JavaScript for App Development

By Mark Lassoff, Founder Framework Television

Section Three

Section 3 Goals

In this section of the course your goals are:

- [ ] Understand Arithmetic Operators
- [ ] Apply Arithmetic Operators to coding problems
- [ ] Understand post-fix and pre-fix increment and decrement operators.
- [ ] Understand logical Operators

Watch This: Section 3 Video

As always your course videos are available on YouTube, Roku and other locations. However, only those officially enrolled have access to this course guide, are able to submit assignments, work with the instructor, and get this guide.

Watch this section video at: https://www.youtube.com/watch?v=rfnUcpfefhs&t=8s

Understanding the Arithmetic Operators

The original purpose of computing was to solve complex mathematical problems. While you certainly don't have to be a math expert to successfully code, you do have to understand the basics of arithmetic. Many different aspects of software depend on arithmetic to some degree. Game development, financial software, graphics, audio all have mathematically based algorithms.

Now don't let this scare you in to running out and taking a math class. If you understand arithmetic, you'll do just fine.

So here are the fundamental mathematical operators:

Operator	Description
+	Addition Operator. (Also the concatenation operator, but that's not arithmetic!)
-	Subtraction operator.
*	Multiplication Operator.
/	Division Operator.
%	Modulus operator. This operator gives you the remainder after a division operator and is useful to determine odd and even numbers. 10 % 2 = 0.
++	Increment Operator. Adds one as in x++ adds one to the value of x.
--	Decrement Operator. Subtracts one as in x-- subtracts one from the value of x.

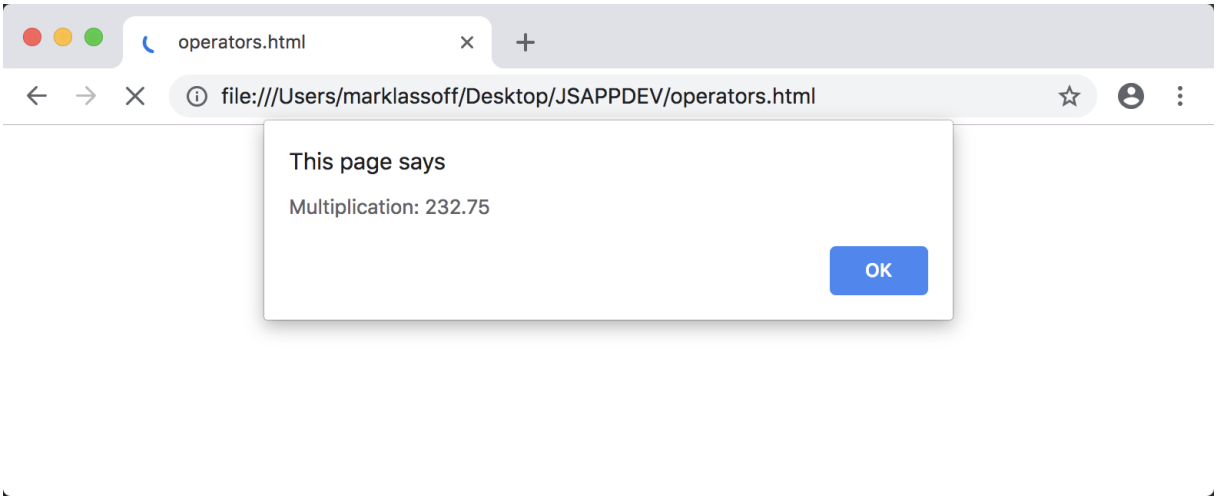
Most of these should be pretty familiar to you. I didn't discover increment, decrement and modulus until I started coding, so if these are unfamiliar to you, you are very likely not alone!

Doing the Math

Of course, these operators are only useful if you are going to put them to work in your code. Let's write a program first that uses all of the standard mathematical operators.

```
<div id="output"></div>
<script>
    var x = 12.25;     //Lazy Variable Names!
    var y = 19;

    alert("Addition: " + (x+y));
    alert("Subtraction: " + (x-y));
    alert("Multiplication: " + (x*y));
    alert("Division: " + (x/y));

</script>
```

The parentheses surrounding the the actual arithmetic give the expression primacy-- so it's evaluated first before the rest of the line is processed. This is a good idea to the '+' for concatenation doesn't get confused with the arithmetic expression.



Modulus

The first of the arithmetic operators that is likely new to you is the modulus operator. It provides the remainder after division has taken place. For example:

```
10 % 3 = 1
15 % 4 = 3
6 %  3 = 0
19 % 2 = 1
```

Hopefully you get the idea.

