

-----HTML-----

1. Các loại CSS? độ ưu tiên CSS?
2. Có 3 bao nhiêu cách để căn giữa 1 phần tử?
3. Các kĩ thuật căn chỉnh bố cục phần tử : flex, grid, float?
4. Tìm hiểu cấu trúc CSS box model?
5. Đơn vị rem trong css?
6. HTML5 là gì? các tính năng mới của html5
7. CSS3 là gì? các thuộc tính mới của css3

-----JS-----

1. Browser xử lý JS như nào ?

2. Cơ chế hoisting là gì ?

3. Callback function là gì?

=> Callback function là một function A được truyền vào một function B dưới dạng một tham số. Tại một thời điểm nào đó tùy thuộc vào cách function B được xây dựng mà function A sẽ được function B gọi để thực thi.

Extra: Hàm map cũng được gọi là blocking callback function, hàm event cũng gọi là callback function

=> Chia làm 2 loại blocking callback(synchronous) và non-blocking callback(asynchronous)

=> Blocking callback, lí do là vì HOF(hàm cha) nên sẽ không hoàn thành nếu hàm callback(hàm con) chưa được chạy xong

```
function map(array, callback) {  
  console.log('map() starts');  
  const mappedArray = [];  
  for (const item of array) { mappedArray.push(callback(item)) }  
  console.log('map() completed');  
  return mappedArray;  
}  
  
function greet(name) {  
  console.log('greet() called');  
  return `Hello, ${name}!`;  
}  
  
const persons = ['Cristina'];  
  
map(persons, greet);  
// logs 'map() starts'  
// logs 'greet() called'  
// logs 'map() completed'
```

example:

```
const sayHello = (name, callback) => {
```

```
const msg = "Hello, " + name;
return callback(msg);
};

const useCallBack = sayHello("Khoa", (msg) => {
  return msg;
});

console.log(useCallBack);
```

4.1. Có mấy cách khai báo function trong js. Khác nhau là gì?

=> Có 2 cách để khai báo function:

- Declaration function (DF):

```
function logSomething() {
  // do something
}
```

- Expression function (EF):

```
const logSomething = function () {
  // do something
};
```

DF bị ảnh hưởng bởi cơ chế hoisting, còn EF thì không. Vì vậy, nếu gọi hàm viết dưới dạng EF chạy trước khi khai báo sẽ bị lỗi.

5. Em hiểu gì về event loop

Link: <https://topdev.vn/blog/hieu-ve-javascript-bat-dong-bo-vong-lap-su-kien/>

6.Sử dụng gì để call api, ngoài axios?

=> Xhr(XML HttpRequest xài dạng callback nhưng kh còn xài)

=> fetch và AJAX

7.Cho câu sau, thứ tự in ra đúng là gì ?

```
setTimeout(() => console.log('a'),200)
setTimeout(() => console.log('b'),0)
```

`console.log('c')`

=> Explain: hàm `setTimeout` là hàm bất đồng bộ, sau khi đọc đến đây. JS Engine sẽ tạm thời đưa vào callback queue để chờ sau khi lệnh `console.log("c")` chạy xong, `setTimeout` nào có thời gian chờ ngắn hơn, sẽ được xếp trước vào callback queue. Vì vậy, kết quả lần lượt là: c
b
a

8. Khác nhau giữa `let`, `var` và `const`

=> `var` hỗ trợ hoisting, phạm vi sử dụng trong một function scope

=> `let` `const` không hỗ trợ hoisting, phạm vi sử dụng trong một block scope (là trong cặp ngoặc {} , không được khai báo lại trong cùng một scope)

9. Có cách nào thay đổi được giá trị của `const` ko? Tại sao?

=> Có thể nhưng cũng không thể, nếu trường hợp giá trị của `const` là kiểu tham trị thì không thể thay đổi giá trị của `const` là đối tượng nhưng ngược lại, n kiểu tham chiếu (array, object) thì chúng ta có thể thêm vào hoặc bớt đi phần tử bên trong. Nguyên nhân là khi chúng ta thay đổi giá trị bên trong array hay object của một biến `const`, việc đó chỉ góp phần thay đổi vùng nhớ của biến chứ không thay đổi giá trị của biến, bản chất biến `const` ta đang đề cập vẫn là một array hay một object.

