IT 105 – Principles of Programming

Day 12

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. There are many types of interaction we can react to. We call these ***events***. A few of these are listed below (note that last class we used `onchange` to react to a button being clicked:
    a. click
    b. change
    c. load
    d. mouseover
    e. submit
    f. select
4. In an series of HTML checkbox'es, there is no limit to the number which can be checked. Also note that the text associated with the checkbox is outside the `input` tag.
5. Let's start with a simple page and add a series of checkbox'es. Note that we name our `form` element and the checkbox'es we wish to react to. Also, it is important to render the page before adding in the Javascript, to confirm that the URL is correct and the `form` element has the proper syntax:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
   </head>
   <body>
      <form name=inclass>
        <input type=checkbox>Animal<br>
        <input type=checkbox>Vegetable<br>
        <input type=checkbox>Mineral<br>
        <input type=checkbox>Fire<br>
      </form>
   </body>
</html>
```

6. Next, we will update our HTML code with some Javascript. In particular, we want to react when one of the checkbox'es is clicked. Also, to promote the coding style we will use in class, we want the event to trigger a call to a Javascript function in the `script` element (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js03.html):

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
      <script>
         function choice()
         {
           alert("good choice");
         }
      </script>
   </head>
   <body>
      <form name=inclass>
        <input type=checkbox>Animal<br>
        <input type=checkbox>Vegetable<br>
        <input type=checkbox>Mineral<br>
        <input type=checkbox
               onclick="choice();">Fire<br>
      </form>
   </body>
</html>
```

7. Note that the `alert()` message is also executed when the box is unclicked.
8. We can use the Document Object Model (DOM) to access values on our pages. There are two different ways to do this:
   a. `document.querySelector("#`***`id-you-use`***`").value`
   b. `document.`***`form-name-you-use.attribute-name-you-use`***`.value`

9. First, let's create the `form` elements on a page, and confirm the URL and the syntax is correct. In particular, we will create three text fields, labeled ***first***, ***last***, and ***full***. Also, it's good programming practice to prompt for any requested input. The user doesn't usually know the thought process of the programmer.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Javascript</title>
   </head>
   <body>
      <form name=inclass>
        First: <input type=text name="first"><br>
        Last:  <input type=text name="last"><br>
        Full:  <input type=text name="full">
      </form>
   </body>
</html>
```

10. We would like to concatenate the first and last names, and output this full name in the ***full*** text field. The operator for concatenate is the + sign. It seems appropriate to ***overload*** the + sign. To overload an operator or a keyword in a programming language means to have more than one use for the operator. The programming language, Javascript here, figures out the appropriate one to use. Since all the values are text and not forced to be numeric, Javascript knows to concatenate them. The line returns below breaking up the one assignment statement are added since it doesn't fit on one line. This might aid in readability. Note that the operation always happens on the right side. Also, the form name `inclass` and the field names `full`, `first`, and `last` are used below:

```
document.inclass.full.value =
                        document.inclass.first.value
                      + document.inclass.last.value;
```

11. Previously, we have used the event `onclick`. We could easily do the same here, including a button to initiate the concatenation, but here we will introduce the `onchange` event. Since we assume that the user will enter the first name and then the last name, we put the event in the last name field (http://jcsites.juniata.edu/faculty/kruse/it105/inClass/js04.html):

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

12. We need to make one last little improvement. The names get concatenated, but there is not space between them. We can update our function to include a space:

```
document.inclass.full.value =
                     document.inclass.first.value
                   + “ “
                   + document.inclass.last.value;
```