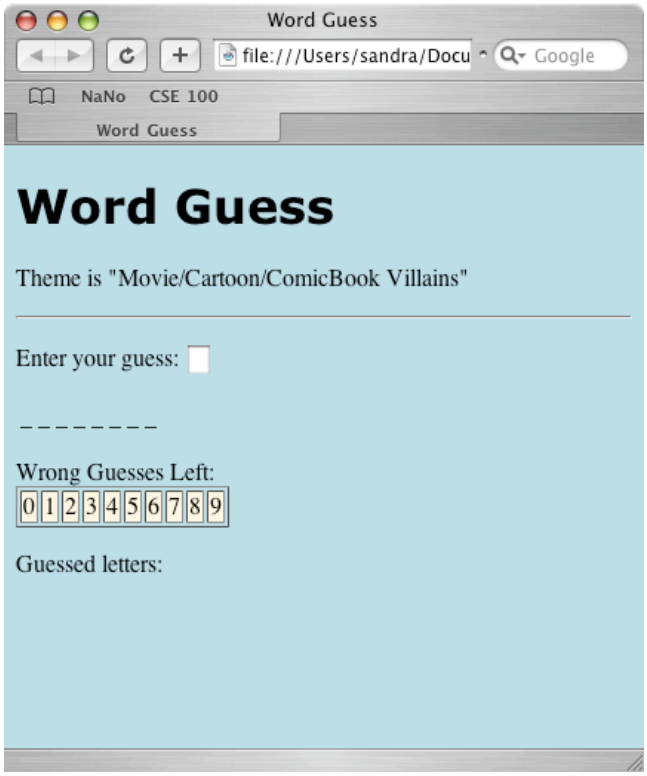
Project 2A: JavaScript Game: WordGuess
CSE 100/INFO 100 Fluency with Information Technology

Project Overview

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

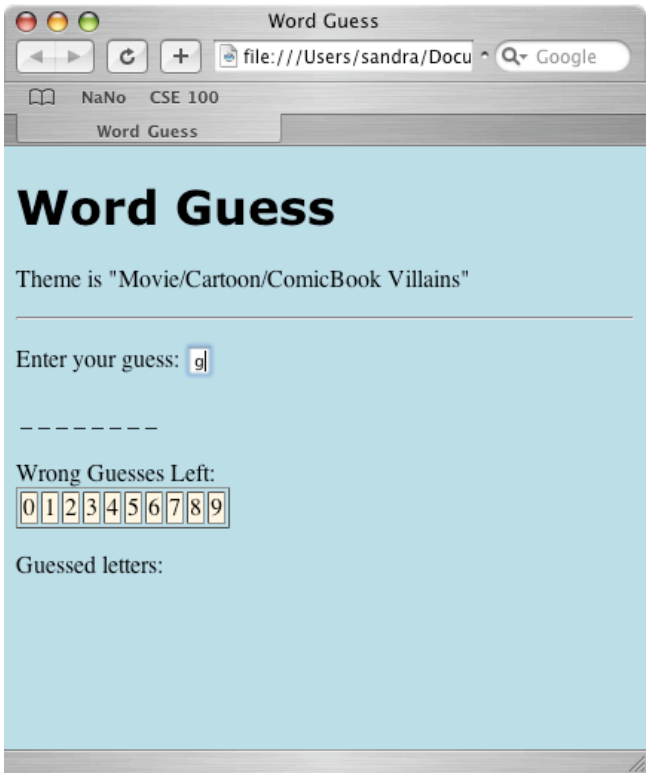
For this project, in addition to programming, you will need to comment your code, as well as include a section on your page that details your process in writing this program, the problems you encountered, the process you went through to debug them, etc.

This project is divided into Part A and Part B. Part A will guide you toward completing this program, whereas Part B will be a little more dependent on your own work. Additionally, there will be extra credit portions that allow you to customize your game.

