**PV ReacJS**

1. Phân biệt props, state là gì, có thay đổi được ggias trị của props không nếu được thì làm như nào?.

* Props là properties của một component .Về cơ bản props là một đối tượng, nó lưu trữ các giá trị của các attribute (thuộc tính) của một thẻ (Tag). Chúng ta có thể thay đổi giá trị props của một component bằng cách truyền dữ liệu từ ngoài vào. Khi một props được truyền vào component thì giá trị của nó là không thay đổi .
* Chú ý: Khi một props được truyền vào component thì nó là bất biến tức là dữ liệu của nó không được thay đổi kiểu như một "pure" function vậy, sẵn tiện để bài viết thêm dài chúng ta cùng xem qua 2 ví dụ về function "pure" và "not pure"
* Giống như props, sate cũng giữ thông tin về component. Tuy nhiên, loại thông tin và cách xử lý nó khác nhau. State hoạt động khác với Props. State là thành phần của component, trong khi các props lại được truyền giá trị từ bên ngoài vào component Có một lưu ý nhỏ là chúng ta không nên cập nhật state bằng cách sử dụng trực tiếp this.state mà luôn sử dụng setState để cập nhật state của các đối tượng. Sử dụng setState để re-renders một component và tất cả các component con. Điều này thật tuyệt, bởi vì bạn không phải lo lắng về việc viết các xử lý sự kiện (event handler) như các ngôn ngữ khác.

1. SetState là hàm đồng bộp hay bất đồng bộ. Sẽ như thế nào nếu hàm này là hàm đồng bộ(\*).

* **setState() là hàm bất đồng bộ, và dĩ nhiên là nó không thay đổi state xong ngay lúc đó mà sẽ tạo một**pending state transition. Có nghĩa là, log trước và để setState() vào [event loop](https://viblo.asia/p/cuoi-cung-thi-event-loop-la-gi-LzD5dX705jY)tiếp theo.
* Hàm setState() thay đổi state và component sẽ re-render. Đây là một expensive operation và điều này có thể khiến gây ra vấn đề browser unresponsive hay một số ảnh hưởng tới performance khác.

1. DOM ảo là gì gì, quá trình render, re-render được như thế nào?.

* DOM ảo không được phát minh ra bởi React, mà React sử dụng nó. DOM ảo là một bản sao chép trừu tượng của DOM thật (HTML DOM). Bạn có thể tưởng tượng nó giống như một bản thiết kế, chứa các chi tiết cần thiết để cấu hình lên một DOM. Ví dụ, thay vì tạo một thẻ
* thật chứa các thẻ <ul> bên trong, nó sẽ tạo một div object chứa ul object bên trong. Cụ thể ở trong React sẽ là các React.div và React.ul. Khi tương tác, ta có thể tương tác với các object đó rất nhanh mà không phải động tới DOM thật hoặc thông qua DOM API. Tiếp theo chúng ta sẽ tìm hiểu cụ thể React tương tác với DOM ảo như thế nào nhé

1. Em code chủ yếu là function Component hay class Component. Điểm khác nhau giữa 2 thằng.

* Sự khác biệt rõ nhất là cú pháp. Một functional component thực tế chỉ là một hàm Javascript đơn giản chấp nhận các props như là một argrument và trả về một phần tử React.
* Một class component yêu cầu chúng ta cần kế thừa từ phần React.Component và tạo một function render trả về một phần từ React. Điều này sẽ đòi hỏi chúng ta phải code nhiều hơn nhưng cũng sẽ cung cấp một số lợi ích mà chúng ta sẽ thấy sau này.

1. Em hãy trình bày về lifecycle: tuỳ theo mình làm việc với react vesion bao nhiêu mà trình bày ở đây là vesion < 16.4 nên trình bày lifecycle cũ.Còn nếu xài từ 16.4 > thì có thể trình bày them 1 chút về lifecycle cũ.

* Một loạt các thay đổi mà Component trải qua từ khi nó được tạo ra cho tới khi nó kết thúc (bị hủy) được gọi là một vòng đời (lifecycle) của Component. Trong quá trình tồn tại của Component các phương thức sẽ được gọi, dưới đây là hình ảnh minh họa vòng đời của Component và các phương thức sẽ được gọi trong các giai đoạn khác nhau. Các phương thức trong React Component Lifecycle có thể chia ra làm 3 pha chính là: Mounting, Updating và Unmounting.

