How to build a personal webpage quickly

Yan Gobeil

March 11, 2017

This is a summary of some very basic things to know to create a personal webage. I wrote it down so everyone can build a simple personal page on the McGill website without having to check anywhere else. Of course this is extremely basic and if you want to do more you can Google as much as you want. I am far from an expert, but this is what I used to build my own page and it seems to work.

For example of things that you can do and to give you ideas you can look at the following webpages of some McGill people:

1. `http://www.physics.mcgill.ca/~maloney/`
2. `http://www.physics.mcgill.ca/~rhb/`
3. `http://www.physics.mcgill.ca/~yangob/`
4. `http://www.physics.mcgill.ca/~pfduc/`
5. `http://www.physics.mcgill.ca/index.html`

If you want more ideas you can browse through everybody's webpages here.

1 Basics to know

To build and view a webpage like we are doing here, the only things that you need are a text editor and a web browser. The text editor is where you write all of your code. You can use whatever you want and there are also very good editor that you can download. The web browser is what will allow you to see the webpage that you are building. Opening your file with the text editor gives you a big code and opening the same file with a web browser gives you a beautiful webpage.

In what I present there are two layers of coding: HTML and CSS. HTML is the content of your page: the text. The language is very simple and you use it to structure the words, pictures and links that will appear online. It's possible to do your webpage using just HTML and it works fine even if it's not super beautiful. The first two links above are made using only HTML. If you want to add more complex structures to your page you need to use CSS, which allows you to personalize your things with colors, put different things in boxes, etc. The amount of things you can do with CSS is almost infinite and I will only discuss some basic ones here, but when you know the rules it's simple to find online how to do something specific. The third and fourth links show examples of webpages that use CSS. At this point, everything in your webpage is static. If you want to add interactions between the site and the mouse or things that move you have to use Java, which I will not discuss here. The last link show what you can do with Java.

The structure of the files that you will have when you will be done is the following. First there is an HTML file for each page of your website. The main one is called *index* to allow the McGill server to find it and the rest are called whatever you want. Next there is only one CSS file for the whole website. For big sites it can be useful to use more but we should not need more. All of these will be contained in your webpage folder along with all the images and PDFs that you want to put online. To put the website online you just have to copy all the files in your folder into the WWW folder on your McGill desktop. The files are updated every day so it will not go online immediately.

2 HTML

To start building your website, open your favorite text editor and save the first file as *index.html*. This will be the front page of the site. Every other file that you will create for a new page will be called as you like but always with *.html* after. When you code in HTML you have to tell the computer the nature of what you are writing. This is done by using tags. In general you start something with an opening one $< tag >$ and end with a closing one $< /tag >$. In some cases you only need an opening tag, like when you insert an image. There are also options for different tags, which are written only in the opening one.

An HTML file always contains the following things:
$<!DOCTYPE\,html>$
$< html >$
$< head >$
$< title > \quad < /title >$
$< /head >$
$< body >$

$< /body >$
$< /html >$

The first line just says that the file is an HTML website. The $< html >$ tag contains all of the code so it closes at the very end. The $< head >$ tag is where you write all the reference to other files, like the CSS.

The title is what appears as the name of the tab on your web browser and as the name of the page on Google for example. The body is simply all the content of the website. This is the only thing that will appear online.

It's good to note that if you want to write something as a comment without it appearing on the page you use $<! - - comment - - >$

Now that the basic structure is clear, here is a list of the most important tags that you will use to create your webpage:

