

**JS**

# JavaScript

**Interview Questions with  
Answers**

**Author: Syed Baqir Raza**

 [@thebaqirraza](#)

## Disclaimer:

All the questions cover almost all basic concepts of JavaScript. So, if you have a JavaScript interview due, prepare these questions. Covering these questions will cover almost 80-90% of your interview preparation.

## What is JavaScript?

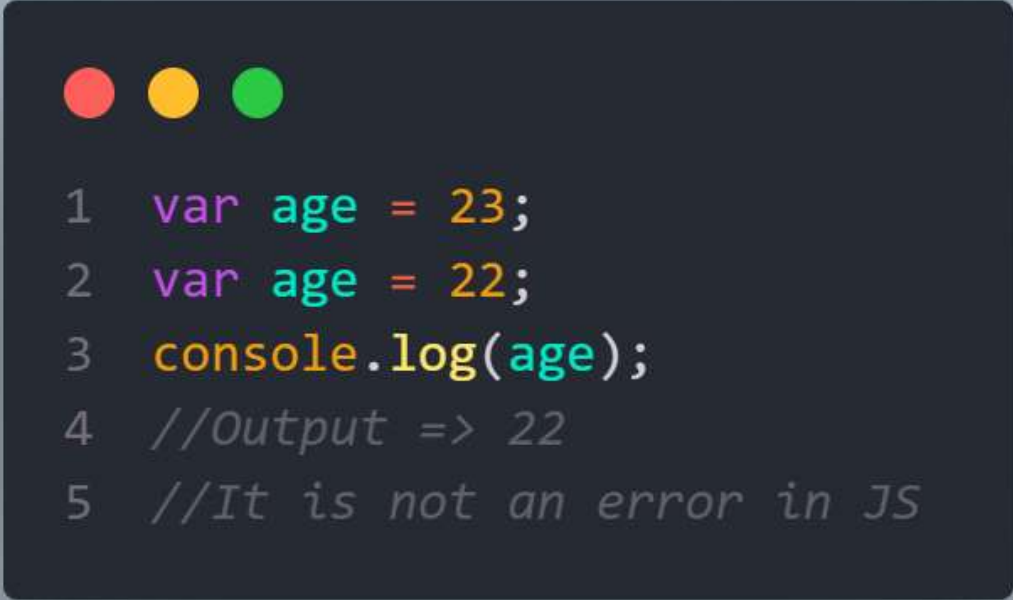
JavaScript is a scripting or programming language. It is an interpreted language. It does not get compiled but is interpreted as the script runs.

## Is JavaScript single-threaded or multi-threaded?

JavaScript is single-threaded.

## What is difference between 'let', 'const' and 'var' keywords?

**Var** keyword is globally scoped. It allows redeclaration within the same scope.



```
1  var age = 23;  
2  var age = 22;  
3  console.log(age);  
4  //Output => 22  
5  //It is not an error in JS
```

**Let** keyword is block scoped. It does not allow redeclaration within the same scope.

```
1 let age = 23;  
2 let age = 22;  
3 console.log(age);  
4 //It is an error in JS because variable 'age' has been redeclared within the same scope.  
5
```

**Const** keyword is also block scoped. It cannot be updated or redeclared. In case of objects, properties of the object can be updated but the object itself cannot be updated.

```
1 const age = 23;  
2 age = 24;  
3 console.log(age);  
4 //It is an error in JS because variable 'age' has already been initialized using 'const' keyword.  
5
```

**What are the spread operator, rest operator, and default parameter?**

The **spread operator** helps us expand an iterable such as an array where multiple arguments are needed. It also helps to expand the object expressions. There can be more than one spread operator in JS functions.

```
1  const a = [1, 2, 3];
2  const b = [4, 5, 6];
3  const c = [...a, ...b];
4
5  console.log(c);
6  //Output => [ 1, 2, 3, 4, 5, 6 ]
```

