Top 30 JavaScript Interview Questions & Answers

1. What is the difference between let, const, and var?

let and const are block-scoped while var is function-scoped. const cannot be reassigned.

2. What are closures in JavaScript?

Closures are functions that remember their lexical scope even when called outside of it.

3. Explain the concept of Promises in JavaScript.

Promises represent the eventual result of an asynchronous operation. They can be pending, fulfilled, or rejected.

4. What is the difference between == and ===?

== compares values with type conversion, while === compares values without type conversion.

5. How does the this keyword work in JavaScript?

The value of this is determined by how a function is called. It can refer to an object, global context, etc.

6. What are arrow functions, and how do they differ from regular functions?

Arrow functions are more concise and have no this binding. They inherit this from the enclosing scope.

7. What is event delegation?

A technique where a parent element handles events for its child elements.

8. Explain the concept of hoisting in JavaScript.

Hoisting is JavaScript's default behavior of moving declarations to the top of their scope.

9. What are IIFEs (Immediately Invoked Function Expressions)?

A function that is executed immediately after it is defined.

10. What is the event loop in JavaScript?

The event loop handles asynchronous code by managing the execution of tasks.

11. Explain the difference between null and undefined.

null is an intentional absence of value, while undefined is the default value for uninitialized variables.

12. What is the use of the bind() method?

bind() creates a new function that, when called, has a specific this value.

13. What is a callback function?

A function passed into another function to be executed later.

14. How do call() and apply() methods differ?

call() takes arguments separately, while apply() takes arguments as an array.

15. What is the difference between synchronous and asynchronous programming?

Synchronous code is executed sequentially, while asynchronous code allows other tasks to be processed before completion.

16. What are JavaScript data types?

JavaScript has primitive types: number, string, boolean, null, undefined, symbol, and object types.

17. Explain the concept of prototypal inheritance.

Prototypal inheritance allows objects to inherit properties from other objects.

18. What is the purpose of the Array.prototype.map() method?

It creates a new array by applying a function to each element of the array.

19. What is the difference between forEach() and map()?

forEach() processes each array element without returning a value, while map() returns a new array with transformed elements.

20. How do you handle errors in JavaScript using try, catch, and finally?

try executes code, catch handles errors, and finally executes code after try/catch regardless of outcome.

21. What is a promise chain?

A series of promises where each is dependent on the resolution of the previous one.

22. What is async/await, and how does it relate to Promises?

async/await is syntactic sugar over Promises, allowing asynchronous code to be written more like synchronous code.

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

A function that takes other functions as arguments or returns them.

24. Explain the concept of destructuring assignment.

Destructuring allows unpacking values from arrays or objects into variables.

25. What is the difference between spread operator and rest parameter?

Spread expands elements, while rest gathers elements into an array.

26. What is the purpose of Object.assign()?

It copies properties from one or more source objects to a target object.

27. How does JavaScript handle asynchronous operations?

It uses callbacks, Promises, and async/await.

28. What is the fetch API, and how is it used?

fetch() is used to make HTTP requests, returning a Promise that resolves to the response.

29. Explain the concept of currying in JavaScript.

Currying transforms a function so it can be called with arguments one at a time.

30. What are generators and how do they differ from regular functions?

Generators are functions that can pause and resume their execution using yield.