100 JavaScript Technical Implementation Questions

Array Methods & Data Processing

1. Write a function to find sum of 10 numbers using reduce method
2. Create a function that filters an array of ages to return only adults (age >= 18) using filter method
3. Write a function using map method to convert array of temperatures from Celsius to Fahrenheit
4. Implement a function that finds the maximum number in an array using reduce method
5. Create a function using forEach method to print each element of an array with its index
6. Write a function that uses find method to locate first student with grade above 90
7. Implement a function using some method to check if any number in array is negative
8. Create a function using every method to verify if all passwords in array have minimum 8 characters
9. Write a function that uses sort method to arrange array of objects by name property
10. Implement a function using includes method to check if specific fruit exists in fruits array

Object Manipulation & Methods

11. Write a function to add a new property 'fullName' to user object by combining firstName and lastName
12. Create a function that deletes a specific property from an object using delete operator
13. Implement a function using Object.keys() to count total number of properties in an object
14. Write a function using Object.values() to find sum of all numeric values in an object
15. Create a function using Object.entries() to convert object into array of key-value pairs
16. Implement a function that merges two objects using Object.assign() method
17. Write a function using hasOwnProperty() to check if object contains specific property
18. Create a function that clones an object using JSON.parse() and JSON.stringify()
19. Implement a function using Object.freeze() to make an object immutable
20. Write a function that uses Object.create() to create new object with specific prototype

String Methods & Manipulation

21. Write a function using split() method to convert comma-separated string into array
22. Create a function using join() method to combine array elements into single string with custom separator
23. Implement a function using substring() to extract first 5 characters from a string
24. Write a function using charAt() to find character at specific index in a string
25. Create a function using indexOf() to find position of first occurrence of substring

26. Implement a function using replace() to replace all spaces in string with hyphens
27. Write a function using toUpperCase() and toLowerCase() to convert string case
28. Create a function using trim() to remove whitespace from beginning and end of string
29. Implement a function using startsWith() to check if string begins with specific prefix
30. Write a function using endsWith() to verify if string ends with specific suffix

Functions & Scope Concepts

31. Write an arrow function that takes two parameters and returns their multiplication
32. Create a function expression that calculates area of rectangle using length and width
33. Implement a higher-order function that takes another function as parameter and executes it twice
34. Write a closure function that maintains private counter variable and returns increment function
35. Create a function with default parameters that greets user with custom or default message
36. Implement a function using rest parameters to calculate sum of unlimited numbers
37. Write a function that uses spread operator to combine multiple arrays into one
38. Create a recursive function to calculate factorial of given number
39. Implement a callback function that processes array elements and executes provided function on each
40. Write a function that returns another function (function factory pattern)

Conditional Logic & Control Flow

41. Write a function using if-else to determine grade (A, B, C, D, F) based on score
42. Create a function using switch statement to return day name based on day number (1-7)
43. Implement a function using ternary operator to check if number is even or odd
44. Write a function using logical AND (&&) operator to validate user login credentials
45. Create a function using logical OR (||) operator to provide default values for parameters
46. Implement a function using nested if statements to categorize age groups (child, teen, adult, senior)
47. Write a function using multiple conditions to determine shipping cost based on weight and distance
48. Create a function that uses short-circuit evaluation to avoid errors when accessing object properties
49. Implement a function using nullish coalescing operator (??) to handle null and undefined values
50. Write a function using optional chaining (?.) to safely access nested object properties

Loops & Iteration

51. Write a for loop to print numbers from 1 to 100 that are divisible by both 3 and 5
52. Create a while loop that continues until user enters correct password
53. Implement a do-while loop that asks for user input at least once
54. Write a for-in loop to iterate through object properties and print key-value pairs
55. Create a for-of loop to iterate through array and calculate sum of all elements
56. Implement nested loops to create multiplication table from 1 to 10
57. Write a loop with break statement to stop when specific condition is met
58. Create a loop with continue statement to skip certain iterations
59. Implement a loop that iterates backwards through an array from last to first element
60. Write a loop that processes 2D array (array of arrays) to find maximum value