10. Arrow function khác gì với function thường ?

=> Arrow function không làm thay đổi ngữ cảnh của con trỏ **this** và không bind arguments , không được hoist, không phù hợp làm cho method, không sử dụng làm hàm constructor, không có thuộc tính prototype, ngắn gọn hơn

11. ES6 offer các tính năng mới gì cho mình sử dụng, kể tên?

=> Arrow function, classes, promise, `let` & `const`, template string, Multi-line string, Default Parameters, Enhance Object Literal (Computed Property Key, Property value Shorthand, Method definition shorthand), Destructuring Assignment, modules -

12. Kể tên các higher order function trong ES6

=> `reduce`, `map`, `filter`, `foreach`

13. Tại sao lại gọi là High order function ?

=> function chứa tham số là một callback, hoặc function trả về một function khác (closure)

14. Spread operator là gì ?

15. Con trỏ **this** là gì? Scope là gì?

=> <https://viblo.asia/p/hieu-scope-va-context-trong-javascript-3P0lPArm5ox>
<https://viblo.asia/p/scope-trong-javascript-RQqKLnW6l7z>

16. Promise là gì? Async await là gì?

15. 4 tính chất của đối tượng?

=> Đóng gói: Encapsulation

Bảo vệ dữ liệu, bảo mật

=> Kế thừa: Inheritance

Tái sử dụng

1 class có kế thừa 1 class nhưng kế thừa được nhiều interface

=> Đa hình: Polymorphism

Overloading, overwriting

=> Trừu tượng: Abstraction

Abstract class và interface

16. Nếu có 3 promise đang chạy, nếu muốn chờ xong cả 3 mới làm thì dùng cách nào ?

=> **promise.all**

17. Thuật toán Bubble sort, quick sort chạy như thế nào?

18. Dự án trước làm về gì ? Trong dự án đó có gì e cảm nhận là khó nhất

19. JSON là gì?

=> JSON(JavaScript Object Notation) là định dạng trao đổi dữ liệu dưới dạng (chuỗi) lightweight data-interchange format.

=> tất cả ngôn ngữ đều có thể sử dụng

=> Thể hiện kiểu dữ liệu: Number, Boolean , Null,String , Array,Object

=> Mã hóa (Encode) và giải mã(decode) /

=> Stringify : Từ JS type => Json và Parse từ Json => JS type (Number, Boolean , Null,String , Array,Object ...)

20. AJAX là gì?

=> Asynchronous JS And XML

=> là công nghệ giúp chúng ta tạo ra những trang web động mà hoàn toàn không cần load lại trang nên rất mượt

=> được viết = ngôn ngữ JS chạy trên client, mỗi máy user chạy độc lập không ảnh hưởng tới nhau

21. Restful API là gì?

API : Application Programming Interface.

- Là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web để quản lý các resource(tệp văn bản, ảnh, âm thanh, video, dữ liệu động). được truyền tải dưới dạng http

22. JS ProtoType?

23. 24. bind(), call(), apply() . tìm hiểu về 3 hàm này

25. HTML DOM là gì?

- Khi một tài liệu HTML được tải vào trình duyệt web, nó sẽ được biên dịch ra thành một mô hình DOM

- **DOM** Là viết tắt của (Document Object Model), Là một chuẩn được định nghĩa bởi W3C(Tổ Chức Web Toàn Cầu – World Wide Web Consortium)•

-Có 3 thành phần chính là :

1. Elements.
2. Attribute.
3. Text.

26.JSON là gì?.