1. Component Will receiveprops được sử dụng trong trường hợp nào.

* Mục đích chính là để Updating state based on props với param truyền vào của method là nextProps
* ComponentWillReceiveProps sẽ không được gọi khi initial render . Việc setState cần được set trong contructor khi component được init .
* ComponentWillReceiveProps sẽ call mỗi khi nhận được new Props . Vì thế nó sẽ setState lại bất cứ khi nào có props thay đổi mặc dù có thể data của props để setState không thay đổi
* ComponentWillReceiveProps là **synchronous** nhưng setState() **asynchronous** . Điều gì sẽ xảy ra khi một hàm **synchronous** call nhiều lần để xử lý hàm **asynchronous**

1. shoudcomponentupdate được sử dụng trong trường hợp nào .return true/false có nghĩa là gì?.

* shouldComponentUpdate() được thực thi trước khi một component re-render, nó sẽ được gọi khi state hoặc props của component này thay đổi.
* Theo mặc định, mỗi khi parent component re-render sẽ kéo theo các child component re-render, khi state và props của component không đổi, nhưng vẫn phải re-render, đây là điều không cần thiết(ảnh hưởng đến hiệu suất) và chúng ta sẽ tìm cách để ngăn các lần re-render này ^^. Và shouldComponentUpdate() là một lựa chọn cho chúng ta trong việc ngăn re-render của một component .
* shouldComponentUpdate() nhận vào 2 tham số, tham số đầu tiên là nextProps sẽ được so sánh với this.props, tham số thứ 2 là nextState so sánh với this.state. Nếu một trong 2 thay đổi so với trước đó thì component sẽ re-render.
* Theo mặc định thì shouldComponentUpdate() sẽ return về true, tức là sẽ re-render khi parent component re-render, kể cả props và state của nó không đổi.
* Các bạn nên lưu ý rằng việc return false sẽ không ngăn các child components re-render khi state của chúng thay đổi.
* Ví dụ parent component ta set shouldComponentUpdate(){ return false; };. Có nghĩa là parent component sẽ bỏ qua re-render và vì thế các child component không bị re-render. Nếu child component có state thay đổi ví dụ như khi sử dụng các event như onChange, onClick,... Thì child component này vẫn sẽ bị re-render bất kể parent component sử dụng shouldComponentUpdate(){ return false; };.
* Các bạn có thể thấy Counter sử dụng shouldComponentUpdate() sẽ không bị re-render khi parent thay đổi state. Để quan sát rõ hơn thì các bạn click vào clear Console và click vào button Increment.
* Bằng cách sử dụng shouldComponentUpdate() ta có thể hạn chế được các re-render component không cần thiết, điều này sẽ giúp trang web của chúng ta cải thiện được hiệu suất.

1. Trước khi có version 16.8.0 code của function component được sử dụng để làm gì?

* Trước khi ra đời Hooks trong React, các **functional components** được gọi là các**stateless components**  (component không trạng thái) và nằm sau các **class components** trên cơ sở tính năng. Sau khi giới thiệu Hooks, các **functional components** tương đương với các **class components**.

1. Trong function Component(hook) lifecycle được viết ở đâu. Có những loại nào?
2. Tại sao team react lại viết chung lifecycle vào trong useEffect mà không dung tách riêng?

* Sau khi chúng ta đã tìm hiểu xong những cách dùng cơ bản của useEffect Hook,  hẳn bạn sẽ có cảm giác liên tưởng tới lifecycle của component đúng không?
* Thực ra đúng là có sự liên quan nhè nhẹ ở đây.
* Vòng đời của Component trong ứng dụng React,  bạn vẫn còn nhớ những hàm như:
* componentDidMount()
* componentDidUpdate()
* componentWillUnmount()

1. Giải thích ý nghĩa của Dependency trong useEffect.

* Chúng ta đã nhắc đến **useEffect** là một nơi quản lý các **side-effect** bên trong một React Components.
* Là một mảng chứa các đối số mà **useEffect** sẽ phụ thuộc vào đó để thực thi. Trong ví dụ trên là một mảng trống, nó đồng nghĩa với việc chỉ thực hiện một lần duy nhất sau khi component render.
* Khi **dependencies** là các **props, state** bên trong một mảng **[props1, props2,.. stateA]**. **React** **useEffect** sẽ dựa vào giá trị **props**, **state**. Trong lần **render** tiếp theo, nó sẽ kiểm tra giá trị của **props**, **state** mới với giá trị **props**, **state** trước đó. Nếu khác nhau sẽ thực hiện useEffect callback sẽ được gọi. Ngược lại thì không có gì xảy ra
* Cơ chế này tương tự như bạn sử dụng **Life Cycle** **componentDidUpdate** và **shouldComponentUpdate** của **Class Component.**