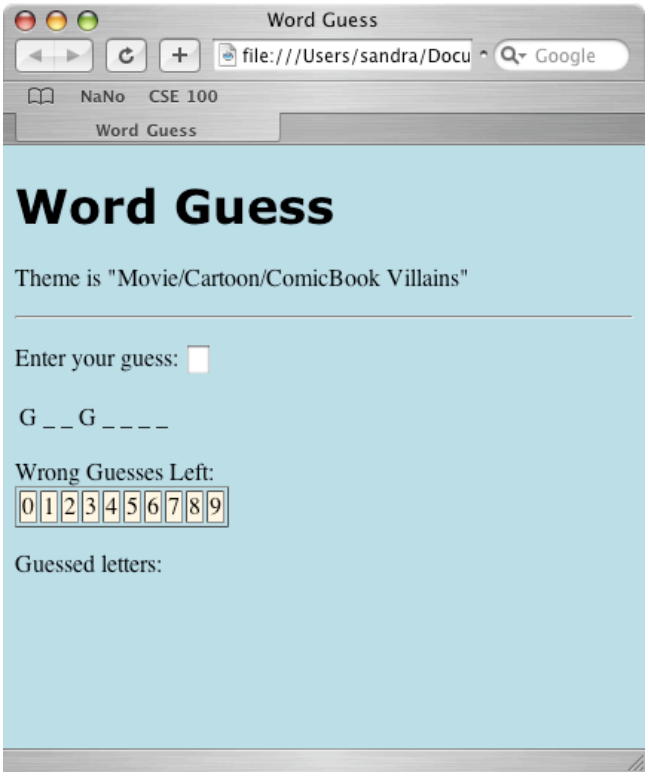


Game Overview

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 spaces (indicated by underscores), each of which represents a letter from the word. For example, if the secret word is "Gargamel," the computer will display eight blank spaces, 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.

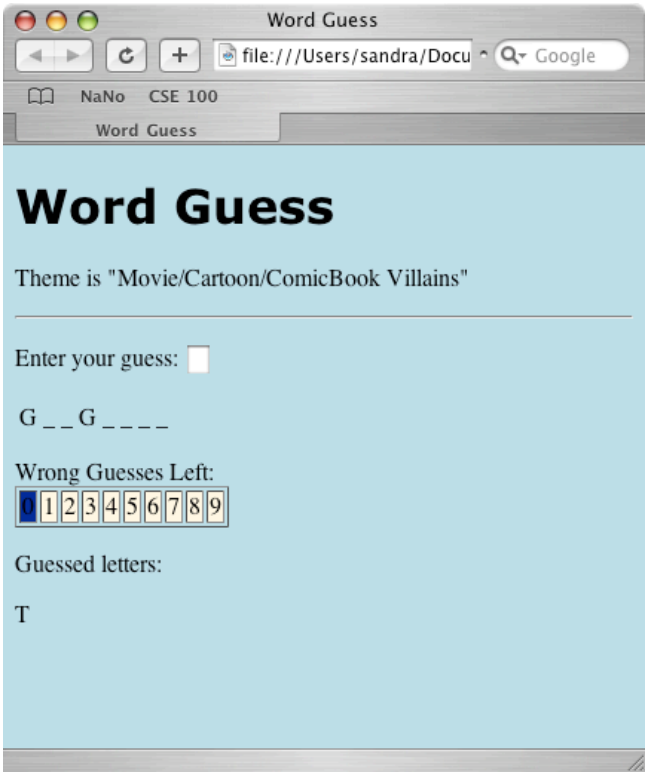


a. If the letter is in the secret word, our program will replace the blanks it created earlier with the letter, in all the places that the letter appears. In the "Gargamel" example, if players enter "g" as their guess, the computer will display "G _ _ G _ _ _ _". Notice that our game does not bother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

