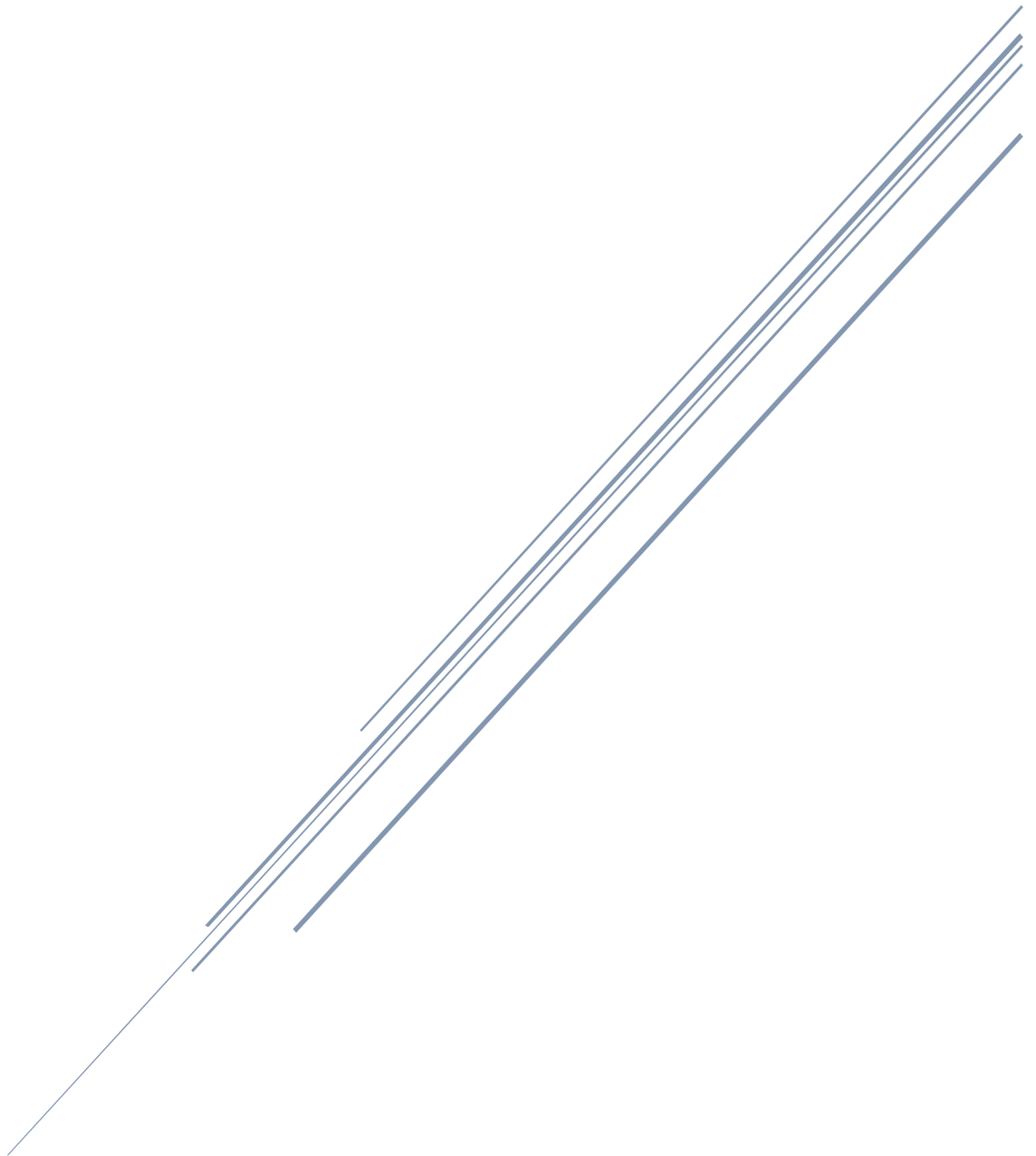


Lab#3

CST8285 F2022



LAB OBJECTIVE

The objective of this lab is to get familiar with the following:

- *Semantic HTML and Cascading Style Sheets (CSS)*

Earning

To earn your mark for this lab, each student should finish the lab's requirements, submit your lab on the Brightspace and demonstrate the working code to the instructor.

