## ZÁCâu 1: Phân biệt khai báo var, let ,const trong javascript

Trả lời:

Var có 2 scope là globalscope và function scope. Trừ trường hợp được khai báo trong function thì var có scope là global scope, nghĩa là nó có thể truy cập được ở bất kỳ đâu trong chương trình. Từ EMCA2015, người ta đã khuyên không dung var mà thay bằng let, const.

Var có tính hoisting: Hoisting nghĩa là dù khai báo biến sau khi gán giá trị, khi thực thi chương trình việc khai báo biến với var vẫn được chuyển lên đầu phạm vi scope của nó

LET: let có phạm vi blockscope. Blockscope là phạm vi giới hạn bởi cặp ngoặc nhọn. Biến khai báo với let thì có thể ghi đè được, let cũng có hoisting nhưng nó không khởi tạo giá trị nào cho biến ban đầu bằng undefined như ở khai báo Var nên với let ta phải khai báo biến trước khi sử dụng

CONST: const cũng giống như LET cũng có phạm vi blockscope và cũng hoisting giống như LET. Khi khai báo bằng const thì biến không thay đổi được giá trị với primitive value (number, string, Boolean, null, undefined) còn với các reference thì vẫn có thể thay đổi được giá trị của các thuộc tính bên trong

## Câu 2: EventLoop trong JS

Event loop trong JS là một vòng lặp vô tận. Vì JS là single thread đơn luồng nên khi nó chạy các tác vụ mất nhiều thời gian phản hồi như callAPI, bất đồng bộ, setTimeout thì nó sẽ được chuyển sang cho WEB API xử lý. Web aPI sẽ chuyển những tác vụ đó xuống Quece (hàng đợi) đến khi callstack trống (xử lý xong các tác vụ đồng bộ) thì eventloop sẽ chuyển các tác vụ bất đồng bộ từ quece vào callstack.

Các tác vụ được chuyển từ WEBAPI (với nodejs sẽ là nodeAPI) vào quece được chia làm 2 loại là microtask và MacroTask. MicroTask như promise, callback sẽ được ưu tiên đưa vào callstack trước MacroTask (mouse event, timer ….)

## Câu 3: Asynconous and Sync

Async là các tác vụ bất đồng bộ, tốn thời gian để xử lý như callAPI, Request

Sync là các tác vụ đồng bộ thực hiện theo thứ tự tuần tự trong JS

## Câu 4: RestFul API là gì? HTTP method là gì, các phương thức HTTP, status http

Restful API là một tiêu chuẩn thiết kế API.

Giao thức HTTP (HyperTextTransferProtocol) giao thức truyền tải siêu văn bản được sử dụng để truyền tải dữ liệu từ client lên server và ngược lại

Các phương thức HTTP cơ bản: GET, POST, PUT , PATCH, DELETE

Các status code phổ biến:

200 – thành công;

201 – create trên server thành công và trả về đối tượng mới được tạo,

204 – DELETE thành công , không có dữ liệu được trả về.

304 – Not modified-Client có thể sử dụng dữ liệu cached

404- Link url không đúng

401 – [Unauthorized](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401)

408- Quá thời gian request phản hồi (request timeout)

429 – Too many Request

500 – Server Error

Sự khác biệt giữa PUT và PATCH

Giống nhau: Cả 2 đều được dùng để modify dữ liệu trên resource

Khác nhau:

Khi gửi dữ liệu với PUT, dữ liệu được gửi lên sẽ thay thế dữ liệu cần được update. Ví dụ ta cần update tại id = 3 đang có {firstname: “Trung”, lastName:”Pham’} mà ta gửi dữ liệu kèm PUT lên chỉ là {firstname: “Nam”} thôi thì dữ liệu tại id = 3 sẽ bị thay thể mất trường lastname chỉ còn là {firstname: “Nam”}

Khi gửi dữ liệu lên với PATCH, dữ liệu được gửi lên sẽ được bổ sung vào dữ liệu hiện có, nếu có 🡺

{ firstname; “Nam”, lastName: “Pham”}

## Câu 5. State là gì, props là gì. Sự khác nhau giữa state và props

|  |  |
| --- | --- |
| State | Props |
| được dùng để quản lý bên data trong Component | Nó được truyền từ Component cha xuống Component con |
| State được thay đổi bởi việc sử dụng hàm setState. | Read-only |
| Khi State thay đổi thì Component re-render | tăng tính tái sử dụng của Component |
| Có thể coi như Local variable | Có thể coi như đối số của function |

## Câu 6: Bạn biết những Hooks nào trong ReactJS công dụng của chúng, UseMemo và useCallBack dùng để làm gì, chúng khác nhau như thế nào, Vòng đời của React, Usecontext và Redux dùng để làm gì, chúng khác nhau ntn. Khi nào sử dụng reactContext, khi nào sử dụng redux

### a/ Các hooks in ReactJS hay sử dụng:

useState, useEffect, useLayoutEffect, useRef, useMemo, useCallback, customHooks, useContext, useReducer

ForwardRef là gì, dùng nó để làm gì?

