How to Add a Webpage

Carolyn Anderson
last updated May 21, 2015

Some webpages for the Learn Mi’gmaq site are generated from the XML file. These include lessons, units, and sections. If you want to add one of these, you should read the How to Add a Lesson guide instead.

Other pages, like the About the Project page and the Acknowledgements page, need to be written directly. Let’s say that you want to add a new page explaining some of the history of the Mi’gmaq language. This page would need to be written as an **HTML** file.

To start, you will need a program to write files in. Your computer should have one for free: Notepad on Windows or TextEdit on Mac.

You can also download a program online for free that will do some formatting to make it easier for you, for instance, it may have something like spell-check, or auto-complete. I have tried out two free programs that I found online: Sublime and Brackets. Brackets is designed for web-programming, but Sublime has features for a wider range of programming languages. Either will work for writing a new webpage.

Creating a new file

Start by opening a new file in whichever editing program you have chosen. You should save the file in the main directory with a name that has no spaces and ends in *.html*.

For instance, we might call our History of the Language page *history.html*.

Picking a layout

Next, you will need to pick which **layout** to use with your page. A layout is some pre-written code that is added to each page by **Jekyll**, the program that generates our site. For instance, our layouts add the side menu, the top navigation bar, the copyright notice at the bottom of the page, and link to the stylesheets that tell the computer how to display our page. For more on Jekyll, see the How to Use Jekyll guide.

The two layouts that have been written for the Learn Mi’gmaq website so far are called *frame.html* and *lesson.html*. *Lesson.html* is used for lessons, units, and sections, while *frame.html* is used for everything else. You will probably want to use the *frame* layout on any page you are writing yourself.

The first thing in your new HTML document should be a markup telling Jekyll which layout to apply. Jekyll markups begin and end with three hyphens to tell Jekyll to start and stop reading. Then, you will tell Jekyll which layout to use:

```
---
layout: frame
---
```

(I will be showing what is new at each step in green.)

Setting other variables

You can also define other variables to use on the page in the Jekyll markup. Variables are useful for storing a value that you want to use many times in the same page. If you have a value used in multiple places that you think you might want to change something later, you can store it in a variable, and then you will only need to change the variable definition, instead of tracking down every place you used the value.

One variable that should be set on every page is the page title. This is important because the layout uses the page variable. This allows the layout to apply to pages with many different titles.

```
---
layout: frame
title: About the Mi’gmaq Language
---
```

You can use a variable within the page by writing *{{ page.name }}*, where *name* is what you called the variable in the markup. There must be a space between the opening curly brackets and the variable, and between the variable and the closing curly brackets.

For instance, you can use the title variable like so:

The title of this page is {{ page.title }}.

The line would appear on the website like this:

The title of this page is About the Mi’gmaq Language.

Title

Now you need to decide what should appear on your page. Somewhere near the top of your page, you should place the title of the page, using the title variable you have defined.

The page title should be large so that it is easy to read. Elsewhere on the site, titles have been placed in *<h1>* tags. The *h* is for heading, while the number indicates the size. Headings come in sizes 1-6, where 1 is the largest.

```
---
layout: frame
title: About the Mi’gmaq Language
---
<h1>{{ page.title }}</h1>
```

All HTML elements begin with an **opening tag**, indicating what kind of element it is, and end with a **closing tag**. In the example above, for instance, the *<h1>* opens the heading and the *</h1>* closes it.

CSS

I said above that the number part of the heading tag indicates what size it is. In fact, the numbers only indicate the relative size: that is, a 1 is bigger than a 6.

The actual sizes are defined in the **CSS** files linked to in our layouts. CSS (Cascading Style Sheets) is what controls a lot of the appearance of the site. CSS defines what font sizes, colors, borders, and other styles web browsers use to display our pages.

There are three important CSS files for the Learn Mi'gmaq site.

The *bootstrap.css* file contains the Bootstrap styles we use (see below).

The *yeti.css* file is a Bootswatch theme that our site uses. This gives our site extra-nice fonts and other things.

The *custom.css file* is our own CSS written for the Learn Mi'gmaq site specifically. **If you ever need to modify or add CSS, it should be done in the custom.css file.**

What happens if you write conflicting rules in two files? **The last file to be linked to takes precedent.** For our system, we link to *bootstrap.css* first, then *yeti.css*, then *custom.css*, because we want our own styles to be able to override the Bootstrap and Bootswatch styles.

Bootstrap

Bootstrap is a framework for web programming. We use it on the Learn Mi'gmaq site because it is useful in making things display nicely on different screen sizes. It is also a very popular tool, so it is easy to find examples online for various things you might like to do.

Bootstrap is mainly a CSS file defining many different styles. This CSS file is linked to in our layouts, so that each page can access it.

For example, one useful CSS class that Bootstrap defines is columns. HTML only defines boxes (<*div*>s): to make columns, you used to have to manually adjust the widths of different boxes.

