

# Assignment: Color Guessing Game

Submit by June 19, 11:59 PM PDT

## Important Information

It is especially important to submit this assignment before the deadline, June 19, 11:59 PM PDT, because it must be graded by others. If you submit late, there may not be enough classmates around to review your work. This makes it difficult - and in some cases, impossible - to produce a grade. Submit on time to avoid these risks.

## Instructions

My submission

Discussions

## Instructions

By making a game, you will gain experience in JavaScript, including variables, arrays, loops and functions.

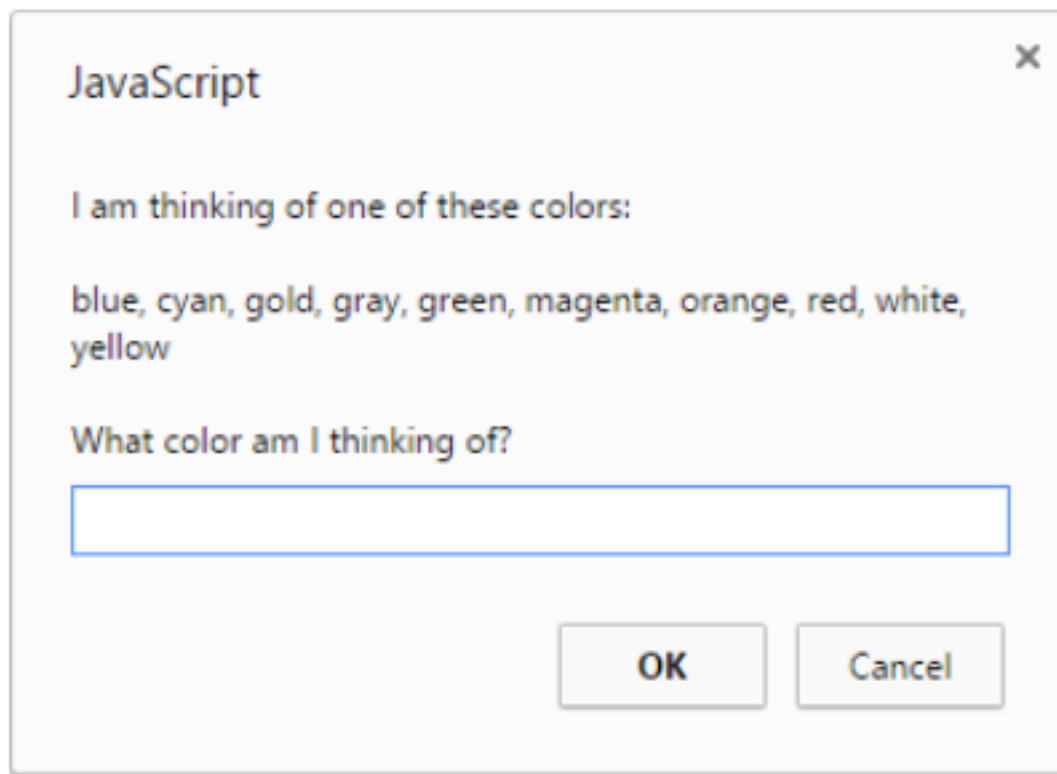
### Instructions

less ^

### The Task

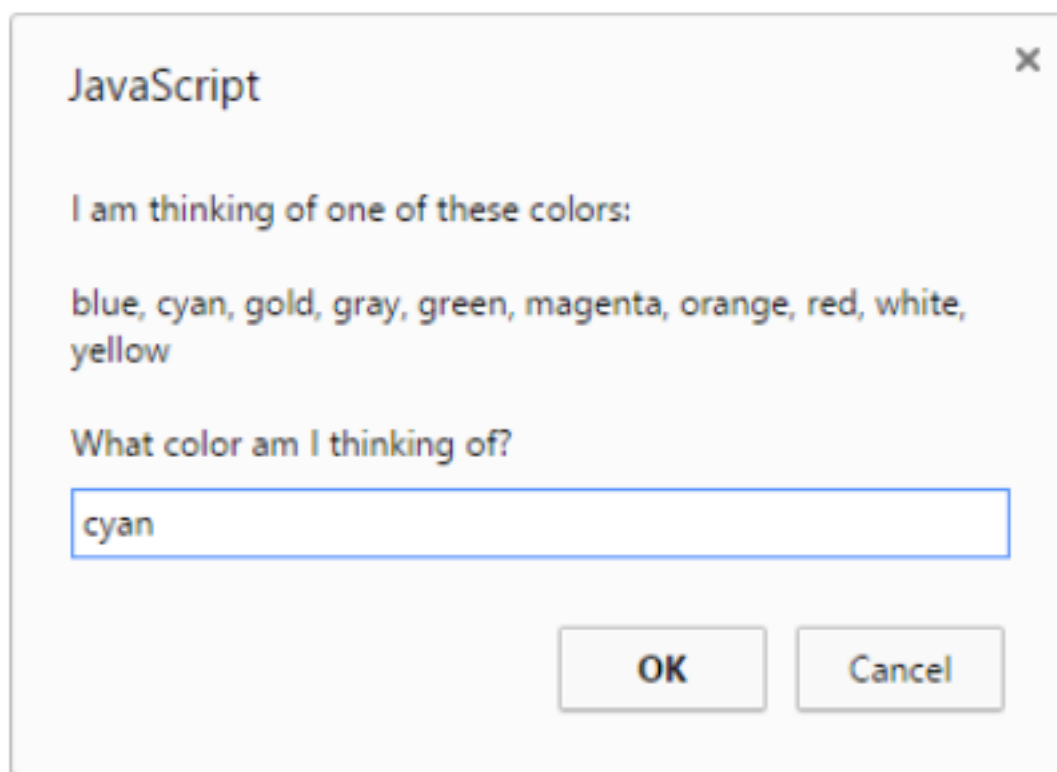