Assignment

Read the entire lab instruction before starting.

This lab is to be completed on BrightSpace any lab worksheets handed in will be discarded. Carefully follow the procedures outlined in this lab worksheet. If at any time you are unsure or are having problems, consult your lab instructor.

Part I: Implementing a Look and Feel for a Webpage

In this lab you will take the provided unformatted HTML document and format that document using a cascading style sheet (CSS). HTML5 has done a great job of differentiating tasks. As we have established already, HTML is primarily responsible for describing our web documents and CSS is primarily responsible for managing the look and feel of those documents. One of the first tasks that we will undergo in integrating CSS into our web documents is to ensure that we can specify all the elements of the HTML documents that we will need. This lab will walk you through some of these design decisions and your job will be to use well formed HTML and CSS code to realize the design.

The <head> Element

In the starter code there is a meta data tag for the author of the page, begin by making sure that your name is included as the content for this page. You will also change the text in the browser tab for your webpage to read "Lab 3: HTML and CSS".

In your code editor create a new file and name it lab3 with the file extension .css. Add a comment at the top of this file using C-style block comments (*/* ... */*) that includes your name. This will be your external CSS stylesheet. We will need to tell our html files where to find this stylesheet. The reason we use an external CSS stylesheet is for modularity. If we use inline or even embedded CSS then when we want to change the style rules, we must visit all the places in our html files where those styles are defined. By using an external linked stylesheet, we make it much easier to manage our web site formatting.

In your html pages we will add a link in the <head> element. Remember from the cascade that proximity to the element is evaluated, so if you are to include embedded styles in a page you will want to put them after this link. The <link> tag is used to establish the relationship between the HTML document and another document. link requires a few attributes:

- rel= which defines the relationship that the linked document has to the HTML document
- type= which includes both the encoding format for the file and the type of file that it is (text/css)
- href= which contains a URI to the file we wish to link to our HTML page.

If I called my stylesheet style.css then I would format my link as:

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

When your HTML document is being rendered, the browser will request and include the CSS stylesheet applying the style rules to the document that is rendered. For this lab we will do all of our styling with this one linked CSS file.

The <body> Element

The body of the webpage already contains much of what you will work with. You will surround the elements on the page with the appropriate semantic HTML tags. Note that if you have multiple elements, like paragraphs, which can have different formatting based on where they are in our document, you can use identifier attributes to name the elements. There are two identifiers that are globally available to all HTML tags: id= and class=. An id must be unique in the HTML document and an element can only have one id value. Class names are different, many elements can have the same class name and an element can actually have more than one class you separate classes with a single space.

Our overall semantic design structure will be:

```
BODY
  HEADER
    A title for our page
    NAV
      Links to each of the three major sections of our webpage
    /NAV
  /HEADER

  MAIN
    HEADING
    PARAGRAPHS
    LINKED HEADING
    PARAGRAPHS WITH IMAGE
    LINKED HEADING
    PARAGRAPHS
    LINKED HEADING
    PARAGRAPHS
  /MAIN

  FOOTER
    NAV
      Link to top of the page
    /NAV
    Text Formatting and code by add your name.
  /FOOTER
/BODY
```

The content you will use for this lab is included in the starter code for this lab. You will add the HTML tags to realize the semantic design structure and then CSS to format the design structure as indicated in the steps below.

