**Assignment #4: HTML Me Something**

You’ve learned a bit of HTML and some CSS, but you have likely only used it in bits and pieces so far, adding or modifying content in exercises or pre-existing files. Here, you are going to take another step forward by building an entire page from scratch. You will also get some practice using Git.

There are two parts to this exercise, one focused on HTML and another focused on CSS. HTML makes up the *structure and content* of web pages, while CSS dictates the *visual style*.

Best practices dictate that *content* and *style* should be kept as separate as possible. To that end, we will build the HTML portion of our page first, and afterwards we will add a few styles with CSS. We do this to avoid using HTML tags to change the general appearance of our page. For example, what if we want all of our main headings to be *red*? We can either add this style one time in the CSS file, or we must include **style="color:red"** in EVERY h1 tag. Especially for large websites, CSS provides the best place to control the overall appearance of a page.

**Sections:**

1. [Getting Started](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-something.html#getting-started)
2. [Part 1: HTML](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-part1.html#html-me-part1)
3. [Part 2: CSS](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-part2.html#html-me-part2)
4. [Submitting Your Work](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-something.html#submitting-your-work)

**Getting started**

Follow the steps below to create a folder for your project and initialize it as a Git repository:

**Setup the Project**

1. Navigate into the parent folder where you keep all your course materials (e.g. **lc101/** or **code/**). Only you know where that folder lives in your file system, but you want to do something like:
2. $ cd ~/lc101/
3. Make a new folder for this assignment:
4. $ mkdir html-me-something

Your directory structure should now look like the below (or something similar):

lc101/

|

+--- html-me-something/

|

... etc

1. Within your new **html-me-something/** directory, use the **touch** command to create and save a new file called **index.html**:
2. $ touch index.html

**Note**

The filename **index.html** is a standard convention for the name of the root page of a website. Most web servers will treat **index.html** as the default file to load from a given directory.

1. Open up your new file in a text editor. Add a single line with the following HTML:
2. <p>YOUR NAME</p>
3. Save your file.
4. Finally, open up the file in a web browser. You can do this by selecting *File > Open File* in your web browser and navigating to the location of your new HTML file.
   1. If you need help finding where **index.html** is located, use the command **pwd** for “print working directory”:
   2. $ pwd
   3. /FolderName/OtherFolderName/lc101/html-me-something
   4. The output shows the path to your current directory. Follow this path after selecting *File > Open File* to locate and open **index.html**.
   5. Once you open your file, you should see a blank white page with your name in the top-left corner.

**Use Git**

Now let’s incorporate Git into the picture.

1. Initialize the project as a Git repository.

In your terminal, make sure you are inside your **html-me-something** folder, and then use the **git init** command to initialize that folder as a Git repository:

$ pwd

/Users/adalovelace/lc101/html-me-something

$ git init

Initialized empty Git repository in /Users/adalovelace/lc101/html-me-something/.git/

**Note**

Your name is (probably) something other than **adalovelace**.

Now your project is a Git repository, which will enable you to use all of the magic Git powers:

* 1. Version-control to manage your changes.
  2. Syncing your local repository with a remote repository on Github.com.

**Note**

You only have to do the **git init** step once, at the beginning of a project.

1. Check your status.

Back in the terminal, use the **git status** command to check the status of your newly created repo:

$ git status

On branch master

Initial commit

Untracked files:

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

index.html

nothing added to commit but untracked files present (use "git add" to track)

This message says a lot of things, but for now, the most important point is that **index.html** is currently “untracked”. We need to **add** and then **commit** the file so that Git can help us manage its changes.

1. Add your work to the repo.

Use the **git add** command to track your **index.html** file so that it will be staged for your next commit:

$ git add index.html

Now check your status again.

You should see that your change (the creation of the new file) is staged to be committed:

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: index.html

1. You are now ready to **commit** the changes you staged, along with an appropriate message describing what you changed:
2. $ git commit -m "Created index.html file"

Check your status again. Your status should be *clean*:

$ git status

On branch master

nothing to commit, working directory clean

Congrats! You are officially up and running with a version-controlled project.

**Getting to Work**

It’s time to build out your page! Dive into each of the two parts below:

1. [Part 1: HTML](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-part1.html#html-me-part1)
2. [Part 2: CSS](https://education.launchcode.org/intro-to-professional-web-dev/assignments/html-me-part2.html#html-me-part2)

**Submitting your work**

When you are ready to submit, complete the following steps:

**Github**

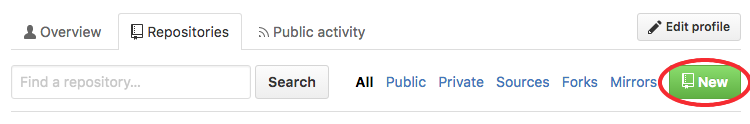
Github.com is a website that hosts Git repositories “in the cloud”. A repository on Github often functions as the central hub for a project, so a developer can do work across multiple machines, or multiple developers can work together on the same project.

