Web Client-Side Programming – Homework 8

Pulling it all together

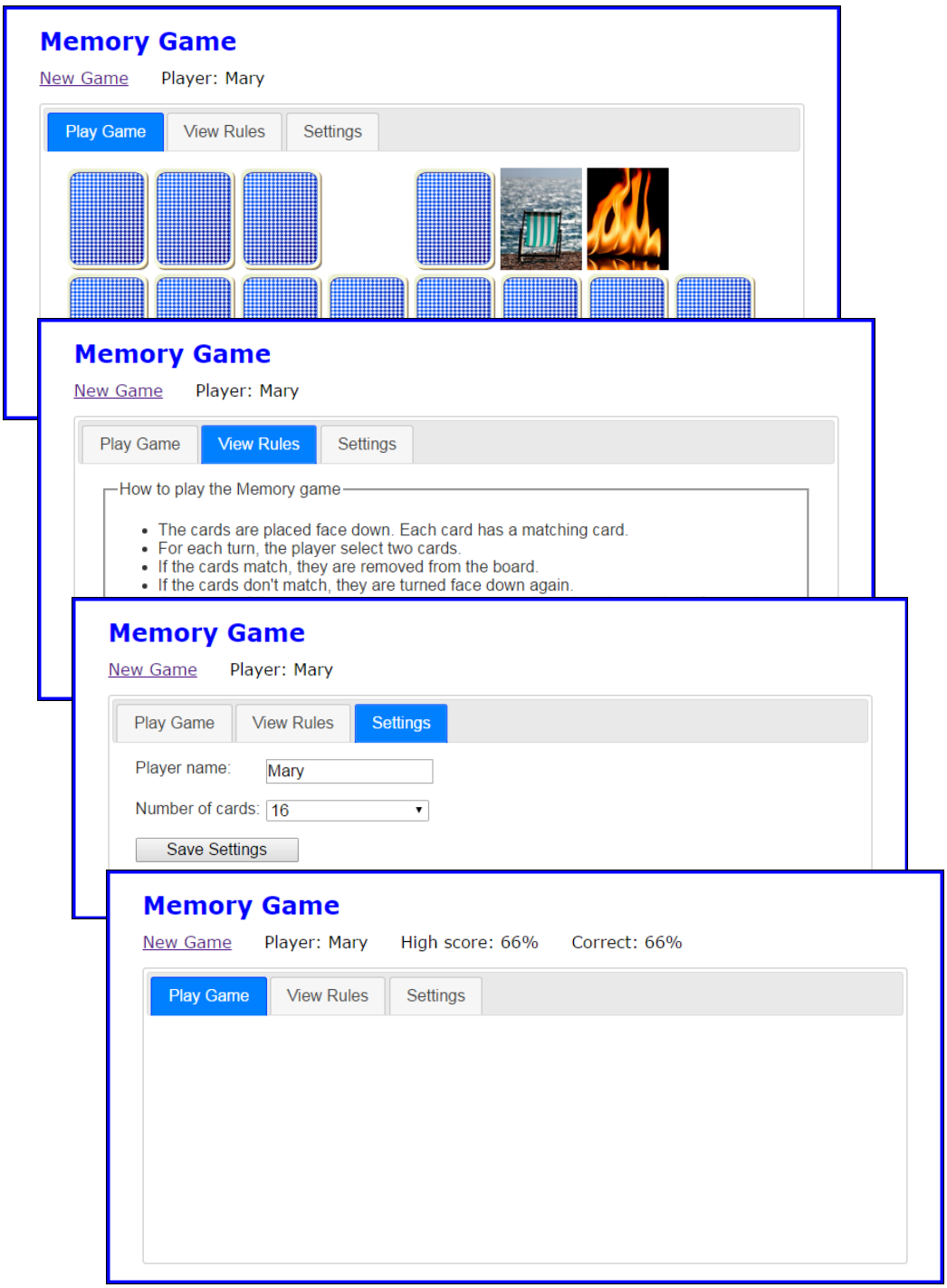
Description

For this assignment we will create a memory game.