This assessment task requires you to make a simple color guessing game. This game is similar to the example number guessing game we have discussed on the course, but there are some significant differences. Please see the video for an example walk-through.

When the web page is loaded, the player sees a message like this.



A JavaScript dialog box titled "JavaScript" with a close button (X) in the top right corner. The text inside reads: "I am thinking of one of these colors:" followed by a list of colors: "blue, cyan, gold, gray, green, magenta, orange, red, white, yellow". Below this, it asks "What color am I thinking of?" and provides an empty text input field. At the bottom, there are two buttons: "OK" and "Cancel".

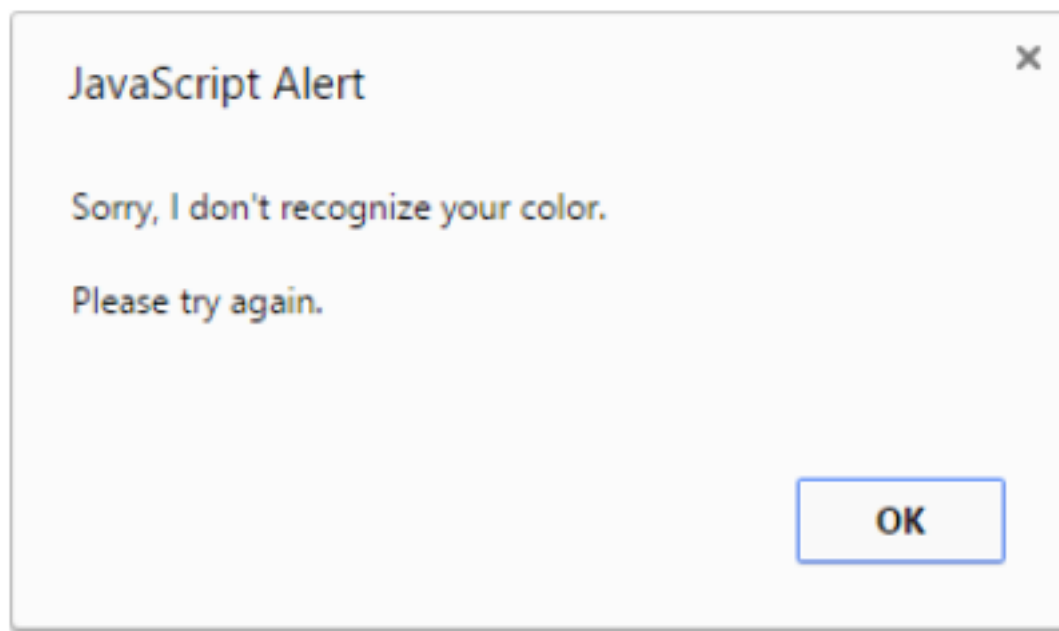
As you can see, the objective of the game is for the player to guess the color which your program is thinking of. The player needs to enter their guess, such as cyan.



