**JS Questions**

Contents

[**1.** **Tell about yourself, experience** 2](#_Toc189845519)

[**2.** **What are the possible ways to create objects in JavaScript** 3](#_Toc189845520)

[**3.** **What is a prototype chain** 3](#_Toc189845521)

[**4.** **What is JSON** 3](#_Toc189845522)

[**5.** **compare Object and Map** 3](#_Toc189845523)

[**6.** **compare Array and Set** 3](#_Toc189845524)

[**7.** **arrow functions** 4](#_Toc189845525)

[**8.** **What is the Temporal Dead Zone** 4](#_Toc189845526)

[**9.** **differences between cookie, local storage and session storage** 4](#_Toc189845527)

[**10.** **What is the difference between null and undefined** 4](#_Toc189845528)

[**11.** **What is the difference between window and document** 4](#_Toc189845529)

[**12.** **What is event bubbling** 4](#_Toc189845530)

[**13.** **IIFE(Immediately Invoked Function Expression)** 4](#_Toc189845531)

[**14.** **Imperative vs Declarative programming** 4](#_Toc189845532)

[**15.** **Factory function vs Constructor function vs Class** 5](#_Toc189845533)

1. **Tell about yourself, experience**
2. thank you for giving me this opportunity to introduce myself. My name is Tan and currently I am a Front End developer. I love to create beautiful interfaces and create good experience websites for user . I not only work on web application on browsers but also on the mobile app. I can do the jobs individually and work with the partners as well. I enjoy learning the new things ready to share what I learn to other member in team. I hope that I can be a good member in your company if I have a chance. so That's all about me! thank you
3. Strength
   * I am a **self-learn person** so I can work on my own,
   * I have a **positive attitude when Facing the problems**.
   * I love to learn new things and enjoy to share the knowledge to other people
4. Weakness
   * I think **my communication skill** is not very good to other people. sometimes, I wanna show my ideas to other but when I told them the ideas, I could not explain clearly so they misunderstood the ideas. Therefore, I am trying to improve my communication skills better and better.
   * The 2nd weakness is about **work-life balance**. sometimes, I stay up late to finish the tasks. I know this is not good for health so I try to arrange the time and tasks to balance my daily life.
5. WHY QUIT THE PREVIOUS JOB
   * because in my previous company, they **changed the roadmap** in this year, especially the **roadmap of Front End** team. so I realized that the roadmap was **not suitable** for me anymore, Moreover, the director **didn't care about the FrontEnd team**, he just only focus the BackEnd and designer team. that's the reason I decided to leave the previous job
6. tell about the project most proud of and your role is that project

* the project I proud of and maybe it was the most difficult project is Customer Tickets, in that module. I was a front end engineer. And my mission is optimize the module build time from 30 minutes to 10 minutes.
* At the first time, I failed because there are many errors that I didn’t know how to fix them.
* Then at the 2nd time, I failed again. Even though I fixed the previous bugs, the build time was not optimise
* Finally at the third time, I success. I used some new technique such as react lazy, and the important thing was that I had to change to webpack config file to optimise the build time. After that, I test the whole module again, and I need my partner help me to test.
* As a result, I build the new optimise code to production and it run correctly, no errors, no bugs.

1. why want to work this company

* work in new area, new environment, learn new things

1. see yourself in 5 years in term of career path

* I will be a team leader

1. most factor for you for looking for job

* factor is the job features: is the new job suitable with me, what can I learn from this job

1. technical problems and how to solve them?

* In BMS, at first we used nested loop, nested array to store data and update data. Because of this, the website so laggy and so slow. Therefore, I research a new technique and found that the hash map, and I implement the hash map by using object in javascript to store and update data. As a result, the website is better and faster than before.

1. Have you ever failed in technical and you could not solve that problem, how to recover it?

* Yeah, I used to integrate CkEditor5 in module and it didn’t work. Even though in demo on documations is works fine , but in my project it didn’t work. So I did it again, and this time I did own my custom setting and at that time it works correctly.

1. Tell about the partners you hate and how to collaborate with them?
2. Dealing with rush deadline (use STAR: situation – task – action – result)
3. **What are the possible ways to create objects in JavaScript**

* Object literal

var object = {

name: "Sudheer",

age: 34

};

* New Object()
* Object.create(null)

1. **What is a prototype chain**

* Dùng để tạo object mới dựa trên các thuộc tính của một object có sẵn. Giống inheritance (kế thừa) OOP

difference between Call, Apply and Bind

call(arg1, arg2,arg3,...): gọi function với this dc gán cho đối số đầu tiên. Các đối số thứ 2 trở đi là dc truyền vào hàm gọi call().

apply(arg1, arg2): gọi function với this dc gán cho đối số đầu tiên. đối số thứ 2 phải là array và dc truyền vào hàm gọi call().

bind(arg): gọi function với this dc gán cho đối số đầu tiên. Ko có đối số thứ 2.

1. **What is JSON**

Text-based data. Use to transmit data through network if native data is big.

Note: JSON.parse() doesn’t parse value of undefined.