### b/ Công dụng của từng hooks

**+ UseState:** lưu giữ thông tin, giá trị bên trong component, render lại component mỗi khi state thay đổi

**+ UseEffect:** cũng re-Render lại component phụ4 thuộc vào dependency được truyền trong useEffect. Nếu không có dependency thì useEffect sẽ chạy mỗi khi component render 🡺 tương đương Component DidUpdate, sau khi component render useEffect sẽ được chạy và cập nhật lại giao diện. Nếu truyền dependency là mảng trống thì useEffect sẽ chỉ chạy một lần đầu tiên =🡺 tương đương componentDidMount, nếu truyền dependency là mảng với các tham số thì useEffect sẽ chạy khi các tham số trong depentdency thay đổi. Khi useEffect return thì nó xóa các sideEffect được tạo ra của lần render trước 🡪 tương đương lifecycle Unmount

useEffect thường được dùng để thực hiện các sideEffect call api, thực hiện các hoạt động bất đồng bộ, handle Event, cancel event, lắng nghe xử lý và xóa bỏ các sideEffect như socket, setTimeout….

**+ UseLayoutEffect** cũng tương tự như useEffect nhưng nó chạy đồng bộ trước khi DOM được render

**+ useRef:** giá trị của useRef là không đổi qua mỗi lần Re-render. UserRef cũng được dùng để gán vào element thay cho việc querySelector

**+ useMemo:** usememo được sử dụng để một giá trị không đổi không cần phải được tính toán lại mỗi lần render (memorize value). Vì mỗi lần render Component sẽ tạo ra một phạm vi mới, kể cả các giá trị không đổi vẫn được tính toán lại làm ảnh hưởng performance vì vậy việc sử dụng useMemo sẽ giúp ngăn chặn việc tính toán lại làm giảm performance này

**+ UseCallback** thì cũng khá giống với useMemo. Nếu như useMemo là nhớ giá trị thì với useCallback sẽ là nhớ function. Việc render lại component sẽ không tạo ra function mới. Nếu truyền dependency của useCallBack là empty thì sẽ không bao giờ tạo ra callback mới

\*Lưu ý: Nhược điểm của useMemo và UseCallback: tốn bộ nhớ, với những ứng dụng nhỏ thì không nên sử dụng useCallback và useMemo

**+ UseReducer**: Một sự thay thế cho useState với các state phức tạp. Nó hoạt động giống như redux có các hàm reducer và cũng dispatch action lên reducer, dựa trên các actiontype để cập nhật state

**+ CustomHooks:** Khi sử dụng một cùng một logic state ở nhiều component thì ta sẽ nghĩ đến customHooks. Một customhooks vẫn có sử dụng các useState và useEffect bình thường, các giá trị trả ra của Hooks custom sẽ được sử dụng trong các component khác. Một custom hooks sẽ phải bắt đầu với từ khóa **use (VD useFetch)**

**+ Usecontext:** Việc truyền props giữa các component từ cha xuống con, từ con xuống cháu theo từng bước 1 (props dilling) khá phức tạp 🡺 tăng số lượng code -> khó quản lý. Việc sử dụng useContext làm cho dữ liệu được quản lý tập trung các component con, cháu, chắt muốn lấy dữ liệu từ component gốc đơn giản hơn nhiều

### c/ Vòng đời của React

ComponentWillMount: Trước khi react render vào DOM

ComponentDidMount: Ngay sau khi react render DOM lần đầu tiên

ComponentDidUpdate: Ngay sau khi react update render component

ComponentUnMount: Khi Component bị xóa khỏi DOM

Trong React Class component vòng đời sẽ là mounting-updating-Unmounting

Component willmount -render - Component didmount -componentwillReceiveProps – shouldComponentUpdate – Component willupdate -