The **rest operator** compresses the elements of an iterable. There can only be one rest operator in JS functions. It allows a function to accept an indefinite number of arguments as an array

```
1  const calcAverage = (...args) => {
2    let sum = args.reduce(
3      (accumulator, currentValue) => accumulator + currentValue
4    );
5    sum = sum / args.length;
6    console.log(sum);
7  };
8
9  calcAverage(4, 5, 6);
10 //Output => 5
11 //Using rest operator/parameter a function can accept indefinite number of arguments.
```

**Default parameters** allow named parameters to be initialized with default values if no value or undefined is passed.

```
1 const numberSum = (a, b = 0) => {  
2   console.log(a + b);  
3 };  
4 numberSum(5);  
5 //Output => 5  
6 //If second argument is passed it will override the default value
```

## What is "Deep Copy" and "Shallow Copy" in JS?

To copy an object in JavaScript, you have three options:

1. Use the **spread (...)** syntax
2. Use the **Object.assign()** method
3. Use the **JSON.stringify()** and **JSON.parse()** methods

**Deep copy** means that all the values of the new variable is disconnected from the original variable.

A **shallow copy** means that some values are still connected to the original variable.

### Note:

Spread operator and `Object.assign()` are used for **shallow copy** while `JSON.stringify()` and `JSON.parse()` methods are used for **deep copy**.

## What is "Promise", "Callback Function", "async", and "await" in JS?

**Promise** in JS is something that takes some time to do. There are two possible outcomes of a promise.

1. Either run and resolve the promise.
2. Some error occurs along the line and the promise is rejected.

Promise takes two functions as a parameter i.e., **resolve** and **reject**.

```
1  const fun = (num) => {  
2    let myPromise = new Promise((resolve, reject) => {  
3      num < 5 ? resolve(`The given number is ${num}`) : reject("Error");  
4    });  
5    myPromise.then((result) => console.log(result));  
6  };  
7  fun(4);  
8  //Output => The given number is 4  
9  //If number is greater than or equal to 5 promise will be rejected.
```

**Async** keyword is used to define asynchronous functions.

**Await** stalls JavaScript from assigning value to a variable until promise is resolved.

## What is "Event Bubbling" and "Event Capturing" in JS?

**Event Bubbling:** Whenever an event happens on an element the event handlers will first run on it and then on its parent and finally all the way up to the ancestors.

**Event Capturing:** It is the reverse of the event bubbling and here the event starts from the parent element and then to its child element.

## What is higher order function in JS?

A function that takes another function as an argument or returns a function is called a **higher-order function**.

### Example:

Array methods like, **map()** , **filter()** , **reduce()** , **find()** , and many more. All these functions take another function as an argument to apply it to the elements of an array.

## What are two different types of functions in JS?

The two different types of functions in JS are **pre-defined** functions and **user-defined** functions.

## What is arrow function in JS?

An arrow function expression is a compact alternative to a traditional function expression, with some semantic differences and deliberate limitations in usage:

- Arrow functions don't have their own bindings to this, arguments, or super, and should not be used as methods.
- Arrow functions cannot be used as constructors. Calling them with new throws a TypeError. They also don't have access to the new.target keyword.
- Arrow functions cannot use yield within their body and cannot be created as generator functions.

## How many ways to create an object in JS?

```
1 //There are three ways to create an Object
2 //1. Object Literal
3 const person1 = {
4   name: "John",
5 };
6
7 //2. Using instance of an Object
8 const person2 = new Object({
9   name: "John",
10 });
11
12 //3. Using Constructor Functions
13 function Person() {
14   this.name = "John";
15 }
16
17 const p = new Person();
```



## What is temporal dead zone in JS?

The temporal dead zone is the period during which the 'let' and 'const' declarations cannot be accessed.

TDZ starts when the code execution enters the block which contains the 'let' and 'const' declaration and continues until the declaration has been executed.

## What is function currying in JS?

Currying is the transformation of a function with multiple arguments into a sequence of single-argument functions.

This means converting a function **f(a,b,c,...)** into a function like this **f(a)(b)(c)**.

## What is MutationObserver in JS?

The **MutationObserver** interface provides the ability to watch for changes being made to the **DOM tree**.

## What is memoization in JS?

Memoization is a technique for speeding up the applications by caching the results of expensive function calls and returning them when the same inputs are used again.

