Neil Thawani

E-Learning Design Principles

Self-Directed Project
Final Project Report and Reflection

Performing CRUD Operations on the DOM Using JavaScript

9 December 2019

Table of Contents

Motivation

The motivation for this module was initially to create a course that taught students HTML and CSS using a curriculum developed by the W3C. I drafted an initial proposal and located resources to teach students basic web development concepts. However, one evening I was speaking to a friend in the MHCI program who is enrolled in Dr. Jason Hong's Programmable User Interfaces (PUI) course, who was struggling to complete Lab Assignment #6 (hereafter referred to as LA6). In LA6, students were instructed to create an e-commerce page with a set of items, "Add" buttons for each item, and a shopping cart. The main performance task of LA6 was to write code so that, when users clicked the button, it would add a quantity of items to the shopping cart. After a conversation with this student, I changed the scope of this instructional design project to focus on teaching students how to complete LA6. This "Add to Cart" task was a milestone in LA6 that was challenging to this student for a number of reasons, which were elucidated over the course of this project.

Although one of the prerequisites for PUI is an Introduction to Python course, students have a hard time transferring concepts and skills from Python programming to JavaScript and the web development medium. The breadth of this project, then, was calibrated to target students who have little to no JavaScript development experience. Concepts such as scope and variable type were defined as learning objectives in the first iteration of this module with the intent of teaching transferable skills to students who want to pursue roles in frontend software development. Initially, I was advised by Elizabeth "Mimi" McLaughlin that I should be wary of my expert blind spot, as I initially thought that my career foundation as a frontend developer would aid me in designing this course. Thus, I reached out to Dr. Jason Hong and his Teaching Assistants (TAs) for the course to get advice about difficulties prior PUI students had encountered during this and related JavaScript assignments.

My initial objective, before designing instruction, was to draft a set of learning goals for students to acquire in order to successfully complete LA6. These goals were written with the intent of enabling students to develop the skills to engage in near transfer performance to LA6.

Initial Goals and Assessment

The initial draft of goals for this instructional module were developed to cover the scope of LA6.

Concepts	Skills	Dispositions
1. When to use localStorage (sometimes) vs. Global Variables	1. Binding JavaScript functions to HTML buttons which, when clicked,	1. Confident in debugging problems that may arise when debugging the DOM

(almost never). 2. Variables types, loose and strict equality 3. Variable type coercion/parsing 4. The difference and applications of JavaScript scope and context	modify the DOM	2. Knowledgeable about and able to communicate the issues involved with designing and building with web standards.

Table 1. Initial Draft of Goals

This initial draft, however, was insufficient in a number of ways. It discounted novice HTML/JavaScript learners, neglected to target teaching the concepts related to "creating, reading, and updating the DOM in JavaScript," and required a breakdown of the component skills relevant to this task. I was advised to "articulate goals in the language I want students to use." This resulted in a major revision of my goals after just a week of planning.

Concepts	Skills	Dispositions
● C0: document is the top-level object for web pages in the browser ● C1: the difference between classes and IDs in HTML ● C2: how to retrieve elements from the DOM ● C3: the three main primitive variable types in JavaScript (boolean, number, string) plus arrays ● C4: the difference between loose (==) and strict (===) equality in JavaScript ● C5: how to find the type of a variable in JavaScript (typeof variableName, variableName.constructor)	● P0: retrieve elements by class and ID ● P1: bind JavaScript functions to event attributes in HTML (onclick) ● P2: retrieve the text content within an HTML element (innerText, textContent, outerText) ● P3: apply methods for variable type conversion (String to Number) ● P4: update the text within HTML elements ● P5: retrieve all the methods available to an object in JavaScript	● D0: when they don't know how to move forward in a coding problem, search the methods available to an object in JavaScript ● D1: always refer to Google Search or to the documentation (W3C or MDN) under conditions of ambiguity when they don't know how to move forward in a coding problem

Table 2. Revised Learning Objectives

Core to this instruction were the dispositional goals **D0** and **D1**: discover what you can do with an object in JavaScript by looking at its methods and and always refer to the documentation under conditions of ambiguity. As a professional web developer, keyword and error message search using Google was my immediate strategy for solving coding problems. I knew it would be key to students' success in LA6.

The initial summative pre- and post-test assessment questions for this instructional module were drafted to align with the learning objectives above.

The correct answer for each check-all-that-apply and multiple-choice question question is in **bold**.

