JavaScript Master Q&A: Variables, Operators & Expressions

Q1. [Interview] What are variables in JavaScript?

What are variables in JavaScript?

Ans: Variables are containers for storing data values.

```
let name = 'John';
```

Q2. [Interview] Difference between var, let, and const?

Difference between var, let, and const?

Ans: `var` is function-scoped; `let` and `const` are block-scoped. `const` cannot be reassigned.

```
var x = 5;
let y = 10;
const z = 15;
```

Q3. [Interview] What are operators in JavaScript?

What are operators in JavaScript?

Ans: Operators are symbols used to perform operations on operands.

```
+, -, *, /, %, ++, --
```

Q4. [Interview] Difference between == and ===?

Difference between == and ===?

Ans: `==` checks value, `===` checks value and type (strict equality).

```
'5' == 5 // true
'5' === 5 // false
```

Q5. [Interview] What is an expression in JavaScript?

What is an expression in JavaScript?

Ans: Any valid set of literals, variables, operators that evaluates to a value.

```
3 + 4, name = 'Shekhar'
```

Q6. [Interview] Explain the assignment operator in JS.

Explain the assignment operator in JS.

Ans: It assigns values to variables and supports shorthand syntax.

```
x += 10; // same as x = x + 10
```

Q7. [Interview] What is operator precedence?

What is operator precedence?

Ans: Defines the order in which operators are evaluated.

```
let a = 5 + 3 * 2; // 11
```

Q8. [Interview] What is the use of typeof operator?

What is the use of typeof operator?

Ans: Returns the data type of a variable or expression.

```
typeof 'hello' // string
```

Q9. [Interview] Can const variables be updated?

Can const variables be updated?

Ans: No. `const` values can't be reassigned after initialization.

```
const pi = 3.14; // immutable
```

Q10. [Interview] What is the result of typeof NaN?

What is the result of typeof NaN?

Ans: 'number'. NaN stands for Not-a-Number but its type is still number.

```
typeof NaN // 'number'
```

Q11. [Interview] What is short-circuit evaluation?

What is short-circuit evaluation?

Ans: Using logical operators to return a value based on truthy/falsy logic.

```
false && expr // returns false
true || expr // returns true
```

Q12. [Interview] Explain the comma operator.

Explain the comma operator.

Ans: Evaluates multiple expressions and returns the last.

```
let x = (2, 3); // x = 3
```

Q13. [Interview] What is the output of '5' + 3?

What is the output of '5' + 3?

Ans: String '53'. JS coerces number to string.

```
'5' + 3 // '53'
```

Q14. [Interview] What is unary operator?

What is unary operator?

Ans: Operates on one operand (like +a, -a, !a).

```
let b = -a;
```

Q15. [Interview] Explain increment and decrement operators.

Explain increment and decrement operators.

Ans: ++ increases and -- decreases the value by 1.

```
x++; --y;
```

Q16. [DSA] Find the max of 3 numbers using expression.

Find the max of 3 numbers using expression.

Ans: Use Math.max to find largest among a, b, c.

```
Math.max(a, b, c);
```

Q17. [DSA] Check if a number is even using expression.

Check if a number is even using expression.

Ans: Use modulus operator.

```
const isEven = n => n % 2 === 0;
```

Q18. [DSA] Swap two variables without temp variable.

Swap two variables without temp variable.

Ans: Use destructuring assignment.

```
[a, b] = [b, a];
```

Q19. [DSA] Calculate sum of numbers 1 to n.

Calculate sum of numbers 1 to n.

Ans: Use arithmetic series formula.

```
let sum = n * (n + 1) / 2;
```

Q20. [DSA] Count digits of a number.

Count digits of a number.

Ans: Use loop with division.

```
while (num > 0) { count++; num = Math.floor(num / 10); }
```

Q21. [DSA] Reverse a number using loop.

Reverse a number using loop.

Ans: Use % and division in loop.

```
while (n) { rev = rev * 10 + n % 10; n = Math.floor(n/10); }
```

Q22. [DSA] Check if a number is palindrome.

Check if a number is palindrome.

Ans: Compare original and reversed number.

```
if (num === rev) return true;
```

Q23. [DSA] Find factorial using loop.

Find factorial using loop.

Ans: Use for loop to multiply from 1 to n.

```
for (i=1; i<=n; i++) fact *= i;
```

Q24. [DSA] Find GCD of two numbers.

Find GCD of two numbers.

Ans: Use Euclidean algorithm.

```
while (b !== 0) { [a, b] = [b, a % b]; }
```

Q25. [DSA] Check if number is prime.

