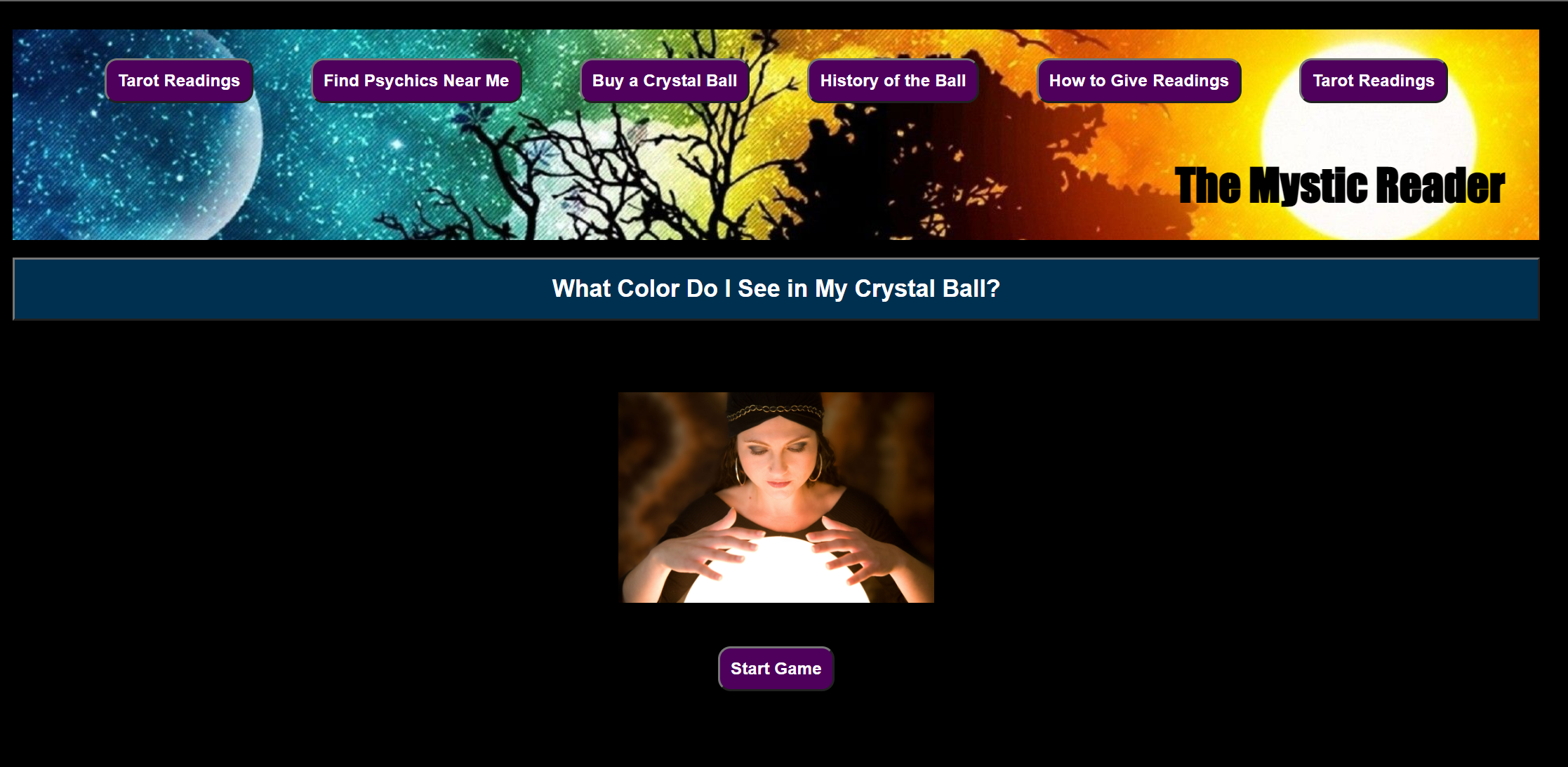
**Magic 8 Ball**

https://medium.com/swlh/creating-a-magic-8-ball-in-html5-with-javascript-40df2a0c6efb

**Number Guessing Game**

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First\_steps/A\_first\_splash

**The Color Changing Game**



Walkthrough

<https://youtu.be/5A2DXOem8bM>

For this assignment, you will write a color guessing game.

This game is similar to the number guessing game exercise from earlier this week, but there are some significant differences. For example, that game involved the generation of one single random number, and no arrays were used in that game. This assessment task requires an array of colors e.g. ['aqua', 'black', 'cyan', . . . ] and the target color which the player has to guess will be a randomly selected color from that array.

