IT 105 – Principles of Programming

Day 13

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. Let's pick up where we finished last class, where we were concatenating the values for first and last names and updating a text field. But here, rather than naming our text fields, we will give them an `id`, and then use `querySelector` to access the value.
4. `querySelector` can access elements with a defined `id`, or it can access entire html elements, like `body`.
5. The code below is modified in a few ways:
   a. The `first` and `last` text fields are not `name`-ed, but have `id`'s.
   b. When we use `querySelector` to access the values, since they are `id`'s we need the `#`.
6. Here is the code: (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js05.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
      <script>
         function glue()
         {
           document.querySelector("#full").value =
             document.querySelector("#first").value
           + " "
           + document.querySelector("#last").value
         }
      </script>
   </head>
   <body>
      <form name=inclass>
        First: <input type=text id="first"><br>
        Last:  <input type=text id="last"
                          onchange="glue();" ><br>
        Full:  <input type=text id="full">
      </form>
```

```
    </body>
</html>
```

7. We need to make one last little improvement. Let's add a `reset` button to clear any entries (note that it needs to be nested ***inside*** the form element with the entries it will clear):
```
<input type=reset value="clear entries">
```
8. `querySelector` will also allow us to change the source of the html webpages we define. Here, to indicate the area we would like to modify, we will use the `<div>` tag. We'll start with our previous code, adding a `<div>` tag at the end, using the `id` attribute in the `<div>` tag, and then with some plain text nested inside the `<div>` tag: (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js06.html):
```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
      <script>
         function glue()
         {
           document.querySelector("#full").value =
             document.querySelector("#first").value
           + " "
           + document.querySelector("#last").value
         }
      </script>
   </head>
   <body>
      <form name=inclass>
        First: <input type=text id="first"><br>
        Last:  <input type=text id="last"
                      onchange="glue();" ><br>
        Full:  <input type=text id="full">
        <input type=reset value="clear entries">
      </form>
      <div id=changeText>
      Here is some text.
      </div>
   </body>
</html>
```
9. Now, will add a function that runs when `first` is changed. That function will use `querySelector` to access and change the `<div>` tag with the id

changeText. This will also use innerHTML rather than value, since we are changing the HTML.

10. Here is the updated code (be careful that your new Javascript function hey() is outside the previous glue() function, and note for the command to fit on one line, the indentation is not used:

```
<!DOCTYPE html>
<html lang=”en”>
   <head>
      <title>Javascript</title>
      <script>
         function glue()
         {
           document.querySelector(“#full”).value =
             document.querySelector(“#first”).value
           + “ “
           + document.querySelector(“#last”).value
         }
         function hey()
         {
document.querySelector(“#changeText”).innerHTML =
             "Hi " +
document.querySelector("#first").value;
         }
      </script>
   </head>
   <body>
      <form name=inclass>
        First: <input type=text id="first"
                        onchange="hey();" ><br>
        Last:  <input type=text id="last"
                        onchange="glue();" ><br>
        Full:  <input type=text id="full">
        <input type=reset value="clear entries">
      </form>
      <div id=changeText>
      Here is some text.
      </div>
   </body>
</html>
```

11. Finally, we will address how Javascript treats numeric values.

12. We'll start with defining a page with three text fields and a button, where we will add the entries in the two text fields and display the result in the third text field. As usual, we will confirm the form elements render properly before we add the Javascript: (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js07.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
   </head>
   <body>
      <form name=inclass>
        <input type=text id="first">
        <input type=button value="Addtion, +">
        <input type=text id="second">
            =      
        <input type=text id="full"><br>
      </form>
   </body>
</html>
```

13. Note the ` ` , which are used to add spaces on the rendered webpage.
14. Here, we add an `onclick` event on the button, and the Javascript to do the addition:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
      <script>
        function add()
        {
          document.querySelector("#full").value
         = document.querySelector("#first").value
         + document.querySelector("#second").value;
        }
      </script>
   </head>
   <body>
      <form name=inclass>
        <input type=text id="first">
        <input type=button value="Addtion, +"
                           onclick="add();">
```

```
        <input type=text id="second">    
                      =      
        <input type=text id="full"><br>
      </form>
    </body>
</html>
```

15. The problem with the code above, is that it still does a concatenate. The two values in the text fields are "glued" together. We have to tell Javascript to treat the values typed into the text fields as numeric. Javascript has two functions for this:
    a. `parseInt()` – if we want the value to be a whole number.
    b. `parseFloat()` – if we want the value to be numeric, but doesn't need to be a whole number.
16. Here, we'll use `parseFloat()`, with the Javascript now:

```
function add()
{
document.querySelector("#full").value =
  parseFloat(document.querySelector("#first").value)
+ parseFloat(document.querySelector("#second").value);
}
```