IT 105 – Principles of Programming

Day 15

1. Just a reminder, when you save your pages make sure they are text, but that they have the `.html` extension and not `.txt`. As you modify your pages to add functionality, be sure to save separate copies.
2. Let's use Javascript to change the background color of the web-page. The page will just have one button, which when pushed, will change the background color of the page. Once again, we can use `querySelector` to modify a style element, in this case `backgroundColor`. Following our normal design procedure, we render the form first, making sure we have the correct URL and the button is rendered. Once that is set, we need to add:
    a. The event in the `body` element.
    b. The Javascript in the `script` element.

    This give us (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js11.html):

```
<!DOCTYPE html>
<html lang=”en”>
   <head>
      <title>Hello!</title>
      <script>
        function makeRed()
        {
document.querySelector('body').style.backgroundColo
r="red";
        }
      </script>
   </head>
   <body>
      <form name=inclass>
        <input type=button value="Red"
                          onclick="makeRed();">
      </form>
   </body>
</html>
```

3. Here, once the button is clicked the background stays red. How could we make the background color toggle between red and white as the button is pressed? We could make use of a global variable, and then test the global variable's value. Depending on the value, either a red or white background is set. We also would need to change the toggle value as well.
4. Note that we make use of a Boolean value for our toggle variable. In Javascript (and many other languages), a Boolean variable is either ***true*** or ***false***. So our condition is just the value of the Boolean, there is no need to test. Here's what the script element looks like now:

```
<script>
  var bgToggle=true;
  function makeRed()
  {
     if (bgToggle) {

document.querySelector('body').style.backgroundColo
r="red";
       bgToggle=false;
     } else {

document.querySelector('body').style.backgroundColo
r="white";
       bgToggle=true;
     }
  }
</script>
```

5. The last control structure we will consider in Javascript is iteration, or looping. Let's create a page with one button, which when pushed will display a sequence of alerts, counting up from 0 to 4. Following our normal design procedure, we render the form first, making sure we have the correct URL and the button is rendered. Once that is set, we need to add:
   a. The event in the `body` element.
   b. The Javascript in the `script` element.

   This give us (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js12.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Hello!</title>
      <script>
        function doCount()
        {
```

```
            var i;
            for (i=0; i<5; i++) {
               alert("Your number is "+i);
            }
         }
      </script>
   </head>
   <body>
      <form name=inclass>
        <input type=button value="Counting"
onclick="doCount();">
      </form>
   </body>
</html>
```

6. `for` is the Javascript keyword to iterate a specified number of times. It has three parameters inside the parentheses `( )`, and all the lines executed in the loop are nested inside the curly-braces `{ }`.
7. The control parameters for the for loop above, `for (i=0; i<5; i++)` are:
   a. The variable `i` is the ***loop control*** variable. It is initialized to 0 with `i=0;`
   b. The ***stopping criteria*** for `i` is set with `i<5;`
   c. `i` is incremented, made one larger, each iteration of the loop, with `i++`
8. So, over the course of the execution of this loop, the variable `i` has the values 0, 1, 2, 3, and 4.
9. To decrement, or count-down, the loop control variable could be initialized with a large value, the ***stopping criteria*** could be greater than a smaller value, usually 0, and the loop control variable would be decremented: `for (i=5; i>0; i--)`.

10. Finally, let's create a webpage which prompts the user for an input value, called ***N***, and then displays the sum 1 + 2 + 3 + … + N. Before attempting the Javascript, we follow our normal design procedure, rendering the form first, making sure we have the correct URL. Once that is set, we have: (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js13.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Hello!</title>
      <script>
        function doSum()
        {
           var sum=0;
           var N = parseInt(
document.querySelector("#N").value );
           for (i=1; i<=N; i++) {
             sum += i;
           }

document.querySelector("#sumHere").innerHTML =
"Your sum is: "+sum;
        }
      </script>
   </head>
   <body>
      <form name=inclass>
        Enter an integer to sum to: <input
type=text id="N"><br>
        <input type=button value="Start the Sum"
onclick="doSum();">
        <div id="sumHere">
          Your sum is:
        </div>
      </form>
   </body>
</html>
```

11. There is a lot going on here…
    a. We need a variable to keep track of the running total. We call ours ***sum*** and set it equal to 0 initially. Each time through the loop, as ***i*** increments, the value of ***i*** is added to ***sum***, using the ***sum += i;***. We could have also used ***sum = sum+i;***, the ***+=*** is a shortcut.
    b. We named our text field ***N***. To use this value in our Javascript, we need to use `querySelector` to access it, so we can assign it to a Javascript variable ***N***. But since the default type for HTML text field is text, and not numeric, we convert it to an integer using the Javascript function ***parseInt()*** .
    c. And finally, we use `div` tags and `innerHTML` to update the page to display the sum.