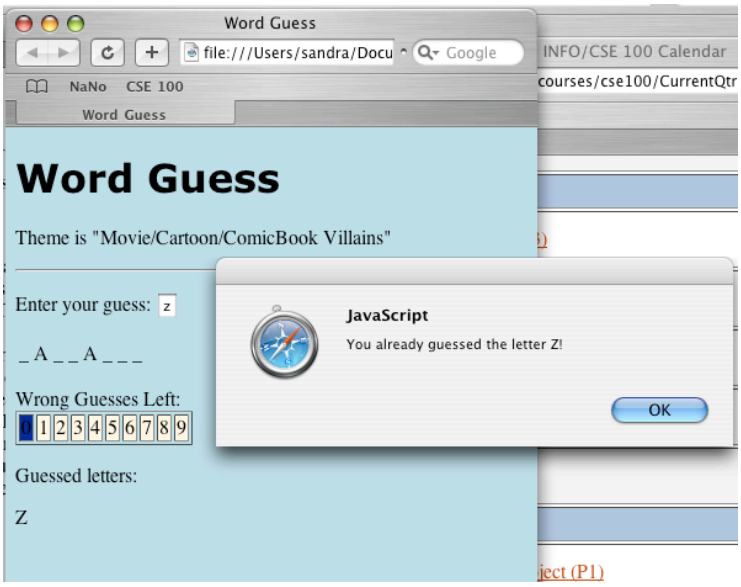


b. If the guess was wrong, the computer will write to an area of the screen that displays incorrect guesses. Additionally, the progress bar will move by one box, to show the user that they have used up one of their wrong guesses. The player gets 10 wrong guesses. (Notice that the progress bar has 10 boxes, from 0-9.) So, if the user guesses the letter "t" in the "Gargamel" example, nothing will

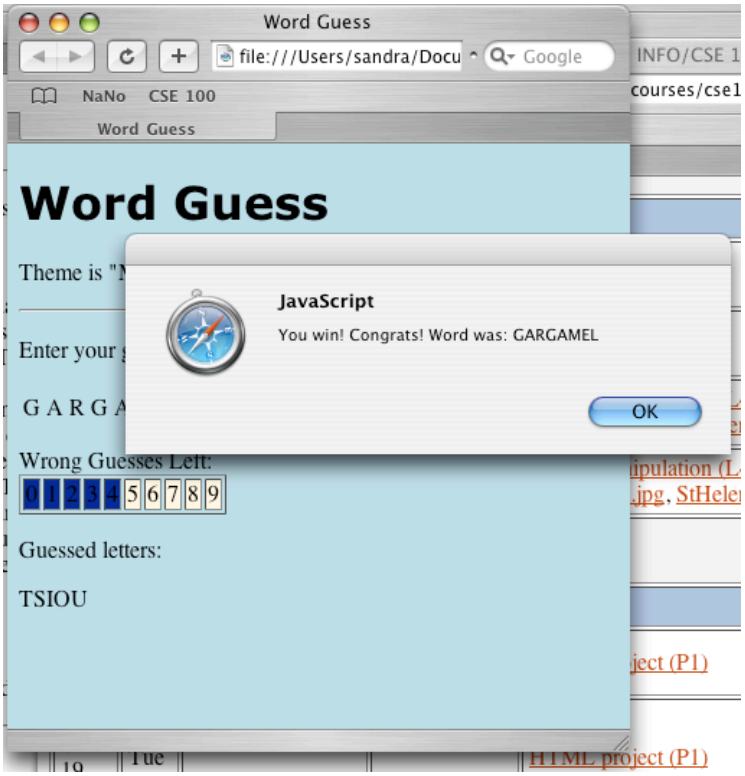
be written on the blank spaces, and the letter "t" will appear in the incorrect guesses area, and one of the boxes will be shaded. The correct guesses (say we had guessed “g” first, and then “t”) stay where they are.



