Comm 362: Digital Media Foundations
Assignment 2: Web Pages

Deadline: Wednesday, 10/16, 9am (one hour before class begins)

Objectives:
- Understand basic web browser functions
- Make a basic web page and share it online
- Differentiate semantic and non-semantic mark-up (e.g., via HTML vs. CSS)
- Use a web form to send data across the Web to a computer program
- Begin learning good coding practices

Folder to Turn In:
In this assignment you will be producing multiple files (at least five [5] HTML files and two [2] CSS files). However, you should upload **ONE** compressed folder that contains these files within it to Canvas. Instructions to compress the folder containing your files are included at the end of this assignment. This folder will be uploaded to Canvas just as you uploaded the individual files for Assignment #1. The files within the folder will include:

- ☐ One HTML file that you create in Part B (index.html)
- ☐ Three HTML files you create in Part C / D
    - ☐ page1.html (with a form)
    - ☐ page2.html (with an embed)
    - ☐ page3.html (plain)
- ☐ One CSS file you create in Part C (style1.css)
- ☐ One or more HTML files you create in Part E (names up to you)
- ☐ One CSS file you create in Part E (names up to you)
- ☐ The image files that you use in your web pages (at least four)
    - ☐ Image 1
    - ☐ Image 2
    - ☐ Image 3
    - ☐ Image 4
- ☐ Your lab report (a MS Word document)

NOTE: Make sure you keep all your files together in the same folder.

Part Zero: Before You Begin

To complete this assignment, you'll need:

- More than one web browser (e.g., IE, Chrome, Firefox, Opera, Safari, …)
- An advanced text editor. On the lab computers you will be using Notepad++. Mac users may want to use BBedit (linked on course webpage) or Atom.
- Internet access
- A place to save your work (a USB drive, Google Drive, your computer's hard drive, etc.)

Part A: Understanding Browsers

<assignment>

OVERVIEW: In this section you'll be experimenting with different browsers and the functions within them. Some of the steps below require you to write a text response. **Save these text responses** in a word processor, text editor, draft email, or anywhere else you can save text. You'll be using these responses later in Part B.

Scaling & Resizing the Browser

1. Open a **web browser** and **navigate** to
   http://digitalmedia.wtf/assignment2/

2. In addition to displaying web pages, browsers can manipulate the content on the page. One way you can manipulate a web page is by zooming in and out. For instance, in Internet Explorer, navigate to 'View > Zoom' and select a setting. If you have a scroll wheel, you can also try holding the 'ctrl' key and scrolling up and down to zoom in and out in most browsers (this only works on Windows). **Find the zoom settings on your browser**. Try out a number of zoom settings and take a mental note of how the content on the page changes.

3. Now take a look at how resizing the browser window affects how the page is displayed. Use your mouse to **resize your browser window** to various sizes by dragging the lower right corner of the window to different shapes. Note how the content shifts on the page (if it shifts at all).

4. What did you see when you zoomed in and out of the page? What did you see when you resized the page? Explain how this changed the layout (for example, making some parts of the page unusable). **Answer these questions** in plain language in at most a few of sentences. Write and save your answer somewhere you can save text (such as a text editor or word processor). You'll be using it later in Part B. An example answer could begin: "When I resized the page, I noticed that …"

The Find Tool

5. Some web pages have a large amount of text that you may not want to wade through in order to find the information you need. Web browsers, and most text editors, have a handy build-in text search tool. In most browsers you can access the tool by pressing ctrl + F (PC) or command + F (Mac) keys at the same time. If you're not a fan of keyboard shortcuts you can also find the tool in your browser's menu. For instance, in Internet Explorer, go to 'Edit > **Find on this Page**.' After you hit the find command a search box should appear near the top or the bottom of the web browser screen, depending on which browser you're using. *This does not search the Internet, it only searches the content on the page you are on.* **Locate the "Find on this Page" tool** (or the equivalent tool) in your browser and try it out.

6. Make sure that you're still viewing this page: http://digitalmedia.wtf/assignment2/
Using the "Find" tool, **count how many times the letter "a" appears** on the page. The search bar should indicate this number. Record your answer in one sentence, saving it the same place you recorded your answer for step 4. An example answer could be:
"The letter 'a' appears 5 times."

COMMON PITFALL: For this question make sure you are using **the correct URL**. It is this: http://digitalmedia.wtf/assignment2/ It is NOT the assignment .pdf that you are reading. Just be careful, the links can look similar.

Viewing Source Code

7. Web browsers display web pages by interpreting HTML code. You can also use the browser to see this code directly. You can **view the source code** of any web page in most browsers by right clicking the page (control + click on Mac) and selecting 'View Source' or 'View Page Source' depending on the browser. You'll be directed a new page that displays the code the browser interprets. In plain English, describe a familiar thing you see in the source code and note an unfamiliar thing that you see there (if any). One sentence is fine. For example, you answer could begin: "I recognize the <lolcat> tag from the InterACT book…" Record your answer.

TIP: The command to "view source code" in Chrome is [Ctrl]+[U] (Mac: [Cmd]+[U].)

8. A basic HTML file contains the *content* of the web page as well as HTML *markup*. In order to add color, layout, and style, the web browser must be given commands in another language called CSS. HTML documents can contain a link to the CSS stylesheet (which is also interpreted by the browser) in the <head> container of the source code. *While you are viewing the source code on the example page*, you can **look at the stylesheet** by finding the link titled "part4style.css" and clicking on it if you are using Firefox, Chrome, or Opera. This does not work in IE or Safari. Click that link.

9. In plain English, **describe the first few familiar things you see** in the CSS and note the first few unfamiliar things that you see there (if any). Please answer the question: How does this CSS differ from HTML? One sentence is fine. For example, you answer could begin: "The CSS file lolcat.css differs from an HTML file because it…" Record your answer.

COMMON PITFALL: Clicking on the css link in step 8 above does not work in IE or Safari.

10. **Find a Wikipedia page** describing something you like that begins with the same letter as your last name (e.g., if your last name is "Sandvig," you could choose to look up "sandpiper"). Complete steps 2-3, 5, and 7 above, but do so for this Wikipedia page. That is, zoom and resize the page, use the find tool, and view the source code. Note that the source code for a page like Wikipedia is much more complex than that of our previous example page. (You do not need to write new answers for this Wikipedia page.)

BONUS WORK: (Extra Credit) Can you explain how the CSS stylesheets work on a Wikipedia page? Note that this is a **hard question** if you are an HTML/CSS beginner. If you complete the assignment early you can come back to this question for extra points. Turn in extra credit answers at the bottom of your lab report. Label your answer "Extra credit: How Wikipedia pages use CSS:".

