Please Read:

In this assignment, we will be discussing two types of loops: a **for loop** and a **while loop**. We are going to be using loops to print a sentence a given number of times to demonstrate the differences between a JavaScript for loop and while loop.

Recall that for an **if/else statement**, there is a set of instructions that gets executed if the condition is true, and a set of instructions that gets executed if the condition is false. The same thing happens in JavaScript 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. Loops in JavaScript function in the same way that they have in Python.

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 it), whereas a **while** loop is used in all other cases (e.g. “while” it’s raining, do something; “while” you’re hungry, do something).

A **while** loop will execute the loop body as long as the while loop condition is met. Therefore, the flow chart for a **while** loop is:
A **For** loop is similar to a **While** loop, but in a more specific manner. **For** loops are generally used to repeat code a certain number of times, e.g. for counting. The three parts of a **For** loop condition statement will be discussed later in this assignment, but is illustrated in the following flow chart:

![For Loop Diagram](image)

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; the counter is declared and incremented as part of the conditional code and is updated after the body of the loop is executed. A **While** loop also uses a counter, but the counter is declared before the body of the loop is executed and updated after each loop.

**Assignment:**

**Step 1:**
In your CSE3 folder, create and save a new Notepad++ file called **Loops.html**.

**Step 2:**
Write out the basic HTML skeleton (if you don't remember what this looks like refer back to one of the earlier labs on HTML). In the head tags, set the **title** to “JavaScript Loops” using the title tags.

**Step 3:**
In the body tags, create `<center>` tags. Include a heading 1 at the top of your webpage that says “While and For Loops”. Inside the center tags, add the following JavaScript tags:

```html
<center>
  <h1>While and For Loops</h1>
  <script type="text/javascript">
  </script>
</center>
```

**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, let’s create our loop. The first kind of loop we are going to use is 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 conditional statement (inside the while parentheses) is false. While loops in JavaScript operates the same as While loops you have seen in Python. In our case, we want to make a loop that keeps executing the loop body until the conditional statement is false, meaning that numLoop == numPrinted is reached.

Step 1
First let’s outline the While loop:

```javascript
while (condition) {
  document.write("Sentence to print.");
}
```

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("Sentence to print.");
}
```

This loop is saying:

“While the number of times we have printed is less than the number of times we want to print it, keep printing it.”

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 printing that is `document.write()`.

```
numPrinted++;
```

This is our loop counter, and it will keep track of the number of times the loop has executed.
All together it should look like this:

```
while (numPrinted < numLoop) {
    numPrinted++;
    document.write("Sentence to print.");
}
```

**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, “Sentence to print.”
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 concatenating using the ‘+’ operator:

```
document.write("While loop #: " + numPrinted + ">\n");
```

**NOTE 1:** Only the words and the break tag should be inside the quotation marks (strings)

**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 webpage output should look like the following:
Step 6
Right now, our print statements seem a bit boring in black font. Let’s fix that by having them print in **SILVER**. 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("silver"));
```

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 `text` now has a specified font color (in this example it is green). The `document.write()` method simply outputs that newly colored statement onto the webpage. The output should now look like the following:
While and For Loops

Let’s make our print statements more dynamic by changing the font colors based on values.

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 from Python, but in case you forgot it is represented “%”.

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, if X%Y returns 0, then we can conclude that X is divisible by Y.

For this assignment, we want statements that contain values **divisible by 9** to be printed in **AQUA**, and statements that contain values **divisible by 3** to be printed in **PURPLE**. Otherwise, if the value is **neither divisible by 9 nor 3**, then the statement will be printed in **SILVER**.

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.
NOTE: The order you check if a number is divisible by 9 or by two matters… Are numbers that are divisible by 3 also divisible by 9? What about the reverse of that statement? This is a popular problem often referred to as “Fizz Buzz”. You can read more about it here: https://en.wikipedia.org/wiki/Fizz_buzz

NOTE: 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:
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 `numPrinted++` inside the loop:

### Code Snippet

```javascript
for ( (1) declare counter; (2) conditional statement; (3) increment/decrement counter) {
    // Loop body
}
```

Note the three main parts. Normally in a **While Loop**, we increment the counter in the body of the loop. In the above example, notice that the counter is declared and incremented within the set of parentheses.

**Step 1**

To emulate our **While loop**, we would add the following after the **While loop**:

```javascript
for ( numPrinted = 30; numPrinted > 0; numPrinted-- ) {
    document.write("Sentence to print. <br />");
}
```
What this For loop is doing step by step in the order listed:
1. Set numPrinted to 30
2. Check to see if numPrinted is greater than 0
3. If (numPrinted > 0) is true, execute “document.write(sentence)” 
4. Decrement numLoop and repeat starting at step 2.

Step 2
Now modify our sentence so that the correct number is printed. To do this we can simply replace the sentence with the variable text:

```javascript
for (numPrinted = 30; numPrinted > 0; numPrinted--) {
    var text = "For loop #: " + numPrinted + "<br />";
    document.write(text);
}
```

Save your file and launch it on a webpage. What do you notice? The For Loop printed the sentence 30 times, starting the count at 30. Therefore, it prints between “30” and “1” inclusively.

Your output should now look like the following: (After “While loop #: 30”)

```
For loop #: 30
For loop #: 29
For loop #: 28
For loop #: 27
For loop #: 26
For loop #: 25
For loop #: 24
For loop #: 23
For loop #: 22
For loop #: 21
For loop #: 20
For loop #: 19
For loop #: 18
For loop #: 17
For loop #: 16
For loop #: 15
For loop #: 14
For loop #: 13
For loop #: 12
For loop #: 11
For loop #: 10
For loop #: 9
For loop #: 8
For loop #: 7
For loop #: 6
For loop #: 5
For loop #: 4
For loop #: 3
For loop #: 2
For loop #: 1
```
Step 3
For the next part, we want to do the same thing as in the While Loop, printing statements and their values in corresponding colors. 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 9 should have AQUA text, values divisible by 3 should have PURPLE text, and all other values should be SILVER. You can re-use the same code structure that you had for While loop.

If implemented correctly, your FINAL output should look exactly like the following:

**While and For Loops**

```plaintext
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
While loop #: 16
While loop #: 17
While loop #: 18
While loop #: 19
While loop #: 20
While loop #: 21
While loop #: 22
While loop #: 23
While loop #: 24
While loop #: 25
While loop #: 26
While loop #: 27
While loop #: 28
While loop #: 29
While loop #: 30
For loop #: 30
For loop #: 29
For loop #: 28
For loop #: 27
For loop #: 26
For loop #: 25
For loop #: 24
For loop #: 23
For loop #: 22
For loop #: 21
For loop #: 20
For loop #: 19
For loop #: 18
For loop #: 17
For loop #: 16
For loop #: 15
For loop #: 14
For loop #: 13
For loop #: 12
For loop #: 11
For loop #: 10
For loop #: 9
For loop #: 8
For loop #: 7
For loop #: 6
For loop #: 5
For loop #: 4
For loop #: 3
For loop #: 2
For loop #: 1
```
NOTE: Common problems to pay attention to:
   1. Verify that the numbers are counting from 1-30, inclusive for the **While loop**.
   2. Verify that the numbers are counting from 30-1, inclusive for the **For loop**.
   3. Verify that there is only one of each number, (i.e. 15 doesn’t show up twice).
   4. Verify that all numbers correspond to their correct color assignment.

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

**Putting it online**

Put everything online to get it checked off.

**REMEMBER** to drag your ENTIRE CSE3 folder into the WHITESPACE of your **public_html** folder!

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