Hacking Hot Potatoes: An introduction to customizing your exercises

Index

How Hot Potatoes uses source files - Windows Internet Explorer

https://hotpot.uvic.ca/howto/hacking_workshop/source_file_system.htm

File Edit View Favorites Tools Help

Favorites | Hacking Hot Potatoes: An int... | How Hot Potatoes uses s...

Normal | Custom

How Hot Potatoes uses source files

Index | Next

JQuiz engine	Base source folder
jquiz6.ht_	jquiz6.ht_
hp6.cs_	hp6.cs_
hp6navbar.ht_	hp6navbar.ht_
hp6buttons.js_	hp6buttons.js_
hp6showmessage.js_	hp6showmessage.js_
hp6utilities.js_	hp6utilities.js_
hp6hotpotnet.js_	hp6hotpotnet.js_
jquiz6.js_	jquiz6.js_
hp6checkshortanswer.js_	hp6checkshortanswer.js_
hp6timer.js_	hp6timer.js_
hp6browsercheck.js_	hp6browsercheck.js_

Done | Internet | 100%

How Hot Potatoes uses source files - Windows Internet Explorer

https://hotpot.uvic.ca/howto/hacking_workshop/source_file_system.htm

File Edit View Favorites Tools Help

Favorites | Hacking Hot Potatoes: An int... | How Hot Potatoes uses s...

Normal | Custom

How Hot Potatoes uses source files

Index | Next

JQuiz engine	Custom source folder	Base source folder
jquiz6.ht_	jquiz6.ht_	jquiz6.ht_
hp6.cs_	hp6.cs_	hp6.cs_
hp6navbar.ht_	hp6navbar.ht_	hp6navbar.ht_
hp6buttons.js_		hp6buttons.js_
hp6showmessage.js_		hp6showmessage.js_
hp6utilities.js_		hp6utilities.js_
hp6hotpotnet.js_		hp6hotpotnet.js_
jquiz6.js_		jquiz6.js_
hp6checkshortanswer.js_		hp6checkshortanswer.js_
hp6timer.js_		hp6timer.js_
hp6browsercheck.js_		hp6browsercheck.js_

Done | Internet | 100%

Three types of code, and how they interact

XHTML	CSS	JavaScript
XHTML consists of content elements on the page: • headings • paragraphs • lists • links • buttons • textboxes • [...]	CSS controls how those elements are displayed: • colours • size • position • text style • alignment • borders • visibility • [...]	JavaScript changes and manipulates the elements on the page, by changing the CSS and XHTML to: • hide and show things • move things • change the colour of things • change the text of things • check answers • calculate scores • [...]

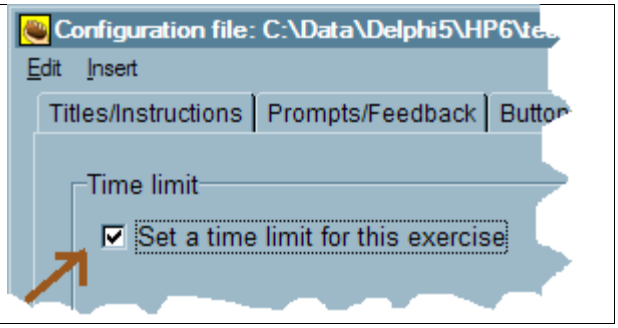
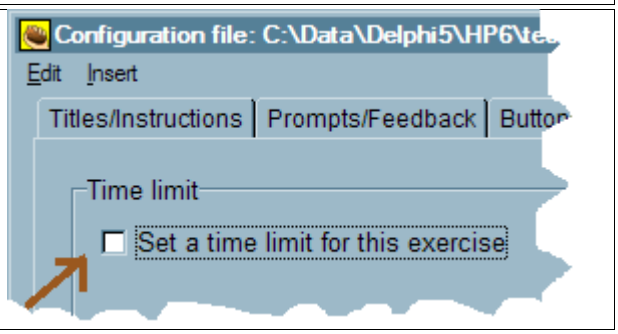
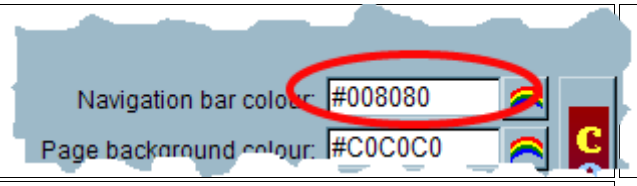
Examples of the three types of code