11. Open a **different browser** than the one you usually prefer. Go back to http://digitalmedia.wtf/assignment2/ and try steps 2-3, 5 and 7 again with a new browser. That is, zoom and resize the page, use the find tool, and view the source code. How does this compare to the browser you are used to? In at most a few sentences of plain language, note the name of the browsers you compared and write what differences you noticed (if any) in how they handled the page. Record your answer.

AWESOME PRO TIP: The "view source" command is a tool for Web developers, not users. Many Web browsers include tools ***like*** "view source" that help Web developers do their work. For example, both Chrome and Firefox includes a whole set of tools called "**Developer Tools**" that you can access by pressing [Ctrl] + [Shift] + [I] (on a Mac [Cmd] + [Opt] + [I]). Both Firefox and Chrome include tools called "Page Info" and or "Site Info." Internet Explorer has developer tools that only appear when you press [F12] (on Windows). How many of these kinds of tools can you find in the browser that you prefer? Check them out -- they can be very helpful when you are looking at a Web page and you want to figure out how it works!

(image credit: http://www.wisdump.com/css/css-aid-tables-without-tables-misses-the-point-dark-side-web-standards/)

Part B: A First Web Page

OVERVIEW: At the end of this section of the assignment, you will have produced a simple HTML web page that can be read by any browser. It will resemble the source code you viewed in Part A.

- A few tips:
    - Be sure to **save periodically as you work**. You'll be saving with the file name "index.html". When first saving your document, make sure to change the file extension to .html instead of the default option for your text editor. In Notepad++, this has the added benefit of making it easier to write your HTML because Notepad++ automatically highlights common elements of HTML with color.

    - You can always **open your document in a web browser** during any step of this process to see how it is interpreted. If you are new to this, you might want to do this after every change. After saving it from your text editor, just click "**reload**" or "**refresh**" in the browser that is viewing the page.

    - You'll be writing lots of tags in this assignment. Always **close your container tags**. One particularly useful strategy is to type both the start and end tags for a container first and then fill the container with other text. That way you won't forget to include the end tag for the container. For example: " This is some text " In this example, " " is the open tag and " " is the closing tag for the container. Closing tags are the same as the opening tags, but with a "/" at the beginning of them.

Ok, let's go.

1. **Install a recommended text editor** if you do not have one already from the software page on http://digitalmedia.wtf/. In this document we'll refer to the PC text editor, **Notepad++** but the other text editors will work in a similar fashion (**TextWrangler**, **Caret; BBedit** and **Atom** work well for macs).

COMMON PITFALL: The text editor that came with your computer will not work.

2. On the top of your page, write the necessary **<!DOCTYPE>** tag as described in the required readings. There are many different doctypes, but for this assignment you should use the doctype declaration for HTML 5, which is " <!DOCTYPE html> ".

3. Begin the HTML portion of your page by adding the **<html> container** of two tags. Here is a screenshot of Notepad++ showing this container:

```
3  <html>
4  </html>
5
```

4. Use the **<head>** tag correctly to create a head portion for your page. (See InterACT Ch. 12.) The <head> should contain metadata**.** Specify your page's character set by adding the exact tag: `<meta charset="UTF-8">` All pages should also have a "keywords" and "description" meta tag as indicated in the InterACT book. You do not yet know what your page is about, so there is no way to know what keywords and description to include in these <meta> tags inside your <head>. Go ahead and create the two meta tags for "keywords" and "description" and just leave them empty -- you can fill them both in later once you know what your page is about. Finally, give this page the <title> tag "uniqname's Index" where "uniqname" is your UM uniqname.

COMMON PITFALL: Don't forget to fill in your two <meta> tags for "keywords" and "description" once you know what your page is about. You can leave them blank for now.

PITFALL: The <head> container exists to hold ***metadata*** (that is, information about the information) in a Web page. Contents of the <head> aren't shown to the user in the main browser window. That's why the <head> is a separate container from the <body>, and why the *one tag **isn't allowed** to contain the other*. The <head> is usually smaller than the <body>.

5. Next, use the **<body>** tag to write the body portion of your page (see InterACT Ch. 13). The rest of your content visible to the user will be written in this section. At the very beginning of this step, a screenshot of your progress might look like this (the correct meta data has been removed to keep from giving you the answers to the assignment):

```
3  <html>
4
5  <head>
6        <!--Your meta-data goes here-->
7  <title>NetID's Index</title>
8  </head>
9
10 <body>
11 </body>
12
13 </html>
14
```

6. Write a **level 1 heading** (see InterACT Ch. 13) that says "Behind the Digital Screen – uniqname's Index" where "uniqname" is replaced by your UM uniqname.

7. At this point, you should have made a note of some text from Part A (steps A4, A6, A7, A9, and A11). In order to turn in your answers, you'll be including them in the <body> of this Web page. **Create five headings** named after the question they correspond to. For instance, to record your answer for step A4, the heading would read, "Answer to A4."

8. After each of your headings, **create paragraphs** for each answer to the corresponding question. Rendered in a browser, your page should look something like the screenshot below:

Intro to Digital Media - NetID's Index

Answer to A4

Your answer here.

Answer to A6

Your answer here.

Answer to A7

Your answer here.

Answer to A8

Your answer here.

Answer to A10

Your answer here.

COMMON PITFALL: Remember that HTML has reserved characters that cannot be used in the content of the document because they have special meanings for the HTML language. (The most common examples are: **< > & "**) When you add the content of your HTML pages in this assignment you need to look for reserved characters and replace them with **character entities** or your HTML will not display correctly (see InterACT Ch. 10, p. 168).

PRO TIP: Using a character entity is also called an "**escape**." That is, you are going to "escape" from the browser's normal interpretation of the character (&) and make it do something else.

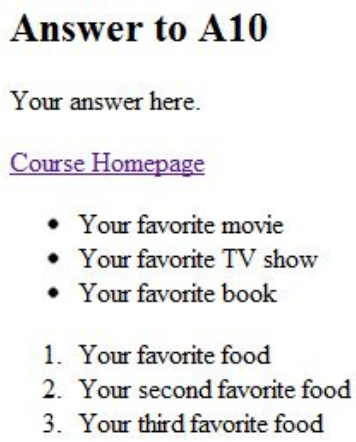
Adding a Link

9. After your Part A answers, create an **absolute link** to the URL for the course homepage (digitalmedia.wtf) using the proper syntax for the <a> tag. The text for the link should say "course homepage." In other words, if you do this right, the user should be able to click on the words "course homepage" and be taken to the course homepage URL. Your text might look like this: "Visit the course homepage." Or this: "I sure do love the course homepage, don't you?" The user should not see the URL. (See InterACT Ch. 15 if you are stuck.)

