

Javascript

1. Ce este un closure?

Closure este combinația dintre o funcție și lexical environment în care a fost declarată acea funcție. Practic un closure este atunci când o funcție internă are acces la variabilele și proprietățile funcției sale externe.

2. Un exemplu de closure folosit din JS?

Am impresia că este `setTimeout()`.

Alt exemplu:

```
function makeAdder(x){  
    return function(y) {  
        return x+y; };  
}  
  
var add5 = makeAdder(5);  
var add10 = makeAdder(10);  
  
console.log(add5(2)); → result 7  
console.log(add10(2)); → result 12
```

3. Lexical scope

Este atunci când o variabilă care este definită în afara funcției (domeniului de aplicare), este automat inclusă/accesibilă în funcție (scope), dacă funcția este definită după declararea variabilei.

În iar sensul opus, adică o variabilă definită în interiorul funcției nu va fi accesibilă în exteriorul ei.

4. Ce este Hoisting

...

...

...

5. Ce este obj. destructuring

Ne ajuta sa extragem valorile din fiecare parametru si sa le stocam in variabile separate mult mai simplu si fara a fi repetitive.

```
const address = {
  street: '',
  city: '',
  country: ''
};

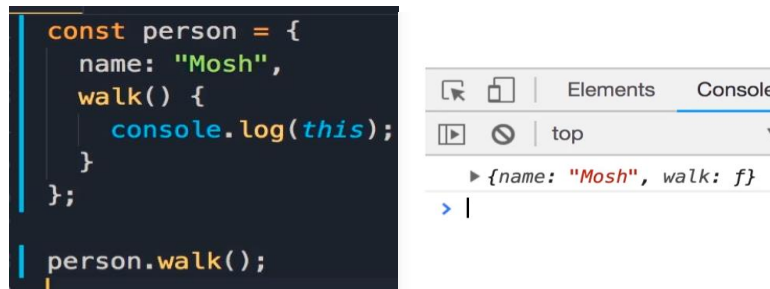
const street = address.street;
const city = address.city;
const country = address.country;

const { street: st } = address;
```

6. Ce este this?

Valoare lui **THIS** este determinata dupa modul cum o functie este apelata. Daca apelam o functie ca o metoda intr-un obiect, va returna mereu o referinta a acelui obiect. (Fig. 1)

Daca apelam o functie ca stand alone object sau in afara obiectului, aceasta va returna mereu obiectul global, care e window object in browsere. (Fig. 2)

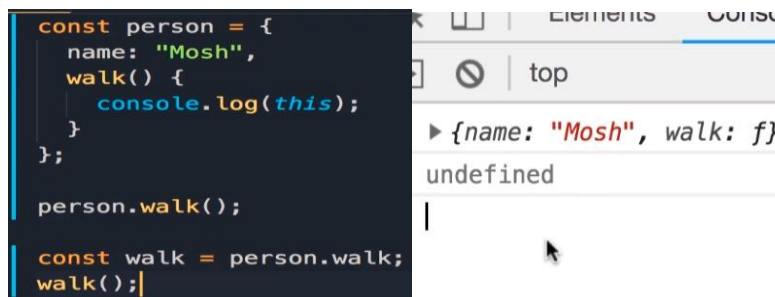


```
const person = {
  name: "Mosh",
  walk() {
    console.log(this);
  }
};

person.walk();
```

The browser console shows the output of the `console.log(this);` call as `{name: "Mosh", walk: f}`, indicating that `this` refers to the `person` object.

Fig. 1. **THIS** returneaza o referinta a acelui obiect (person)



```
const person = {
  name: "Mosh",
  walk() {
    console.log(this);
  }
};

person.walk();

const walk = person.walk;
walk();
```

The browser console shows the output of the `console.log(this);` call as `undefined`, indicating that `this` refers to the global object (window) when the function is called as a standalone function.

Fig. 2. **THIS** returneaza global object (window object)

In JavaScript va arata obiectul global, dar in React.js va arata undefined deoarece React are `strict mode` = True.

Cu metoda **bind** putem seta valoare lui **THIS** permanent. Setam bind la acea functie (walk) si obtinem o noua functie (walk) in care valoare se bazeaza pe parametrul setat in metoda bind. Fig. 3

7. Conceptul de prototype

Din cate stiu se folosea in versiunile mai vechi de JS, pe cand nu exista conceptul de class si ca sa scriem codul sub forma de class foloseam prototype.

8. Care este diferenta dintre map – filter – reduce ?

- **filter()** este practic un filtru caruia ii dam o conditie si care returneaza un array nou cu elementele care trec de filtrare.
- **map()** itereaza prin valorile array-ului si atribuie fiecarui element functia pe care i-o dam si returneaza un array nou cu elementele.
- **reduce()** are 2 parametrii, aceia fiind accumulator si currentValue. Itereaza prin elemente la fel si returneaza ce vrem noi, adica putem pune sa returneze elementele intr-un obj sau array.

