

# ***Important JavaScript Concepts***



---


**For Interviews**

---

***Part 2***

# Explain Temporal dead zone.

Temporal Dead Zone is a behaviour that occurs with variables declared using `let` and `const` keywords. It is a behaviour where we try to access a variable before it is initialized.



```
1  num = 23; // Gives reference error
2  let num;
3
4  function func() {
5      greeting = "Hi"; // Throws a reference error
6      let greeting;
7  }
8  func();
```

# What is closure in JavaScript?

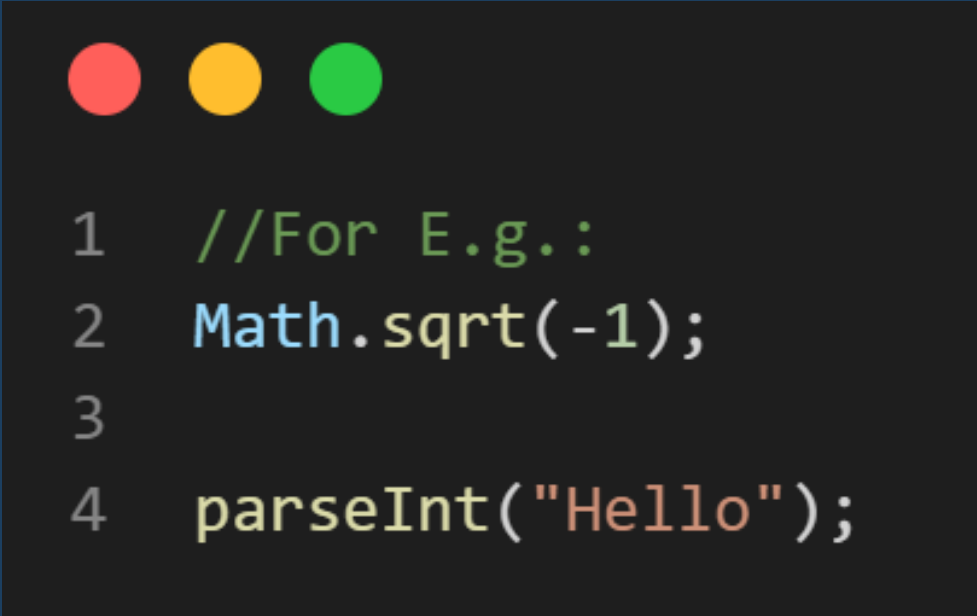
A closure consists of references to the surrounding state (the lexical environment) and a function that has been wrapped (contained). In other words, a closure enables inner functions to access the scope of an outside function. Closures are formed whenever a function is created in JavaScript, during function creation time.

## What are differences between “==” & “===”?

The == operator performs a loose equality comparison that, if necessary to enable the comparison, applies type coercion. On the other hand, the === operator conducts a strict equality comparison without type coercion and necessitates that the operands be of the same type (as well as the same value).

# What is NaN property?

The NaN property is a global property that represents "Not-a-Number" value. i.e, It indicates that a value is not a legal number. It is very rare to use NaN in a program but it can be used as return value for few cases.



```
1 //For E.g.:
2 Math.sqrt(-1);
3
4 parseInt("Hello");
```

# What is Critical Rendering Path?

The Critical Rendering Path is the sequence of steps the browser goes through to convert the HTML, CSS, and JavaScript into pixels on the screen. Optimizing the critical render path improves render performance. The critical rendering path includes the Document Object Model (DOM), CSS Object Model (CSSOM), render tree and layout.

# What is the difference between null and undefined?

null	undefined
It is an assignment value which indicates that variable points to no object.	It is not an assignment value where a variable has been declared but has not yet been assigned a value.
Type of null is object	Type of undefined is undefined
The null value is a primitive value that represents the null, empty, or non-existent reference.	The undefined value is a primitive value used when a variable has not been assigned a value.
Indicates the absence of a value for a variable	Indicates absence of variable itself
Converted to zero (0) while performing primitive operations	Converted to NaN while performing primitive operations

# What are the terms BOM and DOM in JavaScript?

DOM stands for **Document Object Model** and BOM for **Browser Object Model**.

DOM: An element may be added, changed, or removed from a document using the Document Object Model (DOM), a programming interface for HTML and XML documents. It specifies how a document is accessed and handled, as well as its logical structure. The DOM allows the webpage to be represented as a structured hierarchy, making it simple to access and modify HTML tags, IDs, classes, attributes, and elements using the Document object's provided commands and methods. This makes it easier for programmers and users to understand the document.