XHTML (*.ht_)	CSS (hp6.cs_)	JavaScript (*.js_)

`<div class="Feedback" id="FeedbackDiv">`
 	
`<div class="FeedbackText" id="FeedbackContent" > </div>`
 	
`<button`
	`div.Feedback {`
	`  left: 33%;`
	`  width: 34%;`
	`  border-style: solid;`
	`  border-`
		`function HideFeedback(){`
		`  document.getElementById('FeedbackDiv').style.display = 'none';`
		`[...]`
		`  if (Finished == true){`
		`  Finish();`
		`  }`

XHTML (*.ht_)	CSS (hp6.cs_)	JavaScript (*.js_)
id="FeedbackOKButton" class="FuncButton" [...] onclick="HideFeedback(); return false;">OK</button> 	width: 1px; position: absolute; display: none; font-size: small; [...] }	}

Replacements and includes

	=>	... [inclTimer] setTimeout('StartTimer()', 50); [/inclTimer] ...
	=>	
	=>	... div.NavButtonBar{ background-color: [strNavBarColor]; text-align: center; margin: 2px 0px 2px

```
0px;
  clear: both;
  font-size: small;
}
...
```

Five basic types of source file

[AppName]6.ht_	[AppName]6.js_	hp6*.ht_	hp6.cs_	hp6*.js_
This is the basic framework for the Web page for this exercise type, into which all the other code is inserted. For example, **jquiz6.ht_** is used to create JQuiz exercises.	This is specialized JavaScript used only by this exercise type. For example, **jquiz6.js_** contains the code used in JQuiz exercises.	This is XHTML used in multiple locations. For example, **hp6navbar.ht_** contains the code used for navigation buttons in all the exercise types.	This file contains all the CSS code used in all the exercises.	This is JavaScript code used for one specific set of functions, but used in multiple exercise types. For example, **hp6card.js_** contains the code used to handle dragging

[AppName]6.ht_	[AppName]6.js_	hp6*.ht_	hp6.cs_	hp6*.js_
				and dropping of elements in JMix and JMatch exercises.

Good practices when editing source files

=>	Create a special source folder for your project.
=>	Never change original source files — edit the copies in your custom source folder.
=>	Tell the Potato where to look for source files (`Control + Alt + Shift + S`).
=>	Pick a "`handle`" and use it to identify all your modifications (e.g. "`MDH_Custom`")
=>	Comment out original code, but leave it there rather than deleting it.
=>	Explain your changes in comments.
=>	Maintain standards-compliance (& validate!).

Preparing for your customization tasks

=>	On your hard drive, create a special source folder for your project. Call it "`hp_custom_source`".
=>	Invent a "handle" to identify all your modifications. Make it your initials followed by "_Custom". You will use this to mark all the changes you make to the source files, so you can easily find them again in the future.

=> Start JQuiz, and tell the program where to look for your custom source files. (Press `Control + Alt + Shift + S`, then find and select your `hp_custom_source` folder.)

Task 1: Making the reading text scroll independently

=> Find the Hot Potatoes source folder, and copy the `hp6.cs_` file into your custom source folder.

=> Open the `hp6.cs_` file in your text editor. (Make sure you open the copy in your custom folder, not the original one!)

=> Search for this text:

```
.ReadingText
```

This is the beginning of the code which handles the appearance of the reading text div.

=> The code for this selector looks like this:

```
.ReadingText{
  text-align: left;
}
```

All it does right now is to make the text of the reading left-aligned.

=> Type a return character after the line `text-align: left;`. You're going to insert your code starting on the next line. Type a slash followed by an asterisk (a star). This marks the beginning of a comment. Then leave a space, and type your "handle", and a short explanation of what you're doing, like this:

```
/* MDH_Custom: added two lines to make reading text
scroll independently. */
```

Don't forget the `*/` at the end, to close the comment. Now everything between `/*` and `*/` has been commented out, so the browser will ignore it, but we can easily find our code again when we need to, because we can search for the handle.

=> Now type a return, and the following three lines:

```
  overflow: auto;
  height: 15em;
/* MDH_Custom: end customized code.*/
```

We're telling the browser to fix the height of the reading text container to 15 "em" characters, and if the text is too long for that ("overflow"), to handle it automatically. This will cause the browser to add a scrollbar.

