HTML/CSS EXERCISE

Follow these steps to create an HTML page based on the Tattoo Picture Story. Complete the exercise, compress (zip) your folder and submit to Canvas.

PREP TIME (10 MINS.)

1. Download the **html-css-exercise** compressed folder from Canvas > Files > Course Material. Move the compressed folder from Downloads to your Desktop (for easy access).

2. VERY IMPORTANT!! Unzip the folder by double-clicking. Remove the compressed folder.

3. Review the contents of the exercise folder: it includes an *index.html* document, an images folder and a folder for your CSS, which will specify the presentation aspects of your page. You can find text files for today's exercise in the Resources folder: captions and story, plus a CSS reset.

4. Go to **atom.io** and download the code-editing software we will be using. After de-compressing this folder, you will see the Atom application. For convenience, move it to your Applications folder.

5 . Launch Atom. Close any windows that appear by clicking in the small "X" on the right of the tabs.

6. Under the Atom menu at top left, pull down to "Preferences." Click the Editor tab, and scroll down to "Soft Wrap." Click the box so that a blue check is showing. This will allow you to see all your text within the application window. You can now close the Settings tab at the top of the window.

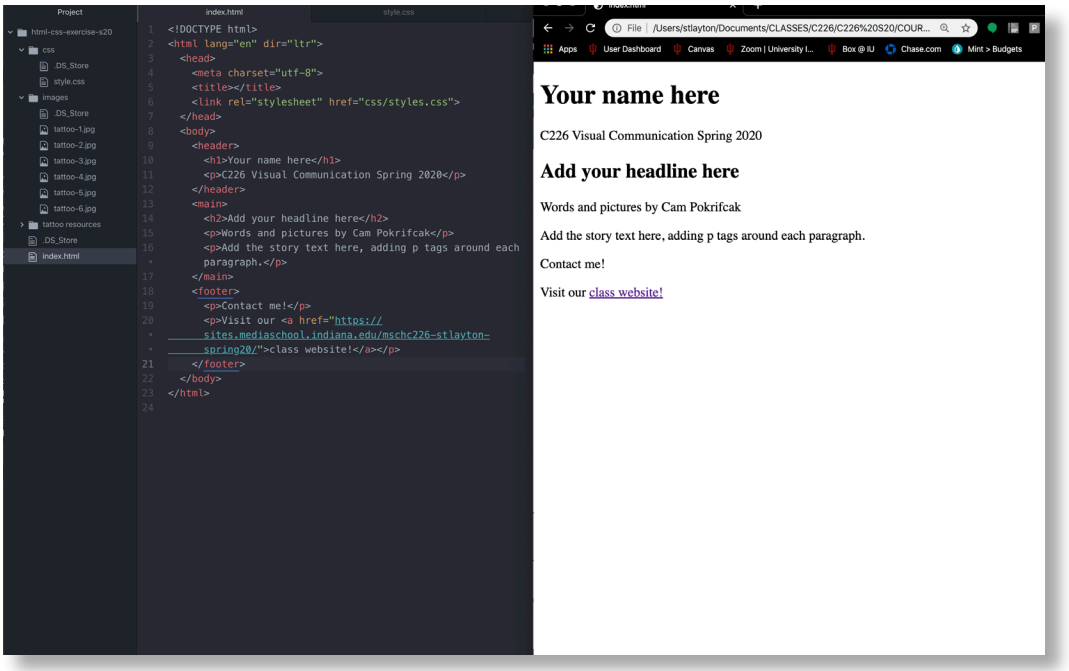
7. Go to File > Open, and navigate to the **html-css-exercise** folder. You should see the files in the Project sidebar.

MAKE SURE you have NOT opened the zip file!

8. Make the Atom program take up about half your screen, and launch a browser (like Chrome) on the other side.

9. Click on the *index.html* file in Atom's project sidebar, and drag the tab from Atom to the tab on your browser. The *index.html* file will display there.

The configuration should look like what you see below:



10. Every time you make a change to your code, save the file in Atom (Command-S), and then reload the page (Command-R) in the browser to see the effects of these changes.

