Diary Entry

Link to SCIS website: http://student.athabascau.ca/~matsph/

Work for this Unit:

Pre-coding Activities:

Skimmed the Unit 6 JQuery FAQ

Skimmed many videos to find useful features of JQuery methods and classes

Read the 'Using jQuery Core' chapter of the JQuery learn site <http://learn.jquery.com/>.

Step 1: JQuery proposal

Enhancements:

1. Drop-down menus for each of the main page buttons with sub-pages/categories.
   The 'Solving Methods" menu button will have drop-down buttons for beginner, advanced, and other methods.
   The 'Solving Records' menu button will have drop-down buttons for Worldwide and U.S. results.
   The 'Facts and History' menu button will have drop-down buttons for 'Facts' and 'History'.
   - This would help Sean to look through each of the main pages of the site and access the beginner methods directly, and would help Robert to go to the advanced solving methods from the home page, without having to click on the 'Solving Methods' link first.
2. Color/theme change option for website. A settings icon can be clicked to show radio buttons, icon buttons, and drop-down menus that can change settings like background-color, foreground-color, enable/disable images, theme, and font.
   - This would improve the browsing enjoyability of Janice, who personally likes exploring different options and enjoys customizing things.
   - Sean could change the color or theme to his liking, to make browsing the site more attractive to him, so he can more easily take up an interest in cubing.
3. Moveable navigation. Allow the user to drag the navigation bar on the left side of the other puzzles and special patterns pages, and move it around and un-dock it from the side to make more room for the tables.
   - This would help the personas in Unit 1 search for other puzzles and special patterns more easily now that they have more room, and can move the navigation out of the way. Janice and Robert can look for what interests them or they think is cool in a more customized and potentially more efficient way.
4. Resizable message box on the contact page. In the message box for the 'Message to send to fould12@gmail.com', allow the user to resize it with their mouse using a drag-tab at the bottom-right of the message box.

- This would increase the efficiency and accessibility of the contact page, making the contact page do what it was originally intended to do in Unit 1, which was allow people to Contact people apart of the site team, but in a better and easier way.

5. Accordion for the facts and history page, starting expanded, but collapsible.
    - Makes the facts and history page illustrated in the Unit 1 site mock-up more navigatable.
6. Replace the header on each page with JQuery HTML insert of the header from a single source.
    - This would remove duplicate code, which is better for editing and saving on storage space and code complexity.
7. Replace text selecting in JQuery sort and filter functions with more simple JQuery code.
    - This would simplify the JavaScript files to make them easier to manage and understand.
8. Make each item in a table hide-able/collapsible.
    - This would help the personas in Unit 1 search for other puzzles and special patterns more easily now that they can eliminate what they don't like from the page. Janice and Robert can customize and filter their searches for the right puzzles and patterns even more with this new feature.

Is this too much JQuery work? Are they all good ideas?

I decided to do options [list options picked].

Step 2: Coding

5. **Accordion:**

I first decided to work on the accordion for the facts and history page, because they seemed like the easiest.

First, I looked up 'Jquery accordion' in YouTube, and found this by thenewboston:

https://www.youtube.com/watch?v=vwJ6LKIVt5Y

I then added the HTML script loading tag for the JQuery file into the header of every html page (since the JQuery I will be making later will be changing the DOM on all pages). I used an online version, since that's the one I already used for my Unit 4 "floater.js" script.

I pondered using a local copy of JQuery, but I decided against it, as the only advantages were that the source code wouldn't change, and the website wouldn't need to externally load a file. The disadvantages for me were greater, as I would need to maintain my own copy of the source code, which could become outdated, and I would have to store it with the rest of my JavaScript files, as well as reference it correctly, which for me was much more difficult than just referencing the online version.

I also added a loading tag for another JavaScript file called "changer.js", which is to contain all JQuery scripts that edit or work with every page.

After that, I added a JavaScript file called “accordion.js” for the accordion on the facts and history page; I also changed the header to contain the new facts and history pages.

Then, I made the facts and history page; mostly modeled off of the index page, with some inspiration from other_puzzles.

I then continued to watch the tutorial by thenewboston, and found out I needed to add the JQuery UI; so I added an extra loading tag for the UI, getting it from the same place as I got the main JQuery file.