=> This is what you should see:

```
.ReadingText{
  text-align: left;
/* MDH_Custom: added two lines to make reading text
scroll independently. */
  overflow: auto;
  height: 15em;
/* MDH_Custom: end customized code.*/
}
```

=> Save the file, then create an exercise in JQuiz with a long reading text, and view it in your browser. Here's an example.

Task 2: Removing styling from buttons

=> If you haven't already done this for Task 1, find the Hot Potatoes source folder, and copy the `hp6.cs_` file into your custom source folder.

=> Open the `hp6.cs_` file in your text editor. (Make sure you open the copy in your custom folder, not the original one!)

=> Search for this text:

```
/*BeginNavBarStyle*/
```

This is the beginning of the code which handles navigation bar appearance.

=> The first selector, `div.NavButtonBar`, controls the navigation bar itself. We don't want to change that, so move below it to the next selector, `.NavButton`.

=> On the line before before `.NavButton`, type a slash followed by an asterisk (a star). This marks the beginning of a comment. Then leave a

space, and type your "handle", and a short explanation of what you're doing, like this:

```
/* MDH_Custom: commented out navigation button styles
to make buttons appear like standard HTML buttons.
```

=> Now scroll down until you see this:

```
/*EndNavBarStyle*/
```

This is where we close our comment. On the line before, type an explanation, then a star followed by a slash, like this:

```
MDH_Custom: end commented-out section. */
```

Now everything between `/*` and `*/` has been commented out, so the browser will ignore it.

=> This is what you should see:

```
/*BeginNavBarStyle*/

div.NavButtonBar{
[inclNavBarColor] background-color:
[strNavBarColor];[/inclNavBarColor]
[...]
}

/* MDH_Custom: commented out navigation button styles
to make buttons appear like standard HTML buttons.
.NavButton {

[...]

MDH_Custom: end commented-out section. */
/*EndNavBarStyle*/
```

=> Save the file, then create an exercise in JQuiz and view it in your browser. Here's an example.

Show All
Index
Next

Task 3: Horizontal multiple-choice answers

=> In the `hp6.cs_` file, search for this text:

```
ol.MCAnswers li{
```

This is the beginning of the code which handles `list items` in the `ordered list` of `multiple-choice answers` in JQuiz.

=> Add your handle and explanation, inside a comment:

```
/* MDH_Custom: Next two lines make m/c answers list horizontally. */
```

=> Add the following line to the code:

```
display: inline;
```

=> Now, because the answers will be next to each other, we need to create some space after each answer to separate them. Add the following line to the code:

```
margin-right: 4.0em;
```

=> This is what you should see:

```
ol.MCAnswers li{
/* MDH_Custom: Next two lines make m/c answers list horizontally. */
  display: inline;
  margin-right: 4.0em;
  margin-bottom: 1em;
}
```

=> Save the file, then create an exercise in JQuiz and view it in your browser. Here's an example.

Show All
Index
Next

Task 4: Using an external stylesheet

=> First, we're going to use the Masher program to create an external stylesheet. Start the Masher, then click on the Appearance tab. Choose the colours and settings you want to use. Then click on `Actions /`

	`Create complete HotPot stylesheet`. Save the stylesheet in the same folder as the exercises you have created. Call it `external.css`.
=>	Find the Hot Potatoes source folder, and copy the `jmix6.ht_` file into your custom source folder. Open this file in your text editor.

=>	Our task here is to prevent JMix from adding the normal stylesheet into the exercise itself. Search for these three lines:
	
	`<style type="text/css">`
	`[strStyleSheet]`
	`</style>`
	
	Delete these lines, and add a comment to explain what you've done:
	
	`<!-- MDH_Custom: Deleted the placeholder for the internal stylesheet, in order to use an external one. -->`
	
	`[strStylesheet]` is the "placeholder" that is replaced by the CSS code when JMix creates the exercise; if it's not there, then the CSS code will not appear in the page.

=>	Start JMix, and tell the program where to look for your custom source files. (Press Control + Alt + Shift + S, then find and select your hp_custom_source folder.)
=>	Now make a little JMix exercise to test with. It doesn't matter what goes in it; "This / is / a / test" will do.
=>	Now we need to insert the link to the external CSS file into the page. Go to the Configuration screen, and click on the Custom tab. In the box at the bottom, "Code for insertion into <head> tag", type the following: `<link rel="stylesheet" type="text/css" href="external.css" />`
=>	Press `OK` to exit the Configuration screen, then export your exercise and make sure you save it in the same folder where you saved the `external.css` file. The exercise should look the same, but it is different. You can prove this by temporarily renaming the `external.css` file so the browser can't find it. Here's an example.