What is the top-level object on a webpage in the browser? **document** body window this
How do you get an element by its class name? document.getElementsByClassName("field-value")[0]
How do you get an element by its ID? document.getElementById("unique-name")
What are the three main primitive variable types in JavaScript? **string** **number** **boolean** array
How do you convert a string to an integer? parseInt(value, 10)
What is the HTML attribute(s) used to bind JavaScript functions to events? **onclick** click submit button

What is the difference between loose and strict equality? Loose equality compares variable type. Strict equality compares content and variable type. Loose equality compares content and variable type. Strict equality compares content. Loose equality compares content. Strict equality compares variable type. **Loose equality compares content. Strict equality compares content and variable type.**
What is the operation that loose equality implicitly == performs? Type conversion **Type coercion** Type casting Type script
How do you get the type of a variable in JavaScript? **typeof variableName** **variableName.constructor** console.log(variableName) return variableName;
What are the attributes used on an HTML element to retrieve its text? textValue value **innerText** **textContent** **outerText**
How can you find out what the attributes or methods of an object are? for (var i in variableName) { console.log(i); }
Write a function to change the innerText of an HTML element.

Table 3. Initial Assessment Design

Coupled with this initial goal/assessment design and alignment, I conducted a theoretical cognitive task analysis (CTA) to break down the steps involved with each procedural performance task.

Theoretical CTA

1. Inspect element on the DOM to get the class name or ID.
2. Type the command document.getElementsByClassName or document.getElementById
   a. If class name, access the appropriate index of the array that's returned.

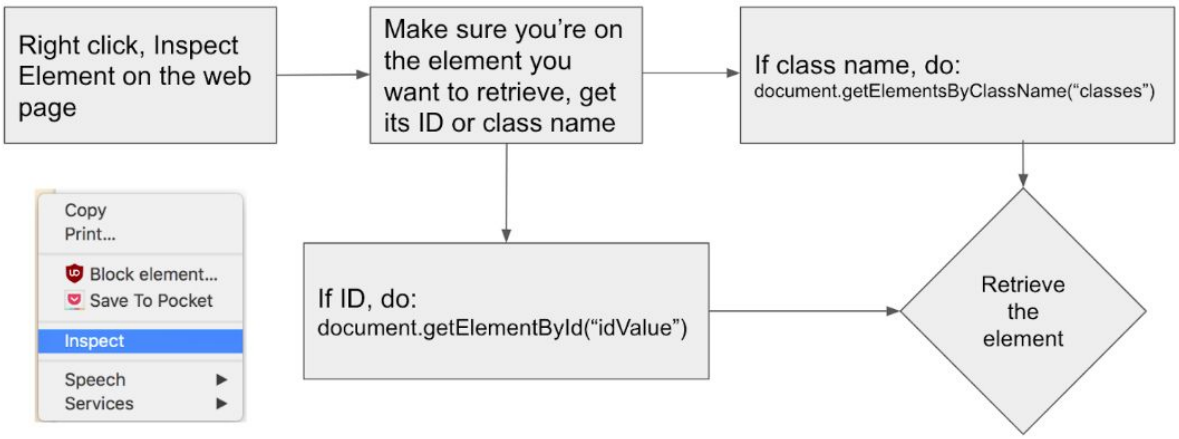


Figure 1. Theoretical CTA of P0: retrieve elements by class and ID

1. Know what the attribute is (onmouseover, onclick, etc.).
2. Place the attribute in the its respective HTML tag and make its value a function.
3. Write the function in a JavaScript file.
4. Import the JavaScript file to your HTML file.
5. Test the element to make sure it works.

KCs:
- Event bindings, specifically onclick
- Attributes, specifically onclick
- Functions in JavaScript
- Importing JavaScript into an HTML file

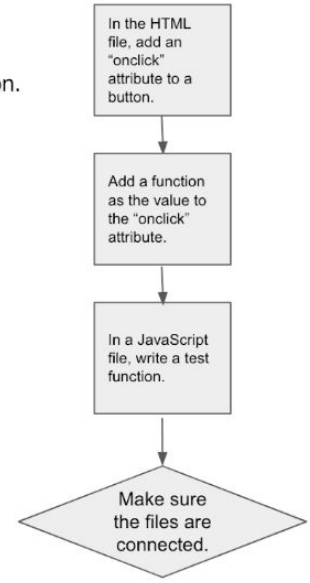


Figure 2. Theoretical CTA of P1: bind JavaScript functions to event attributes in HTML (onclick)

1. Get the HTML element in JavaScript by retrieving it by class or ID and store it in a variable.
2. Do the variable.innerText. Sometimes it will be .textContent or .outerText.

