Building Your First Website

6 users recently completed this Project

Congratulations! You've learned enough HTML and CSS to build your very first website. Although most modern projects use the dense and complicated frameworks and technologies that you'll learn in the program, it's still valuable practice to try doing it the traditional way.

By completing this exercise, you'll practice everything you've learned so far. You'll also end up with something cool that you can put on the internet and show to your friends and family - by sending them a link!

Requirements

You can use the topic, theme, or purpose of your choosing for this project. Make a profile site, fan page for your favorite band, team, or show, a site all about your cat - whatever you want to show off to your friends and family!

Because you'll be putting it on the internet for the world to see, you'll want to pick a topic your comfortable sharing. However, you don't need to worry about a future employer judging you for your code here. Hiring managers absolutely love seeing measures of growth, development, and experimentation in your online profile.

At a minimum, your site must have the elements and features listed below.

You can add to this if you'd like, but you're usually better off starting small and expanding after you have completed the first set of tasks. It's much better to make a small site that is complete than succumb to feature creep and end up with a large site that is unfinished.

- Page 1
  - Navigation Bar linking to all 4 pages
  - 1 header using H1
  - 3 sections of text with H2 headers
  - 1 image
- Page 2
  - Navigation Bar linking to all 4 pages
  - HTML form with a minimum of:
    - 1 fieldset
    - 3 inputs
      - 2 text
      - 1 checkbox
    - 1 textarea
    - Appropriate labels for the above
    - 1 button
- Page 3
  - Navigation Bar linking to all 4 pages
  - 1 header using H1
  - 1 ordered or unordered list of links to 5 other websites
- Page 4
  - Navigation Bar linking to all 4 pages
  - 1 header using H1
  - 6 images displayed 3 wide and 2 down
    - A label for each image
    - Clicking on the image opens the source of that image in a new window

Each page must be appropriately and pleasantly styled using CSS.

Phase 0: Setup

Create a new repository in GitHub. You can go ahead and make it Public. Companies that are hiring a junior developer like to see evidence of growth and development. Later on, you'll work with your career coach to clean up your GitHub materials. For now, go ahead and take this opportunity to get used to the idea of random people taking a look at your code - and keep it clean, neat, and organized!

You do want a README, so check that box. You don't need a .gitignore file yet, or a license. Not choosing a license at all is actually the strongest option - it's assumed you reserve all rights!

Clone your new repo and open it locally. Create a new file called `planning.md` and open it.

Phase 1: Planning

Use the tool of your choice to draw the layout of each of your pages. Once you're satisfied with your design, go through and mark the appropriate HTML tags for each item on each page.

At this point, make decisions for the look and feel of your site. Select at most two different fonts - one for text and one for headers. Pick your color palette and write down hex codes. If you aren't very artistic, there are lots of tools on the internet that can help with this! Just search for "website color palette generator".

Don't write any CSS just yet, but organize these graphical elements into classes, and mark which class goes with each HTML element. For example:

```
H1 headers
Font:  Arial, strong, large
Font color: #B7B6C1
Text underlined with color #94958bs
```

Phase 2: Construction

After you've completed your site, start building! Before you write a single line of CSS, you should complete all of your HTML. There are two reasons for this strategy. First, it lets you focus on one language at a time. Shifting back and forth between languages is difficult for experienced developers. When you're first learning, it's even harder to "load" and "unload" different languages or technologies into your brain without becoming overwhelmed.

Second, it's much more efficient to handle style and layout after you've put everything onto the page. Otherwise, you'll find yourself spending a large amount of time perfecting and beautifying an element, only to find out that it doesn't fit, doesn't match, or can't be aligned with something else.

After the HTML is done, test it thoroughly to make sure that all the links work.

Phase 3: Styling

With your site feature and content complete, now is the time to make it pretty!

Using your planning work as a guide, write CSS classes to handle the layout and style of the elements on each of your pages. Try to be efficient, but don't worry too much about re-using or not re-using your code. The most important thing to focus on is organization and clarity. Apply what you've learned so far to use classes or selectors to apply each style to each element in a way that would be obvious to another developer when they look at your code.