Project 2A: Javascript Game: WordGuess
CSE 100/INFO 100 Fluency with Information Technology

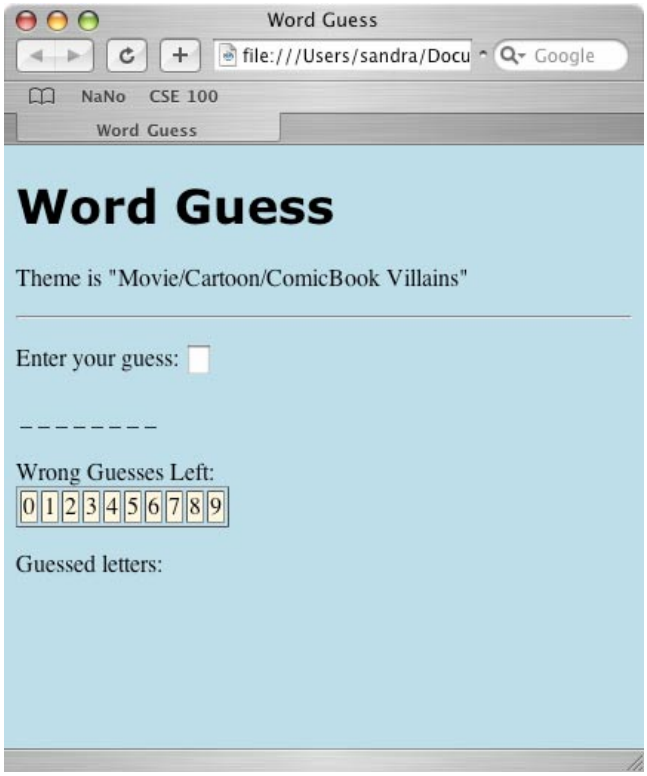
Project Overview

In this project, you will create a web browser game called WordGuess (also known as "Hangman") using Javascript and HTML. The goal someone playing the game is to guess a secret word.

For this project, in addition to programming, you will need to comment your code, as well as include a report section on your page that details your process in writing this program, e.g. the problems you encountered, the process you went through to debug them, etc.

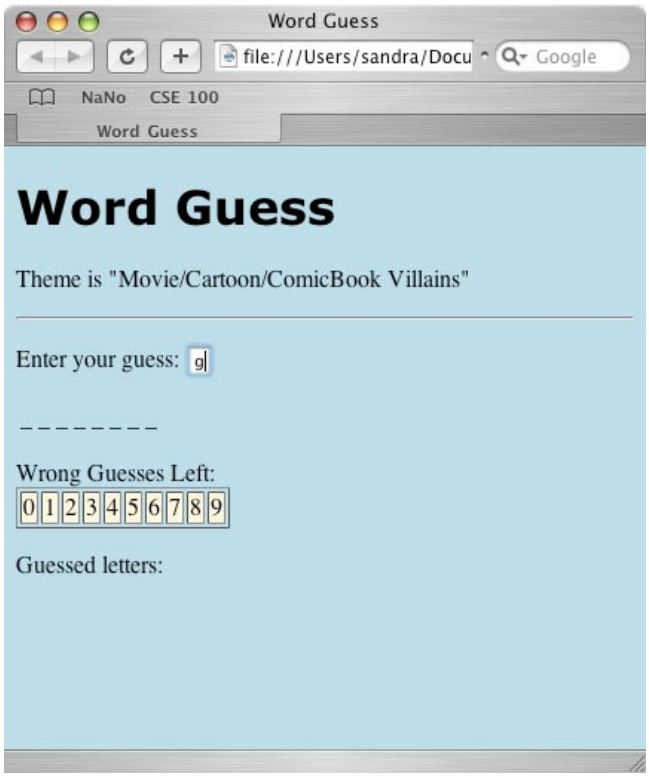
This project is divided into a Part A and a Part B. Part A will be done this week as a lab. In Part A, we will guide you step by step to completing this program. Part B will be dependent on your own work. Like a homework, it is something to complete on your own time. There will be an electronic turn-in similar to Project 1.

The screenshots on this page were created with Safari on Mac OS X; if you use a different operating system or browser, your project may look slightly different.

