JavaScript Interview Questions for Freshers

Let's discuss some common questions that you should prepare for the interviews. These questions will be helpful in clearing the interviews specially for the frontend development role.

1. What are the differences between Java and JavaScript?

JavaScript is a client-side scripting language and Java is object Oriented Programming language. Both of them are totally different from each other.

- **JavaScript:** It is a light-weighted programming language ("scripting language") for developing interactive web pages. It can insert dynamic text into the HTML elements. JavaScript is also known as the browser's language.
- **Java:** Java is one of the most popular programming languages. It is an object-oriented programming language and has a virtual machine platform that allows you to create compiled programs that run on nearly every platform. Java promised, "Write Once, Run Anywhere".

2. What are JavaScript Data Types?

There are three major Data types in JavaScript.

- Primitive
  - Numbers
  - Strings
  - Boolean
  - Symbol
- Trivial
  - Undefined
  - Null
- Composite
  - Objects
  - Functions
  - Arrays

3. Which symbol is used for comments in JavaScript?

Comments prevent the execution of statements. Comments are ignored while the compiler executes the code. There are two type of symbols to represent comments in JavaScript:

- **Double slash:** It is known as a single-line comment.

```
// Single line comment
```

- **Slash with Asterisk:** It is known as a multi-line comment.

```
/*
Multi-line comments
...
*/
```

4. What would be the result of 3+2+”7"?

Here, 3 and 2 behave like an integer, and “7” behaves like a string. So 3 plus 2 will be 5. Then the output will be 5+”7" = 57.

5. What is the use of the isNaN function?

The number isNan function determines whether the passed value is NaN (Not a number) and is of the type “Number”. In JavaScript, the value NaN is considered a type of number. It returns true if the argument is not a number, else it returns false.

6. Which is faster in JavaScript and ASP script?

JavaScript is faster compared to ASP Script. JavaScript is a client-side scripting language and does not depend on the server to execute. The ASP script is a server-side scripting language always dependable on the server.

7. What is negative infinity?

The negative infinity is a constant value represents the lowest available value. It means that no other number is lesser than this value. It can be generate using a self-made function or by an arithmetic operation. JavaScript shows the NEGATIVE_INFINITY value as -Infinity.

8. Is it possible to break JavaScript Code into several lines?

Yes, it is possible to break the JavaScript code into several lines in a string statement. It can be broken by using the **backslash n ‘\n’**.

For example:

```
console.log("A Online Computer Science Portal\n for Geeks")
```

The code-breaking line is avoid by JavaScript which is not preferable.

```
let gfg= 10, GFG = 5,
Geeks =
gfg + GFG;
```

9. Which company developed JavaScript?

Netscape developed JavaScript and was created by Brenden Eich in the year of 1995.

10. What are undeclared and undefined variables?

- **Undefined:** It occurs when a variable is declare not not assign any value. Undefined is not a keyword.
- **Undeclared:** It occurs when we try to access any variable which is not initialize or declare earlier using the var or const keyword. If we use ‘typeof’ operator to get the value of an undeclare variable, we will face the runtime error with the return value as “undefined”. The scope of the undeclare variables is always global.

11. Write a JavaScript code for adding new elements dynamically.

html

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <title>Document</title>
</head>

<body>
    <button onclick="create()">
        Click Here!
    </button>

    <script>
        function create() {
            let geeks = document.createElement('geeks');
            geeks.textContent = "Geeksforgeeks";
            geeks.setAttribute('class', 'note');
            document.body.appendChild(geeks);
        }
    </script>
</body>
</html>
```

Output:

Video Player

00:00
00:03

12. What are global variables? How are these variables declared, and what are the problems associated with them?

In contrast, global variables are the variables that define outside of functions. These variables have a global scope, so they can be used by any function without passing them to the function as parameters.

Example:

javascript

```javascript
let petName = "Rocky"; // Global Variable
myFunction();

function myFunction() {
    console.log("Inside myFunction - Type of petName:", typeof petName);
    console.log("Inside myFunction - petName:", petName);
}

console.log("Outside myFunction - Type of petName:", typeof petName);
console.log("Outside myFunction - petName:", petName);
```

Output

