

Introduction to Javascript Operations

Assignment Questions

Q1. Explain the role of operators in JavaScript. Why are they essential in programming?

Answer: Operators in JavaScript are symbols used to perform operations on values and variables. They are essential because they allow you to manipulate data, compare values, control program flow, and perform calculations. Operators make code more efficient, readable, and powerful by enabling tasks like arithmetic, comparisons, logical checks, and assignments. Without operators, you'd be unable to perform basic tasks like adding numbers, making decisions, or altering data, making them fundamental to all programming.

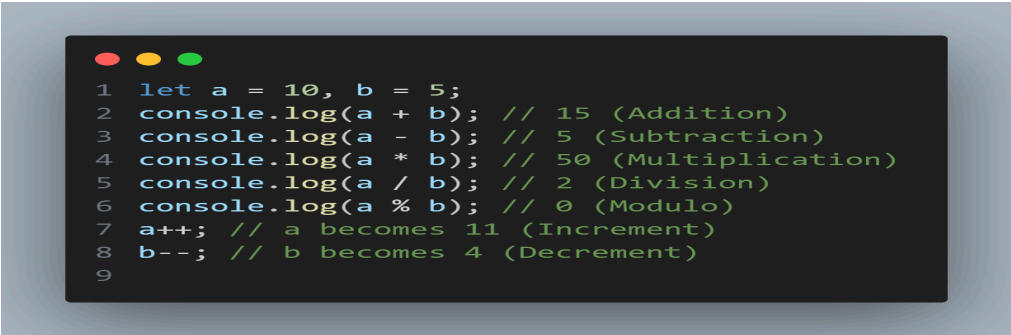
Q2. Describe the categorization of operators in JavaScript based on their functionality. Provide examples for each category.

Answer: In JavaScript, operators can be categorized based on their functionality into several key types. Here's a breakdown of the major categories with examples:

1. Arithmetic Operators

These operators are used to perform mathematical calculations.

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Division)
- % (Remainder/Modulo)
- ++ (Increment)
- -- (Decrement)



```
1 let a = 10, b = 5;
2 console.log(a + b); // 15 (Addition)
3 console.log(a - b); // 5 (Subtraction)
4 console.log(a * b); // 50 (Multiplication)
5 console.log(a / b); // 2 (Division)
6 console.log(a % b); // 0 (Modulo)
7 a++; // a becomes 11 (Increment)
8 b--; // b becomes 4 (Decrement)
9
```

2. Assignment Operators

These operators are used to assign values to variables, sometimes performing an operation simultaneously.

- = (Simple assignment)
- += (Add and assign)
- -= (Subtract and assign)
- *= (Multiply and assign)
- /= (Divide and assign)
- %= (Modulo and assign)

Example:

let x = 10;

x += 5; // x becomes 15

x *= 2; // x becomes 3

3. Comparison Operators

These are used to compare values, typically in conditional statements (returning **true** or **false**).

- == (Equal to)
- === (Strict equal to)
- != (Not equal to)
- !== (Strict not equal to)
- > (Greater than)
- < (Less than)
- >= (Greater than or equal to)
- <= (Less than or equal to)

Example:

let a = 10, b = 5;

console.log(a == b); // false (Equality)

console.log(a !== b); // true (Strict inequality)

```
console.log(a > b); // true (Greater than)
```

4. Logical Operators

These are used to combine multiple conditions, often used in control flow statements like **if** or **while**.

- **&&** (Logical AND)
- **||** (Logical OR)
- **!** (Logical NOT)

Example:

```
let x = true, y = false;
```

```
console.log(x && y); // false (AND)
```

```
console.log(x || y); // true (OR)
```

```
console.log(!x); // false (NOT)
```

Operators are essential in JavaScript because they allow us to manipulate values, perform calculations, and control logic in our programs. These categories form the core functionality needed for most programming tasks.

Q3. Differentiate between unary, binary, and ternary operators in JavaScript. Give examples of each.

Answer: In JavaScript, operators are classified based on the number of operands they work with. Here's a differentiation between unary, binary, and ternary operators, along with examples for each:

1. Unary Operators

- **Definition:** Unary operators work on a single operand.
- **Function:** They perform operations like incrementing, decrementing, negating a value, or determining the type of a variable.

Examples:

- **Increment (++):** Increases a variable's value by 1.
- **Decrement (--):** Decreases a variable's value by 1.
- **Unary Plus (+):** Converts a variable to a number (if it's not already).
- **Unary Negation (-):** Converts a value to its negative.
- **typeof:** Returns the type of a variable.

- **!**: Converts a value to a boolean and negates it.

