Logical Operators in JavaScript

Logical operators in JavaScript are used to perform logical operations on boolean values. These operators allow you to combine or modify the results of boolean expressions. The three main logical operators are **AND (`&&`)**, **OR (`||`)**, and **NOT (`!`)**.

1. Logical AND (`&&`)

The logical AND operator returns `true` if all conditions are `true`; otherwise, it returns `false`.

Example 1: Basic Logical AND with Comparison and `if` condition

```javascript
let value1 = 5;
let value2 = 10;

if (value1 > 0 && value2 > 0) {
  console.log("Both values are greater than 0");
} else {
  console.log("At least one value is not greater than 0");
}
```

- In this example, the logical `AND` operator`(&&)`is used to check if both `value1` and `value2` are greater than 0.
- The if condition evaluates to true only if both conditions are satisfied. If `true`, it logs "Both values are greater than 0";
- otherwise, it logs "At least one value is not greater than 0."

Example 2: Logical AND with Multiple Conditions and `if` statement

```javascript
let isLogged = true;
let isAdmin = false;

if (isLogged && isAdmin) {
  console.log("Access granted: User is logged in and is an admin");
} else {
  console.log("Access denied: Either user is not logged in or is not an admin");
}
```

- In this example, the logical `AND` operator is used to check if both `isLogged` and `isAdmin` are true.
- The if condition logs "Access granted: User is logged in and is an admin" only if both conditions are true;
- otherwise, it logs "Access denied: Either the user is not logged in or is not an admin."

Example 3: Logical AND with Ternary Operator and `if` condition

```
let age = 25;
let hasLicense = true;

let result = age >= 18 && hasLicense ? "Allowed to drive" : "Not allowed to
drive";

console.log(result);
```

- In this example, the logical `AND` operator is used in conjunction with the ternary operator `(?)` to determine the value of the `result` variable.
- it checks if both age is greater than or equal to 18 and hasLicense is true. If true, it assigns "Allowed to drive" to result;
- otherwise, it assigns "Not allowed to drive." The final `console.log(result)` prints the result.

2. Logical OR (`||`)

The logical OR operator returns `true` if at least one condition is `true`.

Example 1: Basic Logical OR with Comparison and `if` condition

```
let temperature = 25;

if (temperature < 0 || temperature > 100) {
  console.log("Temperature is either below freezing or above boiling");
} else {
  console.log("Temperature is within the acceptable range");
}
```

- In this example, the logical OR operator (`||`) is used to check if the temperature is either below freezing (less than 0) or above boiling (greater than 100).
- The if condition evaluates to true if at least one of these conditions is satisfied, logging "`temperature` is either below freezing or above boiling."
- If none of the conditions are true, it logs "`temperature` is within the acceptable range."

Example 2: Logical OR with Multiple Conditions and `if` statement

```
let isGuest = false;
let isAdmin = true;

if (isGuest || isAdmin) {
  console.log("Welcome! Access granted to guests or admins");
} else {
  console.log("Access denied: You are neither a guest nor an admin");
}
```

- Here, the logical OR operator is used to check if the user is either a guest (`isGuest` is true) or an admin (`isAdmin` is true).
- The if condition logs "Welcome! Access granted to guests or admins" if at least one of the conditions is true.
- If both conditions are false, it logs "Access denied: You are neither a guest nor an admin."

Example 3: Logical OR with Ternary Operator and `if` condition

```
let income = 50000;
let creditScore = 700;

let result = income > 60000 || creditScore > 750 ? "Loan approved" : "Loan
denied";
console.log(result);
```

- In this example, the logical `OR` operator is used with the ternary operator to determine the value of the result variable.
- It checks if either the `income` is greater than $60,000 or the `credit Score`is greater than 750.
- If true, it assigns "Loan approved" to result; otherwise, it assigns "Loan denied." The final console.log(result) prints the result.

3. Logical NOT (`!`)

The logical NOT operator returns `true` if the value is `false`, and `false` if the value is `true`.

Certainly! Let's break down each example:

Example 1: Basic Logical NOT with Comparison and `if` condition

```
let isDisabled = false;

if (!isDisabled) {
  console.log("The button is enabled");
} else {
  console.log("The button is disabled");
}
```

Explanation:

- `let isDisabled = false;`: Declares a variable `isDisabled` and initializes it with the value `false`.
- `if (!isDisabled) {`: Checks if the logical NOT (`!`) of `isDisabled` is `true`.
- If `isDisabled` is `false`, the condition evaluates to `true`, and the message "The button is enabled" is logged to the console.
- If `isDisabled` is `true`, the condition evaluates to `false`, and the message "The button is disabled" is logged to the console.

Example 2: Logical NOT with Multiple Conditions and `if` statement

```
let isLoggedIn = false;
let hasAuthToken = true;

if (!isLoggedIn && !hasAuthToken) {
  console.log("User is not logged in and does not have an auth token");
} else {
  console.log("User is either logged in or has an auth token");
}
```

Explanation:

- `let isLoggedIn = false;`: Declares a variable `isLoggedIn` and initializes it with the value `false`.
- `let hasAuthToken = true;`: Declares a variable `hasAuthToken` and initializes it with the value `true`.
- `if (!isLoggedIn && !hasAuthToken) {`: Checks if the logical NOT (`!`) of both `isLoggedIn` and `hasAuthToken` are `true`.
- If both conditions are `true`, the message "User is not logged in and does not have an auth token" is logged to the console.
- If any of the conditions is `false`, the message "User is either logged in or has an auth token" is logged to the console.

Example 3: Logical NOT with Ternary Operator and `if` condition

```
let isBlocked = true;

let result = !isBlocked ? "User is unblocked" : "User is blocked";
console.log(result);
```

Explanation:

- `let isBlocked = true;`: Declares a variable `isBlocked` and initializes it with the value `true`.
- `let result = !isBlocked ? "User is unblocked" : "User is blocked";`: Uses the ternary operator (`? :`) to assign a value to the variable `result` based on the logical NOT (`!`) of `isBlocked`.
- If `isBlocked` is `true`, the logical NOT is `false`, so "User is blocked" is assigned to `result`.
- If `isBlocked` is `false`, the logical NOT is `true`, so "User is unblocked" is assigned to `result`.
- The value of `result` is then logged to the console.

Why logical Operators Matter?

Logical operators are fundamental in programming because they facilitate decision-making processes by allowing the evaluation of conditions and relationships between various data values. A solid grasp of these operators is vital for crafting code that is not only functional but also capable of responding dynamically to different scenarios.

Logical operators empower programmers to establish logical expressions that dictate the flow of a program. These expressions often involve comparisons between different values, and the outcome of these comparisons determines the subsequent actions the program will take.