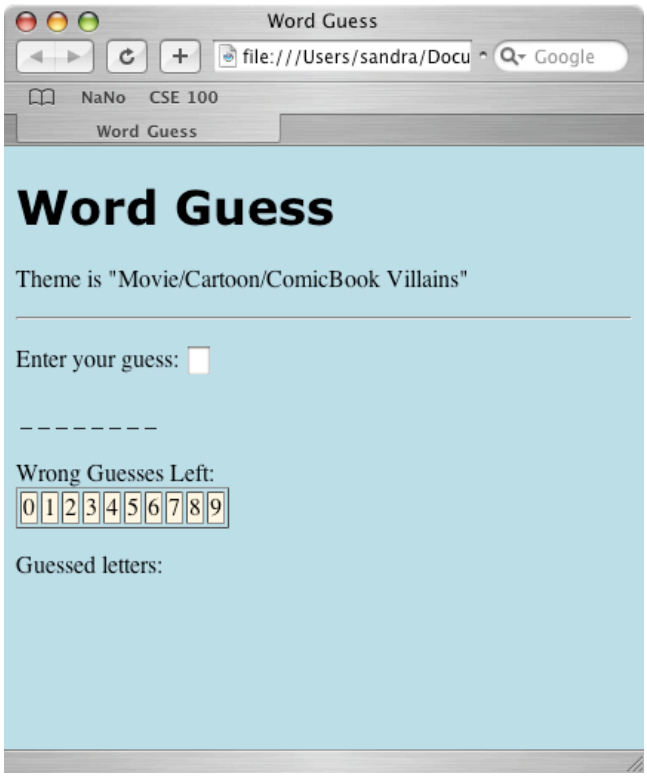
Project 2A: Javascript Game: WordGuess
CSE 100/INFO 100 Fluency with Information Technology

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.

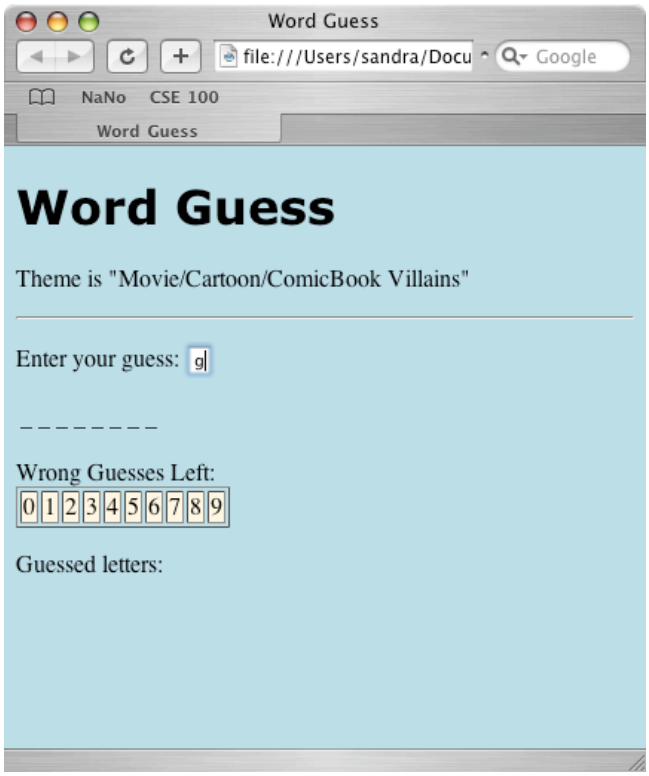
For this project, in addition to programming, you will need to comment your code, as well as include a section on your page that details your process in writing this program, e.g. the problems you encountered, the process you went through to debug them, etc.

This project is divided into a Part A and a Part B. Part A will guide you along more closely to 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.

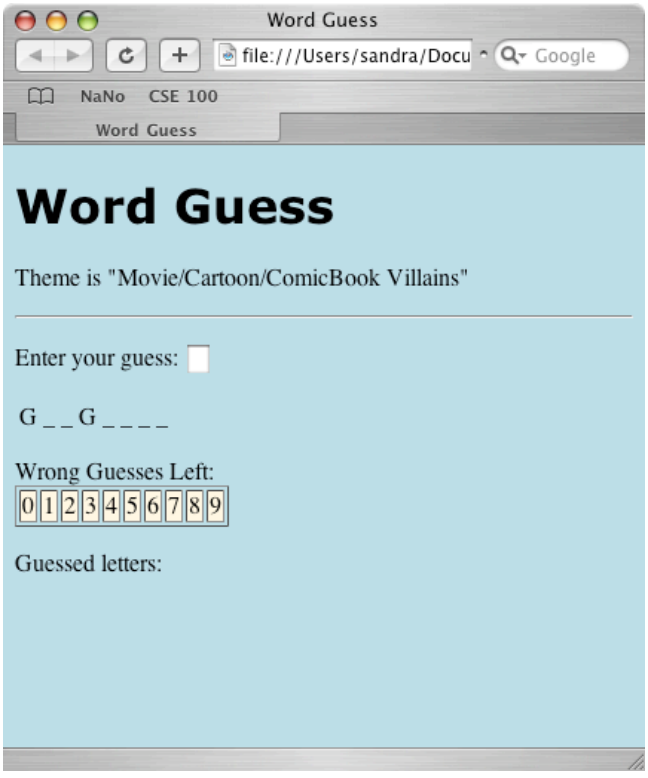


Game Overview

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 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.