Example Code:

```
let x = 5;  
x++; // Unary Increment: x becomes 6  
x--; // Unary Decrement: x becomes 5 again  
  
let y = "42";  
console.log(+y); // Unary Plus: converts "42" to number 42  
  
let z = -10;  
console.log(-z); // Unary Negation: changes z to 10  
  
console.log(typeof x); // "number" (checks type)  
console.log(!true); // false (negates boolean)
```

2. Binary Operators

- Definition: Binary operators work on two operands.
- Function: They perform operations between two values, such as mathematical operations, comparisons, and logical operations.

Examples:

- Arithmetic Operators (**+**, **-**, *****, **/**, etc.): Perform mathematical operations between two numbers.
- Comparison Operators (**==**, **===**, **>**, **<**, etc.): Compare two values and return a boolean.
- Logical Operators (**&&**, **|** |): Combine two boolean expressions.

Example Code:

```
let a = 10, b = 5;  
  
// Arithmetic Operations  
console.log(a + b); // 15 (Addition)  
console.log(a * b); // 50 (Multiplication)  
  
// Comparison Operations  
console.log(a > b); // true (Greater than)  
console.log(a == b); // false (Equality)
```

// Logical Operations

```
let condition1 = true, condition2 = false;
```

```
console.log(condition1 && condition2); // false (AND)
```

```
console.log(condition1 || condition2); // true (OR)
```

3. Ternary Operator

- **Definition:** The ternary operator is a **conditional** operator that works on three parts: a condition, a value if true, and a value if false.
- **Function:** It evaluates a condition and returns one of two values based on whether the condition is **true** or **false**.

Syntax: `condition ? value_if_true : value_if_false;`

Example:

```
let age = 20;
```

```
let result = (age >= 18) ? "Adult" : "Minor";
```

```
console.log(result); // "Adult"
```

Q4. Discuss the precedence and associativity of operators in JavaScript. Why is understanding these concepts important?

Answer: Operator Precedence in JavaScript determines the order in which operators are applied in an expression. Operators with higher precedence are evaluated before those with lower precedence. For example, multiplication (*) has higher precedence than addition (+), so in `2 + 3 * 4`, multiplication happens first.

Operator Associativity defines the direction in which operators of the same precedence are evaluated. Most operators are evaluated left to right, but some, like assignment (=) and the ternary (? :), are evaluated right to left.

Why It's Important:

- **Avoid Errors:** Misunderstanding precedence can lead to incorrect results in calculations.
- **Write Clear Code:** Knowing how operators are evaluated helps you write more predictable and efficient code.
- **Control Expression Evaluation:** It allows you to structure expressions effectively and ensures the correct order of operations.

Q5. Write a JavaScript program that calculates the simple interest using the formula Simple interest = (principal * rate * time) / 100.

Answer:

```
1 // Function to calculate Simple Interest
2 function calculateSimpleInterest(principal, rate, time) {
3     let interest = (principal * rate * time) / 100;
4     return interest;
5 }
6
7 // Example inputs
8 let principal = 1000; // Principal amount
9 let rate = 5;         // Annual interest rate (in percentage)
10 let time = 3;         // Time period in years
11
12 // Calculate Simple Interest
13 let interest = calculateSimpleInterest(principal, rate, time);
14
15 // Output the result
16 console.log("The simple interest is: " + interest);
17
```

Q6. Write a Javascript program to calculate the Body Mass Index (BMI) using the formula BMI = weight (kg)/ height * height.

Answer:

```
1 // Function to calculate BMI
2 function calculateBMI(weight, height) {
3     let bmi = weight / (height * height); // Formula for BMI
4     return bmi;
5 }
6
7 // Example inputs
8 let weight = 70; // Weight in kilograms
9 let height = 1.75; // Height in meters
10
11 // Calculate BMI
12 let bmi = calculateBMI(weight, height);
13
14 // Output the result
15 console.log("Your Body Mass Index (BMI) is: " + bmi.toFixed(2)); // Display with 2 decimal points
16
```

Q7. Write a program in JavaScript to calculate the area of a circle given its radius value of 10. Use appropriate arithmetic operators.

Answer:

```
1 // Function to calculate the area of a circle
2 function calculateAreaOfCircle(radius) {
3     const pi = Math.PI; //  $\pi$  value in JavaScript
4     let area = pi * radius * radius; // Formula:  $\pi * r^2$ 
5     return area;
6 }
7
8 // Given radius
9 let radius = 10;
10
11 // Calculate area
12 let area = calculateAreaOfCircle(radius);
13
14 // Output the result
15 console.log("The area of the circle is: " + area.toFixed(2)); // Display with 2 decimal points
16
```

Full Stack Web Development