```
Inside myFunction - Type of petName: string
Inside myFunction - petName: Rocky
Outside myFunction - Type of petName: string
Outside myFunction - petName: Rocky
```

It is difficult to debug and test the code that relies on global variables.

13. What do you mean by NULL in JavaScript?

The NULL value represents that no value or no object. It is known as empty value/object.

14. How to delete property-specific values?

The **delete keyword** deletes the whole property and all the values at once like

```
let gfg={Course: "DSA", Duration:30};
delete gfg.Course;
```

15. What is a prompt box?

The prompt box is a dialog box with an optional message prompting the user to input some text. It is often used if the user wants to input a value before entering a page. It returns a string containing the text entered by the user, or null.

16. What is the 'this' keyword in JavaScript?

Functions in JavaScript are essential objects. Like objects, it can be assign to variables, pass to other functions, and return from functions. And much like objects, they have their own properties. 'this' stores the current execution context of the JavaScript program. Thus, when it use inside a function, the value of 'this' will change depending on how the function is defined, how it is invoked, and the default execution context.

17. Explain the working of timers in JavaScript. Also elucidate the drawbacks of using the timer, if any.

The timer executes some specific code at a specific time or any small amount of code in repetition to do that you need to use the functions **setTimout**, **setInterval**, and **clearInterval**. If the JavaScript code sets the timer to 2 minutes and when the times are up then the page displays an alert message "times up". The **setTimeout()** method calls a function or evaluates an expression after a specified number of milliseconds.

18. What is the difference between ViewState and SessionState?

- **ViewState:** It is specific to a single page in a session.
- **SessionState:** It is user specific that can access all the data on the web pages.

19. How to submit a form using JavaScript?

You can use **document.form[0].submit()** method to submit the form in JavaScript.

20. Does JavaScript support automatic type conversion?

Yes, JavaScript supports automatic type conversion.

JavaScript Intermediate Interview Questions

21. What are all the looping structures in JavaScript ?

- **while loop:** A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

- **for loop:** A for loop provides a concise way of writing the loop structure. Unlike a while loop, for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.
- **do while:** A do-while loop is similar to while loop with the only difference that it checks the condition after executing the statements, and therefore is an example of Exit Control Loop.

22. How can the style/class of an element be changed?

To change the style/class of an element there are two possible ways. We use document.getElementByID method

```
document.getElementById("myText").style.fontSize = "16px;
document.getElementById("myText").className = "class";
```

23. Explain how to read and write a file using JavaScript?

- The **readFile()** functions is used for reading operation.

```
readFile( Path, Options, Callback)
```

- The **writeFile()** functions is used for writing operation.

```
writeFile( Path, Data, Callback)
```

24. What is called Variable typing in JavaScript ?

The **variable typing** is the type of variable used to store a number and using that same variable to assign a “string”.

```
Geeks = 42;
Geeks = "GeeksforGeeks";
```

25. How to convert the string of any base to integer in JavaScript?

In JavaScript, parseInt() function is used to convert the string to an integer. This function returns an integer of base which is specified in second argument of parseInt() function. The parseInt() function returns Nan (not a number) when the string doesn’t contain number.

26. Explain how to detect the operating system on the client machine?

To detect the operating system on the client machine, one can simply use navigator.appVersion or navigator.userAgent property. The Navigator appVersion property is a read-only property and it returns the string that represents the version information of the browser.

27. What are the types of Pop up boxes available in JavaScript?

There are three types of pop boxes available in JavaScript.

- **Alert**
- **Confirm**
- **Prompt**

28. What is the difference between an alert box and a confirmation box?

An alert box will display only one button which is the OK button. It is used to inform the user about the agreement has to agree. But a Confirmation box displays two buttons OK and cancel, where the user can decide to agree or not.

29. What is the disadvantage of using innerHTML in JavaScript?

There are lots of disadvantages of using the innerHTML in JavaScript as the content will replace everywhere. If you use += like “innerHTML = innerHTML + ‘html’” still the old content is replaced by HTML. It preserves event handlers attached to any DOM elements.

30. What is the use of void(0) ?

The void(0) is used to call another method without refreshing the page during the calling time parameter “zero” will be passed.

For further reading, check out our dedicated article on ***Intermediate Javascript Interview Questions****. Inside, you’ll discover over 20 questions with detailed answers.*

JavaScript Interview Questions for Experienced

31. What is the ‘Strict’ mode in JavaScript and how can it be enabled?

Strict Mode is a new feature in ECMAScript 5 that allows you to place a program or a function in a “strict” operating context. This strict context prevents certain actions from being taken and throws more exceptions. The statement “use strict” instructs the browser to use the Strict mode, which is a reduced and safer feature set of JavaScript.

32. How to get the status of a CheckBox?

The DOM Input Checkbox Property is used to set or return the checked status of a checkbox field. This property is used to reflect the HTML Checked attribute.