9. Diferenta dintre Arrow function si Normal function?

Pai arrow function nu are constructor si este o functie anonima, iar functia normala dupa ce a fost declarata o sa fie automat dusa sus si apoi vor fi luate restul.

Functia normala poate fi instantiata, iar arrow function nu poate fi din cauza hoisting-ului.

10. Promise

Stiu ca promise-ul poate avea 3 stari.

- **rejected** care inseamna ca operatia a dat fail si aici dam un mesaj de eroare
- **pending** care e starea initiala, adica nu e nici rejected, nici fulfilled
- **fulfilled/resolved** adica operatia a avut succes si se pot aplica actiuni async await

In principal luam un request cu fetch, apoi transformam datele, de ex: daca e un JSON in transformam in asa fel incat sa putem lua datele cu async await si apoi putem folosi datele.

11. Prin ce moduri poti face handle la erori?

Un mod ar fi **Try and Catch**.

12. Conceptul de immutable data

Pai ca sa nu lucram cu array-ul in sine, adica sa ii alteram datele. Facem un array nou in care copiem array-ul principal. (practice facem o copie)

Selectorii de acces: private, public, protected.

13. Copierea unui object in alt object

- folosim spread operator
- object.assign
- iterem prin proprietatile primului object si le punem in al 2-lea

14. Diferenta dintre TypeScript si JavaScript

TypeScript e un strong type, adica variabilele trebuie sa le initializam si trebuie sa le definim cu un type sau cu any. Iar TypeScript e mai strict decat JavaScript.

15. Unde salvam datele ca sa ramana in memorie?

Pai putem in local storage (inainte sa manipulam datele in front-end le luam cu Get si Set) sau o baza de date.

16. Ce inseamna pure function ?

Practic e asemanator cu un dummy component, adica daca avem o functie si orice parametru ar primi ea returneaza acelasi tip de data, adica pentru acelasi input returneaza acelasi output. (primeste string returneaza string, primeste numar returneaza numar).

17. Ce e un dummy component ?

E o componenta simpla care are doar un html ce afiseaza ceva, un numar, un string, fara ca componenta sa aiba prea multa logica.

18. Diferenta dintre var si let ?

O variabila depoziteaza date temporare in memoria computerului.

Cu **var** se definesc variabile globale, adica variabile care se pot modifica in alte sectiuni din script sau locale, variabile in corpul functiilor.

Iar **let** permite definirea de variabile care sunt limitate la sectiune, adica se pot modifica sau apela doar in expresia unde au fost definite/declarat.

19. Diferenta dintre let si const ?

Cu **const** declaram variabilele pe care nu le mai schimbam valoarea pe parcursul codului. Iar cu **let** declaram variabilele care trebuie sa-si schimbe valoarea pe parcurs.

20. Diferenta dintre Factory function si Constructor function

Pai prima diferenta ar fi notatia, **Factory function** se foloseste **Camel notation** (ex: oneTwoThreeFour), iar in Constructor function se foloseste Pascal notation (ex: OneTwoThreeFour).

Alta diferenta ar fi ca **Factory function** doar o apelam si aceasta functie va returna un object nou. Pe cand la **Constructor function** folosim operatorul **new**, care creaza un object gol si seteaza **this-ul** sa se duca la object-ul nou creat, si in loc sa returnam un object folosim **this-ul**.

21. Diferenta dintre == si ===

== adica loose equality trebuie sa aiba doar valoarea egala, iar tipul variabilei se va converti automat

=== adica strict equality trebuie sa aiba acelasi tip si aceasi valoare ca sa fie egale.

22. Ce este Conditional (ternary) operator ?

Practic este o scurtatura a statement-ului **if**.

23. Diferenta dintre For.. in loop si For.. of loop

Pai **For.. in loop** il folosim ca sa iteram printr-un object, pe cand **For.. of loop** il folosim ca sa iteram printr-un array.

24. Diferenta dintre forEach() si map()

- Pai **forEach** face cam aceasi chestie ca si **map()**, doar ca nu returneaza nimic (ex: salvarea elementelor in baza de date).

- Pe cand **map()** returneaza ceva (ex: transformarea unei liste de siruri in majuscule).

25. Ce este un object?

Obiectul este o entitate/tip de data in JavaScript. Obiectul are o serie de proprietati ce pot fi modificate.

Un **object** este ca un object din viata reala, adica un exemplu ar fi o persoana, are nume, varsta, etc, fiind alcatuit din variabilele caracteristice lui.

Pentru a schimba valoarea unei variabile din object folosim: Dot notation sau bracket notation.

Dot notation - `person.name = "Alin";`

Bracket notation - `person['name'] = "Alin";`

26. Ce este un Array ?

Un **Array** este o structura care reprezinta o lista de iteme/obiecte. (Listele de objects se depoziteaza intr-un Array).

Fiecare element din array are un index, care semnifica pozitia elementului. (incepand cu indexul 0).

27. Ce este o functie ?

Este unul din blocurile fundamentale din Javascript. Adica este un statement care face un task sau calculeaza o valoare.