ADDING YOUR CONTENT

1. Within the <title> tag, add your name and "C226".

```
<title>Ernie Pyle C226</title>
```

2. Inside the <h1> tag, write your name.

```
<h1>Ernie Pyle</h1>
```

3. Inside the <h2> tag, write this headline, replacing "Add your headline here."

```
<h2>Welcome to my life, tattoo</h2>
```

4. Inside the first <p> tag, add your name instead of "Yourname Here*".

```
<p>Words and pictures by Ernie Pyle</p>
```

5. Open the "Resources" folder and copy (Command-C) all the story text for the Tattoo story.

6. Back in *index.html*, swipe over the line of text in the second <p> tag, and paste your copied text (Command-V).

7. Each of the paragraphs in the story should be contained inside a <p> tag. Add opening and closing paragraph tags to each of your paragraphs.

```
<p>Tattooing is an art that has been
practiced for a very long time. While the
meaning and purpose differentiates across
cultures, tattoos in the United States
have recently gravitated from a form of
rebellion to a form of self-expression.</p>
```

8. We will now add an image. Images are added to HTML via the <img> tag. This is an *empty* tag, meaning it cannot contain content. It calls in the image to the page as an attribute, telling the browser specifically which image we want to display. Return to the first line after the <main> tag (line 14) and hit a Return.

9. On Line 14, add the first image:

```
<img src="images/tattoo-1.jpg">
```

This tells the browser to find the first tattoo image inside the *images* folder and display it.

10. Next we'll add a caption to this image. For this, we will wrap our <img> tag with a <figure> tag — it will open before the <img> tag and close after it. Before the <figure> closes, we will add a <figcaption> tag, which will contain the text of the caption.
Here is how this chunk of code will look:

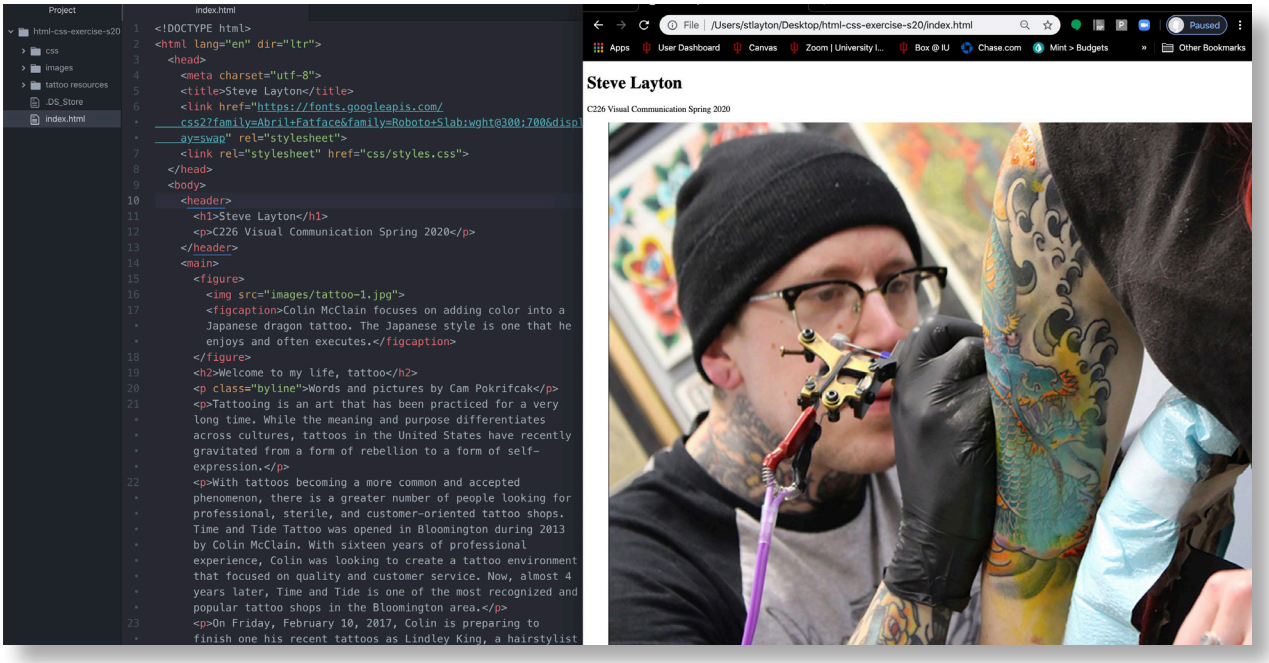
```
<figure>
    <img src="images/tattoo-1.jpg">
    <figcaption>Write the text for your
    caption here.</figcaption>
</figure>
```

