



Published in JavaScript in Plain English

This is your **last** free member-only story this month. [Upgrade for unlimited access.](#)



AI - @thenaubit [Follow](#)

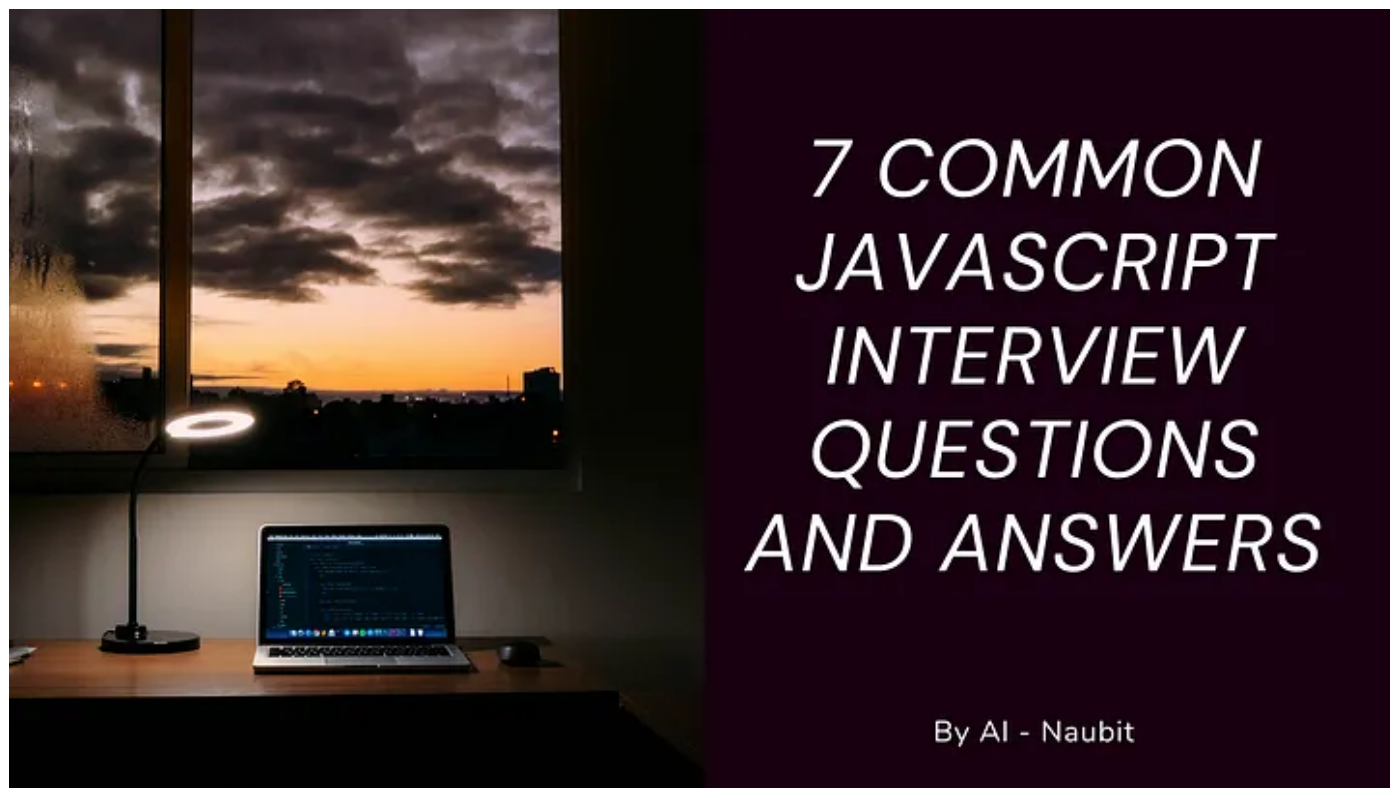
Jan 25 · 4 min read · · [Listen](#)



Save



Ace Your Next JavaScript Interview With These 7 Common Questions and Answers



In this article, I will cover the most common questions you may encounter during a **JavaScript interview**, providing detailed answers and examples to help you stand out

from the competition. Whether you are a beginner or an experienced developer, this guide will give you the confidence you need to **impress your interviewer and land the job**.

