Modifying Accessible Web Pages – The Preferred Method

If you want to put together a set of interlinked accessible web pages using the AWP templates you will need a better method than the "***Download, web page complete***" method that is fine for single, standalone web pages. So here, in detail, I will describe the preferred method.

What you will need:

1. A copy of the AWP project that includes the templates and other resource files.
2. A proper text editor that supports coding like Geany, or Notepad++.
3. An up-to-date web browser
4. You will need to know how to use the awp built in editing tool, and how to copy and paste.

Get a copy of the AWP project:
To get a copy of the AWP project go to https://github.com/jamjolu/awp and click the green ***Clone or download*** button. Then select ***Download ZIP*** to wherever you want. (There is also a download for the project's zip in the ***release*** section of the github site.) After downloading and scanning the zip file for malware, extract the contents (in Windows – right-click and choose ***Extract all***.) The AWP project is stored at Github which acts as a repository, and, offers tools that help developers collaborate and manage versions of evolving projects. You need only to concern yourself with the download button.

Get a proper text editor for coding:
In this how-to I will be using images from the Geany text editor because there are versions of it that work for most popular operating systems. You can get Geany for Mac or Windows here:
https://www.geany.org/Download/Releases
You can also get it for Linux using the software installer that came with you Linux distribution. Be sure to install Geany (or your preferred coding editor) prior to starting editing any awp pages.

Text editors like Geany support the syntax and structure of coding languages like html automatically offering visual cues that help the coder to see the structure of their code. You can also have multiple documents open in separate tabs unlike simple text editors like Window's ***Notepad***. Html documents are internally organized into blocks of text that are surrounded by "tags". Text editors like Geany allow you to collapse a block of tag surrounded text by clicking the little minus sign that precedes the first tag of a block. This makes it very easy to replace a whole long block of text without having to scroll around. *This feature will come in very handy.*

Explore the *awp-master* folder and make a new folder for your custom pages:
If you explore the folder you downloaded called ***awp-master***, you will see it contains another folder called ***awp-master***, and *inside* that folder you will see the entire awp collection of templates and resources. If you open the ***index.html*** file by double-clicking you will see the main "Index Page for Accessible Web Pages" web page. It is in this *inner **awp-master*** folder you will create your own folders for you own materials. Let's say you want to create a page set based on the theme "The Ducks of Upstate New York". Before you start modifying templates, create a new folder inside the inner awp-master folder called ***UNY_ducks***. It is a good idea to not use spaces in file or folder names. Instead use the underscore character or a dash if you want to separate words. Now let say we want this page set to have a table of contents and a set of story type pages, one for each kind of duck. There are actually dozens of ducks found in upstate NY, but we can start with two: the mallard and the wood duck.

The awp-master folder with the UNY_ducks folder created

Copy templates to your custom pages folder and rename them:
Next, we will go into the *templates* folder and copy the *6_link_TOC_builder.html* file and the *storyBuilderPicLeft.html* file. Paste these inside *UNY_ducks* folder. Make an extra copy of the story builder file. Rename the 6 link file to *UNY_ducks.html*, and rename the other two story files *mallard.html* and *woodduck.html*, respectively.

The new duck themed pages inside the UNY_ducks folder

**You are ready to edit your pages, and the workflow will go as follows**:
1. Open any pages you want to edit in Geany (in Windows you can right-click on a file and select *Open with Geany*.
2. Open the same files in a web browser – the examples shown will use Google Chrome.
3. Edit the files in the browser using the built-in editor.
4. Using the *Developer Tools* in the Chrome browser, copy the html content of the edited page.
5. Paste the copied html content over the html content of the same filename in Geany.
6. Save the altered Geany file and test it.

You can repeat steps 3 to 6 to work out any kinks until you are satisfied.

The editing process in detail:

1.   Be sure you are in the *UNY_ducks* folder and open any pages you want to edit in Geany (in Windows you can right-click on a file and select *Open with Geany*). Each file will appear as a separate tab. Note all the different colors assigned to the text. Also note the little minus sign next to line 2 that begins with *<html lang="en">*. That is the html tag that wraps around nearly the entire web page. It also has a bit code that defines the default web page language. You could change the two letter code enclosed in quotes to specify another language. Click the minus sign. The entire html block collapses hiding the content! But, don't worry. Click the plus sign on line 2 and it all reappears. After we edit our

