**SOFT SKILLS QUESTIONS**

**Tell me about the most recent project you worked on. What were your responsibilities?**

Preferred answer includes a brief description of the project, rundown of technologies used, describing if the project was done in a team or alone and then expanding on the features of the project and the tech stack.

**How do you keep your technology skills current?**

Preferred answer includes a reflection on a recent tech book you read or are currently reading, being an active member on Stack Overflow. Talking about the dev meet-up that you visited or that you attend regularly or talking about the platform you use to level up your skills such as Frontend Masters, Egghead.io or other tutorials, doing katas daily.

**Describe the type of work environment in which you are most productive.**

Preferred answer would be talking about the preferred technical setting (additional screen, favorite code editor, linter, npm package), working space and valuing precisely defined hours providing uninterrupted blocks of time, which shows that person strives to being productive.

**TECHNICAL INTERVIEW QUESTIONS**

**Difference between** [“](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/)**[==](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/)**[”](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/) **[and](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/)** [“](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/)**[===](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/)**[”](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/) **?**

* ” ==” only compares values “===” compares both values and type.

**Explain how this works in JavaScript?**

1. If the new keyword is used when calling the function, this inside the function is a brand new object.
2. If apply, call, or bind are used to call/create a function, this inside the function is the object that is passed in as the argument.
3. If a function is called as a method, such as obj.method() — this is the object that the function is a property of.
4. If a function is invoked as a free function invocation, meaning it was invoked without any of the conditions present above, this is the global object.
5. If the function is an ES2015 arrow function, it ignores all the rules above and receives the this value of its surrounding scope at the time it is created.

### **Explain how prototypal inheritance works**

All JavaScript objects have a prototype property, that is a reference to another object.

When a property is accessed on an object and if the property is not found on that object, the JavaScript engine looks at the object's prototype, and the prototype's prototype and so on, until it finds the property defined on one of the prototypes or until it reaches the end of the prototype chain

## **How to empty an array in JavaScript?**

const arrayList = ['a', 'b', 'c', 'd', 'e', 'f']

#### **Method 1**

arrayList = [];

#### 

#### **Method 2**

arrayList.length = 0;

#### 

#### **Method 3**

arrayList.splice(0, arrayList.length);

#### **Method 4**

while(arrayList.length) {

arrayList.pop();

}

### **What is a closure, and how/why would you use one?**

When a function returns the other function the returning function will hold its environment and this is known as closure.

Closures are functions that have access to the outer (enclosing) function's variables—scope chain even after the outer function has returned.

A closure gives a function access to variables that would be unavailable to it without the closure.

Why would you use one?

* Data privacy / private methods with closures. Common in the [module pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatternjavascript).
* [Partial applications or currying](https://medium.com/javascript-scene/curry-or-partial-application-8150044c78b8#.l4b6l1i3x).

**Write a simple closure?**

function outter () {

var secret = 'abc123';

return function () {

return secret;

}

}

inner = outter();

inner();

## **Write a *multiply* function which will work properly when invoked with following syntax.**

console.log(multiply(2)(3)(4)); // output : 24

console.log(multiply(4)(3)(4)); // output : 48

### **Answer**

Below is the code followed by the explanation of how it works:

function multiply (a) {

return function (b) { // anonymous function

return function (c) { // anonymous function

return a \* b \* c;

};

};

}

### **What would be the output of following code?**

var strA = "hi there";

var strB = strA;

strB="bye there!";

console.log (strA)

The output will 'hi there' because we're dealing with strings here. Strings are passed by value, that is, copied.

### **What would be the output of following code?**

var objA = {prop1: 42};

var objB = objA;

objB.prop1 = 90;

console.log(objA)

The output will {prop1: 90} because we're dealing with objects here. Objects are passed by reference, that is, objA and objB point to the same object in memory.

BONUS

### **What would be the output of following code?**

for (var i = 1; i < 5; i++) {

setTimeout( () => console.log(i), 1000)

}

**ANSWER:** The output will 5, 5, 5, 5 .

**Why?**

The for loop executes first, then it looks for the i value, which is 5, and then outputs four times, one for each loop iteration.

**How can we ensure to get each distinct i value for each function call?**

**Create a function that is invoked for each iteration of the loop and receives distinct value:**

function doSetTimeout(i) {

setTimeout(function() { console.log(i); }, 100);

}

for (var i = 1; i < 5; ++i) {

doSetTimeout(i);

}

**Use the ES6 let variable**

for (let i = 1; i < 5; ++i) {

setTimeout( () => console.log(i), 1000)

}