1 What is JavaScript, and how is it different from Java? 🤔

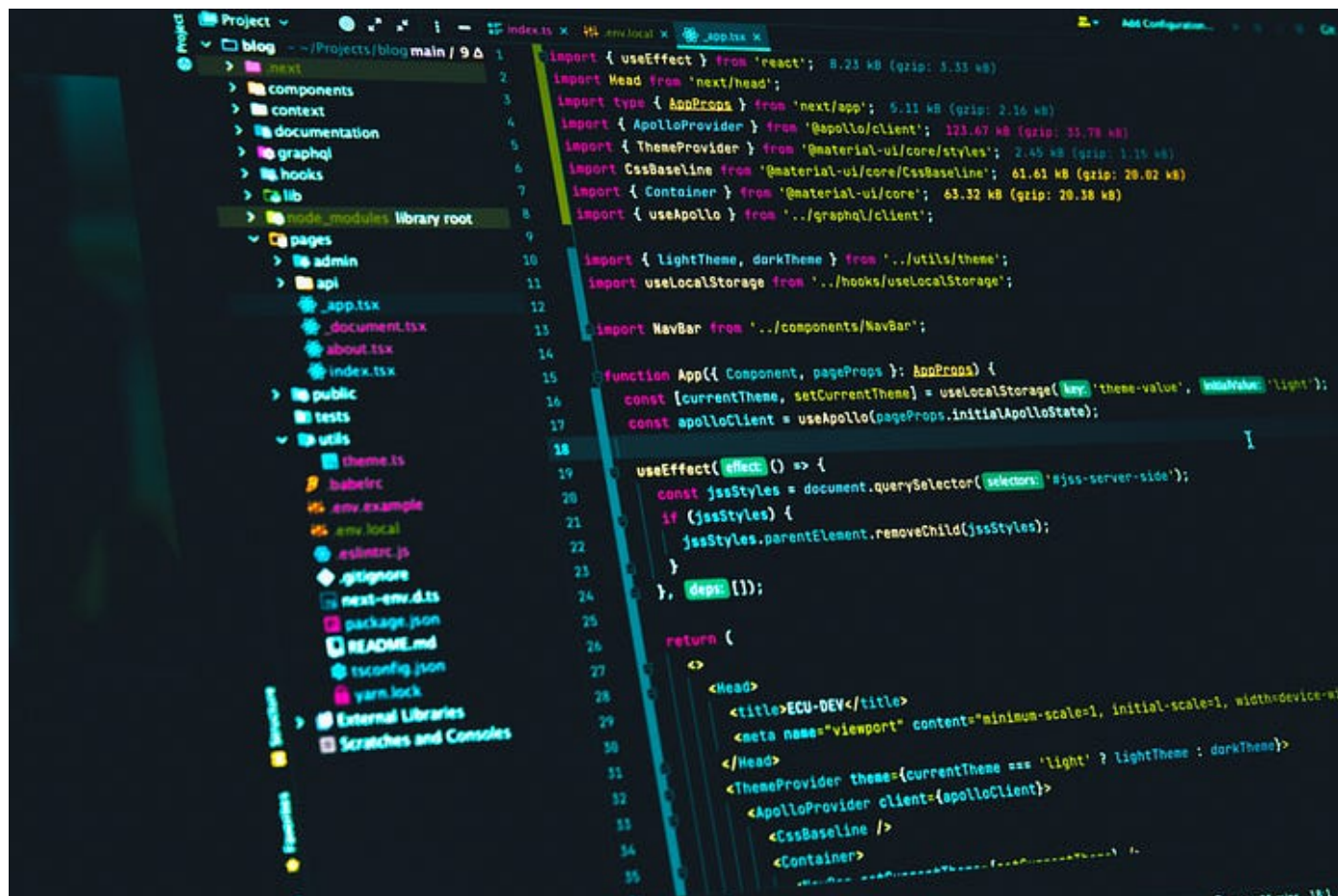


Photo by [Juanjo Jaramillo](#) on [Unsplash](#)

JavaScript and Java are both programming languages, but they are used for different purposes and have some key differences.

JavaScript is a scripting language primarily used to **create interactive and dynamic websites**. At the same time, Java is a general-purpose programming language that can be used to create a wide range of applications, including mobile and desktop apps, web applications, and backend systems.

The browser executes JavaScript code, while Java code is typically compiled and executed on a virtual machine or a specific device.

2 What is the difference between let and var?



Photo by [Nubelson Fernandes](#) on [Unsplash](#)

`var` and `let` are used to declare variables in JavaScript, but they have some key differences in terms of hoisting and scoping. `var` variables are hoisted to the top of their scope and are accessible throughout the entire scope. `let` variables, on the other hand, are **only accessible** within the block they were declared in and do not get hoisted

```
// var hoisting example
console.log(x); // undefined
var x = 5;
console.log(x); // 5

// let hoisting example
console.log(y); // ReferenceError: y is not defined
let y = 5;
console.log(y); // 5
```

3 What is the difference between == and ===?

== and === are used for equality comparison in JavaScript, but they behave differently regarding type coercion. == compares values after type coercion, while === compares values without type coercion.

```
console.log(5 == "5"); // true
console.log(5 === "5"); // false
```

It's generally recommended to use === in most cases to avoid unexpected behavior due to type coercion.