pages in the browser we will use this feature to simplify replacing the html block with our edited content.
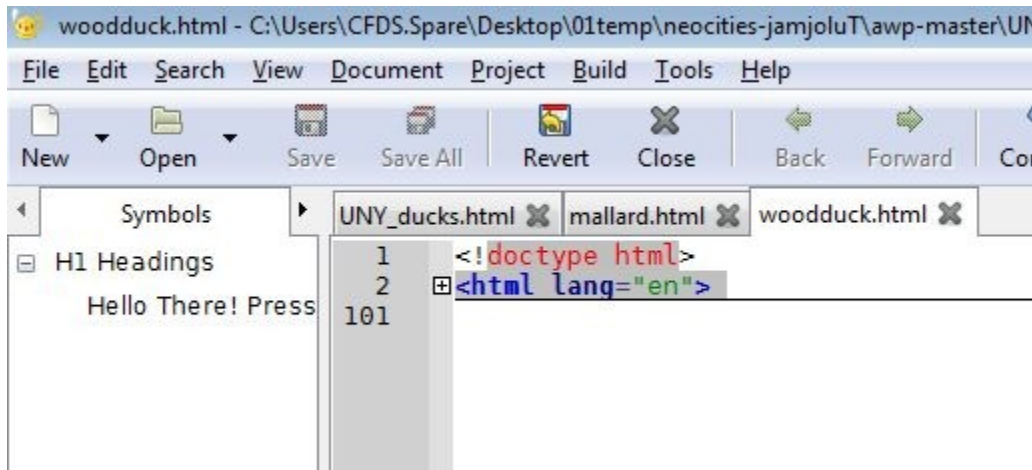
The Geany editor with 3 files open and the <html lang="en"> tag collapsed in woodduck.html

2.   Open the same files you opened in Geany in a web browser (most desktop browsers will do, but the example here uses Google Chrome.)

3.   Start with any one page and edit the content using the built-in editor (press alt-e to reveal the editor drop down.) There are more tutorials in this project including the basic one about editing pages. You can review it by opening to the *index.html* page and following the *How to: An Easy Method..* link. Most of the details here will be be helpful, but we will be using a very different method for saving our edited pages.


The Male Mallard Duck

Editing the mallard.html page- Remember to close the editor before saving the page

4.   When you have made some changes using the editor you will need to save them. This method requires that we use the *Developer tools* that is a feature of most modern browsers. In Chrome, click the controls button in the upper right hand corner of the browser window (it looks like 3 stacked dots.) The choose *More tools*, and then *Developer tools*.
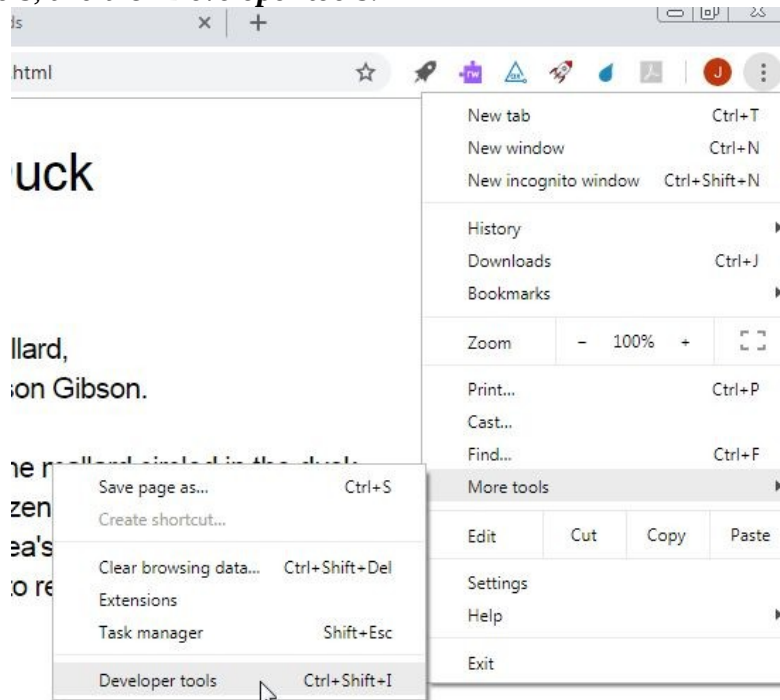
***Open the Developer Tools using the Chrome control menu***

This will open a complex looking pane on the right side of the screen. Towards the top of the developer tools pane you should see a space where the *<html lang="en">* tag appears. Right click on the *<html lang="en">* tag. From the menu that opens choose *Copy* and then *Copy outerHTML*. Now you have a copy of your edited page in the computer's clipboard.
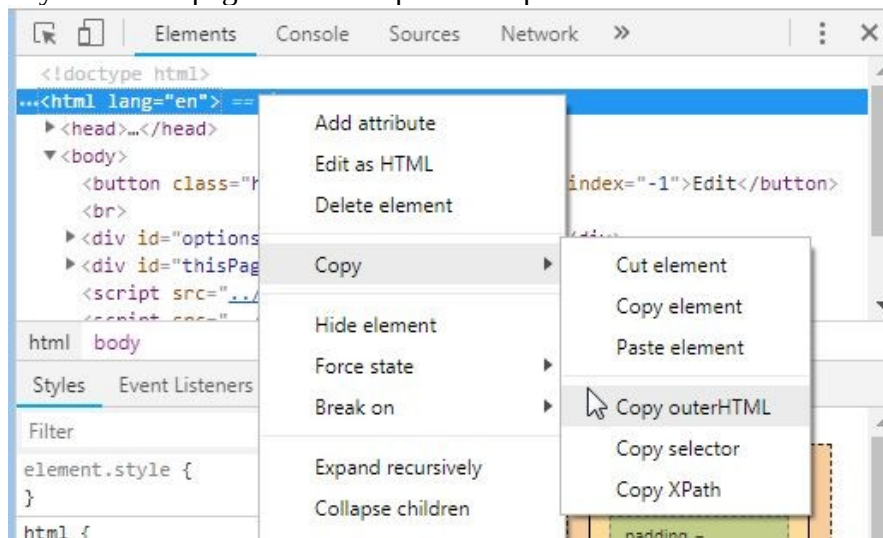

Copying the "outerHTML" of the <html lang="en"> tag. This gets everything we need.