Working with Lists

10. Write an **unordered list** that includes your favorite movie, TV show, and book. To make it an 'unordered' list, use the <ul> tag. Within the <ul> container use <li> containers correctly to specify each list item. (See InterACT Ch. 17.)

11. Write an **ordered list** of your three favorite foods. This works just like an unordered list except instead of using the <ul> tag you will use the <ol> tag. At this point, if you haven't already, you should check to be sure your page displays the way you want it to. The last part of your Web page, viewed in a Web browser, might look like this screenshot:



Adding a Table

12. Use the content you created in the last two steps (B10-B11) to **create a 2x4 table** (two columns, four rows) using the **<table>** tag and associated tags like <tr> and <td>. The first cell of the first column should be a table header (<th>) that says "Favorite Media" and the first cell of the second should be a table header that says "Favorite Foods". The remaining cells beneath these headers should be your favorite media and foods from your lists. You'll be making use of the <th> (table header), <tr> (row), and <td> (column) tags. Set the border="1" attribute on the <table> start tag so that you can see the table lines.

TIP: The <table> tag is explained in detail in the Duckett reading, Ch. 6: Tables.

BAD PRACTICE: It is OK to use the border="1" attribute now because we told you to, but ideally you would not use an HTML attribute to control display or layout. This is a bad practice. You would instead use CSS. We haven't taught you how to do that in CSS yet, though. So don't worry about it this time.

POSSIBLE PITFALL: Remember when you are writing more advanced HTML and CSS that <table> tags should never be used for the visual layout of the page, they should indicate that you are trying to present tabular content, like a list of names and addresses (or in this case, a list of preferences).

Working With Images

13. Let's talk about images. Track down the final image (.svg file) you created at the very end of Assignment 1 and **make a copy** in same folder as your web page. For the purposes of this assignment, **rename** the copy of your logo to "logo" (the ending should still be ".svg").

14. Insert your logo with the <img> tag. Remember that your image **will need to be in the same folder** as your HTML page for the browser to find it. Within your <img> tag, include relevant "alt" text. Remember that image tags don't act as a container and don't need a closing tag. (see InterACT Ch. 16.)

1. Your favorite food
2. Your second favorite food
3. Your third favorite food

Favorite Media	Favorite Foods
Movie	Food
TV Show	Food
Book	Food

Commenting Your Code

15. It is good programming practice to **document your code** with hidden comments so that other programmers can understand what you did. The start comment tag is "<!--" and the end tag is "-->". Comments may span more than one line, and comments can contain other HTML

tags. Review your HTML code and insert meaningful comments that would help another Web developer understand what you were trying to do. **Insert at least three HTML comments**: one just before your Part A answers, one before your lists, and one before your image. You can choose what to write within your comments, but make sure it's something that explains what you're doing with the code and something your GSI can understand. For instance, if you were writing a comment before your table, you might write something like "<!-- table with favorite media and foods -->".

PITFALL: HTML comments (starts with **<!--**) don't work the same way as CSS comments (starts with **/***). Don't try a CSS comment in an HTML document or vice versa—it won't work!

16. There you have it. You've coded a simple web page in HTML complete with headings, paragraphs, lists, images, tables, and comments. This page will act as a homepage of sorts for the rest of the assignment. If you haven't yet tried it, **view it in a Web browser** to ensure it displays correctly.

PITFALL: SVG graphics may not work if you are using an old Web browser, particularly an old version of **Microsoft Internet Explorer** (IE). Since you are using SVG graphics on this page, view your page in a browser other than IE if your graphic is not showing up. (Or upgrade to the latest version of IE.)

Real Web developers don't like IE very much. Web developers made these:

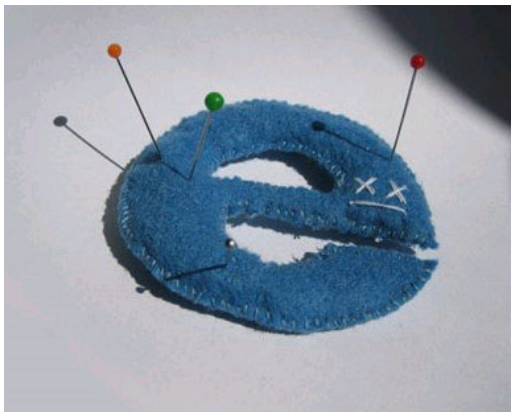


Image Credits: http://www.daobydesign.com/blog/tips/css-html/hate-internet-explorer-take-a-stand-against-ie6-browsers/, http://www.ecwid.com/forums/showthread.php?p=41225, http://www.twenty4.com.au/tag/css

Part C: Intro to CSS

For the next two sections you will be creating three different web pages.

The content of these web pages will be **three different scenes** from a **book, movie, TV show**, or your choice of entertainment media (if you prefer, it can be **lyrics to three different songs**). Don't use the same choice as anyone else you know in this class. These scenes should be class appropriate.

Write up the three scenes in a word processor. If the scenes are from a written medium you can simply copy the text from the scenes straight from an existing transcript (if you can find one), or the source (if it is online). If the medium is something visual, such as a TV show or movie, you can write up your own representation of the scene. Save this writing somewhere that you'll easily find it again, because you'll be using it later. Right now we don't care about your creativity in writing them up—the point of this exercise is to provide each person in the class **a different set of text** that they can use to make three Web pages.

IMPORTANT: You should have *at least two paragraphs* of text for each of your three scenes. For example, lyrics from three very, very short songs will not work.

TIP: The complete text of many old books that are no longer protected by copyright can be found online for free at Project Gutenberg: http://www.gutenberg.org/

Example choices:

- Three scenes from the novel Tess of the d'Urbervilles by Thomas Hardy: (1) Tess becomes a poultry keeper, (2) Angel asks Tess to marry him, (3) Tess goes to jail.
- The lyrics from three songs by Consolidated: (1) Born of a Woman, (2) Consolidated [Adorno strength mix], (3) No Answer for a Dancer.

1. Open **Notepad++** or your text editor. **Create a new document** and save it with the file name "page1.html"

2. Add the **basic tags** <!DOCTYPE>, <html>, <head> and <body> to your document, just as was explained in Part B (steps B2-B5). Use three **meta tags** in the <head> as you did in Part B (charset, keywords, description) but write them so that they describe the content you chose.

3. In the <head> section of your document, use the **<title>** tag to give your web page an appropriate name. Remember, you will be using content from the three scenes you chose earlier, so it would be a good idea to choose a title related to your content.

4. Without closing the HTML document, open up a new document in the text editor. This will be the **CSS style sheet** for this page. Save it as "style1.css".