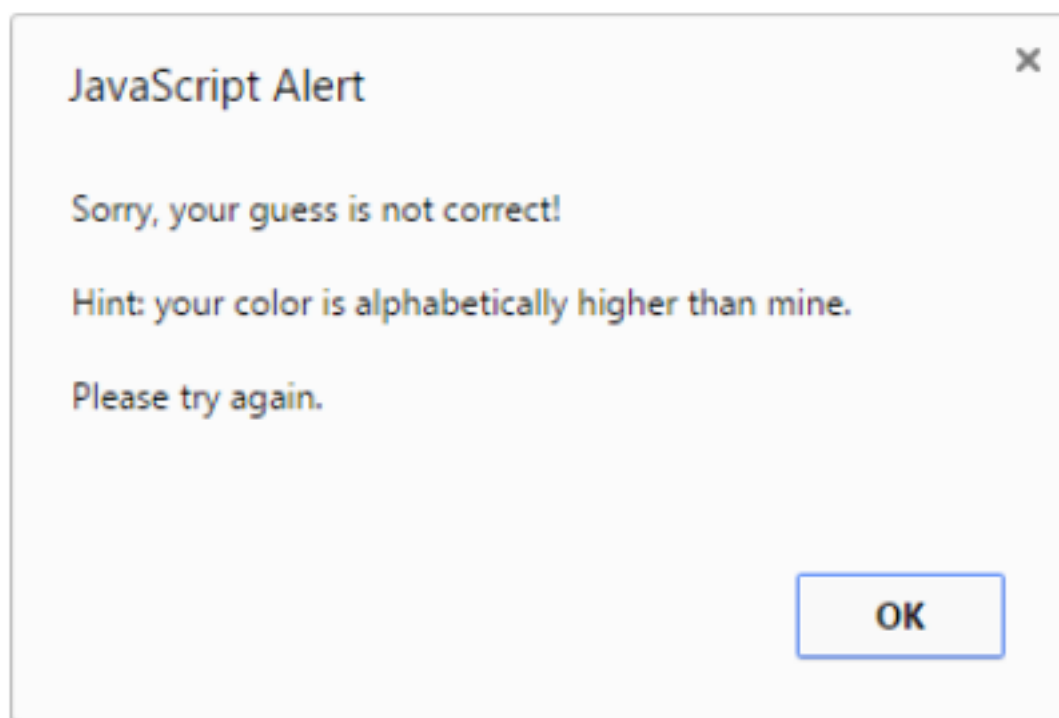
A JavaScript dialog box titled "JavaScript" with a close button (X) in the top right corner. The text inside reads: "I am thinking of one of these colors:" followed by a list of colors: "blue, cyan, gold, gray, green, magenta, orange, red, white, yellow". Below this, it asks "What color am I thinking of?" and provides a text input field containing the word "cyan". At the bottom, there are two buttons: "OK" and "Cancel".

A response from the browser will then be shown.

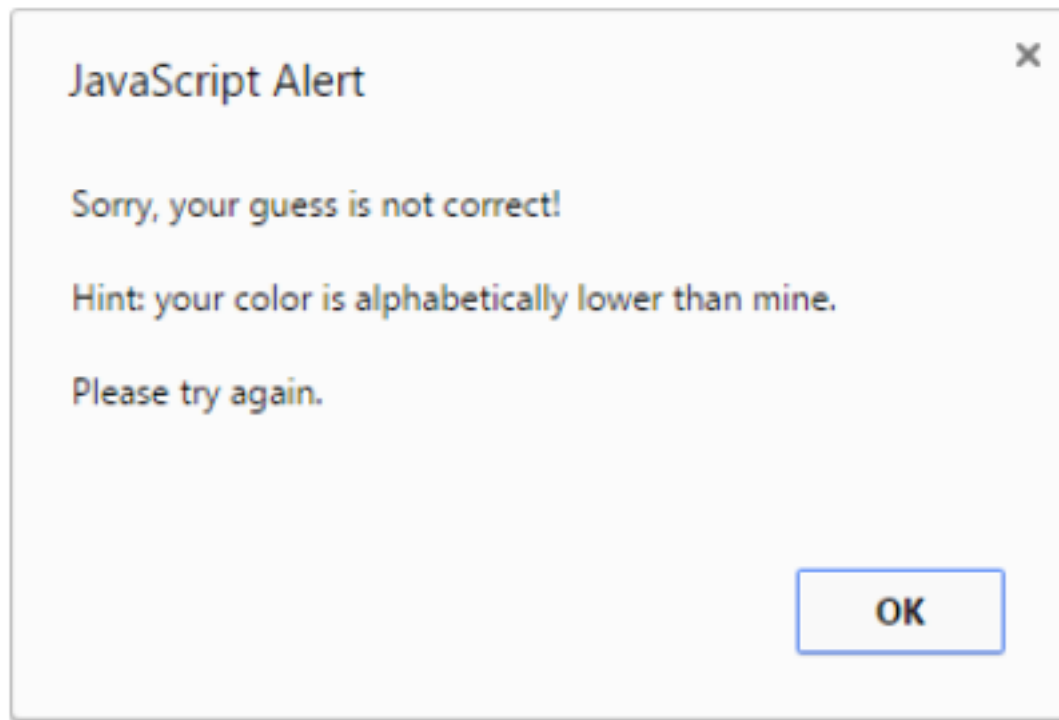
1. If the color entered by the player is not in the array of colors used by the game, an appropriate message like this is shown:



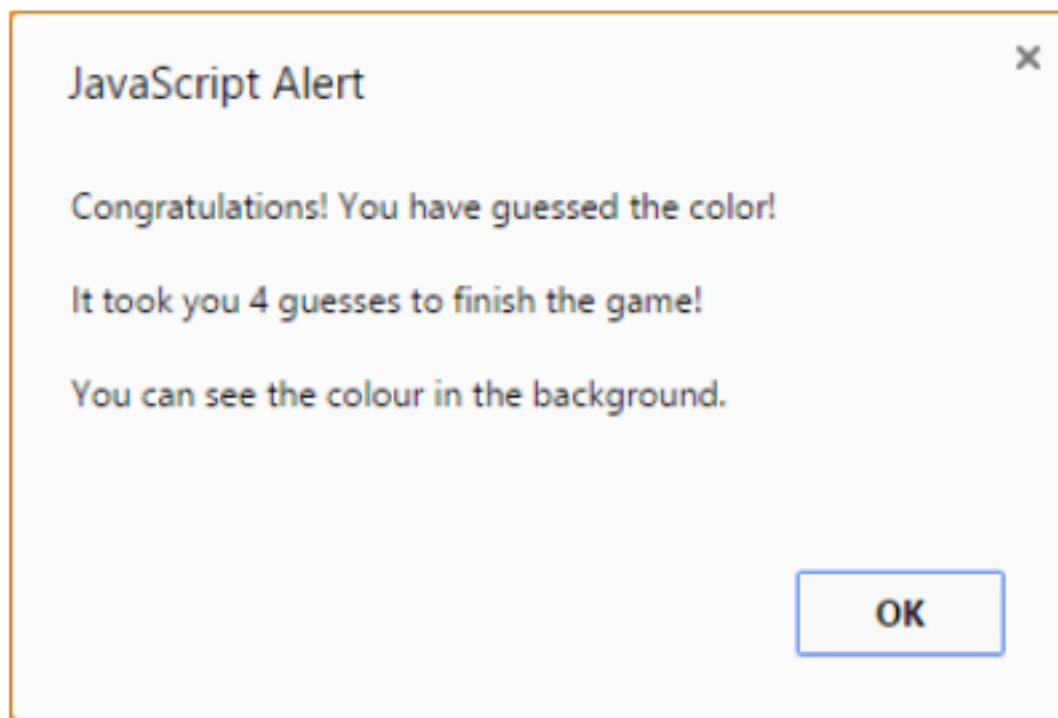
2. If the color entered by the player is in the list of colors used by the game but the color entered by the player is alphabetically higher than the answer, an appropriate message like this is shown:



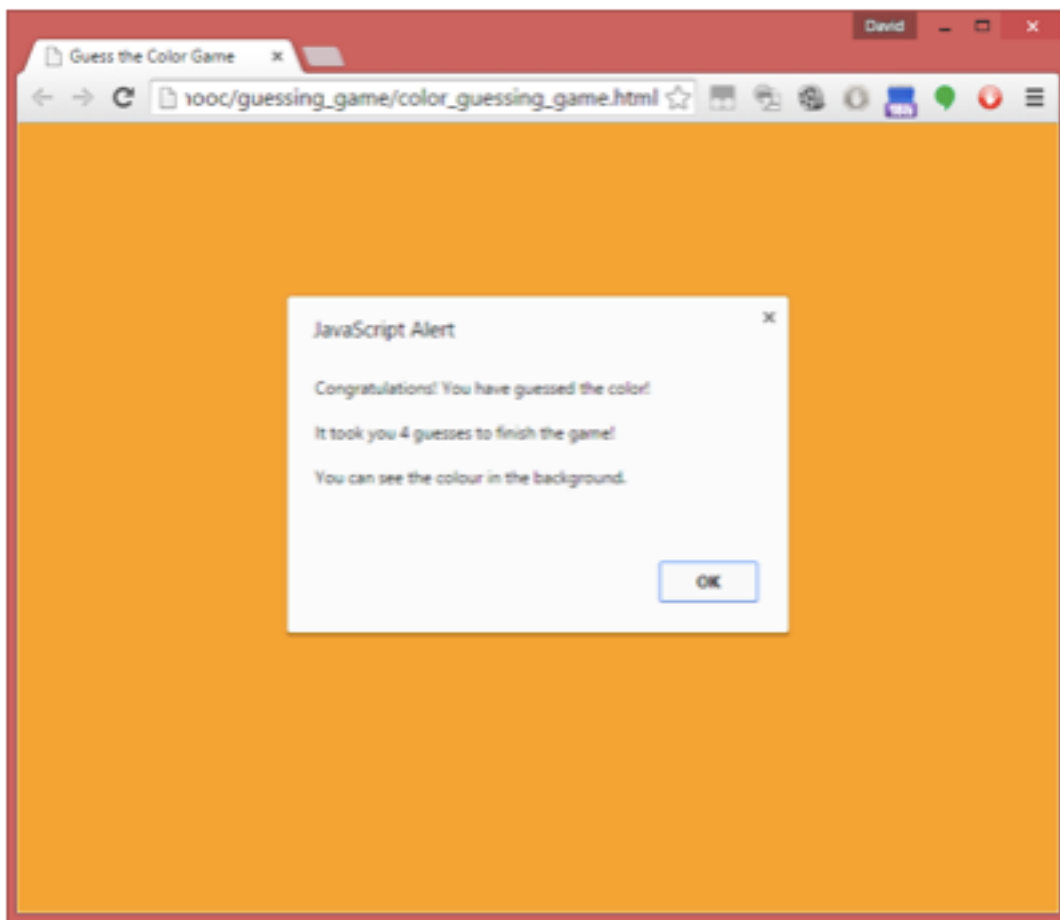
3. If the color entered by the player is in the list of colors used by the game but the color entered by the player is alphabetically lower than the answer, an appropriate message like this is shown:



4. If the color entered by the player is correct the web page background is changed to that color and an appropriate message is shown:

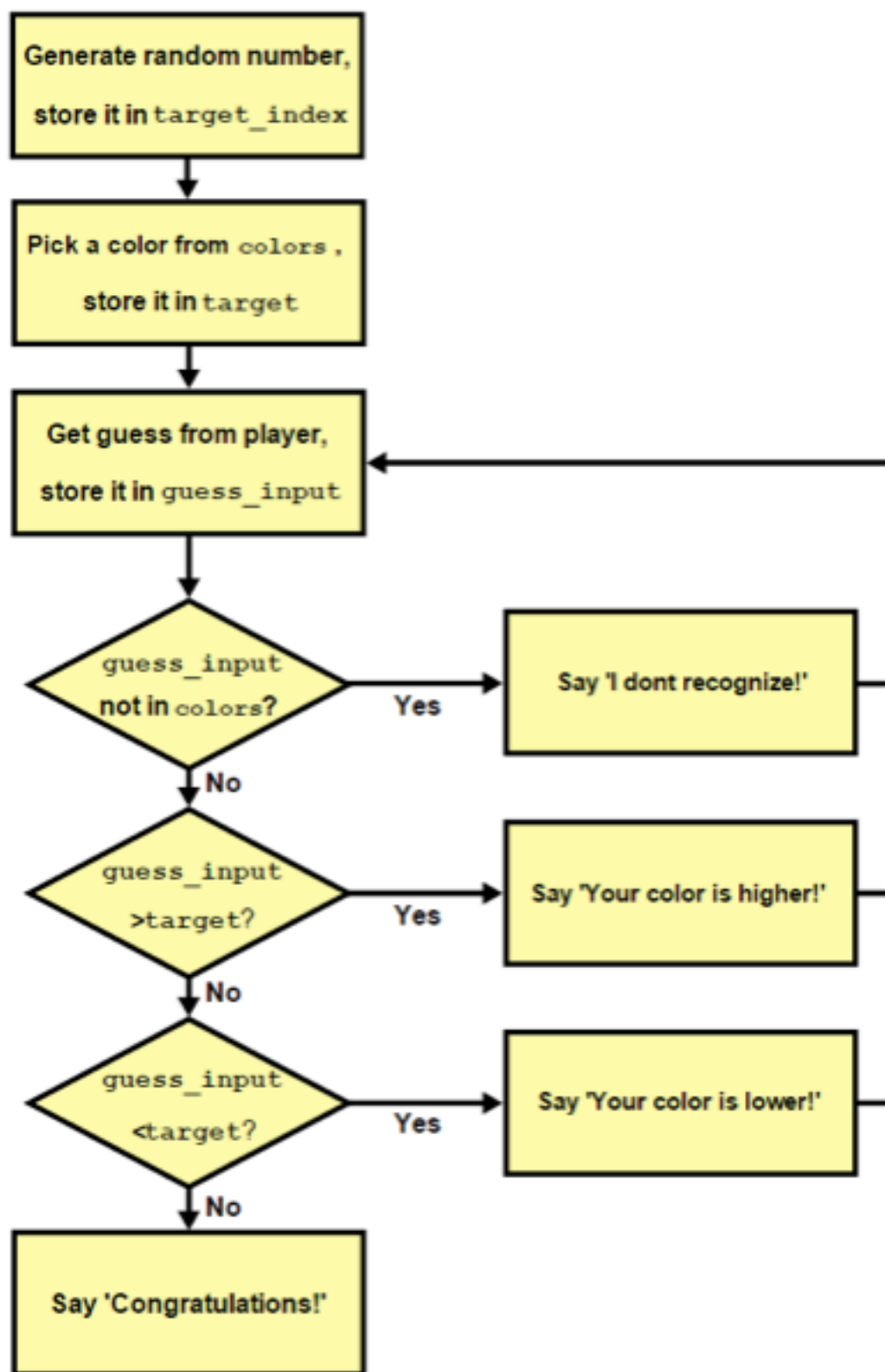


This is what a web page looks like when the message is shown on top of a web page where the background has been changed to orange.



## A Flowchart

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



### A Note on Alphabetic Order

In the game we use alphabetical order to provide hints to the player. Here are some examples of two strings where the first string is alphabetically higher than the second string.

- `sat > sad`
- `bags > bag`
- `thin > fat`
- `good > bad`

Here are some examples of two strings where the first string is alphabetically lower than the second string.

- `rag < rat`
- `bit < bite`
- `food < water`

- potato < potatoes

In JavaScript you can easily determine whether one string is alphabetically higher or lower than another string e.g.

```
var string1="hat";  
var string2="hit";  
if (string1 > string2)  
    alert("string1 is alphabetically higher");  
else  
if (string1 < string2)  
    alert("string1 is alphabetically lower");
```

Wikipedia has a good discussion of alphabetical order at  
[https://en.wikipedia.org/wiki/Alphabetical\\_order#Basic\\_order\\_and\\_example](https://en.wikipedia.org/wiki/Alphabetical_order#Basic_order_and_example)

## Technical Overview

This task is similar to the example number guessing game we have discussed on the course. However, 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.

```
colors=["aqua", "black", "cyan", . . . .];
```

The target color which the player has to guess is a randomly selected color from that array.

The names must be HTML color names. This is so that when the player successfully guesses the color it can be used for the web page background color, to show what the color actually looks like.

You can find a list of HTML color names at

[http://www.w3schools.com/html/html\\_colornames.asp](http://www.w3schools.com/html/html_colornames.asp)

and other places on the web. Some of them are regular English names such as “red”, “black”, and so on, but there are also more unusual color names you can use such as “DarkSeaGreen” and “RebeccaPurple”.

In your array the colors do not have to be listed in alphabetical order. However, in terms of playing the game, when the list of colors is shown to the player it would be more helpful if they are in alphabetical order. If you wish, you can easily use JavaScript to sort them, as we have discussed on the course.

