

javascript interview questions with answers

JavaScript interview questions with answers are an essential part of the hiring process for many tech companies. JavaScript is a versatile and widely-used programming language, primarily known for its role in web development. As companies seek candidates with strong JavaScript skills, they often include a variety of questions that test both fundamental and advanced knowledge of the language. In this article, we will explore some of the most common JavaScript interview questions, along with detailed answers and explanations.

Basic JavaScript Interview Questions

1. What is JavaScript?

JavaScript is a high-level, dynamic, untyped, and interpreted programming language. It is a core technology of the World Wide Web, alongside HTML and CSS. JavaScript enables interactive web pages and is an essential part of web applications. It is also used on the server-side with environments like Node.js.

2. What are the data types supported by JavaScript?

JavaScript supports several data types, which can be categorized into two main groups: primitive and non-primitive.

Primitive Data Types:

- String: Represents text. Example: `"Hello, World!"`
- Number: Represents both integer and floating-point numbers. Example: `42` or `3.14`
- Boolean: Represents a logical entity and can have two values: `true` or `false`.
- Undefined: A variable that has been declared but has not yet been assigned a value.
- Null: Represents the intentional absence of any object value.
- Symbol (ES6): A unique and immutable data type used primarily as object property keys.
- BigInt (ES11): A numeric data type that can represent integers with arbitrary precision.

Non-Primitive Data Type:

- Object: A collection of key-value pairs. Example: `{name: "Alice", age: 25}`

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

The difference between `==` and `===` lies in how they compare values.

- `==` (Equality Operator): This operator checks for value equality with type coercion. It converts the operands to the same type before making the comparison.

Example:

```
``javascript
console.log(5 == '5'); // true
``
```

- `===` (Strict Equality Operator): This operator checks for both value and type equality without performing type coercion.

Example:

```
``javascript
console.log(5 === '5'); // false
``
```

Intermediate JavaScript Interview Questions

4. 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. This means that a closure can "remember" the environment in which it was created.

Example of Closure:

```
``javascript
function outerFunction() {
  let outerVariable = 'I am from outer function';

  function innerFunction() {
    console.log(outerVariable);
  }

  return innerFunction;
}

const closureFunction = outerFunction();
closureFunction(); // Output: I am from outer function
``
```

...

5. Explain the concept of "hoisting".

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their containing scope during the compile phase. In other words, you can use functions and variables before they are declared.

Example of Hoisting:

```
```javascript
console.log(myVar); // Output: undefined
var myVar = 10;
```

```
myFunction(); // Output: "Hello, World!"
function myFunction() {
 console.log("Hello, World!");
}
...

```

## 6. What are promises in JavaScript?

Promises are objects that represent the eventual completion (or failure) of an asynchronous operation and its resulting value. A promise can be in one of three states: pending, fulfilled, or rejected.

Example of a Promise:

```
```javascript
let myPromise = new Promise((resolve, reject) => {
  const success = true; // Change to false to see rejection
  if (success) {
    resolve("Operation was successful!");
  } else {
    reject("Operation failed.");
  }
});

myPromise
  .then(response => console.log(response))
  .catch(error => console.error(error));
...

```

Advanced JavaScript Interview Questions

7. What is the "this" keyword in JavaScript?

The `this` keyword in JavaScript refers to the context in which a function is called. Its value can vary depending on how the function is invoked.

Examples of `this`:

- In a regular function, `this` refers to the global object (window in browsers) if not in strict mode.
- In an object method, `this` refers to the object that owns the method.
- In a constructor function, `this` refers to the newly created object.
- In an arrow function, `this` retains the value of the enclosing lexical context.

8. What are JavaScript callbacks, and how do they work?

A callback is a function that is passed as an argument to another function and is executed after some kind of event or operation. Callbacks are often used for asynchronous operations, such as API calls or timers.

Example of a Callback:

```
```\javascript
function fetchData(callback) {
 setTimeout(() => {
 console.log("Data fetched!");
 callback();
 }, 2000);
}

fetchData(() => {
 console.log("Callback executed after data fetching.");
});
```
```

9. Explain the concept of "event delegation".

Event delegation is a technique in JavaScript where a single event listener is added to a parent element instead of adding individual listeners to each child element. This leverages event bubbling, where events propagate from child elements to parent elements.

Example of Event Delegation:

```
```html
```

- Item 1
- Item 2
- Item 3

...

## Conclusion

JavaScript is a rich language with many features that can be tested during interviews. From basic concepts to advanced topics, candidates need to be prepared for a variety of questions. Understanding JavaScript's core principles, including closures, hoisting, promises, and the `this` keyword, is crucial for success in technical interviews. By practicing these common interview questions and their answers, candidates can build confidence and improve their chances of securing a position in the competitive tech industry.

## Frequently Asked Questions

### What is the difference between '==' and '===' in JavaScript?

'==' is the equality operator that performs type coercion, meaning it converts the operands to the same type before comparing them. '===' is the strict equality operator that checks both the value and type without any coercion.

### What are closures in JavaScript?

Closures are functions that have access to the outer scope, even after the outer function has finished executing. They allow for private variables and encapsulation.

### What is the purpose of 'use strict' in JavaScript?

'use strict' is a directive that enables strict mode, which helps catch common coding errors and 'unsafe' actions such as defining global variables unintentionally.

### Explain event delegation in JavaScript.

Event delegation is a technique that allows you to attach a single event listener to a parent element instead of multiple listeners on child elements. This improves performance and allows for dynamic content

handling.

## **What are promises in JavaScript?**

Promises are objects that represent the eventual completion (or failure) of an asynchronous operation and its resulting value. They provide a cleaner alternative to callback functions for handling asynchronous code.

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

'var' is function-scoped or globally-scoped and can be re-declared. 'let' is block-scoped and cannot be re-declared within the same block. 'const' is also block-scoped but creates immutable bindings, meaning the variable cannot be reassigned.

## **How does the 'this' keyword work in JavaScript?**

The 'this' keyword refers to the context in which a function is called. Its value varies depending on how the function is invoked: it can refer to the global object, an object instance, or be undefined in strict mode.

## **What is hoisting in JavaScript?**

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their containing scope during the compile phase. This means you can use variables and functions before they are declared.

## **What are template literals in JavaScript?**

Template literals are string literals that allow for embedded expressions. They are enclosed by backticks (`) and can include multi-line strings and interpolation using the `\${expression}` syntax.

## **What is the purpose of the 'async' and 'await' keywords?**

'async' is used to define an asynchronous function that returns a promise. 'await' can be used within an async function to pause execution until the promise is resolved, making asynchronous code easier to read and write.

## **[Javascript Interview Questions With Answers](#)**

Javascript Interview Questions With Answers

## Related Articles

- [itbs practice test 7th grade](#)
- [item response theory for psychologists](#)
- [john deere 275 disc mower parts diagram](#)

[Back to Home](#)