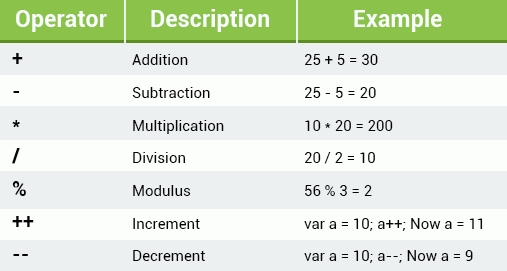
**Step 3.**

**Arithmetic Operators**

The name might be a bit of a giveaway but, **Arithmetic operators** pretty perform arithmetic functions on numbers (both literals and variables).



Below you can see the **addition operator** (+) in action determining the sum of two numbers.

var x = 10 + 5;   
document. Write (x);

You can add as many numbers or variables together as you want or need to.

var x = 10;   
var y = x + 5 + 22 + 45 + 6548;   
document. write (y);

**Heads up!**  
You can get the result of a string expression using the eval() function, which takes a string expression argument like eval("10 \* 20 + 8") and returns the result. If the argument is empty, it returns undefined.

**Multiplication**

Want to hear a joke?  
What tool is best suited for math?...Multi-pliers!  
JavaScript is pretty good at math too though!  
  
We use the **\* operator** to multiply one number by the other.  
**Like this:**

var x = 10 \* 5;   
document.write(x);

**Heads up!**  
10 \* '5' or '10' \* '5' will give the same result. But trying to multiply a number with string values that aren’t numbers, like 'sololearn' \* 5 will return NaN (Not a Number).

**Division**

What’s a swimmer's favorite kind of math?... Dive-ision!  
Sorry! Back to business!  
  
We use the **/ operator** to perform division operations.  
**Like this:**

var x = 100 / 5;   
document.write(x);

**Heads up!**  
Beware of situations where there could be a division by 0, things get messed up when we do impossible math!

**The Modulus**

Time to talk remainders. You hated them in school, but they’re pretty easy here, promise.  
  
The **Modulus (%) operator** returns the division remainder (what’s left over).  
  
**Like this:**

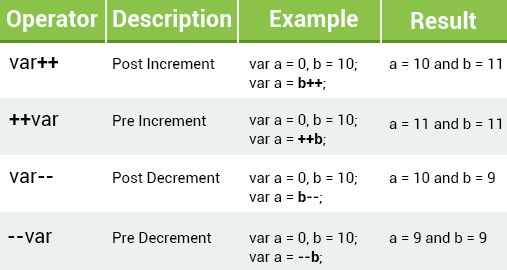
var myVariable = 26 % 6;

Because you’re left with a remainder of 2 when you divide 26 by 6.

**Heads up!**  
In JavaScript, we can use the modulus operator on integers AND on floating point numbers.

**Increment & Decrement**

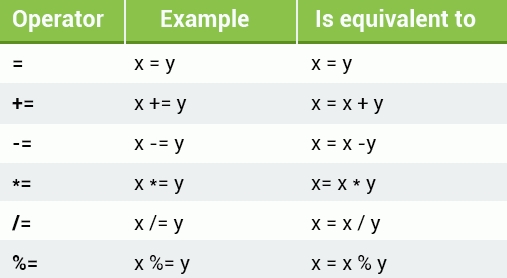
"Wait I've heard of an increment, but what the heck is a decrement?" We hear some of you say. Well, throw an increment into reverse and presto, you got yourself a decrement. Let’s dig a little deeper...  
  
**Increment ++**  
The increment operator **increases** the numeric value of its operand **by 1**. When placed before the operand, it’ll return the incremented value. When placed after it, it’ll return the original value and then increments the operand.  
  
**Decrement --**  
The decrement operator **decreases** the numeric value of its operand **by 1**. When placed before the operand, it’ll return the decremented value. When placed after the operand, it’ll return the original value and then decrements the operand.  
  
**Some examples:**



**Heads up!**  
Just like the math you learned in school, you can change the order of the arithmetic operations by using parentheses.  
Like this: var x = (100 + 50) \* 3;

**Assignment Operators**

Next in a series very logically named operators is...  
**Assignment operators!**  
  
And you guessed it, we use these guys to assign values to JavaScript **variables**.



**Heads up!**  
You can use multiple assignment operators in one line, such as x -= y += 9.