I watched the rest of the video, adding the headings and div tags according to the way he said to lay it out, and then I tried it, and it didn’t work.

I tried copying everything he had, in the exact way he did it. Same header tags, same anchor tags, same id names, and nothing worked. So I looked up another example of an accordion, and found this:

https://jqueryui.com/accordion/

I saw that what I did wrong was not embed the JQuery command inside a function; after doing that it worked. I also added the JQuery UI CSS loading tag, which made the accordion feature much more clean.

Then, I thought that it needed better functionality, since at this time only one tab of the accordion could be open; with no option for both or neither. So I watched more of thenewboston’s tutorial series, and in this video:

https://www.youtube.com/watch?v=uHixbmuVfak

He shows how to make all tabs collapsible, which is what I want, since the point of it was to decrease the need for navigation between the areas. I added the option, along with the setting for all accordion tabs to be closed at the beginning, so that the user will have access to all tabs in easy access as soon as they enter the page.

3. Moveable navigation:

This seemed the second-easiest, so I thought it would be a good step-up. I again used thenewboston’s tutorial series, starting here:

https://www.youtube.com/watch?v=WLdlB76wqv0

Before I did or watched anything, I added JQuery loading tags into the other_puzzles and special_patterns pages, and created a new JavaScript file called “dragger.js”.

I continued to watch the tutorial, added “drag” id names to the nav tags to make them identifiable by JQuery, added the draggable functionality, containment within the document, and added a moving cursor.

But now I wanted to add an ‘un-dock feature’, which requires changing of CSS, and stopping the ‘floater.js’ from changing the class name of the ‘nav’ element. To do this, I added a check to the ‘floater.js’ so that it will check if the ‘nav’ has been un-docked by seeing if it has a class named ‘un-

docked'. I also gave the html 'nav' tag a class attribute named 'docked', and changed the "dragger.js" to remove the 'docked' class when the draggable function is started.

Other code I added to the daraggable start function was CSS reformatting for the table, and fixed positioning for the 'nav' element.

4. Resizable message box:

This seemed like a fun and useful feature, especially when making a large message. As with the other enhancements, I used thenewboston's tutorial series, starting here:

https://youtu.be/JL5oHGxovFk

Before I watched it, I first added the JQuery loading tags into the other_puzzles and special_patterns pages, and created a new JavaScript file called "resizer.js". I also added the id 'mailform' to the contact form on the contact page.

I after about 15 minutes struggling because I misspelled "resize[r].js", I got it to work. I also set the containment to parent so it can't be stretched outside the form, and the maximum and minimum height to around the normal input-box height.

6. Replace the header on each page with JQuery HTML insert of the header from single source

This does not really enhance the look or functionality of the site, but it just makes the code more clean and manageable, which I think is very important.

To find out how to do this, I looked up on Yahoo! Search "jquery add html from file" and found this:

http://stackoverflow.com/questions/8277462/jquery-append-external-html-file-into-my-page

I used the '$.get' and '.append' JQuery functions as used by Lloyd:

http://stackoverflow.com/a/8277537

I put this in the 'header.js' file, since it is included on every page, and that was its original function.

So I first used the '.append' function he used, but it didn't seem to work. So I then tried the '.load' function, which looked to be better for my purpose, but it still didn't work. I tried debugging it, but the debugger wouldn't dig into the JQuery document to show me what was going on, but it at least showed that the code executed without error. I finally found out that I forgot to add a '#' in the specifier of the tag to be targeted by the JQuery. After I fixed that it worked, and I removed the html inside the header div for all pages, since they were now being loaded automatically!

How I have met the Learning Outcomes:

Be able to use JavaScript libraries (e.g. ,JQuery) to create dynamic pages.

In my website, I have utilized JQuery in every '.js' file except for 'sorter.js' I have used many JQuery functions and properties, including:

- Accordion: With collapsible and active attributes
- Draggable: With containment, cursor, and a custom start function
- removeClass: To remove class attributes from elements
- css: To change/add CSS to elements
- offset: To get the vertical offset of an element
- scroll: To run a custom function when the window is scrolled
- scrollTop: To find the current scrolling position of the window
- hasClass: To check if an element has a class
- load: To load external information into a file
- ready: To run a function once the document is finished loading
- resizable: With containment, minHeight, and maxHeight attributes

This has added much functionality to the site, and has used an array of different features JQuery offers.

