Assignment #3

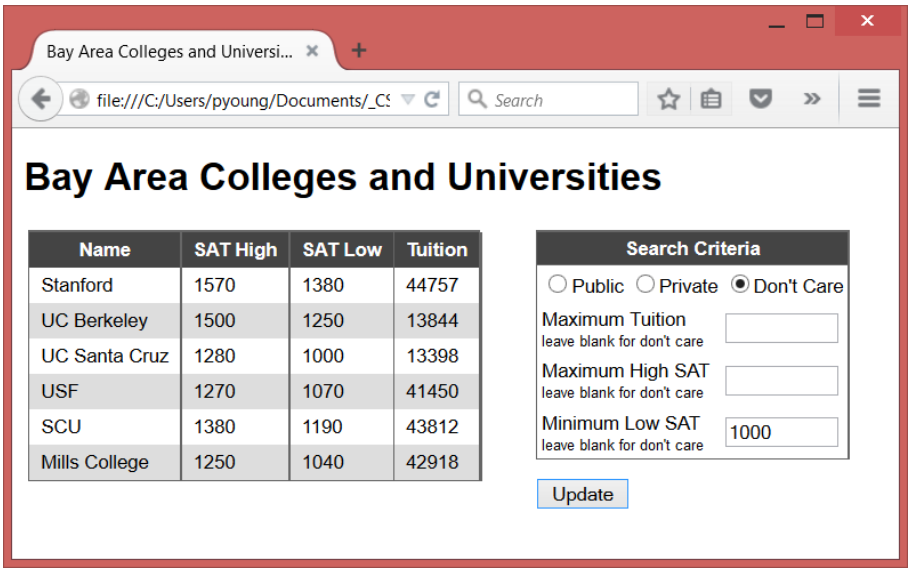
JavaScript

CS193C Summer 2022, Young

We now get a chance to explore some of JavaScript's more advanced features. All problems should work in Firefox and Chrome. This assignment is due Thursday July 21st at 1:30pm.

College Information

Our first task is to create an online college database. Here's a screenshot of this webpage:



As you can see the webpage features a table displaying information on Bay Area Colleges and Universities. At the right is a set of controls to allow searching. The Search Criteria allows the user to limit the display based on the Maximum Tuition, Maximum High (75 percentile) SAT score or the Minimum Low (25 percentile) SAT score. When the "Update" button is clicked the table is regenerated.

Note that the search criteria is combined as an "AND", so universities listed must match all criteria.

Implementation Notes

Warning: Don't forget that the web browsers will create a <tbody> element inside a table, even if one is not explicitly declared. Depending on how you decide to access or generate the table, this may make a difference.

If you're looking for some help implementing, you might want to try implementing in the following order:

- Design the output table using static data, so you know what you want it to look like. In other words, write an HTML file without any JavaScript that displays data in the table format you want.

- Now try generating some sample data in the format of the output table, ignoring the search criteria. In other words, try generating the output table from the previous step by using JavaScript. Ignore the search criteria.

- At this point, you should have convinced yourself that you can dynamically create a table from JavaScript. Now set that work aside and work on making sure you can meet the search criteria. Develop a separate webpage that lists universities which match the search criteria in an "alert" dialog box.

- Now, put the two halves together by generating the output table based on the search criteria.

If you're interested in getting more practice, consider make each of the headers clickable and use them to sort the colleges listed. For example, if the user clicks on the "name" header, list all colleges in alphabetical order. If they click on the "tuition" header order the colleges in order of their tuition. You may also want to display the in a more user-friendly format (e.g., $27,204 instead of 27204).

The data is from US News & World Report's Best Colleges 2015 version.

Check the assignment downloads for a javascript file with the Bay Area university and college data.

Matching

For our last question we will develop a simple matching game with cards. This was originally a midterm question, so if you want to get a sense of how difficult midterm questions will be, here's a good example.

