**\* References:**

- <https://vuilaptrinh.com/>

- <https://hocjavascript.net/>

- <https://anonystick.com>

- <https://jsonplaceholder.typicode.com/>

- <https://github.com/sudheerj/reactjs-interview-questions?fbclid=IwAR0DAEhllJy9vomzciaQNEA0o9qEztzQdWrajqNCGbcQlgkQj6nin8aECbM>

- <https://topdev.vn/blog/hoan-doi-hai-gia-tri-ma-khong-can-bien-phu-trong-javascript/>

- <https://devhints.io/>

- https://github.com/yeungon/In-JavaScript-we-trust

**\* How to uninstall npm packages?**

- use "npm uninstall <package\_name> --save": remove from your dependencies

- use "npm uninstall <package\_name> --save-dev": remove from your devDependencies

- use "npm uninstall -g <package\_name>": remove global

**\* How to prevent scroll when open Modal**

- use 'useEffect' & document.body.style.overflow(X or Y) = 'hidden' or 'unset'

\* **Handle variable name has special characters**

https://stackoverflow.com/questions/29482226/json-object-with-dash-character-on-element-name

**\* What is Module?**

- Module is a file, or can say "One script is one module"

+ Import: allow import the functionality from other modules.

+ Export: declare variables or functions that allow other modules to access and use.

/ Named Export: Trong JavaScript ES6, named export được sử dụng để xuất nhiều thứ từ một module bằng cách thêm keyword export vào khai báo của chúng. Những thứ được export sẽ được phân biệt bằng tên. Sau đó import những thứ chúng ta cần sử dụng bằng cách bao quanh chúng cặp dấu ngoặc nhọn { }. Tên của module đã nhập phải giống với tên của module đã xuất.

/ Default Export: Default Export trong Javascript ES6 chỉ cho phép xuất một mặc định cho mỗi file. Default Export có thể cho một function, class hoặc một object.

https://anonystick.com/blog-developer/noi-ve-import-va-export-trong-javascript-202005116068041

**\* new Set([iterable]): lets you create Set objects that store unique values of any type.**

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Set/Set

**\* Prefix (tiền tố) vs Postfix (hậu tố):**

- Prefix: + First: Calculate

+ Final: Return value

- Postfix: + First: Copy variable

+ Second: Calculate

+ Final: Take "copy variable", return it

**\* For loop:**

- For-in: lặp các key của object, index của array.

- For-of: lặp các value của object, value của array, string.

**\* Break vs Continue**

- Break: The break statement "jumps out" of a loop.

- Continue: The continue statement "jumps over (skip)" one iteration in the loop and continues execution of the loop with the next iteration.

**\* Logical Operators:**

- &&: return first "falsy" value, if all are "truthy" return final value.

- ||: return first "truthy" value, if all are "falsy" return final value.

**\* Value type & Reference type:**

- Value type: number, string, boolean, undefined, null, BigInt, Symbol

+ Lưu dữ liệu trực tiếp trong ô nhớ

+ Khi tạo biến mới thì tạo ô nhớ mới cho biến đó

+ Thay đổi giá trị của biến này sẽ không ảnh hưởng tới biến khác

- Reference type: array, object, function

+ Lưu dữ liệu vào trong ô nhớ. Trả về địa chỉ đã lưu và gán lại cho biến.

+ Gán lại biến cho 1 (object, array, function) mới thì sẽ có 1 địa chỉ mới được tạo ra

+ Thay đổi giá trị của biến này sẽ làm thay đổi giá trị biến khác

**\* Timing:**

- setTimeout(function, milliseconds): Thực thi một function sau một thời gian quy định.

- setInterval(function, milliseconds): Thực thì một function sau một thời gian quy định và và tiếp tục lặp lại nó

**\* JSON:**

- Là một định dạng dữ liệu (chuỗi).

- JSON thể hiện các kiểu dữ liệu: number, string, boolean, null. array, object

- Có 2 phương thức:

+ Stringify: chuyển từ Javascript data types => JSON (gửi dữ liệu lên server)

+ Parse: chuyển từ JSON => Javascript data types (nhận dữ liệu từ server)

**\* Promise:**

- Promise.all(): all asynchronous function will run parallel.

**\* Async/Await:**

- async: khai báo một hàm bất đồng bộ

+ Keyword before a function makes a function return a Promise. (Tự động biến đổi một hàm thông thường thành một Promise.)

+ Khi gọi tới hàm async nó sẽ xử lý mọi thứ và được trả về kết quả trong hàm của nó.

