# **ReactJs cơ bản**

## **1, Components**

* Phân chia UI thành các thành phần nhỏ
* 2 loại component:
  + Class component: Lifecycle method
  + function component: Hooks

## **2, Props**

* Được truyền từ component cha
* Không thay đổi giá trị được
* Đa dạng

## **3, State**

* Thay đổi được
* Dùng khi chỉ được sử dụng bởi **một component** hiện tại
* VD:
  + Thông tin lấy từ API
* Lift-up state:

Lift-up (nâng lên), ở đây chúng ta hay gọi là ‘lifting state up’, một kỹ thuật hay được dùng để chia sẻ data trong ứng dụng React. Cụ thể, chúng ta sẽ đưa ‘state của component con’ sang cho ‘component cha’ quản lý.Component con sẽ dùng data thông qua ‘props truyền từ cha xuống’. Như vậy, ‘lift-up state’ ám chỉ việc đưa state của ‘component con’ lên ‘component cha’.Do React thiết kế theo mô hình cha-con, data chảy từ trên (cha) xuống dưới (con), nên gọi là lift-up (đẩy lên/nâng lên).

Chúng ta cần lift-up state khi ‘component cha có nhiều con’, và các con của nó có quan hệ với nhau. Ở đây, chúng ta muốn 1 hành động xảy ra ở ‘1 đứa con này’, sẽ cần làm gì đấy ‘ở 1 đứa con khác’, thành ra, chúng ta sẽ nhờ ‘cha của các con’ làm (thông qua việc lift-up state). Cha có thể nói chuyện với con thông qua ‘props’, còn các con không thể nói chuyện với nhau.

Ví dụ:

<Parent>

<Child1>...</Child1>

<Child2> ...</Child2>

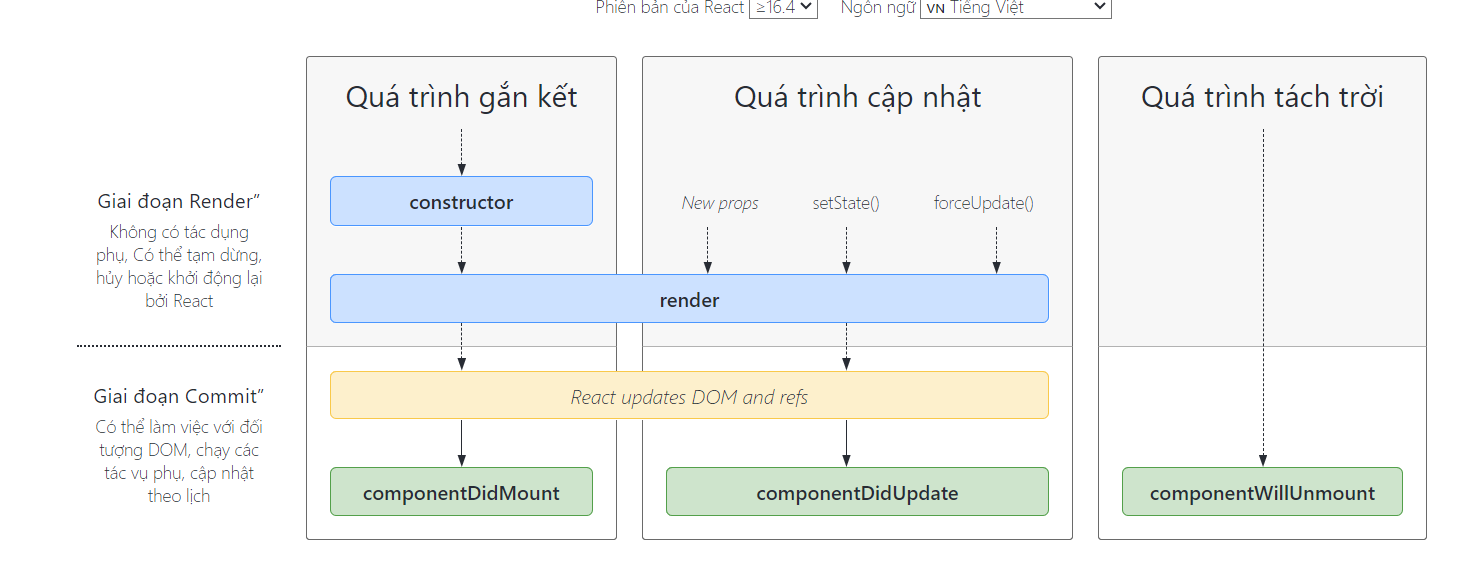
</Parent>

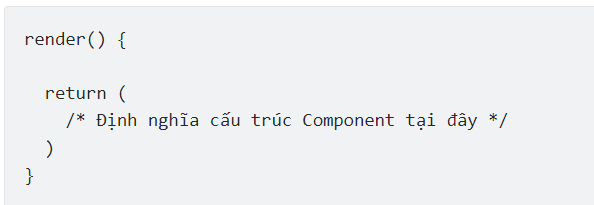
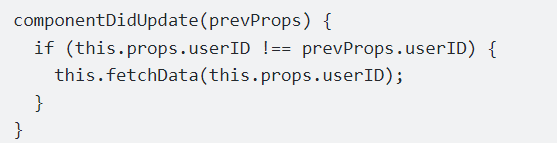
## **4, Global State (Redux)**

* Thay đổi được
* Dùng khi được dùng bởi **nhiều component** khác nhau
* VD:
  + Thông tin logged in user
  + Thông tin của giỏ hàng

## **5, Lifecycle của component (class component)**

* 3 giai đoạn:
  + Được tạo ra (Mounting) (add vào trong Dom-hiển thị ra UI)
  + Qua nhiều thay đổi (Updating): Updating là giai đoạn khi React Component cần cập nhật giao diện mỗi khi props hoặc state của nó thay đổi.
  + Bị hủy bỏ (Unmounting): Unmounting là giai đoạn khi React Component bị xoá khỏi DOM tree.
* **Số lần thực hiện**
  + Mounting, Unmounting: 1 Lần
  + Updating: nhiều lần



* **Component vs PureComponent** 
  + Nên dùng PureComponent
  + Vì có shallow comparison trong hàm shouldComponentUpdate(): so sánh giá trị trước và giá trị sau nếu bằng không gọi render nếu không bằng nhau gọi render
* **Constructor():**
  + Được phép dùng
  + Nhớ có super (props): gọi đến hàm khởi tạo cha React.Component. Nếu thiếu thì this.props sẽ là undefined.
  + Khai báo state
  + Định nghĩa properties của component
  + Bind method (xử lý event, sử dụng trong setTimeout hoặc setInterval) với this.
* **render(**)
  + Đây là phương thức duy nhất bắt buộc phải có đối với React Component và có cấu trúc như sau:
  + Phương thức này dùng để miêu tả cấu trúc của Component sau khi nó được chèn vào DOM tree. Nó bắt buộc được gọi lần đầu tiên để chèn Component vào HTML, và có thể được gọi lại để cập nhật giao diện mỗi khi state của Component thay đổi.
  + Phương thức này dùng để miêu tả cấu trúc của Component sau khi nó được chèn vào DOM tree. Nó bắt buộc được gọi lần đầu tiên để chèn Component vào HTML, và có thể được gọi lại để cập nhật giao diện mỗi khi state của Component thay đổi.
  + Đặc biệt, bạn nên để phương thức này là Pure Function – nghĩa là nó không làm thay đổi state của Component, không tương tác với trình duyệt, không lấy dữ liệu từ server,…
  + Chỉ đơn giản là nó lấy data từ this.props và this.state để xây dựng và cập nhật giao diện
* **render() & shouldComponentUpdate()**
  + Phương thức render() trong pha Updating có thể được gọi hoặc không, phụ thuộc vào phương thức shouldComponentUpdate().
* **componentDidUpdate()**
  + Phương thức này được gọi sau khi việc update kết thúc – component với những dữ liệu mới đã được cập nhật xong lên giao diện. Trong phương thức này, bạn có thể xử lý việc lấy dữ liệu từ server
  + Tức là nếu giá trị của props userID thay đổi thì bạn sẽ lấy dữ liệu từ server xuống và làm một số thứ sau đó. Ở đây, bạn cần chú ý điều kiện trong if. Nếu không có điều kiện này thì việc this.fetchData từ server xuống vẫn được thực hiện dù cho giá trị userID không thay đổi.
  + Ngoài ra, bạn cũng có thể xử lý DOM node trong phương thức này, ví dụ như: ẩn hiện 1 phần tử, thay đổi width/height của nó để phù hợp với dữ liệu mới,…
  + Hoặc thậm chí bạn cũng có thể thay đổi state của Component tại đây. Nhưng phải cẩn thận với điều này vì: khi bạn gọi this.setState thì componentDidUpdate() lại được gọi. Nếu bạn không xử lý điều kiện if else hợp lý thì rất có thể vòng lặp vô hạn sẽ xảy ra
* **componentWillUnmount()**
  + phương thức này cũng chỉ được gọi 1 lần duy nhất. Và quan trọng là những thứ bạn khởi tạo, đăng ký ở componentDidMount() thì bạn phải xoá, huỷ đăng ký trong phương thức componentWillUnmount()