1. **compare Object and Map**

|  |  |  |
| --- | --- | --- |
|  | Object | Map |
| Keys | Must be string or symbol (JavaScript automatically converts keys to a string a value.) | * Any value * ordered |
| size property | No | yes |
| Iterable | No. Use strict like Object.keys | yes |
| Prototype | Yes | No |
| (other) |  | better addition or removal of key pairs |

1. **compare Array and Set**

|  |  |  |
| --- | --- | --- |
|  | Array | Set |
| Index | Yes (in order) | no |
| Present | [1,2,3,4,5,6] | {1,2,3,4,5,6} |
| Duplicate Item | Yes [1,1,2,2,3] | No (distinct) {1,2,3} |
| QueryBy | Index | key |
| Constructor | [1,2,3,4,5], Array.from(), new Array() | new Set([1,2,3,4,5]) |
|  |  |  |

1. **arrow functions**

* shorter syntax for a function expression and does **not have its own this, arguments, super, or new.target**. So, not used to contructor or method in class or object

1. **What is the Temporal Dead Zone**

* that occurs when declaring a variable with the let and const
* The time span when that happens, between the creation of a **variable’s binding** and **its declaration**, is called the temporal dead zone

1. **differences between cookie, local storage and session storage**

|  |  |  |  |
| --- | --- | --- | --- |
|  | cookie | local storage | session storage |
| Accessed | Client and server | Client | Client |
| Lifetime | Manual | Manual | Until tab is closed |
| Data size | 4kb | 5MB | 5MB |
| Sent with request | Yes | No | No |

1. **What is the difference between null and undefined**

|  |  |  |
| --- | --- | --- |
|  | null | undefined |
| Value | Dc khai bào và dc gán giá trị là null ko chỏ đến obj nào | Dc khai báo nhưng Ko dc gán giá trị |
| Type | object | undefined |
| Operations (+ - \* /) | Convert to 0 | Convert to NaN |

1. **What is the difference between window and document**

|  |  |
| --- | --- |
| window | document |
| Root element | Direct child of window Document Object Model(DOM) |
| Mặc định, có thể gọi mà ko cần từ khoá window ở trước | Phải thông qua **window.document** or **document.** |
| Có các phương thức, thuộc tính như alert(), confirm() and conlog(), document, location | Có các phương thức như getElementById, getElementsByTagName, createElement.  Phải thêm **document** trước các phương thức này |

Note: những methods này do webAPI cung cấp, ko phải do JS

1. **What is event bubbling**

* Events trigger from the **child element** to the **parent element (in to out)**

What is event capturing

* Events trigger from the **parent element** to the **child element (out to in)**

1. **IIFE(Immediately Invoked Function Expression)**

* Private data, variables
* Use as module because we can not access variables from outside

1. **Imperative vs Declarative programming**

|  |  |
| --- | --- |
| Imperative | Declarative |
| * Provides a step-by-step description of what the computer should do: * tạo ra các câu lệnh để computer chạy theo ý mún của mình | * focuses on what the desired outcome is. * Đã được abstraction rồi, nên ko cần quan tâm bên trong chạy như nào * Chi tiết đã được dấu đi |
| * Dài dòng, nhưng rõ ràng | * Ngắn gọn nhưng trừu tượng hóa |
| * Expressiveness | * Abstraction |

1. **Factory function vs Constructor function vs Class**

// factory Function

    function person(*name*) {

        return {

            name: *name*,

            sayHi() {

                console.log(`${*name*} says Hi`)

            }

        }

    }

    const Bob = person('Bob');

    const Alice = person('Alice');

    Bob.sayHi() //  Bob says Hi

    Alice.sayHi() // Alice says Hi

    console.log(Bob.sayHi === Alice.sayHi) // false

**factory function** return 1 object. Mình có thể assign cho nhiều biến khác nhau.

Nghĩa là **define 1 function** nhưng dc sử dụng **nhiều lần**

* **ưu điểm**: ko cần quan tâm đến context, ngắn gọn, ko dùng từ khóa new
* **nhược điểm**: mỗi lần gọi sẽ tạo ra 1 object riêng biệt và chiếm 1 vùng nhớ.

 // Constructor function

    function Person(*name*) {

        this.name = *name*;

        this.sayHi = function () {

            console.log(`${this.name} says Hi`)

        }

    }

    const Bob = new Person('Bob'); // object instantiation

    const Lee = new Person('Lee');

    // \* keyword 'new': create {} => bind this to the object => return this object

    Bob.sayHi() // Bob says Hi

    Lee.sayHi() // Lee says Hi

    console.log(Bob.sayHi === Lee.sayHi) // false

    // phải dùng keyword 'this'

class Person {

        constructor(*name*) {

            this.name = *name*;

        }

        sayHi = function () {

            console.log(`${this.name} says Hi from Class`)

        }

        sayBye() {

            console.log(`${this.name} says Bye from Class`)

        }

    }

    const Bob = new Person('Bob'); // object instantiation

    const Lee = new Person('Lee');

    Bob.sayHi() // Bob says Hi

    Lee.sayBye() // Lee says Hi

    console.log(Bob.sayHi === Lee.sayHi) // false

    console.log(Bob.sayBye === Lee.sayBye) // true