## What is "null" and "undefined" in JS?

**Null:** It is the intentional absence of value. It is one of the primitive values of JS.

**Undefined:** It means the value does not exist. It is a property of the global object. It is also one of the primitive values of JS.

## What are falsy values in JS?

A falsy value is something that evaluates **FALSE**, for instance when checking for a variable. There are only six falsy values in JS.

1. undefined
2. null
3. NaN
4. 0
5. "" (empty string)
6. false

## What is "setTimeout()" and "setInterval()" in JS?

**setTimeout()**: Executes a function after waiting a specified number of milliseconds.

**setInterval()**: Same as setTimeout(), but repeats the execution of the function continuously.

### **Note:**

*Write code to understand the difference more clearly.*

## What is "Object.seal()" and "Object.freeze()" in JS?

Both freeze and seal are used to create non-extensible objects in JS, but there are plenty of differences between them.

**Object.seal()** allows changes to the existing properties of an object whereas **Object.freeze()** does not allow so.

**Object.freeze()** makes an object immune to everything even little changes cannot be made.

**Object.seal()** prevents from deletion of existing properties but cannot prevent them from the external changes.



```
1  const person = {  
2    name: "John",  
3    age: 30,  
4  };  
5  
6  Object.seal(person);  
7  person.age = 31;  
8  person.city = "Washington";  
9  console.log(person);  
10  
11 //Output => {name: 'John', age: 31}  
12
```



```
1  const person = {  
2    name: "John",  
3    age: 30,  
4  };  
5  
6  Object.freeze(person);  
7  person.age = 31;  
8  person.city = "Washington";  
9  console.log(person);  
10  
11 //Output => {name: 'John', age: 30}  
12
```

## What is "Map()" and "Set()" in JS?

A JavaScript **Set** is a collection of unique values. Each value can only occur once in a Set. A Set can hold any value of any data type.

A **Map** holds key-value pairs where the keys can be any datatype. A Map remembers the original insertion order of the keys. A Map has a property that represents the size of the map.

```
1 //Set()
2 const letters = new Set();
3
4 letters.add("a");
5 letters.add("b");
6 letters.add("c");
7
8 console.log(letters);
9 //Output => Set {'a', 'b', 'c'}
10 //If same value is added than it will not be repeated
11
12 //-----
13
14 //Map()
15 const fruits = new Map();
16
17 fruits.set("apples", 500);
18 fruits.set("bananas", 300);
19 fruits.set("grapes", 400);
20
21 console.log(fruits);
22 //Output => Map {'apples' => 500, 'bananas' => 300, 'grapes' => 400}
23 //If same value is added than it will not be repeated
24
```

## What are "localStorage", "localStorage" and "cookie" in JS?

**LocalStorage:** It is a web storage method that helps us store data on the client's computer in the form of key/value pairs in a web browser. The data is stored in local storage for a lifetime unless the user manually deletes it from the browser.

**SessionStorage:** It is very similar to local storage. The main difference lies in the lifespan as it persists in the browser until its current tab is open. Once you close the tab or terminate it, the data on sessionStorage also gets lost.

**Cookies:** Cookies are data, stored in small text files on your computer. When a web server has sent a web page to the browser, the connection is shut down, and the server forgets everything about the user.

Cookies are invented to solve the problem of “How to remember information about the user”.

## What is "JSON.stringify()" and "JSON.parse()" methods in JS?

**JSON.stringify()** function is used to convert a JS object into a string.

**JSON.parse()** function is used to convert a string into a JS object.

## What is generator function in JS?

When we stop the execution of a function from anywhere inside the function it is called a generator function. Execution can be stopped using “**yield**” keyword.

## How to stop event propagation in JS?

To stop an event from further propagation in the capturing and bubbling phases, we call **Event.stopPropagation()** method in the event handler.

## What is closure in JS?

Closure gives access to an outer function's scope from an inner function. A closure determines what variables will be accessible to an inner function.

## What is hoisting in JS?

JS hoisting refers to the process where the interpreter appears to move the declarations of functions, classes, or variables to the top of their scope, prior to the execution of the code.