**HTML Me Someting, Part 1**

In Part 1, you will get comfortable with writing markup, and with separating *content* from *design and layout*.

**Getting Started**

Stub out **index.html** with these basic elements:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | ***<!DOCTYPE html>***  <**html**>  <**head**>  <**title**></**title**>  </**head**>  <**body**></**body**>  </**html**> |

Fill each element with some appropriate content. You can start by removing the snippet **<p>YOUR NAME</p>** that you used during setup.

**Getting to Work**

Your mission is to build a page that:

1. Tells a story. This can be personal or impersonal, funny, serious or neither. You can do whatever you like, but it should be something in the range of 3-10 paragraphs or sections. [Here is an example](http://education.launchcode.org/html-me-something/submissions/chrisbay/index-nocss.html), and here are some other ideas:
   1. Create a résumé page that tells the story of your professional journey to-date, and where you want to go as a coder.
   2. Describe a trip you took.
   3. Talk about one of your hobbies or passions.
2. Does each of the following:
   1. Uses each of the following structural HTML5 tags: **<p>**, **<header>**, **<footer>**, **<main>**, **<article>**. If you need to review any of these tags, check out the [HTML tag reference](http://www.w3schools.com/tags/default.asp) at w3schools.
   2. Uses at least one **<img>** tag (hopefully more). When placing images in your page, put them in a new subfolder called **images** within your **html-me-something** directory.
   3. Uses at least one [HTML entity](http://www.w3schools.com/html/html_entities.asp). Hint: putting a copyright notice in your footer will afford you the opportunity to use **&copy;**, but you should also try to get creative here.
   4. Demonstrates creativity. Don’t stop with these items or tags. Have some ideas for your page, and make it great. And dig into the [w3schools HTML reference](http://www.w3schools.com/tags/default.asp) to learn more about other tags, their usage and attributes!

**Notes and Tips**

1. Use your browser developer tools to look at [the example page](http://education.launchcode.org/html-me-something/submissions/chrisbay/index-nocss.html), or to troubleshoot things that don’t look right. You can mimic the document structure of this example, but do NOT just copy/paste!
2. Use the example to learn how your HTML elements might be structured, and build your page to fit your own content.
3. Don’t add any CSS yet. Really, we mean it! If you think your page looks boring now, that’s okay. We’ll get there soon enough.
4. As you make changes, you will obviously want to see the results. To do so, re-save the file in your text editor, then click Refresh in the browser window (or use cmd+R on a Mac, ctrl+R on Windows).
5. Rely on the reference sites linked on this page, or find others online. We haven’t taught you every detail about every tag that you may want or need.
6. You’re free to use tags that haven’t been explicitly introduced in class. We’ve given you enough background to get started, and are intentionally leaving some of the learning up to you.

**Add and Commit Your Changes on Git**

Once you finish your page, use Git to **add** and **commit** your **index.html** changes.

*Why commit again?*

The reason is that you added a bunch of new HTML code to your **index.html** file. It is now in a very different state compared to the first time you committed it. See this for yourself by checking the status:

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: index.html

no changes added to commit (use "git add" and/or "git commit -a")

Git tells us:

*I see that you have modified****index.html****. Use****git add****if you want these changes to be included in your next commit.*

The Git workflow more or less comes down to this:

1. Create some initial stuff
2. **init**
3. **add** and **commit**
4. make some changes
5. **add** and **commit**
6. make some more changes
7. **add** and **commit**
8. etc…

The general rule is that *whenever you make any changes, you must add and commit those changes to Git.*

So let’s do that. From within your root **html-me-something/** directory:

$ git add index.html

$ git commit -m "Finished work on HTML page"

You might be wondering: *How do I know when it’s time to pause working and do another commit?*

This is somewhat subjective, and is ultimately up to you. The good habit is to pause and commit any time you reach a natural stopping point or complete a coherent chunk of work.

**Done!**

Well done! Time to [dive into some CSS](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-part2.html).

**HTML Me Something, Part 2**

In Part 2, you’ll get comfortable with using CSS selectors and rules to dictate display, while keeping your styles separate from your content.