Bootstrap, however, gives us a grid system, with different column sizes that we can use. It divides the page into 12 columns, and you can specify that a div should take up from anywhere from one to twelve of them, by adding a column class to its tag:

<div class="col-md-9">

The *col* stands for column. The *9* specifies that the column should take up 9 out of 12 columns of the page (3/4ths of the page). The *md* stands for medium--- this column will be size 9/12 on screen-sizes up to medium size.

If the screen size is small or extra-small (think tablets and smartphones), then the page will stop using the grid system, and the <div> will take over the whole width of the page.

For more on the Bootstrap grid system, see this reference page: http://getbootstrap.com/examples/grid/.

Content Body

I can't spell out instructions for every kind of thing you might like to put on the page. But I can give some advice for how to learn to write webpages.

First, look at existing pages. You may be able to copy the copy from them, and just change the text inside.

Second, there are a lot of examples and templates of how to use Bootstrap online. It's perfectly fine to borrow code you find online, and you can often get good help this way.

Third, if there's something you see on another website that you would like to have on the Learn Mi'gmaq website, you may be able to use your browser's **Developer Tools** to look at their code.

Developer Tools

Many web browsers have Developer Tools to help website programmers. These tools allow you to look at the code behind a website. They even allow you to modify the code to see what happens.

On Google Chrome, you can find these tools by going to the menu, finding More Tools, and clicking on Developer Tools, or by control+clicking on a page and choosing Inspect Element.

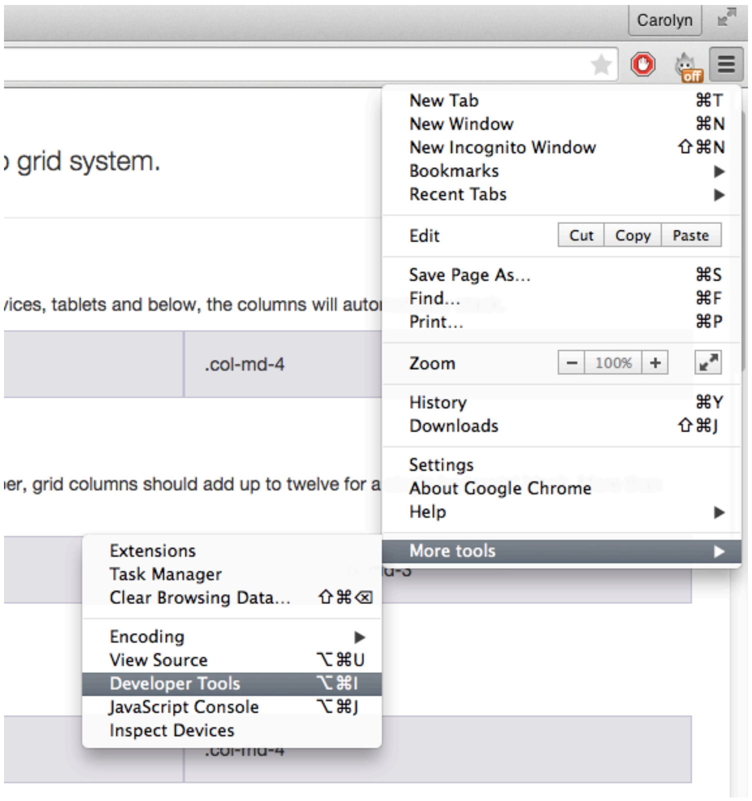
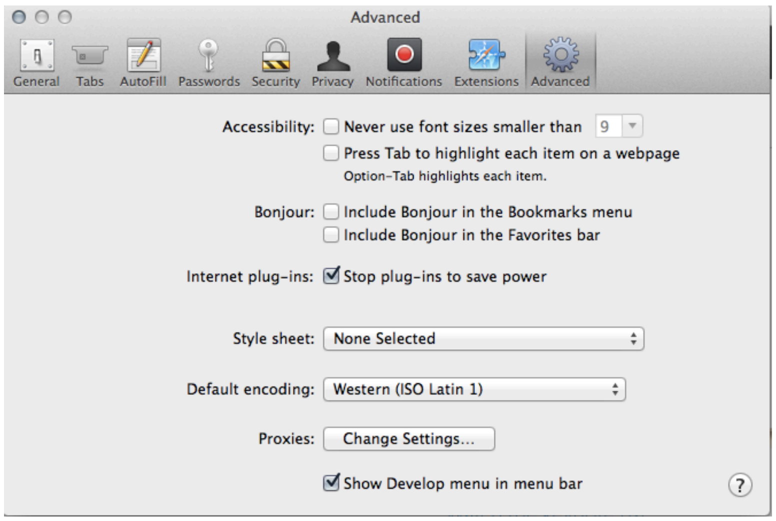


Figure 1. Finding Developer Tools in Google Chrome.