How do we determine what this will be? Enhance this by looking it up or doing an iteration over the object.

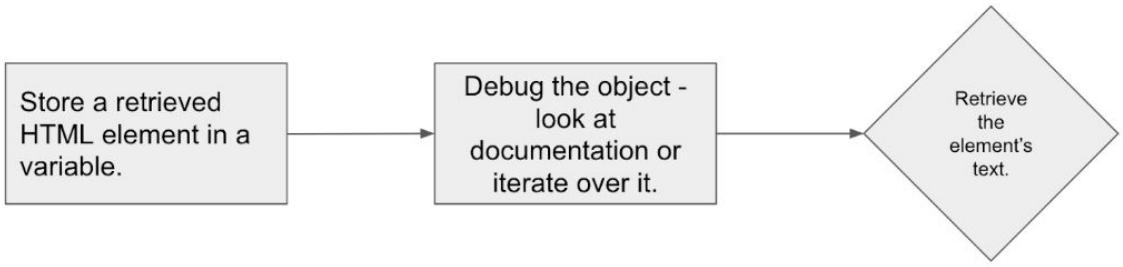


Figure 3. Theoretical CTA of P2: retrieve the text content within an HTML element (innerText, textContent, outerText)

This is a variable condition/constant response KC. It's either a category (non-verbal) or concept (verbal). Type conversion is a concept, but its application is dependent on the context in JavaScript.

var stringAsInt = parseInt(stringVarName)

Component KCs:
- Debugging to verify variable type
- typeof/.constructor to get variable type
- Looking up documentation/Google to find type conversion methods

Figure 4. Theoretical CTA of P3: apply methods for variable type conversion (String to Number)

1. Get the element[s] by doing get element by ID or classNames and save it in a variable.
2. Do variableName.innerText = "newText";

Component KCs:
- Save retrieved HTML elements in variables.
- Get innerText by documentation or Googling. Update the value.

Figure 5. Theoretical CTA of P4: update the text within HTML elements

for (var i in object) { console.log(i); }

or look at the documentation

Component KCs:
- Documentation or Googling for the answer

Figure 6. Theoretical CTA of P5: retrieve all the methods available to an object in JavaScript

Informal Interviews

While I developed models for theoretical CTA, I conducted two informal interviews - one with a current PUI student and another with a PUI TA - to uncover difficulties and opportunities for this instructional module.

The student, who has no prior experience in HTML/JavaScript development, stated that they looked at documentation and "played around a lot" when presented with a problem they didn't know how to deal with. This student also stated that they didn't learn some of the learning objectives intended to be taught by this instructional module, namely loose/strict equality and how to convert variable types in JavaScript.

The TA I spoke to suggested incorporating the Personalization Principle to formative and summative assessment questions, JavaScript code comprehension questions (such as the output of code snippets), and debugging tasks. Because this TA worked with undergraduates enrolled in PUI, their input was extremely valuable. However, it was during this time that I realized that debugging was a core competency that students needed to develop in order to succeed not only in the near transfer task of LA6, but also throughout the remainder of the PUI course and, for some, their careers.

Because of the complex nature of the objectives the PUI TA advised me to implement in this instructional module combined with the input from the PUI student, I decided to incorporate worked examples throughout instruction and a new module three quarters of the way through the unit to scaffold students' knowledge in variable types in JavaScript.

I then conducted empirical CTA with two experts (previous web developers) and one novice, who knew how to read from the DOM using JavaScript, but was unfamiliar with the theoretical concepts I intended to teach in order for students to achieve near transfer to PUI LA6.

Empirical CTA

Expert CTA was fruitful in terms of solidifying the learning objectives for this course. Experts were asked to complete the initial draft of the summative assessment and think aloud while doing so. Several components of the assessment were improved upon as a result of these think-alouds. In addition, novice CTA helped to identify which components of instruction were to be emphasized in order to complete the overall objectives of the course in attaining near transfer to PUI LA6.

