Lab 7

Experience programming JavaScript statements

FIT100 Winter 2006

Goal: The purpose of this assignment is get experience writing JavaScript applications. Key ideas are:

- Declaring variables
- Initializing variables
- Computing values
- Creating Web Page Components using `Document.write`.
- Learning to debug Javascript in the Javascript console

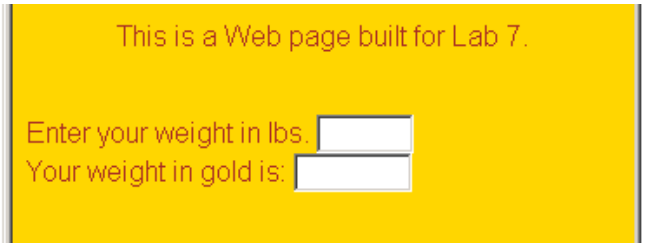
Resources: The lab extends the forms concepts of Lab 6, and the JavaScript programming ideas of Lecture 11. Also, the material is covered in Chapter 19 of the textbook.

Getting Started: Create a HTML Web page with the following look. Save the file as `lab7.html` on the desktop. When you're finished, it'll go in a `lab7` folder under your `fit100` directory. (The background is gold, and the text is brown.)


```
<html>
  <head><title>Worth Your Weight</title></head>
  <body bgcolor=gold text=brown> <font face='helvetica'>
  <h1 align=center>Gold </h1>
  <p align=center> This is a Web page built for Lab 7.</p>
  </font>
  </body>
</html>
```

Computing Our Weight in Gold: Gold was selling for $566.99 per troy ounce on January 30. We want to compute how much each of us would be worth if we were "worth our weight in gold," as the old saying goes.

A. Using your knowledge of forms, create a request to the user to enter his or her weight. The request should be followed by a text box with the name `weight`, and size of 5. On the next line should be the text reporting the result, followed by another textbox for the output. Its name should be `out` and its size should be 7. Your page should look as follows:

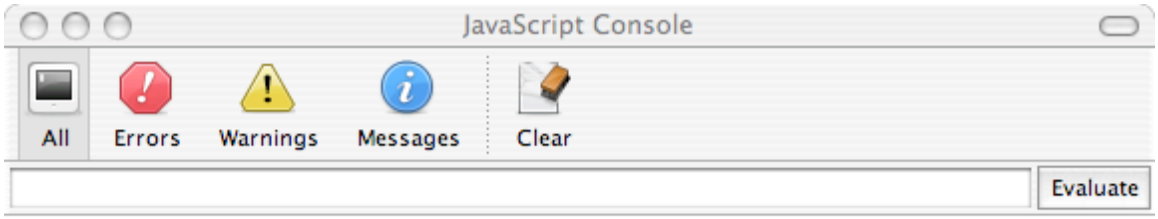


B. Next define the event handler for the `weight` computation. Define the `onChange` event handler to compute the user's weight in gold. There are 12 troy ounces in each pound and the price of gold is $566.99 per troy ounce. (Just use the number, don't include a $ symbol!) Multiply the three values together and assign it as the value for the output window, that is, `out.value`.
C. Check your result with a value of 1 lbs, and verify that your display shows 12 times the price of an ounce of gold, $6803.88

Debugging your code: Hopefully, you haven't made any errors in your programming thus far, but chances are that you have. One helpful tool built into the Mozilla and Mozilla Firefox browsers is the Javascript console, which allows you to test out Javascript and displays any Javascript errors that have occurred on your webpage. Let's try out the latter.

Only Mozila and Mozilla Firefox browsers have the Javascript console, so make sure you're not trying to do this in Internet Explorer. If you are, close the window, right-click on the lab7.html icon on your desktop, select "Open with…" and open it with Mozilla Firefox or Mozilla.

The Javascript console is located in Tools→Javascript Console for Firefox, and Tools→Web Development→Javascript Console for Mozilla. It should be blank, as the image below shows. If it is not, it either means you have an error on your page, there was an error on your page but you fixed it (the Javascript console doesn't automatically erase old error messages) or that these are errors from another page that you opened in this browser (yes, even professionally designed webpages have Javascript errors sometimes). In any case, you can hit "clear" to clear it for now.



Now, let's introduce an error into your code intentionally to see what happens.

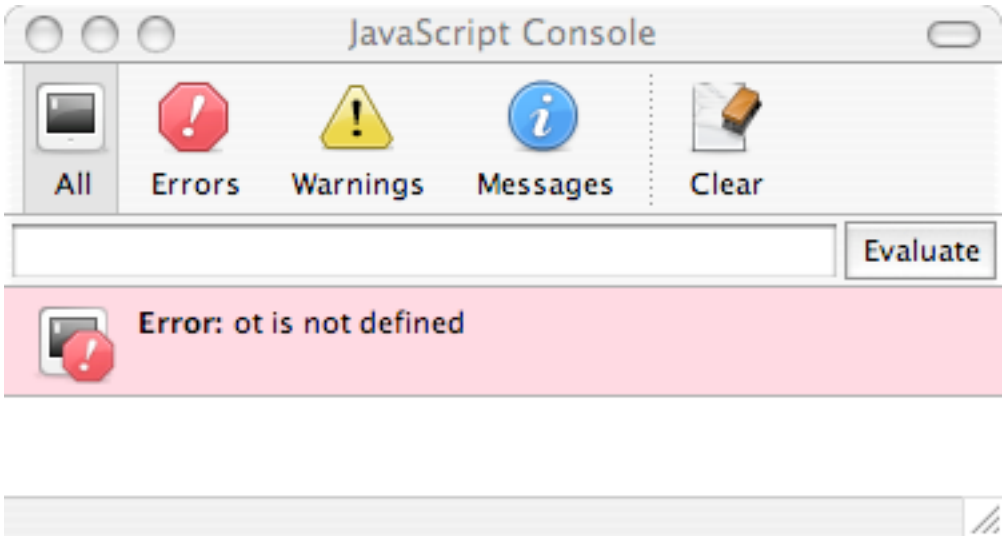
Let's say your error is that you misspelled "out" as "ot" in `out.value` which is your onChange event handler. Make the change to `ot.value` now. Do not change the spelling of "out" anywhere else, only change that one instance of it.