Do This: Get To Know Your New Friend, Modulus

Write a program that prompts the user twice for a number between 1 and 1,000. When the user types the number store it in a variable. When the user has provided both numbers perform the modulus operation and output the result to the user. Test to make sure your program is working correctly.

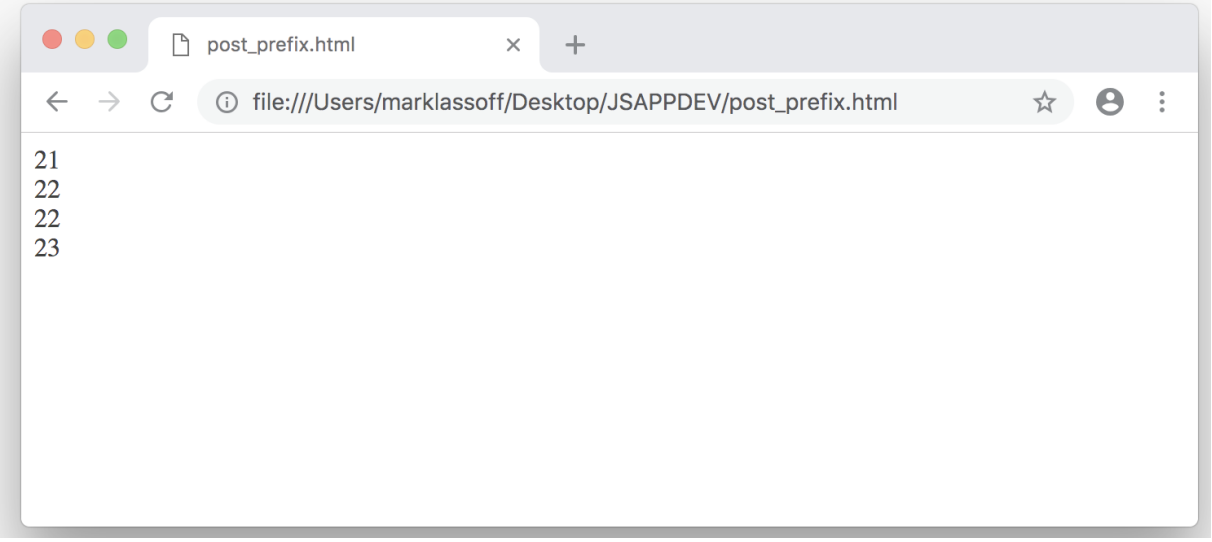
Postfix and Prefix

The increment and decrement operators seem fairly easy to understand. Right? There is one wrinkle that makes things a bit more complex, however. Increment and decrement can be applied postfix or preFix. Essentially a pre-fix operator is applied before the rest of the expression is evaluated. A post-fix expression is applied after the rest of the expression is evaluated. Take a look at the following program. (You're definitely going to want to enter this one in to your text editor and follow the execution as it runs!)

```
<div id="output"></div>
<script>
    var x = 20;
    var y = 25;

    x++;
    document.getElementById('output').innerHTML = x  + "<br/>";
    document.getElementById('output').innerHTML +=  ++x + "<br/>";
    document.getElementById('output').innerHTML += x++ + "<br/>";
    document.getElementById('output').innerHTML += x++ + "<br/>";
</script>
```

Let's take a closer look at the program and the output.

```
1  <div id="output"></div>
2  <script>
3      var x = 20;
4      var y = 25;
5
6      x++;
7      document.getElementById('output').innerHTML = x  + "<br/>";
8      document.getElementById('output').innerHTML +=  ++x + "<br/>";
9      document.getElementById('output').innerHTML += x++ + "<br/>";
10     document.getElementById('output').innerHTML += x++ + "<br/>";
11 </script>
12
```



Initially we assigned to x the value 20. On line six we increment that by one. So when the value of x is output on line 7 we see 21 for the first value.

On line eight we apply a *prefix increment* and we see the value of x increase to 22. However on lines nine and ten we apply a *postfix increment* so the value is displayed and then the increment is operator is applied, which is why the next output is 22 again. It is applied, however, as we can see with the next output which is 23.

After this code is all executed what is the last value held by x?

Do This: Make Sure You Understand

Using the y variable go through a similar series of steps with postfix and prefix decrements and see if you can predict and follow what happens.

