Homework 2: JavaScript Wordish Game

Due date: September 12, 2022 at 11:59pm (9/7: Minor updates, marked in red.)

This homework is the second in a series of three homework assignments working a “Wordle-like”game that we call “Wordish”. In this homework, you will implement the game functions using JavaScript running client-side in the web browser. (For the third homework, you will implement the game functions by sending web requests to a Django application running server-side.)

For this assignment, you will use the front-end interface you created in the last assignment and add functionality by implementing the necessary behaviors with JavaScript.

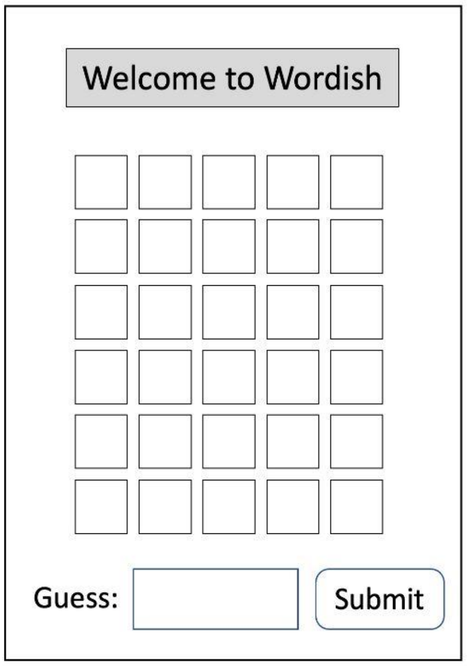
The learning goals for this assignment are to:

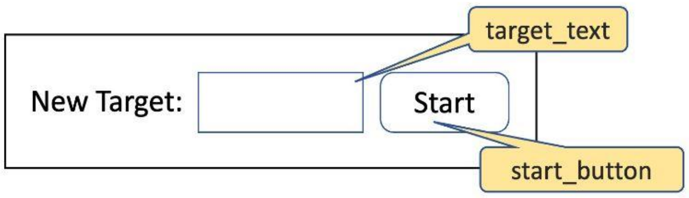
- Gain familiarity with client-side programming by using JavaScript to add functionality to a static front end through DOM manipulation and event dispatch.
- Ensure robustness of a small-scale application with some simple validation.

Specification

This section describes the behavior of the Wordish game that you will implement.

- You must modify the front end you wrote for the previous homework, but you must keep using the same id attributes on all the HTML elements.
- To initialize the game for testing, you must provide an additional label, textbox and button to start a new game using a specific target word. The additional text and button elements will have `id="target_text"` and `id="start_button"` as shown in the sketch below. These new elements can appear anywhere on the page as long as everything is still visible without scrolling.
- Game play:
  - Show a “welcome” message in `#status` when the game starts.
  - After the target word has been provided, show a “start” message in `#status`.
  - The user makes guesses using the `#guess_text` and `#guess_button` elements (from HW1)
  - If a user enters invalid input, display an “invalid input” error message in `#status`
  - When the player enters a valid word, `#status` should reflect successful input and the matrix should be updated to show the guessed letters with proper backgrounds (discussed in HW#1) using three different colors of your choice.
  - When the user wins/loses the game,notify the user with a corresponding message (containing the word “win” or “lose”) in `#status`.
- The game ends when either of the following circumstances is met:
  - The player wins when entering a word that is the same as the target word.
  - The player loses after 6 incorrect guesses.



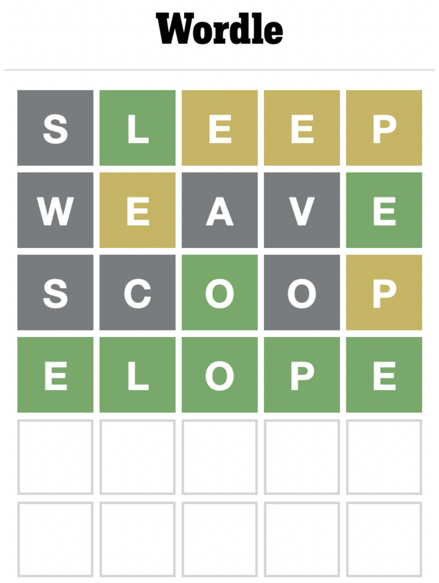


Requirements

Your submission must also follow these requirements:

- Your submission must follow all of the requirements specified in the last homework assignment. Specifically, you must include the IDs on the HTML elements, as specified in Homework #1.
- You may not use any external libraries (e.g. jQuery, Bootstrap, etc.).
- Your Wordish game must be able to validate all inputs and accept only strings of five English letters.
- Your Wordish game must have 4 colors for different statuses of the matrix cells.
- Your JavaScript may not crash at any user input—there must not be any errors reported by the browser JavaScript console.
- All your JavaScript functions must be defined in a file called `wordish.js`. Minimize the amount of JavaScript written directly in your HTML file.
- Cite all resources in a README.md file, using the same format as HW#1.

Implementation hints



One subtlety of computing the background colors for guessed words is the handling of repeated letters. (The rules are the same for Wordle and Wordish.) Consider the example provided in the HW#1 specification shown to the right. Notice:

- Target word is ELOPE.
- First guess is SLEEP. It has two E letters which are both yellow, since they are both in the wrong place
- Second guess is WEAVE. It has two E letters, one in the correct place, one in the wrong place, so a green and a yellow.
- Third guess is SCOOP. It has two O letters, one in the correct place and one in the wrong place, but since ELOPE has only one O letter, one O is gray and the other O is shown with a color (yellow or green). Priority goes to the O in the correct location, so it's green and the other is gray.

Consider creating a JS function that takes as parameters the guess and the target word. That's all you need to compute the colors. Have the function return an array of five colors (string values for the colors). To implement this function, first determine which letters are green, then the yellows, the rest are gray.

Optional input validation

The minimum requirement for input validation is to give an error if a guess or target word is not a five-letter string. However, the actual Wordle game accepts only five-letter English words. If you wish, you may have your game reject guesses and target words that are not five-letter English words by checking this list:

https://www.cmu-webapps.org/wordish/all_words.js

The AutoGrader tests will always use five-letter words from the above list (unless it's testing inputs that are not five-letter strings).

Optional random target words

You may have your Wordish game provide an option to select its own target words randomly as long as the user can still specify the target word as described above. (The AutoGrader will not test this. It always specifies a target word.)

If you wish to provide this functionality, words are usually selected from a list of common words that users are familiar with. You may use this list, if you wish:

https://www.cmu-webapps.org/wordish/target_words.js

Turning in your work

Your submission should be turned in via Git in a directory called **`hw2`** and should consist of an HTML file called `wordish.html`, a JavaScript file called `wordish.js`, and associated assets (CSS and images). Organize your files with the CSS and JavaScript files in their own folders as shown:

```
[YOUR-ANDREW-ID]/hw2/
  |-- wordish.html
  |-- css/
     |-- wordish.css
  |-- js/
     |-- wordish.js
  |-- README.md
```

Grading

You must run the grading script to get credit for this assignment. The grading script is (will be) available at https://grader.cmu-webapps.org. You may correct, resubmit, and regrade your homework as described in the syllabus.