```
document.getElementById("GFG").checked;
```

If the CheckBox is checked then it returns True.

33. How to explain closures in JavaScript and when to use it?

The closure is created when a child functions to keep the environment of the parent’s scope even after the parent’s function has already executed. The Closure is a locally declared variable related to a function. The closure will provide better control over the code when using them.

JavaScript

```javascript
// Explanation of closure
function foo() {
    let b = 1;
    function inner() {
        return b;
    }
    return inner;
}
let get_func_inner = foo();

console.log(get_func_inner());
console.log(get_func_inner());
console.log(get_func_inner());
```

Output

```
1
1
1
```

34. What is the difference between call() and apply() methods ?

Both methods are used in a different situation

- **call() Method:** It calls the method, taking the owner object as argument. The keyword this refers to the 'owner' of the function or the object it belongs to. We can call a method that can be used on different objects.
- **apply() Method:** The apply() method is used to write methods, which can be used on different objects. It is different from the function call() because it takes arguments as an array.

35. How to target a particular frame from a hyperlink in JavaScript ?

This can be done by using the **target** attribute in the hyperlink. Like

```
<a href="/geeksforgeeks.htm" target="newframe">New Page</a>
```

36. Write the errors shown in JavaScript?

There are three different types of errors in JavaScript.

- **Syntax error:** A syntax error is an error in the syntax of a sequence of characters or tokens that are intended to be written in a particular programming language.
- **Logical error:** It is the most difficult error to be traced as it is the error on the logical part of the coding or logical error is a bug in a program that causes to operate incorrectly and terminate abnormally.
- **Runtime Error:** A runtime error is an error that occurs during the running of the program, also known as an exception.

37. What is the difference between JavaScript and Jscript?

JavaScript

- It is a scripting language developed by Netscape.
- It is used to design client and server-side applications.
- It is completely independent of Java language.

Jscript

- It is a scripting language developed by Microsoft.
- It is used to design active online content for the word wide Web.

38. What does *var myArray = [[]];* statement declares?

In JavaScript, this statement is used to declare a two-dimensional array.

39. How many ways an HTML element can be accessed in JavaScript code?

There are four possible ways to access HTML elements in JavaScript which are:

- **getElementById() Method:** It is used to get the element by its id name.

- **getElementsByClass() Method:** It is used to get all the elements that have the given classname.
- **getElementsByTagName() Method:** It is used to get all the elements that have the given tag name.
- **querySelector() Method:** This function takes CSS style selector and returns the first selected element.

40. What is the difference between innerHTML & innerText?

The innerText property sets or returns the text content as plain text of the specified node, and all its descendants whereas the innerHTML property sets or returns the plain text or HTML contents in the elements. Unlike innerText, inner HTML lets you work with HTML rich text and doesn't automatically encode and decode text.

41. What is an event bubbling in JavaScript?

Consider a situation an element is present inside another element and both of them handle an event. When an event occurs in bubbling, the innermost element handles the event first, then the outer, and so on.

*For further reading, check out our dedicated article on **Advanced Javascript Interview Questions**. Inside, you'll discover 20+ questions with detailed answers.*

FAQs on JavaScript Interview Questions

1. What are the primitive data types in JavaScript?

There are six: number, string, boolean, null, undefined, and symbol.

2. How do you explain 'hoisting' in JavaScript?

Variable declarations are hoisted to the top of their scope, allowing access before their actual definition.

3. What's the difference between '===' and '=='?

=== checks for strict equality (value and type), while == performs type coercion before comparison.

4. How can you loop through the elements of an array?

Use a for loop or a forEach method to iterate over each item in the array.

