Web Design 1 **Name: _____**

Project #9: Sporclone **Date: _____ Period: _____**

Project Description:

Create a working clone of the game Sporcle (http://www.sporcle.com/). On this project, you will demonstrate your ability to integrate JavaScript into a web page as well as your ability to modify JavaScript to suit specific needs. You will not need to write HTML, CSS, or JavaScript from scratch on this project. Templates are given.

The instructions below describe how you will need to edit the template files to make a working game. By the time you finish, you will have created three games. At least one of your games should be image based (like the logo quizzes). No game should have fewer than 10 answers. At least one of your games should have at least 20.

Your grade will be based on both the rubric and on reviews of each of your games by your classmates.

Understand that on this project you are not writing large blocks of HTML or JavaScript from scratch. Nearly every step requires you to modify an existing line of code or to add a single line of code. If you find yourself wanting to do more than that, then you are probably misunderstanding the instructions.

Before you ask for help...

1. Have you looked for a page you made previously that solves a problem similar to the one where you are stuck?
2. Have you Googled possible solutions to your problem?
3. Does the person next to you know how to solve your problem?
4. Have you actually tried by typing code and testing? It's okay to guess sometimes. You can always undo.

Also, I will not answer any questions about this project unless all prior work is completed.

Instructions:

Most of the numbered steps refer to a letter which identifies a particular location in the HTML and JavaScript files. Letters *a-n* are in game.html, and *o-z* are in sporclone.js. Follow the steps in order of numbers. The letters are only labels and do not reflect the order in which code should be completed.

Periodically you will be asked to preview and test your page. If the page does not work exactly as intended, then you should not move on from that section.

Part 1 - Project Files

1. Save the game.html, sporclone.js, and style.css files to your Projects/09 Sporcle directory. If you preview the game.html file now, you'll see that it is an unstyled and non-functioning quiz on South Carolina mountain peaks.

2. Change the name of the game.html file so that it reflects the name of your puzzle. Be sure that your file name is in all lowercase letters. Use underscores instead of spaces.

Part 2 - Quiz Appearance

3. Change the title tag (a) so that it describes your puzzle. Do not remove the "Sporclone:" from the title. Also, you can shorten your puzzle name. For example, "Name the ten highest mountain peaks in South Carolina" can be shortened to "South Carolina Peaks".

4. Add a line of code at (b) to connect the HTML file to style.css.

5. Change the h2 tag (i) to reflect the topic of your quiz.

6. Change the h3 at (k) to be an appropriate input prompt for your quiz.

7. Find an image representative of your game and save it to your images folder. Then crop the image so that it is a perfect square. The CSS is written to scale the image automatically, but the image still shouldn't be too large as it will slow down page load time. If necessary resize your image so that it is no larger than 200x200 pixels. Edit the image source (h) so that your image appears as an icon for your game.

8. Edit the footer (n) to give yourself credit for making the quiz and cite sources for your puzzle. Be sure you provide a working link to each source. Your source cannot be another Sporcle quiz. Google is not a source either.

9. Edit the table (h) to provide space for each answer. The IDs of the span tags for each answer must be answer0, answer1, answer2, etc. Choose dimensions for your table that best display your answers. If possible, prevent the need to scroll while playing. DO NOT ACTUALLY TYPE ANY ANSWERS INTO THE TABLE.

Preview your html page. At this point you see that the page has a nice layout since you have applied the CSS. Make sure the title tag is correctly displaying in the tab and that the question text indicates the topic of your quiz. The image icon should display next to the h2 that states the topic of your quiz. The dimensions of the table should provide enough spaces for all answers. Your name should be in the footer. Test that your source link works as well. We haven't modified any of the JavaScript to make the quiz functional, but the appearance of the quiz should be correct.

DO NOT MOVE ON UNLESS EVERYTHING TO THIS POINT WORKS CORRECTLY.

Part 3 - JavaScript Configuration

10. Locate the script tags containing the game configuration variables. Edit the `answers` array (c) to contain all answers to your quiz. Be sure that each value is in quotes. Double check that the length of the `answers` array corresponds to the number of answer span tags in the table you created.

11. Edit the `time_limit` variable (d) to give an appropriate amount of time (in seconds) to play the game.

12. Connect this page to the sporclone.js file at (e). It is important that the connection to the external script comes after the game configuration variables.

13. Add an onload event handler to the body tag (f) that calls the `setup()` function.

Preview the page again. Also, the time limit should appear under the Time heading. If you do not see the time limit, then you have made an error in your configuration variables or in the line that connects the HTML file to the JavaScript file and will need to double check steps 9-12

DO NOT MOVE ON UNLESS EVERYTHING TO THIS POINT WORKS CORRECTLY.

Part 4 - The Timer

14. As the game is played, a variable `time_remaining` tracks the amount of time the player has left to guess all of the answers. The `tick()` function is responsible for decreasing the time remaining and for ending the game when time runs out. Add a statement at location (u) that decreases the `time_remaining` variable by 1.