- **JSON** là viết tắt của **JavaScript Object Notation**
- Là một định dạng dữ liệu (dạng chuỗi).
- Là một kiểu định dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. **JSON** là một tiêu chuẩn mở để trao đổi dữ liệu trên web.
- Trong đó:
 1. Thể hiện kiểu chuỗi và key thì cần cho vào ngoặc kép
 2. Còn Number, Boolean, Null thì không cần cho vào ngoặc kép

27. Có bao nhiêu data type trong JS? vd:string, number, boolean....

28. Em hiểu gì về độ phức tạp của thuật toán? $O(1)$, $O(n)$, $O(n^2)$.

-----REACT-----

1. E hiểu gì về state , props. Giống và khác

=> **Props** là nơi lưu trữ dữ liệu được lưu truyền giữa các Component, giúp chúng tương tác và trao đổi dữ liệu với nhau. Được truyền vào Component cha và có thể truy cập được ở các Component con.

=> **State** cũng lưu trữ dữ liệu như Props, nhưng cách hoạt động hoàn toàn khác. State là một thành phần của Component và chỉ có thể bị thay đổi bởi chính Component chứa nó.

=> **Giống:** Cả state và props đều được sử dụng để lưu trữ dữ liệu trong Components.

=> **Khác:**

| State | Props |
|-------|-------|
|-------|-------|

| | |
|---|---|
| State là dữ liệu có thể bị thay đổi, nên sẽ được dùng để theo dõi sự thay đổi bên trong component và re-render Component. | Được thiết lập từ Component cha, truyền vào các Component con và cố định trong suốt vòng đời. Vì vậy, props được dùng trong các trường hợp cần trao đổi dữ liệu giữa các Component. |
|---|---|

2. React là thư viện hay framework? Thư viện và framework khác nhau ở điểm nào ? tham khảo

[:https://viblo.asia/p/su-khac-nhau-giua-framework-va-library-maGK7D6DZj2](https://viblo.asia/p/su-khac-nhau-giua-framework-va-library-maGK7D6DZj2)

React là thư viện

3.React là oneway binding hay two way binding?

=> Oneway vì khi đổi model thì view ko đổi, phải thay đổi state

=> One-way binding vì dữ liệu nó được truyền từ trên xuống, sự kiện được truyền từ dưới lên

4. Em hiểu gì về lifecycle của react?

=>

5. Khi thay đổi state,component render lại, có cách nào để khi thay đổi state mà component ko cần render lại ko?

=> dùng shouldComponentUpdate => return false

6. Sử dụng react hook thì làm sao để quản lý render?

=> Sử dụng.

7. Em đã từng sử dụng các thư viện quản lý state nào ?

=> Redux, mobx.

8. Cơ chế so sánh của redux?

=> Shallow equality checking (hay so sánh tham chiếu) là phép so sánh chỉ cần kiểm tra xem 2 biến được so sánh có cùng tham chiếu đến 1 object hay không, ngược lại với nó là thuật ngữ deep equality checking (hay so sánh giá trị) phải kiểm tra mỗi giá trị của từng thuộc tính ở cả 2 objects. Như vậy, chỉ cần 2 biến cùng tham chiếu đến 1 object thì Shallow equality checking sẽ trả về giá trị là true, ngược lại, nếu chúng không cùng tham chiếu đến 1 object thì Shallow equality checking sẽ trả về giá trị là false (dù cho 2 biến có giống hệt nhau đi nữa mà không cùng tham chiếu đến 1 object thì Shallow equality checking vẫn sẽ trả về giá trị là false)

=> Do đó shallow equality check đơn giản hơn và nhanh hơn deep equality checking =>

Redux sử dụng shallow equality check để cải thiện performance Ngoài ra, điểm đặc biệt của shallow equality checking đó là: nó không thể phát hiện được rằng 1 object có bị thay đổi hay

không sau khi được xử lý bởi 1 function nếu function đó thay đổi chính object được truyền vào cho nó

=> state trên reducer là dạng immutable Obj

9. Công dụng của middleware là gì?

=> middleware là lớp nằm giữa Reducers và Views. Vị trí mà Middleware hoạt động là trước khi Reducers nhận được Actions và sau khi 1 Action được dispatch(). Middleware trong Redux được biết đến nhiều nhất trong việc xử lý ASYNC Action - đó là những Action không sẵn sàng ngay khi 1 Action Creator được gọi tới, thông thường ở đây là các API request và handle Error nếu có.

