CSC 105 Lab 8: CSS

Lab 8 Topics

1. FileZilla
2. HTML
    - Forms
    - Inputs
3. JS
    - Alerts
    - Functions
    - Printing to Console
    - Variables
    - Accessing HTML
    - Writing to HTML

Goal

The goal of today's lab exercise is to introduce you to JavaScript. JavaScript (JS) is the first of the three languages we have learned in this class that is most like *normal* programming languages you learn about in Computer Science. JS allows you to do read inputs, do computations, send instructions to other languages, ouput content, and even build entire web applications -- which is different than just a web page. HTML and CSS are special languages built for a particular purpose, however, JS is much like Java, C, Python, etc. in that it has constructs such as: variables, conditionals, loops, and functions. Today's lab will only touch on variables and functions, and in next week's lab we will learn about conditionals and loops.

Deliverable

By the end of the lab show the instructor a completed webpage, the result should be similar to last week, only now the Javascript you have written will allow information to be transfered from inputs to the table we have.

Exercise

The following section describes steps to guide you through the exercise to integrate JS into your webpage, allowing for a simple movement of text from inputs, to a different place in your HTML page. Next week will use JS for something more powerful, but just as an introduction we will begin simple. Before we begin, however, we are going to show you how to utilize your FREE UVic domain hosting.

1) Make Your Website Live!

The lab machines have "Secure File Transer Client" installed, but if you are using your own machine you'll have to download FileZilla. The set of steps to follow are slightly different for either program, but they both have the same general flow: connect to the UVic server, and transfer your files.

Secure File Transer Client (Lab Machines)

1. Search Computer for: `Secure File Transfer Client`, open program
2. Click: `Quick Connect`
3. Enter the following information:
    - Host Name: `unix.uvic.ca`
    - User Name: Your UVic Netlink ID (without `@uvic.ca`)
    - Port Number: `22`
4. Click: `Connect`
5. "Host Identification" will appear, click: `Yes`
6. You will be prompted for your password, type in your UVic Netlink ID password
7. "Enter Authentication Response" will appear, click `OK`
8. You are in!

FileZilla (Personal Machines)

1. Download FileZilla (Select the proper version for your operating system), and install
2. Open FileZilla
3. Enter the following information:
    - Host: `unix.uvic.ca`
    - Username: Your UVic Netlink ID (without `@uvic.ca`)
    - Password: Your UVic Netlink ID password
    - Port Number: `22`
4. Click: `Quickconnect`
5. You are in!

Transfer Files (Lab Machines & Personal Machines)

Now that we have a connection to the UVic server, we want to **navigate to the `www` folder** on the server, which is what UVic serves up when you navigate to `web.uvic.ca/~<Your Netlink ID>`.

The left-hand side of the program is your local machine, the right-hand side is the server. You can drag files from one side to the other, to transfer files from your computer to the server, and vice versa.

Transfer Existing Website Files to the UVic Server

If you transfer the files you have from last week's labs to the UVic Server, you can see your website live! Transfer your Lab7 folder into the `www` folder on the server, go to your web browser, and go to the URL: `web.uvic.ca/~<Your Netlink ID>`. For example, if your UVic Netlink ID is `sassycat@uvic.ca`, then your web hosting can be reached by navigating to `web.uvic.ca/~sassycat`.

2) Add Additional HTML

Here is the HTML that we need for today's lab. Please **write the following code into your HTML file manually**. Copy-pasting probably won't work.

Markup

```
<form name="clientForm">
    <input name="id" class="form-control pull-left" placeholder="Client ID">
    <input class="form-control pull-left" placeholder="Client Name">
    <button onclick="" type="button" class="btn btn-primary"> Add Client </button>
</form>
```

Side note: the inputs we just put on the screen are really wide, use CSS to make them a set width (100px?).

3) JS and HTML Connection

When writing a JS file, we need to tell HTML where to find the JS file. This is done in a similar manner to the CSS connection. **In your HTML file**, write the following code into your <head> tag.

Markup

```
<head>
    <script type="text/javascript" src="./script.js" defer></script>
</head>
```

4) Writing Our First JS

You can actually execute JS inside HTML, and this is required to make our first action happen when you click the `Add Client` button. Inside the <button> tag, notice the attribute `onclick=""`? If we put

`alert('Hello')` inside there, we should be able to see an action happen when we click the button.

Markup
```
<button onclick="alert('Hello')" class="btn btn-primary"> Add Client </button>
```

Did you see what popped up? `alert()` is a function that makes a popup happen in your browser, with the text inside the alert displaying in the popup.

5) Actually Writing JS

Now, let's write our first JS, a function. We would like to execute custom JS when someone clicks that button. **Open up your script.js** file and write the following:

JavaScript
```
function addClient() {
    alert('Hello');
}
```

Now, open up your HTML file and change the `onclick` to call this new function we wrote instead.

Markup
```
<button onclick="addClient()" class="btn btn-primary"> Add Client </button>
```

When you click the button, do you see the alert pop up? Try changing the text inside the alert in your JS file, does that new text display when you click the button? The function we just wrote in JS is being called (activated) by the button press. Now let's do more than just call an `alert()` in this new function we wrote.