DOM provides several methods to find & manipulate the behavior of the HTML element:

- getElementById() Method
- getElementsByClassName() Method
- getElementsByName() Method
- getElementsByTagName() Method
- querySelector() Method
- querySelectorAll() Method

**BOM: Browser Object model** is a browser-specific convention referring to all the objects exposed by the web browser. The BOM allows JavaScript to “interact with” the browser. The window object represents a browser window and all its corresponding features. A window object is created automatically by the browser itself. JavaScript’s window.screen object contains information about the user’s screen.

### **Window properties of BOM are:**

- screen.width
- screen.height
- screen.availWidth
- screen.availHeight
- screen.colorDepth
- screen.pixelDepth

### **Window methods of BOM are:**

- window.open() Method
- window.close() Method
- window.moveTo() Method
- window.moveBy() Method
- window.resizeTo() Method



# What are basic JavaScript array methods?

Some of the basic JavaScript methods are:

**push() method:** adding a new element to an array. Since JavaScript arrays are changeable objects, adding and removing elements from an array is simple. Additionally, it alters itself when we change the array's elements.

- Syntax: `Array.push(item1, item2 ...)`

**pop() method:** This method is used to remove elements from the end of an array.

- Syntax: `Array.pop()`

**slice() method:** This method returns a new array containing a portion of the original array, based on the start and end index provided as arguments

- Syntax: `Array.slice (startIndex , endIndex);`



**map() method:** The map() method in JavaScript creates an array by calling a specific function on each element present in the parent array. It is a non-mutating method. Generally, the map() method is used to iterate over an array and call the function on every element of an array.

- Syntax: `Array.map(function(currentValue, index, arr), thisValue)`

**reduce() method:** The array reduce() method in JavaScript is used to reduce the array to a single value and executes a provided function for each value of the array (from left to right) and the return value of the function is stored in an accumulator.

- Syntax: `Array.reduce(function(total, currentValue, currentIndex, arr), initialValue)`

# What is the rest parameter and spread operator?

## Rest parameter (...):

- It offers a better method of managing a function's parameters.
- We can write functions that accept a variable number of arguments using the rest parameter syntax.
- The remainder parameter will turn any number of inputs into an array.
- Additionally, it assists in extracting all or some of the arguments.
- Applying three dots (...) before the parameters enables the use of rest parameters.



```
1  Syntax:
2
3  function extractingArgs(...args){
4
5  return args[1];
6
7  }
8
9  // extractingArgs(8,9,1); // Returns 9
10
11 function addAllArgs(...args){
12
13 let sumOfArgs = 0;
14
15 let i = 0;
16
17 while(i < args.length){
18
19 sumOfArgs += args[i];
20
21 i++;
22
23 }
24
25 return sumOfArgs;
26
27 }
28
29 addAllArgs(6, 5, 7, 99); // Returns 117
30
31 addAllArgs(1, 3, 4); // Returns 8
```

## Spread operator(...):

- Although the spread operator's syntax is identical to that of the rest parameter, it is used to spread object literals and arrays. Spread operators are also used when a function call expects one or more arguments.

```
1 Syntax:
2
3 function addFourNumbers(num1,num2,num3,num4){
4
5   return num1 + num2 + num3 + num4;
6
7 }
8
9 let fourNumbers = [5, 6, 7, 8];
10
11 addFourNumbers(...fourNumbers);
12
13 // Spreads [5,6,7,8] as 5,6,7,8
14
15 let array1 = [3, 4, 5, 6];
16
17 let clonedArray1 = [...array1];
18
19 // Spreads the array into 3,4,5,6
20
21 console.log(clonedArray1); // Outputs [3,4,5,6]
22
23 let obj1 = {x:'Hello', y:'Bye'};
24
25 let clonedObj1 = {...obj1}; // Spreads and clones obj1
26
27 console.log(obj1);
28
29 let obj2 = {z:'Yes', a:'No'};
30
31 let mergedObj = {...obj1, ...obj2}; // Spreads both the
32 objects and merges it
33
34 console.log(mergedObj);
35
36 // Outputs {x:'Hello', y:'Bye',z:'Yes',a:'No'}
```

# Follow for more great tips



**Kailash G**

@kailash1203

## Part 3 Coming Soon