Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Class 03: Operator

Example: Operator and Expression

Operators: Think of operators as tools that help you perform different tasks or actions. Just like how you use a calculator to add, subtract, multiply, or divide numbers, operators in programming allow you to do similar operations on data.

- **Arithmetic Operators**: These are like your basic math operations (+ for addition, - for subtraction, * for multiplication, / for division). For example, if you have 5 apples and you buy 3 more, you can use the + operator to find out how many apples you have in total.
- **Comparison Operators**: These are used to compare values and determine if one value is greater than, less than, equal to, or not equal to another value. For instance, if you want to compare the prices of two items to see which one is cheaper, you can use comparison operators like < (less than) or > (greater than).
- **Logical Operators**: These are used to combine conditions and make decisions based on multiple conditions. Imagine you have two criteria for choosing a movie to watch: it should be both funny AND family-friendly. Here, the AND operator (&&) is used to combine the conditions.

Expressions: An expression is like a sentence made up of words and punctuation marks, but in programming, it's made up of variables, values, and operators that perform some computation or produce a result.

- **Numeric Expressions**: These involve numbers and arithmetic operators. For example, 5 + 3 is a numeric expression that calculates the sum of 5 and 3.
- **String Expressions**: These involve text (strings) and string operators. If you combine "Hello" with "World" using the + operator, you get the string "Hello World".
- **Boolean Expressions**: These involve logical operators and compare two conditions to produce a true or false result. For example, if you compare whether 10 is greater than 5 using the > operator, you get a true result.

Note: Operator

An operator in programming is a symbol or keyword that performs a specific operation on one or more operands (values or variables) to produce a result. Operators are fundamental to programming as they enable you to perform mathematical calculations, compare values, combine conditions, assign values, and more.

There are different types of operators in programming, including:

1. **Arithmetic Operators**: Used for basic mathematical operations such as addition (+), subtraction (-), multiplication (*), division (/), and modulus (remainder, %).

```
// Arithmetic Operators
var a = 10;
var b = 5;
var sum = a + b; // Addition
var difference = a - b; // Subtraction
var product = a * b; // Multiplication
var quotient = a / b; // Division
var remainder = a % b; // Modulus (remainder)
```

2. **Comparison Operators**: Used to compare values and return a boolean result. Examples include equal to (== or ===), not equal to (!= or !==), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=).

Strict Equality (===):

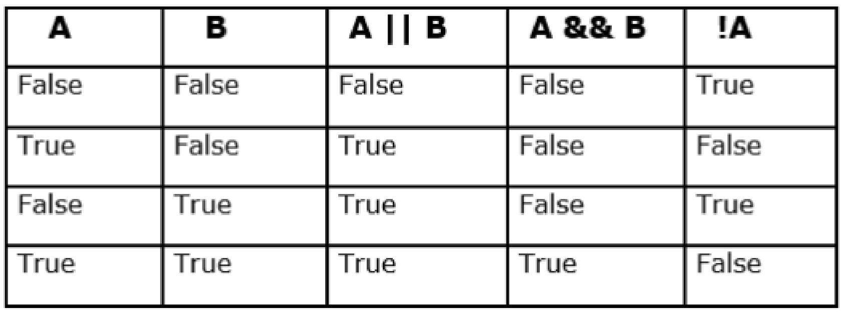
- Checks both value and type.

Strict Inequality (!==):

- Checks both value and type.

```
// Comparison Operators
var x = 10;
var y = 5;
var isEqual = x === y; // Equal to
var isNotEqual = x !== y; // Not equal to
var isGreater = x > y; // Greater than
var isLess = x < y; // Less than
var isGreaterOrEqual = x >= y; // Greater than or equal to
var isLessOrEqual = x <= y; // Less than or equal to
```