1. Ngoài useEffect em còn biết các hook nào không, em đã custom được hook nào chưa.
2. HOC là gì?

* Thành phần bậc cao hơn (HOC) là một kỹ thuật nâng cao trong React để sử dụng lại logic thành phần. HOC không phải là một phần của API React. Chúng là một mẫu xuất hiện từ bản chất cấu thành của React.
* Một HOC nhận vào là một component và trả về một component mới
* Công dụng chính của HOC là để nâng cao khả năng tái sử dụng của các component cụ thể trong nhiều mô-đun hoặc thành phần. Chúng ta cũng có thể bao gồm các component khác nhau. Hầu hết các thư viện của bên thứ ba đang sử dụng tính năng này.
* Kế thừa các logic.

1. Promise, callback, async/await là gì/

* Promise sinh ra để xử lý kết quả của một hành động cụ thể, kết quả của mỗi hành động sẽ là thành công hoặc thất bại và Promise sẽ giúp chúng ta giải quyết câu hỏi "Nếu thành công thì làm gì? Nếu thất bại thì làm gì?". Cả hai câu hỏi này ta gọi là một hành động gọi lại (callback action)
* Callback là gì? Nói một cách đơn giản: Callback **là một hàm sẽ được thực hiện sau khi một hàm khác đã thực hiện xong** - vì thế nó có tên là callback. Cụ thể hơn: Trong JavaScript, hàm là đối tượng. Do đó, các hàm có thể lấy các hàm làm đối số và có thể được trả về bởi các hàm khác
* **Async / Await** là một tính năng của JavaScript giúp chúng ta làm việc với các hàm bất đồng bộ theo cách thú vị hơn và dễ hiểu hơn. Nó được xây dựng trên Promises và tương thích với tất cả các Promise dựa trên API. Trong đó:
* **Async** - khai báo một hàm bất đồng bộ (async function someName(){...}).
  1. Tự động biến đổi một hàm thông thường thành một Promise.
  2. Khi gọi tới hàm async nó sẽ xử lý mọi thứ và được trả về kết quả trong hàm của nó.
  3. Async cho phép sử dụng Await.
* **Await** - tạm dừng việc thực hiện các hàm async. (Var result = await someAsyncCall ().
  1. Khi được đặt trước một Promise, nó sẽ đợi cho đến khi Promise kết thúc và trả về kết quả.
  2. Await chỉ làm việc với Promises, nó không hoạt động với callbacks.
  3. Await chỉ có thể được sử dụng bên trong các function async.

1. Trình bày lý do khi nào nên sử dụng redux.
2. Ngoài redux còn cách nào để share dữ liệu không.
3. Em thường làm việc với các middleware nào.
4. Em biết rudux-thunk không?
5. Em biết SASS không.
6. Quy tắc đặt tên BEM.
7. Ngoài ra em có biết them thư viện hỗ trợ css hay xâu dựng UI nào không(tailwind,…).
8. Em có biết typescript.
9. Em có biết về thư viện hay framework nào hỗ trợ server-side-rendering không ?
10. Theo em reactjs là CSR hay SSR.
11. useMemo() là gì?
12. useRef() là gì?
13. useReducer() là gì?
14. Inversion of Control (IoC) **là một nguyên lý thiết kế trong công nghệ phần mềm trong đó các thành phần nó dựa vào để làm việc bị đảo ngược quyền điều khiển khi so sánh với lập trình hướng thủ thục truyền thống.**
15. Document Object Model - DOM ("Mô hình Đối tượng Tài liệu"), là một giao diện lập trình ứng dụng (API). DOM được dùng để truy xuất các tài liệu dạng HTML và XML, có dạng một cây cấu trúc dữ liệu, và thông thường mô hình DOM độc lập với hệ điều hành và dựa theo kỹ thuật lập trình hướng đối tượng để mô tả tài liệu.