Expert CTA identified discovery of attributes or methods of an object in JavaScript as a core Knowledge Component required for success in all of the assessment tasks. Although this debugging skill was too complex to be emphasized directly during instruction, it was determined that students would need ready access to documentation, which was already being targeted with learning objective **D1**: the instinct to readily refer to the documentation. I also managed to include some browser autocomplete examples in my instructional video to best convey the debugging strategies available to students given the constraints on time and scope that I was discovering.

As my participants conducted their think alouds, I came to understand how they structured their knowledge and extrapolated the prerequisite knowledge for each concept I was planning to assess. The following insights were uncovered during expert and novice CTA.

- It is absolutely crucial to go over the DOM during instruction to connect HTML to JavaScript in the minds of learners.
- The fundamentals of HTML attributes should be taught to prime students to think about event listeners, which connect buttons on web pages to actions in JavaScript.
- Relevant links to the documentation should be provided to students throughout the instructional module.
- Experts may have some commands memorized.
- Experts may not have theoretical programming language concepts, such as primitive variable types, memorized. Google Search was a strategy experts used to determine primitive variable types in JavaScript.
- It will be useful to pair examples of variable type conversion with comprehension questions about primitive variable types and terminology about comparison operators (loose/strict equality) in JavaScript.
- Students may fly through the documentation, which contains answers to their programming questions, without truly reading it.
- The lesson must have examples of how to access text within HTML elements.

Expert CTA further reinforced that teaching students the concept of scope in JavaScript was outside the realm of possibility in this instructional unit. After conducting informal interviews and empirical CTA, I updated the goals and assessments for the course. For

the sake of the length of this report, those updates are not recorded but are available upon request.

After refining my goals and objectives, I implemented the instructional modules for this unit in OLI.

Instruction

To match the performance task for PUI LA6, I created a website called Eames Aisle with items, buttons, and a shopping cart.

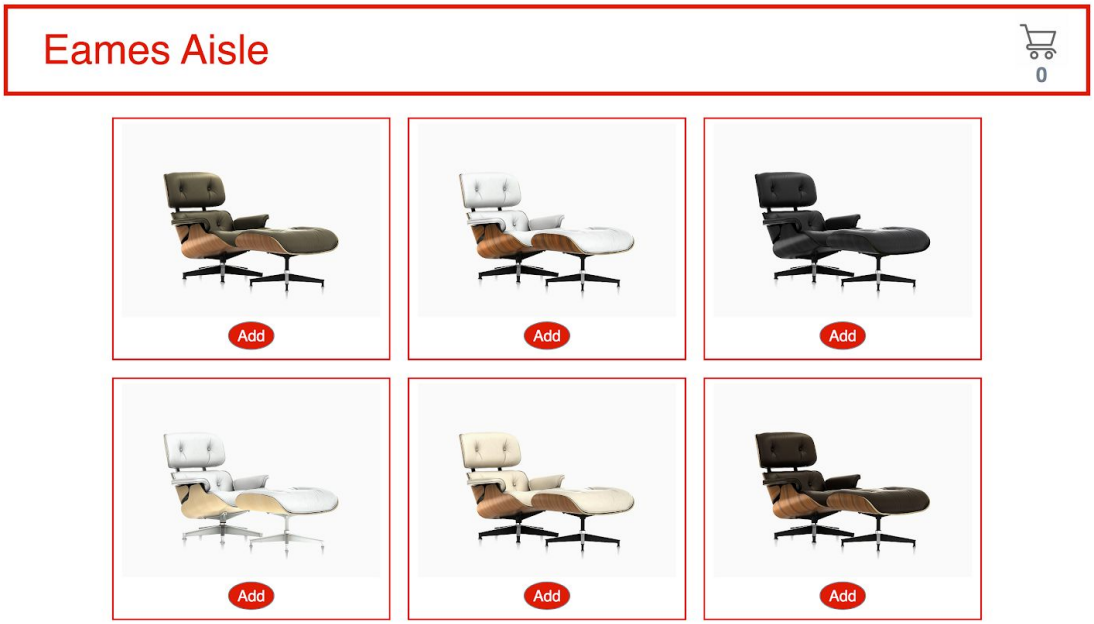
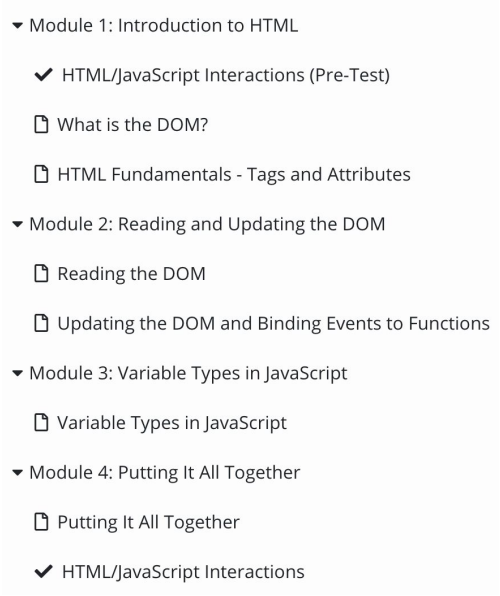