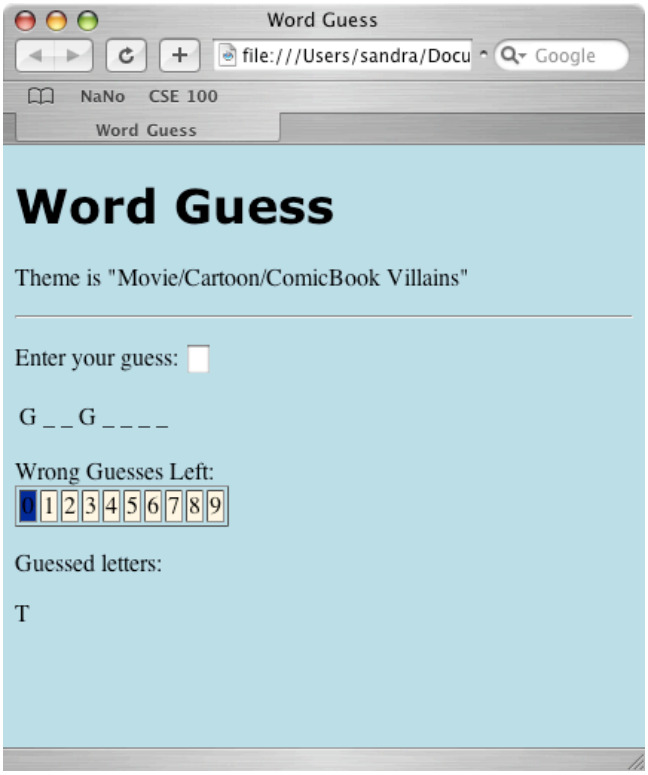


a. If the letter is in the secret word, our program will replace the blanks it created earlier with the letter, in all the places that the letter appears. In the "Gargamel" example, if players enter "g" as their guess, the computer will display "G _ _ G _ _ _ _". Notice that our game does not bother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

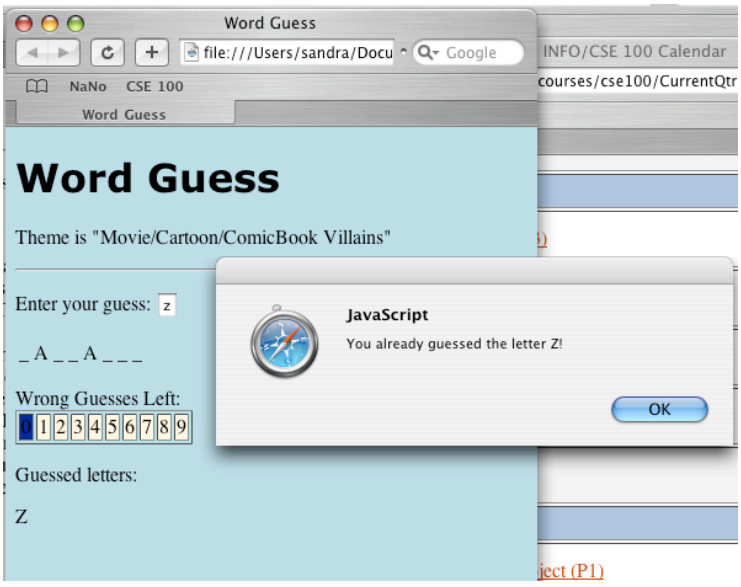


b. If the guess was wrong, the computer will write to an area of the screen that displays incorrect guesses. Additionally, the progress bar will move by one box, to show the user that they have used up one of their wrong guesses. The player gets 10 wrong guesses. (Notice that the progress bar has 10 boxes, from 0-9.) So, if the user guesses the letter "t" in the "Gargamel" example,

