JavaScript Interview Questions and Answers

Last Updated : 27 Dec, 2024

JavaScript (JS) is the most popular lightweight, scripting, and interpreted programming language. JavaScript is well-known as a scripting language for **web pages, mobile apps, web servers**, and many other platforms. It is essential for both **front-end** and **back-end developers** to have a strong command of JavaScript, as many job roles require fluency in this language.

Below, we have compiled the **Top 60+ JavaScript Interview Questions and Answers** tailored for both **freshers** and **experienced developers.** Here, we will cover everything, including **Core JavaScript Concepts**, ES6+ features, DOM manipulation, asynchronous JavaScript, error handling, JavaScript frameworks and libraries, and more, that will surely help you crack your next JavaScript interview.

JavaScript Interview Questions

Note: *Before proceeding to learn **JavaScript interview questions and answers**, if you are completely new to the language, we recommend building a solid foundation first by exploring our free JavaScript Tutorial*

Table of Content

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?

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

- **Java:** It 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".
- **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.

To give yourself an extra edge, consider taking our **JavaScript Course**, which not only covers core topics but also provides interview preparation tips, practice problems, and insights from industry experts.

2. What are Data Types in JavaScript?

JavaScript data types are categorized into two parts i.e. primitive and non-primitive types.

- **Primitive Data Type:** The predefined data types provided by JavaScript language are known as primitive data type. Primitive data types are also known as in-built data types.

  - Numbers
  - Strings
  - Boolean
  - Symbol
  - Undefined
  - Null
  - BigInt
- **Non-Premitive Data Type:** The data types that are derived from primitive data types are known as non-primitive data types. It is also known as derived data types or reference data types.
  - 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 but 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

```html
<html>
<head>
</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>
```

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 the difference between `null` and `undefined` in JavaScript?

A deeper comparison between the two special values, `null` (intentional absence of value) and `undefined` (uninitialized variables).

16. 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.

17. 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.

18. Explain the working of timers in JavaScript. Also explain 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.

19. 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.

20. How to submit a form using JavaScript?

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

21. Does JavaScript support automatic type conversion?

Yes, JavaScript supports automatic type conversion.

22. What is a template literal in JavaScript?

A template literal in JavaScript is a way to define strings that allow embedded expressions and multi-line formatting. It uses backticks (`) instead of quotes and supports `${}` for embedding variables or expressions inside the string.

23. What is a higher-order function in JavaScript?

A higher-order function in JavaScript is a function that either takes one or more functions as arguments, or returns a function as its result. These functions allow for more abstract and reusable code, enabling functional programming patterns

For example, `map()` and `filter()` are higher-order functions because they take callback functions as arguments.

JavaScript Intermediate Interview Questions

24. 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.

25. What is lexical scope in JavaScript?

Lexical scope in JavaScript refers to the way variables are resolved based on their location in the source code. A variable's scope is determined by the position of the code where it is defined, and it is accessible to any nested functions or blocks. This means that functions have access to variables in their own scope and the outer (lexical) scopes, but not to variables in inner scopes.

JavaScript

```javascript
let outer = "I am outside!";
function inner() {
    console.log(outer);
}
inner();
```

In this example, `inner()` can access the `outer` variable because of lexical scoping.

26. How does lexical scoping work with the `this` keyword in JavaScript?

In JavaScript, **lexical scoping** primarily applies to variable resolution, while the behavior of the `this` keyword is determined by how a function is called, not by its position in the code. The value of `this` is dynamically determined at runtime based on the function's context (e.g., whether it's called as a method, in a global context, or with `call`, `apply`, or `bind`).

JavaScript

```javascript
const obj = {
    name: "JavaScript",
    greet: function () {
        console.log(this.name);
    }
};
obj.greet(); // "JavaScript"
```

Here, `this` refers to `obj` because the function is called as a method of the object. Lexical scoping affects variable lookups but doesn't alter how `this` behaves.

27. 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)
```

28. 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";
```

29. What is hoisting in JavaScript?

Hoisting in JavaScript is the behavior where variable and function declarations are moved to the top of their containing scope during compilation, before the code is executed. This means you can reference variables and functions before they are declared in the code. However, only declarations are hoisted, not initializations.

