Project 5: Mascots Game, redux

Using Objects to generate results on the fly

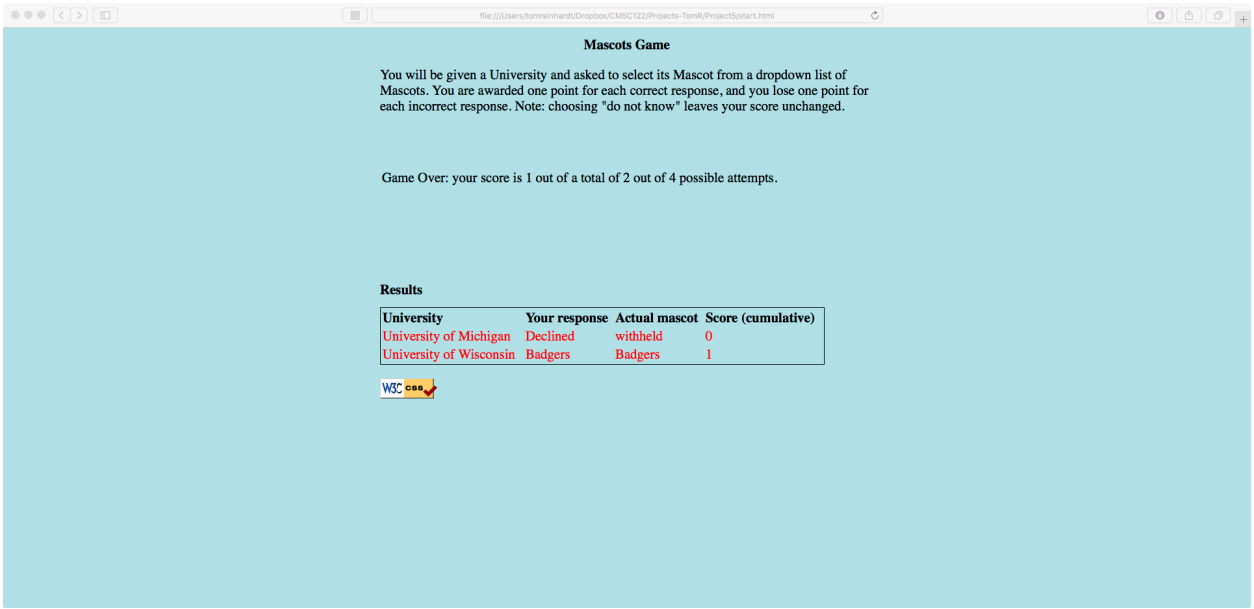
Overview

As promised, you will revise your Mascots Game (from Project 3) by augmenting your representation scheme to use Objects to encode essential data/relationships:

- You will represent the “association” between each University and its Mascot as an Object.
- Your will represent each “round of play,” as an Object that records the user’s response, the correct (expected) response, and the running score.

These are two distinct Object types, and both will reside in separate arrays of objects.

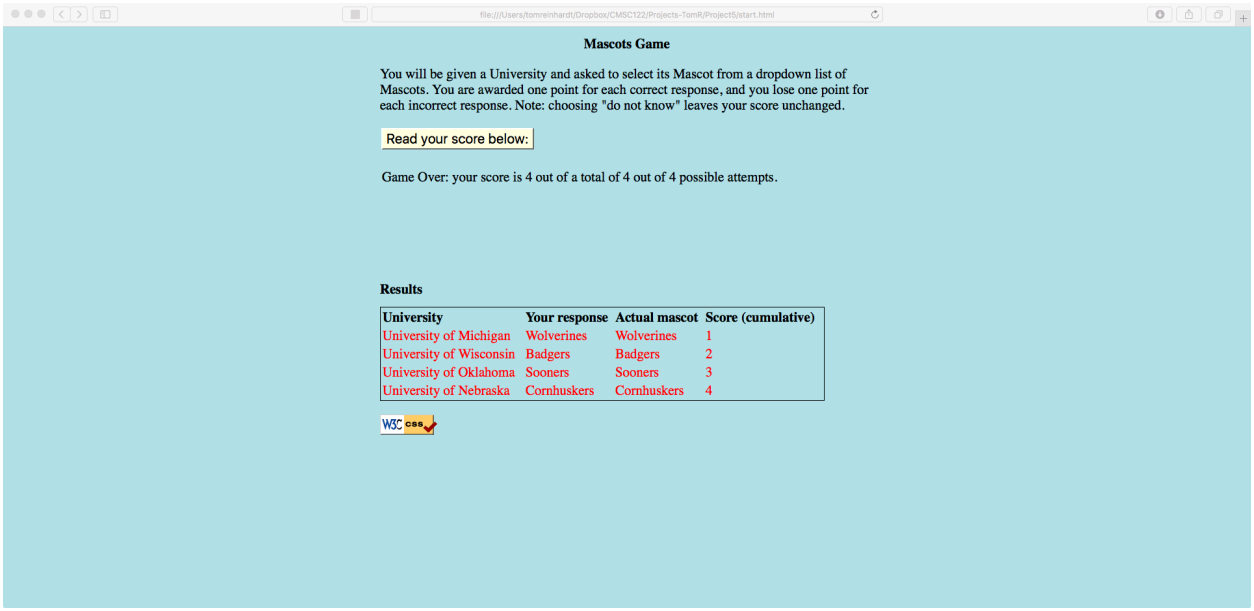
By doing this, it will be easier for you to modify your original HTML and CSS files to provide for an area at the bottom of the page reserved for the generation of a table containing results of each user interaction from the beginning of the current game until either the user exits by choosing the “Exit” button or the user completes all of the questions. Here are some screenshots showing that new area and how it may appear:



In this display, the user exited after the second question, choosing “Don’t Know” for the first question and getting the correct response for the second question.

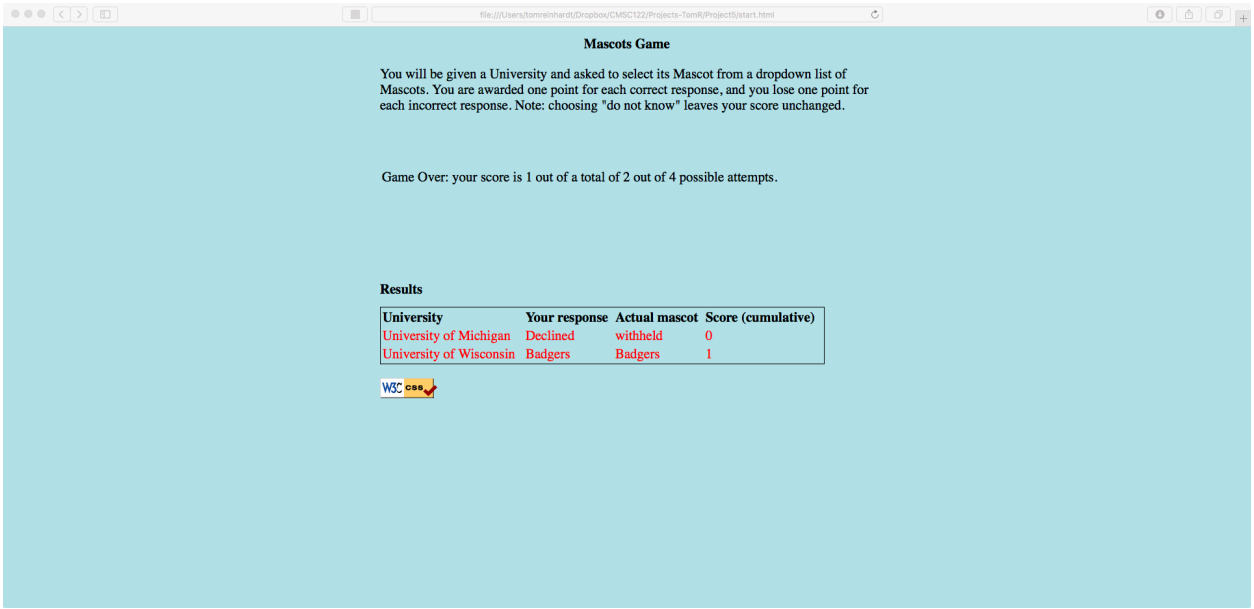
One way to think of the design strategy is to consider each table entry as a textual visualization of the data contained by each “Object” that your program creates in response to a “round” of play. The “array” of response objects, therefore, contains a “history” of questions, responses, and cumulative scores.

Continuing in that vein, suppose that the user completed the game with a perfect score; we should expect something that looks a lot like:



(By the way, the “Read you score below:” text that appears in the Prompt Button is purely optional … your implementation may choose to do something different, as long as it’s correct.)

And, of course, our logic needs to account for intermediate results. For example: suppose that the player does not know the first question, answers the second question correctly, and then exits the game. We should then show:



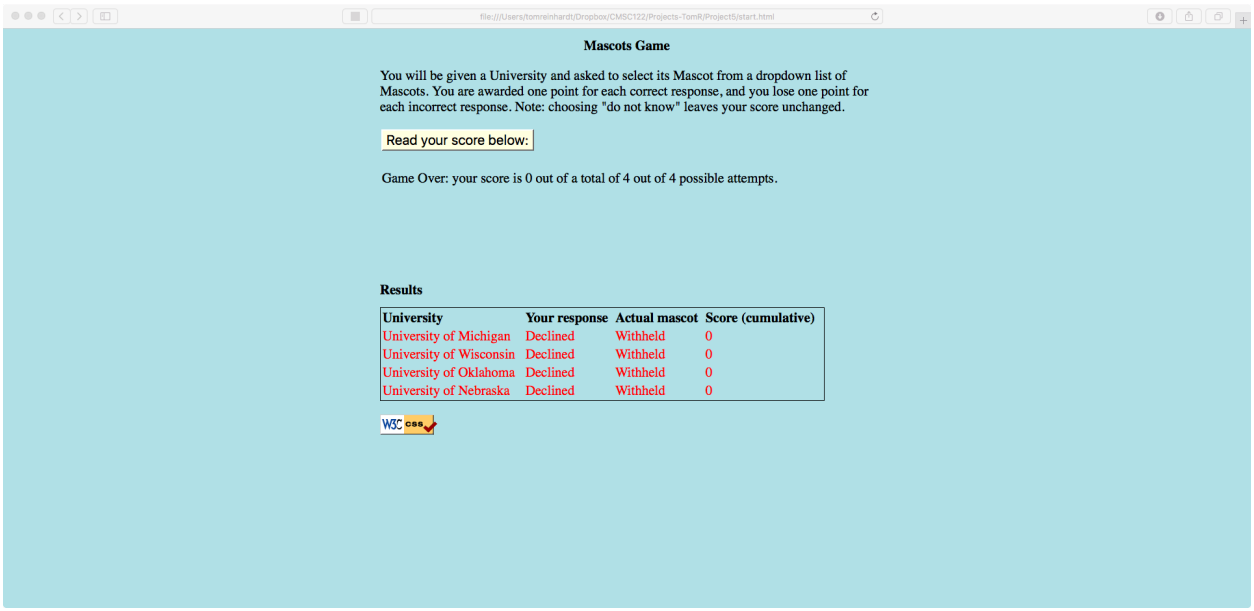
Note that the table is “dynamically generated.”

The Details …

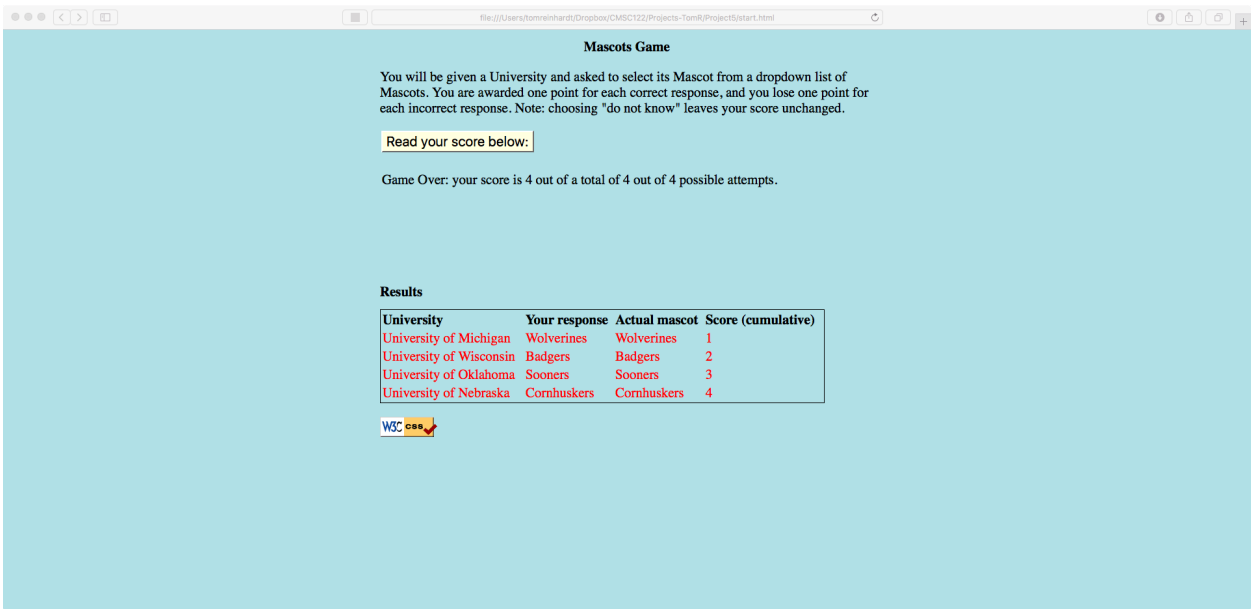
I suggest that you define two arrays of different object types.

The first array holds objects that embody a “pair,” which is common is Computer Science; you’ve seen such “pairings” all semester. Your pairing might be `[ University, Mascot ]`, where `University` and `Mascot` are `Strings`. Use this array to “generate” the next question in response to the user clicking on the prompt button. When no more pairs are available, the game is over.

The second array will contain objects that embody the information that appears in the results table that you will generate upon the user’s exiting the game. Be careful not to give away the correct answers too easily. Suppose the user “doesn’t know anything” then we should expect something that appears as:



And, of course, if the user knows everything, then they have



Naturally, your logic should be able to generate tables that provide for all of the possible response combinations (given 4 University/Mascot pairs and 5 possible responses, how many possible tables could we generate?).

Submission Guidelines

How you will be graded.

- Naturally, we will be looking for your definition and use of JavaScript Objects where your earlier implementation may have relied upon nested if-statements, or some other ad hoc algorithm.
- We will also be looking at how your JavaScript interacts with your CSS tags to show and hide HTML elements based upon game flow/interactions.