### d/ UseContext và Redux dùng để làm gì? Chúng có điểm gì khác nhau? Cái nào tốt hơn cái nào? Khi nào thì sử dụng

|  |  |
| --- | --- |
| **useContext** | **Redux** |
| **Giống nhau:** đều được dùng để quản lý state (state management). Bất kỳ component nào muốn dùng state đều có thể lấy được mà không cần phải propsdilling như trước. tùy từng nhu cầu dự án mà ta lựa chọn sử dụng redux hoặc react-context | |
| Là một hooks của react | Là một library có thể được sử dụng với react hoặc những library khác |
| Nếu dự án nhỏ, không quá phức tạp thì sử dụng react-context | Nếu dự án lớn, phức tạp hơn thì ta nên sử dụng Redux |

### e/Flow của Redux

Redux hoạt động theo luồng như sau:

User kích hoạt một event phía browser 🡺 việc này sẽ kích hoạt dispatch action lên store 🡺 reducer trong store sẽ xử lý dựa trên state cũ và action được gửi lên mà trả ra một state mới 🡺 tiến hành re-render lại component

Sự khác biệt giữa redux và redux-toolkit: redux toolkit (bộ công cụ redux) đã được team redux tối ưu hơn. Thay vì việc phải tạo các action, các reducer thủ công thì redux-toolkit ta chỉ cần tạo các slice và các actioncreator và reducer sẽ được tạo ra từ các slice đó.

## Câu 7: Closure trong Javascript

Bình thường trong javascipt khi một biến được khai báo global, nó sẽ tồn tại trong suốt chương trình. Nếu biến được khai báo trong hàm, biên sẽ chỉ tồn tại trong hàm, khi hàm thực thi xong thì biến cũng sẽ bị xóa. Nhưng với closure, là các biến của hàm con có khả năng truy vấn đến các biến của hàm cha, mặc dù hàm cha đã thực thi xong nhưng biến của hàm cha vẫn tồn tại để hàm con, cháu, chắt có thể truy cập được

## Câu 8: CallBack, Promise, async await trong JS

Giống nhau: Chúng đều có thể được dùng để xử lý bất đồng bộ trong Javascript

Khái niệm callback: Một hàm được sử dụng, được gọi trong một hàm khác để biểu diễn một nhiệm vụ gì đó. Callback có nhược điểm là khi có quá nhiều callback lồng nhau thì dẫn đến code khó đọc, khó maintain 🡺 Promise ra đời

Promise thì có mang 2 function làm đối số là resolve và reject. Promise có 3 trạng thái fulfilled, pending, rejected. Trạng thái pending là đang chờ, trạng thái fulfilled là lời hứa đã được giải quyết thành công, trạng thái rejected là trạng thái lời hứa thất bại. fulfilled thì câu lệnh trong .then được thực thi, rejected thì lỗi được bắt ở catch và finally thì dù resolve hay reject thì đều được thực thi

Async await: Từ khóa async đứng trước function sẽ return 1 promise. Từ khóa await chỉ có tác dụng trong hàm async và đứng trước một promise. Việc sử dụng async, await làm cho code đẹp hơn, chạy theo từng dòng nên việc maintain dễ dàng hơn

## Câu 9: ORM là gì, sequelize

ORM (Object relation mapping) là một ánh xạ chuyển đổi cơ sở dữ liệu sang dạng object để dễ dàng thao tác với ngôn ngữ lập trình

## Câu 10: setState là hàm đồng bộ hay không đồng bộ. Sẽ ra sao nếu setState là hàm đồng bộ

setState trong react là hàm bất đồng bộ, chúng thường được gom nhóm các state theo từng đợt render để tăng hiệu suất. Nếu setState là hàm đồng bộ performance của chương trình sẽ bị giảm do việc render component bị trì hoãn, nặng hơn có thể bị treo

## Câu 11: CreateAsyncthunk là gì. Vì sao sử dụng nó

createAsyncthunk là một hàm được tạo sẵn trong reduxtoolkit. Nó mang tham số là một string tiền tố, 1 callback function mà giá trị return của callback được tự động dispatch lên reducer. CreateAsyncthunk cũng tự động tạo ra 3 trạng thái như một promise là pending, fulfilled, rejected. Dựa vào 3 trạng thái này ta có thể setloading trong quá trình pending, xử lý trong trường hợp thành công, thất bại

## Câu 12: JSX là gì trình duyệt có đọc trực tiếp JSX không, nếu không thì dùng gì để đọc, DOM ảo là gì, component là gì? Oneway binding là gì ? Tại sao cần key trong reactJS

JSX là mã JS được viết chung với HTML. Trình duyệt không đọc được JSX trực tiếp mà qua babel

DOM ảo là một bản sao, đại diện của DOM thực nhưng nhẹ hơn. Khi có thay đổi đối tượng DOM ảo chỉ thay đổi đối tượng đó trong DOM thực thay vì cập nhật tất cả đối tượng 🡺 tăng hiệu suất