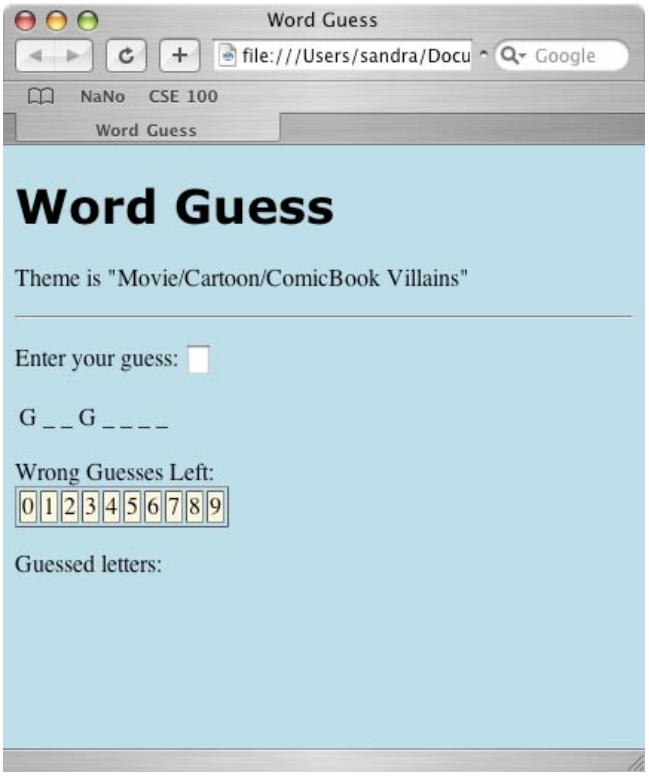


Game Overview

1. The computer will pick a word from a list of words that it has stored. The player does not know what this word is.
2. The computer will display a number of blank lines, each of which represents a letter from the word. For example, if the secret word is "Gargamel," the computer will display eight blank lines, one for each letter in the word "Gargamel."
3. Now, the player must guess what the secret word is. The player does this by guessing one letter at a time, and the computer will let the user know whether he or she is right or not. There will be an input text box for the user to enter a letter.

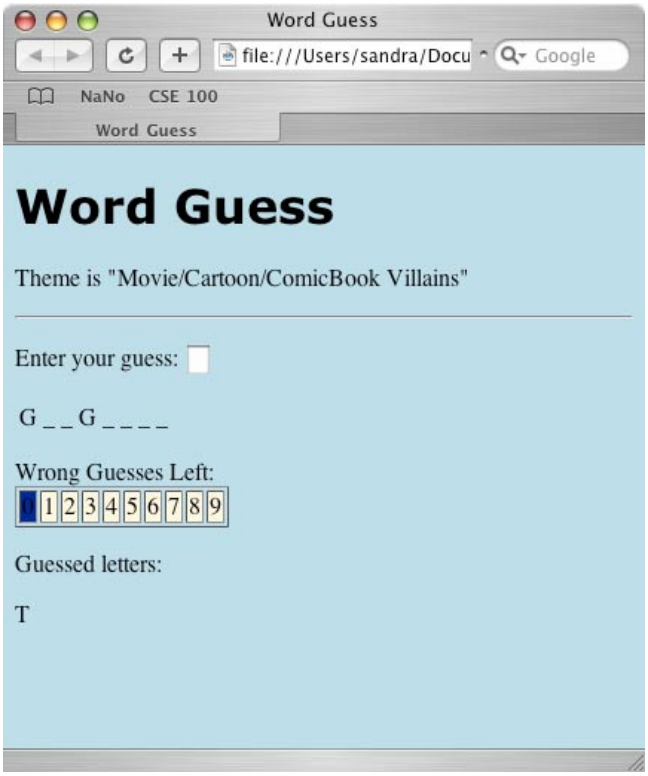


a. If the letter is in the secret word, our program will replace the blanks it created earlier with the letter, in all the places that the letter appears. In the "Gargamel" example, if players enter "g" as their guess, the computer will display “G _ _ G _ _ _ _”. Notice that our game does not bother with capitalization (nor spaces, for that matter).