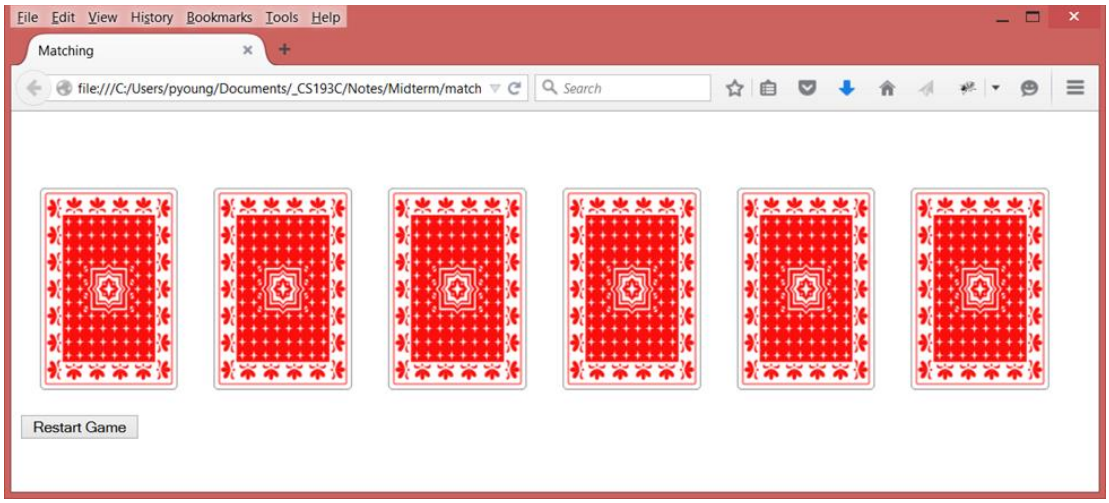
In order to do this problem, you'll need to get a set of image files from the assign3.zip file.

Here's a description from the user's perspective. We have a deck of six cards. The cards are from the hearts and the clubs suites. They are numbered 1, 2, and 3.[1] The deck will be shuffled and laid face down on the webpage. The user will click on an initial card which will stay face up, the user will then click on a second card. Both cards will stay face up until 1.5 seconds after the second card is flipped. Then if both cards have the same value – say for example one is the 2 of hearts and one is the 2 of clubs, they will both disappear. If instead the cards have different values, they will both flip to face down. The user can then click another card and the process repeats itself until all matches have been found. *To keep things simple, we don't do anything special after all the matches have been discovered.* Just leave the page with all the cards gone until the user hits the "Restart" button.

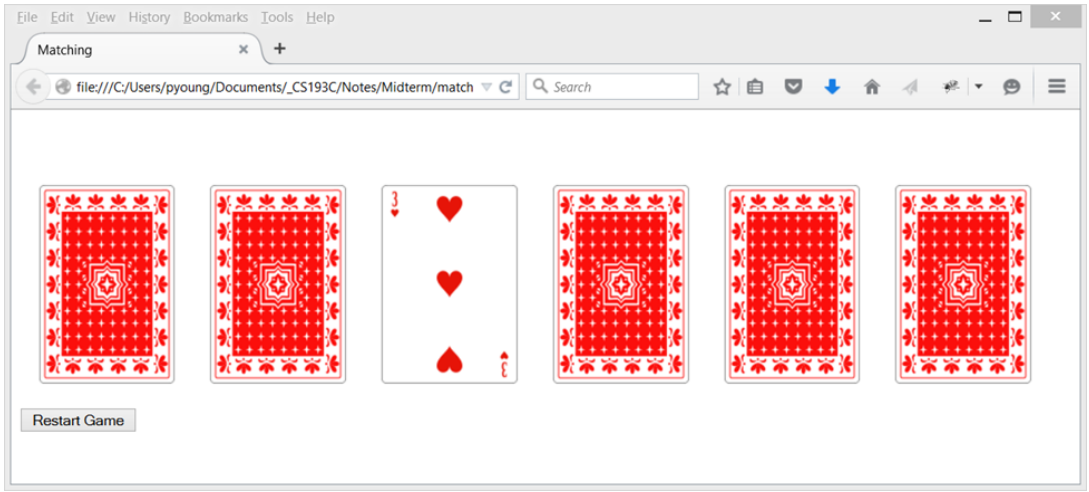
At the bottom of the page is a "Restart Game" button that can be used to reset the page – reshuffling the deck and laying out six cards face down again. The user can click this at any point in time and it should respond correctly.