10.componentWillUnmount chạy trước hay sau khi component bị hủy?

=> Chạy trước khi Component bị hủy

11.PureComponent và memo dùng để làm gì?

=> Để ngăn việc re-render không cần thiết.

12. Hai cái trên sử dụng kiểu so sánh nào ? Shallow hay deep ?

=> Shallow

13. Làm sao để so sánh deep comparison?

=> sử dụng useCallBack

14. Em đã sử dụng context API bao giờ chưa? context API truyền dữ liệu như nào? có thể truyền giữa các component anh em được không?

=> <https://freetuts.net/context-trong-reactjs-2431.html>

15. Một số cách để tối ưu ứng dụng reactjs

=> Sử dụng useCallBack, memo, useMemo để ngăn việc re-render không đáng có.

=> Tối ưu dung lượng ảnh, hạn chế sử dụng svg.

=> sử dụng lazyload.

16. Cho một code react, tìm lỗi sai?

=>

17. HOcs là gì? thường trong dự án e sử dụng khi nào?

1. những cách xử lý form
2. Loading
3. Array function
4. map hoạt động sao

5. redux: trên store sẽ có 1 cái initialValues chứa 1 state có giá trị khởi tạo là gì do chúng ta quy định, store gửi lên views 1 cái state để render ra giao diện, khi view có 1 hành động gì đó hay 1 sự kiện nào đó thì sẽ gửi 1 các actions lên store, action này là 1 object có 2 phần tử là Type và Payload thực hiện 1 logic nào đó mà chúng ta muốn, reducer sẽ nhận actions đó và cập nhật lại state mới trả về cho view để render lại.
6. axios interceptor: axios interceptors là lớp đứng giữa client và server dùng để handle 1 việc gì đó khi chúng ta gửi request vd như handleToken hay handleError lên cho server và từ server gửi response về chúng ta cũng có thể handle 1 việc gì đó vd như handleError hay format lại response (vd như res.data, res.data.content) thành res để sử dụng cho tất cả mọi nơi mà mình call api trả về.
7. css.
8. bất đồng bộ vs đồng bộ.
9. effects khi nào thì sử dụng: để tránh tình trạng bị rò rỉ bộ nhớ khi sử dụng hàm event hay setTimeout, vv, hay trong 1 số trường hợp khác cần clear bộ nhớ trước mà chúng ta xử lý logic sử dụng bộ nhớ để lưu giữ giá trị.

The screenshot shows a web browser with a course page on the left and a video player on the right.

Course Page (Left):

- Tổng quan react
- 1/ Tổng quan reactjs
- 2/ Tìm hiểu cấu trúc dự án
- 3/ Tạo React Functional Component
- 4/ Tạo React Class Component
- 5/ Làm bài tập tạo Component React
- 6/ Databinding React
- 7/ Event react khái niệm về closure function
- 8/ Cấu trúc điều khiển If else
- 9/ Làm việc với thuộc tính State, phương thức SetState & ReRender
- 10/ Ôn tập Event State Style React
- 11/ Vòng lặp trong react phần 1

Video Player (Right):

