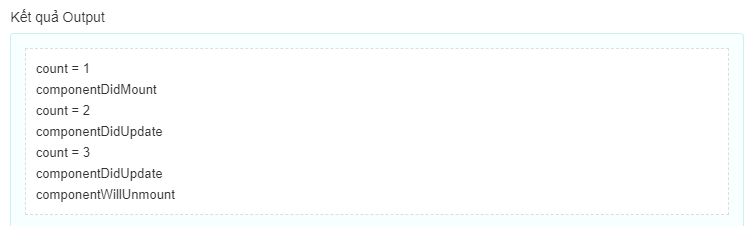
**Loại component ra khỏi DOM khi giá trị đạt đến 3**

Ta tận dụng việc gọi một lần duy nhất của phương thức componentDidUpdate để loại bỏ component ra khỏi DOM khi giá trị đạt đến 3.

Sử dụng ReactDOM.unmountComponentAtNode() để loại bỏ component ra khỏi DOM.

Sau khi Component đã được loại bỏ khỏi DOM, ta cần sử dụng phương thức componentWillUnmount() để xóa interval đã đăng ký từ componentDidMount(), nếu không sẽ báo lỗi vì vòng đời không được kết thúc.

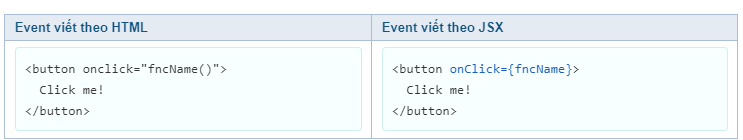




Ta thấy kết quả đã dừng lại ở 3, khi này sau khi gọi phương thức componentWillUnmount() thì vòng đời đã kết thúc.

Bên trên là một ví dụ đơn giản về cách tạo vòng đời component, trong thực tế tùy vào mục đích sử dụng mà ta có thể điều chỉnh vòng đời dừng tại đâu cho phù hợp, ví dụ nếu tạo một đồng hồ điện tử, thì ta sẽ không cho vòng đời kết thúc, khi đó sẽ không cần sử dụng componentWillUnmount().

**React.js xử lý sự kiện (event)**



**Giải Thích**

Thay đổi bên trên khác biệt ở phần onClick={} thay vì onclick="", (cẩn thận chữ C viết hoa).

Thay đổi khác là ở phần gọi function:

Khi bạn khai báo fncName() thì function sẽ được gọi khi render.

Khi bạn khai báo fncName thì function sẽ được gọi khi button được click.

**React với function Button**



2) Tìm hiểu các thuộc tính đặc thù của reactjs: jsx, component, prop, state, react hook