JavaScript

```javascript
console.log(a); // undefined
var a = 5;
```

In this case, the declaration of `a` is hoisted, but its value (`5`) is not assigned until the code execution reaches that line. Hoisting applies differently for `var`, `let`, `const`, and function declarations.

30. 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.

31. 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.

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

There are three types of pop boxes available in JavaScript.

- Alert
- Confirm
- Prompt

33. 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.

34. What are JavaScript modules, and how do you import/export them?

JavaScript modules allow you to split your code into smaller, reusable pieces. They enable the export of variables, functions, or objects from one file and the import of them into another. To export an element, you use `export` (either named or default). To **import** it, you use `import`

JavaScript

```javascript
// In file1.js
export const greet = () => "Hello";

// In file2.js
import { greet } from './file1';
console.log(greet());
```

Modules help organize code and avoid global namespace pollution. They are natively supported in modern JavaScript through `import` and `export` statements.

35. What are `WeakMap` and `WeakSet`, and how are they different from `Map` and `Set`?

A **WeakMap** is a collection of key-value pairs where keys are objects and the values can be any data type. The key-value pairs in a WeakMap are “weakly” held, meaning if no other references to a key exist, the entry can be garbage collected. A **WeakSet** is a collection of unique objects, and like WeakMap, the objects are weakly held.

The main difference from **Map** and **Set** is that in **Map** and **Set**, entries are strongly held, meaning they prevent garbage collection, while in **WeakMap** and WeakSet, entries can be garbage collected if no other references to the objects exist.

36. What is the role of the `setImmediate` function in Node.js, and how is it different from `setTimeout`?

In Node.js, the **`setImmediate()`** function schedules a callback to be executed in the next iteration of the event loop, after the current event handling phase. It's commonly used to execute I/O tasks after the current operation completes.

The key difference between **`setImmediate()`** and `setTimeout()` is that `setImmediate()` runs after the I/O events in the event loop, while `setTimeout()` runs after a specified delay (even if the event loop is still busy). `setTimeout()` schedules tasks with a minimum delay, while `setImmediate()` executes as soon as the event loop is free.

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

37. 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.

38. What are the advantages and disadvantages of using `async/await` over traditional callbacks or promises?

Advantages of async/await:

- **Improved readability**: Async/await makes asynchronous code look like synchronous code, making it easier to read and maintain.
- **Simplified error handling**: With `try/catch`, error handling is more straightforward compared to `.catch()` with promises or callback-based error handling.
- **Avoids callback hell**: It eliminates deeply nested callbacks, reducing complexity in asynchronous logic.

Disadvantages of async/await:

- **Requires modern JavaScript**: It's supported in ES2017 and above, so older environments may need transpiling.
- **Limited concurrency control**: Unlike promises with `.all()` or `.race()`, async/await can be less flexible for handling multiple parallel tasks.

38. 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
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
```

39. 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.

40.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>
```

41. 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.

42. 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.

43. 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.

44. 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.

45. Explain the concept of memoization in JavaScript?

Memoization in JavaScript is an optimization technique that stores the results of expensive function calls and reuses them when the same inputs occur again. This reduces the number of computations by caching the results. Memoization is typically implemented using an object or a map to store function arguments and their corresponding results. When the function is called with the same arguments, the cached result is returned instead of recalculating it. This improves performance, especially for functions with repeated calls and expensive computations.

46. What is the difference between `==` and `===` in JavaScript?

In JavaScript, `==` is the **loose equality** operator, which compares two values for equality after performing type coercion if necessary. This means it converts the operands to the same type before comparing.

`===` is the **strict equality** operator, which compares both the values and their types, without performing type conversion.

47. Explain the concept of promises and how they work.

A Promise in JavaScript is an object that represents the result of an asynchronous operation. It can be in one of three states: **pending**, **fulfilled** (resolved), or **rejected**. You create a promise using `new Promise()`, passing an executor function with `resolve` and `reject` callbacks. When the operation succeeds, `resolve()` is called; if it fails, `reject()` is used. Promises are handled with `.then()` for success and `.catch()` for failure. They can be chained to handle sequences of asynchronous tasks in a more readable way.

