JavaScript Variables

PLEASE READ:

In this assignment, we will be discussing two types of loops: a For loop and a While loop.

Loops in JavaScript function the same way that they have in Python; loops continue to execute a set of instructions until some condition is no longer true - i.e. it loops through a set of instructions unless the condition is false.

Recall that for an if/else statement, there is a set of instructions that it executes if the condition is true, and a set of instructions that it executes if the condition is false. The same thing happens in For loops and While loops – the set of instructions in the loop body (what’s inside the curly braces) will repeatedly execute until the loop condition is false. When that condition is false, the loop stops executing.

For loops are generally used to repeat code a certain number of times, e.g. in the case of counting. The three parts of a For statement will be discussed later in this assignment, but can be illustrated using the following flow chart:

A While loop is similar to a For loop, but in a more generalized state. A while loop will execute the body as long as some condition is met. In practice, a for loop and a while loop can potentially do the same things (as with this assignment), however a For loop is generally for cases where we want to execute something a certain number of times (i.e. we know exactly how many times we want to repeat

Homework 7
CSE 3, Winter 2016
Due at the beginning of lab, week 9
In this homework we are going to learn about JavaScript loops and how they can make repetitive tasks easy.
it), whereas a while loop is used in all other cases (e.g. “While” the weather is sunny, do something; “While” you’re less than 21 years old, do something). This is why the flow chart for a while loop is similar:

As you can see, the main difference between a For loop and a While loop is the use of a counter. Counters count how many times a body of code is executed. The For loop utilizes a counter to keep track of how many times the instructions execute and is updated after the body of the loop is executed. A While loop also uses a counter, but the counter can be updated before the body of the loop is executed.

Today we are going to be using loops to print a sentence a given number of times and demonstrate the difference between a For loop and a While loop.

**Step 1:**
In your Homework folder save a new Notepad++ file called loops.html.

**Step 2:**
Write out the HTML skeleton (if you don't remember what this looks like refer back to one of the earlier labs).

**Step 3:**
Create `<center>` tags and inside the center tags add the following JavaScript tag:

```html
<script type = "text/javascript"> </script>
```

**Step 4:**
Now inside your script tags, let’s create two variables. The first is a variable that tells us how many times we should print the sentence we are going to create (numLoop), and a second variable that will tell us how many times we have already printed the sentence (numPrinted).
**While Loop**

Now that we have our variables in place let’s create our loop. The first kind of loop we are going to use is called a **While Loop**. A **While Loop** can be thought of as being similar to an **if statement**, only it repeats the code inside the brackets until the statement inside the parentheses (the condition) is false. While Loops in JavaScript operates the same as the While Loops you have seen in Python. In our case we want to make a loop that prints the sentence inside the brackets until the condition `numLoop == numPrinted` is reached.

**Step 1**
First let’s outline the while loop:

```javascript
while (condition) {
    document.write(some sentence);
}
```

**Step 2**
Since we want to print the sentence `numLoop` times, we need to use our `numPrinted` variable to check if we have printed the sentence `numLoop` times:

```javascript
while (numPrinted < numLoop) {
    document.write(some sentence);
}
```

This loop is saying:

“**while** the number of times we have printed is **less than** the number of times we want to print it, print it again”

**Step 3**
At some point we need `numPrinted` to be equal to `numLoop` so that the condition is rendered false, allowing us to break out of the loop. To achieve this, we increment `numPrinted` each time we go through the loop. Add the following line before our call to `document.write()`.

```
numPrinted++;
```

This is called a **loop counter**, and it will count the number of times the loop has been executed.
All together it should look like this:

```javascript
while (numPrinted < numLoop) {
    numPrinted ++;
    document.write(some sentence);
}
```

**NOTE:** when we use the ‘++’ operator after a variable it means that we are incrementing the variable by 1. `numPrinted++` is simply shorthand for saying `numPrinted = numPrinted + 1`

**Step 4**
Now we need to put a sentence inside the parenthesis where it says `some sentence`. Use the sentence:

“While loop: \(X\)"

Where \(X\) is the number of times the loop has been printed (remember our variable?)

To accomplish this we are going to use what is called string concatenation.

**Step 5**
Since we know that `numPrinted` counts how many times our `sentence` has been printed, if we substitute it for \(X\) it should give us the right result. We do this using the ‘+’ operator:

```javascript
document.write("While loop: " + numPrinted + "</br>"));
```

Note 1: Only the words and the break tag should be inside the quotation marks
Note 2: Notice that there is a space after the colon and before the first closing quotation mark. This is so our sentence is printed:

```
While loop: X
```

And not:

```
While loop: X
```

Once this is completed, the output should look like the following:

```
While loop: 1
While loop: 2
While loop: 3
While loop: 4
While loop: 5
While loop: 6
While loop: 7
While loop: 8
While loop: 9
While loop: 10
While loop: 11
```
Step 6
Right now, our print statements seem a bit boring in black font. Let’s fix that by having them print in red. We can do this by adding an extra line of code and altering our `document.write()` function:

```javascript
var text = "While loop: " + numPrinted + "</br>";
document.write(text.fontcolor("red"));
```

What this code does is assign a piece of text we want to print to some variable named `text`. Variable `text` will then call the method `fontcolor()` so that the string inside of `text` now has a specified font color (in this example it is red). The `document.write()` method simply outputs that newly colored statement onto the webpage. The output should now look like the following:

```
While loop: 1
While loop: 2
While loop: 3
While loop: 4
While loop: 5
While loop: 6
While loop: 7
While loop: 8
While loop: 9
While loop: 10
While loop: 11
While loop: 12
While loop: 13
While loop: 14
While loop: 15
```

Step 7
Let’s make our print statements a little more dynamic by having the font change colors depending on the given value.