On Safari, you display the Safari Develop menu in your menu bar by choosing Safari > Preferences, clicking Advanced, and selecting the checkbox for Safari Develop:



Accessibility: ☐ Never use font sizes smaller than 9
☐ Press Tab to highlight each item on a webpage
Option-Tab highlights each item.

Bonjour: ☐ Include Bonjour in the Bookmarks menu
☐ Include Bonjour in the Favorites bar

Internet plug-ins: ☑ Stop plug-ins to save power

Style sheet: None Selected

Default encoding: Western (ISO Latin 1)

Proxies: Change Settings…

☑ Show Develop menu in menu bar

Figure 2. Finding Developer Tools in Safari.

<u>Adding Links</u>

To make an HTML element go to another page on a click, or to display content from another file, like an image or an audio file, you will need to create a link.

First, you'll need to know what folder the file you are linking is in, or the url of the outside website you want to link to.

For an image, this will probably be img. For an audio file, this will probably be audio. For a main page, there probably won't be a folder, while lessons, units, and sections are stored in separate folders.

If you want to link to another webpage when the user clicks, you will wrap the element in an <a> element:

```
<a href="http://www.migmaq.org">
	<p>Click here to visit Mi'gmaq.org!</p>
</a>
```

In this example, you are linking to an outside website. If you want to link to a page in the Learn Mi'gmaq website, the address is a little more complicated.

```
<a href="{{ site.baseurl }}/sections/1.html">
	<p>Click here to go to the first section!</p>
</a>
```

The first part is a variable defining the base URL for the site. The second part is the folder that contains the file (none if the file is in the main folder). The third part is the name of the file. Each part should be separated by a slash.

Adding Images

To add an image to a webpage, you should first place the image file in the *img* folder. Next, you'll need to write the code to include it on the page. There are many ways to do this depending on the style you want, but the way it has been done many times on the website so far is below:

```
<img class="img-responsive thumbnail" src="{{ site.baseurl }}/img/file_name.jpg">
```

Make sure you get the file type right at the end: it may be a JPEG, GIF, or PNG image, and it won't display if you get it wrong.

(Images are an exception to the ending tag rule: they're a stand-alone element and don't need a closing tag.)

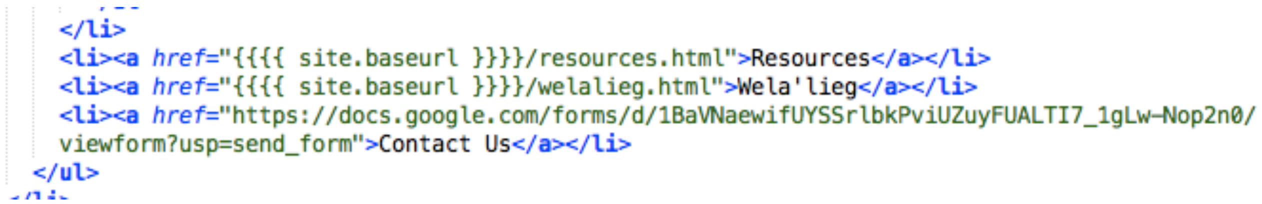
Linking to Your File

You will also need to link to your file, so that website users can find it. The most likely place to add it is to the side menu.

To add your file to the side menu, you will need to edit the *sidenav.xsl* file. It is in the data folder. Open it in your editing program.

You will need to find an example in *sidenav.xsl* of how to link a file, duplicate it, and change the name and link to those of your file.

For instance, if you want your file to appear on the menu between Resources and Wela'lieg, find those names in *sidenav.xsl*.



```
    </li>
    <li><a href="{{{{ site.baseurl }}}}/resources.html">Resources</a></li>
    <li><a href="{{{{ site.baseurl }}}}/welalieg.html">Wela'lieg</a></li>
    <li><a href="https://docs.google.com/forms/d/1BaVNaewifUYSSrlbkPviUZuyFUALTI7_1gLw-Nop2n0/
    viewform?usp=send_form">Contact Us</a></li>
  </ul>
```

Figure 3. Finding an example in *sidenav.xsl*.

Next, copy and paste the Resources <li>.

<li><a href="{{{{ site.baseurl }}}}/resources.html">Resources</a></li>
<li><a href="{{{{ site.baseurl }}}}/resources.html">Resources</a></li>
<li><a href="{{{{ site.baseurl }}}}/welalieg.html">Wela'lieg</a></li>

Then edit the link and the name to match your file.

<li><a href="{{{{ site.baseurl }}}}/resources.html">Resources</a></li>
<li><a href="{{{{ site.baseurl }}}}/history.html">About the Mi'gmaq Language</a></li>
<li><a href="{{{{ site.baseurl }}}}/welalieg.html">Wela'lieg</a></li>

Last, you will need to re-run the code to make the side menu. You will need to run the following code in the command line from the main folder.

touch data/master.xml; make

The side menu should automatically update with the new link to your page.