15. The `start()` function is responsible for getting the timer to start running. At location (p), set the value of the variable timer so that it uses the `setInterval()` function to call the tick function once per second.

16. Add an onclick event handler to the start button (j) that calls the `start()` function.

Preview your page now and test the start button. You should see the time count down. For testing purposes, you may want to use a small time_limit of a few seconds so that you can see what happens when the clock stops ticking. If the start button does not cause your timer to count down, then you need to double check steps 13-15.

17. A `formatTime()` function has been included in the sporclone.js file. Apply the `formatTime()` function to time_remaining at (o) so that it is initially displayed in mM:SS form.

18. Also apply `formatTime()` to `time_remaining` at (s) so that the `time_remaining` is displayed in mM:SS form while the clock is running as well.

Preview and test the timer one more time. Make sure that the time is displayed in mM:SS form when the page is initially previewed and that it continues to display correctly while the clock is ticking.

DO NOT MOVE ON UNLESS EVERYTHING TO THIS POINT WORKS CORRECTLY.

Part 5 - Game Play

19. The `check()` function is responsible for comparing user input to each element in the array of answers. One feature of the `check()` function is that it counts the number of correct answers and displays the total. The global variable `correct` holds the current number of correctly guessed answers. Write a statement at (v) that increases the variable `correct` by 1.

20. Write a statement at (w) that puts correct/total in the score span tag. The `setup()` function already does this. Look there to see how.

21. When the actual game of Sporcle is played, you may have noticed that you don't need to actually hit enter to get your answers checked. Instead, Sporcle responds to the onkeyup event. Add an onkeyup event handler to the guess field (l.) that calls the `check()` function.

22. If you look in `check()`, you'll notice that both the guess and answer are passed to the convertToAlphanumericLowerCase() function prior to being compared. The convertToAlphanumericLowerCase() function returns a string with all non-alphanumeric characters removed. This is nice because answers like "Pac-Man", "Pac Man", and "PacMan" will all be treated as equivalent strings. The game is still case-sensitive however. Add a statement at (z) which converts `str` to lowercase before `str` is returned.

Preview and test again. Start the game and begin typing in answers. Each time you enter an answer correctly, it should show up in the corresponding location in the answer table and the score should update accordingly. If you guess all of the answers, then the game should end. Also, if time runs out, the remaining answers should be filled in the table at the end of the game.

DO NOT MOVE ON UNLESS EVERYTHING TO THIS POINT WORKS CORRECTLY.

Part 6 - Additional Styling

23. Add a statement in the `start()` function at (q) that changes the color of the time text to green.

24. Add a statement in the `end()` function at (x) that changes the color of the time text to red.

25. The `end()` function fills in any missing answers once the game is complete. It should also highlight correct and incorrect answers. Add statements to the conditional at (y) that highlights each correct answer one color and each wrong answer another. (The variable id in the loop will contain the actual id for the answers location in the table. Highlight by changing the background color of the element with this id.)

At this point, your game should be fully playable. Try it out!

It is okay to move on even if you aren't able to complete steps 23-25. Just undo anything you tried so that the game back to a playable state before you do move on. If you do figure out how to get these items complete, then the extra games you make will all still work.

Part 7 - Finishing Touches

26. We can make the `start()` function automatically place the cursor in the guess field so the user doesn't have to click in the field to play. Write a statement at (r) to do this. (Google 'JavaScript focus'.)

27. Rather than just have the time green while the game is being played and red when it ends, see if you can turn the color of the time yellow when under 10 seconds remains. This should be done in the `tick()` function. You'll need to modify the conditional at (t) by adding an else if.

28. It is not required, but you may optionally write include custom CSS in the style tags located in the head of the games HTML file. This space is provided because there may be instances where additional CSS may be helpful because of varying table layout among games. (Don't modify style.css as it contains CSS which applies to all games.)

It is okay to move on even if you aren't able to complete steps 26-28. Just undo anything you tried so that the game back to a playable state before you do move on. If you do figure out how to get these items complete, then the extra games you make will all still work.

Part 8 - Additional Games

Creating extra games does not require any updates to the sporclone.js or style.css files. You'll only need to edit the game's HTML file. Make two copies of your working HTML file.

29. For each,

    a) Give the quiz an appropriate file name.
    b) Edit title and heading tags.
    c) Edit the game configuration section.
    d) Edit the table so that you have cells to correspond to each answer.
    e) Update the sources in the footer.

    Remember, at least one of your games should be image based (like the logo quizzes) and that no game should have fewer than 10 answers. At least one of your games should have at least 20 answers.

30. Edit the links in the "Additional Games" section (g) so that each game links to the two others you created.

Part 9 - Final Review

Congratulations! Now you should self-grade each of your quizzes according to the rubric provided. You should also have a couple classmates play your games and complete the game review forms. After they review your games, make any changes necessary and submit your project for grading.