HTML Me Something, Part 2: CSS

‹ Back to Overview page (..)

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. **(This is optional.)** Add a normalization stylesheet (see the **Resources** section below; either of those will work). 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 it in to your HTML doc. This will "reset" some of your browser's built-in (and often unsightly) styles so that you are starting with a cleaner slate when you add your own styles.

Getting to Work

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

Requirements:

Here are some specific requirements you should fulfill:

- 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.
- Use at least one of each of the following types of selectors: **element (http://www.w3schools.com/cssref/sel_element.asp)**, **class (http://www.w3schools.com/cssref/sel_class.asp)**, **id (http://www.w3schools.com/cssref/sel_id.asp)**.
- Don't break these rules:
  - At all costs, avoid adding HTML elements in order to achieve a specific visual effect.
  - Use document-level and inline styles sparingly, and only when absolutely necessary.
- As before, be creative! Make your page look great, and don't 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:

- In order to see any visible change, make sure to link the stylesheet to your main document.

- 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*).
- 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)**.
- And be sure to use the Resources section below as you go.

Resources

General CSS:

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

CSS Normalization:

- **Eric Meyer's reset.css (http://meyerweb.com/eric/tools/css/reset/)**
- **normalize.css (http://necolas.github.io/normalize.css/)**

Add and Commit Your Changes on Git

When you have finished making your page look spiffy, take a moment once again to commit your changes on Git:

```
$ git add styles.css
$ git commit -m "Added some killer CSS styling"
```

If, in the course of editing your styles, you also made some tweaks to your `index.html` file, then you will 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 Object
ively Best Puppies'"
```

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

The answer comes back to that notion of batching your changes together into 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. This opportunity is what allowed us, in the example above, to perform two separate commits, each with its own message pertaining to its own chunk of work.

Note, however, that you certainly can, and often will, `add` multiple files to the same commit. For example, suppose you have made some changes to both files, `index.html` and `styles.css`, but those changes can all reasonably be lumped together as part of the same unit of work. In that case you would simply add them both before committing:

```
$ git add index.html
$ git add styles.css
$ git commit -m "Added puppy image with thick yellow border"
```

Finally, there is a convenient command, `git add .`, for those (frequent) occasions where you just want to include ***every*** changed file without having to type each one out manually. The following example is exactly equivalent to the previous one (assuming you have only changed those two files):

```
$ 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 (../#submitting-your-work)** and follow the submission instructions there.