Now, imagine we want our font to be a specific color based on whether the value is divisible by a
certain number. This kind of check can be achieved with the use of the **modulus operator**. You may remember the modulus operator back when we were doing Python, but in case you forgot it is represented as a percentage symbol “%”.

You can think of the modulus as essentially dividing two numbers. But instead of returning the result (known as the quotient), it will return the remainder. For example, 7%4 will return 3, 21%5 will return 1, and 16%8 will return 0.

Therefore in general, if we had X%Y return 0, then we can conclude that X is **divisible** by Y.

For this assignment, we want statements that contain values divisible by 2 to be printed in **red**, and statements that contain values divisible by 5 to be printed in **green**. Otherwise, if the value is neither divisible by 2 nor 5, then the statement should be printed in **blue**.

Use the template below to add an **if/else if** statement inside of the loop to perform this check. You will need to determine what goes in the spots designated as “???” as well as determining what goes in the body of the **if/else if** statements.

```javascript
if (numPrinted % ??? == ???) {
  // Print numbers divisible by 2 in red
} else if (numPrinted % ??? == ???) {
  // Print numbers divisible by 5 in green
} else {
  // Print numbers divisible by neither in blue
}
```

**NOTE:** You may notice in the code above that **else if** is spelled out in full. In Python, else if statements can be typed as “**elif**”, but in Javascript it **MUST** be typed out exactly as “else if” (including the space!).

Your output should now look like the following:

```
While loop: 1
While loop: 2
While loop: 3
While loop: 4
While loop: 5
While loop: 6
While loop: 7
While loop: 8
While loop: 9
While loop: 10
While loop: 11
While loop: 12
While loop: 13
While loop: 14
While loop: 15
```

Notice how in this case the number 10 is printed in red but not green. Think about why this is the case.
For Loop

The next type of loop we will be using is called a **For Loop**. In Python, we used a For Loop to iterate through a list and printed out all elements contained in that list. For Loops in JavaScript does not have that same functionality, but instead act more like While Loops. The difference is that the incrementing of the loop counter in a For Loop is done for you instead of having to add `numPrinted++` inside of the loop:

```
for (set counter ; conditional statement ; increment counter){
    what to do in the loop
}
```

Normally in a While Loop, we increment the counter in the body of the loop. In the above example, notice that the increment counter is typed within the set of parenthesis.

**Step 1**
So to emulate our while loop we would add the following:

```
for (numPrinted = 1; numPrinted <= numLoop; numPrinted++) {
    document.write(sentence);
}
```

What this loop is doing step by step in the order listed:

1. set numPrinted to 1
2. check to see if numPrinted is less than numLoop
3. if numPrinted <= numLoop then perform `document.write(sentence)`
4. increment numLoop and repeat starting at step 2.

**Step 2**
Save your file and run it in Firefox. What do you notice? The For Loop printed the sentence 15 times, but it started the counting at 1. Therefore it prints values will at “1”, and ends at “15”. This is because unlike our **While Loop**, which increments `numPrinted` before it prints the sentence, the **For Loop** increments `numPrinted` after the sentence is printed.

Now modify our sentence so that the correct number is printed. To do this we can simply replace `sentence` with the variable `text`:

```javascript
var text = "For loop: " + numPrinted + "</br>";
```
**Step 3**
For the second part, we want to do the same thing as in the While Loop, printing statements and their corresponding values in red. You can re-use the code that you had in your While Loop, but remember to change “While Loop:” to “For Loop:” and `numPrinted` to `(numPrinted + 1)`.

When you save and run your file you should now see the following output EXACTLY:

```
While loop: 1
While loop: 2
While loop: 3
While loop: 4
While loop: 5
While loop: 6
While loop: 7
While loop: 8
While loop: 9
While loop: 10
While loop: 11
While loop: 12
While loop: 13
While loop: 14
While loop: 15
For loop: 1
For loop: 2
For loop: 3
For loop: 4
For loop: 5
For loop: 6
For loop: 7
For loop: 8
For loop: 9
For loop: 10
For loop: 11
For loop: 12
For loop: 13
For loop: 14
For loop: 15
```

**Step 4**
For our last step, we want to add if/else if statements to determine the font color of our text based on their values much like we did in our While Loop. The same restrictions will apply: values divisible by 2 should have red text, values divisible by 5 should have green text, and all other values should have blue text.

If implemented correctly, your final output should look like the following:
NOTE: Common problems to pay attention to:
1. Verify that the numbers are counting from 1-15 and not 0-15, or 0-14.
2. Verify that there is only one of each number, (i.e. 10 doesn’t show up twice)
3. Verify that all numbers correspond to their correct color assignment

There must be no errors to receive credit for this assignment.

Putting it online

Step 1:
Put everything online and get checked off.

REMEMBER to either drag your ENTIRE CSE3 folder into your public_html folder OR into the WHITESPACE inside the public_html folder!!!!

Checkoff: Go to your homepage via the class webpage and demonstrate to the TA/Tutor that your loops.html is complete. Be prepared to present and explain your code!