Save your page.

Reload your page in your browser. Check the Javascript console, it should still be empty.

Now, type in your weight and see if your form gives you your weight in gold. It shouldn't, because you've intentionally placed an error in your Javascript.

Now look at the Javascript console. It should have one error saying "ot is not defined". This basically means that you are asking the browser to do something with "ot" but the browser can't find anything named "ot." In our case, that's because there IS nothing named "ot", the text box we're looking for is named "out."



Often, there will be a line number as well. Check the line number and find the matching line in your code, look for the error, and fix it. Here, the problem was obviously a misspelling. In the future, there may be other reasons for things being "not defined" but you'll discover that for yourself as you practice more Javascript.

Be sure to fix your error before you move on.

Display Five Pictures: Grab `the liberty.jpg` image from the class Web page under "Files and Stuff" and place it on your desktop. We want to display this image five times in a row on the page, which uses the usual tag:

```
<img src=liberty.jpg>
```

Copy and paste this tag into the file five times so the images line up in a row.

Display Five More: We will now write JavaScript commands to insert a tag into the HTML file. This is a common operation used to build Web pages on the fly. (We won't be doing that quite yet, but we want the knowledge now.)

To write JavaScript, insert the tags after the images in your Web page:

```
<br>
<script language=JavaScript>

</script>
```

Inside of these `script` tags, the browser will understand the commands as JavaScript; it will NOT understand HTML inside the tags. Between the opening and ending `script` tags, write the line

```
document.write("<img src=liberty.jpg>");
```

This command tells JavaScript to write the `<img ...>` tag into the HTML file at the position where the `<script>` tag is placed. That will cause a sixth copy of the image to be placed on the page. Replicate the `document.write` line 4 more times so it appears a total of five times inside the `<script>` tags.

The use of `document.write` had the same affect as our explicit use of the `<img ...>` tags, because after the document write commands have been performed by the JavaScript interpreter, the result is the same: five more `<img ...>` source lines.

This works because when the browser processes the HTML, it scans the file looking for JavaScript. It does the JavaScript first, which for us is to place five more copies of the image, and then it displays the resulting HTML. See pp. 578-579 in the textbook for a fuller explanation.

A Tweek: The first set of images had a break between each image, but the second group produced by `document.write` did not. This is because our command

```
document.write("<img src=liberty.jpg>");
```

had no spaces after the `<img ...>` tag. If we insert a space after it,

```
document.write("<img src=liberty.jpg> ");
```

they will look the same. (Actually, the original `<img ...>` tags were on separate lines, and it wasn't actually a space but the return that created the white space that separated the pictures.) Make the correction and see that it has the proper effect. And, at this point, add a
 to the end of the final `document.write`. Make sure you put that
 in the right place so that it actually gets interpreted through the Javascript `document.write`!

Get Pictures. In the next part we will use four `.gif` images designed to spin around in an animation. (We are not quite ready to animate, but we will at least do one spin.) Grab the four files: `New0.gif`, `New1.gif`, `New2.gif` and `New3.gif` from the class Web page. Put them in a folder `new` on the desktop.

Composing File Names: One great advantage of using JavaScript to write text into our HTML files is that we can "compute" file names. We illustrate this simply today, but it will be the basis for animation and other interesting Web pages later.

Declare a variable `i` right after the opening `<script>` tag and before the five `document.write` statements:

```
var i;
```

After the five `document.write`s, type in

```
i = 0; document.write("<img src=new/New" + i + ".gif>");
```

The first of these instructions sets the variable `i` to 0, i.e. it gives i the value zero. The second is like the earlier `document.write` statements, except that this one *creates* the file name using concatenation (+) and the variable `i`. So, because using a variable is like using its value explicitly, we create the name `new/New0.gif`, which is the first of the four files we loaded. Try it out and see that you get the following:

Other Orientations: Now, replicate the line three more times, changing the value assigned to i each time to one number larger:

```
i = 1; document.write("<img src=new/New" + i + ".gif>");
i = 2; document.write("<img src=new/New" + i + ".gif>");
i = 3; document.write("<img src=new/New" + i + ".gif>");
```

and notice that the result is

What happened? Each time the value of `i` is changed, the "composed" file name is different because it uses the new value. So, the same `document.write` command can create different file names using composition.

Because we are just learning JavaScript, the example isn't very compelling. We could have just typed the different names. But, soon, we will do the incrementing of the `i` value automatically, and then the power of composing file names with `document.write` will pay off.

Grading: You need to come to your lab section to have your lab graded. When you are ready, tell your TA and show them your lab. (If you do not finish in lab in section, tell your TA you are not done, and arrange for it to be graded before the beginning of the next lab session.)