



PRACTICE

COMPETE

JOBS

LEADERBOARD

Search



shaunakbhanarkar ▾

[Dashboard](#) > [Tutorials](#) > [10 Days of Javascript](#) > [Day 1: Arithmetic Operators](#)

# Day 1: Arithmetic Operators

by [AvmnuSng](#)

Problem

Submissions

Leaderboard

Discussions

Editorial

Tutorial

## Arithmetic Operators

### Operator Types

#### Unary

A *unary* operator requires a single operand, either before or after the operator, following this format:

```
operand operator  
operator operand
```

For example, in the expression `a++`, `++` is a unary operator.

#### Binary

A *binary* operator requires two operands, one before the operator and one after the operator, following this format:

```
operand1 operator operand2
```

For example, in the expression `a + b = c`, `+` is a binary operator.

#### Ternary

There is one *ternary* operator, the conditional operator. For example, in the expression `a ? b : c`, the use of `?` and `:` in this manner constitutes the ternary operator. We'll discuss this operator more in the *Conditional Statements* tutorial.

### Arithmetic Operators

An arithmetic operator takes numeric values (either literals or variables) as its operands and returns a single numeric value. The standard arithmetic operators are addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`). Other arithmetic operators are remainder (`%`), unary negation (`-`), unary plus (`+`), increment (`++`), decrement (`--`), and

T.  
O  
A

exponentiation ( \*\* ).

## 1. Addition ( + )

We use this operator in the form `operand1 + operand2`. For example:

```
2 + 3 // evaluates to 5
4 + 10 // evaluates to 14
```

## 2. Subtraction ( - )

We use this operator in the form `operand1 - operand2`. For example:

```
3 - 2 // evaluates to 1
4 - 10 // evaluates to -6
```

## 3. Multiplication ( \* )

We use this operator in the form `operand1 * operand2`. For example:

```
3 * 2 // evaluates to 6
4 * 10 // evaluates to 40
```

## 4. Division ( / )

We use this operator in the form `operand1 / operand2`. For example:

```
6 / 3 // evaluates to 2
3 / 2 // evaluates to 1.5
4 / 10 // evaluates to 0.4
```

## 5. Remainder ( % )

We use this operator in the form `operand1 % operand2`. For example:

```
6 % 3 // evaluates to 0
3 % 2 // evaluates to 1
4 % 10 // evaluates to 4
```

## 6. Exponentiation ( \*\* )

We use this operator in the form `operand1 ** operand2`. This operator is a part of ECMAScript2016 feature set. For example:

```
2 ** 3 // evaluates to 8
3 ** 2 // evaluates to 9
5 ** 4 // evaluates to 625
```

## 7. Unary Negation ( - )

We use this operator in the form `-operand`. For example:

```
-4 // evaluates to -4
-(-5) // evaluates to 5 (not --5)
```

## 8. Unary Plus ( + )

We use this operator in the form `+operand`. For example:

[Go to Top](#)

```
+4 // evaluates to 4
+(-4) // evaluates to -4
```

## 9. Increment (++)

We use this operator in the prefix and postfix forms, forms `++operand` and `operand++`. The prefix form, `++operand`, increments the operand by **1** and then returns the value of the operand. The postfix form, `operand++`, returns the value of the operand and *then* increments the operand's value by **1**. For example:

-	EXAMPLE
1	<code>process.stdin.on('data', function (data) {</code>
2	<code>  main(+ (data));</code>
3	<code>});</code>
4	<code>/**** Ignore above this line. ****/</code>
5	<code>function main(input) {</code>
6	<code>  var a = input;</code>
7	<code>  // Print the value of 'a' and the preincremented value of 'a':</code>
8	<code>  console.log("a(" + a + "), ++a(" + ++a + ")");</code>
9	<code>  // Assign the current value of 'a' to 'b' and then postincrement</code>
10	<code>  var b = a++;</code>
11	<code>  // Print the values of 'a' once and 'b' twice, then postincrement</code>
12	<code>  console.log("a(" + a + "), b(" + b + "), b++(" + b++ + ")");</code>
13	<code>  // Print the final values of 'a' and 'b':</code>
14	<code>  console.log("a(" + a + "), b(" + b + ")");</code>
15	<code>}</code>

Input

Run

Output

Solution

The code above produces this output:

```
a(4), ++a(5)
a(6), b(5), b++(5)
a(6), b(6)
```

## 10. Decrement (--)

We use this operator in the prefix and postfix forms, forms `--operand` and `operand--`. The prefix form, `--operand`, decrements the operand by **1** and then returns the value of the operand. The postfix form, `operand--`, returns the value of the operand and *then* decrements the operand's value by **1**. For example:

-	EXAMPLE
---	---------

```
1 process.stdin.on('data', function (data) {
2   main(+ (data));
3 });
4 /**** Ignore above this line. ****/
5 function main(input) {
6   var a = input;
7   // Print the value of 'a' and the predecremented value of 'a':
8   console.log("a(" + a + "), --a(" + --a + ")");
9   // Assign the current value of 'a' to 'b' and then postdecrement
10  var b = a--;
11  // Print the values of 'a' once and 'b' twice, then postdecrement
12  console.log("a(" + a + "), b(" + b + "), b--(" + b-- + ")");
13  // Print the final values of 'a' and 'b':
14  console.log("a(" + a + "), b(" + b + ")");
15 }
```

Input

4

Run

Output

### Solution

The code above produces this output:

```
a(4), --a(3)
a(2), b(3), b--(3)
a(2), b(2)
```

-

IF-ELSE CONDITIONAL STATEMENTS

Recommended Article

If-Else Conditional Statements

View

-

SWITCH CONDITIONAL STATEMENTS

Recommended Article

Switch Conditional Statements

View