25/ REACT GIỚI THIỆU REDUX

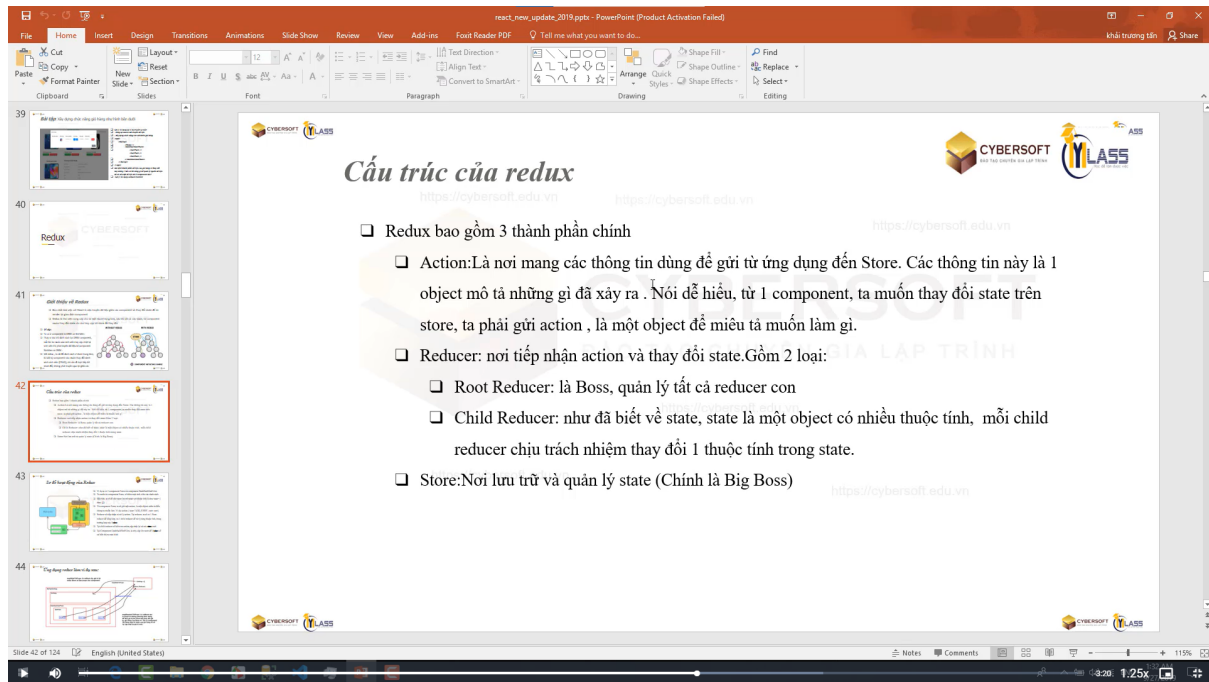
Giới thiệu về Redux

- Bản chất làm việc với React là việc truyền dữ liệu giữa các component và thay đổi state để re-render lại giao diện component
- Redux là thư viện cung cấp cho ta một store trung tâm, lưu trữ tất cả các state, từ component muốn thay đổi state chỉ cần truy cập tới store để thay đổi

Ví dụ:

- Ta có 2 component là DSSV và SinhVien
- Thay vì lưu trữ danh sách tại DSSV component, mỗi lần ta muốn xóa sinh viên hay cập nhật lại sinh viên thì phải truyền dữ liệu từ component SinhVien ra DSSV.
- Với redux, ta sẽ để danh sách ở store trung tâm, từ bất kỳ component nào muốn thay đổi danh sách sinh viên (CRUD), chỉ cần đi trực tiếp tới store để, không phải truyền qua lại giữa các

The video player shows a diagram comparing "WITHOUT REDUX" and "WITH REDUX". In "WITHOUT REDUX", data is passed between components through props. In "WITH REDUX", data is stored in a central "STORE" and components interact with it directly.



=> Sử dụng khi cần wrap các Screen trong một Layout chung như có chung Header, Footer, Side drawer.

18. Khi sử dụng các hook của react như useCallback, useEffect, ta hay truyền 2 tham số, callback function và 1 mảng dependencies, sự khác biệt của truyền mảng rỗng và không truyền là gì?

=> Truyền mảng rỗng tương ứng với việc hook chỉ chạy một lần và không bao giờ chạy lại trong suốt vòng đời của Component.

19. Ưu và nhược điểm của single page application so với multiple page application?