For the remainder of this course, you will use Github to submit your work. Here’s how:

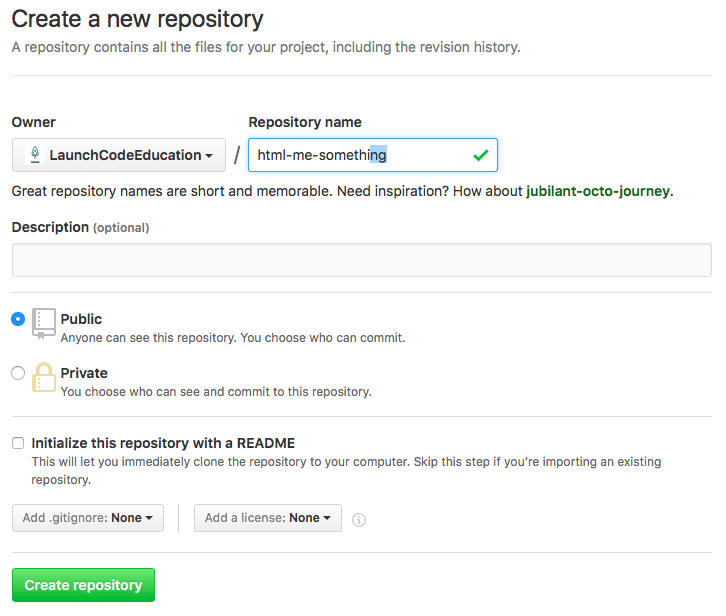
1. Create a repo on Github.

In a browser, visit [Github’s website](http://github.com/" \t "_blank). Make sure you are logged into your account (or create an account if you do not already have one).

On your profile page, create a new repository by clicking the green **New** button on the right side of the screen:



Give your repository the same name as your folder, **html-me-something**, and toggle the rest of the options as specified here:



**Note**

Instead of **LaunchCodeEducation**, you will see your own username.

1. Pair your local repo with your remote repo.

Now you have two repositories: the local one on your computer, and the remote one on Github. You need to sync them.

The first step to syncing is to give your local repo a *reference to* the remote repo. Using the **git remote** command, you can inform your local repo about the existence of the remote one.

* 1. Copy the url for your remote Github repo as follows:



GitHub Clone Url

* 1. Use the command below, *but replace ``PASTE\_REPO\_URL\_HERE`` with the actual url you copied from part a*:

$ git remote add origin PASTE\_REPO\_URL\_HERE

**Note**

Unless you’ve set up an SSH key on your computer and added it to your GitHub account, you should always select the HTTPS version of a repository URL.

By running the **git remote add ...** command on the terminal, you are basically saying:

*“Hey local repo. Please meet your new friend,****origin****, a remote repo, whose url is****https://github.com/...****”*

Note that the name “origin” is simply a standard naming convention for the main remote repo paired with a local repo.

1. Push your local changes up to the remote.

Your local repo is currently *ahead of* your remote repo by a few commits. Locally, you have added and edited a few files, and committed all those changes. However, your remote repo is still entirely empty.

Use the **git push** command to send all your local changes up to the remote:

$ git push origin master

This command means:

*“Hey Git, please push all my local changes to the remote repo called****origin****(specifically, to its****master****branch).”*

Refresh the browser window on your Github page, and notice that your HTML and CSS files have appeared!

**Turning In Your Work**

In Canvas, open the HTML Me Something assignment and click the “Submit” button. An input box will appear.

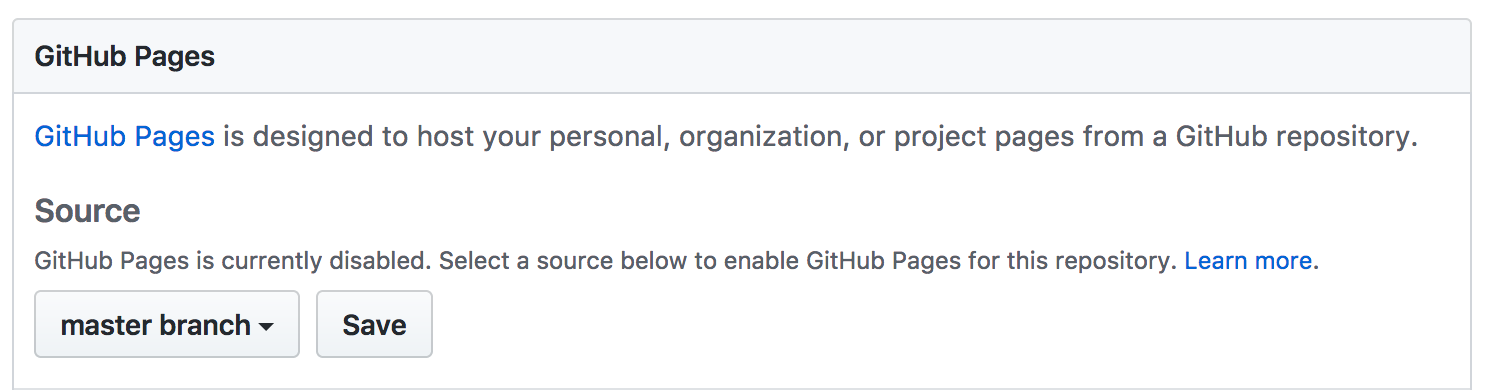
Copy the URL for your GitHub repo and paste it into the box, then click “Submit” again.

**Bonus Mission**

If you want to show off your hard work to all your friends, Github has a cool feature called *Github Pages* that makes this really easy.

Github provides free hosting for any “static” web content (like this project). All you have to do is change a setting on your GitHub repository.

1. In a browser, go to the Github page for your repository.
2. Click on the *Settings* tab
3. Scroll down to the *GitHub Pages* section and enable the GitHub Pages feature by choosing your **master** branch from the dropdown. Hit *Save*.



1. In any browser, you should now be able to visit **YOUR\_USERNAME.github.io/html-me-something** and see your web page!