Error Handling & Debugging

61. Write a try-catch block to handle JSON.parse() errors when parsing invalid JSON string
62. Create a function with try-catch-finally to handle file reading operations
63. Implement a function that throws custom error with specific message for invalid input
64. Write a function using console.log() to debug variable values at different execution points
65. Create a function that uses console.error() to log error messages with timestamps
66. Implement error handling for division by zero scenario in calculator function
67. Write a function that validates email format and throws error for invalid emails
68. Create a function with multiple catch blocks for different error types
69. Implement a function that handles async errors using try-catch with await
70. Write a function that logs execution time using console.time() and console.timeEnd()

Asynchronous JavaScript

71. Write a setTimeout function to execute code after 3 seconds delay
72. Create a setInterval function to execute code every 2 seconds and stop after 10 seconds
73. Implement a Promise that resolves after 2 seconds with success message
74. Write a function using Promise.then() and Promise.catch() to handle async operations
75. Create an async function that waits for user input using await keyword
76. Implement a function using Promise.all() to wait for multiple async operations to complete
77. Write a function using fetch() API to get data from REST endpoint and handle response
78. Create a function that uses Promise.race() to return result of fastest async operation
79. Implement error handling in async function using try-catch with await
80. Write a function that converts callback-based function to Promise-based function

DOM Manipulation

81. Write a function to create new HTML element and add it to specific parent using appendChild()
82. Create a function that changes text content of element using textContent property
83. Implement a function to add CSS class to element using classList.add() method
84. Write a function to remove element from DOM using removeChild() or remove() method
85. Create a function that gets element by ID and changes its background color
86. Implement a function to add event listener to button that alerts message when clicked
87. Write a function that creates HTML table dynamically from array of data
88. Create a function that toggles visibility of element using style.display property
89. Implement a function that validates form input and shows error message in specific div
90. Write a function that updates innerHTML of element with formatted HTML content

Advanced Concepts

91. Write a function using bind() method to set specific 'this' context for object method
92. Create a function using call() method to invoke function with different context and arguments
93. Implement a function using apply() method to pass array of arguments to another function
94. Write a function that demonstrates hoisting behavior with var, let, and const
95. Create a function that uses destructuring assignment to extract values from object and array
96. Implement a function using template literals with embedded expressions and multi-line strings
97. Write a function that uses Symbol to create unique object properties
98. Create a function using Map data structure to store key-value pairs with object keys
99. Implement a function using Set data structure to store unique values and remove duplicates
100. Write a function using WeakMap to associate metadata with objects without preventing garbage collection

Implementation Guidelines

For Each Question:

1. **Write Complete Function**: Include function name, parameters, and return statement
2. **Add Comments**: Explain what each part of code does
3. **Test with Examples**: Provide sample input and expected output
4. **Handle Edge Cases**: Consider empty inputs, null values, invalid data
5. **Follow Best Practices**: Use meaningful variable names and consistent formatting

Example Implementation Format:

```javascript
// Question 1: Write a function to find sum of 10 numbers using reduce method
function sumTenNumbers(numbers) {
    // Validate input array has 10 elements
    if (!Array.isArray(numbers) || numbers.length !== 10) {
        throw new Error('Input must be array of exactly 10 numbers');
    }

    // Use reduce to calculate sum
    const sum = numbers.reduce((accumulator, currentValue) => {
        return accumulator + currentValue;
    }, 0);

    return sum;
}

// Test the function
const testNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
console.log(sumTenNumbers(testNumbers)); // Expected output: 55
```

Learning Path:

- **Start with Array Methods (1-10)**: Master fundamental data processing
- **Move to Objects (11-20)**: Understand object manipulation
- **Practice Strings (21-30)**: Learn text processing techniques
- **Master Functions (31-40)**: Understand function concepts deeply
- **Learn Control Flow (41-50)**: Build logical thinking skills
- **Practice Loops (51-60)**: Understand iteration patterns
- **Handle Errors (61-70)**: Build robust applications
- **Async Programming (71-80)**: Master modern JavaScript
- **DOM Manipulation (81-90)**: Create interactive web pages
- **Advanced Concepts (91-100)**: Explore sophisticated JavaScript features