=> SPA không cần phải load lại trang nhiều lần, tăng trải nghiệm người dùng. Nhược điểm: Giảm SEO.

20. Cho lần lượt các api 1 2 3 4 5 => muốn api 1 2 3 4 cùng thực thi và thực thi xong thì api 5 mới được gọi thì dùng Promise.all.

```
var p1 = new Promise((resolve, reject) => {
  setTimeout(resolve, 1000, "one");
});

var p2 = new Promise((resolve, reject) => {
  setTimeout(resolve, 2000, "two");
});
```

```

});
var p3 = new Promise((resolve, reject) => {
  setTimeout(resolve, 3000, "three");

});
var p4 = new Promise((resolve, reject) => {
  setTimeout(resolve, 4000, "four");
});

```

`Promise.all([p1, p2, p3, p4]).then(values => { //=<nb xong thì mới thực hiện gọi api 5 thực thi hoặc trong trường hợp có bất kỳ api nào bị lỗi cũng sẽ dừng`

```

  console.log(values);
  console.log('request api 5');

}, reason => {
  console.log(reason)
});

```

21. Pure component là gì?

Pure component dùng để ngăn việc render lại những component không cần thiết render lại. những component k chứa props là state thì k cần render lại vd như footer header

22. So sánh sự khác biệt giữa state và props

23. Vì sao không nên lạm dụng useMemo

Vì khi sử dụng useMemo sẽ tốn thêm dung lượng trong bộ nhớ để lưu kết quả của expensive function trả về và làm tăng kích thước bộ nhớ chiếm nhiều bộ nhớ trong máy tính

24. Khi nào sử dụng useMemo?

- Khi truyền một tham số đầu vào chúng trả ra kết quả giống nhau và function trả về tốn nhiều thời gian để tính toán thì nên sử dụng

25. Phân biệt giữa file .jsx và .js

Trong hầu hết các trường hợp, chỉ cần bộ chuyển đổi / gói, có thể không được cấu hình để hoạt động với các tệp JSX, nhưng với JS! Vì vậy, bạn buộc phải sử dụng các tệp JS thay vì JSX.

Và vì React chỉ là một thư viện cho javascript, nên bạn không có sự khác biệt nào khi lựa chọn giữa JSX hoặc JS. Chúng hoàn toàn có thể hoán đổi cho nhau!

Trong một số trường hợp, người dùng / nhà phát triển cũng có thể chọn JSX thay vì JS, vì làm nổi bật mã, nhưng hầu hết các trình soạn thảo mới hơn cũng đang xem chính xác cú pháp phản ứng trong các tệp JS.

26. UseRef để lưu giá trị trước đó của state. Lưu các cái tham chiếu bên ngoài function component

27. Class và Object khác nhau ?

27. Khi nào sử dụng Class và Function

=> Khi chúng ta xài phiên bản cũ trước 16.8 và xử lý state, logic phức tạp thì xài class

