CST8117 Lab 4: Website Development (HTML, CSS)

Note you will need to complete this lab by uploading a .zip file containing all the required files onto BrightSpace. The content of this lab's instructions will be included in the weekly quiz given the week after this lab is due. The due date for this lab is posted in the description on BrightSpace. Late labs are penalized -10% each day up to -50%, and after 2 weeks a lab is no longer eligible for points.

Read the entire lab instruction before starting.

This lab is to be completed on BrightSpace. Please carefully follow the procedures outlined in this lab worksheet. Steps in the tutorial section will be numbered. **If at any time you are unsure or are having problems, consult your lab instructor**.

Required Tools and Resources: Visual Studio Code, zip of collateral files.

Part I: Website Development Workflow

Good news! The folks at Fluffy Friends Cat Daycare love the web page you built for them. They are ready to have you build them a whole website. They hand you a small bundle of collateral including schedules, plans for a cat tree, staff bios, some images, and certifications they would like people to know about. They trust that you will build them something great, but before you do, they have some changes they would like you to make on the landing page (index.html). You agree and take all their feedback and collateral back to your computer and see what you can do.

As your professor I encourage you to work smarter, not harder. You could sit down and code each of these pages one at a time, but it could take you longer than the week allotted to this lab. However, if you follow a simple development workflow then you can easily complete the task in a reasonable amount of time. A workflow is a way of breaking down complex tasks into smaller modules attempting to eliminate as much duplication of effort as possible.

The first step in any front end web development project is to complete one page that is exactly what the client wants and then simply use that one page as the basis for all the other web pages in the web site. This approach ensures **consistency** and **efficiency**. In our project the lab instructions will follow this workflow. First, we

will use the client feedback to perfect the landing page. Second, we will use the list of pages the client wants to build the navigation for the web site. Then we will isolate the CSS and put it in a separate file which all our pages will use, this allows us to keep the design consistent throughout the whole web site. Finally, we will use the perfected landing page as our template to build all the other files for the Fluffy Friends Cat Daycare website. Let us see what we can do about the feedback on the page we developed in our previous lab.

Part II: Tweaking the Landing Page

Note that you are provided with a baseline version of the final webpage from our last lab. It will be a bit different than the one you submitted. And it will serve as the jumping off point for our work in this lab. The baseline lab is called `index.html` *in the collateral zip file.*

This is the feedback that the client gave us:

- Although they liked the nice big header we made, they did not like that when it was on a smaller screen the company name moved down. They would like the header to remain on one line no matter how small the window is made.

- They would like the text in the navigation bar, the main portion of the page, and the footer to be a bit larger. We told them that we went with a 14pt font in the footer and navigation bar and a 16pt font for the rest of the page. They would like the main text to be at least a 20pt font, but they trust that we will not make it too large.

- In the main text they would like the images to line up with the top of the headings and to have the text wrap around the images.

- They have a certification from Pet Diversity Plus and from the Better Business Bureau which they would like to display on their webpage. We suggested that this would be great in the footer on the left-hand side and they agreed.

- We also determined that we will want links to the following pages in our navigation in addition to the home page: Staff page, Schedule page, Registration form, Contact page (at bottom of registration form), and a special Project page which will have plans for making your own cat tree.

1. Unzip the collateral file into a directory so that we can work on the website. Inside you should find a word document, a baseline landing page (`index.html`), and a subfolder called images with several image files. Open the folder you created in VSCode and examine the landing page. Notice that in the `<head>` there is a comment where your standard meta data should be inserted. Remove this comment when you have inserted your meta data. Also examine the embedded CSS in the `<head>` and notice how it is organized. Each selector follows roughly the order it will appear in the web document. This is not necessary for the code to run, but it does make finding the CSS you need to work with much easier. You should also note that when you are working with CSS it is good to clean up any style rules that you do not need, often we will try different rules and find better ways to accomplish our layout. Leaving in extra code might not break things at first, but it does open up that possibility.

Non-breaking Header

2. The reason that the header images move down onto the second line is because all HTML follows the rule of document flow. The document is built from left to right and down. This means that when an element is unable to fit within the viewport (width of the screen) then it is wrapped to the next line. If you open up our landing page in a browser and shrink the width of the screen you will notice that first the company name image will drop below the happy cat logo, then if I keep shrinking it will eventually be cut off. What we would like to do is keep the company name image to the right of the happy cat logo even when there is not enough room to display it. This means that shrinking the width of the page will simply cut off some of that image instead of making it taller.