Figure 7. Eames Aisle Website, Used for Instruction

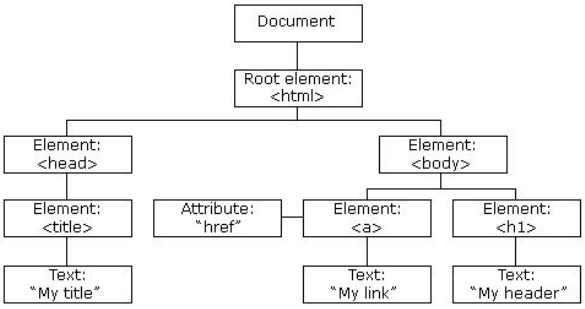
Students were asked to figure out how to update the shopping cart count by clicking on the Add button at the beginning of the lesson and the concepts, procedural and dispositional goals for that overarching task were taught along the way.



Students were given links to the documentation for each module and I modeled performing Google searches, my thought process, and paired video lessons with formative assessments matching many questions on the summative assessment.

The lesson was estimated to take about 40 minutes to complete.

Figure 8. Final Instructional Design



All instruction featured the use of Multimedia Learning Principles, such as the application of the Pre-training Principle in teaching the DOM in the first module.

Figure 9. The Document Object Model

As advised by the PUI TA, the Personalization Principle was applied when I addressed the student throughout the instruction videos and also throughout some of the questions. The Modality Principle was applied when I told students what I was doing during the instructional videos and functioned as a sort of think-aloud task within the instruction. The Segmenting Principle was applied throughout the modules because the instructional videos were split up so that learners could be more self-paced and reduce their cognitive load while internalizing concepts. Lastly, the Coherence Principle was applied because CSS was mentioned, but not elaborated upon throughout the lesson because teaching how HTML and CSS connect to each other was not the intent of this instructional unit.

The innovative principle applied to the experimental B section of this instruction module was Parsons Problems, which are a kind of “jumble” or “refrigerator magnet” problem developed by professors Dale Parsons and Patricia Haden, computer science education

researchers at the School of Information Technology and Electrotechnology at Otago Polytechnic in Dunedin, New Zealand. Barbara Ericson, a computer science educator at the University of Michigan's Information School, expanded upon Parsons and Haden's (2006) work in her 2017 dissertation. In Ericson's words:

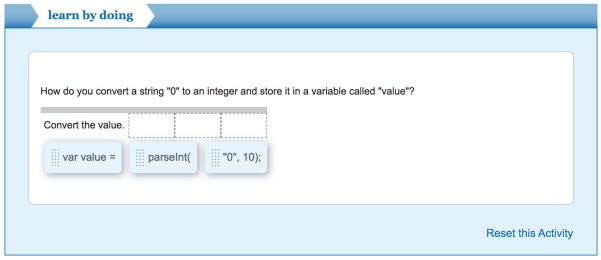


Figure 10. Example Parsons Problem

"Mixed-up code problems (Parsons problems) provide the correct code to solve a problem, but the code is broken into code blocks and mixed up. The learner must drag the blocks into the correct order. You can see an example of Parsons Problems at https://tinyurl.com/ParsonsEx."

For my control group, instead of Parsons Problems, I used fill-in-the-blank input fields, since debugging syntax (such as semicolons, braces, and parentheses) has been noted as a primary difficulty for novice programmers learning JavaScript.

The instructional module took about a day to build, and after its completion, I conducted learner testing and recorded two more think alouds.

Instruction Feedback

After creating this instructional module, I spoke to two students. The first student delivered some feedback that was not necessarily implemented in the final instructional module; however, it may be addressed in the future. This student made the following observations and objections to the instructional module:

- use dropdowns instead of textboxes for formative/summative assessment tasks
- combine videos into a single visual block in OLI for better UI (note: This violates the Segmenting Principle.)
- combine sequentially ordered formative assessments into a single visual block in OLI for better UI
- include more hints in formative assessments
- variable type coercion/conversion should be explained more thoroughly
- summative assessment questions should be static, not random

This student also said that they did not learn that `document` is not the top-level object and that they did not learn how to look up the `document`.