Show All

Task 5: Adding a new navigation button

=> Find the Hot Potatoes source folder, and copy the `hp6navbar.ht_` file into your custom source folder.

=> Open the file in your text editor. You'll see a `<div>` tag containing three separate blocks of code, one for each button on the navigation bar. The third one looks like this:

```
[inclNextEx]
<button class="NavButton"
[...]
onclick="location='[strNextExURL]'; return
false;">[strNextExCaption]</button>
[/inclNextEx]
```

=> For our custom button, we're going to copy some of this code, to create a new button, then modify it. We don't need the include instructions, so ignore them; just copy the `<button>` tag (`<button ... </button>`), and paste it between the last `[/inclNextEx]` and the closing `</div>` tag at the end of the file.

=> The last few lines of your file should look like this:

```
[/inclNextEx]

<button class="NavButton" onfocus="NavBtnOver(this)"
onblur="NavBtnOut(this)" onmouseover="NavBtnOver(this)"
onmouseout="NavBtnOut(this)"
onmousedown="NavBtnDown(this)"
onmouseup="NavBtnOut(this)"
onclick="location='[strNextExURL]'; return
false;">[strNextExCaption]</button>

</div>
```

=> The next thing to do is to comment our changes. Add a comment before and after your new code like this:

```
<!-- MDH_Custom: new button added to navigation bar. --
>
```

```
<button class="NavButton"
[...]
onclick="location='[strNextExURL]'; return
false;">[strNextExCaption]</button>

<!-- MDH_Custom: end of new button. -->

</div>
```

=> Next, we need to change the caption of the button to its new caption. Right now, the caption is a placeholder (`[strNextExCaption]`). Change this to "`HotPot`".

=> Finally, we need to add the URL the button will go to. At the moment, that's another placeholder (`[strNextExURL]`). Change it to this:

```
http://web.uvic.ca/hrd/hotpot/
```

Make sure you don't remove the single quotes surrounding the URL.

=> Now your code should look like this:

```
<button class="NavButton" onfocus="NavBtnOver(this)"
onblur="NavBtnOut(this)" onmouseover="NavBtnOver(this)"
onmouseout="NavBtnOut(this)"
onmousedown="NavBtnDown(this)"
onmouseup="NavBtnOut(this)"
onclick="location='http://web.uvic.ca/hrd/hotpot/';
return false;">HotPot</button>
```

=> Save the file, then create an exercise in JQuiz and view it in your browser. Here's an example.

Show All
Index
Next

Task 6: Making the timer count up instead of down

=> Find the Hot Potatoes source folder, and copy the `hp6timer.js_` file into your custom source folder. Open this file in your text editor. This file contains the code which makes the timer work.

=> First, we need to make the timer start from zero, instead of from the time

limit specified in the configuration screen. Find this line:

```
var Seconds = [intSeconds];
```

Add a comment to explain your change:

```
//MDH_Custom: set the timer to start from zero
```

Notice that this comment begins with two slashes; that's how you make a line into a comment in JavaScript.

=> Next, make another copy of the line of code, so you have two copies. Comment out the first one, then modify the second, so you have this:

```
//var Seconds = [intSeconds];
var Seconds = 1;
```

Now you can see what the line used to be, and what it is now.

=> Next, we need to make the timer count upwards instead of downwards. Find this line:

```
Seconds--;
```

And change it to this:

```
//MDH_Custom: made the timer count upwards instead of
downwards.
//Seconds--;
Seconds++;
```

=> Now make a little JQuiz exercise to test with. Remember to include a timer. It doesn't matter what time limit you set, though; the time limit will not be used. Here's an example.

Show All
Index
Next

Task 7: Hiding and showing the gapfill word list in JCloze

=> First, open the `hp6.cs_` file from your custom source folder in your text editor. Find this section:

```
.ClozeWordList{
  text-align: center;
  font-weight: bold;
}
```

Modify it like this:

```
.ClozeWordList{
  text-align: center;
  font-weight: bold;
/*MDH_Custom: added the next line to hide the word list
initially.*/
  display: none;
}
```

=> Find the Hot Potatoes source folder, and copy the `jcloze6.ht_` file into your custom source folder. Open this file in your text editor.