HTML Preparation

1. In the header for the webpage, we have the title “CSS for Web Development” you will wrap this header in an appropriate tag so that we can format it later.
2. Wrap the navigation elements with the nav tag. We will add the links in step 4.
3. Identify the four headings in the main section and tag them appropriately. Add a unique id for each section heading tag, these will be the targets for our navigation links.
4. The links in the top nav element should be linked to the section headings they match with and the bottom nav element should be linked to the first section heading “Where Does CSS Come From?”
5. In our document we will have at least two types of paragraphs: general paragraphs, and quotation paragraphs. You will use an appropriate class attribute to identify each paragraph.
6. You will add the provided waterfall image so that it is inline with the section “How a Style Sheet Works”. We will format this image with CSS, for now just include the image.
7. The footer also includes a line indicating who wrote the text and who did the coding. You will add your name as the formatter and coder of this webpage. Make sure you wrap this line of text in an appropriate semantic tag so that we can format it later.
8. Make sure your code is clean, indented reasonably, and includes no inline CSS. You should open up the file in a browser to inspect how it looks without CSS formatting. We will add more html tags when they are required to identify part of our content for special formatting with CSS.

CSS Formatting

With CSS we can realize (make happen on our webpage) a number of style and layout features. This includes having a colour scheme that includes a background colour, text colours, as well as border and decoration colours. Our style will also include typographic formatting such as choosing appropriate font(s) and how those fonts appear. Additionally, CSS will handle any layout for our elements. The following instructions will help you to achieve all of these goals.

1. The header and footer should have a different background colour than the main page. When you are building your style rules you can combine life features by listing selectors separated by commas. You can also use a universal selector (*) to add the same style to all of the elements. Using the



colour palette below, which includes hex values for the colours, make the background of the entire page Green Yellow Crayola (#DDD78D) and the background of the header and footer Umber (#60594D). The text colour for the document should be a shade of black (not pure black) and the text on the

header and footer will use the accent colour Morning Blue (#93A29B) or a very light shade of gray.

2. Choose a sans-serif font for your webpage and make that the font that is used for the whole page. You can include a full font stack, or you can make sure that the last font listed is sans-serif.
3. Add a 4-pixel border around the header and footer in the same black as the main text on the page.
4. Add 10px of padding all around the header and footer so that the elements inside are not touching the border.
5. Change the font size of the title in the header to be the largest sized text on your page without wrapping in the header element on a normal computer screen.
6. The navigation elements in the header You can play with the formatting CSS to make the edges rounded. Clicking on the nav element should take you to the targeted sections of the webpage.
7. Center the navigation elements in the header, space them out evenly using a **flexbox**.
8. For all anchor text on your page remove the underline (text-decoration) and add a hover pseudo element to change the text color to Rose Taupe (#8B635C). You will need to target the navigation elements differently and make the background of the navigation elements change colour on a hover event. The colour of the anchor text when not hovered over should be the same as the other text in the same context (body, header, etc.). All nav element text should be 12-point font.
9. In the main section, make each of the section headings bold and larger than your main text font. All of your paragraphs will have a 14pt font size. Paragraphs that are not quotations will be left justified. Paragraphs that are quotations will have a Gold Crayola (#DCBF85) background and will have centered text.
10. Add a margin to the top and bottom of your paragraphs of 10 pixels. Add a margin to the left and right of quotation paragraphs of 20 pixels. Add 10 pixel of padding to all sides of the quotation paragraphs.
11. Any unordered line elements should be bold (just the part immediately following the bullet). Line items should also all have the same properties as a general paragraph (do this without duplicating the CSS for the general p elements).
12. Float the image of the waterfall to the right of the second main section. You should also set the size of the image to 25% of the screen width and add a 10-pixel margin around the image so that it is not squished up with the text.
13. In the footer leave the link as text making sure that the decoration (underline) is removed and that when you roll over the link it turns Rose Taupe. This should already be done based on previous steps. Using CSS center this link and below it the author line should be right justified, italicized, and shrunk to a 10pt font size. Make sure that there is at least 20px between the link and the author line without using a br tag. You should have no br tags on your web page.

