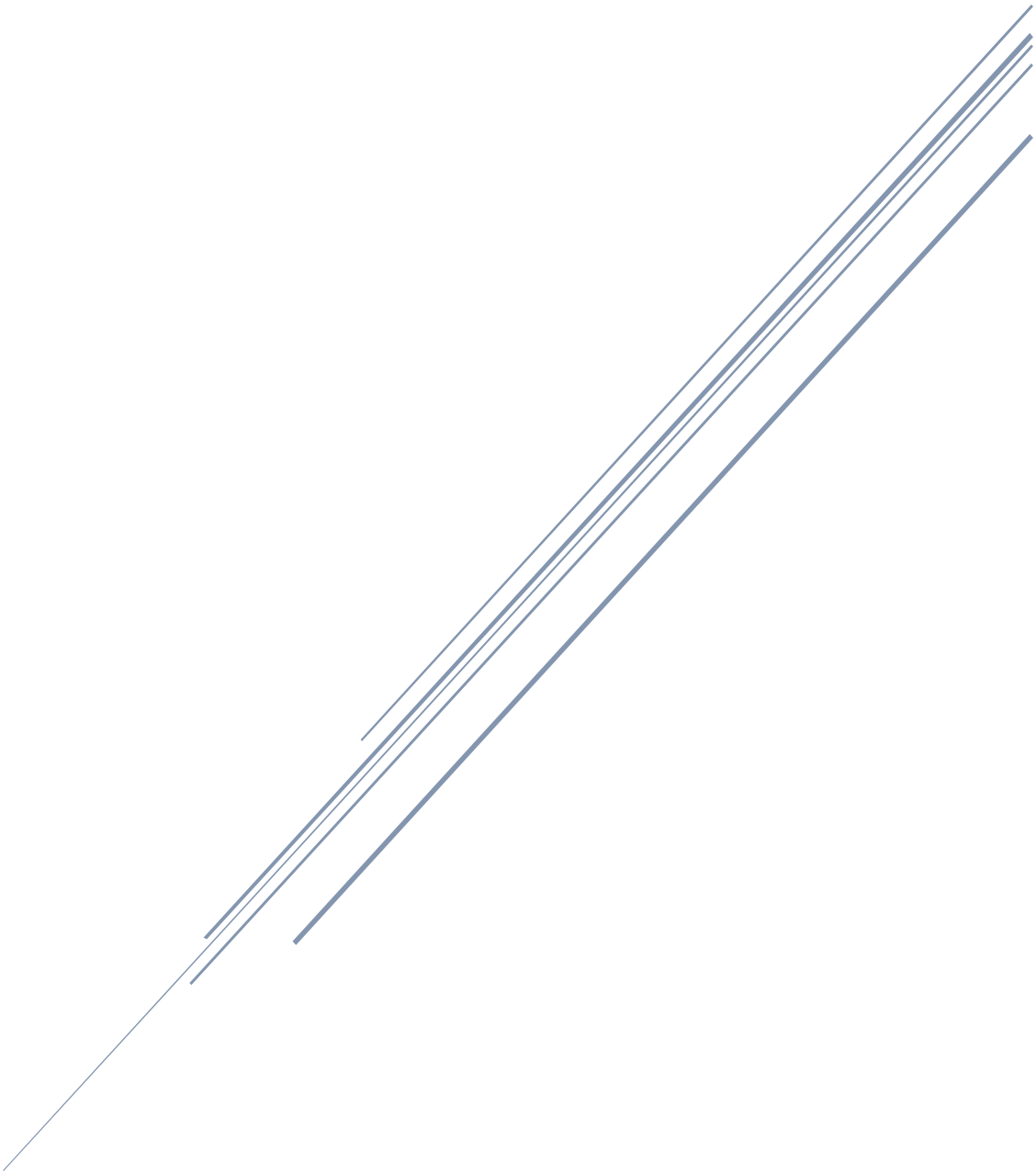
Lab#3

CST8285 S2023



ALGONQUIN COLLEGE | to transform hopes and dreams into lifelong success

LAB OBJECTIVE

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

- *Semantic HTML and Cascading Style Sheets (CSS)*
- *Experimenting with CSS library*

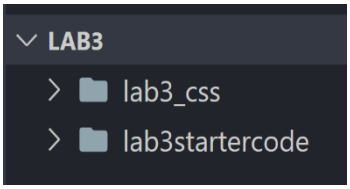
Earning

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 followthe procedures outlined in this lab worksheet. If at any time you are unsure or are having problems, consult your lab instructor.