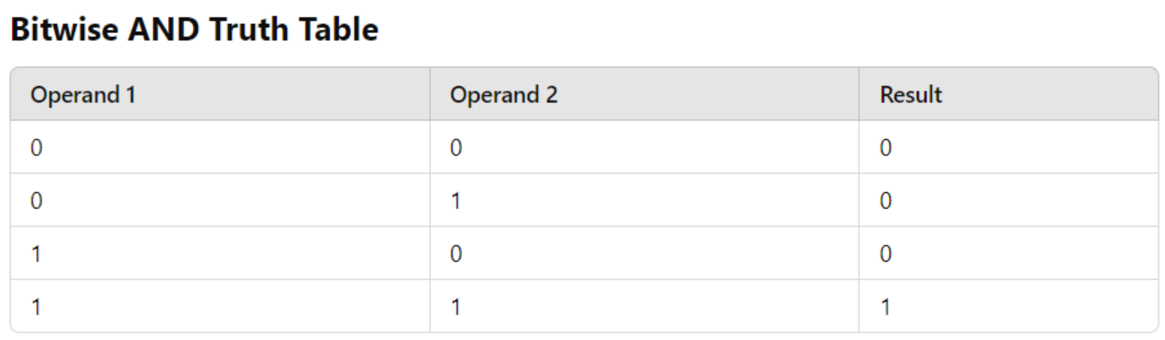
3. **Logical Operators**: Used to combine multiple conditions and return a boolean result. Common logical operators include AND (&&), OR (||), and NOT (!).



A	B	A || B	A && B	!A
False	False	False	False	True
True	False	True	False	False
False	True	True	False	True
True	True	True	True	False

```
// Logical Operators
var isSunny = true;
var isWarm = false;
var isGoodWeather = isSunny && isWarm; // AND operator
var isAnyWeather = isSunny || isWarm; // OR operator
var isNotSunny = !isSunny; // NOT operator (negation)
```

Bitwise Operators:



Bitwise AND Truth Table

Operand 1	Operand 2	Result
0	0	0
0	1	0
1	0	0
1	1	1

4. **Assignment Operators**: Used to assign values to variables. Examples include the assignment operator (=), addition assignment (+=), subtraction assignment (-=), multiplication assignment (*=), division assignment (/=), and modulus assignment (%=).

```
// Assignment Operators
var num = 10;
num += 5; // Addition assignment, equivalent to num = num + 5
num -= 3; // Subtraction assignment, equivalent to num = num - 3
num *= 2; // Multiplication assignment, equivalent to num = num * 2
num /= 4; // Division assignment, equivalent to num = num / 4
num %= 3;
```

5. **Unary Operators**: Operate on a single operand. Examples include the increment (++) and decrement (--) operators.

```
// Unary Operators
var count = 0;
count++; // Increment operator, equivalent to count = count + 1
count--; // Decrement operator, equivalent to count = count - 1
```

6. **Type Operator**: The typeof operator is used to determine the type of a variable or value. It returns a string indicating the type of the operand.

```
console.log(typeof 123);        // "number"
console.log(typeof "hello");    // "string"
console.log(typeof true);       // "boolean"
console.log(typeof undefined);  // "undefined"
console.log(typeof null);        // "object"
```

7. Ternary Operator:

The ternary operator, also known as the conditional operator, is a concise way to perform conditional expressions in JavaScript. It is the only operator that takes three operands, hence the name "ternary."

```
condition ? expressionIfTrue : expressionIfFalse
```

condition: An expression that evaluates to true or false.

expressionIfTrue: The expression that is executed if the condition is true.

expressionIfFalse: The expression that is executed if the condition is false.

```
var age = 18;
var message = (age >= 18) ? 'You are an adult.' : 'You are a minor.';
console.log(message); // Output: "You are an adult."
```

Conditional Statement:

If-Else Statements in JavaScript

The if-else statement in JavaScript is a fundamental control flow structure that allows you to execute certain pieces of code based on specified conditions. It helps in decision making by executing code blocks conditionally.

```
if (condition) {
   // Code to execute if condition is true
} else {
   // Code to execute if condition is false
}
```