14. Inspect your webpage in a browser to make sure everything is laid out as expected. It should look something like this:
15. Submit your code to an HTML validator (<https://validator.w3.org/>) and take a screen shot of a properly validated HTML page, do the same with a CSS validator (<https://jigsaw.w3.org/css-validator/>).

Part II: Submitting your Work

When your code is complete, make sure you zip up all the files (lab3.zip) created in this lab including a copy of any images that you have used in your growing website.

- Your modified version of the starter code.
- css stylesheet
- screenshots of both HTML and CSS validation showing successful validation of code.
- all images used

CSS For Web Development

How a Style Sheet Works

The Order of the Cascade

The Anatomy of the Selector

Where Does CSS Come From?

The idea of style sheets is as old as standardized general markup languages. Cascading Style Sheets, usually referred to as CSS, were initially proposed in 1994. This was during the times of the browser wars and CSS was the least consistently rendered language for the browsers. Opera was an early almost perfect adopter, but the W3 Consortium were trying to create a standard with buy in from the more popular browsers Internet Explorer and Netscape. As CSS became standardized web browsers began to improve their support and today most modern browsers support CSS in a fairly consistent manner. The HTML5 standard has eliminated much of its design formatting features and offloaded that work to the CSS standard. HTML5 expects you to use CSS to make your pages look good. This is because HTML was always great at describing documents, but often terrible at making them look really great. And making webpages look great is where CSS shines.

CSS easily lets you build stylesheets for whole websites so you can make changes to your corporate identity in one place instead of having to touch every file in your website. If you decide to move all your website backgrounds to blue from brown, no problem! Just modify and publish that master style sheet CSS file and you are done.

How a Style Sheet Works

The word cascade is used to describe how CSS works and here should remind you of a waterfall. The water falls down and hits the various ledges until it looks a certain way. Waterfalls are beautiful, so it is a fitting metaphor for making the web a beautiful place.

This idea of a waterfall effect is what Håkon Wium Lie, the inventor of CSS and chief technical officer for Opera, intended for us to think about when he proposed CSS in 1994. In CSS we can have many different style rules ranging from general to very specific, and often it is the last, most specific rule that will take precedence over how the browser will display the web page content. Think of it like this: we wash the styles over the document until it looks just the way we want it too. While Håkon Lie called this process of styling the webpage the cascade, others have called it the war of the styles.



The Order of the Cascade

Understanding the rules of the cascade are important. Often when students have unexpected results from CSS it is because they do not understand how CSS decides what style rule wins in a given situation.

When two elements have the same specificity value then the declaration that comes last in the source code wins.

There are three levels of priority in the evaluation of the CSS cascade: *Origin and Importance*, *Specificity of the Selector*, *Order of Appearance*, and *Inheritance*. Each of these priority levels has different rules as to how they are evaluated, but when we have a rule that clearly wins at a level then it is applied to the webpage, otherwise the next level is evaluated. This is why sometimes you add a style rule and it seems to make no change to your webpage, something with a greater priority has applied its rule first.

• Origin and Importance

Origin has to do with where the style rule originates. This allows the browsers and even the end user to make design decisions based on their needs, which is helpful for greater accessibility in web browsing. You can include an important operator (!) to a style rule to override the origin priority. Note that all browsers have default style rules which have the lowest priority at this level unless they are given an important operator (!) which will give them the highest priority.

• Specificity of the Selector

Selector specificity has to do with how specific our selector is. A selector on a generic HTML tag has the lowest priority here. The order of preference for selector specificity is:

1. Inline style rules that are part of the HTML tag (style= attribute). These are the most specific because they only apply to the element to which they are attached.
2. Identifier (id=) selectors which use the # symbol in CSS. Because these are meant to be unique identifiers they are the next most specific way to select an element.
3. Class, pseudo-class, and attribute identifiers are the next level of specificity. A class can reference many elements so it is less specific than an id= attribute. I can also reference states of an element like hover (though) or find an element that has an attribute with a specific value like