48. What is the difference between a shallow copy and a deep copy?

A shallow copy creates a new object but copies references to the original nested objects, meaning changes to the nested objects affect both the original and the copy. A deep copy, on the other hand, creates a new object and recursively copies all nested objects, ensuring that the original and the copy are completely independent. In a shallow copy, nested objects are shared, while in a deep copy, they are fully duplicated.

49. Explain the concept of the event loop and the call stack in JavaScript. How does JavaScript handle asynchronous code execution?

In JavaScript, the event loop manages the execution of code, handling both synchronous and asynchronous operations. The call stack stores function calls and executes them in a Last In, First Out (LIFO) order. When asynchronous code (e.g., `setTimeout`, promises) is encountered, it's offloaded to the **callback queue** once its execution context is ready. The event loop continuously checks the call stack and moves tasks from the callback queue to the stack when it's empty, allowing asynchronous code to run without blocking the main thread.

50. What are Web Workers, and how do you use them to run scripts in the background?

Web Workers are JavaScript threads that run in the background, separate from the main thread, allowing long-running scripts to be executed without blocking the user interface. You can create a Web Worker using the `Worker` constructor, passing a JavaScript file as an argument. Once created, the worker can perform tasks asynchronously, and you can communicate with it via `postMessage` and `onmessage` events, ensuring the main thread remains responsive.

51. Explain the concept of "debouncing" and "throttling" in JavaScript. How can these techniques optimize performance?

Debouncing and throttling are techniques used to optimize performance by limiting the frequency of function executions in response to events like scrolling or resizing.

- **Debouncing** ensures that a function is only executed after a certain amount of idle time, i.e., it delays the execution until the event stops

triggering for a specified time (e.g., for search input).
- **Throttling** limits the number of times a function can be executed in a given period, ensuring it runs at regular intervals (e.g., during scroll or window resizing).

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

JavaScript MCQ Coding Interview Questions

52. Which of the following is a JavaScript data type?

Options:

1. `number`
2. `string`
3. `boolean`
4. All of the above

Answer:

4

Explanation:

- JavaScript has several built-in data types, and `number`, `string`, and `boolean` are all valid types.
- A `number` represents numeric values, a `string` represents sequences of characters, and a `boolean` represents either `true` or `false`. All of these are fundamental data types in JavaScript.

53. What is the result of the following code?

JavaScript

```
let a = [1, 2, 3];
let b = a;
b[0] = 100;
console.log(a);
```

Options:

1. `[100, 2, 3]`
2. `[1, 2, 3]`
3. `[100, 100, 100]`
4. `undefined`

Answer :

1

Explanation :

- The code snippet represents two arrays `[100, 2, 3]` and `[1, 2, 3]` being compared using the equality operator (`==`).
- In JavaScript, arrays are reference types, meaning each array is a reference to an object in memory.
- Since the two arrays are different objects in memory, the comparison results in `false`, which evaluates to `undefined` when logged. Therefore, the result is `undefined`.

54. What is the output of the following code?

JavaScript

```
console.log([] + []);
```

Options:

1. `null`
2. `undefined`
3. `''`
4. `[]`

Answer :

```
3
```

Explanation:

- The `+` operator concatenates two empty arrays, which results in an empty string `''`.

55. What will be the output of the following code?

JavaScript

```javascript
(function() {
    var a = b = 5;
})();
console.log(typeof a);
console.log(typeof b);
```

Options:

1. `typeof a: "undefined"`
   `typeof b: "number"`
2. typeof a: “number”
   typeof b: “number”
3. typeof a: “undefined”
   typeof b: “undefined”
4. typeof a: “number”
   typeof b: “undefined”

Answer:

```
1
```

Explanation:

- Inside the IIFE, `b = 5` is treated as a global variable (since no `var`, `let`, or `const` keyword is used).
- However, `a` is declared with `var` and is local to the function, so it is `undefined` outside.

56. What will be logged in the console?

```
console.log(1 < 2 < 3);
console.log(3 > 2 > 1);
```

Options:

1. `true, true`
2. `true, false`
3. `false, true`
4. `false, false`

Answer:

```
2
```

Explanation:

- `1 < 2 < 3` is evaluated as `(1 < 2) < 3`, which becomes `true < 3`. In JavaScript, `true` is treated as `1`, so `1 < 3` is `true`.
- `3 > 2 > 1` becomes `(3 > 2) > 1`, which results in `true > 1`. Since `true` is `1`, the comparison becomes `1 > 1`, which is `false`.

