React Hook

**React hook là gì?**

* Hooks là một bổ sung mới trong React 16.8.
* Với Hooks bạn có thể sử dụng state và lifecycles mà không cần dùng ES6 Class.

**Tại sao chúng ta cần React Hooks?**

Sau một thời gian làm việc với React thì có lẽ chúng ta sẽ bắt gặp một trong số các vấn đề sau:

* Việc sử dụng lại logic giữa các components khó khăn, các pattern render props và hoc có thể gây ra hiện tượng “Wrapper hell” các component được lồng (nested) vào nhau nhiều tạo ra một DOM tree phức tạp.
* Sự rắc rối của Lifecycles trong class
* Component quá lớn (các phương thức của lifecycle + logic)

**Lợi ích của hook**

* Khiến các component trở nên gọn nhẹ hơn => kích thước bundle sẽ được giảm đáng kể bởi code sử dụng Hooks.
* Dễ tiếp cận với người mới, các khái niệm về lifecycle và class trong React và Javascript tương đối phức tạp

Note:

* Hook hoàn toàn không bắt buộc: Bạn có thể thử dùng Hooks trong một số components mới mà không cần phải viết lại code hiện tại. Bạn cũng không cần phải học hoặc sử dụng Hooks nếu bạn không muốn.
* 100% tương thích ngược: Hooks không chứa các thay đổi có thể khiến break ứng dụng của bạn.
* React hook ra đời không phải là để xóa bỏ các class. Mà Hook đơn giản là thêm một giải pháp để lập trình viên có nhiều cách viết code hơn.

Demo cách viết dùng hook vs class

**useState**

useState dùng để khai báo một “state variable” (biến state). Đây là cách để “lưu giữ” các giá trị giữa các lần gọi hàm. Hàm useState nhận tham số initial state, sau đó sẽ trả về một mảng 2 phần tử, phần tử đầu tiên là state hiện tại, phần tử thứ 2 là hàm để update state - (hàm thứ hai này giống với setState khi chúng sử dụng dạng Class)

**useEffect**

Effect Hook cho phép thực hiện side effect bên trong các function component. Side-effects có thể là những thứ sau trong ứng dụng:

* Lấy dữ liệu – fetching data
* Thay đổi DOM.
* Cài đặt một subscription
* …

useEffect sẽ tương đương với các hàm componentDidMount(), componentDidUpdate() và componentWillUnMount() trong lifecycle của Component

useEffect nhận 2 tham số, tham số đầu tiên là 1 fucntion nơi chúng ta xử lý các side effect, thứ 2 là một mảng các variables, khi mà những variables này thay đổi thì useEffect sẽ được gọi => khi ko truyền tham số thứ 2 thì nó sẽ chạy mọi lần component render. Nếu muốn chỉ chạy 1 lần thì ta truyền vào mảng rỗng.

useEffect có thể trả về 1 hàm và hàm này thực hiện các tác vụ dọn dẹp tương tự componentWillUnmount()

**useContext**

Cách cung cấp Context giống nhau cả ở class và function component, nhưng các sử dụng sẽ khác nhau

**useReducer**

Phiên bản nâng cấp cho useState thường áp dụng khi state chứa nhiều thông tin (e.g: object lồng nhau nhiều cấp, mảng các objects)

**customHook**

Để tái sử dụng logic React cho phép chúng ta tự định nghĩa những hooks khác nhau tuỳ mục đích

Đặc điểm của Custom Hooks

* Là function nhận input và trả ra output
* Tên bắt đầu bời **use**
* Khác với function component, custom hooks trả về dữ liệu bt ko phải jsx
* Khác với function bt, custom hook có thể sử dụng các hooks khác như useState, useContext…