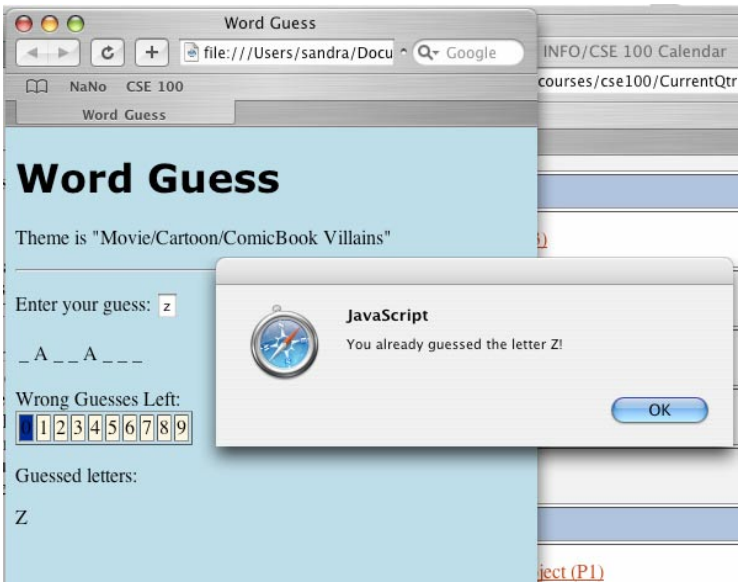


b. If the guess was wrong, the computer will write to an area of the screen that displays incorrect guesses. Additionally, the progress bar will move by one box, to show the user that they have used up one of their wrong guesses. The player gets 10 wrong guesses. (Notice that the progress

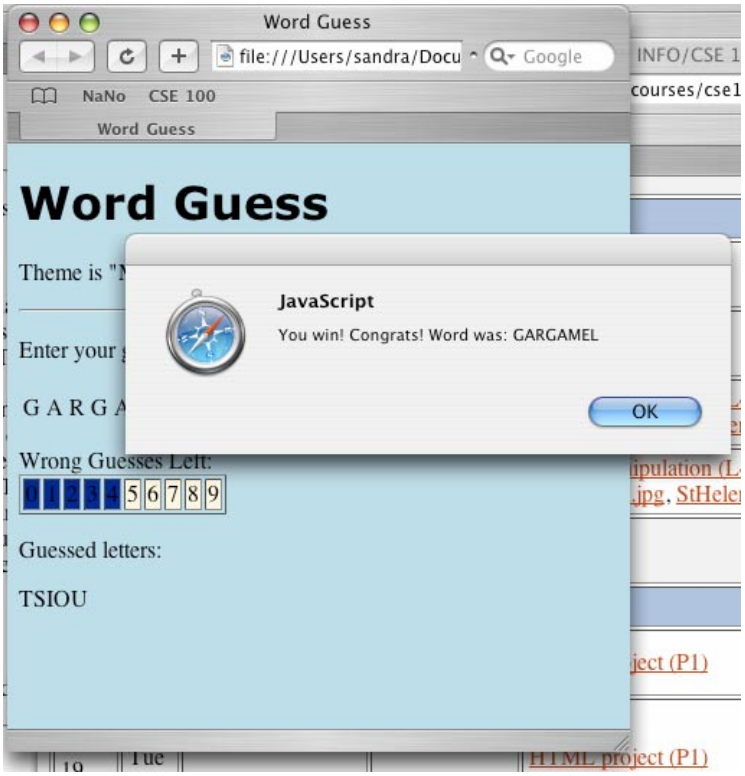
bar has 10 boxes, from 0-9.) So, if the user guesses the letter "t" in the "Gargamel" example, nothing will be written on the blank lines, and the letter "t" will appear in the incorrect guesses area, and one of the boxes will be shaded. The correct guesses (say we had guessed “g” first, and then “t”) stay where they are.



c. If you guess a letter that you’ve already guessed, nothing happens, and a pop-up window will tell you that you’ve already guessed that letter.



4. The game continues in this fashion, with the player guessing one letter at a time. If the player correctly guesses all the letters in the word before the progress bar has reached the last box, the player wins.



5. If the player uses up the boxes in the progress bar before the word has been guessed, the player loses, and a box pops up informing the user of this.

Project 2A

Part A is is due by the end of your second lab this week. It will take the form of a file called **project2a.html**. Please turn this file in electronically as well as place it under the directory **public_html/fit100/project2a** in your Dante home directory, as with Project 1.

