This assignment is going to be the continuation of the Guessing Game Lab.

Before you begin tackling the problem below, you will need to be sure that you first open your GuessingGame html file and CSS file and save them as UpdatedGuessingGame.html (where LastName is your Last name) and rename HW10.css to HW11.css.

You will also need to be sure to open your html file and update the link to the CSS file so that it is now HW11.css as opposed to HW10.css.

The rest of the assignment will require you to create a JavaScript file that will model the logic of the game below.

Say that you have been given the following prompt:

I want you to create a simple guess the number type game. It should choose a random number between 1 and 100, then challenge the player to guess the number in 10 turns. After each turn the player should be told if they are right or wrong, and if they are wrong, whether the guess was too low or too high. It should also tell the player what numbers they previously guessed. The game will end once the player guesses correctly, or once they run out of turns. When the game ends, the player should be given an option to start playing again.

We can create a JavaScript file to be embedded in our html file so that we can play the guessing game.

First, open a new text file and save it as **guessingGame.js.**This will create a new JavaScript file.

Then, what you will need to do is to break the prompt down into a series of steps:

1. Generate a random number between 1 and 100.
2. Record the turn number the player is on. Start it on 1.
3. Provide the player with a way to guess what the number is.
4. Once a guess has been submitted first record it somewhere so the user can see their previous guesses.
5. Next, check whether it is the correct number.
6. If it is correct:
   1. Display congratulations message.
   2. Stop the player from being able to enter more guesses (this would mess the game up).
   3. Display control allowing the player to restart the game.
7. If it is wrong and the player has turns left:
   1. Tell the player they are wrong.
   2. Allow them to enter another guess.
   3. Increment the turn number by 1.
8. If it is wrong and the player has no turns left:
   1. Tell the player it is game over.
   2. Stop the player from being able to enter more guesses (this would mess the game up).
   3. Display control allowing the player to restart the game.
9. Once the game restarts, make sure the game logic and UI are completely reset, then go back to step 1.

Let's now move forward, looking at how you can turn these steps into code.

Inside of your guessing game script (**guessingGame.js**), you will add the following lines of code:

var randomNumber = Math.floor(Math.random() \* 100) + 1;

var guesses = document.querySelector('.guesses');

var lastResult = document.querySelector('.lastResult');

var lowOrHi = document.querySelector('.lowOrHi');

var guessSubmit = document.querySelector('.guessSubmit');

var guessField = document.querySelector('.guessField');

var guessCount = 1;

var resetButton;

These lines of code are used to set up the variables needed for the guessing game. The first line generates a random number between 1 and 100. Specifically, the floor command is used to ensure that the number generated does not have any decimal places.

The next three variables (guesses, lastResult, and lowOrHi) are used in order to each made to store a reference to the results paragraphs in the HTML, and are used to insert values into the paragraphs into the html file you created in the last lab.

The next two variables (guessSubmit, guessField) store references to the form text input and submit button and are used to control submitting the guess later on.

The final two variables (guessCount, resetButton) store a guess count of 1 (used to keep track of how many guesses the player has had).

Next, we will need to create a function that will check the user's guess against the random number generated.

Start by creating a function named **checkGuess** after the resetButton variable. The function should start off looking similar to the one created below:

function checkGuess() {

}

Then within the checkGuess function, do the following:

* Declare a variable called userGuess and sets its value to the current value entered inside the text field. Also, this value will be run through the built-in Number() method, just to make sure the value is definitely a number using the following code:

var userGuess = Number(guessField.value);

* The next few lines will define a conditional to tell whether or not it is the player's first guess. If the player is on their first guess, it will display that the previous guesses are empty, afterwards, the program will display the player's previous guesses.

if (guessCount === 1) {

guesses.textContent = 'Previous guesses: ';

}

* After the first conditional, add the following line so that the user's previous guesses will continue to be displayed separated by spaces:

guesses.textContent += userGuess + ' ';

* Afterwards, add the next set of conditionals, that will do the following:
  + The first if(){ } checks whether the user's guess is equal to the randomNumber set at the top of our JavaScript. If it is, the player has guessed correctly and the game is won, so we show the player a congratulations message with a nice green color, clear the contents of the Low/High guess information box, and run a function called setGameOver(), which we'll discuss later.
  + Now we've chained another test onto the end of the last one using an else if(){ } structure. This one checks whether this turn is the user's last turn. If it is, the program does the same thing as in the previous block, except with a game over message instead of a congratulations message.
  + The final block chained onto the end of this code (the else { }) contains code that is only run if neither of the other two tests returns true (i.e. the player didn't guess right, but they have more guesses left). In this case we tell them they are wrong, then we perform another conditional test to check whether the guess was higher or lower than the answer, displaying a further message as appropriate to tell them higher or lower.
* Add the code for these operations:

if (userGuess === randomNumber) {

lastResult.textContent = 'Congratulations! You got it right!';

lastResult.style.backgroundColor = 'green';

lowOrHi.textContent = '';

setGameOver();

} else if (guessCount === 10) {

lastResult.textContent = '!!!GAME OVER!!!';

setGameOver();

} else {

lastResult.textContent = 'Wrong!';

lastResult.style.backgroundColor = 'red';

if(userGuess < randomNumber) {

lowOrHi.textContent = 'Last guess was too low!';

