JAVASCRIPT - OPERATORS

What is an operator?

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
- Comparision Operators
- Logical *orRelational* Operators
- Assignment Operators
- Conditional *orternary* Operators

Lets have a look on all operators one by one.

Arithmetic Operators

JavaScript supports the following arithmetic operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No	Operator and Description
1	**+ *Addition*** Adds two operands **Ex:** A + B will give 30
2	**- *Subtraction*** Subtracts the second operand from the first **Ex:** A - B will give -10
3	*** *Multiplication*** Multiply both operands **Ex:** A * B will give 200
4	**/ *Division*** Divide the numerator by the denominator **Ex:** B / A will give 2
5	**% *Modulus*** Outputs the remainder of an integer division

	Ex: B % A will give 0
6	**++ *Increment*** Increases an integer value by one **Ex:** A++ will give 11
7	**-- *Decrement*** Decreases an integer value by one **Ex:** A-- will give 9

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

Example

The following code shows how to use arithmetic operators in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 33;
            var b = 10;
            var c = "Test";
            var linebreak = "<br />";

            document.write("a + b = ");
            result = a + b;
            document.write(result);
            document.write(linebreak);

            document.write("a - b = ");
            result = a - b;
            document.write(result);
            document.write(linebreak);

            document.write("a / b = ");
            result = a / b;
            document.write(result);
            document.write(linebreak);

            document.write("a % b = ");
            result = a % b;
            document.write(result);
            document.write(linebreak);

            document.write("a + b + c = ");
            result = a + b + c;
            document.write(result);
            document.write(linebreak);

            a = a++;
            document.write("a++ = ");
            result = a++;
            document.write(result);
            document.write(linebreak);

            b = b--;
            document.write("b-- = ");
            result = b--;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      Set the variables to different values and then try...
   </body>
</html>
```

Output

```
a + b = 43
a - b = 23
a / b = 3.3
a % b = 3
a + b + c = 43Test
a++ = 33
b-- = 10
Set the variables to different values and then try...
```

Comparison Operators

JavaScript supports the following comparison operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No	Operator and Description
1	**= = *Equal*** Checks if the value of two operands are equal or not, if yes, then the condition becomes true. **Ex:** $A == B$ is not true.
2	**!= *NotEqual*** Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. **Ex:** $A! = B$ is true.
3	**> *Greaterthan*** Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. **Ex:** $A > B$ is not true.
4	**< *Lessthan*** Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. **Ex:** $A < B$ is true.

5

>= ***GreaterthanorEqualto***

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.

Ex: $A >= B$ is not true.

6

<= ***LessthanorEqualto***

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.

Ex: $A <= B$ is true.

Example

The following code shows how to use comparison operators in JavaScript.

```
<html>
   <body>
   
      <script type="text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";
         
            document.write("(a == b) => ");
            result = (a == b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a < b) => ");
            result = (a < b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a > b) => ");
            result = (a > b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a != b) => ");
            result = (a != b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a >= b) => ");
            result = (a >= b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a <= b) => ");
            result = (a <= b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      
      Set the variables to different values and different operators and then try...
   </body>
</html>
```

Output

```
(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
(a >= b) => false
a <= b) => true
Set the variables to different values and different operators and then try...
```

Logical Operators

JavaScript supports the following logical operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No	Operator and Description
1	**&&** ***LogicalAND*** If both the operands are non-zero, then the condition becomes true. **Ex:** A && B is true.
2	**||** ***LogicalOR*** If any of the two operands are non-zero, then the condition becomes true. **Ex:** $A||B$ is true.
3	**!** ***LogicalNOT*** Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. **Ex:** ! A && B is false.

Example

Try the following code to learn how to implement Logical Operators in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = true;
            var b = false;
            var linebreak = "<br />";

            document.write("(a && b) => ");
            result = (a && b);
            document.write(result);
            document.write(linebreak);

            document.write("(a || b) => ");
            result = (a || b);
            document.write(result);
            document.write(linebreak);

            document.write("!(a && b) => ");
            result = (!(a && b));
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
try...</p>
   </body>
</html>
```

Output

```
(a && b) => false
(a || b) => true
!(a && b) => true
Set the variables to different values and different operators and then try...
```

Bitwise Operators

JavaScript supports the following bitwise operators −

Assume variable A holds 2 and variable B holds 3, then −

Sr.No	Operator and Description
1	**& *BitwiseAND*** It performs a Boolean AND operation on each bit of its integer arguments. **Ex:** A & B is 2.
2	**| *BitWiseOR*** It performs a Boolean OR operation on each bit of its integer arguments. **Ex:** $A\|B$ is 3.
3	**^ *BitwiseXOR*** 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. **Ex:** A^B is 1.
4	**~ *BitwiseNot*** It is a unary operator and operates by reversing all the bits in the operand. **Ex:** B is -4.
5	**<< *LeftShift*** 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.