Being Logical

In addition to the arithmetic operators that we just reviewed, JavaScript depends on logical operators,. When a logical operator is evaluated the result is always true of false. (Variable can store 'true' and 'false' as values and when they do they are known as Booleans.

Let's take a look at the logical operators.

Operator	Description
==	Equivalency operator. Returns true if the values are equal.
===	Equivalency operator. This tests for value and type. For example: `4 === 4` is true but `4 === "4"` is false.
!=	Not equivalent
>	Greater Than
<	Less Than
>=	Greater Than or Equal To
<=	Less Than or Equal To

We'll put these to greater use once we start working with conditionals in the next section. For now you can test them with various expressions like this:

```
<script>
    var x = 5;
    var y = 10;

    alert(x < y);
    alert(x == y);
    alert("4" === 4);
    alert("4" == 4);
</script>
```

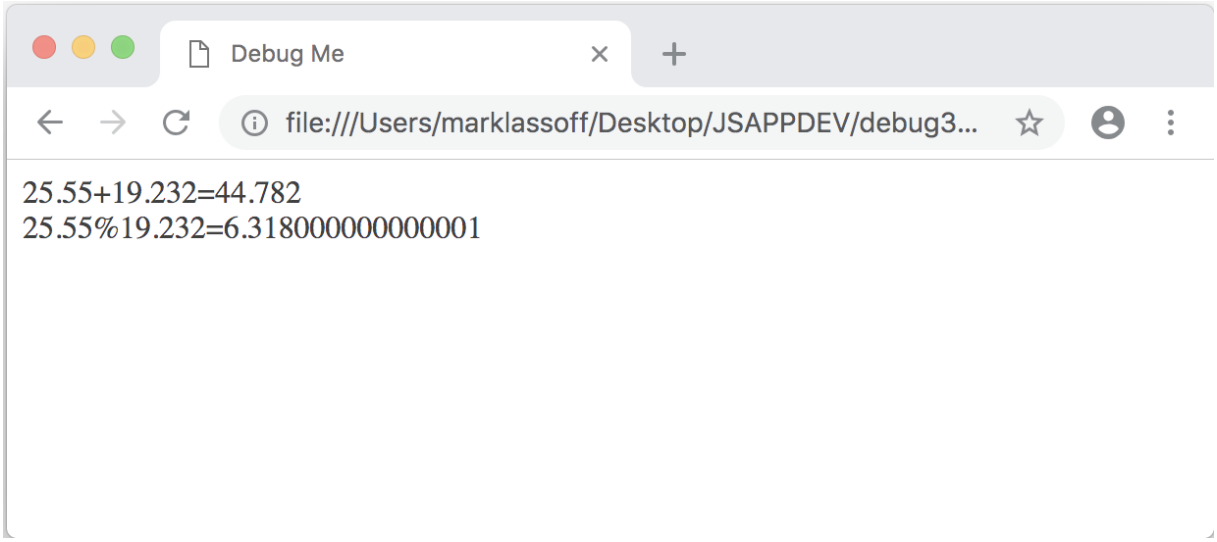
Run some tests yourself using each of the logical operators.

Do This: Debugging

This code has several errors. See if you can debug it and get it working as it should. When working correctly it should display a story, inserting a nouns, adjectives and verbs as indicated. As is true in the real world, these errors are not always apparent and easy to spot. Good luck.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Debug Me</title>
    </head>
    <body>
        <div id="output"></div>
        <script>
            var x = 25.55;
            var y = 19.232;

            var add = x + y;
            var mod = x % y;

            var out == "";
            out += x + "+" + y + "=" + (x+y;
            out += "<br/>";
            out += x + "%" + y   "=" + x%y);

            document.getElementById('output').innerHTML = out;

    </body>
</html>
```

If you debug correctly your output should look like this:



Submit This: Lab Exercise

Create a program that prompts the user twice for numbers. Store those variables in variables called

`operand1` and `operand2`. Using mathematical operators, display the answers to the following questions:

What is the sum of operand1 and operand2?
What is the result if you subtract operand2 from operand1?
What is the result if you multiple operand1 by operand2?
Is operand1 equal to operand2?
Is operand1 smaller than operand2?
Can you divide Operand2 evenly in to operand1?

Please save your file in the following format to insure proper credit:
`LastName_Exercise3.html`.

Remember every exercise must be submitted in order for you to earn certification. Once you've completed this exercise, you're ready to move on to Section 4.

Submit your exercises to: https://www.dropbox.com/request/Nl7bSaAe11Zl0PgLRonA