Building your own academic website

Meredith Tamminga
University of Pennsylvania
June 2013

These are the notes for a workshop on putting together a personal academic website. It is intended as a total beginner's guide. It uses CSS templates, requires you to learn a little bit of HTML, and doesn't cover any alternative possibilities. It also doesn't cover the process of setting permissions and uploading your website etc. This is strictly a guide to putting together the content and style of a website in a very basic way. Babel is the name of the server for the Penn linguistics department, which is what the references to Babel refer to here.

Step One: Understand what's behind a website

In a very basic sense, your website will be a collection of documents in a folder. Most of the documents will be HTML documents, which are like marked-up text documents with the extension '.html'. You will keep a copy of the website folder on your own computer, where you edit it, and then you will upload a copy to Babel. Once you've put these documents in the right place on Babel and set the permissions correctly, they will be able to be found on the Internet at <http://ling.upenn.edu/~yourpennkey>.

Whenever you want to update your website, you will edit the HTML files in a text editor on your computer and then upload the new version to Babel to overwrite the old one. I use TextWrangler to edit my files, but there may well be better options. Just don't try to use TextEdit, the built-in text editor on Mac. It will show you what the HTML will look like in your browser window instead of the actual code, and then you'll be confused.

The other kind of file you will need is a CSS file. It also is like a marked-up text file, but the mark-up language is different and it does different things. You will have a CSS file called 'style.css'. The difference between HTML and CSS files is that the HTML files will determine the content and structure of each page on your website, and the CSS files will determine the appearance of that content and structure across all the pages on your website. You can edit your CSS file in a text editor just like the HTML files.

Besides these files, you can also put image files and PDFs into the folder for your website. Then you will be able to link to them.

Step Two: Get a CSS template

There are lots of free CSS templates available online. I found one I liked at freecsstemplates.org but if you google "free css templates" you will find other options. Once we start editing it won't be at all easy to switch between different templates, so choose carefully!

When you find a template you like, download it and unzip it. You should now have a folder with the name of the template. We're just going to edit this folder and its contents directly. You can rename the folder but don't rename the contents.

Step Three: Set up the structure for your website

You're probably going to end up with a bunch of files in your website folder, and you want to keep it clean and organized. You can put sub-folders inside the website folder to help keep things organized. Here's the structure I use for my website:

- Website
  - documents
    - abstract1.pdf
    - abstract2.pdf
    - paper1.pdf
    - ...
  - images
    - me.jpg
    - mycats.jpg
  - pages
    - about.html
    - research.html
  - index.html
  - style.css

The red items are files and the blue items are folders. You can just make the folders the way you normally would to organize your computer.

The file called 'index.html' will be your homepage. It should stay in the top level folder so don't move it somewhere else. You also should not call it anything different. This file already has some dummy content in it, which will make it easy to get started.

Step Four: Edit the content of your home page

Open the file 'index.html' in the text editor of your choice. At the very top it should say something like
<!DOCTYPE html PUBLIC "-W3CDTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">.
Don't touch this part! Just leave it there.

Next will come a commented-out note about the template. In HTML you begin a comment with <!-- and end the comment with -->. If you're using TextWrangler, everything inside the comment will be grey, making it easy to see how far the comment goes. You should also just leave this there, as it will remind you where you got your template if you ever need to find it again.

Next will come a line that says something like:
<html xmlns="http://www.w3.org/1999/xhtml">
Don't touch this part either! This is the beginning of the document. HTML is structured with tags that have an opening part and a closing part. The opening part of a tag looks like <tagname (optional-stuff-about-the-tag)>, and then you put the content that goes inside the tag after the >, and then when you want to close the tag it looks like </tagname>. So, if you scroll all the way down to the bottom of 'index.html', you should be able to find the closing html tag, </html>.

Next will come the head. You should see the tag <head> and then a few lines later be able to find the tag </head>. Ignore and leave intact everything in the head except the title. You can edit

the words between <title> and </title> to give your page a more informative title. For instance, mine is 'Meredith Tamminga - Linguistics Department - University of Pennsylvania'.

After the head comes the body. This is where the main content of your page will live. You'll need to recognize a couple new tags to edit this part:

-   A div is like a box that you put content into. Divs can have IDs, like id="page" or id="box". Later your CSS will be able apply different styles to these divs based on their IDs. I think of it like putting sticky notes on the outside of boxes saying what's in them so that you know which ones should be wrapped in birthday wrapping paper, holiday wrapping paper, etc. And just like boxes, you can put divs inside of divs.
-   This one is easy! This is the tag for a paragraph. If you want multiple paragraphs inside a div, just put each one inside a pair of these tags.