If-Else If-Else Example

For more complex decision-making, you can use else if to check multiple conditions.

```
var score = 85;

if (score >= 90) {
   console.log("Grade: A");
} else if (score >= 80) {
   console.log("Grade: B");
} else if (score >= 70) {
   console.log("Grade: C");
} else if (score >= 60) {
   console.log("Grade: D");
} else {
   console.log("Grade: F");
}
```

Task:

1. **Arithmetic Operators Task**: Calculate the sum, difference, product, and quotient of two numbers (e.g., 10 and 5) using arithmetic operators (+, -, *, /).
2. **Comparison Operators Task**: Compare two values (e.g., 10 and 5) using comparison operators (==, ===, !=, !==, >, <, >=, <=) and print whether they are equal or not.
3. **Logical Operators Task**: Check if a person's age (e.g., 25) is between 18 and 65 (inclusive) using logical operators (&&, ||, !) and print "Allowed" or "Not Allowed" accordingly.
4. **Assignment Operators Task**: Increment a number (e.g., 10) by 5 using assignment operators (+=) and print the result.
5. **Unary Operators Task**: Toggle a boolean value (e.g., true) using unary operators (++ and --) and print the toggled value.
6. **Ternary Operator Task**: Check if a number (e.g., 7) is even or odd using the ternary operator and print "Even" or "Odd" accordingly.

Interview Questions:

What are the different types of operators available in JavaScript?

- **Answer:** JavaScript operators include arithmetic operators (+, -, *, /, %), assignment operators (=, +=, -=), comparison operators (==, ===, !=, !==, >, <, >=, <=), logical operators (&&, ||, !), bitwise operators (&, |, ^, ~, <<, >>, >>>), and others like the ternary operator (? :) and typeof operator.

Explain the difference between == and ===.

- **Answer:** == is the abstract equality operator that performs type coercion if the operands are of different types before comparing them. === is the strict equality operator that does not perform type coercion and returns false if the operands are of different types.

What is the result of the following expression: 5 + '5'?

- **Answer:** The result is '55' because the + operator with a number and a string performs string concatenation.

What are logical operators and how do they work in JavaScript?

- **Answer:** Logical operators are used to combine or invert boolean values. They include:
  - && (AND): Returns true if both operands are true.
  - || (OR): Returns true if at least one operand is true.
  - ! (NOT): Inverts the boolean value of its operand.

Describe the precedence and associativity of operators.

- **Answer:** Operator precedence determines the order in which operators are evaluated. Associativity determines the order in which operators of the same precedence are processed. For example, multiplication (*) has higher precedence than addition (+), and both are left-associative, meaning they are evaluated from left to right.

Explain short-circuit evaluation in logical operators with examples.

- **Answer:** Short-circuit evaluation means that in a logical expression, evaluation stops as soon as the outcome is determined.
  - For && (AND): If the first operand is false, the second operand is not evaluated.
  - For || (OR): If the first operand is true, the second operand is not evaluated.

What will be the output of the following code snippet and why?

```
let result = (true && false) || (false && true) || !(false && false);
console.log(result);
```

Answer: The output will be true because:

- (true && false) evaluates to false
- (false && true) evaluates to false
- !(false && false) evaluates to !(false) which is true
- Combining these with ||: false || false || true evaluates to true.

Explain the difference between the postfix (i++) and prefix (++i) increment operators.

- **Answer:** The postfix increment operator (i++) increases the value of i but returns the original value before the increment. The prefix increment operator (++i) increases the value of i and returns the new value after the increment.

How do you avoid using multiple if-else if-else statements when there are many conditions?

- **Answer:** You can use a switch statement, an object lookup, or a map to replace multiple if-else if-else statements for better readability and maintainability.

How does the typeof operator work in JavaScript?

- **Answer**: The typeof operator returns a string indicating the type of the operand. For example, typeof 42 returns "number".

What is the purpose of the ! operator in JavaScript?

- **Answer**: The ! operator is the logical NOT operator, which inverts the boolean value of its operand. For example, !true returns false.