# JavaScript Interview Questions

1. What is JavaScript?
    - JavaScript is a high-level, interpreted programming language used for web development.
2. What are the key features of JavaScript?
    - JavaScript is lightweight, interpreted, and versatile. It supports both object-oriented and procedural programming.
3. How do you include JavaScript in an HTML file?
    - You can include JavaScript in an HTML file using the `<script>` tag, either in the head or body of the HTML document.
4. What is the purpose of the `defer` attribute in the `<script>` tag?
    - The `defer` attribute indicates that the script should be executed after the HTML document has been parsed.
5. What is the purpose of the `async` attribute in the `<script>` tag?
    - The `async` attribute specifies that the script should be executed asynchronously, without blocking the parsing of the HTML document.
6. What are JavaScript data types?
    - JavaScript has several data types, including number, string, boolean, object, array, null, and undefined.
7. How do you check the data type of a variable?
    - You can use the `typeof` operator to check the data type of a variable.
8. Explain the difference between `null` and `undefined` in JavaScript.
    - `null` is a value that represents the intentional absence of any object value, while `undefined` is a variable that has been declared but not assigned a value.
9. What is hoisting in JavaScript?
    - Hoisting is a JavaScript behavior where variable and function declarations are moved to the top of their containing scope during compilation.
10. What is closure in JavaScript?
    - A closure is a function that has access to its own scope, the outer function's scope, and the global scope, allowing it to capture and remember values from its lexical environment.

**Variables and Scope:**

11. What is the difference between `var`, `let`, and `const`?
    - `var` is function-scoped, `let` and `const` are block-scoped. `const` cannot be reassigned, while `let` can be.
12. What is block scope?
    - Block scope is a region of code where a variable is only accessible within that block.
13. What is the scope chain in JavaScript?
    - The scope chain is the hierarchy of scopes in JavaScript, which determines the order in which variables are looked up.
14. What is variable hoisting?

- Variable hoisting is a JavaScript behavior where variable declarations are moved to the top of their containing function or block during compilation.

15. What is the global scope in JavaScript?
    - The global scope is the outermost scope in JavaScript, accessible from any part of the code.
16. Explain the concept of "shadowing" in variable scope.
    - Shadowing occurs when a variable declared within a function or block has the same name as a variable in an outer scope, effectively hiding the outer variable.
17. How do you declare a constant in JavaScript?
    - You can declare a constant using the `const` keyword.

javascriptCopy code

```javascript
const PI 3.14159
```

18. Can you reassign a value to a constant declared with `const`?
    - No, you cannot reassign a value to a constant declared with `const`.
19. What is the Temporal Dead Zone (TDZ)?
    - The Temporal Dead Zone is the period between entering scope and the variable being declared where accessing the variable results in a `ReferenceError`.

**Data Types and Operators:**

20. What are the six primitive data types in JavaScript?
    - The six primitive data types in JavaScript are undefined, null, boolean, number, string, and symbol.
21. How do you convert a string to a number in JavaScript?
    - You can use `parseInt()` or `parseFloat()` functions to convert a string to a number.

javascriptCopy code

```javascript
let "123" let parseInt
```

22. What is type coercion in JavaScript?
    - Type coercion is the automatic conversion of values from one data type to another by the JavaScript engine.
23. What is the difference between `==` and `===` in JavaScript?
    - `==` checks for equality after type coercion, while `===` checks for strict equality without type coercion.
24. What is NaN in JavaScript?
    - NaN stands for "Not-a-Number" and is a special value representing an unrepresentable or undefined value in numeric operations.
25. How do you check if a variable is NaN?
    - You can use the `isNaN()` function to check if a variable is NaN.

javascriptCopy code

```javascript
isNaN
```

26. What is the ternary conditional operator in JavaScript?
    - The ternary operator (`? :`) is a shorthand for an `if-else` statement, used to assign a value based on a condition.

javascriptCopy code

```javascript
let
```

27. What is a template literal in JavaScript?

- A template literal is a string literal that allows for embedded expressions, denoted by backticks (`).

javascriptCopy code

```
let          "John"  let           `Hello, ${name}!`
```

28. Explain the `typeof` operator in JavaScript.
    - The `typeof` operator returns a string indicating the data type of a given expression.

javascriptCopy code

```
typeof
```

29. What is the purpose of the `instanceof` operator in JavaScript?
    - The `instanceof` operator checks if an object is an instance of a particular class or constructor function.

javascriptCopy code

```
        instanceof Constructor
```

30. How do you compare two objects for equality in JavaScript?
    - JavaScript objects are compared by reference, not by their content. To compare their content, you would need to implement a custom comparison function.