- await: tạm dừng việc thực hiện các hàm async.

+ Keyword before a function makes the function wait for a Promise. (Khi được đặt trước một Promise, nó sẽ đợi cho đến khi Promise kết thúc và trả về kết quả.)

+ Can only be used inside an async function.

**\* Function:**

- Cho tham số nhưng k truyền đối số thì khi gọi hàm sẽ trả về undefined.

- Không truyền tham số thì arguments sẽ là tất cả giá trị khi gọi hàm có truyền vào đối số.

- In the function, keyword "return" will return any data value. If the function not "return" any value, the function return undefined.

+ Declaration Function: hoisting, can call function before defined.

+ Expression Function: not hoisting, obligate declare function before call it. Function expressions can be made "self-invoking".

+ IIFE (Immediately Invokable Function Expression): là khởi tạo một function và thực thi ngay lập tức sau đó.

+ Callback: is a function. được truyền qua đối số khi gọi hàm khác

\* Variable:

- Global and local variables with the same name are different variables. Modifying one, does not modify the other.

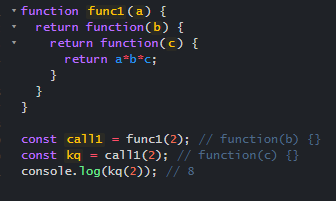
- Variables created without a declaration keyword (var, let, or const) are always global, even if they are created inside a function.

- Variable Lifetime: + Global variables live until the page is discarded, like when you navigate to another page or close the window.

+ Local variables have short lives. They are created when the function is invoked, and deleted when the function is finished.

**\* Currying Function:**

**-** Là function trả về function

****

**\* JSX rules:**

- Return single element.

- Div/ section/ article/ React.Fragment.

- Use camelCase for HTML attribute.

- ClassName instead of class.

- Close every element.

- Formatting.

**\* Props:**

- An empty object.

- Passed from Father component to Child components.

- Child components pass propsName=propsValue and received 1 parameter (called props or whatever).

**\* useEffect:**

- Dependency:

+ No dependency: always render.

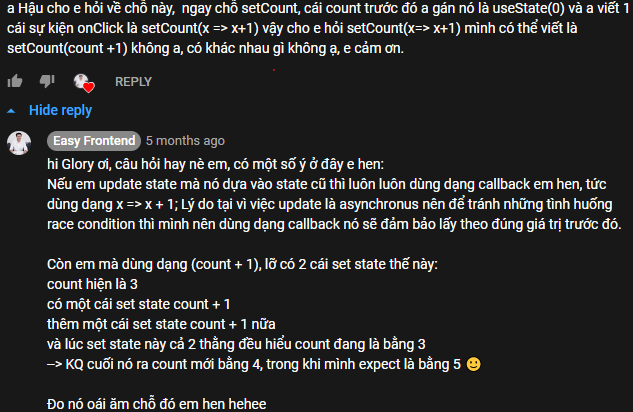
+ Dependency (empty array): render only 1 time.

+ Dependency (value): re-render when value changed.

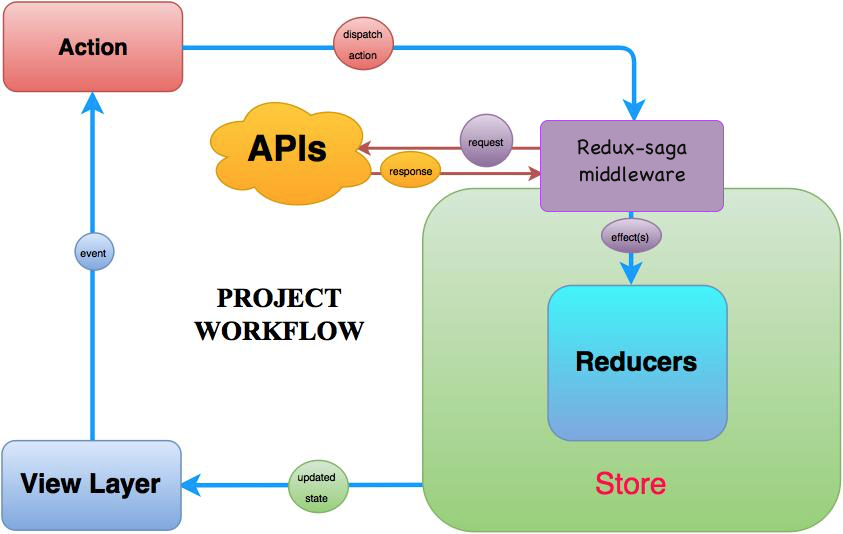
- If has "cleanup function": always call before useEffect run and avoid re-render.