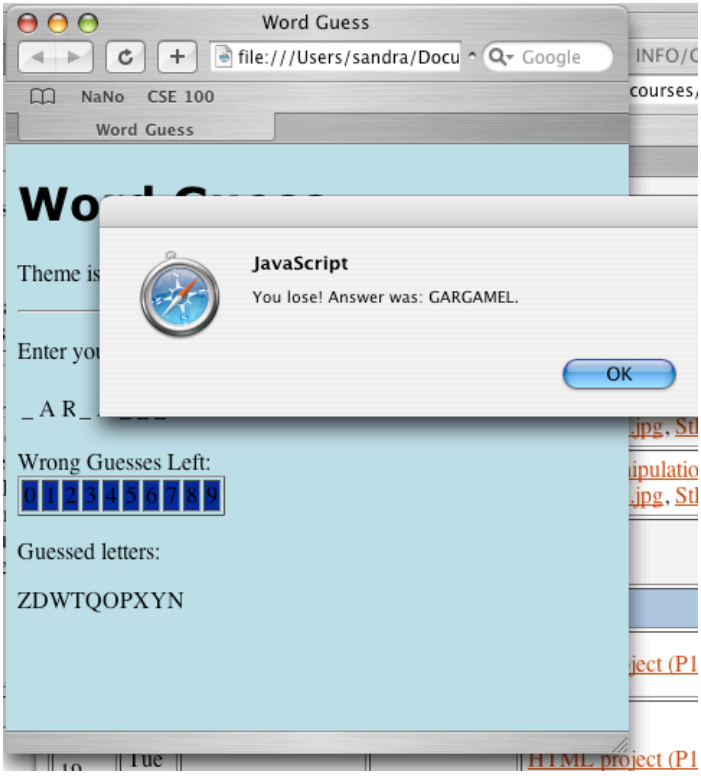
c. If you guess a letter that you’ve already guessed, nothing happens, and a pop-up window will tell you that you’ve already guessed that letter.



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 worth **30 points** and is due on Monday, May 8, 2006, by 10 pm. It will consist of a file called **project2a.html**. Please turn this file into Catalyst, as well as place it under the directory **public_html/fit100/project2a** in your Dante home directory.

1. **The .html document**
   Create an HTML document:
   a. **(2 points)** File is in public_html/fit100/project2a directory and called 'project2a.html'
   b. **(1 point)** 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.
   c. **(1 point)** It must have a descriptive title and a page heading.
   d. **(5 points)** 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. If you do not do a thorough job here, you will lose points

2. **Commenting**
   (5 points) Document all your JavaScript code, using `//` or `/* . . . */` for comments. If you don't do a thorough job here, you will lose points.
   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 and what that loop/`if-else` statement does.
   d. Anything else you want to document—remember, comments help you as much as they help 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.

3. **Introduction and Game Rules**
   (1 point) 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 my rules) so that visitors to your site know how to play.

4. **Input Box**
   (3 points) Create a new paragraph in the body of your HTML document. It should have the text "Enter your guess:" and then an `input` box next to it, inside the paragraph. 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. You must restrict the number of letters that can be inputted into this textbox to 1. You may do this by adding a `maxlength` attribute to your input tag. For example: `maxlength="5"`
   d. Your `onchange` attribute should call a function that will be written in the next section: `checkGuess()`.
   e. The `size` attribute determines the visible length of the box. Set the size of the input box to be no more than 1 character length. For example, `size="5"` will be approximately five characters long.
   f. Reload your webpage to make sure your input box shows up properly.

5. **checkGuess() function**

(2 points) 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, use the `<script>` tag, with the proper attribute(s) inside the tag, i.e. the `type` attribute should be set to `text/javascript`, in the `head` element of the document to indicate that you are going to write JavaScript. Be sure to end the script tag with. `</script>`
b. Now within the script area, write the `checkGuess()` function with an empty parameter list, and put inside the `checkGuess()` function body a statement to run the alert function. The alert box should say "Running checkGuess function!" This should be the ONLY thing inside your `checkGuess()` function for now.
c. Reload your page and enter some input into your input box to make sure your alert box shows up.

6. **Choosing a Theme and Creating Secret Words**

(3 points) 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 statement under your heading that says "Theme is:" and saying what your theme is.
b. 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. The following example holds a list of 12 items:

```
var wordlist = new Array(12);
```

c. 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. The following example code stores "Gargamel" in the 12th cell:

```
wordlist[11] = 'Gargamel';
```

d. Test your array by adding an alert box after the last array index assignment to show that the 10th item (the item at index 9) has been stored. The following code displays the 12th cell of an array.

```
alert('My 12th secret word is: ' + wordlist[11]);
```

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!