One of the tools we can use to break out of the document flow is the `position` style rule. We need to apply `position` to the image that we would like to remain where it sits in the initial document flow. Because we want the element to stay in the exact same spot on our page no matter what we need to give `position` the value of `absolute`. If we do not specify an offset from our initial position it is as if we set the initial position to the value `initial`. This means that we can make the change we would like with just one line of code.
Often when we are working with CSS we need to determine exactly where (specificity) we need to apply our style rule(s). In this case we might assume that applying this rule to the images in our header would achieve the desired effect. So we add the following CSS code to our `<style>` section and examine the effect on our web page.

```
header img {
    position: absolute;
}
```

Notice that the effect is not exactly what we are going for. This is because `position` is applied to both images in the header. Position removes the element from the document flow which means that the next element in line, in this case the the `<nav>` element is brought to the top of the page. Additionally, the happy cat logo was removed first so it gets covered up by the company name image. We will need to leave one of the images in the document flow so that the header does not disappear. It we leave the happy cat logo it should work better. But how can I target just the one image?

The simplest of ways is to give the image we want to break out of the document flow a name with an `id=` attribute. Then we can simply replace `header img` as our selector with the #name we gave to the image. If we added `#id="coname"` to the second `img` tag in our header then our selector becomes `#coname`. Try this and see if it works.

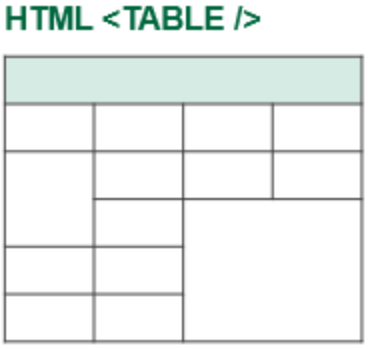
Later in the course we will learn how to make elements on our page responsive to various browser configurations so that they behave more reasonably when the width or height of the screen changes.

Increase the Font Size

3. Change the font size of the main sections of our page to be at least 20pt finding something that looks appealing. When you have settled on a font size make the text in the nav and footer elements to be 2pt less than the main text. This process of changing things and examining how it looks can be tedious, you should try using the inspect element tool in your web browser to adjust the style rule values and see the results instantly. When you have the styles the way you like you can simply copy them into your embedded CSS. This is a massive time saver.

Flow Text Around Images in Main Sections

4. For the next adjustment we are going to have to remove the tables that we had used to lay out our web pages. If you want to see why doing text wrapping with tables is a nightmare, try to accomplish the layout that they requested using tables: Line up the top of the image with the `h2` heading, and have the text wrap around the image (so it should go under the image when the bottom of the image is reached). Even when you get the layout looking ok, adding in another image, or resizing an image means reworking your tables. A better way is to remove the tables and use CSS to layout the elements. Remove the table and we will explore the joys of CSS Float.



In the lectures we look at some of the 'design' capacities of HTML tables. Early web designers were quite resourceful in levering this tool and accomplished some interesting designs despite tables not being intended for such usage. One big drawback to tables is that they are restricted to the HTML document flow. This means that you must force the table cells into the shape and size you need to have any hope of a reasonable layout. Many times, you simply want to have an image or an aside that sits on the page with other elements, like text, flowing around them. With tables you can do this although it is not possible to have text in one cell flow naturally into another cell. This means that changing your view size often produces unattractive results or, if the tables are fixed in place, results in the web page being cut off in the browser. The first evolution from tables was the notion of position which we have seen already in this lab. Position allows us to break out of the document flow but any item no longer in the document flow no longer interacts with other elements on the page – which is why you can end up with images over top of text. The next evolution then was meant to solve this problem – breaking an element out of the document flow without giving up its impact on the overall document flow. We call this tool CSS float.

In the image on the left you can see float applied to the first picture on our web page. Notice that the text flows naturally around the image and if we adjust the screen width it will continue to flow naturally around the image even though the image has been taken out of the natural flow of the document. Note that to use

float I needed to remove the table leaving me with a section code that looked something like this:

```
<section>
    <h2>
        Let Your Cat Annoy Someone Else
    </h2>

    <img src="images/sleepingcats.jpg" alt="sleeping cats">

    <p>
      Fluffy Friends Cat Daycare offers premium cat care services
      during normal business hours (weekdays 9AM to 5PM). If your
      cat is lonely at home while you are working at the office.
      If your cat is annoying you while you try to work from home.
      Dropping off your cat at Fluffy Friends is a great solution!
      Our caretakers are experienced cat owners who will feed, pet,
      and play with your cat throughout the day. Your cat will
      develop lifelong friendships with the other cats at our daycare.
      We also have a total of 42 cat trees in our tiny location!
      Great places for your cat to sleep and scratch their day away.
    </p>
</section>
```

We have three elements that we need to establish the relationships amongst: A heading, an image, and a paragraph. The client has asked that the image be in line with the heading and that the paragraph text will flow naturally around the image. We will achieve this by floating the image to the right of the heading. To set a starting point from where our image will break out of the document flow, we need to place the element in its relative starting point, usually in relation to a block element. To align the image with the heading block element, as a starting point, we need to put the image element (the HTML tag) as the first item in the block level element to which it should be aligned. In this case move the whole `<img>` tag into the `<h2>` element right before the heading text of 'Let Your Cat Annoy Someone Else'.