**\* useState:**

- Trong setState có thể viết function với tham số là state cũ hoặc state hiện tại trước khi update: setState((prevState) => // code... return prevState);



**\* Redux:**

****

- Khái niệm trong redux:

+ Action creator: Là một function trả về một action (plain object: plain object có chứa 1 field là type và bất kì dữ liệu nào ta muốn thêm vào).

+ Combine reducers: Trong initialState có nhiều object. Khai báo reducer tương ứng với mỗi object. Dùng combine reducers để gộp các reducer lại (combine trả về object)

Vd: const reducer = combineReducers({ key1: reducer1, key2: reducer2,...,key(n): reducer(n) });

\*\*\* Redux Thunk

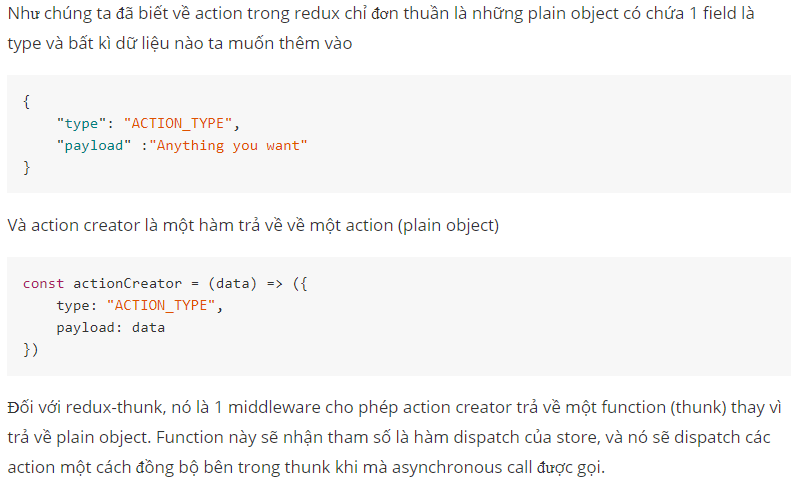
<https://kipalog.com/posts/Xu-ly-Asynchronous-Redux-action-voi-Redux-thunk>

<https://user3141592.medium.com/understanding-the-redux-thunk-source-code-b3f8b930faf6>

- *Cách hoạt động của Redux-middleware: Khi user thực hiện một action, action sẽ được dispatch và đưa tới reducer. Trên đường tới reducer sẽ đi qua middleware, tại đây store đã có giá trị và được gửi tới reducer. Tại reducer, store sẽ được cập nhật lại giá trị tương ứng với mỗi action và reducer gửi giá trị ra màn hình để render lại.*

- Redux middleware cho phép chúng ta can thiệp vào giữa thời điểm dispatch một action và thời điểm action đến được reducer.

- Khi một action được dispatch vào reducer, thì nó sẽ kiểm tra xem action đó có thực thiện bất đồng bộ hay không, nếu có nó sẽ chờ cho action bất đồng bộ thực hiện xong rồi mới đưa action vào trong reducer.



\*\*\* Redux Saga

* **Fork:**
* Sẽ truyền vào 1 hàm: là generator function hoặc function bình thường. Hàm này dùng để theo dõi các action.
* Trong hàm rootSaga có thể có nhiều fork khác nhau.
* Mỗi fork là non-blocking: nghĩa là có thể dispatch nhiều action cùng một lúc.
* **Take:**
* Take là blocking.
* Lệnh take sẽ được kích hoạt và tham gia vào saga khi một action được dispatch.
* Khi nào dispatch xong action thì các câu lệnh dưới yield take() mới được thực thi.
* **Call:**
* Call là blocking.
* Thường dùng để gọi API.
* Sẽ truyền vào 1 hàm: là generator function hoặc function bình thường. Kết quả trả về là Promise và sẽ tạm dừng saga cho đến khi Promise được resolved.
* **Put:**

Put là non-blocking.

Dùng để dispatch action.

* **Delay:**

Delay là blocking.

Chặn thực thi trong khoảng thời gian (miliseconds).

* **TakeLatest:**

Là phiên bản nâng cấp của fork.

Thay thế cho fork, take và infinity loop.

Nếu thực thi một loạt action thì takeLatest chỉ thực thi và lấy kết quả cuối cùng.