**Functions:**

31. What is a JavaScript function?
    - A function is a reusable block of code that performs a specific task or calculates a value.
32. How do you declare a function in JavaScript?
    - You can declare a function using the `function` keyword.

javascriptCopy code

```
function functionName
```

33. What is a function expression?
    - A function expression is a function that is defined within an expression and can be assigned to a variable.

javascriptCopy code

```
let        function        return
```

34. What is a named function expression?
    - A named function expression is a function expression with a name, which can be used for self-reference and debugging.

javascriptCopy code

```
let             function multiply        return
```

35. What is an anonymous function?
    - An anonymous function is a function without a name.

javascriptCopy code

```
let         function        return
```

36. What is the difference between a function declaration and a function expression?
    - A function declaration is hoisted, meaning it can be used before it's declared, while a function expression is not hoisted and must be defined before use.
37. What is a higher-order function in JavaScript?
    - A higher-order function is a function that takes one or more functions as arguments or returns a function as its result.
38. What is a callback function in JavaScript?

- A callback function is a function that is passed as an argument to another function and is executed after that function completes.

39. What is a closure in JavaScript, and how is it useful?
    - A closure is a function that has access to its own scope, the outer function's scope, and the global scope, allowing it to capture and remember values from its lexical environment. Closures are useful for creating private variables and maintaining state.
40. What is the purpose of the `apply()` and `call()` methods in JavaScript?
    - The `apply()` and `call()` methods are used to invoke functions and set the `this` value within the function. They are often used to borrow methods from one object and apply them to another.

**Arrays:**

41. How do you create an empty array in JavaScript?
    - You can create an empty array using square brackets.

javascriptCopy code

```
let
```

42. What is the length property of an array?
    - The `length` property of an array represents the number of elements in the array.
43. How do you access elements in an array?
    - You can access elements in an array using square bracket notation with the index of the element.

javascriptCopy code

```
let 1 2 3 let 0
```

44. How do you add elements to an array?
    - You can add elements to an array using the `push()` method.

javascriptCopy code

```
let 1 2 3 push 4
```

45. How do you remove elements from an array?
    - You can remove elements from an array using the `pop()` method to remove the last element or the `splice()` method to remove elements at specific indices.

javascriptCopy code

```
let 1 2 3 4 pop splice 1 1
```

46. How do you iterate over elements in an array?
    - You can iterate over elements in an array using `for` loops, `forEach()`, `map()`, `filter()`, `reduce()`, and other array methods.

javascriptCopy code

```
let 1 2 3 forEach function console log
```

47. What is the difference between `forEach()` and `map()`?
    - `forEach()` iterates over an array and executes a function for each element without changing the original array. `map()` also iterates over an array but returns a new array based on the results of the function.
48. How do you check if an element exists in an array?
    - You can use the `indexOf()` method or the `includes()` method to check if an element exists in an array.

javascriptCopy code

```
let 1 2 3 let indexOf 2 1 let includes 4
```

49. What are JavaScript array methods for adding and removing elements?
    - Methods for adding elements include `push()`, `unshift()`, and `splice()`. Methods for removing elements include `pop()`, `shift()`, and `splice()`.
50. Explain the concept of multidimensional arrays in JavaScript.
    - Multidimensional arrays in JavaScript are arrays of arrays, allowing you to create matrices or tables of data.

javascriptCopy code
```
let 1 2 3 4 5 6 7 8 9 let 1 2
```

**Objects:**

51. What is an object in JavaScript?
    - An object is a collection of key-value pairs, where keys are strings (or Symbols) and values can be of any data type.
52. How do you create an empty object in JavaScript?
    - You can create an empty object using curly braces.

javascriptCopy code
```
let
```

53. How do you access properties of an object?
    - You can access properties of an object using dot notation or bracket notation.

javascriptCopy code
```
let name "John" age 30 let let
"age"
```

54. How do you add properties to an object?
    - You can add properties to an object by assigning values using dot notation or bracket notation.

javascriptCopy code
```
let "John" "age" 30
```

55. How do you remove properties from an object?
    - You can remove properties from an object using the `delete` keyword.

javascriptCopy code
```
let name "John" age 30 delete
```

56. What is the purpose of the `for...in` loop in JavaScript?
    - The `for...in` loop is used to iterate over the enumerable properties of an object.

javascriptCopy code
```
for let in
```

57. What is JSON, and how is it related to JavaScript objects?
    - JSON (JavaScript Object Notation) is a lightweight data interchange format that is a text representation of JavaScript objects. It is often used for data exchange between a client and a server.
58. What is object destructuring in JavaScript?
    - Object destructuring is a feature that allows you to extract properties from objects and assign them to variables in a more concise way.