- The paragraph tag $< p >< /p >$ lets you write a paragraph. Note that the browser doesn't see if you write on different lines in your code. To go to the next line you have to use the break tag $< br/ >$.
- Titles of sections, paragraphs and so on are written using the tags $< h1 >< /h1 >$,...,$< h6 >< /h6 >$. The most important titles use h1 and then you decrease the order. Of course many different titles can have the same importance.
- To emphasize certain words, there are different options. The tag $< b >< /b >$ puts a word in bold, $< i >< /i >$ puts words in italic and $< u >< /u >$ underlines words. You will be able to change the appearance of the words with CSS so don't worry too much about how they look like at this point.
- To make lists, you can use $< ul >< /ul >$ to have the elements with bullet points or you can use $< ol >< /ol >$ to have numbers. The elements of a list are noted with $< li >< /li >$.
- To create a hyperlink to another website, you use the tag $< a\ href = “address'' > text < /a >$. If you want to link to another page of your own website or to a PDF that you have in your folder, you use the same tag but in the reference you just write the name of the file with the extension. Note that if the file you are linking to is not in the same folder you have to use the path that leads to it, but we will keep it simple here. A useful link to try is the reference $mailto:$ followed by your email address. Obviously this will open directly an email to be sent to you.
- The tag to insert an image is simply the orphan tag $< img\ src = “source''\ alt = “alternate'' >$. The source part is just the place where the image is. In our case you simply write the name of the file since it is in the same folder. The alternate part is a text that describes the picture and will be written if the picture can't load. Images must always be inserted inside paragraphs and at the place you want them to be.
- It is also possible to put a figure outside of a paragraph by inserting it inside the tag $< figure >< /figure >$. Many images can be inserted inside one figure and the tag $< figcaption >< /figcaption >$ gives a caption to these images.

There are many other tags that can be used so you can check online if you want, but the most useful ones for our purposes are here.

3 CSS

Now that the basis of the website is done, let's make it more attractive! The CSS code is written in a new text file that you call “style.css” that you save in the same folder. To include it into the HTML code you need to add the line $< link\ rel = “stylesheet''\ href = “style.css'' >$ in the head of each page of your website. The same CSS file will be used for all the pages.

The first thing that you can do with CSS is personalize all the HTML tags that you used. The format of the code is the following:
tag
{
$property1 : value1;$
$property2 : value2;$
}
This code will give all the tags of this type the given value for the properties. For example, you can use the tag $h1$ and put all of them with color blue. You can also give the same property to many tag by separating them by a coma. If you want a specific tag to have some property, you need to go back to the HTML code and add $class = ``''$ inside the opening tag to give it a name. Then you use the same CSS code as before but you write $.nameoftag$ instead of tag. If the thing you want to personalize is not inside a whole tag (like a specific word in a paragraph), you can put it inside the universal tag $< span >< /span >$ and give a name to this element. Finally, if you want to select for example all the links inside a given paragraph, you list them in order of biggest to smallest with spaces between them. in this example you would write ".para a".

Now that we know how to change the properties of the website, let's see what we can do specifically.

- To change the size of the text, the most efficient way is using the property "font-size" and the possible values are multiples of "em". For example, the value "em" puts the text to normal size, the value "2em" puts it twice the normal size, and "0.5em" puts it half the normal size. Any decimal number can be used.
- The property "font-family" determines the font of the text. You just write the font that you want in the value, and it is suggested to write many different ones separated by comas in case your favorite one is not supported by the web browser. Fonts with more than one word should be put inside quotation marks. You can Google the fonts available to see what they look like.
- You can modify the alignment of the text by using the property "text-align". The possible values are left, right, center, justify.
- You change the color of the text in an element you just use the property "color". At the basic level there are 16 colors available, which are: white, silver, gray, black, red, maroon, lime, green, yellow, olive, blue, navy, fuchsia, purple, aqua, teal. It's possible to choose more precise colors and if you want to do it it's easy to find how online.
- The property used to change the color of the background is of course "background-color" and the options are the same as for the color of the text. It's possible to put an image as background, using "background-image" with the value being the name of the file. Many options are available in that case. The property "background-attachment" determines if the image is fixed when you scroll the page (value "fixed") or if it follows the text (value "scroll"). By default the image is repeated in a mosaic on the background so if you want to change that you use "background-repeat". The possible values are "no-repeat", "repeat-x" and "repeat-y". In the case where you ask it to not repeat, you can decide where the image is by using "background-position", which can be set to top, bottom, left, right, center or any combination of these (like top right for example).

- It's good to know that we can play with the borders of the boxes that contain a given tag. See internet for that.