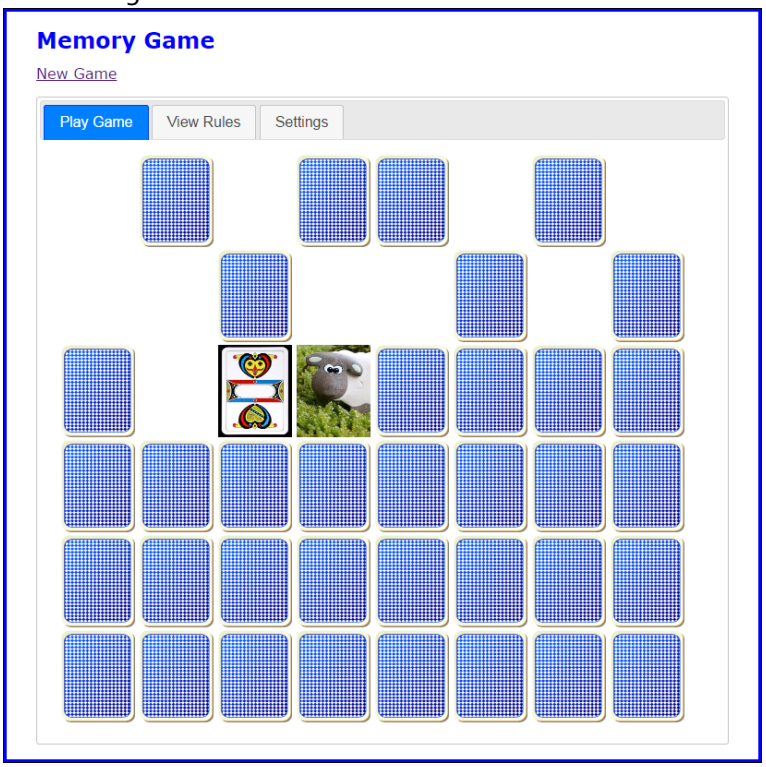
Specifications

For this assignment, you'll create an application that allows the user to play the Memory game. The objective of this game is to turn over two cards at a time, trying to find cards that match until all cards have been matched.
Prerequisites:

- **User interface**



- The main portion of the user interface should be displayed in a Tabs widget with the three tabs shown above.
- Images are provided for 24 cards, the back of the cards, and a blank card. The images are given to you in the images.zip file.
- By default, the game should display six rows of cards with eight cards in each row, for a total of 48 cards (two of each card). The user can change the number of cards used as well as the player name from the Settings tab.
- When the user clicks the Save Settings button, the player name and number of cards should be saved in a cookie. In addition, the page should be reloaded so the player's name, the player's high score if previous games have been played, and the correct number of cards are displayed.
- Use an images array to store the images for the cards used by a game. Use a cards array to store the src attributes of the images for the cards that will be displayed on the board (two for each image).
- Create the HTML for the displayed cards by randomly selecting elements from the cards array. The cards for each row should be stored in a div element, and the div elements for all the rows should be stored in the div element with an id of "cards".
- When the user clicks on a card whose back is displayed, the back of the card should be faded out over half a second and the front of the card should be faded in over half a second.
- When the user finds two matching cards, the cards should be hidden after one second using a sliding motion over half a second. If the cards don't match, they should be faded out over half a second after two seconds and the back of the cards should be faded in over half a second.
- Each time the user completes a game, the user's high score should be updated and displayed and the percentage of correct selections for the game that was just completed should be calculated and displayed. The high score should also be stored in a cookie, so it can be compared against the score for the user's next game.
- When the game ends, the user can click the New Game link to start another game.
- The HTML for each card should consist of an <a> element with its id set to the src attribute for the card and its href attribute set to "#". The <a> element should contain an img element with its src attribute set to the source of the image for the back of the card and its alt attribute set to "".



- Use objects to modularize the code.
- The objects should be coded in these four JavaScript library files: library_settings.js, library_scores.js, library_cards.js, and library_card.js. These files are in addition to the main.js file.

- The library_settings.js file should create a settings object that keeps track of the player name and the number of images used for a game. This object should contain the functions that save this data to and retrieve it from cookie storage. Because you need just a single instance of the settings object and you don't need to protect any private state, you can create the settings object with an object literal.
- The library_scores.js file should create a scores object that keeps track of the score for each game as well as each player's high score. This object should contain the two integer variables that keep track of the number of turns and the number of turns that result in matches. It should also contain the functions that save the scores to and retrieve the scores from local storage, compare scores, and display a player's high score.
- You'll also need to add public methods that increment the integer variables, check if all the cards have been removed from the board, and calculate the percentage of correct selections. These new methods should replace code that previously worked directly with the integer variables.
- The library_cards.js file should create a cards object that works with the cards. This object should contain the functions that preload and store images, store the src attributes for the cards, create the HTML for the cards, flip a card using a fade effect, and flip a card using a slide effect.
- In addition, the application should use strings throughout the application to specify the src attributes for the card back and blank card, these attributes are stored in variables in this library.
- The library_card.js file should create a Card object for working with a single card when it's clicked. Since a new Card object will need to be created each time a card is clicked, a constructor function should be used. The constructor should accept a parameter that represents the <a> tag that was clicked, and it should use the jQuery selector to get the img tag that's a child of the <a> tag. The Card object type should have two properties that store this img object and the value in the id attribute of the <a> tag. The Card object type should also include two methods. The first one should check whether the user has clicked on a card that is blank or has already been revealed. The second one should accept a parameter that represents the first card that was clicked, and it should check if the id attribute of the <a> tag for this card is equal to the id attribute of the <a> tag for the current card.
- The library files should be included in the index.html file in the proper order, so each library is included after any libraries it depends on.

Documentation

You will need to create a document (.docx, .rtf, .pdf) which contains the following:

- A screenshot of your page running on your local webserver. Your screenshot needs to include the address bar with the localhost url and show all the validation messages.
- DO NOT forget the JSDoc
- You will need to validate the index.html (link on Canvas) and include a screenshot of the successful validation (Green Bar).

What to Submit

You need to submit your index.html, CSS file, and your external JS file along with your Document. Make sure your document is in the correct format and includes your name.