28. Use nào để DOM đến element

29. Session Storage

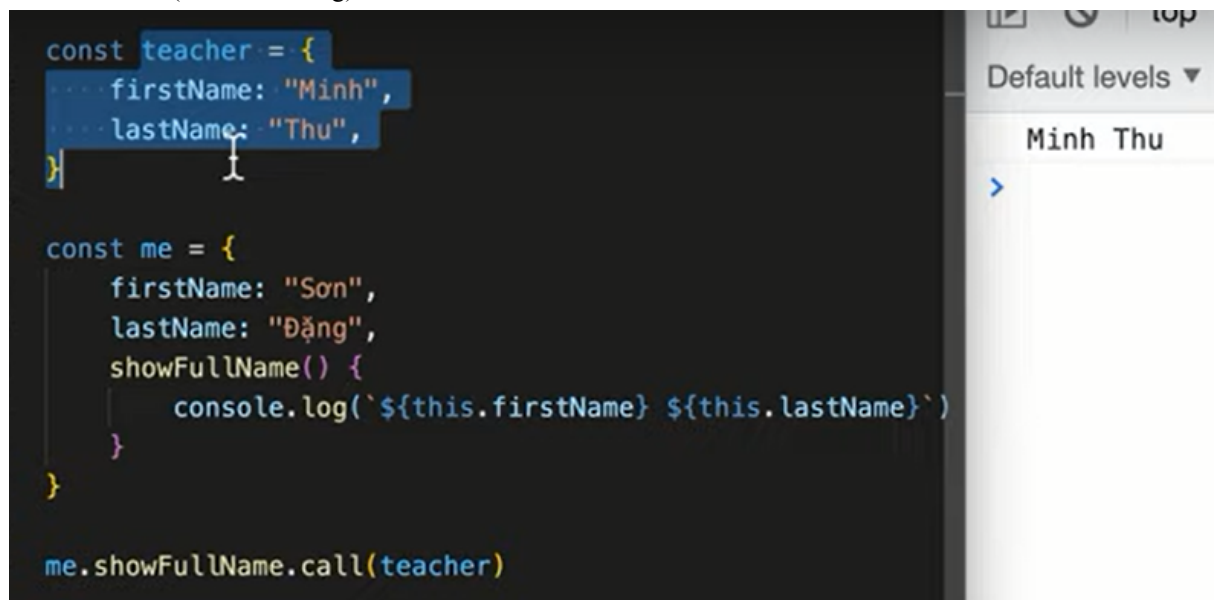
30. Call() và apply()

- Call() là một method trong prototype của Func Constructor, thường dùng để gọi hàm hoặc bind(this) cho hàm.
Ứng dụng:
 - Gọi hàm với call method.
 - Thể hiện tính kế thừa trong OOP
 - Mượn hàm với (func borrowing) VD:

```
const teacher = {
  firstName: "Minh",
  lastName: "Thu",
}

const me = {
  firstName: "Sơn",
  lastName: "Đặng",
  showFullName() {
    console.log(`${this.firstName} ${this.lastName}`)
  }
}

me.showFullName.call(teacher)
```



- Syntax: `call(thisArg, arg1, ... , argN)`

```
1 function Product(name, price) {
2   this.name = name;
3   this.price = price;
4 }
5
6 function Food(name, price) {
7   Product.call(this, name, price);
8   this.category = 'food';
9 }
10
11 console.log(new Food('cheese', 5).name);
12 // expected output: "cheese"
13
```

31. Sự khác biệt giữa useMemo và useCallback

- `useMemo` giữ cho một hàm không được thực thi lại nếu nó không nhận được một tập hợp các tham số đã được sử dụng trước đó. Nó sẽ trả về kết quả của một function. Sử dụng nó khi bạn muốn ngăn một số thao tác nặng hoặc tốn kém tài nguyên được gọi trên mỗi lần render.
- `useCallback` giữ cho một hàm không được tạo lại lần nữa, dựa trên mảng các phần phụ thuộc. Nó sẽ trả về chính function đó. Sử dụng nó khi mà bạn muốn truyền function vào component con và chặn không cho một hàm nào đó tiêu thời gian, tài nguyên phải tạo lại.

32. UseCallback và Promise

33. Khi chưa có class JS dùng gì để tạo

34. Tại sao ngta sử dụng React ?

- Vì nó có chai làm nhiều component có thể tái sử dụng và gom lại các thành phần thành component khác
- Vì nó xử lý trên DOM ảo nên tốc độ nhanh

35. window và document khác nhau điểm nào?.

36. indexOf() ?

- Array **indexOf** trong javascript là một phương thức của đối tượng mảng, nó dùng để tìm kiếm vị trí một phần tử trong mảng dựa vào giá trị truyền vào tham số của hàm.

Cú pháp : `array.indexOf(item, start)`.

- `item` là giá trị của phần tử cần tìm.
- `start` là vị trí bắt đầu tìm kiếm.

37. Web apps # Website ?