It is important to note that any property that are used for a tag apply for all the other tags that it contains, except if you specifically give them some other property. For example if you put the background of the "body" tag in yellow, all of the tags that it contains (so everything) will have a yellow background, except the ones for which you specified another background color.

4 Structure

At this point, the content of the webpage is decided and the properties of the text are setup. However your work still doesn't look like a real website so we need to talk about how to structure things. For this we will have to go back to the HTML code and tell the computer what is what in the text. We will then use CSS to place the different parts where we want on the page.

Before starting, to make sure that every web browser understands your code, you have to add the following line in your header:
$<! - -[if\ lt\ IE\ 9 >$
$< script\, src = "http : //html5shiv.googlecode.com/svn/trunk/html5.js" >< /script >$
$<!/[endif] - - >$

The new tags that you should know are the following:

- The header tag $< header > < /header >$ will include the elements of the top of the page, like an image or a slogan.
- The tag $< nav > < /nav >$ will include the navigation bar of your webpage, where there will be links to the other pages of your site.
- $< footer > < /footer >$ contains whatever you want to put at the bottom of the page, like your address or ways to contact you.
- The tag $< section > < /section >$ is obviously where you will separate the core of the page. There can be many sections, depending on how you decide to structure your things.
- The useful tag $< div > < /div >$ can be used for anything that doesn't fit with the other tags.

There are also tags for articles and asides that are often inside sections so you can use them if you want. You should divide your HTML code into these parts by thinking about the order and not the specific position and then we will work with that.

Now let's come back to CSS and discuss about the formatting of what we will call blocks, the tags that basically contain a part of the website, like all the new ones introduced and for example $< p >$ and $< h1 >$. Of course what we will do is mostly useful for the new tags but it can be used for the other blocks.

- To change the dimension of the block, you use the properties “width” and “height” and the simplest way of setting the values is with percentages of the total size of the page. You can also use a number of pixels, but the size will not adapt to the size of the window of the person who views it. If you use this property to rescale an image, you can set one of the dimensions in percentage and set the other one to auto to keep the ratio of the picture.
- The way of modifying the size of the interior margins of the blocks is with “padding”, this time specified in pixels with the letters “px”. You can specify independently every margin by using “padding-top” and the same with bottom, left and right. The way of playing with the exterior margins, so how much space to leave outside of the block, is with “margin”, which is used exactly in the same way.
- If you decide to fix the size of a block and think there could be no more room for the text at some point, use “overflow” to decide what to do with the extra text and set it to auto to let the browser decide.
- When building your navigation bar, it could be useful to use the property “list-style: none” to get rid of the bullets in a list.

The last element that is needed in order to organize the webpage properly is the “float” property for a block. Originally this is supposed to be used to make the text go around an image inside a paragraph, but we can use it to position things. Basically what it does is take an element out of the flow of the code and puts it to the left or the right depending on the value you give it. Every float objects align with each other if there is room and this is why we can use it to put sections next to each other. Sometimes this way of doing it may cause problems and using the property “clear: both” for the elements that are after the floated ones can help position them correctly.

5 Putting the site online

Now that you know how to build your webpage, it would be good to be able to put it on the McGill website. If you have access to a computer from the department, it’s really easy. You just have to copy all the files for your webpage to the folder called WWW. If you want to do it from your computer, you can download the software Filezilla at `https://filezilla-project.org/`. Then you connect to your account by using the host name calys.physics.mcgill.ca, you McGill physics username and passwords and the port 22. After you connect you just copy the files to the folder WWW.

No matter which way you use to put the files on your account, the files are pushed to the server every day so it will not be online directly. To push it right now, you open a terminal on a McGill computer or you download Putty at `http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html` if you are on your own computer. Then you do the following:

1. Connect to the server with Putty or via a Unix terminal by typing
   > ssh username@calys.physics.mcgill.ca
   and press ENTER. Enter your password and press ENTER again.

2. Move to /WWW by typing
   > cd WWW
   and press ENTER.

3. Type
   > ssh -x ataura webpush
   and press ENTER. Enter your password and press ENTER again. Press ENTER another time.

4. To logout, you can do Ctrl+D or type
   > exit
   and press ENTER.