Check if number is prime.

Ans: Loop till sqrt(n) and check divisibility.

```
for (i=2; i<=sqrt(n); i++) if (n%i==0) return false;
```

Q26. [DSA] Calculate power of number (x^y).

Calculate power of number (x^y).

Ans: Use loop to multiply x, y times.

```
for (i=0; i<y; i++) res *= x;
```

Q27. [DSA] Check Armstrong number.

Check Armstrong number.

Ans: Sum of cubes of digits == number.

```
while (n) { sum += (d**3); n = Math.floor(n/10); }
```

Q28. [DSA] Convert decimal to binary.

Convert decimal to binary.

Ans: Use bitwise shift or toString(2).

```
n.toString(2);
```

Q29. [DSA] Find sum of even digits in number.

Find sum of even digits in number.

Ans: Use % to check even digits.

```
if (digit % 2 === 0) sum += digit;
```

Q30. [DSA] Count frequency of digit d in n.

Count frequency of digit d in n.

Ans: Loop and check remainder == d.

```
while (n) { if (n % 10 == d) count++; n = Math.floor(n/10); }
```

Q31. [LeetCode] Output of console.log(2 + true + undefined)?

Output of console.log(2 + true + undefined)?

Ans: 2 + true = 3, 3 + undefined = NaN.

```
NaN
```

Q32. [LeetCode] Evaluate: console.log(3 * '2')

Evaluate: console.log(3 * '2')

*Ans: '2' is coerced to number, so 3*2 = 6.*

```
6
```

Q33. [LeetCode] Predict output: '10' - 5 + 2

Predict output: '10' - 5 + 2

Ans: '10' - 5 = 5 (coerced), 5 + 2 = 7

```
7
```

Q34. [LeetCode] Find type: typeof typeof 1

Find type: typeof typeof 1

Ans: typeof 1 is 'number'; typeof 'number' is 'string'

```
'string'
```

Q35. [LeetCode] Is `null == undefined` true?

Is `null == undefined` true?

Ans: Yes, loosely equal.

```
true
```

Q36. [LeetCode] typeof NaN === 'number'?

typeof NaN === 'number'?

Ans: Yes, NaN is of type number.

```
true
```

Q37. [LeetCode] Predict: console.log([] + [])

Predict: console.log([] + [])

Ans: Empty arrays become empty string: ''

```
''
```

Q38. [LeetCode] Predict: true + true + false

Predict: true + true + false

Ans: true -> 1, false -> 0; 1+1+0 = 2

```
2
```

Q39. [LeetCode] Output: '3' * '2' + true

Output: '3' * '2' + true

Ans: '3' and '2' are numbers, true is 1; result 6 + 1 = 7

```
7
```

Q40. [LeetCode] Evaluate: !'false' == false

Evaluate: !'false' == false

Ans: !'false' -> false, false == false -> true

```
true
```

Q41. [Conceptual] Why typeof null is 'object'?

Why typeof null is 'object'?

Ans: Due to a bug in original JS implementation.

```
typeof null // object
```

Q42. [Conceptual] Difference between primitive and reference types?

Difference between primitive and reference types?

Ans: Primitive: number, string, boolean; Reference: objects, arrays.

```
let a = 10; let b = [1,2];
```

Q43. [Conceptual] What is coercion in JS?

What is coercion in JS?

Ans: Automatic or implicit type conversion.

```
5 + '5' // '55'
```

Q44. [Conceptual] What does ++a and a++ do?

What does ++a and a++ do?

Ans: ++a increments first, a++ increments after.

```
let a=5; ++a //6, a++ //5
```

Q45. [Conceptual] Is 0 == false?

Is 0 == false?

Ans: Yes, loosely equal.

```
true
```

Q46. [Conceptual] Does NaN == NaN?

Does NaN == NaN?

Ans: No, NaN is never equal to itself.

```
NaN == NaN // false
```

Q47. [Conceptual] Can you reassign const object's properties?

Can you reassign const object's properties?

Ans: Yes, but not the reference itself.

```
const obj = {}; obj.name = 'Shekhar';
```

Q48. [Conceptual] What is the output of typeof undefined?

What is the output of typeof undefined?

Ans: 'undefined'

```
typeof undefined // 'undefined'
```

Q49. [Conceptual] What is an immediately invoked function expression?

What is an immediately invoked function expression?

Ans: Function invoked right after it is defined.

```
(function(){})()
```

Q50. [Conceptual] Is [] == ![] true?

Is [] == ![] true?

Ans: Yes. [] is truthy, ![] is false -> [] == false -> true

```
true
```