Now that we have placed the image in the correct starting point of the document flow we need to add some CSS to break it out of that document flow. To add some CSS to our image element we need to be able to target it from our CSS. There are many ways to do this. Because we have multiple images on the page, we cannot simply apply the CSS to all the images. We could apply it to the section and write a CSS selector to target the images in a specific section. But being able to float

images might be nice to have available for a variety of contexts. Therefore, it makes sense to make a class of right floating elements that we can apply to other images or even aside text elements in articles, this way we can just add the class to any element we want to float right. In our CSS we need to add the following class selector and style rule:

```
.rightfloater {
    float: right;
}
```

You can come up with your own creative class name, I always prefer something descriptive so that the code becomes its own natural comment. To apply this class to the element we need to add the following attribute to the image tag of the image we want to float right: `class="rightfloater".` View the results in a web browser playing with the width of the browser window. Notice that the image gets uncomfortably close to the text at times, we can fix this by adding some padding around the image. We can add this to the rightfloater class as we likely want all of our right floating images to behave in a similar way: lined up with the top of the header, tight to the right edge of the viewscreen and a 20pt buffer on the remaining sides. Remembering our TRouBLe mnemonic, we set the `padding` to `0px 0px 20px 20px` in our CSS:

```
.rightfloater {
    float: right;
    padding: 0 0 20px 20px;
}
```

5. Repeat this same process to make a class of left floating elements and apply this to the cat porch image. Do not forget to remove the table first.

Add Certification Logos

6. In the collateral file you will find two certification logos as `.png` images. The client would like these added to the left-hand side of the footer. The leftmost icon should be the Pet Diversity Plus logo followed by their Better Business Bureau A+ rating logo. Both images should be the same height and there should be a small space, 10px, to the right of the images.

Begin by adding image tags for each of the logos into the footer so that they are

before the `<address>` element. Do not forget to add a reasonable `alt=` attribute.

The next step is to target our images for formatting in our embedded CSS. In this case we want all the `img` elements in the `footer`, so we can use the selector `footer img` to target all images that are children of the `footer`. Now we can create some style rules:

```
footer img {
    height: choose a reasonable size;
    padding-right: 10px;
    float: left;
}
```

You should play with the height property to choose something that looks good on the page, icons like this are often small. If you choose to make them larger than the height of the text in the footer, then you can have a situation where the images overflow their containing block element. If you do the solution is to apply a clearfix hack to your page. I will discuss this in a future lecture, but for now the w3schools webpage on clearfix is an excellent resource.

Add Links for Navigation

7. Now that we have addressed our client's concerns about the design of our page it is time to prepare for building the rest of the website. Because our landing page will be our template, we should add into our page hypertext links to all the other web pages that will be part of our complete website.

The idea of a landing page is that this is the first page a web server will present to a visitor. Only the landing page will be named `index.html`, in honour of the history of the web where the first page used to be an index of the web site itself. Note that index.html is not the only option for a landing page and we can set the order of possible landing pages in the web server's settings. By default, the landing HTML page is `index.html` so we will simply use that for our main web page.

In our nav element we will add links to the other pages in our website: `staff.html`, `schedule.html`, `registration.html`, `registration.html#contact`, and `project.html`. Note that the fourth page is an anchor location on the registration page so we should make the element of our link say something like 'contact'. Come up with single word names for each of these links and add them to the `<nav>` element of your landing page. Do not

worry that these pages do not exist yet and that the links do not yet work, you will be building them in this lab. As you build each page you will make sure that the links are correctly working so that you can easily navigate your whole website.

Part III: Linking a Site Wide CSS

At this point we have a pretty good template to speed up our work. Before you tackle this next step tweak the embedded CSS to your liking by adjusting style settings such as image sizes and font spacing and the placement of elements in the `<body>` of the html document. It is important to have as complete a template as possible before the next step because once we start making more pages any change to the foundational layout will need to be changed in all the documents. Remember, the goal is to work smarter, not harder.

