* Component là một block code độc lập, có thể tái sử dụng, nó chia UI thành nhiều phần nhỏ.
* React có 2 loại component: **Functional (Stateless)** và **Class (Stateful)**.

**Class (Stateful) Components**

* Các Class components là những class ES6. Chúng phức tạp hơn functional components ở chỗ nó còn có: phương thức khởi tạo, life-cycle, hàm render() và quản lý state (data)
* Một React class component là:
* 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

**Functional (Stateless) Components**

* Functional component là một hàm Javascript (hoặc ES6) trả về 1 phần tử/1 element React.
* QUAN TRỌNG: Functional components cũng được nói với một cái tên là stateless components bởi vì chúng ta không thể làm nhiều thứ phức tạp như quản lý React State (data) hoặc phương thức life-cycle trong functional components.
* Vậy 1 React Functional Component là:
* Một function Javascript / ES6 function
* Phải trả về 1 React element

Nhận props làm tham số nếu cần

**ReactJS Event**

React là một thư viện dựa trên Javascript, về cơ bản không có nhiều khác biệt trong cách xử lý sự kiện giữa ReactJS và Javascript.

Với Javascript, khi sự kiện xảy ra một hàm sẽ được gọi để thực thi.

Nhưng với React, khi sự kiện xảy ra, sẽ có một phương thức của Component được gọi.

Một số Event được sử dụng nhiều trong các dự án hiện nay:

* Mouse Events: onClick
* Keyboard Events: onKeyDown, onKeyPress, onKeyUp
* Focus Events: onFocus, onBlur
* Form Events: onChange, onSubmit
* UI Events: onScroll

**Tổng quan về ReactJs props**

**Props** là một object được truyền vào trong một components, mỗi components sẽ nhận vào props và trả về react element.

Props *cho phép chúng ta giao tiếp giữa các components với nhau bằng cách truyền tham số qua lại giữa các components*.

**Tất cả các component phải hoạt động giống như 1 pure function với các giá trị prop của chúng.** Khi một component con nhận props từ component cha thì nó chỉ có thể đọc (read only) mà không thể chỉnh sửa giá trị của props đó.

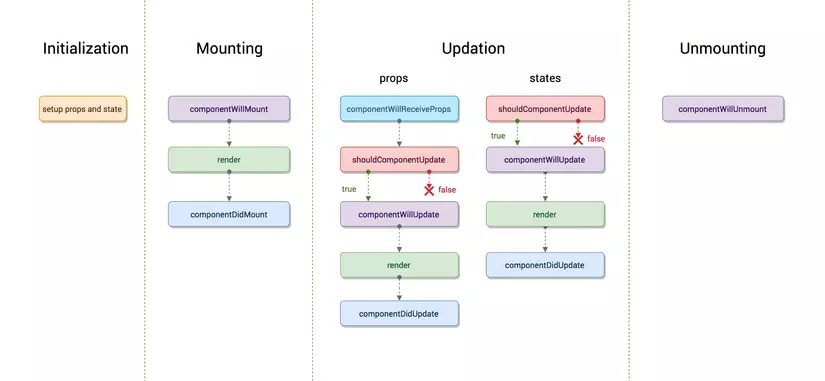
**Tổng quan về state**

**State** là một object có thể được sử dụng để chứa dữ liệu hoặc thông tin về components. State có thể được thay đổi bất cứ khi nào mong muốn. Khác với props bạn có thể truyền props sang các components khác nhau thì state chỉ tồn tại trong phạm vi của components chứa nó, mỗi khi state thay đổi thì components đó sẽ được render lại.

Trong các dự án React, state được dùng để phản hồi các yêu cầu từ người dùng, hay lưu trữ một dữ liệu nào đó trong components.

State được khai báo trong constructor và được update lại qua việc sử dụng phương thức setState của component

**ReactJs Lifecycle**



**Các phương thức làm việc của Lifecycle**

|  |  |  |
| --- | --- | --- |
| Mount | componentWillMount() | Phương thức này sẽ được gọi trước khi render. Không nên sử dụng setState() tại đây vì chưa có DOM để tương tác. |
| **componentDidMount()** | Phương thức này sẽ được gọi một lần duy nhất sau khi component được render xong, thường sử dụng setState() vì khi này DOM đã được tạo, thường được dùng khi muốn:   * Get data từ server. * Lấy giá trị cố định của thành phần như: width, height, offset, ... * Khai báo setInterval cho hành động lặp lại. * Khai báo sự kiện như: load, scroll, resize, … * ... |
| Updating | shouldComponentUpdate() | Phương thức này xác định component nên được update hay không, phương thức này có 2 giá trị:   * true (mặc định) khi này sẽ tiếp tục thực hiện componentWillUpdate() và componentDidUpdate(). * false sẽ dừng thực hiện componentWillUpdate() và componentDidUpdate(). |
| componentWillUpdate() | Phương thức này được gọi một lần duy nhất sau shouldComponentUpdate(), sử dụng trước khi re-render component. |
| **componentDidUpdate()** | Chúng ta gọi phương thức này sau khi DOM đã được update xong, tại đây thường được dùng khi muốn:   * Xử lý dữ liệu được lấy từ server. * Thay đổi giao diện dựa vào dữ liệu nhận được. * ... |
| Unmount | componentWillUnMount() | Phương thức này sẽ được gọi một lần duy nhất trước khi component được loại bỏ từ DOM, những thứ đã khai báo tại componentDidMount() cần phải được hủy. Đây cũng là phương thức kết thúc vòng đời của component. |