Other Requirements

Add dynamic content to your page using the JQuery library that satisfies the requirements identified in Unit 1 and developed since then.

Scenarios:

Persona 2: Janice Miranda

Scenario 1: Finding special Rubik's Cube patterns

The navigation can now be undocked from the side, giving her the ability to make more room for the table, and customize her search more.

Scenario 5: The 5*5*5

The navigation can now be undocked from the side, giving her the ability to make more room for the table, and customize her search more.

Persona 3: Robert Connelly

Scenario 6: Buying a new puzzle

The navigation can now be undocked from the side, giving him the ability to make more room for the table, and customize his search more.

Mock-up:

Facts and History: The facts and history page now is more easily navigable with the addition of accordion tabs. This allows the user to have easy access to the 2 categories.

Contact: The message form is now expandable, allowing the user to have more control over the message they can edit and send.

Look at and comment on the JQuery work of other students

I made comments on these posts:

https://landing.athabascau.ca/blog/view/2248131/unit-6-learning-diary

https://landing.athabascau.ca/blog/view/2180496/unit-6-learning-journal

https://landing.athabascau.ca/blog/view/2151205/unit-6-libraries-learning-diary-reflections

https://landing.athabascau.ca/blog/view/2117642/unit-6-jquery-coding-journal-entry

Notes:

Went well:

The draggable navigation and external header file replacement went well. They didn't have many errors, and they gave functionality quickly and rapidly.

Didn't go well:

The resizable input box for the contact page didn't go as well, as I thought that I would be able to make it a multiline input box, but that wasn't possible unless making it a textarea box, so trying to add the feature took some time, and I decided instead just to make the box horizontally resizable.

What was most difficult and why:

The accordion feature was the most difficult, because it was the first JQuery feature file I set up, and I had many bugs while loading the external JQuery files, along with the JQuery UI file and CSS file.

If done again, would it be done differently and why:

Yes, I would have found a way to put all of the JQuery functions into one file, and reference them all inside the html pages for which one I wanted. I chose to have individual files this time because I thought it would be more difficult to load them all from separate functions in one file, but I later found out this is probably not true.

How did previous experience help/hinder completing the tasks:

Previous experience with JQuery loading from Unit for helped complete the task, as I used many of those functions to help me with the JQuery structure and features.

Most surprising thing learned:

JQuery has custom CSS and UI abilities with their own customizable icons and animations, which goes beyond what I thought JQuery was composed of.

Most useful thing learned:

JQuery has customizability for almost every property of its custom functions, allowing each feature to be customized and adapted to many unique and different situations.

Map to Course Outcomes

In the original site mock-up, all of the header sections of each of the pages were to be exactly identical, unchanging from page-to-page. This became hard to maintain as the number of pages of my site grew, so the file ‘header.js’ with the JQuery inside helped eliminate code duplication, and ensure all headers are identical like in the site mock-up.

Also, for the files ‘accordion.js’ and ‘resizer.js’, I have included descriptions mapping the features of those files to the site mock-up from unit 1 in my learning diary. These mappings can be found in the ‘Other Requirements’ section under the subsection ‘Mock-up:’

The ‘dragger.js’ file is meant to add functionality to the navigation illustrated in the ‘Other Puzzles’ mock-up. The functionality wasn’t originally intended, but it fits with the original purpose of helping the user navigate through the different puzzles in their own customizable way.

Self-assessment:

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 made 7 scenarios that give structured processes of how the anticipated audience will address their needs with the website. I also took into account the constraints users might have, such as different operating systems, slow computers, and browsing the site using mobile devices.	B	
Design dynamic websites that meet specified needs and interests.	The JavaScript code in the ‘floater.js’ and ‘sorter.js’ files give the website dynamic content, which meets the needs and interests of the personas as described in ‘Other Requirements’ in the diary entries of Unit 4 and Unit 5.	A	

