Logical Operators in JavaScript

Instructor : Pramod Kumar Jena

What are Logical Operators?

Logical operators are used to combine two or more conditions and return a boolean result (`true` or `false`). They help connect multiple statements and check multiple conditions simultaneously.

In the last class, we learned about **conditional statements** that allow us to perform different actions based on a condition being true or false. For instance, in a traffic light system:

- If the light is green, then move.
- If the light is red, then stop.

But in reality, some results depend on **multiple conditions**. For example:

- Suppose you need to submit two documents (PAN card and License ID) to get admission into USBM. You will only get admission if **both** the PAN Card and License ID are available. In this case, both conditions must be true.

1. AND Operator (&&)

The **AND (&&) operator** connects two conditions and returns `true` only when both conditions are true. If either condition is false, it returns `false`.

Example:

PAN Card and License ID Scenario:

- I will only get admission if I have both the PAN Card and License ID.

PAN Card	License ID	Admission
True	True	True
True	False	False
False	True	False
False	False	False

Code Example 1: AND Operator with Booleans

```
var a = true;
var b = true;
var c = a && b;
console.log(c); // true

a = true;
b = false;
console.log(a && b); // false

a = false;
b = true;
console.log(a && b); // false

a = false;
b = false;
console.log(a && b); // false
```

Code Example 2: AND Operator with Numbers

```
var a = 5 > 3; // true
var b = 6 > 3; // true
var c = a && b;
console.log(c); // true
```

Code Example 3: Combining Multiple Conditions

```
if(5 > 3 && 6 > 3) {
  console.log("Both are true");
}
```

Code Example 4: Student Task - Rahul's Exam Result

Rahul will pass English if he scores at least the passing marks (70).

```
var subject = "english";
var passing_marks = 70;
var rahul_marks = 75;
var rahul_subject = "english";

if((rahul_subject == subject) && (rahul_marks >= passing_marks)) {
  console.log("Rahul Passed");
} else {
  console.log("Rahul did not pass");
}
```

Code Example 5: Marriage Eligibility Check

A male can marry if he is 21 or older. A female can marry if she is 18 or older.

```
var gender = "male";
var age = 21;

if((gender == "male" && age >= 21)) {
  console.log("Male: Can Marry");
} else if((gender == "female" && age >= 18)) {
  console.log("Female: Can Marry");
} else {
  console.log(gender, "Cannot Marry");
}
```

2. OR Operator (`||`)

The **OR (`||`) operator** connects two conditions and returns `true` if at least one of the conditions is true. If both conditions are false, it returns `false`.

Example:

In DriveZy, a bike rental service, you can submit any of the following identity documents to rent a bike: Aadhar, PAN Card, License, or Voter ID.

Aadhar Card	PAN Card	License	Voter ID	Result

True	False	False	False	True
False	False	True	False	True
False	False	False	False	False

Code Example 6: OR Operator with Booleans

```
var a = true;
var b = false;
console.log(a || b); // true

a = false;
b = false;
console.log(a || b); // false
```

Code Example 7: Mom's Dinner Plan

Mom will cook dinner if Sunny buys **either** potatoes or paneer.

```
var potato_available = true;
var paneer_available = false;

if(potato_available || paneer_available) {
  console.log("Dinner is possible");
} else {
  console.log("Dinner is not possible");
}
```

Code Example 8: Marriage Eligibility with OR Operator

Check if someone is eligible to marry based on gender and age.

```
var gender = "female";
var age = 18;

if((gender == "male" && age >= 21) || (gender == "female" && age >=
18)) {
  console.log(gender, "Can get married");
} else {
  console.log(gender, "Cannot get married");
}
```

3. NOT Operator (`!`)

The **NOT (`!`) operator** negates the value of a boolean expression, returning `false` for `true` and `true` for `false`.

Example:

```
var a = true;
console.log(!a); // false
```

Switch Case in JavaScript

Whenever we have multiple options and want to select one specific option, we use the **switch case** statement.

For example, in an ATM machine, we have different options like **Deposit, Withdraw, Change PIN**, etc. Each option corresponds to a different block of code.

Structure of Switch Case:

```
switch(expression) {
  case value1:
    // code block for value1
    break;
  case value2:
    // code block for value2
    break;
  default:
    // code block for default case
}
```

Example: ATM Machine Options

Let's simulate an ATM machine with multiple options.

```
var option = 3;

switch(option) {
  case 1:
    console.log("Deposit");
    break;
  case 2:
    console.log("Withdraw");
    break;
  case 3:
    console.log("Change PIN");
    break;
  default:
    console.log("Invalid option");
}
```

Example: Day Schedule

We can use a switch case to handle different day schedules.

```
var day = 2;

switch(day) {
  case 1:
    console.log("Day 1: Scrum + Coding");
    break;
  case 2:
    console.log("Day 2: Scrum + Coding + Skillathon");
    break;
  default:
    console.log("Holiday");
    break;
}
```