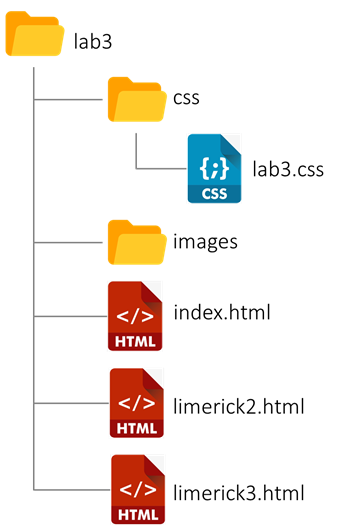
Some things you will want to know for Lab 7.

## How to build a multiple page website.

Each page of a multiple page website is a separate HTML file. Your home page will be placed in a file called index.html. Each of your other pages will be placed in HTML files with a differently named prefix. All three HTML files will go directly into your lab3 folder.

You will build out your CSS in a single lab3.css file that will be linked to all three HTML files. The single CSS file is placed into a css subfolder of lab3.

Your images for all three HTML files will be placed in a single images subfolder of lab3. The specific images files are not shown at right.

By placing your CSS for all of your HTML pages into a single CSS file, you gain several advantages

1. One CSS file means you are doing less work.
2. One CSS file means you only need to make changes in one place.
3. One CSS file means your presentation is more likely consistent across all the pages of your site.

PRO TIP: Build your index.html and get it working. Build out the CSS for your index.html. Go through your debugging and validation process so you know you will make minimal, if any, changes to the index.html going forward.

Then use the index.html file as a template to create your other two HTML files (that is: copy it and save as new name). Building your site this way will reduce your overall work (that is: it will take you less time), and it will make all three of your pages more consistent in look and feel for your user.

## How to set up a menu for a multiple page website.

Inside the body of your HTML file, the menu will be a nav element. For a simple menu, anchor elements can be placed directly inside the nav. *If you look up menu tutorials, almost all of them will place the anchors inside of list-items in an unordered list. That works, but I think it is overkill in HTML5.*

<nav >

<a href="index.html" -data-current>My very first Limerick</a>

<a href="limerick2.html">This one is better</a>

<a href="limerick3.html" class="active">My best poem yet!</a>

</nav>

The menu is identical in all of your html files, except for the -data-current attribute. Apply that attribute only to the link for the HTML file you are currently in. So, above, we must be editing the limerick3.html file.

Then you can use the attribute to style the menu button slightly differently for the page the user is in.

## How to write CSS for menu buttons.

Assuming you use HTML similar to mine above, it might look like this:

nav {

display: flex;

flex-flow: wrap;

justify-content: center;

}

The <nav> is probably a flex-item of your full page flexbox (the CSS for its flex-item properties are not shown here), but it is also a flex-container for the three menu items (and that is shown here.) So, it is set to display:flex, making it a flex-container. Then you decide whether to use flex-direction row or column. I am using row (not stated as row is the default value). Since I am using flex-direction: row, I need to instruct that the rows will wrap (nowrap is the default). If I was using flex-direction: column, then I would not need to make them wrap. The width of the <nav> will be managed as part of the <nav> being a flex-item of the page level flexbox. I wanted my menu items centered, so I used justify-content. But note if you set flex-direction to column, you need to use align-items: to center the menu items.

Add additional CSS for colors and borders, etc.

a {

flex: 0 1 60%; /\* 60% may work for vertical, but not horizontal\*/

height: 3em;

text-align: center;

line-height: 3em;

border: solid .125em gray;

margin: .5em 0;

}

Each menu button is a flex-item of the <nav> flex-container. The CSS likely includes these properties. Flex: is the compound property that flex-grow, flex-shrink, and flex-basis (the width). Given I set flex-direction: row, I want to make sure only one button shows up on a row. So, I need to make the width of each button more than 50% of the width of the <nav> and then each button will wrap to its own row. Visually, I decided I didn’t want the buttons to be 100 percent of the <nav>, but perhaps you do. I turned flex-grow off, as when it is on flex-box expands each button to 100 percent width, no matter what the flex-basis is. In this instance, flex-shrink is doing nothing and it doesn’t matter what I set it to.

I had to manually set the height of each button.

Text-align will horizontally center the text in each button.

There are several methods for vertically centering text. The one I like for menu buttons, where the text won’t wrap is to apply a line-height that is the same as the height of the button. It \*magically\* vertically centers the text. It won’t work if there are multiple lines of text.

I used border to create the effect of this looking like a button. [More on this next class.] and I added a vertical margin to separate the buttons so they don’t touch.