**Loops and nested loops**

Let's say I want to output a custom multiplication table.

This is a perfect use case scenario for nested loops.

The outer loop's counter variable will act as the first number to be multiplied, and the inner loop counter variable will act as the second number to be multiplied.

Here's my code:

//single loop

for (var firstNum = 0; firstNum < 2; firstNum++) {

    console.log(firstNum);

}

The output of the above code will be:

0 1

This means that my for loop starts at 0 and stops after 1.

So now I can code what will later become the inner loop, whose counter variable will be the second number in this multiplication:

//single loop

for (var secondNum = 0; secondNum < 10; secondNum++) {

    console.log(secondNum);

}

This time, the output is:

0 1 2 3 4 5 6 7 8 9

Now's the time to combine the first and the second loop:

//nested loops - one inside another

for (var firstNum = 0; firstNum < 2; firstNum++) {

    for (var secondNum = 0; secondNum < 10; secondNum++) {

        console.log(firstNum + ", " + secondNum);

    }

}

Now that I'm nesting the second for loop inside the first one, and that I'm console logging the values of both counter variables as the loops are progressing, the output looks like this:

0, 0 0, 1 0, 2 0, 3 0, 4 0, 5 0, 6 0, 7 0, 8 0, 9 1, 0 1, 1 1, 2 1, 3 1, 4 1, 5 1, 6 1, 7 1, 8 1, 9

Now that I have a list of all the numbers that will be multiplied, having the actual result of this multiplication is as easy as updating the **console.log()** call:

//nested loops - one inside another

for (var firstNum = 0; firstNum < 2; firstNum++) {

    for (var secondNum = 0; secondNum < 10; secondNum++) {

        console.log(firstNum + " times " + secondNum + " equals " + firstNum \* secondNum);

    }

}

The output now is:

0 times 0 equals 0 0 times 1 equals 0 0 times 2 equals 0 0 times 3 equals 0 0 times 4 equals 0 0 times 5 equals 0 0 times 6 equals 0 0 times 7 equals 0 0 times 8 equals 0 0 times 9 equals 0 1 times 0 equals 0 1 times 1 equals 1 1 times 2 equals 2 1 times 3 equals 3 1 times 4 equals 4 1 times 5 equals 5 1 times 6 equals 6 1 times 7 equals 7 1 times 8 equals 8 1 times 9 equals 9

This makes for some very interesting combinations.

For example, I can make a custom division table:

//nested loops - one inside another

for (var i = 100; i > 10; i = i - 10) {

    for (var j = 10; j > 4; j = j - 5) {

        console.log(i + " divided by " + j + " equals " + i / j);

    }

}

Here's the output of the above nested loop:

100 divided by 10 equals 10

100 divided by 5 equals 20

90 divided by 10 equals 9

90 divided by 5 equals 18

80 divided by 10 equals 8

80 divided by 5 equals 16

70 divided by 10 equals 7

70 divided by 5 equals 14

60 divided by 10 equals 6

60 divided by 5 equals 12

50 divided by 10 equals 5

50 divided by 5 equals 10

40 divided by 10 equals 4

40 divided by 5 equals 8

30 divided by 10 equals 3

30 divided by 5 equals 6

20 divided by 10 equals 2

20 divided by 5 equals 4

Feel free to try out some other combinations of nested loop iterations, and see what kind of output you'll get.