CSC 105 Lab 7: CSS

Lab 7 Topics

1. Basic CSS
    - General Selectors
        - body
        - img
    - Specific Selectors
        - ids
        - classes
    - Styles
        - Box-Model
        - margin
        - border
        - padding
        - content
        - heigh / width
        - background color
        - text color
2. Bootstrap
    - container
    - pull-left / pull-right
    - tables

Goal

The goal of today's lab exercise is to introduce you to CSS (Cascading Style Sheets). Last week we learned how to write basic HTML, which we learned is just the content and basic structure of a webpage. But how do we make our websites look good? There are plenty of good looking websites out there, and currently our website looks.. lacking. That is because we haven't made a CSS file that connects to our HTML page. A CSS page allows us to make the background pink, make the text lime green, and even make our font Comic Sans!

Let's walk through the process of connecting CSS to HTML, and learning a few basic constructs of CSS.

Deliverable

By the end of the lab show the instructor a completed webpage, the result should be similar to:

CSC 105 Time-Tracker

By: Lloyd Montgomery - LinkedIn

My name is **Lloyd Montgomery**. I am a Masters student at the University of Victoria. I have a BSc in Computer Science, and I hope to one day have a PhD as well. My passions include teaching, creating websites, and playing Squash.

Clients

Client ID	Client Name
1	Alpha Client
2	Beta Client

© Lloyd Montgomery

However, this lab includes a lot of "free-form" and therefore everyone's completed lab should look slightly different. I have left mine pretty simple as I find simple webpages look the best.

Concepts

The following section discusses the key concepts required to complete the exercise described above. If you are familiar with these concepts feel free to skip to the exercises.

CSS

CSS can be thought of as a set of instructions as to how things should look. Webpages can exist without CSS, they just look plain. In fact, HTML existed before CSS, and they had special styling they did inside the HTML directly, which we do not do anymore. So, how do we use CSS to style our webpages? Well it starts by connecting the HTML to the CSS, then we tell the CSS what we are changing on the HTML page, and finally we tell the HTML exactly how to look. These steps are explained in detail below.

CSS & HTML Connection

When writing a CSS file, we need to tell HTML where to find the CSS file that we are writing our styles in. This is done through a simple <link> tag in the <head> tag of the HTML page:

```
<head>
    <link rel="stylesheet" href="./style.css">
</head>
```

The CSS file MUST BE right next to the HTML file given the above code. If you wish to store your CSS file somewhere else, you'll have to adjust the 'href' above to match the location of your CSS file.

Styling

CSS is really just a set of styles that are being applied to HTML. Each one disconnected from the others, listed one after the other. This is done by listing a tag name, id, or class, followed by the styles you want applied to it. Below we have listed the 'body' tag as the target tag to style, and we have chosen to make the background color pink.

```
body {
    background-color: pink;
}
```

Selectors

CSS Selectors are the way in which we connect our CSS to the HTML we have written. As mentioned earlier, there are three main kinds of selectors: tag names, ids, and classes. You can list any tag name and apply styles to it. For example,

```
section {
    background-color: red;
}
```

This will make all <section> tags in your HTML page appear red. This kind of styling is often too broad for what we want though, so use this sparingly. Instead, it is important to get familiar with ids and classes.

IDs and Classes

The first step in using IDs and Classes is to put them in our HTML page. This is actually quite simple. Here is an example of using an ID in a section tag:

```
<section id="random">
    This is a small section in which we have text.
</section>
```

Notice that the `id="random"` is inside the section tag, before the '>'. That is important. Also, the actual name that we give the id is important, because that is what we use to link it to our CSS. Here is an example CSS style being applied to the ID: random.

```
#random {
    background-color: blue;
}
```

Notice that when mentioning an ID, you have to put a '#' before the name of the ID. The syntax for classes is very similar. You just have to say 'class' instead of 'id' and use a '.' in the CSS intead of a '#'.

```
<section class="big-font">
    This is a small section in which we have BIGGER text.
</section>
```

```
.big-font {
    font-size: 24px;
}
```

Both IDs and Classes can be used interchangeably, so which one do you use? In general, IDs should be used when you intend to only use that style once, for one ID. However, some times we want our style to be applied to multiple tags in our HTML page, and this is where we use a class. For example, the class above: 'big-font' can reasonably be used in multiple places, so it fits as a class.