Component nó là một thành phần, bộ phận (có thể như là 1 button, form …). Nó giúp cho việc tái sử dụng trong react

Onewaybinding trong react: dữ liệu được truyền tải một chiều (từ trên xuống dưới) => tạo sự đồng nhất

## Câu 13: For in for of trong JS

For in được dùng để lặp qua các thuộc tính property của object hoặc các index cua array

Câu 14: OOP 4 tính chất

Tính đa hình

Tính đóng gói

Tính trừu tượng

Tính kế thừa

Tính đóng gói trong OOP được thể hiện qua phạm vi truy cập với các từ khóa public, private, protected

Public: có thể truy cập được ở mọi nơi

Protected: có thể truy cập được ở trong dẫn xuất class và trong class của nó

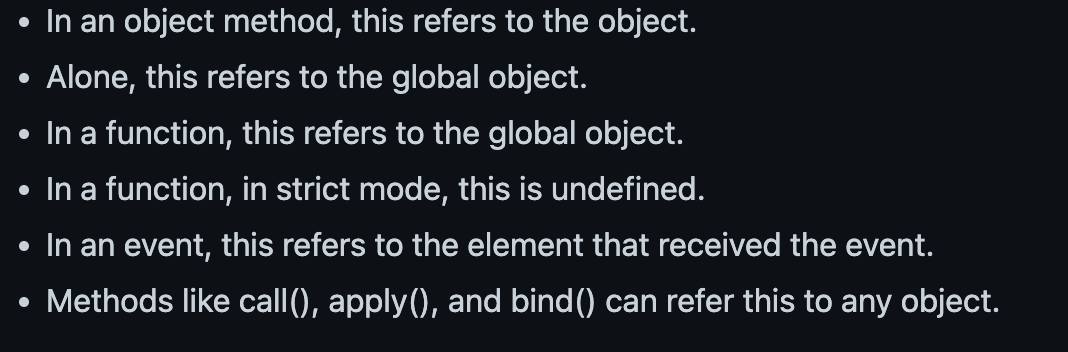
Private: chỉ có thể truy cập được trong class, không thể truy cập trong dẫn xuất hoặc phạm vi bên ngoài

Tính kế thừa: dẫn xuất class kế thừa tất cả các thuộc tính, method dùng với public và protected từ class cha và nó cũng có các thuộc tính riêng của nó

Tính đa hình: giả sử nhóm động vật đều có hành động ăn, nhưng con mèo ăn chuột, con người ăn cơm, bò ăn cỏ. Đa hình thái

Tính trừu tượng: tất cả các loài động vật đều ăn nhưng do chưa định nghĩa cụ thể loài nào nên ta định nghĩa 1 abtract ăn ở class cha

## Câu 14: This trong JS



## Câu 15: Webpack là gì, tại sao lại dùng nó (có lợi ích gì , nhược điểm của nó là gì).

Webpack là công cụ cho phép chúng ta đóng gói (bundle) các file code ,hình ảnh thành các file có thể chạy được trong môi trường web, chúng đóng gói nhiều file thành 1 file, sử dụng các plugin, các loader hỗ trợ (CSS loader, hot module reload….), cho phép developer có thể config project tùy chỉnh theo ý muốn, chỉ load nhưng module và plugin cần thiết

## Câu 16: Có những cách nào tối ưu react:

Spilitting component, useMemo, useCallBack, lazyload, React.memo, hạn chế những component không cần thiết như Modal, hạn chế sử dụng state nhất có thể, sử dụng HOC (higher order components), sử dụng debounce, throttling cho input, đặt state hợp lý

Tối ưu react ta có thể

sử dụng useMemo để ngăn việc tính toán lại các giá trị đắt giá, mất nhiều thời gian khi dependency của nó không đổi

sử dụng useCallBack để ngăn việc tính toán lại function khi dependency ko thay đổi

sử dụng React.memo higher order component để ngăn việc re-render lại component khi props không thay đổi mà component cha re-render dẫn đến component con re-render

sử dụng các custom hooks để tái sử dụng trong project (giảm gọn mã, tối ưu hóa logic… useFetchData, useDebounce, useReize..)

sử dụng lazyload, code spilit để tối ưu hóa thời gian tải trang

sử dụng vitualzation, hoặc initifyScroll để render các list có số lượng data lớn (thư viện có thể sử dụng như react-window, initifyscroll react…..)

tối ưu các dependency trong useEffect, tránh việc render không cần thiết

Chia nhỏ các component sẽ giúp cải thiện hiệu suất ứng dụng hơn

Hạn chế việc sử dụng hooks vô tội vạ, có những trường hợp sử dụng hooks phải set lại state sẽ dẫn đến component re-render

Đặt state đúng vị trí mà component cần nó

Câu 17: Tại sao không nên cập nhật trạng thái trực tiếp mà sử dụng lệnh gọi setState?  
Cách thông thường để cập nhật trạng thái là sử dụng lệnh gọi setState. Nếu không sử dụng nó, người dùng vẫn có thể sửa đổi trạng thái, nhưng nó sẽ không cập nhật DOM để phản ánh trạng thái mới.