A few notes before starting

1. Some general requirements for the HTML document:
    a. It must validate. You may have errors with validating HTML end tags that are inside your Javascript code, take a look at http://www.htmlhelp.com/tools/validator/problems.html#script, which is about writing HTML in a script element
    b. It must have a descriptive title and a page heading.
    c. It must have a section of text at the bottom, divided from the rest of your page by a horizontal rule (a line, or an `<hr>`) that describes the process of doing Part A. It should have its own heading called "My Thoughts on Project 2A." In an organized fashion, every time you encounter a problem, write it down in that section and write the exact steps of how you debugged it. At the end of the project 2a, write what you learned about programming through doing this project.
    d. The whole page should have an attractive and professional appearance when completed.
2. Document your code using `//` or `/* . . . */` for comments. Some specifics:
    a. For each variable you've created, explain what it is used to store.
    b. Above each function, describe what the function does.
    c. Write above any loops or if-else statements what that loop/if-else statement does.
    Anything else you want to document—remember, comments are as much for you as it is for us. You don't want to work on this one night, and then come back the next day and have no idea what you typed the previous night.
    d. All of the code should have a neat, readable appearance. Using indentation and blank lines and spaces are often helpful in this respect.

Steps for Starting

1. Make sure you understand the game! Play it with a friend using pencil and paper.
2. Create the html document and give it a title and page heading (as mentioned earlier).
3. On your webpage, underneath your heading, write an introduction and some brief rules for your game, in your own words (don't copy and paste our rules) so that visitors to your site know how to play, and to convince yourself that you know the rules! If you can't give the rules in English, you won't be able to do so in Javascript :)
4. Create an area on the page where the user will enter input. It should have text like "Enter your guess:" and then an input box next to it. Your input tag should have the following attributes:
    a. The id attribute should have the value "guess"
    b. The type attribute should have the value "text"
    c. Your input text box should hold only one letter. You may do this by adding a "maxlength" attribute to your input tag, as follows: `maxlength="1"`
    d. Your onchange attribute should call the checkGuess() function (which we will write in the next step).
    e. Reload your webpage to make sure your input box shows up properly.
5. Now, write the checkGuess() function. For now, it will not do anything useful nor take any parameters; it will merely display the fact that it exists.
    a. First of all, decide where in the HTML file the function should go.
    b. Now within the script area, write the `checkGuess` function with an empty parameter list, and put inside the body a statement to run the alert function and show a box that says "Running checkGuess function!" This should be the ONLY thing inside your `checkGuess` function for

now. [In this assignment, there are a number of places where you are asked to use alert boxes. The purpose is to help you see how the program is running. These boxes won't necessarily be in the final version of the game that a player would see.]

c. Reload your page and enter some input into your input box. Your alert box should appear! Don't go further until you get this working.

Continuing...

1. Come up with a theme for the words in your game. For instance, in the example, the theme was "Movie/Cartoon/Comic Book Villains," and one of the example secret words was Gargamel, who is the villain from the Smurfs. You may pick anything you like, e.g. "Fruits" and have your secret words be things like "apple," "banana," "orange," etc. Your secret words should not have spaces in them (for now, at least). We will store 10 words.
   a. Indicate your theme by writing a paragraph under your heading that says something like "Theme is:" and saying what your theme is.
2. In the `<head>` of your document, at the beginning of your Javascript area, create an array that holds the secret words. Do this by creating a variable called wordlist and assign it to a new array that can hold 10 items. You can do this by inserting the following:

```
var wordlist = new Array(10);
```

   d. Add the secret words to your wordlist. Do this by accessing each cell of your array one at a time using the index (be sure to start at 0, and then go up to 9), and assigning a secret word to them. For instance:

```
wordlist[0] = 'Gargamel';
```

   e. Test your array by adding an alert box after it to show that the 10th item (the item at index 9) has been stored. Do this by inserting the line

```
alert('My 10th secret word is: ' + wordlist[9]);
```

   Reload your webpage to see if an alert box shows up with your 10th secret word in it. If so, you're in good shape. If not, debug!

3. When the game is played, the program will have to randomly pick a secret word from our wordlist. To do this, we will write a function called pickWord() that returns a word randomly from our list
   a. In the head of your document, at the beginning of your script area, create a variable called secretWord. This will store eventually store the secret word, but right now, it is uninitialized.
   b. Also in the head of your document, in the script section *after* where you filled in all the words for your wordlist and after your alert box, add the following function:

```
function pickWord() {
   var randomNumber = Math.floor(Math.random() * wordlist.length);
   secretWord = wordlist[randomNumber].toUpperCase();
}
```

   Here is an explanation of what all of that means!
      i. The first line in the body of the function uses a couple of built-in Javascript function to select a random number and store it in the variable randomNumber. Math.random() returns a decimal number from 0 to l (0 included, 1 not included). We then multiply that by 10 to get a decimal number between 0 to 10 (0 included, 10 not included). Then we use the Math.floor function, also built-in, to round down our number (i.e. we cut off, or truncate, anything after the decimal point) so that it is an integer.
      ii. The second line uses our randomNumber variable to index into the wordlist array to select a word from the wordlist. It also changes that word to upper case, in case it isn't already. *In our game, everything is going to be done in upper case!* It then returns
3. Now, add to the end of the body of the pickWord() function an alert box that shows what word we just selected. It should show the text "Secret word is: " and show the secret word itself.