javascriptCopy code
```
let name "John" age 30 let
```

59. How do you check if an object has a specific property?

- You can use the `hasOwnProperty()` method or the `in` operator to check if an object has a specific property.

javascriptCopy code
```
let name "John" age 30 let hasOwnProperty "name" let
"gender" in
```

60. What is a constructor function in JavaScript, and how do you create objects using constructors?
    - A constructor function is a function that is used to create and initialize objects. You can create objects using constructors by calling them with the `new` keyword.

javascriptCopy code
```
function Person this this let new Person "John"
30
```

**Prototypes and Inheritance:**

61. What is prototype-based inheritance in JavaScript?
    - In prototype-based inheritance, objects can inherit properties and methods from other objects through their prototype chain.
62. What is the prototype chain in JavaScript?
    - The prototype chain is the mechanism by which objects in JavaScript inherit properties and methods from their prototype object.
63. How do you set up inheritance in JavaScript?
    - You can set up inheritance by using constructor functions, the `Object.create()` method, or the class syntax introduced in ES6.
64. What is the purpose of the `prototype` property in a constructor function?
    - The `prototype` property of a constructor function is used to define methods and properties that will be shared by all instances created from that constructor.

javascriptCopy code
```
function Person this Person prototype function
console log `Hello, my name is ${this.name}`
```

65. What is the difference between `prototype` and `__proto__` in JavaScript?
    - `prototype` is a property of constructor functions used to define shared methods and properties, while `__proto__` is an object's internal reference to its prototype.
66. How do you check if an object is an instance of a particular constructor?
    - You can use the `instanceof` operator to check if an object is an instance of a particular constructor.

javascriptCopy code
```
let new Person "John" let instanceof Person
```

67. What is the purpose of the `Object.create()` method in JavaScript?
    - The `Object.create()` method creates a new object with the specified prototype object, allowing you to set up prototype-based inheritance.

javascriptCopy code
```
let name "John" let Object create
```

**Error Handling:**

68. What is an exception in JavaScript?

- An exception is an event that occurs during the execution of a program and disrupts the normal flow of code.

69. How do you handle exceptions in JavaScript?
    - Exceptions can be handled using `try...catch` blocks. Code that might throw an exception is placed inside a `try` block, and if an exception is thrown, it can be caught and handled in the `catch` block.

javascriptCopy code

```
try catch
```

70. What is the purpose of the `finally` block in a `try...catch...finally` statement?
    - The `finally` block is used to specify code that should be executed regardless of whether an exception is thrown or caught. It is often used for cleanup tasks.

javascriptCopy code

```
try catch finally
```

71. What is the `throw` statement in JavaScript?
    - The `throw` statement is used to manually generate an exception. You can throw an error object or any other value.

javascriptCopy code

```
throw new Error "This is an error message"
```

72. What is an error object in JavaScript, and what are some common error types?
    - An error object is an object that represents an error in JavaScript. Common error types include `Error`, `SyntaxError`, `TypeError`, and `ReferenceError`, among others.
73. What is the purpose of the `Error` constructor in JavaScript?
    - The `Error` constructor is used to create custom error objects with a specific error message and optional error code.

javascriptCopy code

```
throw new Error "Custom error message"
```

**DOM Manipulation:**

74. What is the Document Object Model (DOM) in web development?
    - The DOM is a programming interface for web documents that represents the structure of an HTML or XML document as a tree of objects, allowing JavaScript to interact with and manipulate the document.
75. How do you select an HTML element in JavaScript?
    - You can select HTML elements using methods like `getElementById()`, `getElementsByClassName()`, `getElementsByTagName()`, `querySelector()`, and `querySelectorAll()`.
76. How do you change the content of an HTML element using JavaScript?
    - You can change the content of an HTML element by accessing its `innerHTML` or `textContent` property.

javascriptCopy code

```
let document getElementById "myElement" "New content"
```

77. How do you add and remove HTML elements using JavaScript?
    - You can add new elements using the `createElement()` method and remove elements using the `remove()` method.

javascriptCopy code

```
let document createElement "div" appendChild
remove
```

78. How do you handle events in JavaScript?
    - You can handle events in JavaScript by adding event listeners to HTML elements using methods like `addEventListener()`.

javascriptCopy code
```
addEventListener "click" function
```

79. What is event delegation in JavaScript?
    - Event delegation is a technique where a single event listener is placed on a common ancestor of multiple elements, allowing you to handle events on those elements without attaching individual event listeners.
80. What is the `this` keyword in the context of event handlers?
    - In the context of event handlers, the `this` keyword refers to the element that triggered the event.
81. How do you prevent the default behavior of an event in JavaScript?
    - You can prevent the default behavior of an event using the `event.preventDefault()` method within an event handler.