Create lab3 folder with 2 sub-folders as shown here:



For submission you will zip the lab3 folder and upload it.

Part I: Implementing a Look and Feel for a Webpage

Part I of lab3 will be implemented in lab3-css folder

In this part you will take the provided unformatted HTML document and format that document using a cascading style sheet (CSS). HTML5 has done a great of 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 these 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 newfile 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.

CST 8285 Lab#3

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 HMTL 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 figure 1 below.

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/).

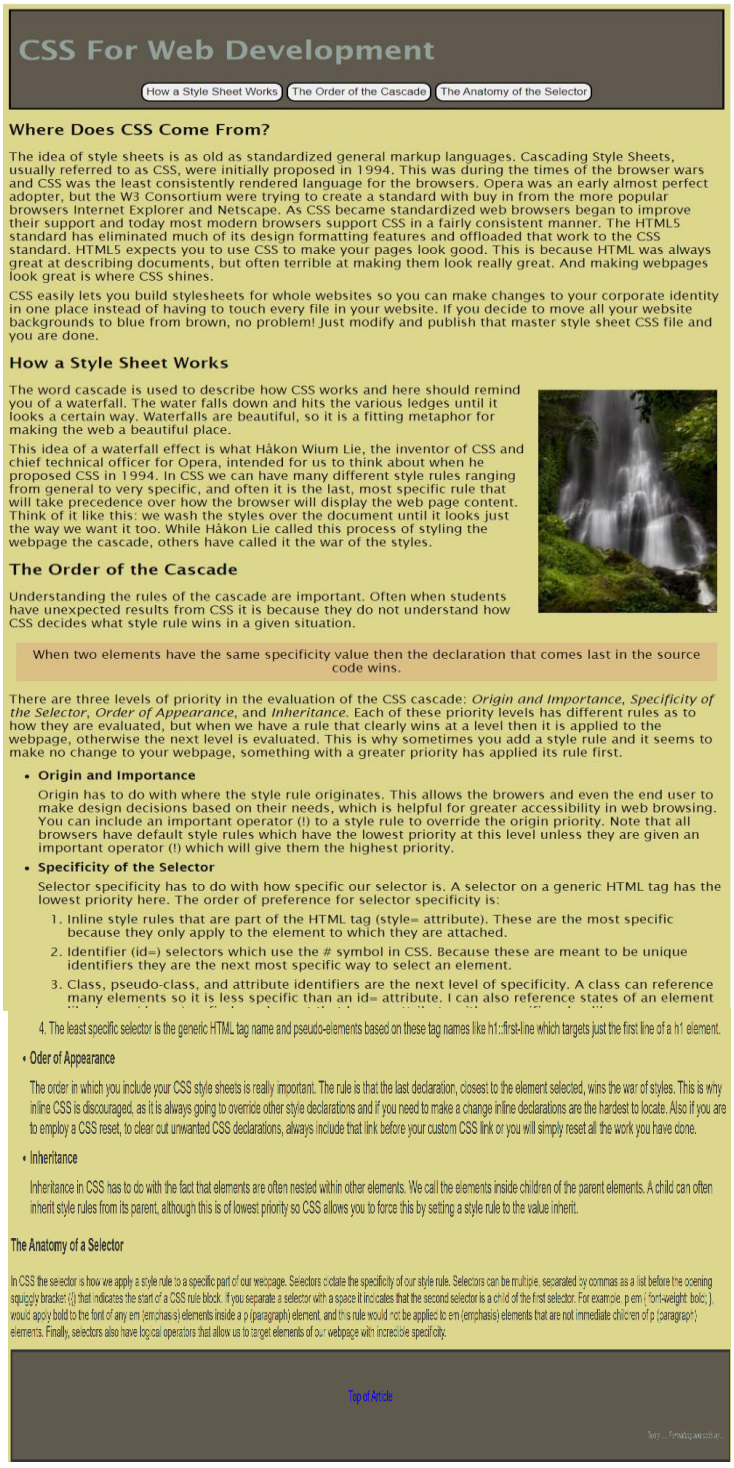


Figure 1

Part II: To understand and practice using CSS flexbox for creating flexible layouts

Part II of lab3 will be implemented in lab3-css folder

This part gave you a hands-on experience in utilizing CSS flexbox to build flexible and responsive layouts.