} else if(userGuess > randomNumber) {

lowOrHi.textContent = 'Last guess was too high!';

}

}

* Last to close out the function, we will add the following code to prepare for the next guess:

guessCount++;

guessField.value = '';

guessField.focus();

* Once your checkGuess function is complete, add the following code so that the code executes when something is entered:

guessSubmit.addEventListener('click', checkGuess);

* afterwards, you will need to create the following function using the following code:

function setGameOver() {

guessField.disabled = true;

guessSubmit.disabled = true;

resetButton = document.createElement('button');

resetButton.textContent = 'Start new game';

document.body.appendChild(resetButton);

resetButton.addEventListener('click', resetGame);

}

* + The first two lines disable the form text input and button by setting their disabled properties to true. This is necessary, because if we didn't, the user could submit more guesses after the game is over, which would mess things up.
  + The next three lines generate a new [<button>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button) element, set its text label to "Start new game", and add it to the bottom of our existing HTML.
  + The final line sets an event listener on our new button so that when it is clicked, a function called resetGame() is run.

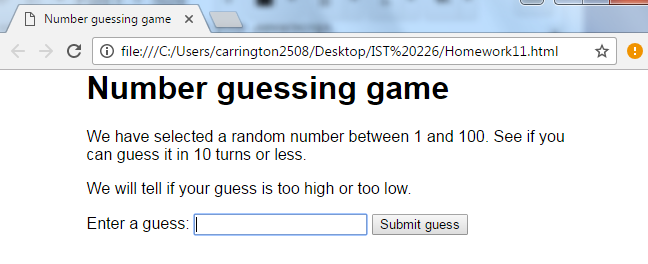
Lastly, define this function too! Add the following code, again to the bottom of your JavaScript:

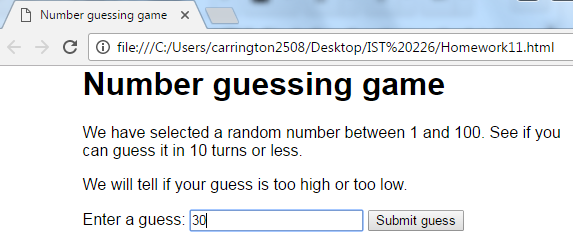
function resetGame() {  
guessCount = 1; var resetParas = document.querySelectorAll('.resultParas p');  
for (var i = 0 ; i < resetParas.length ; i++) {  
resetParas[i].textContent = '';  
} resetButton.parentNode.removeChild(resetButton); guessField.disabled = false;  
guessSubmit.disabled = false;  
guessField.value = '';  
guessField.focus(); lastResult.style.backgroundColor = 'white'; randomNumber = Math.floor(Math.random() \* 100) + 1;  
}

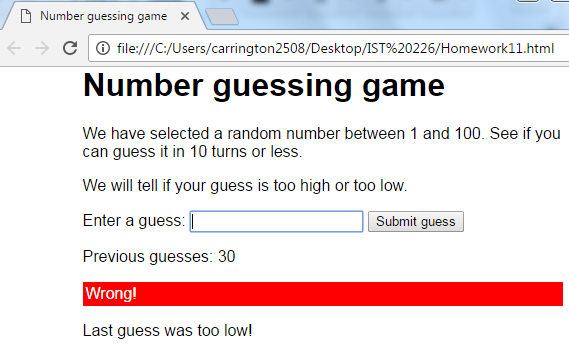
Be sure to save your JavaScript file, then add the following code in your HTML file, before the closing <body> tag:

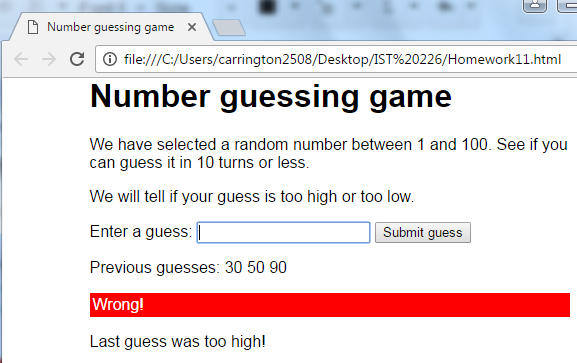
<script src="guessingGame.js"></script>

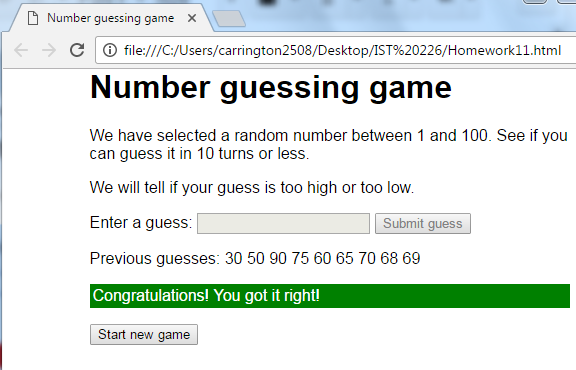
When you open your HTML file in a web browser, the game should be playable like the following example (NOTE: YOUR GAME WILL LIKELY GENERATE A DIFFERENT RANDOM NUMBER THAN THE ONE PICTURED):











As you can see, for my example, my number ended up being 69, and it took quite a few guesses to get it right.