JavaScript - Operators

What is an Operator?

In JavaScript, an **operator** is a symbol that performs an operation on one or more operands, such as variables or values, and returns a result. Let us take a simple expression **4 + 5** is equal to 9. Here 4 and 5 are called **operands**, and '+' is called the **operator**.

JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Bitwise Operators
- Assignment Operators
- Miscellaneous Operators

Lets have a look on all operators one by one.

JavaScript Arithmetic Operators

The JavaScript arithmetic operators are used to perform mathematical calculations such as addition, multiplication, subtraction, division, etc. on numbers. JavaScript supports the following arithmetic operators −

Assume variable **x** holds 10 and variable **y** holds 20, then −

Operator	Description	Example
+ (Addition)	Adds two operands.	x + y will give 30.

- (Subtraction)	Subtracts the second operand from the first.	x - y will give -10.
* (Multiplication)	Multiplies both operands.	x * y will give 200.
/ (Division)	Divides the numerator by the denominator.	y / x will give 2.
% (Modulus)	Outputs the remainder of an integer division.	y % x will give 0
++ (Increment)	Increases an integer value by one.	x++ will give 11.
-- (Decrement)	Decreases an integer value by one.	x-- will give 9.

Addition operator (+) works for Numeric as well as Strings. e.g. "a" + 10 will give "a10".

Learn **Java** in-depth with real-world projects through our **Java certification course**. Enroll and become a certified expert to boost your career.

JavaScript Comparison Operators

The JavaScript comparison operators compare two values and returns a boolean result (true or false). JavaScript supports the following comparison operators –

Assume variable **x** holds 10 and variable **y** holds 20, then –

Operator	Description	Example
== (Equal)	Checks if the value of two operands is equal or not. If yes, then the condition becomes true.	(x == y) is not true.
!= (Not Equal)	Checks if the value of two operands is equal or not. If the values are not equal, then the condition becomes true.	(x != y) is true.
=== (Strict equality)	It checks whether the value and data type of the variable is equal or not. If yes, then the condition becomes true.	(x === y) is not true.
!== (Strict inequality)	It checks whether the value and data type of the variable is equal or not. If the values are not equal, then the condition becomes true.	(x !== y) is true.
> (Greater than)	Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.	(x > y) is not true.
< (Less than)	Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition becomes true.	(x < y) is true.
>= (Greater than or Equal to)	Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.	(x >= y) is not true.
<= (Less than or Equal to)	Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition becomes true.	(x <= y) is true.

JavaScript Logical Operators

The logical operators are generally used to perform logical operations on boolean values. But logical operators can be applied to values of any types not only boolean.

JavaScript supports the following logical operators −

Assume that the value of **x** is 10 and **y** is 0.

Operator	Description	Example
&& (Logical AND)	If both the operands are non-zero, then the condition becomes true.	(x && y) is false
|| (Logical OR)	If any of the two operands are non-zero, then the condition becomes true.	(x || y) is true.
! (Logical NOT)	Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false.	!x is false.

JavaScript Bitwise Operators

The JavaScript bitwise operators are used to perform bit-level operations on integers. JavaScript supports the following seven types of bitwise operators −

Assume variable x holds 2 and variable y holds 3, then −

Operator	Description	Example
& (Bitwise AND)	It performs a Boolean AND operation on each bit of its integer arguments.	(x & y) is 2.
| (Bitwise OR)	It performs a Boolean OR operation on each bit of its integer arguments.	(x | y) is 3.
^ (Bitwise XOR)	It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.	(x ^ y) is 1.
~ (Bitwise Not)	It is a unary operator and operates by reversing all the bits in the operand.	(~y) is -4.
<< (Left Shift)	It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on.	(x << 1) is 4.
>> (Right Shift)	Binary Right Shift Operator. The left operand’s value is moved right by the number of bits specified by the right operand.	(x >> 1) is 1.
>>> (Right shift with Zero)	This operator is just like the >> operator, except that the bits shifted in on the left are always zero.	(x >>> 1) is 1.

JavaScript Assignment Operators

In JavaScript, an assignment operator is used to assign a value to a variable. JavaScript supports the following assignment operators −

Operator	Description	Example
= (Simple Assignment)	Assigns values from the right side operand to the left side operand	z = x + y will assign the value of x + y into z
+= (Add and Assignment)	It adds the right operand to the left operand and assigns the result to the left operand.	z += x is equivalent to z = z + x
−= (Subtract and Assignment)	It subtracts the right operand from the left operand and assigns the result to the left operand.	z -= x is equivalent to z = z - x
*=(Multiply and Assignment)	It multiplies the right operand with the left operand and assigns the result to the left operand.	z *= x is equivalent to z = z * x
/= (Divide and Assignment)	It divides the left operand with the right operand and assigns the result to the left operand.	z /= x is equivalent to z = z / x
%= (Modules and Assignment)	It takes modulus using two operands and assigns the result to the left operand.	z %= x is equivalent to z = z % x

Same logic applies to Bitwise operators so they will become like <<=, >>=, >>=, &=, |= and ^=.

JavaScript Miscellaneous Operators

There are few other operators supported by JavaScript. These operators are **conditional** operator (? :), **typeof** operator, **delete** operator, etc.

In the below table, we have given the JavaScript miscellaneous operators with its explanation.

Operator	Description
? : (Conditional)	If Condition is true? Then value X : Otherwise value Y
typeof	It returns the data type of the operand.
?? (Nullish Coalescing Operator)	It returns its right-hand side operand when its left-hand side operand is null or undefined, and otherwise returns its left-hand side operand.
delete	It removes a property from an object.
, (Comma)	It evaluates its operands (from left to right) and returns the value of the last operand.
() (Grouping)	It allows to change the operator precedence.
yield	It is used to pause and resume a generator function.
… (Spread)	It is used to expand the iterables such as array or string.
** (Exponentiation)	Raises the left operand to the power of the right operand