**Getting Started**

1. Create a file named **styles.css** in your **html-me-something/** directory.
2. *Optional*: Add a normalization stylesheet (follow any of the links in the [Normalization](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-part2.html#normalization) section below). You can either put these normalization rules at the top of your **styles.css**, or you can add another file in the same directory and link to it in your HTML doc. This will reset some of your browser’s built-in styles so that you start with a cleaner slate when you add your own.

**Getting to Work**

Go ahead and start adding styles in your **styles.css** file!

**Requirements**

Your CSS must:

1. Use [margin](http://www.w3schools.com/css/css_margin.asp) and [padding](http://www.w3schools.com/css/css_padding.asp) to space your elements in a visually pleasing way.
2. Use at least one of each of the following types of selectors:
   1. [element](http://www.w3schools.com/cssref/sel_element.asp),
   2. [class](http://www.w3schools.com/cssref/sel_class.asp),
   3. [id](http://www.w3schools.com/cssref/sel_id.asp).
3. Follow these rules:
   1. Avoid adding HTML elements in order to achieve a specific visual effect.
   2. Use document-level and inline styles sparingly, and only when absolutely necessary.
   3. Be creative! Make your page look great, and don’t just settle for checking off the items above. Have a look at [CSS Zen Garden](http://www.csszengarden.com/) for inspiration (use your browser’s [developer tools](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_are_browser_developer_tools) to see how those pages’ styles are built).

**Notes**

1. In order to see any visible change, make sure to link the stylesheet to your main document.
2. Feel free to check out our [styled example](http://education.launchcode.org/html-me-something/submissions/chrisbay/index.html) to see how we did things. Use “View Source” (by right-clicking anywhere on the page and selecting *View Source*).
3. Another thing you might find useful is your browser’s [developer tools](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_are_browser_developer_tools).
4. And be sure to use the *Resources* section below as you go.

**Resources**

**General CSS:**

1. [w3schools CSS Reference](http://www.w3schools.com/css/default.asp)
2. [CSS Zen Garden](http://www.csszengarden.com/)
3. (Advanced) [Specifics on CSS Specificity](https://css-tricks.com/specifics-on-css-specificity/)
4. (Advanced) [Specificity (MDN)](https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity)

**CSS Normalization:**

1. [Eric Meyer’s reset.css](http://meyerweb.com/eric/tools/css/reset/)
2. [normalize.css](http://necolas.github.io/normalize.css/)

**Add and Commit Your Changes on Git**

When you finish making your page look spiffy, take a moment to commit your changes on Git:

$ git add styles.css

$ git commit -m "Added some killer CSS styling"

If you also made tweaks to your **index.html** file, then you need to commit those changes as well (along with a descriptive commit message):

$ git add index.html

$ git commit -m "Changed title from 'My Favorite Puppies' to 'The Objectively Best Puppies'"

Incidentally, this is a good opportunity to address the question: *Why does Git have two separate commands for ``add`` and ``commit`` if I always do them together anyway?*

The answer comes back to the notion of collecting your changes in a “coherent chunk of work” for each commit. The **add**command gives you the opportunity to specify exactly *which* file(s) should be included in the upcoming commit. In the example above, this allowed us to perform two separate commits—each with its own message describing its own chunk of work.

Note that you certainly can **add** multiple files to the same commit. For example, suppose you made changes to both **index.html** and **styles.css**, but those changes are all part of the same unit of work. In that case you would add them both before committing:

$ git add index.html

$ git add styles.css

$ git commit -m "Added puppy image with thick yellow border"

There is a convenient shortcut, **git add .**, for those (frequent) occasions when you want to include *every* changed file without having to type each one individually. The following example is equivalent to the previous one (assuming you only changed **index.html** and **styles.css**):

$ git add .

$ git commit -m "Added puppy image with thick yellow border"

It is usually a good idea to check first (using **git status**) before running **git add .**, so that you don’t mistakenly include unwanted changes.

**Done!**

You are ready to submit! Go back to the [Assignment Page](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-something.html#submitting-your-work) and follow the submission instructions there.