Another difference is that you will use a single HTML file with the JavaScript inside the **script** element, rather than an external JavaScript file. This is to make it easier for your instructor to grade your assignment. Best practice in the real world is still to use external JavaScript.

You will set up an HTML page with a button that, when clicked, will begin the game with a prompt that looks like the screenshot below. The appearance may be different depending on your browser and settings, but the text should be more or less as follows:

As you can see, the objective of the game is for the player to guess the color your program is thinking of. The player needs to enter their guess, such as *cyan*.

 A response from the browser will then be shown, as follows:

1. If the color entered by the player is **not**in the pre-set array of colors used by the game, an appropriate message such as this should be shown:

2. If the color entered by the player **is**in the pre-set array of colors, but the color entered by the player is alphabetically higher than the answer, a message such as this should be shown:

3. If the color entered by the player is in the pre-set list of colors, but the color entered is alphabetically lower than the answer, a message such as this is shown:

4. If the color entered by the player is correct, an appropriate message will be shown, as seen below, along with a count of the total number of guesses. Once the player clicks OK, the color of the webpage background will be changed to that color.

**A Flowchart**

Here is a flowchart for the game, showing the basic behavior.

#### Assignment Tasks

To help you organize your approach, detailed instructions are provided below in the form of 2 main tasks, each separated into smaller parts.

**Summary of Task 1:** Set up the HTML page and the core JavaScript for the game so that you can play the game, make guesses, and receive appropriate responses.

**Summary of Task 2:**Update the code to add extra features, including displaying the final number of guesses and changing the background color when the correct guess is made.

#### TASK 1: Set up the HTML and core JavaScript for the game

##### Part 1: Set up the HTML

* Create a basic valid HTML5 document as you learned to do in Week 1, and give it the name **color-guessing-game.html**. Be sure to set the **DOCTYPE**, the **meta charset**, and an appropriate **title**(such as 'Color Guessing Game').
* In the **body** element, add an **h1** element with the text content of **Color Guessing Game**. Beneath this, add a **button** element with the **type**of **"button"**, an **onclick** attribute with the value of **"runGame()"**, and text content of **"Start Game"**(without the quotes).

##### Part 2:  Set up the script element, the colors array, and two main functions

* Add a **script** element to the HTML page, after the **button** element and before the closing **</body>** tag.
* Inside the **script**element, create a global constant named **COLORS\_ARRAY**. Its value should be an **array**with CSS color name strings as its values. You can use these or substitute your own choice of valid CSS color names:

'blue', 'cyan', 'gold', 'gray', 'green', 'magenta', 'orange', 'red', 'white', 'yellow'

* You can find a list of HTML color names [here](https://www.w3schools.com/colors/colors_names.asp)and at other places on the web.
* Beneath this, declare two separate functions. Name the first function **runGame**, and it should have an empty parameter list. Name the second function **checkGuess**- give it two parameters, **guess** and **target.**

##### Part 3: Set up variables for the runGame function and determine the correct guess

* Declare two variables using the **let**keyword: one named **guess** and one named **correct.**Initialize **guess** to the value of an empty string. Initialize **correct**to the Boolean value of **false**.
* Declare a **const** variable named **targetIndex**. For its value, use what you have learned about arrays and generating random numbers using **Math.random()** and **Math.floor()**to generate a random number that is between 0 and the last index number of the**COLORS\_ARRAY** array.
  + The max number should be calculated dynamically rather than hard coded. Recall what you have learned about checking the length of the array, and that due to zero-indexing, the index of the last number of the array is always one less than its length.
  + For example: If your array has 10 colors in it, your program should automatically generate a number between 0 and 9. If you were to add 1 more color to the array, your program should automatically generate a number between 0 and 10.
* Declare a const named **target**and assign to it the value of the **COLORS\_ARRAY**item that has the array index of **targetIndex**. So for example, if the random number stored in **targetIndex** is 3, the color name in **COLORS\_ARRAY**with the index of 3 should now be stored in **target.**(In the example array given above, that value would be 'gray'.)
* **TIP:**This is not necessary for the game logic, but to make it easier for you as a developer to test the game, we suggest that you add a **console.log** at this point to log the target to the console.