=> Find this block of code:

```
<div id="WordsDiv" class="StdDiv">
<span id="WordList"
class="ClozeWordList">[strWordList]</span>
</div>
```

Add a button and a linebreak to the code, like this:

```
<div id="WordsDiv" class="StdDiv">
<!--MDH_Custom: added a button to show and hide the
word list.-->
<button onclick="ShowHideWords()">Show/hide
words</button><br />
<span id="WordList"
class="ClozeWordList">[strWordList]</span>
</div>
```

=> Finally, we need to add the code which hides and shows the wordlist when we click on the button. We can do this at the top of the file, by adding a special JavaScript tag containing the code. Scroll up to the top of the file, and find this:

```
<script type="text/javascript">
//<![CDATA[
<!--
```

This is the tag into which JCloze inserts all the JavaScript code. We can add our code immediately after this. First, add your explanatory comment:

```
//<![CDATA[
<!--
//MDH_Custom: Added a function for showing and hiding
the answer list.
```

=> Now add this JavaScript function right after your comment:

```
function ShowHideWords(){
  var W = document.getElementById('WordList');
  if (W.style.display != 'block'){
    W.style.display = 'block';
  }
  else{
    W.style.display = 'none';
  }
}
```

=> Now make a little JCloze exercise to test your code. Remember to check the option to "Include word list with text" in the configuration screen. Here's an example.

Show All
Index
Next

Task 8: Controlling question navigation (1)

=> Find the Hot Potatoes source folder, and copy the `jquiz6.ht_` file into your custom source folder.

=> Open the file in your text editor, and search for the XHTML code for the `Show All Questions` button. It's inside a <p> tag, and it looks like this:

```
<p style="text-align: right;">
<button id="ShowMethodButton"
[...]
onclick="ShowHideQuestions(); return
false;">[strShowAllQuestionsCaption]</button>
</p>
```

=> Comment out the whole button, adding your handle and comment at the

same time, like this:

```
<!-- MDH_Custom: This button is commented out to hide
it. -->
<!-- <p style="text-align: right;">
<button id="ShowMethodButton"
[...]
onclick="ShowHideQuestions(); return
false;">[strShowAllQuestionsCaption]</button>
</p> -->
<!-- MDH_Custom: End of section commented out. -->
```

=> Save the file, then create an exercise in JQuiz and view it in your browser to check that the button has disappeared. If everything works OK, then move on to the second part of the task on the next page.

Show All
Index
Next

Task 8: Controlling question navigation (2)

=> Find the Hot Potatoes source folder, and copy the `jquiz6.js_` file into your custom source folder.

=> Open the file in your text editor, and search for the JavaScript `ChangeQ function`. It's about a quarter of the way through the file, and it starts like this:

```
function ChangeQ(ChangeBy){
```

=> Look at the next three lines. They all start with two slashes (//). This is one way to make a comment in JavaScript; two slashes at the beginning of a line will comment out the whole line. These are the lines:

```
//The following line prevents moving to another
question until the current
//question is answered correctly. Uncomment it to
enable this behaviour.
// if (State[CurrQNum][0] == -1){return;}
```

=> First, we're going to add our own comment to explain what we're doing. Add it after the first two lines, and before the JavaScript line:

```
//The following line prevents moving to another
```

```
question until the current
//question is answered correctly. Uncomment it to
enable this behaviour.
//MDH_Custom: Uncommented the next line to enable
behaviour described above.
// if (State[CurrQNum][0] == -1){return;}
```

=> Finally, we just need to uncomment that line of code to make the page work the way we want it to:

```
//The following line prevents moving to another
question until the current
//question is answered correctly. Uncomment it to
enable this behaviour.
//MDH_Custom: Uncommented the next line to enable
behaviour described above.
if (State[CurrQNum][0] == -1){return;}
```

=> Save the file, then create an exercise in JQuiz and view it in your browser. **Remember NOT to set the exercise to "Shuffle questions". If the questions are shuffled, this hack won't work.** Here's an example.

Show All
Index
Next

Task 9: Using an image instead of a button

=> First, we need to get the image we're going to use instead of the button. You can find a good one here:

http://web.uvic.ca/hcmc/clipart/school/correct/correct-vt.gif

Save the image to the folder on your hard drive where you are creating exercises.