One way to set the background color of a web page is

```
myBody=document.getElementsByTagName("body")[0];
```

```
myBody.style.background=name_of_color;
```

## JavaScript Content

Your file is likely to have JavaScript content which is similar to the number guessing game we have discussed before. However, one difference is that for this task you are required to put your JavaScript and HTML in **one single file**, unlike the number guessing game which used a separate file for HTML and JavaScript for better management. The only reason for the need to use one file in this task is to make marking your work and the work of others much easier.

Here is the likely JavaScript content of your game.

1. Declaration of global variables and array e.g. color[].

Setting the initial values.

2. The main game function - do\_game()

2.1. Generate a random number in the range  
[0, length of array-1]

2.2. Assign target to the random color in the array

2.3. A while loop, which repeats until finished is true:

- ask the player for their guess using prompt()

- increase a variable count by 1

- check the player's guess using check\_guess()

3. Check the input - check\_guess()

Check whether the player's guess is:

3.1. not a color in the array:

show an appropriate message and return false

3.2. alphabetically higher than the target:

show an appropriate message and return false

3.3. alphabetically lower than the target:

show an appropriate message and return false

3.4. correct:

change the background color to the target,

show an appropriate message

which includes the total number of guesses

stored in variable count,

return true

The numbers shown above (such as 2.3) are simply to give you an idea of an appropriate order for the code. You don't have to use those numbers in any particular way and you don't have to follow exactly the order shown, although it may help you to do so.



See the flowchart image above for more guidance.

No HTML content is needed for this game, except for a `body` element which is used to show the color when the player has successfully guessed it.

To start the game one way is to use a `body onload` event, in the way that we have discussed on the course. This can be used to trigger the execution of `do_game()`.

## Which Browser to Use

The game was originally developed using the Chrome browser. We have checked that it works in all modern browsers. However, Internet Explorer only allows a small amount of text to be shown in a `prompt()` message, which means Internet Explorer is not suitable for this project.

## Discussion Video

Please watch the accompanying video, which shows and discusses the completed task.

## What to Submit

This assessment task is closely related to the number guessing game discussed on the course. The code for that game has also been released in the system. You are encouraged to watch the video and look at the code, in order to learn from it.

There are 2 parts of this assessment. You need to submit 1 file for part 1 and 1 file for part 2.

For both part 1 and part 2 you can only submit one single HTML file, which includes the JavaScript code within it. That means there are no links to external JavaScript files or any other files in this assessment task.

Part 1 is a simple version of the game. When the file is loaded a color is randomly selected from an array of colors and a loop begins which repeatedly asks the player what the color is. However, no feedback is given to the player depending on what he/she enters. The loop simply finishes when the player enters the correct color. Make sure you save a copy of this as, for example, `part1.html`, before you move on to the next part.

For Part 2, take your part 1 file and extend it by completing the entire game. Make sure you save a copy as, for example, `part2.html`.

Please note that this assignment does not require any style rules or HTML content (apart from a `body`), but you are welcome to add them if you wish.

Here are the more specific requirements and further information for both parts.

## Part 1 Requirements.

- Your part 1 work should be a simple 'game' which uses a loop. In the loop, the player is shown the list of colors and is asked for their guess. However, no feedback is given to the player based upon their input. In other words, the function `check_guess()` is not required in part 1. The loop finishes when the player enters the correct color.
- A summary:
  - Include a list of colors in an array

- In function `do_game()`:
  - Randomly select one of those colors in the array as the target
  - Display the target (to help with debugging and marking) e.g. `alert(target);`
  - Go into a loop which
    - Shows the array of colors and asks the player for their guess
    - Stops if the player's guess is the same as the target

## Part 2 Requirements.

- Part 2 is the completed game, as described in the instructions and demonstrated in the accompanying video.
- Here is a summary:
  - Include a list of colors in an array
  - In function `do_game()`:
    - Randomly select one of those colors in the array as the target
    - Optionally: display the target (to help with debugging and marking) e.g. `alert(target);`
    - Go into a loop which
      - Shows the array of colors and asks the player for their guess
      - Using the function `check_guess()`:
      - Displays a message such as 'I don't recognize that color!' If the text entered by the player is not in the array of colors, OR:
      - Displays a message such as 'Your input is alphabetically higher than mine!' if that is true, OR:
      - Displays a message such as 'Your input is alphabetically lower than mine!' if that is true, OR:
      - If the player's input is the same as the target: changes the web page background color to the target color and displays a message which includes the total number of guesses such as 'You are right! You took 8 guesses!'
      - Stops if the player's input is the same as the target