##### Part 4: Prompt for a guess until correct guess is made

* Write a**do ... while** loop. The condition to exit the loop should be: **!correct** (this is the same as saying**(correct === false)**)
* Inside the loop block, assign the value of the **guess** variable to the return value of calling **prompt()**, using the following string as the text for the prompt:

'I am thinking of one of these colors:\n\n' + COLORS\_ARRAY +

'\n\nWhat color am I thinking of?\n'

* The **\n** characters will cause a newline in the text.
* Below this and still inside the loop block, set up an **if** statement. The if condition should check if **guess === null**. If the player hits "Cancel" on the prompt, this is the value that will be assigned to **guess**. If so, display an appropriate message to the user and use a **return**statement with no return value. This will exit the entire **runGame**function and end the game.
* Below this, still inside the loop block, assign the value of the variable **correct** to the return value from calling the function **checkGuess**. Pass two arguments to the **checkGuess** function: **guess** and **target**.
* After the program has exited the loop - below it, but still inside the **runGame** function, set up an **alert** that gives the user a congratulations message.

##### Part 5: Write the content for the checkGuess function

* Inside the **checkGuess**function, declare a **let** variable named **correct**and initialize it to **false.**
* Set up an **if** statement. For its condition, use an array method to check if **guess** is a color in **COLORS\_ARRAY** at all. At this point you are not checking if the guess is correct, only if it is in the array. If it is not, show the user an appropriate message.
* Set up an **else if** block that checks if the **guess**is higher than the **target**. If so, then show the user an appropriate message.
* Set up a second **else** **if** block that checks if the **guess**is lower than the **target**. If so, then show the user an appropriate message.
* Set up an **else** block. By this point, the guess should be correct. So set the value of **correct** to **true**.
* After the ending curly brace of the **else** block, but still inside the **checkGuess** function, **return correct**. By that point, that variable should only contain the value **true** if the guess was correct, and **false** if not.

**Note:** It *is* possible to set up the **if**statement in a different way than outlined above, and still have it work in the same way. The above steps are a guideline - if you can set it up a different way and your code still works in the same way, that's OK.

##### At the end of Task 1, test your game and make sure that it runs as demonstrated in the workshop video.

#### TASK 2: Update the code

Update the code with more features, including displaying the total number of guesses and changing the background color when the correct guess is made, and more.

##### Part 1: Display the total number of guesses when the correct guess has been made:

* In the **runGame**function, at the top where the other variables were declared, declare a new variable using **let** named **numTries**or a similar name. Initialize it to **0**.
* Increment **numTries** by 1 inside the **do ... while** loop block by, for example, using the addition assignment operator.
* Note that it is important that the **let** declaration for this variable is outside of the **do ... while** loop. If it was declared inside the loop, then it would be reset at every iteration of the loop.
* In the congratulations message to the user, show the number of tries.

##### Part 2: Change the background color to the correctly guessed color:

* Use the code below in an appropriate place to let the player know their guess has been successful:

document.body.style.background = guess;

##### Part 3: Display the color names with each separated by a comma and a space

* Look into using the *array*.join() method to see how you can show the color names from the array using a comma and a space as a separator, so that it shows up like this:

instead of like this:

##### At the end of Task 2, test your game and make sure that it runs as demonstrated at the end of the workshop video.

#### BONUS CHALLENGES

The following challenges are not required by the assignment. If you have time left, try them out!

* Use the *array*.sort() method to sort the list of colors alphabetically when showing them to the user. You'll want to make sure that your colors are *not* in alphabetical order in the array when you test this; otherwise, you won't be able to tell if it's working.
* Research the use of the *string*.toLowerCase() method to help make the guess case insensitive. You can do so by making sure that both the target and the guess are both in the same case before being compared.
* Instead of only showing the number of tries at the end of the game, how would you show it for each try, even for incorrect guesses?
* Add more HTML and CSS to the page to improve its appearance. Your choice in how to carry this out!

#### A Note on Lexicographical Order

In the game, we use lexicographical order to provide hints to the player. When comparing lower case strings of characters from the alphabet, lexicographical order works in this way:

The first characters of each string are compared alphabetically, with **'a'** at the lowest end and**'z'** at the highest end. If the characters are the same, then the next character of each string is compared. Thus**, 'ant' < 'zoo**', **'meow' < 'merman'**, and so on.

Here are some examples of strings where the first string is higher/greater than the second string:

* sat > sad
* bags > bag
* thin > fat
* good > bad

Here are more examples of strings where the first string is lower/less than the second string.

* rag < rat
* bit < bite
* food < water
* potato < potatoes

Arrays & Strings Mad Lib

<https://youtu.be/M97I3tRaIoE>

* [W3Schools - JavaScript Array Methods](https://www.w3schools.com/js/js_array_methods.asp)
* [W3Schools - JavaScript Array Sort](https://www.w3schools.com/js/js_array_sort.asp)
* [JavaScript.info - Array methods](https://javascript.info/array-methods)
* [MDN - Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array)

<https://www.youtube.com/watch?v=PE7Wr-xzTU8&ab_channel=BreeHall%7CBytesofBree>

Troy Towers Form