57. What will be the output of the following code?

```
const obj1 = { a: 1 };
const obj2 = { a: 1 };
console.log(obj1 == obj2);
console.log(obj1 === obj2);
```

Options:

1. `true, true`

2. `true, false`
3. `false, true`
4. `false, false`

Answer:

```
4
```

Explanation:

- In JavaScript, objects are compared by reference, not by value. Since `obj1` and `obj2` point to different memory locations, both `==` and `===` comparisons return `false`.

58. What will be the result of the following code?

```
let x = 10;
let y = (x++, x + 1, x * 2);
console.log(y);
```

Options :

1. `22`
2. `12`
3. `21`
4. `20`

Answer:

```
22
```

Explanation:

- The comma operator `( , )` evaluates all expressions but returns the value of the last one.
- `x++` increments `x` to `11`, but the result of this expression is the original `10`.

- `x + 1` becomes `11 + 1 = 12`, and the final expression `x * 2` evaluates to `11 * 2 = 22`, which is assigned to `y`.

59. What will be the output of this asynchronous JavaScript code?

```
console.log('A');
setTimeout(() => console.log('B'), 0);
Promise.resolve().then(() => console.log('C'));
console.log('D');
```

Options:

1. `A D B C`
2. `A B C D`
3. `A D C B`
4. `A C D B`

Answer:

```
3
```

Explanation:

- The synchronous code runs first, logging `'A'` and `'D'`.
- `Promise` callbacks (microtasks) are executed before `setTimeout` (macrotasks). So `'C'` is logged before `'B'`.

60. What will be the output of this recursive function?

```
function foo(num) {
  if (num === 0) return 1;
  return num + foo(num - 1);
}
console.log(foo(3));
```

Options:

1. `3`
2. `6`
3. `7`
4. `10`

Answer:

```
3
```

Explanation:

- The function works recursively:
  - `foo(3)` ⟶ `3 + foo(2)`
  - `foo(2)` ⟶ `2 + foo(1)`
  - `foo(1)` ⟶ `1 + foo(0)`
  - `foo(0)` ⟶ `1`
- So, the total is `3 + 2 + 1 + 1 = 7`.

61. What will be printed in the following code?

```
let a = [1, 2, 3];
let b = a;
b.push(4);
console.log(a);
console.log(b);
```

Options:

1. `[1, 2, 3]`

2. [1, 2, 3, 4]
   [1, 2, 3, 4]
3. [1, 2, 3]
   [1, 2, 3]
4. [1, 2, 3, 4]
   [1, 2, 3]

Answer:

2

Explanation:

- In JavaScript, arrays are reference types. Both `a` and `b` point to the same array in memory. Modifying `b` also affects `a`.

62. What will be logged by the following code?

```
function test() {
  console.log(this);
}
test.call(null);
```

Options:

1. `null`
2. `undefined`
3. `Window` or `global` object
4. `TypeError`

Answer:

3

Explanation:

- In non-strict mode, calling a function with `this` set to `null` defaults to the global object (`Window` in browsers or `global` in Node.js).

Javascript Frequently Asked Questions – FAQs

What are the primitive data types in JavaScript?

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

How do you explain ‘hoisting’ in JavaScript?

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

What’s the difference between ‘===’ and ‘==’?

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

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.

What is a closure in JavaScript?

A closure is a function that retains access to its lexical scope, even when the function is executed outside that scope.

What is the event loop in JavaScript?

The event loop is a mechanism that allows JavaScript to execute asynchronous code. It continuously checks the call stack to see if there are any functions to execute and if the call stack is empty, it checks the callback queue for events that need to be processed.

What is the difference between synchronous and asynchronous code in JavaScript?

- ***Synchronous code***: *The code is executed line by line, one after another. Each task must be completed before the next one begins.*
- ***Asynchronous code***: *The code allows tasks to run independently of the main program flow, so the program doesn't block other operations while waiting for one task to complete (e.g.,* `setTimeout`*, Promises).*

Comment More info

Next Article

TypeScript Interview Questions and Answers

Similar Reads

Active Directory Interview Questions - Top 50+ Questions and Answ...

Active Directory (AD) is a crucial component of modern enterprise IT infrastructure, providing centralized authentication, authorization, and...

