My solution is 100% dynamic and it is fully object oriented driven.

In the html file I only declared the general structure of the game and initialized only static elements.

In the js file, I implemented a game class which represents the whole game life cycle. I called it MemoryGame class . it’s constructor receives the coloumn and the rows number (the game board dimensions) . and set them to their values (this.values) . and then it initialize all the other variables and functions and event listeners needed to implement the rules of the game.

It initialize the cards number (row\*coloumn) and the pictures number ((row\*coloumn)/2). Moves to be initialized as 0 . matches to be 0 . etc..

And it set the new Game button to operate the reset() function when clicked. And the cards to operate the cardsFlipping(e) function when clicked.

Then the constructor calls the initpallet()function which initialize the game board.

The initPallet() function runs two nested for loops , the first one runs for each row and the second one(which is inside the first runs for each coloumn) . inside the loops , the function creates for each row a “row” div and give the appropriate class name and id on the fly so it can have already it’s style and logical functioning ,and the append to the table (declared static in html). And for each coloumn it declares a “card” div and then append it to the “row” div.

At this point we have the board initialize but without any images in it. So, I ran a loop on the images and for each image I ran another loop of two iterations (because each image is going to appear twice. And for each image I generated a random number between the values of( 0 –> the cards number -1) twice.

And then I placed 1 at the same index of the two random numbers that I have in the occurrences array.

And the next image checks whether it’s own random numbers has 0 or 1 in the occurrences array . if 0 it’ good to be placed in the same index of the array. If it’s 1 it regenerate a new random number till it has a good one.

The reset() function clears the board using jQuery remove method and creates a new board of 5x6.

And Now to the real game algorithm which is implemented in the cardsFlipping(e) function: My algorithm is not really flipping the cards but it just changes the css visibility property of the clicked image to visible (by default its hidden) which means that the images are there all the time but you only see it when clicking it.

My algorithm is pretty simple , first of all , I check if the image is shown or not and I decrease the moves and the totalnum variable accordingly. (the max value of the totalnum is 2 because each image can be seen for two times at the maximum) and the I move the reference of the current image (s[0] to the firstOcc var) so I can catch the next image in the s[0] obj . and now I have two opened images in my two vars , so now I can compare between their sources, and act accordingly. And by going again and again on this algorithm , finally I check if the matches==picsNum then it means I won !