[1] Note for actual card players: The deck I found online had both Ace cards and 1 cards. I thought it would be simpler for those not familiar with a deck to use a "1" card rather than an "A" card. I'm not altogether sure what the "1" cards were for (I've never seen a real deck with them), but they seemed the simplest choice for this midterm problem.

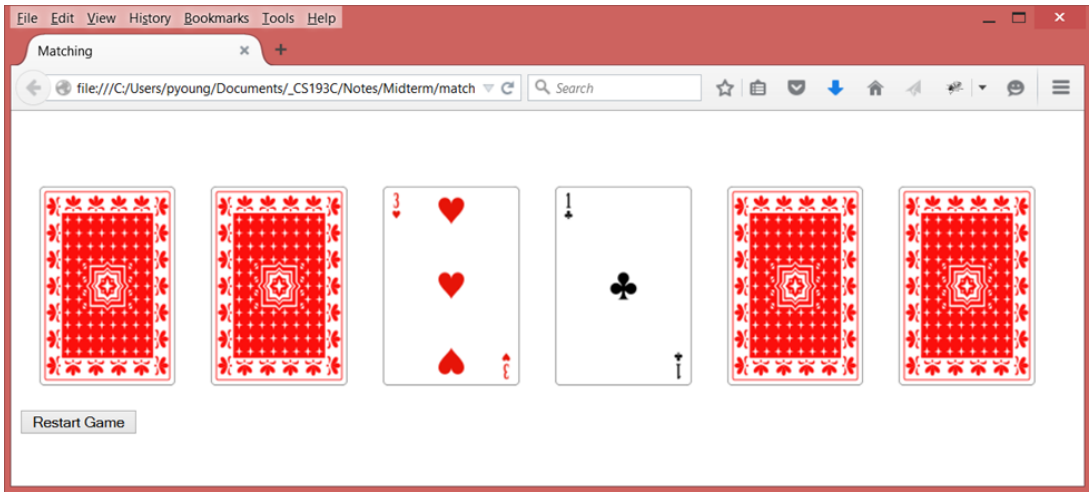
Here are some screenshots of the website in action. Initial placement:



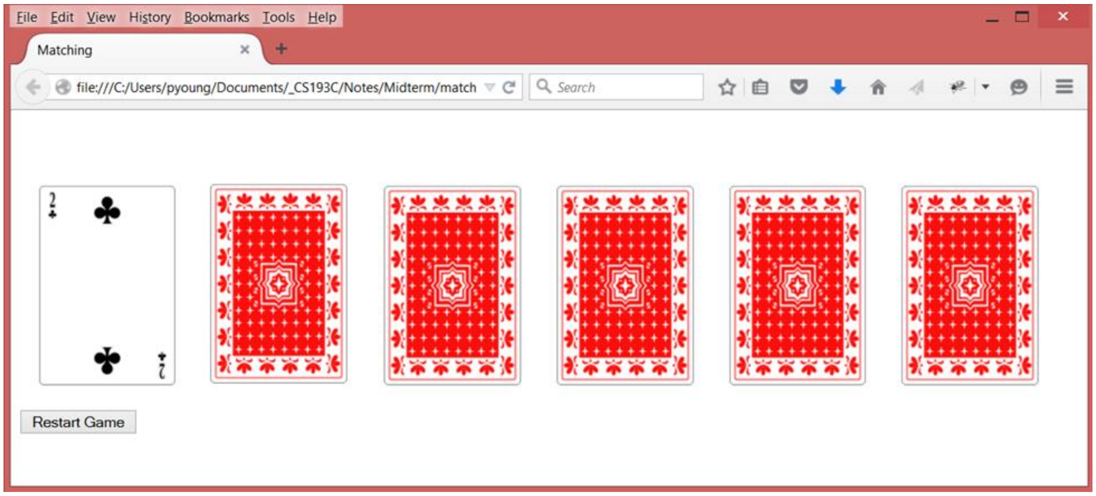
Flipping one card:

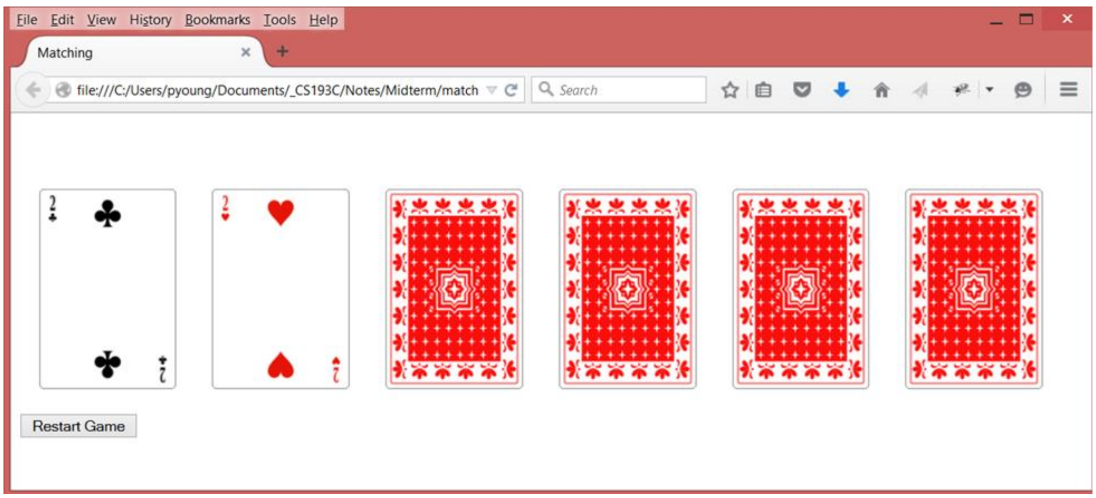


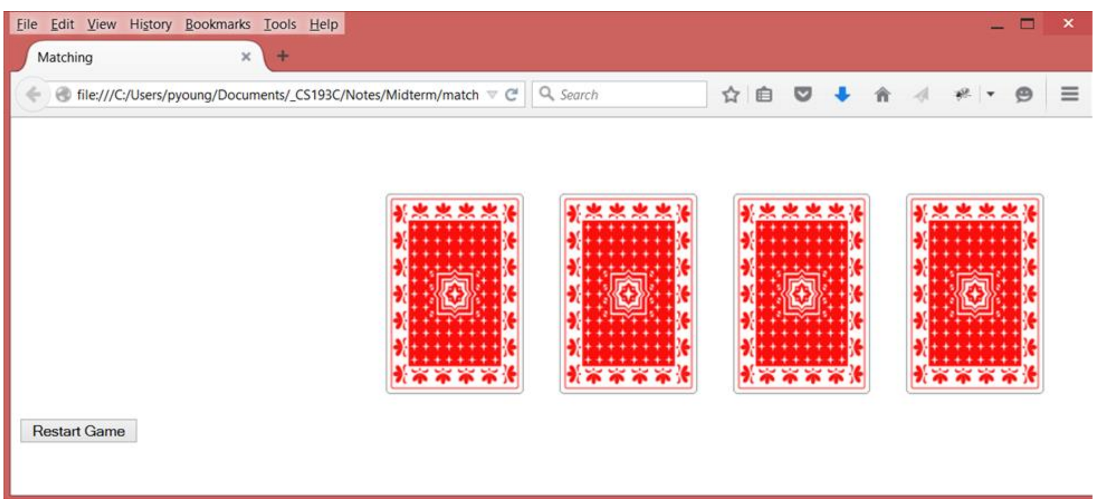
Flipping a second card. Both cards stay visible for 1.5 seconds after the second card is flipped before they both flip back over:



Here I've found a match. Both cards will be visible for 1.5 seconds after the second card is flipped, then instead of flipping back over they will both disappear. Here's the full sequence:







Here are some additional notes for your implementation:

- We aren't particularly concerned with exactly how you've laid out the webpage. The six cards should be placed neatly in a single line, leave a reasonable amount of space between them so they aren't too crowded together.