8. When we are happy with our landing page we need to pull out the CSS so that we can leverage CSS we create for new pages in other new pages without having to constantly update all of our web pages. Having a single main CSS file for the whole website is the best way to do this. If down the road we decide to change the font sizes, we can do that in one CSS file, and it will be consistently changed for all our web pages. The embedded CSS that we built in our template will be used to make this CSS file. Start by making a new file in your working directory, for now we can leave it in the same directory as our `.html` files. Name this new file `main.css`. Cut and paste all the style rules from the `<style>` section of our `index.html` into this new document.

9. Before we delete the `<style>` section form the `<head>` of our landing page we will create a link in the `<head>` to our newly created `main.css`. There are several ways to link our CSS to the HTML files, in this lab we will simply add a `<link>` element to our `<head>`. If you are adding several CSS files to a webpage you must always keep in mind the CSS cascade to make sure that you order their inclusion properly. In the `<head>` add the following element after the closing `</style>` tag:

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

The `link` element tells the web browser to stop parsing the current file until after the linked file is parsed, which allows us to add in CSS formatting before the page resumes building the web page from our HTML. At a minimum we need to tell the browser how the linked file relates to our document and provide a link to that

document. In this case the relationship is that our linked file is a stylesheet for our document and the `href=` attribute points to the CSS file itself. I have also added the `type=` attribute which is the MIME type for the linked document. We can add other properties to the `<link>` tag which we will see later in our course.

10. Delete the `<style>` section from the `<head>` in our landing page and verify that your page is still styled correctly. If it is then you are ready to move on to the part of the lab.

Part IV: Building the Site from the Template (Challenge)

11. In VS Code we can open our landing page (`index.html`) in the editor and from the file menu choose the save as option. Save the file as staff.html. This creates a duplicate of your page which we can use as a template for the staff page. The only part of this page that will change is the `<main>` element, although the main element will be similar to the landing page.

Using the name and bio information in the collateral document as well as the image of each staff member follow the pattern of the landing page (alternating which side the picture is on) and create the staff page. It should have a similar layout to what you see in the collateral document. Add a link in Lillian's bio to the new cat porch article on the landing page as well as a link in her bio to the project page.

12. Using the same process of saving a copy of the index.html page with a different name, make a page for each of the other links in our navigation bar. Follow the guide in the collateral document to populate the web pages. These pages will include the following features:

`schedule.html`

- Use table elements to create two weekly event tables: one for young cats and another for older cats. Make sure that you combine cells as shown in the collateral schedules using the `colspan=` and `rowspan=` attributes.
- Add colour to some of the table cells to make the schedules more attractive.
- Include register here links on the page that take you to the registration page.

`registration.html`

- Use a form element to create a form laid out in a similar manner to what is shown in the collateral document. Note that this form does not need to work,

you simply need to layout the appropriate elements (text boxes, check box, and register button) on the webpage so that they can be filled out. The register button does not need to work. We will learn how to program forms later in the course.

- Include a contact information section that has an anchor with an `id=` attribute so that we can link here from the navigation bar.
- Make sure that the email address is clickable similar to how we did the email address in the footer.

`project.html`

- Use the layout in the collateral document to create a page about building cat trees. This page will use an ordered list of steps to create a cat tree.

Part V: Submitting Your Lab

When your code is complete, make sure you zip up all the files created or modified in this lab, including a copy of all your website's images (`lab4.zip`). Make sure when you zip up the files you add the images subdirectory so that when I extract your lab the image links are not broken. It is important that you zip up your files rather than uploading them individually because BrightSpace will modify any unzipped web code that you submit. You should have the following files in your zip file:

- **`index.html`**
- **`schedule.html`**
- **`register.html`**
- **`staff.html`**
- **`project.html`**
- **`main.css`**
- **`/images folder with all the image files used in your web page`**

Upload the zip file to BrightSpace.

Marking Rubric (/25)

Task	Weight
Web pages are all properly formatted for HTML5	2
Semantic layout is applied to all the web pages	1
Navigation links send you to the correct pages of the website	3
No inline and no embedded CSS	1
Images in the main sections are properly floated	2
CSS is linked properly to all the HTML files	1
Images are reasonably sized	1
Footer and navigation bar text is at least 18pt	1
Icons are properly floated in the footer	1
Web pages follow the layout of the collateral document	2
Tables in schedule are properly formed and colored using CSS	3
Registration form is in a single form element and includes text fields, checkbox, and a submit button with the text ‘register’	3
Project page aligns floated elements with specific paragraph block elements	2
All files accounted for in lab4.zip.	2