5. Leaving the CSS page open, navigate **back the HTML document**. In the <head> tag **create a relative link** to the style sheet using the **<link>** element (If you're confused about what "relative" link means, see InterACT Ch. 15, p. 253). Setting things up this way is called using an "external style sheet" (defined in InterACT Ch. 11, p. 176.) It should look like this:

```
<head>
<!--This is the link to the style sheet-->
<link rel="stylesheet" type="text/css" href="style1.css" />
</head>
```

PITFALL: Different text editors use different color coding, so your version of Notepad++ or another text editor may not use the same colors shown. That's OK.

This <link> tag will connect the CSS style sheet file you just made (ending in .css) to this particular HTML document. CSS rules you place in the style sheet file will be applied the content in the HTML document.

PITFALL: For a relative link like this to work, both your HTML file and your CSS file **should be in the same folder**. All the HTML files you create for this project should be in the same folder, as should all the images you wish to put on your web page.

6. Now in the **<body>** tag of the html document do the following:

- **Structure the Document:** Use a level 1 heading container tag to give your page the title of the book/movie/TV show/band your scene is from. Use the other levels of heading tags to give one or more subheadings (as appropriate) that provide the description of the scene. If you're using a scene from a book you could tell us the chapter title, if it's a TV show give us the name of the episode, if it's a song give us the song name. If it's a movie or from another medium that doesn't use episode or chapter titles make one up that gives us a clue about the scene's content.

- **Add the Content:** Use the <p> tags appropriately to display the text content from one of the scenes you choose. Remember, each new paragraph should have a new <p> tag. <p> tags are containers and must be properly closed. It is important to give credit when you use content you did not create. If you didn't write this text yourself, at the end of the document, be sure to include a clickable link (using the <a> tag) to indicate the source of your content.

- **Add an Image of Your Choice:** Using google images, find an image that is relevant to the content of the scene you've chosen. Using the skills you learned in Assignment #1 (Web Graphics), save the image to the same folder as your HTML and CSS file. Depending on what the file name of the image is you may want to rename it. (It's always better to have descriptive file names, rather than something like "img2353.jpg"). Use an <img> tag like <img src="example_file_name.jpg"> to add the image somewhere on the page. It is important to give credit when you use content you did not create. Be sure to include a clickable link (using the <a> tag) in a paragraph (<p>) near the image to indicate the source of your image.

7. **Save your HTML file** and view it in a Web browser. This screenshot shows a general idea of what it might look like, but without enough scene content:

Book/Movie/TV Show Title

Chapter/Episode Title

Scene content

Scene content

Scene content

Scene content

Scene content

8. Navigate to your CSS file in Notepad++. **In the CSS file**, create a CSS rule that applies to the <h1> tag. Showing just the ***selector***, the ***declaration block*** (surrounded by { }) and three ***properties*** but (but not yet showing the ***values***) your rule should look like this (see InterACT Ch. 11, p. 172):

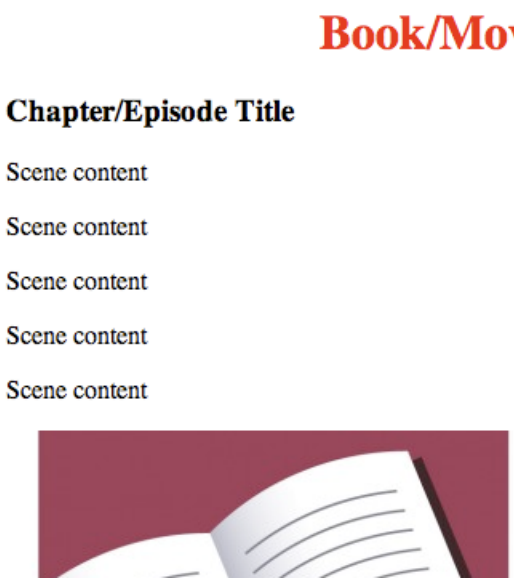
```
h1
{
        font-size:
        color:
        text-align:
}
```

For each of the three ***properties*** above, complete the CSS rule by adding the correct ***values***, properly ending each each line (**";**") so that the following are true:

- font-size - make the font size 30 pixels.
- color - make the font red
- text-align - center the text

If you have questions about CSS syntax or the specifics of certain properties, make sure to review InterACT Ch. 11 (for syntax) and Ch. 14 (for layout).

9. **Save the CSS document**. Navigate back to the web browser where your web page preview is. Refresh the browser. You should notice that the main heading of your web page has changed to fit with your style rule. The top part of your page might look like this:



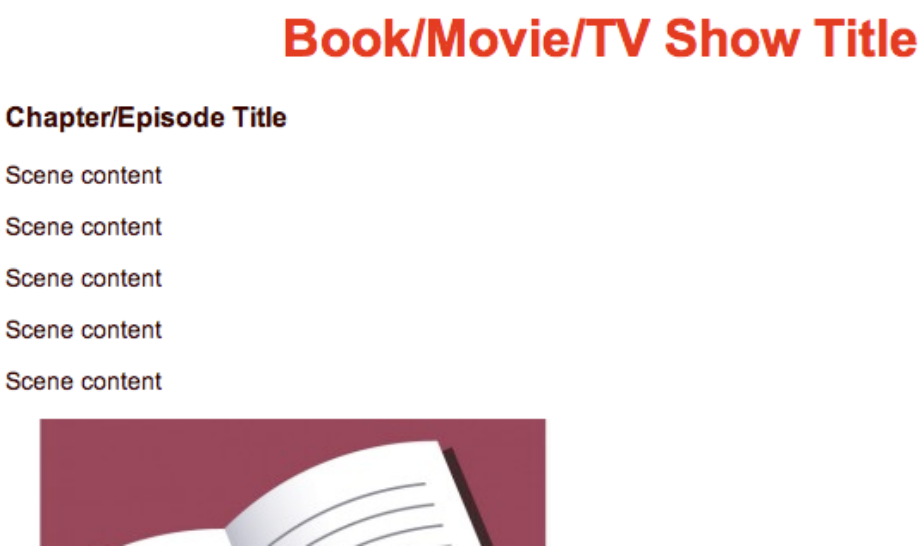
10. Go back to your CSS document and assign styles to the <body> tag. Write your CSS rule so that it includes these properties:

```
body
{
        font-size:
        text-align:
        color:
        text-indent:
        font-family:
}
```

For each of the above **properties**, add the right **values** (and correctly end the line with "**;**") so that the following is true:

- font-size - Make the body font size 1.5em
- text-align - Make the text align left
- color - Make the text some dark color other than black
- text-indent - make the text indent 50px
- font-family - Go to http://www.w3schools.com/cssref/css_websafe_fonts.asp and pick out one of the sans-serif fonts to use. (Make sure you look at the example at the top of the page to see how the formatting of the font-family property works.)