11. At the bottom of your HTML file, find the <footer> tag. It includes two <p> tags: one to allow a user to email you, the second to allow the user to see our class website. For these to actually do something when they are clicked on, we will add an <a> tag (a hyperlink) within the <p> tag. Of course, we will have to tell the browser *where* we want to go, too. We do this through an **href** attribute inside the <a> tag.

```
<p>Visit our <a href="https://sites.
mediaschool.indiana.edu/mschc226-stlayton-
fall20">class website</a>!</p>
```

12. You will build the hyperlink to your email in a similar way, but instead of telling the browser to go to a website, you will be telling it to access your mail program.

```
<p><a href="mailto:youremailhere@indiana.
edu">Contact me!</a></p>
```

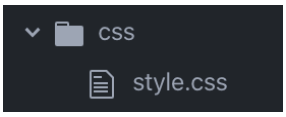


**This story is not by Ernie Pyle, actually, but by Cam Pokrifcak, a C226 student in 2017.*

STYLING YOUR CONTENT

Before we begin adding CSS code, we must first make sure the CSS file and the HTML file are correctly linked. In Line 6 of index.html, you will find the link tag that connects these two documents.

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



This link is directing the browser to search inside the **css** folder for a file called **styles.css**. But the file that is inside your css folder is called **style.css**! In order to correctly link the two files, the names *must match exactly*. Change your HTML code so that the file you are linking to is called **style.css**.

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

Save your index.html file and reload your page in the browser window. You should notice some changes — all the type is the same size and the space between paragraphs is gone. Don't worry — this is actually an indication that the code is working! You are now ready to start adding some of your own style to the design.

1. Go to the ***style.css*** file aside the css folder. It already includes a reset, which strips away most browser defaults from the presentation. After the code ends, on line 48, hit a return a couple of times. We will add our custom CSS styles here.

2. Start by changing the background color of the page. To do this, we will apply a background-color rule to the <html> element.

```
html {
    background-color: peachpuff;
}
```

3. Reload your page. You should see the color of the background change from white to … peachpuff.

4. Next we will add some structure. We will set a width for the <main> element, as well as adding a margin rule to keep it centered in the browser window.

```
main {
    width: 800px;
    margin: 0 auto;
}
```

5. So that we can read the text easier, we will add some space beneath each paragraph, using a variation of the margin rule.

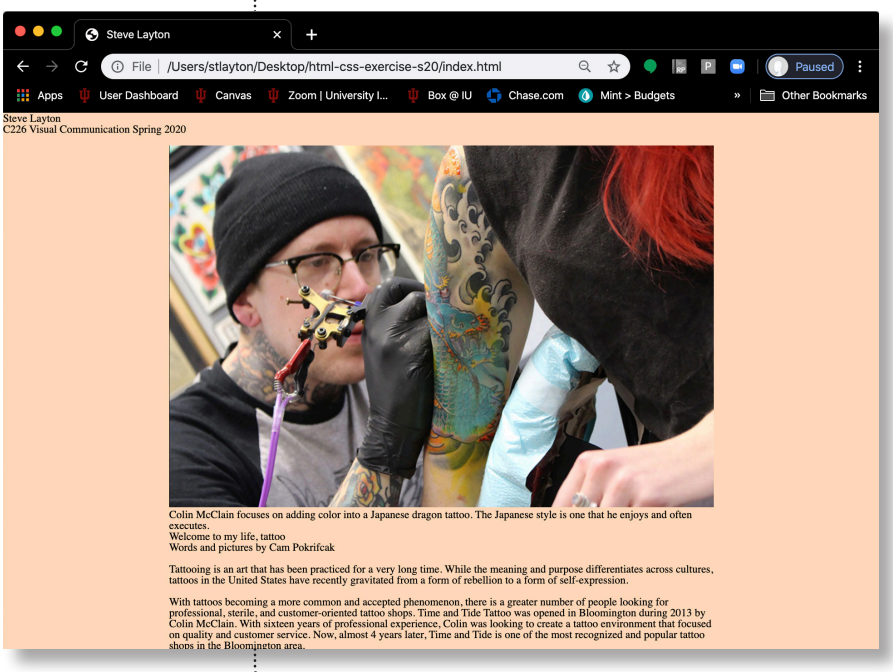
```
p {
    margin-bottom: 16px;
}
```

Notice that your photograph is too big! It is displaying on the page at its actual pixel size — the size, in other words, of the image in pixels.