Instructions:

1- Open the starter code file “flexbox.html”
2- Create a new CSS file and name it "flexbox.css" in lab3-css folder.
3- Open the "flexbox.css" file and add the necessary CSS code to complete the following tasks:

a) Add margin and padding 0 pixel to all the body elements.
b) Add any font family from your choice to all the body elements
c) Set the background color of the header to "#333" (Dark charcoal)and the text color to **white**
d) Add padding of 22 pixels to the header.
e) Center align the text in the header.
f) Create a flex container with the class "container".
g) Set the display property of the container to "flex".
h) Arrange the flex items horizontally and ensure equal space between them.
i) Allow the flex items to wrap onto the next line when there is not enough space.
j) Set the maximum width of the container to 800 pixels and center it horizontally on the page.
k) Add a margin of 40 pixels on the top and bottom of the container.
l) Style the boxes with the class "box" inside the container.
m) Set the background color to "#f1f1f1".
n) Add a border of 1 pixel solid color "#ddd".
o) Set a border-radius of 5 pixels.
p) Add padding of 20 pixels to the boxes.
q) Create equal-width boxes using flexbox properties.
r) Add margin 20 pixels to the paragraph located insides the footer
s) Ensure each box has a minimum width of 200 pixels.
t) Create a responsive layout using media queries.
u) Define a media query for screen widths less than or equal to 600 pixels.
v) Inside the media query, change the flex direction to vertical for the container.
w) Adjust any other properties as needed to create a visually appealing layout for smaller screens.
x) Open the "flexbox.html" file in a web browser to view the initial layout.
y) Test your layout by resizing the browser window to ensure it responds correctly.
z) Add any additional styles or enhancements to make the layout visually appealing.

Inspect your webpage in a browser to make sure everything is laid out as expected. It should look something like figure 2 below.

Part III: 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 for part 1.
- Your css stylesheet file (of Part I and Part 2)
- Screenshots of both HTML and CSS validation showing successful validation of code. (Part I and Part 2)
- all images used

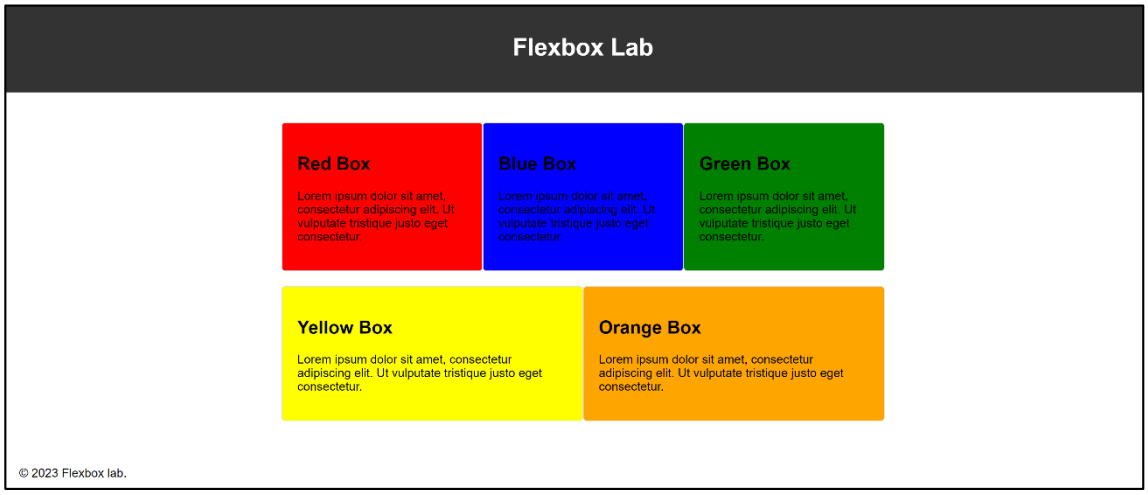

Flexbox Lab
Red Box
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vulputate tristique justo eget consectetur.
Blue Box
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vulputate tristique justo eget consectetur.
Green Box
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vulputate tristique justo eget consectetur.
Yellow Box
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vulputate tristique justo eget consectetur.
Orange Box
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut vulputate tristique justo eget consectetur.
© 2023 Flexbox lab.

Figure 2.