7. **pickWord() and writeBlanks function**

(2 points for pickWord() function, 5 points for writeBlanks() function) Underneath the text input box, we are going to create an area to draw blank spaces (indicated by underscores) to represent the letters of our secret word. We will be adding this code in the body of our HTML document. To do this, we first have to randomly pick a secret word from our wordlist. To facilitate this action we will write a function called `pickWord()` that stores a random word from our array into a global variable. Then we will write a function called `writeBlanks()` to write the blank spaces.

a. In the `<head>` of your document, at the beginning of your `<script>` area, create a variable called `secretWord`. This will 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 statement, add the following function:

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

The following is an explanation of the function you just added:

i. The first line in the body of the function uses a couple of built-in JavaScript functions 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 the value of `randomNumber` to retrieve a word from the `randomNumber`th index of the array `wordlist`. It also changes that word to upper case regardless of any prior capitalization. *In our game, everything is going to be done in upper case!* It then returns the value (a word) of the `randomNumber`th index of `wordlist` into the variable `secretWord`.

c. Add to the end of the body of the `pickWord()` function an alert box that display what word we just selected. The alert box should say, "Secret word is [secret word]" Where [secret word] is the random word picked by the `pickWord()` function.

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

```
pickWord();
```

e. This line will call the `pickWord()` function, which will pick our secret word and store it in the `secretWord` variable. Now we need to write the `writeBlanks()` function, which will write the blank spaces to represent each character of the secret word. We will define the `writeBlanks()` function in the `<script>` section of the `<head>` underneath the JavaScript code we've already written.

i. Create the `writeBlanks()` function in the head of your document inside the script area. In the body of the `writeBlanks()` function, include a statement at the end that creates an alert box that tells the user we just ran the `writeBlanks()` function. Check to see if your function works.

ii. The `writeBlanks()` function will create a table with one row. 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 was "cat" with three characters, this is what we want:

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

Do not type this code. The above code was just an illustration of what the code would look like if our secret word was "cat". Note that each cell has an ID, so that we can refer to it later, i.e. if I wanted to change the first blank to a "c", I can find this item by looking for an element named "blank0".

iii. To do this, we will use the predefined JavaScript `document.write()` function to write HTML into our document. We're using JavaScript commands to write HTML tags; isn't that exciting?

1. To begin creating the table we will add a line at the beginning of the body of the `writeBlanks()` function using the `document.write()` function. Add the following line to the beginning of the body of your `writeBlanks()` function:

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

Once you have added the above line, add the necessary line to close the table.

2. Using the same idea, use the `document.write()` function to add a table row (`'<tr>'`) within the two `table` tags. Next add another `document.write()` function to close the table row.
3. Between the two lines where we create and end the table row, we need 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. To solve this problem, 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 "`<td></td>`" and write an underscore (to represent a blank) inside that cell. Additionally, 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 `id` attribute of each cell will be "`blankx`", where x stands for the number of the cell. What we are trying to achieve is the following: If my secret word was "cat" I would want three blank spaces inside a single row of my table that would be created like this:

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

To do this, our `for` loop needs to loop from 0 to 2. In other words from 0 to one less than the length of our secret word "cat." Create a `for` loop (refer to Lab 7 or the April 27 lecture if you've forgotten how to create loops) in between the `document.write()` lines that begin and end the table rows. For the three statements that govern how many times the loops run, write the following:

   a. To initialize the loop, create a variable called `i` that is initialized to 0.
   b. Next, create the condition under which you want your loop to continue running. We want our loop to run as long as our loop variable `i` is less than the length of our secret word. So, your loop condition should be: `i < secretWord.length`
   c. Finally, you want your `i` variable to increase every time you run the loop, so you should update your loop control index with `i++`.
   d. All your statements should be separated with a "`;`"

Now, for the body of your loop, we need to write the `<td>` tag, with the proper `id` attribute and the underscore. Note that we only have to write this once, because we are within a `for` loop. Do this part yourself.

f. Underneath your `pickWord()` function call, add the necessary line to call your `writeBlanks()` function.
g. Run and check your program to make sure you write the correct number of blank spaces and that everything is properly working.

8. **Validate**
   Double check your page and make sure everything is properly running.

9. **Do Your Notes**
   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!