11. **Save the CSS document**. Navigate back to the web browser where your web page preview is. Refresh the browser. You should notice that the majority of the text on your web page has changed to fit with your style specifications.



12. Return to the CSS document. Next, create a CSS rule where the selector does not reference an HTML tag. Instead, **reference an id** that we will name "firstparagraph." (Confused about selectors for IDs vs. tags? See InterACT Ch. 11, p. 186). It should look like this:

```
#firstparagraph
        {
                font-size:
                color:
                background-color:
                text-align:
                text-transform:
        }
```

For each of the above **properties**, add the **values** (and correctly end the line with "**;**") so that the following are true:

- font-size – Make the font 50% larger than it would otherwise be using the measurement unit "em". (For a discussion of using an em in a CSS rule, see InterACT Ch. 11, p. 181.)
- color - Make the font color a dark color
- background-color - Make the background color a light complimentary color to the font color
- text-align - Make the text right aligned
- text-transform - Make all the letters uppercase

13. Save the CSS file and go back to your HTML file. Find the first instance of the <p> tag and **add the attribute "id"** with the value "firstparagraph". It should look like this:

```
<p id="firstparagraph">
```

PITFALL: Remember, any particular ID value (like "firstparagraph" in this example) can only be used **once** as an attribute in **each** HTML document. To apply a CSS rule to more than one thing in the same HTML page, use a class or tag selector instead.

14. Go back to your web browser with the preview of the website and **refresh the window**. The first paragraph on your page should have the style you previously set up. It might look something like this:

15. Return to your CSS document. **Create new rule using a class selector** named "importantbox" (Confused about selectors for classes vs. IDs vs. tags? See InterACT Ch. 11, p. 186). It should look like this:

```
.importantbox
{
        font-size:
        width:
        margin:
        padding:
        background-color:
        float:
        color:
        border-style:
        border-width:
}
```

For each of the above **attributes** add the **value** (and correctly terminate the line using "**;**") so that the following is true:

- font-size - Make the font size 18px
- width - Make the width 50% of the page
- margin - Make the margins 30px
- padding - Make the padding 10px
- float - make the class float to the left
- color - make the color something dark
- border-style - make the border style solid
- border-width - make the border width 5px
- background-color - make the background color light

For help with layout (like float, margin, and padding), recall the "**Box Model**" in InterACT Ch. 14, pp. 238-239.

PRO TIP: For this assignment, most of the CSS measurements have been in pixels (px). That's because this is usually the easiest measurement unit for people just learning CSS to understand. And it is pretty easy to see and understand pixels – they're the tiny dots on your screen! However, using pixel measurements in CSS is considered **bad design practice** because it makes assumptions about the screen the user will be viewing the Web page with. Later in the class we will discuss this problem in more detail, but for now, just keep in mind that using pixel measurements (px) is a crutch we will be getting rid of later. The preferred way to measure distances in CSS is the typographic unit called the "em."

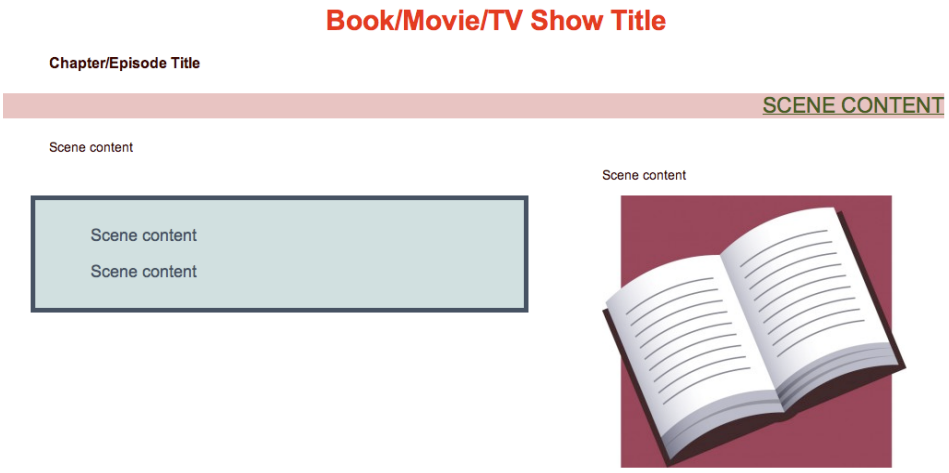
16. Save the CSS document and go back to the HTML document. Choose at least two <p> tags that are next to each other and **put a container** around them. It should look something like this:

```
<div>
        <p> Content </p>
        <p> Content </p>
</div>
```

17. Assign the tag the "importantbox" class by **adding the class attribute** to the starting tag of the container. That will look like this:

```
<div class="importantbox">
```

18. Save the HTML document and **refresh the page** in your Web browser. You should see that all the content within the tag now has that style applied to it.



19. Within the <body> tags of your HTML file **add a relative link** back to the page you made in part B. It could look something like this:

```
<p>Here is a link to <a href="index.html">the first page I ever made</a>.</p>
```

20. Save your file, return to your web browser and refresh. Click the link and **make sure it takes you to the web page** you made in Part B. If not, double check that you spelled the file name correctly in the link and that the files are in the same folder.

21. Go back to your text editor and create two new files. Save them as page2.html and page3.html. They should be located in the same folder as your other HTML file and the CSS file.

23. As before, **add the essential first tags** to your two documents: <!DOCTYPE>, <html>, <head> and <body>.

24. In the **<head>** section of your documents, use the <title> tag to give your webpage an appropriate name. It should be something similar to the title of your first page. Include the appropriate <meta> tags as you did in the <head> of your last HTML page (but describing the new content for your second scene).

25. Now in the <body> tag of the html documents **fill in the page with the content** related to your scenes, using headings, <p>, and choosing a different image of your choice. In other words, repeat step C6 (above) but for two more scenes.

26. If you haven't already, at this point you should save these HTML files in Notepad++, then open it with a web browser to **confirm that it looks the way you want** it to.

27. Go back to editing your HTML documents in Notepad++. In the <head> tags create a **relative link** to the **style sheet** you've already made. This is the same as step C5 (above). It should look like this:

```
<link rel="stylesheet" type="text/css" href="style1.css">
```

Note that the name of the CSS document should be that of **the CSS document you already created**. This will link both HTML pages to the same CSS style sheet.

PRO TIP: Putting layout rules in an **external CSS style sheet** is a powerful design technique because making a single change to one CSS file will control the layout for every single HTML page on your Web site that you have connected to the CSS style sheet with the <link> tag. That could be hundreds of pages!

28. Go back to the web browser and the **preview the new pages** by refreshing the browser. You should see that the content in your <h1> and the <body> tags from HTML document have been affected by the rules in your CSS style sheet.

