Template for Mapping Your Activities to Learning Outcomes

See next page. This page could not be removed, strangely.

Learning Outcome	Evidence of Meeting the Learning Outcome	Your Own Assessment of the Grade You Believe Would Be Appropriate	Tutor's Justification of Grading (optional)
Apply a structured approach to identifying needs, interests, and functionality of a website.	I have used the personas and scenarios technique, which I like to think of as stageplay, to the process of site design after identifying the themes and purposes of the website I would like to create. I have written a detailed reflective learning diary, which contains specific information mapped to this learning outcome. The reflective learning diary is available here: https://landing.athabascau.ca/blog/view/15441799/unit-1-site-design-design-documentation-submission. There is also a wiki page that I created on The Landing, which is also linked to in the reflective learning diary entry above.	A	
Design dynamic websites that meet specified needs and interests.	I found a script which helped me implement a global site directory, which I saved as `accessibleMenu.js` in my `vendor/js/`	A	

	folder. I integrated the script into my website, applying it to every page. I also improved the CSS that was included by allowing myself to have nested lists in the site directory menu.	
Write well-structured, easily maintained, standards-compliant, accessible HTML code.	I have written many HTML files that have a common structure, and follow modern, HTML5 semantics. The code is accessible, being readable and also including as much of the accessibility standards and compliance with WCAG as I had time to study. An example is proper, semantic captions for all images on my website: see lines 225-235 of /index.html for an early example. I extended the HTML example provided by the original author of accessibleMenu.js to include a nested unordered list, which was otherwise not well supported and not styled well. The HTML produced is accessible and compliant, and the JavaScript which manipulates	A

	maintains this accessibility through the two states the element can be displayed in (for visual users).	
Write well-structured, easily maintained, standards-compliant CSS code to present HTML pages in different ways.	I have written several LTR sized pages worth o f CSS. I have a single root CSS file living in /css/root.css. This file would be better split into its component parts, but because it is only three printed pages worth it is still very navagable given the commented, styled headers I inserted to separate different parts of the document. The code is well-structured, as every line is less than eighty characters, except a couple comments. The indentation is consistent, and the use of tabs vs spaces is consistent. The code is mostly standards compliant, and is maintainable because various parts of the document are commented, such as those with “NOTE” in comments, or “DONE”A in comments. NOTE specifies a note.	A

	DONE specifies some special note that I've made **after** a task is completed, or after some discovery prompting the note; usually, DONE means that I'm done experimenting with enabling and disabling some declarations on various elements, and I've discovered what's what and want. I wrote CSS which effectively applies padding-left values to different levels of unordered lists in my site directory menu. If a nested list is present, its label will be styled the same as any other top-level page; the pages which are linked to by the list items in the nested list are styled such that they are further indented, so that the hierarchy of the website is readily visible in the menu.	
Use JavaScript to add dynamic content to pages.	I included an approximately thirty-line script found through a web developers blog (which linked to their codepen project for that code) to add a dynamic site directory to all my web pages.	A

Critique JavaScript code written by others, identifying examples of both good and bad practice.	I critiqued the JavaScript written by James Evers, the original author of the first JavaScript code I integrated into my website. My critique was short, as the code was very standards compliant, modern, well-structured, indented, and properly cased ("lowerCamelCaseNotation" was used). The only practice I could disagree with in the script I integrated was the use of a nested if statement when each of the conditions were short, and when there was no default behaviour when only the outer condition was true but the inner condition was false (hence, syntactically it was an AND situation). I corrected that in my critique, but did not bother to in my use of the code.	
Select appropriate HTML, CSS, and JavaScript code from public repositories of open source and free scripts that improves your site and that enhances the experience of site visitors.	I included an appropriately sized JavaScript source file from a public repository (Codepen.io). The file, accessibleMenu.js, allowed me to implement a global site directory menu,	A

	part of my design documentation. This enhanced the experience of my site visitors, regardless of their ability or their computer. The script was very accessible, using standards compliant ARIA attributes of HTML elements to indicate to screen readers and other accessibility software that the site directory element was opened or closed.	
Modify existing HTML, CSS, and JavaScript code to extend and alter its functionality, and to correct errors and cases of poor practice.	I have rewritten sample HTML documents one and two, which were provided in Unit 2. The full details of the corrections, and my critique of the HTML, noting the errors and poor practice which I corrected, are detailed in my blog post which are in the file `/blog/unit2reflections.html.` `I have extended (by placing my own CSS modifying the classes created by the original author of accessibleMenu.js) the CSS of accessibleMenu.css by placing my own code in /css/root.css to make the menu centerd along with my other menu`	A

	`items, wider, and to properly indent and style nested list content for my blog entries.`	
Write well-structured, easily maintained JavaScript code following accepted good practice, including		A, B, C, D
• general appearance and form: commented, properly laid out, appropriate capitalization		A, B, C, D
• structure: modular, using functions and objects effectively		A, B, C, D
• standards-compliant		A, B, C, D
• accessible		A, B, C, D
Write JavaScript code that works in all major browsers (including IE, Mozilla-based browsers such as Firefox, Opera, Konqueror, Safari, Chrome).		A, B, C, D
Effectively debug JavaScript code, making use of good practice and debugging tools.		A, B, C, D
Use JavaScript libraries (e.g., JQuery) to create dynamic pages.		A, B, C, D
Use JavaScript to access and use web services for dynamic content (AJAX,		A, B, C, D

JSON, etc.).	
Overall	**A, B, C, D**