ES6 - Operators

An **expression** is a special kind of statement that evaluates to a value. Every expression is composed of −

- **Operands** − Represents the data.
- **Operator** − Defines how the operands will be processed to produce a value.

Consider the following expression- 2 + 3. Here in the expression, 2 and 3 are operands and the symbol + (plus) is the operator. JavaScript supports the following types of operators −

Arithmetic Operators

Assume the values in variables **a** and **b** are 10 and 5 respectively.

Show Examples

Operator	Function	Example
+	**Addition** Returns the sum of the operands.	a + b is 15
-	**Subtraction** Returns the difference of the values.	a-b is 5
*	**Multiplication** Returns the product of the values.	a*b is 50
/	**Division** Performs a division operation and returns the quotient.	a/b is 2
%	**Modulus** Performs a division and returns the remainder.	a%b is 0
++	**Increment** Increments the value of the variable by one.	a++ is 11
--	**Decrement** Decrements the value of the variable by one.	a-- is 9

Relational Operators

Relational operators test or define the kind of relationship between two entities. Relational operators return a boolean value, i.e. true/false.

Assume the value of A is 10 and B is 20.

Show Examples

Operators	Description	Example
>	Greater than	(A > B) is False
<	Lesser than	(A < B) is True
>=	Greater than or equal to	(A >= B) is False
<=	Lesser than or equal to	(A <= B) is True
==	Equality	(A == B) is False
!=	Not Equal	(A!= B) is True

Logical Operators

Logical operators are used to combine two or more conditions. Logical operators, too, return a Boolean value. Assume the value of variable A is 10 and B is 20.

Show Examples .

Operators	Description	Example
&&	**And** The operator returns true only if all the expressions specified return true.	(A > 10 && B > 10) is False
||	**Or** The operator returns true if at least one of the expressions specified return true.	(A > 10 || B > 10) is True
!	**Not** The operator returns the inverse of the expression's result. For E.g.: !(7>5) returns false.	!(A > 10) is True

Bitwise Operators

JavaScript supports the following bitwise operators. The following table summarizes JavaScript's bitwise operators.

Show Examples .

Operators	Usage	Description
Bitwise AND	a & b	Returns a one in each bit position for which the corresponding bits of both operands are ones
Bitwise OR	a | b	Returns a one in each bit position for which the corresponding bits of either or both operands are ones
Bitwise XOR	a^b	Returns a one in each bit position for which the corresponding bits of either but not both operands are ones
Bitwise NOT	~ a	Inverts the bits of its operand
Left shift	a << b	Shifts a in binary representation b (< 32) bits to the left, shifting in zeroes from the right
Sign-propagating right shift	a >> b	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off
Zero-fill right shift	a >>> b	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off, and shifting in zeroes from the left

Assignment Operators

The following table summarizes Assignment operators.

Show Examples .

Sr.No	Operator & Description
1	**= (Simple Assignment)** Assigns values from the right side operand to the left side operand. **Example** − C = A + B will assign the value of A + B into C
2	**+= (Add and Assignment)** It adds the right operand to the left operand and assigns the result to the left operand. **Example** − C += A is equivalent to C = C + A
3	**-= (Subtract and Assignment)** It subtracts the right operand from the left operand and assigns the result to the left operand. **Example** C -= A is equivalent to C = C - A
4	***= (Multiply and Assignment)** It multiplies the right operand with the left operand and assigns the result to the left operand. **Example** C *= A is equivalent to C = C * A
5	**/= (Divide and Assignment)** It divides the left operand with the right operand and assigns the result to the left operand.

Note − The same logic applies to Bitwise operators, so they will become <<=, >>=, >>=, &=, |= and ^=.

Miscellaneous Operators

Following are some of the miscellaneous operators.

The negation operator (-)

Changes the sign of a value. The following program is an example of the same.

```
var x = 4
var y = -x;
console.log("value of x: ",x); //outputs 4
console.log("value of y: ",y); //outputs -4
```

The following output is displayed on successful execution of the above program.

```
value of x:  4
value of y:  -4
```

String Operators : Concatenation operator (+)

The + operator when applied to strings appends the second string to the first. The following program helps to understand this concept.

```
var msg = "hello"+"world"
console.log(msg)
```

The following output is displayed on successful execution of the above program.

```
helloworld
```

The concatenation operation doesn't add a space between the strings. Multiple strings can be concatenated in a single statement.

Conditional Operator (?)

This operator is used to represent a conditional expression. The conditional operator is also sometimes referred to as the ternary operator. Following is the syntax.

```
Test ? expr1 : expr2
```

Where,

Test − Refers to the conditional expression

expr1 − Value returned if the condition is true

expr2 − Value returned if the condition is false

Example

```
var num = -2
var result = num > 0 ?"positive":"non-positive"
console.log(result)
```

Line 2 checks whether the value in the variable num is greater than zero. If num is set to a value greater than zero, it returns the string “positive” else a “non-positive” string is returned.

The following output is displayed on successful execution of the above program.

```
non-positive
```

Type Operators

typeof operator

It is a unary operator. This operator returns the data type of the operand. The following table lists the data types and the values returned by the **typeof** operator in JavaScript.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"

The following example code displays the number as the output.

```
var num = 12
console.log(typeof num); //output: number
```

The following output is displayed on successful execution of the above code.

```
number
```