29. Go back to editing the HTML document page2.html and page3.html. Pick a few <p> tags somewhere in the document and put them within a tag, similar to how you did in C16. **Assign the "importantbox" class** to this as you did in C17.

30. Preview the pages by refreshing them in the web browser. You should see that the content you put in the "importantbox" class is **shown the same way** that the content in the in your earlier HTML page is. This is because both divs are in the same class and are linked to a single style sheet that has a rule describing (or selecting) that class. Be sure the content looks the same before continuing.

31. On both page2.html and page3.html, **add a link** back to the page you created in Part B. The instructions for this can be found at step C19.

PRO TIP: In addition to meaningful comments, **proper indenting in your source code** is also good design practice. If you use "view source" to look at HTML written by other people, badly indented (or non-indented) code can be very difficulty to read—try it!

In HTML source code, every new block-level container should be indented one increment more than what came before. In CSS source code typically the properties in each declaration block should be indented the same way, and the closing brace "}" of the block should be flush left. Don't know what these terms mean? Don't worry, **just copy the indenting that is shown** in the code examples from the InterACT book and you will be following good indenting practices.

32. Review your code and ensure that there are at least 3 **meaningful comments** in both the CSS and the HTML files (That is, at least 3 comments in each file). For an explanation of a "meaningful comment" look at part B15. Note that CSS uses a different comment syntax than HTML. (See InterACT Ch. 11, p. 178.) Don't put a CSS comment in an HTML file or vice versa, or it won't work.

Image credit: http://www.guzer.com/pictures/ninja_kitty_kicks_dog.php

PART D: Forms & Embeds

OVERVIEW: In this section you will make your Web pages more useful by adding fill-in forms that send data somewhere. You'll also add Web pages within Web pages.

TIP: You'll need to refer to Ch. 7 of the Duckett book for forms to make sense.

Send a Form to a Computer Program in Finland

1. **Open your existing file page1.html** in Notepad++ or your editor.

2. Let's say that you'd like to make your Web pages more interactive – for example, you'd like your page's users to be able to easily send you a message from a comment form. Fill-in forms are simply special HTML tags that are enclosed by a <form> block. **Add the <form> block** at the bottom of your page's HTML.

3. HTML forms direct the Web browser to send information to a computer program that you specify. This program could be on any Web server. Fourteen years ago, a college student in Finland named Jukka wrote a very simple computer program named "echo" to repeat back any information you sent it. For reasons unknown, they never turned it off after he graduated. So to learn about HTML forms, we'll send our form values to his computer program in Finland. **Set the action attribute** of the <form> tag to:

    http://jkorpela.fi/cgi-bin/echo.cgi

This tells the Web browser to send the form data to Finland when it is submitted. Next, **set the method attribute** to

    POST

4. **Add the following tags** to your form, specifying the attributes for the tag that make sense. (If you aren't sure, refer to the examples in the Duckett reading.) Add regular text before, after or in-between these tags to make a feedback form for your Web page that makes sense. For instance, you might ask people their name, to check a box if they liked the story, and to leave a longer message. These tags are sometimes called **form controls**. You'll need to add ordinary text around the form tags (like "Your name:") for the form to make sense to the user.

- An <input> tag with the type attribute set to "**text**"
- An <input> tag with the type attribute set to "**checkbox**"
- A <**textarea**> tag
- An <input> tag with the type attribute set to "**submit**"

All of these tags should have the **name attribute set** to values of your choice that make sense for each field. For instance, if you had an <input> tag of type "text" to store someone's phone number, you might want to set the name attribute to "phonenumber".

5. Be sure to save your work if you haven't already. If you have correctly formatted your tags, when you **load this page in a Web browser** you will see a fill-in form. If you (acting as the user) then type data into the form and **click "submit"** the Web browser should send this data to Jukka's "echo" program in Finland. Take a screen shot of the result and include it in your lab report. (You can use your operating systems screen shot command, or take a picture with your phone.)

6. **Change the method attribute** of the <form> tag from POST to GET. This controls how the data is sent to Jukka's computer program in Finland. Save and submit the form. Look at the URL in the Web browser window. What do you notice? Write your answer in your lab report now. Then **REVERSE THIS CHANGE** and set the method attribute of the <form> tag **back to POST**.

PITFALL: When you modify the HTML for your fill-in form, you need to save the HTML file and re-load it in your browser for the changes to appear. Just submitting the form again without re-loading it will submit an earlier version of your HTML.

7. Modify your form to add an <input> tag with the type **attribute set to "hidden"**, and the name attribute set to a sentence of text (of your choice). Save and submit the form. Notice that hidden data is not shown to the user on the Web page, but it is still in the HTML source code and it is still sent to Finland.

8. Add an <input> tag with the type **attribute set to "file"** and a name attribute of your choice. Save and submit the form (meaning, upload a file – try using a small file). What does the data with that name attribute look like when it is returned by Jukka's computer program? Write the answer in your lab report, then REMOVE THIS TAG.

PITFALL: Be sure the method attribute of the <form> tag is set to POST (from step 6).

9. Make your comment form longer and turn it into **a mini-survey** by adding the following controls, with the appropriate text so that the form will make sense for the user.

- Add a **drop-down list box** using <select> and <option> with a default value specified with "selected"
- Add a **forced choice** between at least two things using <input> type "**radio**" and specifying a default value with "checked"
- **Group** form controls into a <fieldset> with a visible <legend> (it is your choice which form controls to group)
- Add <**label**> tags for *every* control

9. **Save** and **submit** the completed form. **Take a screenshot** of the result and include it in your lab report.

Use a program that E-Mails form data to yourself.

10. HTML fill-in forms send their data to a computer program. Now let's send this form data to a different computer program that will e-mail you the results of this form. This is a computer program that runs on the www.umich.edu Web server. It is called **htmail**. First, change the **action** attribute of your <form> tag to send the data to a different computer program, **htmail**. Use this address:

http://www.umich.edu/cgi-bin/htmail/**uniqname**@umich.edu

(but replace "**uniqname**" with **your UM uniqname**.)

11. If we allowed htmail to send email to anyone, spammers would probably use it. So htmail needs to be **authorized** for your e-mail address before it will send mail for you. **Visit**:

https://webapps.itcs.umich.edu/htmail-registration/

(UM login required.) **Click the box** next to your uniqname, then press "**Make Changes**".