The second student said that they enjoyed the step-by-step scaffolding throughout the lesson, but although `innerText` was explained nicely, they were unable to internalize the concept. This is a key takeaway from this student and also the previous that some of these programming concepts must be reinforced further.

Because these learner tests were conducted so close to the completion due date of this instructional module, none of these changes were made. However, this input is useful to take into account for future directions of the course.

Results and Data Analysis

The A/B test results of this instruction were not very encouraging. There were five (5) students in the control condition and eight (8) students in the experimental condition of this instruction, which featured Parsons Problems. The pretest scores of the version B participants were significantly higher; however the post-test scores were almost exactly the same in terms of mean. There was a very small effect size in favor of the A group (*D = -0.11*). The reason for this small effect size may be because there were not enough performance tasks featured in this instruction. This was a key takeaway for future iterations of this course.

	Version A (N = 5)				**Version B (N = 8)**			
	Pre-Test Score	Time to complete (mins)	Post-Test Score	Time to complete (mins)	Pre-Test Score	Time to complete (mins)	Post-Test Score	Time to complete (mins)
Min	10%	0	60%	4	0%	1	20%	1
Mean	38%	5	80%	6.8	53.75%	3.125	77.5%	4.125
Max	70%	11	100%	17	70%	8	100%	9

Table 4. Results

Discussion and Future Directions

Overall, this course was a viable introduction to CRUD Operations on the DOM Using JavaScript and may be integrated into PUI LA6 lessons in the future. I think future iterations of this course can include an expanded explanation of the DOM, more programming-related performance tasks to better A/B test the utility of Parsons Problems, and some programming basics that aren't necessarily limited to development in JavaScript, such as functions, variable scope and context, and memory management, storage and retrieval.

To improve the method of instruction, I would like to include more feedback throughout each lesson and within formative assessments while ensuring that students learn how to look up the documentation and search online for answers that may not be obvious, since this will result in far transfer to other programming domains outside of HTML/JavaScript development.

Reflection

I learned many lessons during the creation of this instructional module, but I will limit myself to three key learning points. First, I wish I had deployed my instructional module earlier - before I received learner feedback. That would have potentially removed the need for the learner think alouds I conducted and gotten the module in the hands of learners faster. Second, I learned that creating an online learning module requires a **lot** of planning. I had to predict and monitor the output of my code to ensure it met my learning objectives and primed students appropriately for the formative assessments in my module. Third, I learned that just because I know something, it doesn't mean that students will. There were a lot of hidden KCs I discovered as I developed the course, and its scope kept expanding until I realized I couldn't do everything I wanted to do.

I faced challenges in recruiting students, but persistently messaged and approached people, asking them to help me. I also faced challenges in managing the scope of this instructional module, since I wanted to cover the breadth of what is needed for students to succeed in PUI Lab Assignment #6; however, this would have increased the length of time to complete the module to at least an hour.

I would like to present this module to Dr. Jason Hong, the instructor who leads the Programmable User Interfaces course, and ask him if he would like to integrate it into his lab section's support instruction. I do not think students should be required to take this OLI course, but it does teach concepts and skills that are missing from PUI Lab instruction. If he wants to integrate it into his in-person course, I would like to extend the coverage of this course to cover concepts like local/global scope and `localStorage` in JavaScript to achieve full coverage of the concepts necessary for students to succeed in completing Lab Assignment #6.

I was very happy with how I went about this course, especially given the time constraints enforced by the other classes I am taking. The next time I make an online course, I will deploy and get feedback on my instruction faster. Receiving feedback from assessment takers and my initial instructional testers was the best thing which served this module and its intended students.

Rating and Comment of Each Group Member's Participation

Name: Neil Thawani (Contribution: 100%)
Role and participation level: I was really happy with how I managed to keep up with most of the milestones during this project, although I only managed to deploy my course just before Thanksgiving Break. I enjoyed iterating goals and assessments until I was able to conform my instruction to what the objectives of the course were, and felt very little pressure leading up to the final presentation date as a consequence of being comfortable with the quality of my deliverable submissions and the effectiveness of my course.