## Câu 18,Để render 1 long list hay 1 table data lớn, làm thế nào để tối ưu

ta có thể sử dụng react-window để tối ưu, (nói rõ cách react-window thực hiện)

ta cũng có thể sử dụng react-infiniti-loop để tối ưu (nói cách thực hiện nó)

With react-window, you can efficiently render a large list by only rendering the visible items,

## Câu 19: Typescript là gì, tại sao phải dùng nó

Typescript là một superset của javascript. Nó được thiết kế để cho việc viết code javascript trở nên dễ đọc, dễ hiểu và dễ maintain hơn bởi việc yêu cầu khai báo kiểu cho biến, function. Việc sử dụng typescript trong các project lớn, có nhiều người tham gia sẽ làm cho project clean hơn, dễ dàng đọc và bảo trì sau này

## Câu 20: Pure function là gì, pure components là gì

Pure function, pure component là việc function, component không làm thay đổi giá trị của biến ở bên ngoài function, bên ngoài component. Cùng với 1 input, pure function pure component sẽ cho ra cùng 1 output

## Câu 21: Restful API là gì

## Câu 22: Tấn công XSS, XRSF trình duyệt là gì, bảo mật nó như thế nào, http only

Tấn công xss, xrsf trình duyệt là việc kẻ xấu lợi dụng thông tin trên trình duyệt của user để thực hiện hành vi mà không được sự cho phép của user. Khi user click vào đường link của kẻ xấu, mã được chèn vào sẽ được thực thi và thực hiện ý đồ xấu

## Câu 23: interface với type khác nhau như thế nào

Interface với type về cơ bản là giống nhau, khác nhau là interface thì ta có extends được còn type thì không, 2 interface giống nhau khác tên thuộc tính thì có thể merge vào với nhau còn type thì không

## Câu 24: Có biết CustomHooks không, những custom hooks nào hay sử dụng, ý nghĩa của nó là gì, thử viết nó được không?

useFetchData: tái sử dụng cho việc gọi data

useDebounce: debounce giá trị input kể từ lần nhập cuối, tối ưu hóa hiệu suất vì server sẽ không bị query liên tục (nếu mà search realtime, còn nếu nhập input xong click button search thì không nói làm gì)

useResize: dùng để nhận biết được inner witdth window when resize => phục vụ dễ dàng responsive

……

## Câu 25: useLayoutEffect vs useEffect

hầu hết trường hợp ta có thể sử dụng useEffect thay cho useLayoutEffect, trong 1 số trường hợp ta cần thực hiện hành động nào đó đặc biệt trước khi trình duyệt kịp vẽ lại DOM thì ta nên sử dụng useLayoutEffect. Dưới đây là các trình tự thực hiện của từng Effect

khi 1 component được mount, component được render trước, sau khi render xong trình duyệt sẽ bắt đầu vẽ UI ra màn hình, với useLayoutEffect nó sẽ chạy trước quá trình UI này, nên sẽ tránh được việc bị giật giật màn hình (nhấp nháy) nhưng cũng có nhược điểm là nó sẽ bị trắng trang trong lúc đó vì trình duyệt chưa vẽ UI, trong khi useEffect chờ vẽ xong UI nó mới chạy

1. Render component.
2. **useLayoutEffect** thực thi.
3. Trình duyệt vẽ lại giao diện và cập nhật DOM.
4. **useEffect** thực thi.

## Câu 26. == khác với === ở chỗ nào trong js

* Toán tử **==** thực hiện chuyển đổi kiểu dữ liệu (implicit type conversion) trước khi so sánh giá trị. Do đó, trong một số trường hợp, dù hai biểu thức có kiểu dữ liệu khác nhau nhưng giá trị giống nhau, toán tử **==** vẫn trả về **true** sau khi chuyển đổi kiểu ngầm.
* Toán tử **===** thực hiện so sánh giá trị và kiểu dữ liệu mà không có chuyển đổi kiểu dữ liệu ngầm. Nếu cả giá trị và kiểu dữ liệu của hai biểu thức không khớp, toán tử **===** trả về **false**.