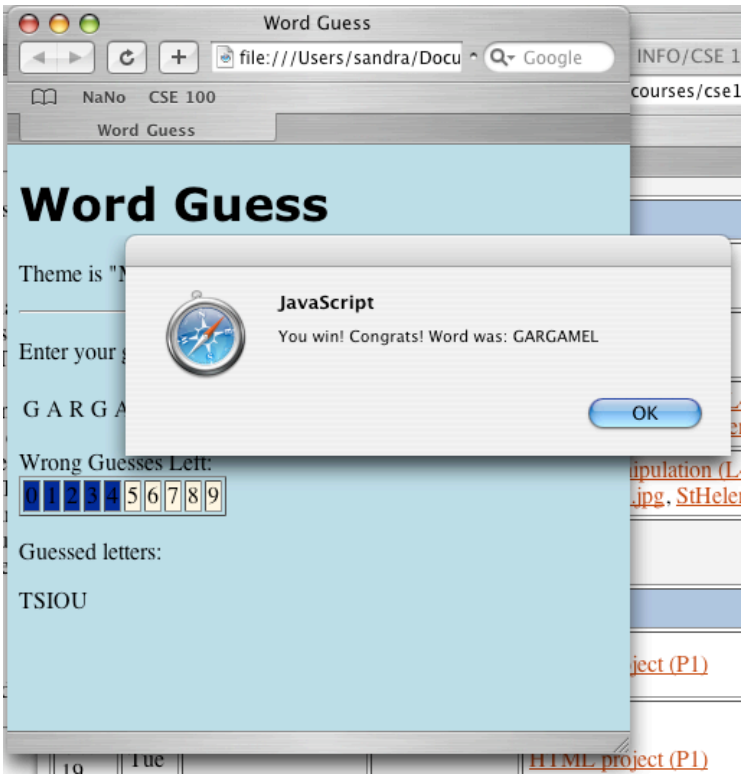
nothing will be written on the blank lines, and the letter "t" will appear in the incorrect guesses area, and one of the boxes will be shaded. The correct guesses (say we had guessed “g” first, and then “t”) stay where they are.



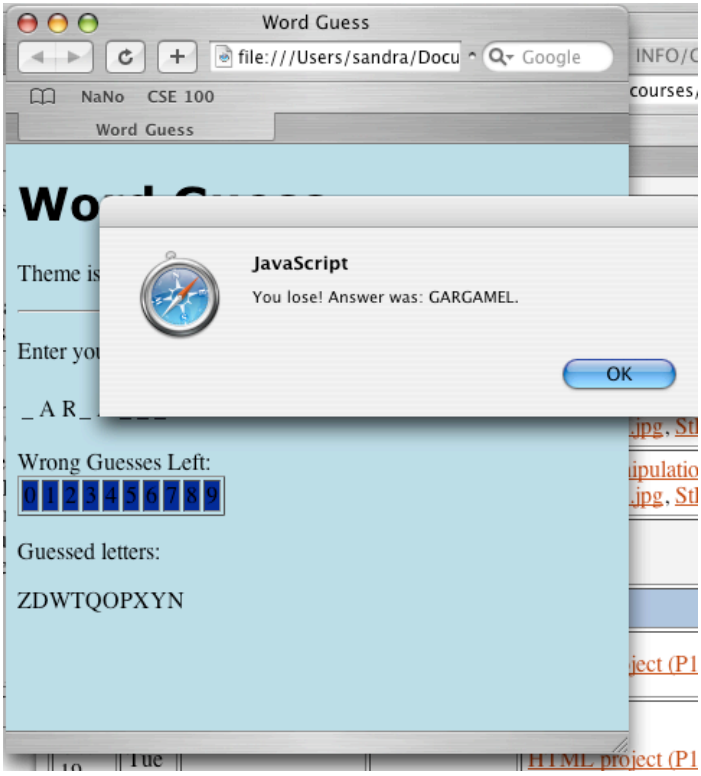
c. If you guess a letter that you’ve already guessed, nothing happens, and a pop-up window will tell you that you’ve already guessed that letter.



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 worth **30 points** and is due on Wednesday, November 10, 2004, before 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, as with Project 1.

1. Create an HTML document:
    a. Turn it in properly, on your Dante account and on Catalyst **(2 points).**
    b. 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. **(1 point)**
    c. It must have a descriptive title and a page heading. **(1 point)**
    d. 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 **(5 points)**
2. **(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 what that loop/if-else statement does.
    d. 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.
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 my rules) so that visitors to your site know how to play. **(1 point)**
4. **(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. 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. **(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 now. Be sure to end the script tag, i.e. `</script>`
    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.
    c. Reload your page and enter some input into your input box to make sure your alert box shows up properly.
6. **(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 an paragraph 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. You can do this by inserting the following:

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

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. For instance:

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

d. 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]);
```

after you have finished storing everything in your array. 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. **(2 points for pickWord() function, 5 points for writeBlanks() function)** Next, we are going to have the area where we draw the blank lines to represent the letters in our secret word in the body of our HTML document, underneath the text input box. But to do this, we first 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. Then we will write a function called writeBlanks() to write the blank lines.
   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();
}
```

      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
      iii. 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.

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

```
pickWord();
writeBlanks();
```

   d. The first line there called 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 lines 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, include a statement at the end that creates an alert box that tells the user we just ran the writeBlanks function. Check to be sure 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 were "cat" with three characters, this is what we want. Note that each cell has an ID, so that we can refer to it later, 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."

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

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 create a table, first add a line at the beginning of the body of the pickWord() function using the document.write() function that will write our table tag, i.e. add the following line to the beginning of the body your function:

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

2. 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.
3. Now, 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. 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 `<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. 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>
```

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 Nov 1 lecture, if you've forgotten how to create loops) in between the document.write lines that end and begin the table rows. For the three statements inside the parentheses of the loop, that governs how many times the loop runs (note that this is NOT the body of your for loop), 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++`.

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

4. Run and check your program to make sure you write the correct number of blank lines and that everything is working properly.

8. Validate your page and make sure everything is running properly.
9. 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!