Styles

There are hundreds of different styles you can apply to your webpage, so we are not going to list them all. It is your job to use Google when you want to know how to do a particular style. How do you change the font color? Google it: 'CSS font color'

Box Model

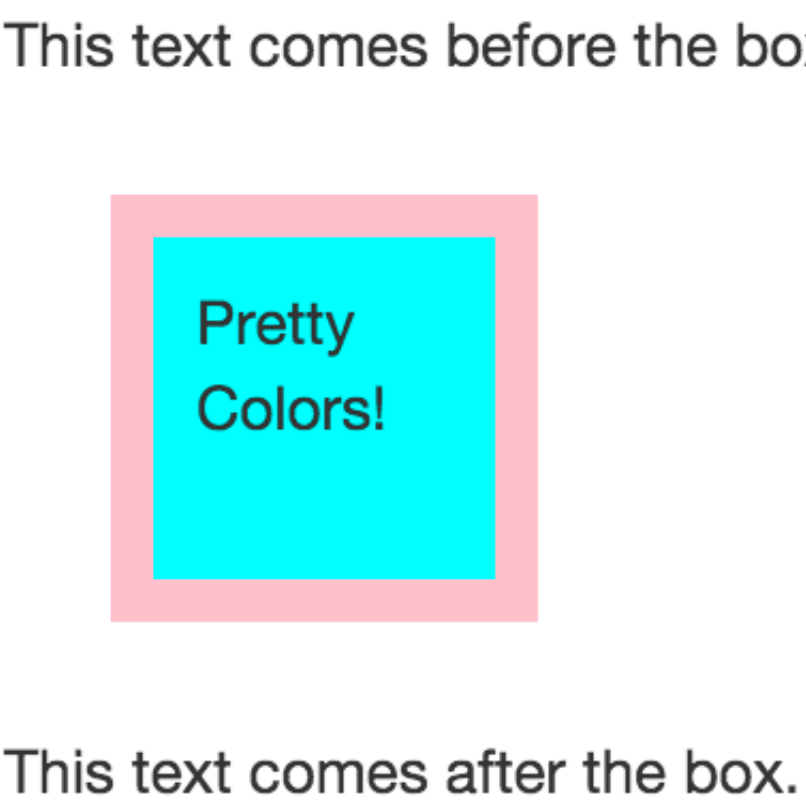
The last thing that needs to be understood before you can go on and experiment with CSS is the CSS Box Model. "All HTML elements can be considered as boxes. In CSS, the term 'box model' is used when talking about design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content."

The following example should hopefully give you a good idea of how these things come together to create the Box Model:

```
This text comes before the box.
<div id="box-model-example">
    Pretty Colors!
</div>
This text comes after the box.
```

```css
#box-model-example {
    width: 100px;
    height: 100px;

    margin: 25px;
    border: 10px solid pink;
    padding: 10px;
    background-color: cyan;
}
```



Play with those numbers and colors until you feel like you have a good understanding of the Box Model. Notice the margin and how it pushes the text around it away? We can't color margins, they exist just to give us space. Same thing goes for padding: it is used to give us space inside elements.

Bootstrap

People have been doing webdev for many years, and they are quite good at it. It is important to learn the basics yourself, but it's also nice to use work that other people have done. One of the most commonly used libraries is called "Bootstrap". By using Bootstrap, we can get a lot of pre-built CSS styles that look really good,

and are easy to use. Learning all of the things available to you in Bootstrap is quite the task in and of itself, but worth it since you can build webpages in hours instead of days.

The basic premise: add classes to your HTML page that Bootstrap has defined and styled for you, and suddenly your HTML has fancy styling. We will go over a few Bootstrap styles today in lab: container, pull-left / pull-right, and tables. Just know that there are plenty more out there to use, and if you do get into webdev it is a good idea to spend some time learning about them.

So how do we use Bootstrap exactly? Well exactly like we used our own CSS page: Add a <link> tag in the <head> tag, and then put pre-defined classes into your HTML. Don't worry about the <link> tag, that is given to you. Remember that table we created last time? It looks pretty boring. Let's add a Bootstrap class to our HTML and see just how easy and good looking it is.