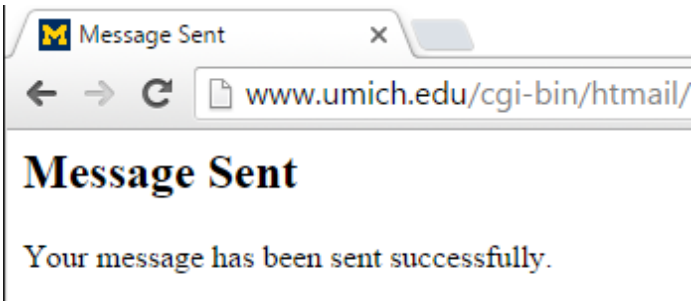
12. We're almost there, but unlike Juka's echo program in Finland, which will *always* try to echo what you send it, the htmail program has a *validation rule*. **The validation rule is**:

- **All submitted forms must have one <input> tag of type "text" with its name attribute set to "from"** so that the user can type an e-mail address to be used on the "From:" line of the e-mail that htmail sends. ***If you don't recall how to do this, refer back to **Step 4** of **Part D**.***

*Try submitting the form without the required field. What happens? Answer in your lab report.

(Image source: K@'s Smiley Gallery)

13. Correct the form to meet the above validation rule and submit it. If it worked, you should receive the message:



PITFALL: If instead of this message you were directed *to a different feedback form*, you didn't set the <form> tag's method attribute back to "post" – this was part of step 6 above.

14. **Check your e-mail** for the message from htmail – it may take a minute or two to arrive, but no longer than that. If it still hasn't arrived, check your spam folder.

15. That e-mail is probably pretty ugly. Can you make it look more like a regular e-mail? htmail looks for certain named pieces of data like "from" (discussed above in **Step 12**) and uses them to format the e-mail like the ones you receive in your inbox (subject, sender, preview of the body). You can send these special pieces of data to htmail by making sure there are form elements that have the right "name" attribute set to these values. In addition to "from", **add <input> tags with name attributes of**:

- subject
- body
- successURL

Use an appropriate form element (your choice) to contain this information. **Hide at least one** of these from the user. (For instance, they probably don't want to see successURL printed on the form.)

TIP: You can switch your <form> tag's "action" attribute back to the echo script in Finland if you want to quickly look at what data your form is sending – e.g., to be sure it is working right.

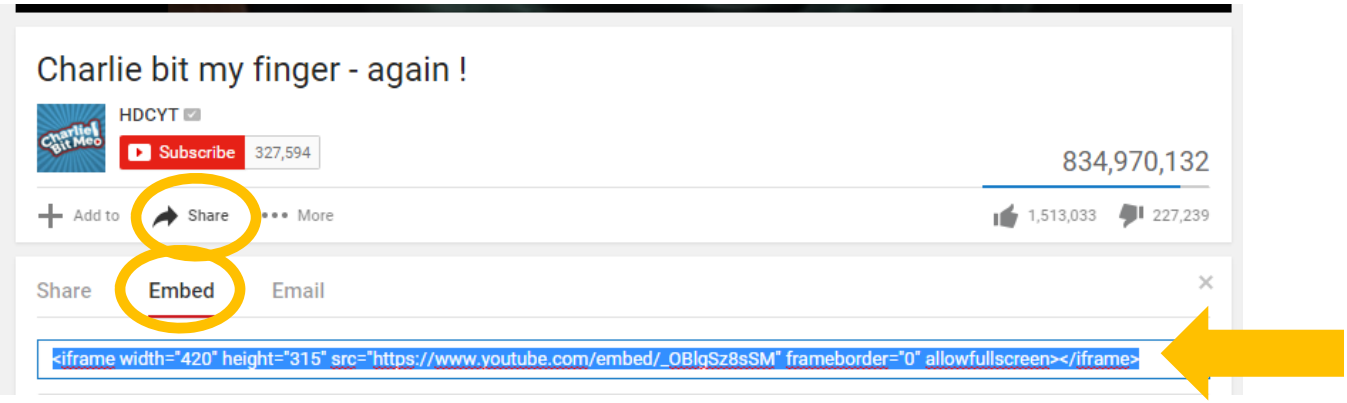
16. Save your work. Reload and submit your form, wait for your email to arrive, and **take a screen shot of your e-mail** from within your e-mail reading program (probably gmail) and paste it into your lab report.

Embed a Web Page Inside Another Web Page.

17. We already know that Web pages can contain images, CSS stylesheets, and other files by referring to them in the right HTML tags. It turns out that Web pages can also include other Web pages. For this final section of Part D, **re-open page2.html in Notepad++** or your alternate editor.

18. Next, visit http://www.youtube.com and find a class-appropriate video that is vaguely related to the content you chose to include in this file.

19. Underneath the video, click "Share" and then "Embed". This should reveal a form element of type="text" that contains some code that you can copy and paste. It is a tag called <iframe> which embeds one HTML page inside another. **Copy the <iframe> tag from the YouTube page and paste it into page2.html**.



20. **Save the file and preview it** in the Web browser. As long as you can see the YouTube player and the preview of the video in your web page, it worked! Optionally, YouTube (and other platforms) will also edit the tag for you to make adding YouTube videos to your Web pages easier and to give you more control over this process. To see the options, **click the "SHOW MORE" button** underneath the embed code you just copied and pasted. You could also make these changes yourself by editing the HTML directly.

PITFALL: As long as you can see the YouTube player and the preview of the video in your web page, it worked! If you actually click on the YouTube video, ***some*** videos may produce the message "**video unavailable**." That because YouTube prohibits some videos from playing on "local" Web pages – meaning on your own machine when you may not be connected to the Internet. This is to keep you from stealing free music with <iframe> tags. If you upload this file to a Web server (the instructions are in Part Z) the video should play.

Part E: On Your Own

OVERVIEW: This section of the assignment gives you the most freedom. You'll be creating an HTML web page and CSS stylesheet based upon whatever you want.

The Web pages we've had you make so far are **pretty ugly**. Most professional web designers use layout, images, and typography to create a compelling style for their users that should match the content in some way. For this final page, visit Web pages that you like and try to figure out how you can also create a compelling and professional Web design on this page.

