For the final project, I decided to design a simple single-player matching card game.

The layout of the html page is rather straightforward as I styled the “layout” page we used in class with a simple game stats section which displays the necessary information about the game; namely a timer counter and a reset button at the header section. I also learned how to style backgrounds using CSS, as practised in the tutorials on http://www.w3schools.com/ and applied it to the background of my game. The body of the html page then holds the game section where the html elements in Javascript are placed. For instance, the gameIntro screen and gameComplete screen, as well as a div called “cards” which holds the deck.

Similar to what we were taught in class on developing a simple interactive game using Raphael for graphical interaction, a “selectCard” function is created. Upon clicking the card div, the function is called. Instead of creating a moveSquare function that counts clicks like we did in class, the class “.card-flipped” is added to the element that has been clicked. I also learned from the class on ‘coding a game’ that we could add an event listener to the object that calls the function. As such, in the context of my game, after we check for matching cards with the isMatchedpattern function, the class “.card-flipped” is removed, and the class “.card-removed” is added. This reduces the opacity of the div to zero such that the card can no longer be seen. After which, I bind the eventlistener webkitTransitionEnd so that when the div has zero opacity, the “removeTookCards” function is run and the div is deleted.

In addition, I learned the use of conditional statements whereby “if” entails the execution of code if a specified condition is true and “else” executes the code if the same condition is false. In my game, if the 2 cards have the same picture *i.e., if (isMatchPattern()) { ,* cards are removed. Otherwise, they are flipped back to the original side, specified with the classes “.card-flipped” and “.card-removed”.

I also learned to include console.log messages at intervals throughout the game so that I can simply check the console window to make sure the game is operating properly.

Last but not least, I sonified my game with two different sounds from http://freesound.org. I learned that these audio elements have a .play() and a .pause() method, and a .loop property that takes a true/false value. Using a techno beat as my background sound, I wanted to created a mood that suit the whole “band” theme; for that uplifted spirit. Also, for every correct matching card flipped, a “yay!” sound is produced. For that, I stored the sound as a property of the cardsmatched object. However, I’m thinking now that the game could be sonified better if I included a .currentTime property that if set to 0 after pausing would start the techno beat from the beginning again.