Operators, Arithmetic, and Methods

Operators

In javascript you can use the following operators.

+	Addition
-	Subtraction
*	Multiplication
/	Division
++	Increment
--	Decrement

Addition

Addition is used by the + symbol in your code

When adding number data types together it will perform normal addition

When adding string data types together it will combine the strings

```
var x = 3;

var y = 2;

console.log( x + y);

Output: 5
```

```
var x = “3”;

var y = “2”;

console.log(x + y);

Output: “32”
```

Lets try it

```
var x = 3;

var y = 2;

console.log( x + y);
```

Output: 3

Subtraction

Subtraction is used by the - symbol in your code

When subtracting number data types together it will perform normal subtraction

If you subtract 2 strings your javascript will attempt to convert the strings to numbers and if it cannot it will error as NaN (Not a number)

```
var x = 3;

var y = 2;

console.log( x - y);

Output: 3
```

Multiplication

Multiplication is used by the * symbol in your code

When multiplying number data types together it will perform normal multiplication

If you multiply 2 strings your javascript will attempt to convert the strings to numbers and if it cannot it will error as NaN (Not a number)

```
var x = 3;

var y = 2;

console.log( x * y);

Output: 6
```

Division

Division is used by the / symbol in your code

When dividing number data types together it will perform normal division

If you divide 2 strings your javascript will attempt to convert the strings to numbers and if it cannot it will error as NaN (Not a number)

```
var x = 6;

var y = 2;

console.log( x / y);

Output: 3
```

Lets try it

```
var x = 6;

var y = 2;

console.log( x - y);

console.log( x * y);

console.log( x / y);
```

What about multiple operators (Arithmetic) in a line of code?

Javascript will follow the order of operations when there are multiple operators in a line of code.

```
var x = 2;

var y = 3;

var z = 4;

console.log( x + y * z); //Based on order of operations y * z will come first and then x will be added

Output: 14
```

Don't forget the parentheses!!

Just like with normal order of operations you can use parentheses

```
var x = 2;

var y = 3;

var z = 4;

console.log( (x + y) * z); //Based on order of operations x + y will come first and then z will be multiplied

Output: 20
```

Lets try it

```
var x = 2;

var y = 3;

var z = 4;

console.log( (x + y) * z); //Based on order of operations x + y will come first and then z will be multiplied

Output: 20
```

Increment

Increment is used by the ++ symbols in your code

Increment adds 1 to the number

```
var x = 6;

x++;

console.log( x);

Output: 7
```

Decrement

Decrement is used by the – symbols in your code

Decrement subtracts 1 from the number

```
var x = 6;

x- -;

console.log( x);

Output: 5
```

What's the point of this ++ and - -??

Incrementing and Decrementing will become very important when we start learning about loops

Lets try it

```
var x = 6;

x++;


console.log( x);


Output: 7
```

```
var x = 6;

x- -;


console.log( x);


Output: 5
```

Functions

We have already learned about Functions when we used onClick with buttons but Functions can do more!

A few topics we'll review on functions:

- Passing variables into a function
- Ending a function early
- Returning a value from a function

Why use Functions?

Functions help reduce redundant code

Functions help organize code

Function help break large pages of code into easier to understand blocks

Review of how to declare a function

```
function NAMEYOUASSIGN(){
        CODE
}
```

How to call a function in Javascript

You can call a function from Javascript

```
sendAlert();


function sendAlert(){
        alert(“Hello”);
}
```

Lets try it

```
sendAlert();


function sendAlert(){

        alert(“Hello”);

}
```

Passing a variable in a function

You can pass variables in a function and then use that variable in your function

```
sendAlert(“hello”);

function sendAlert(x){

        alert(x);

}
```

Lets try it

```
sendAlert(“hello”);

function sendAlert(x){

        alert(x);

}
```

Ending a function early

You can end a function by using the return keyword. Anything below this line will not be executed.

```
sendAlert(“hello”);

function sendAlert(x){

        return;

        alert(x);

}
```

Lets try it

```
sendAlert(“hello”);

function sendAlert(x){

        return;

        alert(x);

}
```

Returning a value from a function

Like passing variables into a function you can also return a value from your function to do this use the return keyword we just learned

To get the return value make sure you assign it to a variable

```
var x = getMessage();

function getMessage(){

        return “Hello”;

}
```

Lets try it

```
var x = getMessage();

function getMessage(){

        return “Hello”;

}
```