Specifically, your new HTML page must include the following:

- ❑ A <link> reference to a **new external CSS stylesheet** distinct from the one you wrote in Part C. (Start it from scratch.)
- ❑ At least **one relevant image**.
- ❑ **Headings** of at least two different levels
- ❑ Thoughtful use of **white space** as described in Ch. 14.
- ❑ The line break tag **
**
- ❑ A with a **background color** different than that of the rest of the body
- ❑ A **link back to the some other page** you created for this assignment (from Part B, or a new HTML page if you want to make two of them for this part)
- ❑ Meaningful **comments** in both the HTML and CSS documents
- ❑ Use **FIVE new HTML tags OR CSS attributes** (in the HTML tags or the referenced CSS file) that were not covered in this assignment so far. Browse the InterACT readings and Web pages like http://w3schools.com/html/ and http://w3schools.com/css/ for ideas. For example, you can experiment with a background image, shadows, indented quotes, floats, kerning/tracking, line height, link hover effects, etc.
    - ❑ 1. _____
    - ❑ 2. _____
    - ❑ 3. _____
    - ❑ 4. _____
    - ❑ 5. _____

    (you don't need to turn this page in; these blanks are just notes for yourself.)
- ❑ In your lab report, **state the five HTML tags or CSS attributes** that you used. For example, you could write "This page uses the new CSS attributes smell-style, touch-width, electron-weight, and bit-recycling, as well as the new BOGUS html tag." (These example tags and attributes are not real.)
- ❑ Write HTML and CSS rules that are **semantically correct**. That is, put layout in CSS rules as much as possible and always use the most appropriate HTML tag that describes the meaning of the content it contains. (See InterACT Ch. 10, pp. 164-165.)

In addition, please modify the first page that you created in Part B so that it includes:

- ❑ **A link to every other HTML page you have created** for this assignment. (Your three scenes, as well as this new HTML page.) In other words, if an instructor looks at the first HTML page that you created, they should be able to follow a link from there to any of the other pages you've made.

PART Z. Host Your Web Pages Publicly

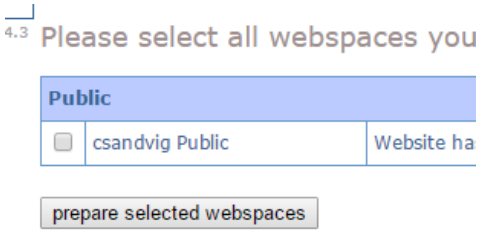
OVERVIEW: You've made some Web pages, but no one can see them because they are located on your own computer (or the lab computer). In this section, you'll upload copies of the files you created to a public Web server so that anyone on the Internet can see them.

IMPORTANT: Be sure to ***do this part of the assignment last***, or be sure to repeat step 8 (below) if you make changes to your files after this point so the changed files are made public.

.

1. If you have never used UM's AFS space to publish Web pages, first **request AFS space** and then visit the UM "**make Web space**" tool. (Not sure? If there are no public Web pages at: http://www-personal.umich.edu/~uniqname – where "uniqname" is replaced by your UM uniqname, you need to use this tool.) First, go to the URL: https://ifsprovisioning.its.umich.edu/ifs_storage/request and request AFS space. It might say you already have AFS space and if so, that's fine – proceed to the next step.

2. If you've never used AFS before you may need to wait 20 minutes before the next step will work! (This is not our fault!) Next, go to the URL: https://mfile.umich.edu/make-webspace/ (you'll need to login for both of these.)In the "Public" box, click the checkbox next to your uniqname (followed by "Public") then click **"prepare selected webspaces**." Wow, you have a public Web site! If you get an error message about a missing home directory, wait 20 minutes and try again.



3. **MFile is a way to access disk space** on a public Web server provided free to all UM students. This is called your AFS space. Visit MFile (you'll need to login) at: https://mfile.umich.edu/

4. **Navigate to the folder** that should hold your Web pages by clicking "Public" then "html".

5. On the left sidebar, click **create a new folder** and title the folder "COMM362" (without the quotes), then press "CREATE".

6. On the left sidebar, click "Upload Files" and **upload ALL of the HTML, CSS, and image files** you have created/saved for this assignment (not your lab report). You can use the checklist on the front of this assignment PDF to be sure you get them all.
   - You'll need to click "**Add Another File**" to add each file after the first. Sorry.
   - You'll need to **click** "**Overwrite** files during upload?" because your html folder comes with an index.html file already in it, and you made a file

7. You have made these files public at the URL:
   http://www-personal.umich.edu/~uniqname/COMM362/

where "uniqname" should be replaced with your UM uniqname. Type this URL in your browser and **confirm that you can see your Web pages**, that the links work, and that the images appear correctly.

PITFALL: If you see a weird page that says "Index of /~uniqname/COMM362" at the top, you didn't upload your index.html file yet, or you made a mistake while naming it.

PITFALL: If your links or images don't work, you probably need to review the material about relative links back in C5. That is likely the error.

8. If you make changes to any of your CSS or HTML files on your own computer (for instance, to correct a mistake), **upload each changed file again** by repeating steps 3-6 above, ensuring that "Overwrite files during upload?" is checked. This will make the revisions public on the Web server.

9. Yay! You did it! Now it's time to write up your lab report. When you're finished, refer to the section "**Turn it in**" at the beginning of this document for a checklist of the files you need to turn in. Unlike last time, for this assignment you'll upload **ONE** compressed folder to with all of your files.

Compressing Your Folder:

Windows Users:

1. Locate the folder that you want to zip using the Windows Explorer or search bar.
2. Right-click the file or folder, select "Send to," and then select "Compressed (zipped) folder." A new zipped folder with the same name is created in the same location.

Mac Users:

1. Locate the folder that you want to zip in Finder.
2. Right-click or command-click on the file to bring up the pop-up menu, then select Compress *filename*.
3. The Mac will begin to compress the file or folder you've selected. Once it's done, you'll find a filename that ends in .zip right next to the file you selected.

</assignment>

***BONUS WORK**:*

Lab assignment 2 is over, but if you are interested in learning more about Web pages, or the assignment was too easy for you, you can receive extra credit for *bonus work*. For this assignment, there are a few options for bonus work. You can decide how to demonstrate you have completed the bonus work. For instance, you might **share a URL, upload an additional HTML file, add a screenshot,** and/or **write a very short note about what you did** at the end of your lab report.

For this assignment each bonus work can receive up to the amount indicated of extra credit on this assignment, to a maximum of +40%. Here are the options:

- **More Tags**. Use the HTML and CSS reference documents on the resources page to teach yourself additional HTML tags and CSS elements that are not covered in this assignment and demonstrate their correct use (up to +2% each)

- **Hack Weebly**. In this assignment you created your own HTML and CSS files and hosted them at the UM, but you can use these same skills to edit the files on a commercial Web hosting provider. Make a new account at http://www.weebly.com (free), use the user-friendly Web-based tools to fill it with some content of your choice (at least one page), then use "My Site" > "Theme" (at the top) > "Edit HTML/CSS" (at the bottom of the left sidebar) to customize the site with your HTML and CSS skills. (up to +20%)

- **Explain Wikipedia**. In your own words, answer the question posed after A10: Explain how the stylesheets work on a Wikipedia page. YOU MAY FIND THIS HARD. (+5%)

***IMPORTANT**: If you are submitting bonus work, be sure you make a note of this on the bottom of your lab report so that you receive credit for it.*

tl;dr