15+ min read

Teacher Interview Questions - Top 70 Questions and Answers for 2024

Teaching is a noble profession that requires a unique blend of knowledge, skills, and passion. As educators, teachers play a crucial role in shaping...

15+ min read

JavaScript String Interview Questions and Answers

JavaScript (JS) is the world's most popular lightweight programming language, created by Brendan Eich in 1995. It is a must-have skill for we...

14 min read

JavaScript Interview Questions and Answers (2024) - Intermediate…

In this article, you will learn JavaScript interview questions and answers intermediate level that are most frequently asked in interviews. Before...

6 min read

JavaScript Array Interview Questions and Answers

JavaScript Array Interview Questions and Answers contains the list of top 50 array based questions that are frequently asked in interviews. The...

15+ min read

JavaScript Interview Questions and Answers (2024) - Advanced Level

In this article, you will learn JavaScript interview questions and answers Advanced level that are most frequently asked in interviews. Before...

6 min read

JavaScript Coding Questions and Answers

JavaScript is the most commonly used interpreted, and scripted Programming language. It is used to make web pages, mobile...

15+ min read

Deloitte Interview Questions and Answers for Technical Profiles

Deloitte is a leading global professional services firm with over 345,000 employees in more than 150 countries. The company offers a variety of...

13 min read

ION Group Interview Questions and Answers for Technical Profiles

ION Group is a global technology leader, providing innovative solutions for digital transformation and delivering cutting-edge services for...

12 min read

KPMG Interview Questions and Answers for Technical Profiles

KPMG is one of the Big Four accounting firms, and it offers a variety of technical roles in areas such as auditing, tax, consulting, and technology....

15+ min read

Article Tags : Interview Questions JavaScript Web Technologies interview-preparation

+2 More

R Language
Android Tutorial

Competitive Programming

Data Science & ML
Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies
HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
NodeJs
Bootstrap
Tailwind CSS

Python Tutorial
Python Programming Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Computer Science
GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps
Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design
High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects
Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar

Commerce
Accountancy
Business Studies
Economics
Management
HR Management
Finance
Income Tax

Databases
SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB

Preparation Corner
Company-Wise Recruitment Process
Resume Templates
Aptitude Preparation
Puzzles
Company-Wise Preparation
Companies

Colleges

Competitive Exams

JEE Advanced
UGC NET
UPSC
SSC CGL
SBI PO
SBI Clerk
IBPS PO
IBPS Clerk

More Tutorials

Software Development
Software Testing
Product Management
Project Management
Linux
Excel
All Cheat Sheets
Recent Articles

Free Online Tools

Typing Test
Image Editor
Code Formatters
Code Converters
Currency Converter
Random Number Generator
Random Password Generator

Write & Earn

Write an Article
Improve an Article
Pick Topics to Write
Share your Experiences
Internships

DSA/Placements

DSA - Self Paced Course
DSA in JavaScript - Self Paced Course
DSA in Python - Self Paced
C Programming Course Online - Learn C with Data Structures
Complete Interview Preparation
Master Competitive Programming
Core CS Subject for Interview Preparation
Mastering System Design: LLD to HLD
Tech Interview 101 - From DSA to System Design [LIVE]
DSA to Development [HYBRID]
Placement Preparation Crash Course [LIVE]

Development/Testing

JavaScript Full Course
React JS Course
React Native Course
Django Web Development Course
Complete Bootstrap Course
Full Stack Development - [LIVE]
JAVA Backend Development - [LIVE]
Complete Software Testing Course [LIVE]
Android Mastery with Kotlin [LIVE]

Machine Learning/Data Science

Complete Machine Learning & Data Science Program - [LIVE]
Data Analytics Training using Excel, SQL, Python & PowerBI - [LIVE]
Data Science Training Program - [LIVE]
Mastering Generative AI and ChatGPT
Data Science Course with IBM Certification

Programming Languages

C Programming with Data Structures
C++ Programming Course
Java Programming Course
Python Full Course

Clouds/Devops

DevOps Engineering
AWS Solutions Architect Certification
Salesforce Certified Administrator Course

GATE

GATE CS & IT Test Series - 2025
GATE DA Test Series 2025
GATE CS & IT Course - 2025
GATE DA Course 2025