Game Behavior

CRITERIA	MEETS SPECIFICATIONS
Memory Game Logic	The game randomly shuffles the cards. A user wins once all cards have successfully been matched.
Congratulations Popup	When a user wins the game, a modal appears to congratulate the player and ask if they want to play again. It should also tell the user how much time it took to win the game, and what the star rating was.
Restart Button	A restart button allows the player to reset the game board, the timer, and the star rating.
Star Rating	The game displays a star rating (from 1-3) that reflects the player's performance. At the beginning of a game, it should display 3 stars. After some number of moves, it should change to a 2 star rating. After a few more moves, it should change to a 1 star rating. The number of moves needed to change the rating is up to you, but it should happen at some point.
Timer	When the player starts a game, a displayed timer should also start. Once the player wins the game, the timer stops.
Move Counter	Game displays the current number of moves a user has made.

Interface Design

CRITERIA	MEETS SPECIFICATIONS
Styling	Application uses CSS to style components for the game.
Usability	All application components are usable across modern desktop, tablet, and phone browsers.

Documentation

CRITERIA	MEETS SPECIFICATIONS
README	A README file is included detailing the game and all dependencies.
Comments	Comments are present and effectively explain longer code procedure when necessary.
Code Quality	Code is formatted with consistent, logical, and easy-to-read formatting as described in the Udacity JavaScript Style Guide.

Suggestions to Make Your Project Stand Out!

- Add CSS animations when cards are clicked, unsuccessfully matched, and successfully matched.
- Add unique functionality beyond the minimum requirements (Implement a leaderboard, store game state using local storage, etc.)
- Implement additional optimizations that improve the performance and user experience of the game (keyboard shortcuts for gameplay, etc).

Development Strategy

It's very important that you plan your project before you start writing any code. Break your project down into small pieces of work and plan out your approach to each one.
It's much easier to debug and fix an issue if you've only made a small change. It becomes much harder if you wait longer to test your code. You don't build a house all at once, but brick by brick.

Start by building a very simple grid of cards.
- Don't worry about styling, just get something clickable on the page.
- Figure out the HTML needed to represent a card. Remember, you have to represent two sides of the card. Are you going to have two separate elements stacked on top of each other?

Add the functionality to handle clicks.
- This should reveal the hidden side of each card.

Work on the matching logic. How does your game "know" if a player guesses correctly or incorrectly?
Work on the winning condition. How does your game “know” if a player has won?
We recommend saving styling until the very end. Allow your game logic and functionality to dictate the styling.

TODO list:

	Does it work?		Does this action needs CSS styling right away?

1 Create your grid + cards and the start modal **in JavaScript**															
	check	1 Create an array that holds all cards as objects													
	check	2 Duplicate that array to get 8 pair													
	check	3 Randomize the cardsArray													
	check	4 Create a grid 8x8 by giving it a classname and css property's (in the *.css ofcourse)													
	check	5 Loop through the cards:													
	check	-Create the cards (create a div and add a classname: "card")	yes!												
	check	-Create the frontside	yes!												
	check	-Create the backside	yes!												
	check	-Append the cards to the grid													
	check	6 Add a classname to style selected cards when 2 cards match '.selected'													
	check	7 start modal? create a div, giv it a classname start-modal													
2 Create an onclick listener to flip the cards and count the moves															
	check	1 Add eventlistener to grid to flip the cards when a card is clicked													
	check	-Make the event target the clicked item													
	check	-Do not allow the grid section to get selected, only the divs inside the grid													
Guess counter	**check**	-Create a function for matching cards: if count < 2 count ++													
	check	if count === 1 assign first guess													
	check	else assign secondguess													
	check	if firstguess && secondGuess then:													
	check	if firstguess === secondguuess then run match with a little delay for the animation to finish													
	check	if matchCount = 16 then pauze the timer and give the win modal a classname													
	check	reset guesses													
	check	2 Create a start/reset function (addEventlistener) that resets firstGuess, secondGuess and the guessCounter and removes the classname '.selected'													
	check	3 Flip with some delay for the animation?													
3 Create -matching logic **-move counter** **-guess counter**															
	check	matching logic: 1 create an if statement If firstguess === secondguess then change css so then won't flip													
		move counter: 2 Create a move counter (onclick.addEventlistener) and show it above the game													
	check	createa variable called moves, and count +1 to it so: if firstGuess && secondGuess {moves++}													
	check	create a div and add classname .moves and add it to the game and modal screen													
	check	guesscounter created in step 2.1													
4 Create star rating															
	check	1. create an if statement: If moves >= 3 then change 1 star classname, If moves >= 6 then change another star													
5 Create a timer															
	check	1 Import the chronometer from easyTimer.js in the html file													
	check	2 Show the timer above the game and on the winning modal screen													
	check	3 Timer should pause when all 16 guesses are right													
6 Create a winning logic															
	check	1 In the code for the matching logic, when firstGuess and secondGuess are equal count 1+ to a variable like winCount and then execute the code If winCount = 16 to add a classname to the winning modal screen													
	check	2 if all 16 guesses are right then stop the timer, show modal screen that shows: time, stars, moves, restart button													
7 Create modal screens															
	check	1 Create 1 modal, add when win some strings and add them to the modal													
	check	2 The restart button sets timer, star rating, moves, guesses to 0, also the game deck has to flip all cards and randomize them. And modal should close (remove winmodalclassname) maybe only one START button instead of START and RESTART													
DEBUG	**check**	cards stay open after restart													
Style															
	check	style the game + cards + modal screens													
Documentation / README.md															
	check	Be sure to check your HTML CSS and JS on the comments, trailing white spaces, etc. check out the http://udacity.github.io/frontend-nanodegree-styleguide/index.html and let it check by a validator like https://validator.w3.org													

extra if there is some time left														
	check	- Add CSS animations when cards are clicked, unsuccessfully matched, and successfully matched.												
		- Add unique functionality beyond the minimum requirements (Implement a leaderboard, store game state using local storage, etc.)												
		- Implement additional optimizations that improve the performance and user experience of the game (keyboard shortcuts for gameplay, etc).												
git / gitHUB														
		- After something is ready to test online or I need some help from others I make a commit (I'm using VScode)												
		A commit has to follow the Udacity's Git Commit Message Style Guide	found here											
		- Second I push the commit to origin https://github.com/ThijsWaalders/Memory-Game.git (that is my gitHUB's repo for the memory game):												
		- And as third I push the branch that I am working on to the master branch so that it is viewable as a page on gitHUB. with this command: git push origin changing-grid:master (where changing-grid is the branch I'm working on and master is the master branch												
		I'm not 100% sure if I'm doing it right, and I have to learn some more about the branches. For the needs of this project I think it's enough.												
Sources:														
		My gitHUB repo for this project												
		My codepen for this project. My gitHUB is more up to date												
		Super Mario Memory Match Game Tutorial to understand how the game works												
		The project in the Udacity classroom	You can also find the styleguides for HTML/CSS/JavaScript/git here											