5.   Switch to the Geany tab that holds the same page. Collapse the ***<html lang="en">*** block by clicking the minus sign if necessary. Highlight the collapsed block by dragging over it. Paste your edited version over the highlighted block (using right-click → paste, or by selecting paste from the edit menu.) Scroll back to the top of the file in Geany. Do you see the ***<html lang="en">*** tag? If not, repeat the step 4 and try again.
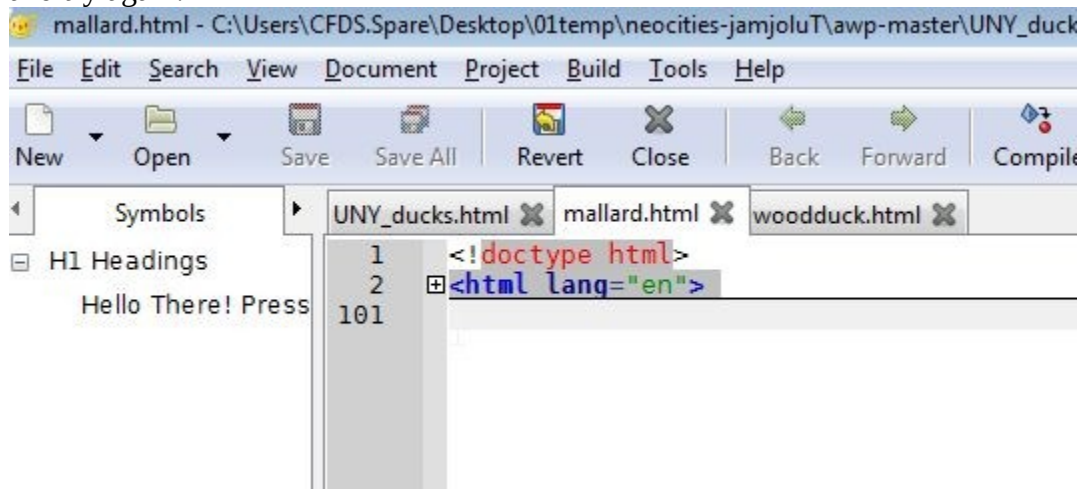
Collapse the <html lang="en"> tag and highlight the line using a mouse drag.
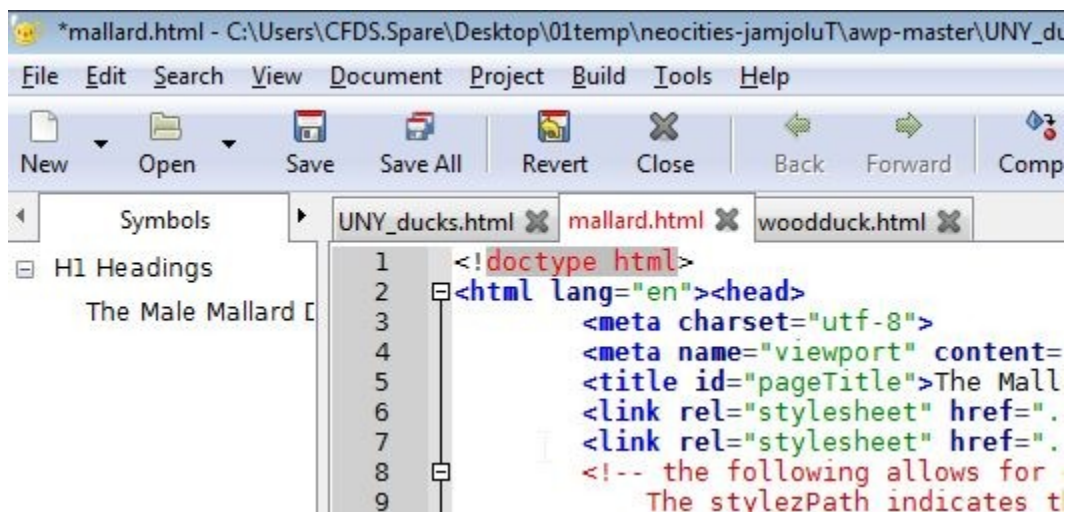
After pasting the clipboard, we replaced the old html content with our new content. Note that we want to preserve line 1 - <!doctype html> - This tells the browser how to interpret the rest of the page.

6.   Save the page (use the file menu → save, or click Geany's Save button.) Reopen your newly edited page in a browser. Does it look OK? If not use Geany's undo (edit menu → undo) and try pasting, saving and re-opening again.

Because you have been using copies of the templates there is very little you can mess up. The entire awp project was built from awp templates using this method.