6) Printing to the Console

An important aspect of developing in a language such as JS is to know what information is being used inside your function. Before teaching you variables, let's learn how to pring values in our code. Write the following code so that your function looks as follows:

JavaScript
```
function addClient() {
    // alert('Hello');
    console.log("Hello World!");
}
```

Notice two things:

- the `alert('Hello')` we had before is now greyed out. This is called "commenting out" code. By

adding two forward slashes in front of any code, it will not be executed; the web browser will not see or run code that is commented out.

- `console.log()` is a function that will print anything you put in the parentheses. For instance, we have coded `console.log("Hello World!")` which will print "Hello World" to the console. But where is the console? We have to go find it in the browser:
    - Right-click in the browser: `Inspect`
    - A window in your browser or outside your browser should appear with a lot of buttons you can click. We want to select: `Console`
    - Reload your browser window, and click the `Add Client` button
    - Check the console, do you see "Hello World"? You should

7) Variables

Now that we can print values in our code, let's print something other than a string, let's print a variable. What is a variable? Good question. A variable in Computer Science has the same concept as the variables you know from math: variables are letters (or words) that store values inside them. E.g. X = 5, so X + 2 = ? Hopefully you guessed 7. Let's see how variables work in JS:

```javascript
var x = 5;
```

Pretty simple right? All we have to do is put `var` before we create a variable. What happens if you print `x` ?

```javascript
var x = 5;
console.log(x);
```

Reload the browser and click the `Add Client` button and see what prints in the console. Hopefully 5! Now try adding 2 to x in the print statement:

```javascript
var x = 5;
console.log(x + 2);
```

Does it print 7? If not, go back and read the instructions again, it should.

Variables can contain Numbers, Strings, and Lists (as well as other things we won't go over in this course). Numbers and Strings should be pretty obvious, and we won't go over Lists until next week. Here is an example of Numbers and Strings:

```JavaScript
var x = 5;  // Number
var y = "Hello" // String

console.log(x);
console.log(y);
```

Lastly, it is important to know that variables can be more than just letters, they can -- and should -- be words, particularly words that describe what the represent. For example:

```JavaScript
var age = 24
var eyeColour = "Green"
var bio = "My name is Lloyd Montgomery"
```

Now let's try something more complicated: Create a variable called `y` that stores `4`, add `x` and `y` together and store the result in `z`, then print `z`.

8) Accessing HTML

HTML has the capability to take user input through <form> tags with <input> tags; however, this is only useful if there is Javascript to take that information, and do something with it. To start, let's see if we can take the information out of the two <input> tags above.

The entire HTML page can be thought of as a giant object, with many variables stored through it. Begin the process of finding the information inside the input tags by writing the following code:

```JavaScript
console.log(document);
```

The `document` variable contains the entire HTML page inside of it (Yes, really). Running the above code shows you the entire page, which isn't very useful. Try refining the results:

```JavaScript
console.log(document.clientForm);
```

The `clientForm` we added to the end of document finds the <form> tag we pasted in earlier. We are slowly getting closer to finding the information stored inside those pesky <input> tags. The next step is to use the `name` attribute inside the <input> tags. Notice what the `name` of the first <input> tag is: `id`

```JavaScript
console.log(document.clientForm.id);
```

This returns the information we want, but it's still wrapped up in the HTML returned from the HTML page. To access the information stored in the actual <input> tag, end the line with `value`.

```JavaScript
console.log(document.clientForm.id.value);
```

So now, if you type something into the "Client ID" input and click `Add Client`, you should see the console print that same value! So now that we can grab information from the HTML page, let's store that value in a variable:

```JavaScript
var clientID = document.clientForm.id.value;
console.log(clientID);
```

Your job is to add the HTML attribute `name` for the other <input> tag, and write the JS to grab that information out of the HTML page.

9) Writing to HTML

First step in writing to the HTML page is to think of a place we want this information to appear. Let's add some important information to our table so we can write to it:

```Markup
<tbody>
    <tr>
        <td id="firstID">1</td>
        <td>Alpha Client</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Beta Client</td>
    </tr>
</tbody>
```

This `id` is necessary for the next step. We want to write the `clientID` back into the table cell where we have the `id="firstID"`:

```JavaScript
document.getElementById('firstID').innerHTML = clientID;
```

The above line of code is quite complicated, but if you look at it long enough, breaking down the individual pieces, it makes sense. Let me break it down for you:

- We know what `document` means, it is the entire HTML page

- `getElementById('firstID')` grabs the HTML tag where the `id` is set to "firstID"
- `innerHTML` is the content inside the tag found in the previous step
- `= clientID` sets the `innerHTML` from the previous step to the "clientID" we saved earlier

Although everything in the above line of code has been explained and should make sense, does it seem intuitive? To someone who has never programmed before, it won't be. All you can do is practice and hopefully memorize these techniques. If you don't program for a couple months, you are likely going to forget the *exact* syntax, but that is why we have Google, to make our lives easier.

So, before you forget everything explained above, write the code in the HTML and JS file to transfer information from the `Client Name` input to the first client name cell of the table.

10) Make Pretty!

We have added new content to our webpage, play with the CSS to make it look good. Try adding some space around the inputs so they don't run into each other. Change the colour of the button if you want!

11) Submit

- Show your lab instructor your working webpage.
- Save the Lab8 folder for next week.
    - **Do not save your work on the lab computer's hard drive.** It will be deleted by the system.