=> Now open the `jcloze6.ht_` file from your custom source folder in your text editor. Find this section, which is the check button below the exercise:

```
<button id="CheckButton2" class="FuncButton"
onmouseover="FuncBtnOver(this)"
onfocus="FuncBtnOver(this)"
onmouseout="FuncBtnOut(this)" onblur="FuncBtnOut(this)"
onmousedown="FuncBtnDown(this)"
onmouseup="FuncBtnOut(this)"
```

```
onclick="CheckAnswers()"> [strCheckCaption] </button>
```

=> Now add an explanatory comment, and also comment out the existing button, by putting `<!--` before it and `-->` after it. You should see something like this:

```
<!-- MDH_Custom: Commented out the existing button to
use an image instead.-->
<!-- <button id="CheckButton2" class="FuncButton"
onmouseover="FuncBtnOver(this)"
onfocus="FuncBtnOver(this)"
onmouseout="FuncBtnOut(this)" onblur="FuncBtnOut(this)"
onmousedown="FuncBtnDown(this)"
onmouseup="FuncBtnOut(this)"
onclick="CheckAnswers()"> [strCheckCaption] </button>--
>
```

=> Now we're going to insert the image itself. Add this code below the old button:

```
<img src="correct-vt.gif"
alt="[strCheckCaption]"
title="Check your answer"
onclick="CheckAnswers()"
style="cursor: pointer;" />
```

=> Now add a comment below, to show where your modifications end:

```
<!-- MDH_Custom: End of modification to Check button.--
>
```

=> Now save your changes, and make a little JCloze exercise to test your code. It would be a good idea to uncheck the "Include 'Hint' button" checkbox in the configuration screen, so that you don't have an ordinary HotPot Hint button right next to your new image. Here's an example.

Show All
Index
Next

Task 10: Branching based on score

=> Find the Hot Potatoes source folder, and copy the `hp6hotpotnet.js_` file into your custom source folder.

=> Open the file in your text editor, and search for the JavaScript `Finish function`. It's about a quarter of the way through the file, and it starts like this:

```
function Finish(){
```

=> This function is useful to us because it's always called at the end of an exercise. If we want to insert any behaviour at the end of an exercise, we can put it inside this function. [Note: if you're going to post exercises on www.hotpotatoes.net, it's best not to mess with this function, in case you disrupt the submission of results!]

=> The best place to insert our code is at the beginning of the function, right after the line that starts with "function". This is the code to insert:

```
if (Score>= 75) {
  alert("Well done! You passed!");
  window.location="http://www.google.com";
}
else{
  alert("Sorry! You scored less than 75%. Try again.");
  location.reload();
}
```

=> Finally, we just need to comment our code:

```
function Finish(){
//MDH_Custom: Redirect the student based on score.
if (Score>= 75) {
  alert("Well done! You passed!");
  window.location="http://www.google.com";
}
else{
  alert("Sorry! You scored less than 75%. Try again.");
  location.reload();
}
//MDH_Custom: End of custom redirect code.
```

=> Save the file, then create an exercise in JQuiz and view it in your browser. Here's an example.

Index
Next

What you've learned in this workshop

Source files	Source folders	Techniques
• 3 types of source file • how placeholders work • how includes work	• source folder locations • how to use a custom source folder	• Commenting out CSS, XHTML and JavaScript • Adding new CSS code • Adding new XHTML code • Adding new JavaScript code • Explaining your changes in comments

Index
Next

Useful resources

- **Online version of this tutorial: http://web.uvic.ca/hrd/hotpot/howto/hacking_workshop/**
- **Editing Hot Potatoes source files (online tutorial): http://web.uvic.ca/hrd/hotpot/howto/editsource.htm**
- **Hacking in Hot Potatoes (online article): http://web.uvic.ca/hrd/hotpot/howto/hacking_hotpot.htm**
- **Documentation for Hot Potatoes Source Files and Placeholders (online reference page for source files and placeholders): http://web.uvic.ca/hrd/hotpot/howto/sourcefiles.xml**
- **W3Schools CSS tutorial: http://www.w3schools.com/css/**
- **W3Schools XHTML tutorial: http://www.w3schools.com/xhtml/**
- **W3Schools JavaScript tutorial: http://www.w3schools.com/js/**