```
/*

let a = (fun) => {
   fun() ;
}
a(snyfun()) ;
// a --> higher order function
// anyfun --> callback function

let higherOrderFun = (callback) => {
   console.log("Higher Order Function called!")
   callback() ;
}
let callback = () => {
   console.log("Callback function called!") ;
}
higherOrderFun(callback) ;

Functional Programming ====> HigherOrderFunction + CallbackFunction

let operation = (task, n1, n2) => {
   task(n1, n2) ;
}
let add = (num1, num2) => {
   console.log(num1+ num2) ;
}
operation(add, 324, 54) ;
let operations = (task, n1, n2) => {
   task(n1, n2) ;
}
let minus = (num1, num2) => {
   console.log(num1- num2) ;
}
operation(minus, 324, 54) ;
let operaten = (task, n1, n2) => {
   task(n1, n2) ;
}
let multiply = (num1, num2) => {
   console.log(num1* num2) ;
}
operation(multiply, 324, 54) ;

let greetings =(person,typeOfGreeting)=> {
   person(typeOfGreeting)
}
let chacha = (typeOfGreeting) => {
   console.log(`hello with my ${typeOfGreeting}greeting`)
}
greetings(chacha,"Namaste")

*/

/*

// ANONYMOUS FUNCTION (function without name)
let demo = function () {
   // function as an expression. [var = fun]
   // functional expression
   console.log("Anonymous stored function called!") ;
}
```

```
demo()
// Anonymous | function as expression | Arrow function
let fun = () => {

}
```

FUNCTIONS IN JAVASCRIPT
1. Simple function with function keyword
2. Arrow function
3. Higher Order function
4. Callback function
5. Anonymous function
6. Named Function
7. Function as an expression
8. Immediate invoke function

```
// Immediate invoke function
(() => {
   console.log("Immediate invoke function called!")
})()

// call stack
console.log("x")
let fun = () => {
   console.log("y") ;
}
console.log("z") ;
fun()

// primitive variables declared with var keyword stored in heap area
// variable like array and object stored
// call stack only contains function

console.log("1") ;
let fun = (a) => {
   console.log(a)
}
console.log("2") ;
fun("xyz") ;
console.log("3") ;

var a = 53 ;
console.log(window) ;
this = window
console.log(this) ;
```

Hoisting | Tremporal dead zone
configuring all declaration before variable execution is called Hoisting

Time lag bw declaration and value assign is called Temporal dead zone

```
------------
console.log(b) ; // error "call before initialization"
let a = 10 ;
let b ;
const c = 20 ;
console.log(b) ;




--------------
console.log(b) ; // undefined
let a = 10 ;
var b ;
const c = 20 ;
console.log(b) ;
```

b is hoisted but not present in temporal dead zone, it is in window object
let and const do not fall in window object..

```
console.log(b) ; // undefined
var b = 23 ;
console.log(b) ;

console.log(b) ; // error
b = 23 ;
console.log(b) ;
```

## declared(let) but not assigned falls in temporal dead zone hence gives error when clg(b)
temporal dead zone -- time span bw declaration and initialization
hoisting -- phenomenon of sending all the declaration to the top

### Difference between Let, var, const

variable scope --

```
let a = 10 ;
if(true) {
   console.log(a) ;
   // local scope uses global scoped variable
}
```

SCOPES ----
1. Global Scope
2. Local Scope

```
let a = "RRR" ;
if(true) {
   console.log(a) ;
```

```
    let b = "Kantara" ;
    const c = "Munjya" ;
    var d = "Kalki" ;
    // 'var' can be accessed outside scope. GLOBAL
}

console.log(a) ;
console.log(b) ; // error
console.log(d) ;

let a = "Bahubali" ;
let func = () => {
    let b = "Pushpa" ;
    const c = "Mirzapur" ;
    var d = "Panchayat" ;
}

func() ;
console.log(d) ;