HTML for Bootstrap

```
<table class="table table-bordered table-striped">
</table>
```

Original Table

Client ID	Client Name
1	Alpha Client
2	Beta Client

Bootstrap Table

Client ID	Client Name
1	Alpha Client
2	Beta Client

The classes we added: `table table-bordered table-striped` are Bootstrap defined classes that give us a nice looking table, and all we had to do was add a couple classes to our HTML page. There is no CSS that we had to write to make this happen. How would you know what classes to add to make this happpen? You don't. You have to either already know what to write, or Google it. For the purposes of this class, you are not expected to know any Bootstrap classes. However, since they are easy to use and look nice, we will use a couple of them to make our webpage look good.

Exercise

The following section describes steps to guide you through the exercise to create an HTML page. However, given the in-depth explanations given about each concept above, the exercises are not going to give as much explanation as prior labs. If you are confused as to how to complete an exercise, you may have to read some of the concepts above in more depth.

1) Open a Text Editing Program

The lab machines have a few text-editing programs installed:

- Notepad++
- TextPad
- jEdit

You are welcome to use any text editing program you want. I personally use and recommend SublimeText, but you'll have to install that on your own machine in your own time if you wish to use it.

2) Create a CSS File

Open up the completed Lab 6 folder from last week, and create a new file in the same directory as the "index.html" file, name this file: "style.css"

3) Open the HTML file in Google Chrome

1. Right-click on your "index.html" file on your Desktop
2. Open with -> Google Chrome

You should see the webpage we created in Lab 6.

4) Link the CSS File to our HTML file

Your HTML page will not know the CSS file exists unless you tell it. Go to your HTML file, and add a <link> tag between the <head> tags.

```
<head>
    <link rel="stylesheet" href="./style.css">
</head>
```

We will not know if this worked until we have something in our CSS file to style our webpage. Go to the next step to add our first CSS style.

5) Style the Webpage

Add your first style to the webpage to see if it is working properly. Let's add a margin to the <body> tag so everything isn't so squished agaist the side of the browser.

```
body {
    margin: 25px;
}
```

Refresh the webpage in Google Chrome, and you should see all of the text move away from the edge of the browser.

6) Artist License Go!

Now that you know how to style a webpage, practice! Add all of the following styles to your webpage in different ways. There is no correct answer to this step, just include at least one of each of the following styles:

- margin
- border (with color)
- padding
- height
- width
- background color
- text color

7) Add the Bootstrap Link Tag

Bootstrap is a library stored online. You access its functionality by adding the following line to your head section:

```
<head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
    bootstrap/3.3.6/css/bootstrap.min.css">
</head>
```

8) Add Bootstrap Elements

Container

Let's space out our webpage even more with nice Bootstrap classes. The class "container" is used to specify high-level elements that need space around them. Add the class `container` to all <section> tags, reload the webpage, and see what happens.

Tables

Let's make our tables look nice. Add the classes `table table-bordered table-striped` to the <table> tag and reload the webpage.

Pull-left / Pull-right

Often times, we want elements on our webpage to always be left, or always be right, and Bootstrap has a solution for that: Pull. If you add the class "pull-left" to a tag, that element will float on the left side of the screen, allowing other elements to wrap around it. Add the class `pull-left` to the <img> tag and see what happens to the content around it.

9) Remove the Inline CSS

Last time we added an image and inside the <img> tag we added `style="width:75px;"` however, this is not proper as all styling should be inside the CSS page. Delete this line so your HTML looks similar to this:

```
1 | <img src="./img/logo.jpg" class="pull-left">
```

You'll notice that if you reload your webpage the image becomes very large, which is not good.. So now you need to style that <img> tag with your CSS file so that it is a smaller size. Once you have done that, go and remove all of the styling inside the HTML file, and move them to the CSS file (if any remain).

10) Make Pretty!

At this point, you should have a pretty good understanding of how CSS works. Now comes the painful part of webdev: being an artist. Webdev is an art form in that there is not a defined right answer to "how should this webpage look?"; therefore, it is up to you and your artist license to decide how to style your webpage. For example, after step 8 the title "CSC 105 Time-Tracker" got awfully close to the picture we have, which in MY opinion looks bad. So if this was my webpage, I would add a little margin to the <img> tag to push the text away. Try `margin-right: 10px;` . Keep adding changes, while keeping in mind that CSS can be frustrating and eat away all of your time.. Have fun!

11) Submit

- Show your lab instructor your working webpage.
- Save the Lab7 folder for next week.
    - **Do not save your work on the lab computer's hard drive.** It will be deleted by the system.