4. In the body of your document, create an area to write your Javascript code beneath your input box. Add the following lines, and then reload your page. You should see the alert box you just created in the previous step.

```
pickWord();
```

If you don't see the secret word in the alert box -- figure out what is wrong and don't continue until you do!

Blanking Out...

1. Now we need to write a writeBlanks() function, which will write the blank lines to represent each character of the secret word. This function will be called immediately after pickWord is called. You can go ahead and add this call now. Where does it go? What does it look like? (Hint: where did the call to pickWord go? What did it look like? If you can't answer these questions, stop and figure it out now.
2. We will define the writeBlanks() function in the script section of the head, underneath the Javascript code we've already written. [Review: what is the difference between "calling" a function and "defining" a function?] Write the outline for the function now, in the proper place. In the body of that function, put nothing for now except an alert box that states we are in the writeBlanks function. Check to be sure your function works, that is, see if the alert box shows up when you reload the page. If not -- figure out why not.
3. The writeBlanks() function is supposed to display a row of '_' characters. We will do this in a rather fancy way, by creating an HTML table with one row. This will be one of the most complex parts of the program. Each column, or cell, in our row, will represent one blank, and we will write an underscore (`'_'`) in that cell. In other words, if my secret word were "cat" with three characters, this is the HMTL we want:

```
<table>
<tr>
     <td id='blank0'>_</td>
     <td id='blank1'>_</td>
     <td id='blank2'>_</td>
</tr>
</table>
```

4. [Note for later (Part B): each cell has an ID, so that we can refer to it elsewhere in the code, i.e. if I want to change the first blank to a "c" when my player has correctly guessed it, I can find this item by looking for an element named "blank0."]
5. To write this HMTL into the document, we will use a series of `document.write()` calls. All of them belong in the body of the writeBlanks function. Start with this one

```
document.write('<table>');
```

6. Now, do the same thing to begin the table row (i.e. `'<tr>'`) and then a separate document.write() to end that table row, and then to end the table. Notice that we did NOT write any <td> tags -- yet.
7. Now, we need HTML to create the columns, or cells, for each character of our secret word. Since we do not know ahead of time what secret word will be picked, we can't always draw the same number of table cells. So, we are going to use a "for" loop here. For each character of our secretWord, our loop will run and write a table data cell `with` an underscore (to represent a blank) inside that cell. As noted before, our `<td>` tag will have an `id` attribute, so that we can refer to each cell later on when we want to write to it. The catch is that each cell should have a *unique* id. To do this, we will form the `id` attribute of each cell will be "`blankx`", where x stands for the number of the cell. So, what we are trying to achieve is the following. If my secret word was "cat" I would want three blank lines, that would be created like this, inside a single row of my table:

```
<td id='blank0'>_</td>
<td id='blank1'>_</td>
<td id='blank2'>_</td>
```

   a. To do this particular example, a for loop would need to loop from 0 to 2, in other words, from 0 to one less than the length of our secret word "cat." Of course, the secret word won't always be

three letters long. Your next task is to write a for loop. Where does it go? Answer: it is creating the cells of the row, so it has to be between the code that generates the <tr> and </tr> tags. Your loop starts with something like this:

for (var i = 0; i < secretWord.length; i++) {

i. Try to understand all the parts of this before continuing.
ii. Now, for the body of the loop, that is, the part inside { }. We need to write one `<td>` tag, with the proper id attribute and the underscore, as shown earlier. Note that we only have to write this once, because we are within a for loop. Do this part yourself.

b. Run and check your program. If no blank lines show up, something is wrong. If the wrong number of blanks show up, something is wrong. Keep working until everything is just right.

Finishing up Part A

1. Validate your page and make sure everything is running properly.
2. Don't forget to make sure you've written all your notes in your "Thoughts" section, and to document your code with comments as specified above. You're done with Project 2A. It should look like the picture below. You may add some style to it if you like. Pat yourself on the back for a job well done!