JavaScript Operators 13th July 2020

What are Operators in JavaScript?

JavaScript includes operators as in other languages. An operator performs some operation on single or multiple operands (operands are data value) and produces a result. For example 1 + 2, where + sign is an operator and 1 is left operand and 2 is right operand. + operator adds two numeric values and produces a result which is 3 in this case.

Basic syntax:

<Left operand> operator <right operand>

<Left operand> operator

JavaScript operators are symbols that are used to perform operations on operands. A good example is when you are adding an item on a shopping cart you performs an operation, either you add to the already existing items or you subtract an item from the already existing items.

Precedence of operators –

The *precedence* of operators determines the order they are applied when evaluating an expression but you can override operator precedence by using parentheses.

JavaScript includes following categories of operators.

1. Arithmetic Operators.
2. Comparison Operators.
3. Logical Operators.
4. Assignment Operators.

Note that these are just the basic operators, JavaScript includes a number of special operators such as Ternary Operator, etc which we will discus in details.

1). **Arithmetic Operators**

Arithmetic operators are used to perform mathematical operations between numeric operands.
Remember JavaScript has the Math object allows you to perform mathematical tasks. We will cover the math objects later.

The following operators are known as JavaScript arithmetic operators.

Operator	
+	Adds two numeric operands.
-	Subtract right operand from left operand
*	Multiply two numeric operands.
/	Divide left operand by right operand.
%	Modulus operator. Returns remainder of two operands.
++	Increment operator. Increase operand value by one.
--	Decrement operator. Decrease value by one.

The following example demonstrates how arithmetic operators perform different tasks on operands.

```
let x = 13, y = 20, z = 8;

x + y; //returns 33
y - x; //returns 7
x * y; //returns 260
y / x; //returns 1.5
x % 2; //returns 1
x++; //returns 14
x--; //returns 12
```

Exponential operator (It is part or arithmetic operators):

Exponential operatorstakes the left operand and raises it to power the right operand.

Example:

```
console.log(2**3); //returns 8
```

In exponentiation operator has right-associativity, whereas other arithmetic operators have left-associativity. It is interesting to note that, the order of evaluation is always left-to-right irregardless of associativity. (We will dicuss this in more details another day).

Note That:

+ operator performs concatenation operation when one of the operands is of string type. This is the most common string operator.

Example of concatenation operation:

```
let namea = "Hello ";
let nameb = "World!";
console.log(namea + nameb); //returns Hello World

let age = 23;
console.log(age + namea); //returns 23Hello
```

2). **Comparison Operators**

JavaScript language includes operators that compare two operands and return Boolean value true or false.
The comparison operators are as follows:

Operator	**Description**
==	Compares the equality of two operands without considering typ
===	Compares equality of two operands with type.
!=	Compares inequality of two operands.
>	Checks whether left side value is greater than right side value. If yes then returns true otherwise false.
<	Checks whether left operand is less than right operand. If yes then returns true otherwise false.
>=	Checks whether left operand is greater than or equal to right operand. If yes then returns true otherwise false.
<=	Checks whether left operand is less than or equal to right operand. If yes then returns true otherwise false.

Note That:

A comparison operator compares its operands and returns a logical value based on whether the comparison is true. The operands can be numerical, string, logical, or object values. Strings are compared based on standard lexicographical ordering, using Unicode values.

The following example demonstrates how comparison operators perform different tasks.

```
var a = 5, b = 10, c = "5";
var x = a;

a == c; // returns true

a === c; // returns false

a == x; // returns true

a != b; // returns true

a > b; // returns false

a < b; // returns true

a >= b; // returns false

a <= b; // returns true
a >= c; // returns true
a <= c; // returns true
```

3). **Logical Operators**

Logical operators are used to combine two or more conditions. JavaScript includes following logical operators, && and || operators actually return the value of one of the specified operands, so if these operators are used with non–Boolean values, they may return a non–Boolean value.

The logical operators are as follows:

Operator	Description
&&	&& is known as AND operator. It checks whether two operands are non-zero (0, false, undefined, null or "" are considered as zero), if yes then returns 1 otherwise 0.
||	|| is known as OR operator. It checks whether any one of the two operands is non-zero (0, false, undefined, null or "" is considered as zero).
!	! is known as NOT operator. It reverses the Boolean result of the operand (or condition)

The following example demonstrates how comparison operators perform different tasks.

```
var a = 5, b = 10;

(a != b) && (a < b); // returns true

(a > b) || (a == b); // returns false

(a < b) || (a == b); // returns true

!(a < b); // returns false

!(a > b); // returns true
```

4. Assignment Operators.

JavaScript includes assignment operators to assign values to variables with less key strokes, It assigns a value to its left operand based on the value of its right operand. The simple assignment operator is equal *(=),* which assigns the value of its right operand to its left operand. That is, a = b assigns the value of b to a.

There are also compound assignment operators and they are all discussed in the following table:

Operator	Description
=	Assigns right operand value to left operand.
+=	Sums up left and right operand values and assign the result to the left operand.
-=	Subtract right operand value from left operand value and assign the result to the left operand.
*=	Multiply left and right operand values and assign the result to the left operand.
/=	Divide left operand value by right operand value and assign the result to the left operand.
%=	Get the modulus of left operand divide by right operand and assign resulted modulus to the left operand.
`??=`	Logical nullish assignment (`x ??= y`) operator only assigns if `x` is `nullish` (`null` or `undefined`).

The following example demonstrates how comparison operators perform different tasks.

```
var x = 5, y = 10, z = 15;

x = y; //x would be 10
```

```
x += 1; //x would be 6

x -= 1; //x would be 4

x *= 5; //x would be 25

x /= 5; //x would be 1

x %= 2; //x would be 1
```

More operators:

- Ternary Operator

The operator :? , which is also referred to as conditional operator assigns a value to a variable based on some condition.
Ternary operator starts with conditional expression followed by ? operator. Second part (after ? and before : operator) will be executed if condition turns out to be true. If condition becomes false then third part (after :) will be executed.

Basic Syntax:

```
var a = 10, b = 5;

var c = a > b? a : b; // value of c would be 10
var d = a > b? b : a; // value of d would be 5
```

Must Know:
A binary operator requires two operands, one before the operator and one after the operator, while a unary operator requires a single operand, either before or after the operator. Delete is a good example of unary operator.

Example implementing delete.

```
x = 42; // implicitly creates window.x
var  y = 43;
```

```
delete x; //returns true (can delete if created implicitly)
delete y; // returns false ( can not delete if declared with var
```

Comma Operator. **(,)**

The comma operator evaluates both of its operands and returns the value of the last operand. This operator is primarily used inside a for loop, to allow multiple variables to be updated each time through the loop. It is regarded bad style to use it elsewhere, when it is not necessary. Often two separate statements can and should be used instead.

we will recap and implement all operators in a future tutorial.

Don't Just Use JavaScript, **Know JavaScript**.

Best Wishes.

Lux Tech Academy.

Email: **luxtinc@gmail.com**.