Write well-structured, easily maintained, standards-compliant, accessible HTML code.	Sample1.html and Sample2.html are edited to show my ability to write standards-compliant and structured HTML. The errors that used to exist in the code are listed in the learning diary under the section “Other Requirements”. The 11 pages I wrote for my website all have links easily clickable on a mobile device, and are all well-structured and standards-compliant.	B	
Write well-structured, easily maintained, standards-compliant CSS code to present HTML pages in different ways.	The CSS I wrote is either in the external CSS file, or in a one-time internal CSS in the ‘other_puzzles.html’ page. In both, the code is organized with comments and well-structured using proper spacing, character case, property ordering, and naming schemes. It is easily maintained as the external CSS file is less than 130 lines long, and the internal CSS only defines styling for one selector. It is standards compliant with WCAG 2.0 by having no obscure or hard to read text, and proper contrast.	A	
Use JavaScript to add dynamic content to pages.	I have added the JavaScript file ‘floater.js’ which dynamically changes the positioning of the navigation field depending on where the user’s viewport is.	B	
Critique JavaScript code written by others, identifying examples of both good and bad practice.	I modified the code written by LearnWebCode as described in the ‘How I have met the Learning Outcomes’ section of the diary entry for Unit 4.	A	
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 selected JavaScript code from LearnWebCode as described in the ‘How I have met the Learning Outcomes’ section of the diary entry for Unit 4, which meets the needs and interests of the personas as described in ‘Other Requirements’ in the diary entry for Unit 4.	C	

Modify existing HTML, CSS, and JavaScript code to extend and alter its functionality, and to correct errors and cases of poor practice.	I edited the templates given in Unit 2 as described in the Unit 2 learning diary under the first requirement under the section 'Other Requirements'. I also modified the code written by LearnWebCode as described in the 'How I have met the Learning Outcomes' section of the diary entry for Unit 4.	C	
Write well-structured, easily maintained JavaScript code following accepted good practice, including	I have evidence of this listed below, as well as in the Unit 5 diary entry under the 'How I have met the Learning Outcomes' section, which both describe how I checked, edited, and wrote the JavaScript code.	B	
• general appearance and form: commented, properly laid out, appropriate capitalization	Using my Unit 4 notes on JavaScript code conventions as well as the HTML comments section of my Unit 2 notes, I added comments and formatted the code according to standards. I also used CamelCase notation, with lowercase first letters of variables.	B	
• structure: modular, using functions and objects effectively	I structured the code according to notes on proper spacing given in The Landing FAQs and the Moodle study guide. I also used 2 functions, 1 custom function, and 29 variables in the 'sorter.js' program code.	B	
• standards-compliant	I tested the sorting and filter elements on Firefox, Opera, Internet Explorer, and Chrome browsers. I also ensured that the new HTML code and JavaScript were compliant with the XHTML 1.0 strict standards and the WCAG 2.0 standards.	B	
• accessible	The website should have an 'A' rating by the WCAG 2.0 standards for the most part, and the new changes to the website have also been tested on an Android device.	B	
Write JavaScript code that works in all major browsers (including IE, Mozilla-based browsers such as Firefox, Opera, Konqueror, Safari, Chrome).	I tested the 'sorter.js' JavaScript code in the Firefox, Chrome, and Opera browsers, as well as on an android tablet.	B	

Effectively debug JavaScript code, making use of good practice and debugging tools.	I used the built-in Firefox debugger, by going line-by-line through code until it reported an error, which I read and researched to find where my code had a bug. I used the breakpoint feature to stop the execution of the code at the point where I wanted to see the execution in detail. I also kept track of the variables in the function scope in the variables tab on the right side of the debugger.	B	
Use JavaScript libraries (e.g., JQuery) to create dynamic pages.	I have used JQuery in 5 different JavaScript files and functions, and used over 11 functions and properties of JQuery, as listed in the Unit 6 diary entry under the section “How I have met the Learning Outcomes”.	D	
Use JavaScript to access and use web services for dynamic content (AJAX, JSON, etc.).		A, B, C, D	
Overall		A, B, C, D	

JQuery Notes

JQuery is a JavaScript library that contains pre-written scripts having objects and functions.

To use internal JQuery on a page, add this line:

```
<script type="text/javascript" src="scripts/jquery.js"></script>
```

JQuery functions start with a dollar sign ($).