IT 105 – Principles of Programming

Day 14

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. Today we will continue using Javascript to make our pages dynamic, changing based on interaction with the user.
3. We would like to keep track of the number of times a button is clicked.
4. First, we simply render the page without any Javascript, ensuring our `form` element is defined correctly: (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js08.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
   </head>
   <body>
      <form name=inclass>
        <input type=button
               name=counter
               value="Click Counter">
      </form>
   </body>
</html>
```

5. We would like to have a way to keep track of how many times the button is clicked. To accomplish this we will add several elements:
   a. An `onclick` event which calls a Javascript function.
   b. The Javascript function will use a variable which will track the number of clicks.
   c. The variable will be defined ***outside*** the Javascript function, which means it is a ***global*** variable. Note that variables defined inside the { } of a function are called ***local*** variables. ***Global*** variables can be used in any function in the `<script>` element. There values are "remembered" each time the function is called, while ***local*** variables are reset each time the function is called.
   d. The variable needs to have an intial value, which here would be 0.
   e. To increment the variable, we need to add 1, and there are a few ways to do this:

```
 i. numClicks = numClicks + 1;
ii. numClicks++;
```

f. In this version of the code, we will output the value to an `alert()` pop-up.

6. Here is the code:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
      <script>
        var numClicks;
        numClicks = 0;
        function countClicks()
        {
          numClicks++;
          alert("You have clicked "+numClicks+"
                 times");
        }
      </script>
   </head>
   <body>
      <form name=inclass>
        <input type=button name=counter
value="Click Counter" onclick="countClicks();">
      </form>
   </body>
</html>
```

7. Finally, how would we output the count to the webpage, rather than in an `alert()`? By using `querySelector()` and `innerHTML`.

8. Here is the updated code: (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js09.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
      <script>
        var numClicks;
        numClicks = 0;
        function countClicks()
        {
          numClicks++;

document.querySelector("#countOut").innerHTML="You
have clicked "+numClicks+" times";
```

```
        }
      </script>
   </head>
   <body>
      <form name=inclass>
        <input type=button name=counter
value="Click Counter" onclick="countClicks();">
      </form>
      <div id="countOut">
      The count goes here.
      </div>
   </body>
</html>
```

9. We have written, and been forced to debug, several Javascript files. It would be helpful to review some common errors.
10. ***Syntax*** errors are "grammatical" errors, where the rules of the language are not followed. Here are some examples:
    a. missing " , {, }, < , or >
    b. mis-spelled HTML or JavaScript keywords
    c. spaces in `id`'s or `names`
    d. case disagreement in `id`'s or `names`
    e. using keywords for `id`'s or `names`
11. ***Semantic*** errors are errors of "intent," where the code is syntactically correct, but the code does not do what it is supposed to.
12. One of the basic control structures we discussed and used previously in Scratch, is ***selection***. In Javascript, as in Scratch, we use the `if` statement for selection.
13. Let's prompt the user to enter a course name. We will render the form first, and then add the Javascript: (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js10.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
   </head>
   <body>
      <form name=inclass>
        Enter your favorite course:
        <input type=text id=course>
      </form>
   </body>
</html>
```

14. Now, when the input field is changed, we will check whether the user has typed in "it105." If they have, we will print an `alert()`. The conditional test for equality is two == signs. Remember than an assignment of a value is only one = sign:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
      <script>
        function favCourse()
        {
if (document.querySelector("#course").value =="it105")
            {
              alert("Hey, great choice!");
            }
        }
      </script>
   </head>
   <body>
      <form name=inclass>
        Enter your favorite course: <input
type=text id=course onchange="favCourse();">
      </form>
   </body>
</html>
```

15. We can add an else block to the if statement, to define code that is only run if the conditional test is false:

```
if (document.querySelector("#course").value =="it105")
{
  alert("Hey, great choice!");
} else {
  alert("Maybe choose a different course?");
}
```

16. Finally, if the user types “IT105” in all caps, the if statement doesn’t recognize it as a “great choice.” So we can make a compound selection statement, testing for both “it105” and “IT105”:

```
if (document.querySelector("#course").value =="it105" ||
    document. querySelector("#course").value =="IT105"){
  alert("Hey, great choice!");
} else {
  alert(“Maybe choose a different course?”);
}
```

17. The logical connective operators are:
    a. `||` for OR (used in the example above, course is either “it105” or “IT105”) either condition can be true for the entire statement to be true.
    b. `&&` for AND both conditions must be true for the entire statement to be true.
    c. `!` for NOT, which just reverses the truth value of the statement.