## 'var' variable doesn't work for function
## 'var' variable declared in function scope are not global

let fun = () => {
    let fun1 = () => {
        console.log("Helllllloooo..")
    }
    fun1() ; // call successful: within the scope
}

fun() ;
fun1() ; // ERROR: cannot be called bcoz of scope

LET/CONST --> Block scoped
VAR       --> Function scope




let a = () => {
    console.log("a block executed!") ;
    let b = () => {
        console.log("b block executed!") ;
        let c = () => {
            console.log("c block executed!") ;
        }
        c() ;
    }
    b() ;
```

```
}

a() ;
```

### JAVASCRIPT CURRYING

```
let a = () => {
    console.log("a block executed!") ;
    let b = () => {
        console.log("b block executed!") ;
        let c = () => {
            console.log("c block executed!") ;
        }
        return c ;
    }
    return b ;
}

a()()() ;
```

parametrized:

```
let a = () => {
    console.log("a block executed!") ;
    let b = () => {
        console.log("b block executed!") ;
        let c = (value) => {
            console.log("c block executed! parameter: "+value) ;
        }
        return c ;
    }
    return b ;

}

a()()(234) ;
```

#### CLOSURE
ability of javascript engine to store required variable in nested function.
### LEXICAL SCOPING --
Finding variable in another function scope or parent function scope is called Lexical scoping

```
let a = () => {
    let val = 10 ;
    let val1 = 20 ;
    let val2 = 30 ;
    // lies in parent function scope of b i.e. 'a'
    let b = () => {
        console.log(val) ;
    }
    b() ;
}

a() ;
```

closure of a :
'a' functional scope contains val = 10
closure -- only those that are searched by other scopes..

```
let a = () => {
    let val = 10 ;
    let b = () => {
        console.log(val) ;
    }
    return b ;
}

a()() ;
```

```
let minku = () => {
    let bike = "Splendor" ;
    let mobile = "Iphone" ;
    let tinku = () => {
        console.log("The bike is : ", bike) ;
    }
    return tinku ;
}

minku()() ; // working
```

---------------------->

```
let tinkuFunReference = minku() ;
```

### minku call stack removed after function call goes over returning tinku() reference..

```
tinkuFunReference() ; // working
```

## even minku removed from call stack the closure of minku is reserved in closure space hence value can be
## executed by external call of tinku i.e. tinkuFunReference()
## op -- The bike is : Splendor

immediate invoke function

```
(() => {
    console.log("Anonymous Function.")
})() ;
```

```
### CONDITIONAL STATEMENT ---

let num = prompt("Enter a number!") ;
## based on operation javascript engine decide to concatenate or add
if(num & 1) {
   console.log(`${num} is an odd number`) ;
} else {
   console.log(`${num} is an even number`) ;
}

console.log("10"+10) ;  // 1010 (concat)
console.log("10"-10) ;  // 0
console.log("10"*10) ;  // 100
console.log("10"/10) ;  // 1
# not recommended..

## Learn typescript, considers and solves issue.



### Ternary Operator
let username = "brijesh367" ;
let password = "qwr4jjfjfj" ;
let user = prompt("Enter the username!") ;
let pass = prompt("Enter the password") ;
console.log(((user == username && pass == password) ? "Logged In Successfully!"
   : "Wrong Credentials Inserted!")) ;

let isHungry = true ;
let isFishavailable = true ;

isHungry ? (console.log("Cat is Hungry!") || (isFishavailable) ?
console.log("Cat will eat the Fish!"): console.log("Cat will drink the milk!"))
: console.log("Cat is not Hungry!") ;

## here '||' operator will let clg



let takla = prompt("Enter your name!") ;
switch(takla) {
   case "Sushant": console.log("4827582357") ;
               break ;
   case "Deepak" : console.log("4827582357") ;
               break ;
   case "Deepak" : console.log("4827582357") ;
               break ;
   default : console.log("Chal be takle tera nam nhi h idhar!") ;
}   */
```