- As previously noted, when two cards have been clicked on, they will both stay face up for 1.5 seconds after the second card has flipped. For full credit, you need to ignore the user clicking on other cards, while the initial pair is still faceup—you do not want to have a situation where the user is somehow able to have three cards flipped up at once. However, the user should still be able to click on the "Restart Game" button at any time.

- The card backs are shown by using the "back.png" file. I didn't actually eliminate the cards when a match occurred, instead I replaced the card with the "clear.png" file which is an empty white square the same size as the card images. However, you're welcome to solve this any way you like. Both the "back.png" and "clear.png" file are included with the assignment downloads. Do make sure clicking on the "clear.png" image doesn't flip a card back faceup.

Implementation Suggestions

This was originally a midterm problem. I was concerned that the students might have trouble finishing it (as it turned out they actually did better than I expected), and provided this section to give students an idea of different parts of the problem that they could break off and implement. The idea was that we wanted students to be able to turn in a solution with some working components if they weren't able to get the entire problem working. Feel free to ignore this, but I do think it's going to be useful to be thinking about how you can break the midterm problems into smaller size chunks that you can do one at a time, building up to a larger more complete solution – we generally do ***not*** *provide this guidance on the exam, this was only because I was worried about the complexity of this specific problem.*

Here are some suggestions for how to go about implementing this. They are in order from what I view as the easiest to hardest, but you are welcome to go out of order if you want.

1. The first and simplest step is to just layout the webpage showing card backs and the reset button.

2. Next focus on getting some of the simpler game mechanics done *before* dealing with randomization and matching. Once you've got the webpage layout working, assign fixed cards to each of the card images. For example, assume that the left-most card is always the 1 of hearts, the second card is always the 2 of hearts and so on. When the user to click on any card flip it over by swapping in the appropriate card image. Leave that card face up.

3. Add the ability for the user to reset the cards to back side up using the "Reset Game" button.

4. If you've got steps 1-3 done, the next step is to get cards to flip back over to show their backs automatically. When a single card is flipped over, leave it face up for 1.5 seconds and then flip it back over.

5. Add blocking, so that while a card is temporarily flipped over, clicking on other cards will be ignored until the first card goes face down again.

6. Add randomization to the order of the cards, so that the user will get a different ordering when they load the webpage and each time they hit the "Reset Game" button. You can use the Math.Random method described here:

   https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

   You may copy the getRandomInt function directly of the Mozilla webpage. [This is the one and only place that you should be directly copying code of the Internet for this assignment.]

7. You're almost done, now you just need to add matching.

Destructuring

For our last part of the assignment, we'll get a little bit of practice using destructuring. This JavaScript language feature provides a compact method to get data out of an object or array and store it in individual variables. It has a number of common uses including getting values and functions out of an existing library in Node.js or modules based on the CommonJS system and allowing us to write functions which return multiple values.

I provide you with three files related to destructuring:

destructuring-techniques.js – This file focuses on the mechanics of using destructuring. It is provided for your information and does not need to be modified or submitted.

destructuring-use-examples.js – This file provides some simplified use cases for destructuring. It is provided for your information and does not need to be modified or submitted.

destructuring-work.html – Modify this file by writing two separate lines of code. One line of code should call the function getData and use destructuring to assign values to the name and year variables. The other line of code should call the function getDorms and use destructuring to assign values to the first, second, and fifth variables. For more details read the file.

Credits

The cards are by David Bellot and were found on the Wikimedia website. They are distributed under GNU Lesser General Public License.

As previously mentioned, the college data is from US News & World Report's Best Colleges 2015 version.