6, Hooks cơ bản(function component)

a, useState

useState giúp sử dụng được State cho Function Component

Cú pháp: const [ state, setState ] = useState(initValue);

Cụ thể: hàm useState ( ) là 1 array, trả ra 2 tham số. Tham số đầu tiên chính là ‘tên của State’ và tham số thứ 2 chính là ‘hàm có thể cập nhật giá trị của state’ => dùng array destructuring để lấy ra 2 tham số này.

b, useEffect

**useEffect ( function\_xử\_lý, các\_biến\_phụ\_thuộc )**

Hàm useEffect nhận vào 2 tham số, tham số đầu tiên là function sẽ khởi tạo khi chúng ta chạy hàm useEffect, và tham số thứ 2, là các biến chúng ta muốn nhờ useEffect ‘quan sát’ sự thay đổi của nó.

1: useEffect( ( ) => {

//code logic here

},[ ]);

Khi chúng ta truyền vào mảng rỗng (tham số thứ 2 của hàm useEffect), thì hàm

useEffect sẽ chạy đúng 1 lần, sẽ có tác dụng như hàm ComponentDidMount của React Class.

2: useEffect( ( ) => {

//code logic here

}, [ tên\_biến ] );

Khi truyền vào tên biến (đối với tham số thứ 2), thì hàm useEffect sẽ được chạy ‘bất cứ khi nào giá trị của ‘biến quan sát’ bị thay đổi.

Như vậy, lúc này, hàm useEffect sẽ có tác dụng như hàm ComponentDidUpdate,

giúp chúng ta có thể so sánh giá trị quá khứ và giá trị hiện tại của biến (props/state).

Nếu giá trị của biến thay đổi, khối code logic của hàm useEffect sẽ được chạy.

c, useContext

* **useContext** (bối cảnh) giúp **đơn giản hóa việc chuyền dữ liệu** từ component cha xuống các component con mà không cần phải sử dụng đến props.
* Tức là chuyền trực tiếp từ component cha xuống các component con mà không cần phải thông qua 1 component gián tiếp.

d, useRef

* **useRef** trả về một object với thuộc tính current được khởi tạo thông qua tham số truyền vào.
* Object được trả về không bị khởi tạo lại khi component render lại.
* Giá trị trong object thay đổi nhưng component không bị render lại (useState thay đổi thì làm component render lại).
* Cú pháp: **const tenBien = useRef(initialValue);**
* Trong đó:
  + **initialValue**: là giá trị khởi tạo.
* **useRef** được sử dụng để truy cập được các phần tử trong DOM.