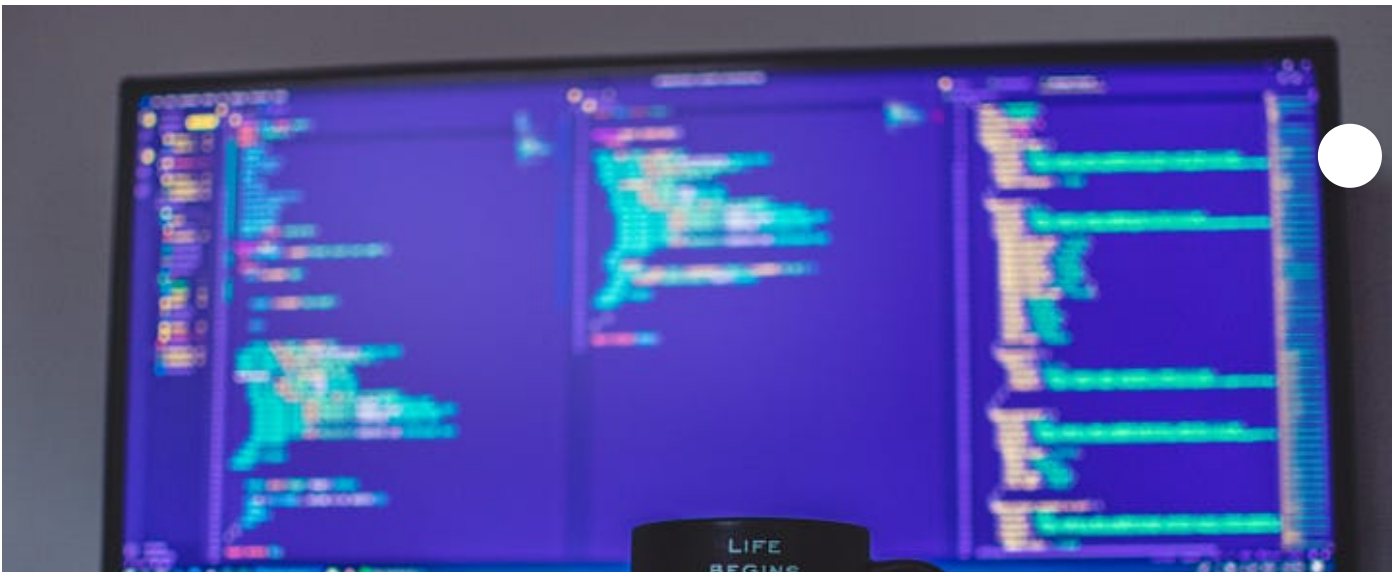
4 What is closure in JavaScript?

A closure in JavaScript is a function that has access to the variables in its parent scope, even after the parent function has returned.

```
function outerFunction(x) {
  return function innerFunction() {
    console.log(x);
  }
}
let closureFunc = outerFunction(5);
closureFunc(); // 5
```

In this example, the innerFunction has access to the variable x from its parent scope, the outerFunction.

5 What is the difference between call and apply?



Open in app ↗

Resume Membership



Search Medium



Photo by [Tudor Baci](#) on [Unsplash](#)


Both `call` and `apply` **are used to call a function** and set the value of `this` inside the function, but they have different syntaxes. `call` takes in the value of `this` as the first argument, followed by any additional arguments for the function.

```
let obj = {name: "John"};
function sayName() {
  console.log(this.name);
}
sayName.call(obj); // John
```

`apply` also takes in the value of `this` as the first argument, but **it expects the additional arguments for the function to be passed in as an array.**


```
let obj = {name: "John"};
function sayName(age) {
  console.log(this.name + " is " + age + " years old.");
}
sayName.apply(obj, [30]); // John is 30 years old.
```

6 What is an event loop in JavaScript?

JavaScript uses an event loop to  14 |  3 | ... handle events as they occur. The event loop constantly checks the message queue and the call stack. When a function is called, it is added to the call stack and executed.

If any events in the message queue need to be handled, they are added to the call stack and executed. This allows JavaScript to **handle multiple events and function calls simultaneously** without freezing the main thread.

7 What is the difference between synchronous and asynchronous code?

Synchronous code is **executed line by line**, and the following line of code can't be executed until the current line is finished.

Asynchronous code, on the other hand, **allows for multiple operations to occur at the same time**. In JavaScript, this can be achieved through callbacks or promises.

```
// synchronous code
console.log("Start");
let x = 0;
for (let i = 0; i < 1000000000; i++) {
  x += i;
}
console.log("End");

// asynchronous code using callbacks
console.log("Start");
let x = 0;
setTimeout(function() {
  for (let i = 0; i < 1000000000; i++) {
    x += i;
  }
  console.log("End");
}, 1000000000);
```

```
}, 0);  
console.log("Moving on...");
```

In the asynchronous example, the for loop is executed after the “*Moving on...*” message is logged, allowing other code to continue executing.

You have made it to the end of this guide and are now well-equipped with the knowledge and tools **necessary to ace your next JavaScript interview**. Remember to practice, stay up to date with the latest features and best practices, and, most importantly, **be confident**. I believe in you and know you will knock your interview out of the park. Good luck! 🍀

🌐 Let's Connect!

- My Twitter: [@thenaubit](#)
- My Substack

More content at [PlainEnglish.io](#). Sign up for our [free weekly newsletter](#). Follow us on [Twitter](#), [LinkedIn](#), [YouTube](#), and [Discord](#).

Interested in scaling your software startup? Check out [Circuit](#).

JavaScript

Javascript Tips

Javascript Development

Coding Interviews

Web Development

Get an email whenever AI - @thenaubit publishes.

Emails will be sent to amaurypv@gmail.com. [Not you?](#)