Ex: $A << 1$ is 4.

6

>> ***RightShift***

Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand.

Ex: $A >> 1$ is 1.

7

>>> ***RightshiftwithZero***

This operator is just like the >> operator, except that the bits shifted in on the left are always zero.

Ex: $A >>> 1$ is 1.

Example

Try the following code to implement Bitwise operator in JavaScript.

```
<html>
   <body>
   
      <script type="text/javascript">
         <!--
            var a = 2; // Bit presentation 10
            var b = 3; // Bit presentation 11
            var linebreak = "<br />";
         
            document.write("(a & b) => ");
            result = (a & b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a | b) => ");
            result = (a | b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a ^ b) => ");
            result = (a ^ b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(~b) => ");
            result = (~b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a << b) => ");
            result = (a << b);
            document.write(result);
            document.write(linebreak);
         
            document.write("(a >> b) => ");
            result = (a >> b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
try...</p>
   </body>
</html>
```

```
(a & b) => 2
(a | b) => 3
(a ^ b) => 1
(~b) => -4
(a << b) => 16
(a >> b) => 0
Set the variables to different values and different operators and then try...
```

Assignment Operators

JavaScript supports the following assignment operators −

Sr.No	Operator and Description
1	**=** ***SimpleAssignment*** Assigns values from the right side operand to the left side operand **Ex:** C = A + B will assign the value of A + B into C
2	**+=** ***AddandAssignment*** It adds the right operand to the left operand and assigns the result to the left operand. **Ex:** C += A is equivalent to C = C + A
3	**−=** ***SubtractandAssignment*** It subtracts the right operand from the left operand and assigns the result to the left operand. **Ex:** C -= A is equivalent to C = C - A
4	***=** ***MultiplyandAssignment*** It multiplies the right operand with the left operand and assigns the result to the left operand. **Ex:** C *= A is equivalent to C = C * A
5	**/=** ***DivideandAssignment*** It divides the left operand with the right operand and assigns the result to the left operand. **Ex:** C /= A is equivalent to C = C / A
6	**%=** ***ModulesandAssignment***

It takes modulus using two operands and assigns the result to the left operand.

Ex: C %= A is equivalent to C = C % A

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

Example

Try the following code to implement assignment operator in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 33;
            var b = 10;
            var linebreak = "<br />";

            document.write("Value of a => (a = b) => ");
            result = (a = b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a += b) => ");
            result = (a += b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a -= b) => ");
            result = (a -= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a *= b) => ");
            result = (a *= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a /= b) => ");
            result = (a /= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a %= b) => ");
            result = (a %= b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
try...</p>
   </body>
</html>
```

Output

```
Value of a => (a = b) => 10
Value of a => (a += b) => 20
Value of a => (a -= b) => 10
Value of a => (a *= b) => 100
Value of a => (a /= b) => 10
Value of a => (a %= b) => 0
```

```
Set the variables to different values and different operators and then try...
```

Miscellaneous Operator

We will discuss two operators here that are quite useful in JavaScript: the **conditional operator** ? : and the **typeof operator**.

Conditional Operator ? :

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Sr.No	Operator and Description
1	**? : *Conditional*** If Condition is true? Then value X : Otherwise value Y

Example

Try the following code to understand how the Conditional Operator works in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";

            document.write ("((a > b) ? 100 : 200) => ");
            result = (a > b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);

            document.write ("((a < b) ? 100 : 200) => ");
            result = (a < b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
try...</p>
   </body>
</html>
```

Output

```
((a > b) ? 100 : 200) => 200
((a < b) ? 100 : 200) => 100
Set the variables to different values and different operators and then try...
```

typeof Operator

The **typeof** operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

Here is a list of the return values for the **typeof** Operator.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

Example

The following code shows how to implement **typeof** operator.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 10;
            var b = "String";
            var linebreak = "<br />";

            result = (typeof b == "string" ? "B is String" : "B is Numeric");
            document.write("Result => ");
            document.write(result);
            document.write(linebreak);

            result = (typeof a == "string" ? "A is String" : "A is Numeric");
            document.write("Result => ");
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
try...</p>
   </body>
</html>
```

Output

```
Result => B is String
Result => A is Numeric
Set the variables to different values and different operators and then try...
```

Processing math: 100%