javascriptCopy code
```
addEventListener "click" function preventDefault
```

82. What is event bubbling in JavaScript?
    - Event bubbling is the propagation of events from nested elements to their parent elements in the DOM tree.
83. What is event capturing in JavaScript?
    - Event capturing is the propagation of events from parent elements to their nested child elements in the DOM tree.
84. What is the purpose of the `stopPropagation()` method in JavaScript?
    - The `stopPropagation()` method is used to stop the propagation of an event in either the bubbling or capturing phase.

javascriptCopy code
```
addEventListener "click" function stopPropagation
```

**Asynchronous Programming:**

85. What is asynchronous programming in JavaScript?
    - Asynchronous programming is a programming paradigm that allows multiple tasks to be executed concurrently without blocking the main thread.
86. What are callbacks in JavaScript, and how do you use them?
    - Callbacks are functions that are passed as arguments to other functions and are executed after the completion of an asynchronous operation.

javascriptCopy code
```
function fetchData setTimeout function callback
1000
```

87. What is a Promise in JavaScript, and how does it work?
    - A Promise is an object representing the eventual completion or failure of an asynchronous operation. It provides a more structured way to handle asynchronous code and can be in one of three states: pending, fulfilled, or rejected.

javascriptCopy code

```
let new Promise function if
resolve else reject
```

88. What are the `then()` and `catch()` methods in a Promise?
    - The `then()` method is used to specify what to do when a Promise is resolved, while the `catch()` method is used to specify what to do when a Promise is rejected.

javascriptCopy code

```
then function catch function
```

89. What is the purpose of the `async` and `await` keywords in JavaScript?
    - The `async` keyword is used to define asynchronous functions, and the `await` keyword is used within an `async` function to pause execution until a Promise is resolved or rejected.

javascriptCopy code

```
async function fetchData let await fetchDataAsync
```

90. How do you handle multiple Promises concurrently in JavaScript?
    - You can use `Promise.all()` to handle multiple Promises concurrently. It waits for all Promises to be resolved and returns an array of their results.

javascriptCopy code

```
let Promise all then function
catch function
```

**Modules and ES6 Features:**

91. What are ES6 modules in JavaScript?
    - ES6 modules are a way to organize and encapsulate code into separate files, each containing its own module. Modules can export and import functions, variables, and classes.
92. How do you export and import modules in JavaScript?
    - To export from a module, you can use the `export` keyword. To import in another module, you can use `import`.

javascriptCopy code

```
export function sayHello console log "Hello!" import
from "./moduleA"
```

93. What is the purpose of the `default` keyword when exporting and importing modules?
    - The `default` keyword is used to specify the default export of a module, which can be imported without specifying a name.

javascriptCopy code

```
export default function console log "Default export" import
from "./moduleA"
```

94. What are arrow functions in JavaScript?
    - Arrow functions are a concise way to define functions in JavaScript. They have a shorter syntax and automatically capture the `this` value from the surrounding context.

javascriptCopy code

```
let add
```

95. What is destructuring assignment in ES6?

- Destructuring assignment is a feature that allows you to extract values from arrays and objects and assign them to variables in a more concise way.

javascriptCopy code

```
let 1 2 let x 10 y 20
```

96. What is the `let` and `const` block-scoping behavior in ES6?
    - Variables declared with `let` and `const` are block-scoped, meaning they are only accessible within the block where they are defined.
97. What is the spread operator (`...`) in JavaScript, and how is it used?
    - The spread operator is used to expand elements in arrays, objects, or function arguments. It allows you to clone arrays, merge objects, and pass multiple arguments to functions.

javascriptCopy code

```
let 1 2 3 let
```

98. What is the rest parameter (`...`) in JavaScript, and how is it used?
    - The rest parameter is used in function declarations to collect multiple arguments into a single array-like parameter.

javascriptCopy code

```
function sum return reduce 0
```

99. What are template literals in ES6, and how are they different from regular strings?
    - Template literals are strings enclosed in backticks (`) that allow for embedded expressions and multi-line strings.

javascriptCopy code

```
let "John" let `Hello, ${name}!`
```

100. What is the `class` syntax in ES6, and how is it used for object-oriented programming? - The `class` syntax in ES6 is used to define classes and constructor functions for object-oriented programming in a more structured and familiar way.

javascriptCopy code

```
class Person constructor this sayHello console log `Hello, my name is
${this.name}`
```

These are 100 JavaScript interview questions along with their answers. Be sure to study and understand each topic thoroughly to prepare for your JavaScript interviews.

AIPRM - ChatGPT Prompts