Those are the two that you really need to pick up to get anywhere. Here are a few more that you will likely find handy if you want to do anything other than put a paragraph on the screen:

-   To make a bullet pointed list use the tag for an unordered list (hence the "ul" in the tag name). Later your CSS will dictate what the bullet point style is.
-   If you're using the tag above, you'll want this tag to put around each list item.
- <h1> </h1> Section headings can be made a variety of sizes with the "h" tags. <h1> is the biggest heading and <h6> is the smallest heading. Make sure the number in the closing tag matches the number in the opening tag!
- Linked text This is how you make some text link to somewhere else on the Internet.
- Email me! This is how you create a link that opens an email window with your address pre-filled in the "To:" field.

You might also want a picture of yourself, or something else, on your homepage. We'll get to that shortly. For now, let's check that everything has gone smoothly. In your Finder window, just double click the index.html file (as opposed to opening it in a text editor). It should open up in a browser window so you can see what you've done. If it opens in a text editor, right-click to open it in a browser instead.

Step Five: Create and edit your other pages

Now you need to decide what other pages you want on your website. If you want to keep it simple and only have one page, skip ahead to Step Seven!

Your CSS template may or may not have come with extra pages. Luckily it's easy to make them yourself. Open up a new blank text document. Copy-paste the entire contents of index.html into your new page. Save the new page with a different name ending in ".html".

Now delete the stuff from your index that you don't want to duplicated, but keep the things that will be the same across all the pages. Definitely keep everything outside of the <html> tags, as well as the <html> tags themselves. I also keep my <head> the same across all my pages to function as a navigation menu. The parts that are different are inside the <body> tags. From here, you can go wild using the same tags as above.

Step Six: Link your other pages to your homepage

You almost certainly want to put links to your other pages on your homepage, so that you can give out your URL and then visitors can find their way around from there. Creating links to other pages within your own website is a lot like creating links to random pages on the Internet. The main difference is that you don't have to use the full URL. This does require you to learn a very minimal amount about paths, though.

The path of a file is like the directions to find it in your folder structure. There are slashes to mark the divisions between folders and also files, and they're listed in the order they're nested. In the file structure I outlined above, the path to my "About" page is /Website/pages/about.html. This is called an absolute path because it lists out the whole thing and thus is useful no matter what folder you're working in. But when I'm working on the HTML for a file that already lives partway into this file structure, I can also use a relative pathname if I want. If I were on my "Research" page, which lives in the same folder as the "About" page, I could just use about.html as the pathname. If I were working on my homepage, I would need the pathname indicate that you first move from my working directory, the top Website folder, into the "Pages" folder, before you go looking for the file. So I could create a link to my "About" page with:
About me

There's one more little twist. If I want the "About" page to refer back to the homepage, I have to make a path that gives directions to go up the file structure instead of down. Do this using dots, like so:
Home again

Step Seven: Add other content to link

A common use for an academic website is to share paper manuscripts, talk slides, and so on. This is very easy to do using the tools we've already learned. First, put the files you want to distribute into a folder in your website folder. They will need to be uploaded to the server as well. Once they're up, you can link to them just the way you link to other pages. For example, I can link to one of my imaginary files from my "Research" page by using:
My abstract

Step Eight: Make minor stylistic changes

We're going to peek into the CSS quickly but not mess around too much. It's actually not very complicated, but this tutorial is getting pretty long.

Step Nine: Teach yourself the rest

If you're anything like me you'll quickly find that there are lots of things you want to do to make your website look great and function in interesting ways. We've just seen the tip of the iceberg so far. Here are two tips for pushing your newfound HTML skills to the next level.

First, I use the website w3schools.com a lot. It's a site that will:

1. Give you short, accessible instructions on the many functions you might be looking for in languages like HTML and CSS

2. Allow you to play around and see how different functions work in real time without messing up your website

You can take their tutorials and work through the material systematically, or you can just go looking for what you need in the navigation bar on the left. The Try-It-Yourself Editor is where you can play with things. That said, I don't think it's exactly cutting-edge so if you are looking to actually get good at HTML maybe you should look elsewhere.

Second, I suggest installing the Chrome extension Firebug Lite. With this extension, you can look under the hood, so to speak, of websites that you like. It will open a little window along the bottom of the page that includes tabs showing the HTML and CSS underlying the page, which you can then study and learn from.

Happy coding!