6. We will next write a rule so that the <img> element within the <figure> is the same width as the main (800px). Because the <figure> is within the <main>, the <figure> element is actually 800 pixels wide — but the <img>, by default, overflows this limit. This makes the image appear too large on our page. We will use a percentage to change this.

```
figure img {
    width: 100%;
}
```

The code above has a *descendant* selector — it states that any **img** element that is contained within a **figure** will be shown at the same width (100%) as the figure.



7. Let's try changing our typography to better reflect our content. To do this, we will be using Google fonts. Open a new tab on the browser side of your screen, and go to

https://fonts.google.com/

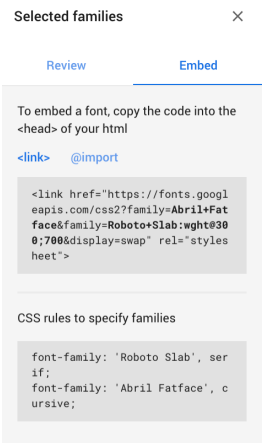
8. Scroll through the available font families — there are more than 900 of them! Find one you like for the body copy, and a more expressive one for your headline.

Light 300

Almost before we knew it, we had left the ground.

+ Select this style

9. Click the typeface to review it, and if you would like to use it, click the blue "Select this font" button. It is now added to your collection in the sidebar at the right of the screen.



10. When you have chosen two fonts — one for body, one for display — click the "Embed" button at the top of the sidebar. This panel includes two vital pieces of info for your code.

11. Swipe over an copy the <link> tag and paste it into your index.html document, just after the <title> tag (line 6).

12. Back in Google Fonts, copy the font-family rule for the body copy font.

13. In your CSS file, add a rule for the <body> tag. Paste the copied rule inside the curly braces to make all the text on your page appear in this font. As well, I have added two additional declarations for the body rule: changing the size of the text to 18 pixels (the default is 16) and opening up the line-height — the CSS equivalent of leading — loosening it from the default (which is 1 — note there is not a unit for this).

```
body {
    font-family: 'Roboto Slab', serif;
    font-size: 18px;
    line-height: 1.5;
}
```

14. Repeat this process for your headline. This time, create a *multiple selector* for both the <h1> and <h2> elements, with the selectors separated by commas. I have also added a (doubled) size using **em** units.

```
h1,h2 {
    font-family: 'Abril Fatface',cursive;
    font-size: 2em;
}
```

15. The <h2> element should appear as larger text, since it spans the actual story. We'll add a rule to make this the case. Make sure this rule appears after the **h1,h2** rule we just wrote — in CSS, order is important, and the last declared style is what the browser will use.

```
h2 {
    font-size: 3em;
}
```

16. Next, we will center-align some of our content. We will write a rule for the header, footer and h2 elements to center them within the layout.

```
header, footer, h2 {
    text-align: center;
}
```

17. I'd like to differentiate the caption and byline so that they will not be confused with story text. The caption has a different HTML tag — **figcaption** — so writing a rule for that is fairly straightforward:

```
figcaption {
    font-style: italic;
}
```

The byline, though, is a <p> tag, so I need a way to single it out from my other paragraphs. You can do this in HTML/CSS by giving the element a *class*.

18. Return to the **index.html** document and add a class attribute to the opening <p> tag for the byline.

```
<p class="byline">Words and pictures by Cam Pokrifcak</p>
```

You can name the class whatever you want, but do not use spaces or capital letters.

19. Back in CSS, create a rule for all paragraphs that have a class of "byline." You reference this with a dot (.) in your selector. No space between the p and the dot!

```
p.byline {
    text-align: center;
    font-weight: bold;
}
```

20. Finally, we will change the color of our hyperlinks, which for this page appear inside the footer element at the bottom. We'll write a pair of rules, changing the appearance of any hyperlinks in normal state and, by using the **:hover** pseudoclass, darkening the red color when the user hovers over the link.

```
a {
    color: red;
}
a:hover {
    color: darkred;
}
```

SUBMITTING THIS EXERCISE

Since we have been saving all our work as we add code, we can now simply quit the Atom program. Our last step to submit this exercise to Canvas will be to *compress* our folder, in the same way that we compressed our packaged folder for the Magazine Design project.

Locate your folder for this project, and right-click. Select “Compress” from the options that pop up. Make absolutely sure that you are submitting your code and not the basic code we started this exercise with! This is one reason why it’s a good idea to trash the original compressed folder once you have de-compressed it.

Canvas only allows you to submit files rather than folders, which is why compressing is necessary.