Before I write any code I just want to do some short brainstorming for myself. I figured I should include it.

Q: What language/framework and why?

A: JS with some kind of lightweight component library. Won't need reactivity

Why?

Pros:

Browser is easy to start with a text-only interface.
Fast REPL loop, no compilation.
Can easily add visualization later if I want to.
Cross-platform, I don't have to worry about the user being unable to run it.
Can use cookies
I like JS for small, fun projects :)

Cons:

JS is untyped and fails in unpredictable ways.
Need external library for testing.
Difficult to extend and maintain without a proper framework.
Code can easily get ugly and fall into bad practices.

Initial Thoughts on the Requirements:

Each word has 5 letters
You get 6 tries per word
Incorrect letters are gray
Correct letters with incorrect locations are yellow
Correct letters in the correct location are green

Statistics:
Played
Win %
Current streak
Max streak

Initial questions about requirements:
How do we effectively represent the three shades of color in text? (wrong letter, right letter wrong spot, right letter right spot)

Fun things to add?
“Difficulty” of a word? Given the probability distribution of letters in a list of all the 5-letter words in the english language— which words have letters/a sequence of letters with the lowest cumulative probability? Finds words where guessing commonly used letters won’t help you.

Show the information gain of a guess (how good was the guess). Since there are 26 english letters, that’s log2(26) = 4.7 bits of information per character. That’s 23.5 bits of information you need to add via your guesses in order to find the word. With 6 guesses you would need to add 3.9 bits of information per guess, or reduce the pool of possibilities 16-fold with each guess.

Visualization. I have been wanting to try out Svelte. This seems like a good reason.

Setup/Dependencies:

Wordle list https://gist.github.com/cfreshman/a03ef2cba789d8cf00c08f767e0fad7b
Allowed words and answers are not the same.

Testing framework (just use console.assert for now, can use Jest later)

Svelte.js?

Initial ideas for implementation:

Guessing:

Determining if a guess is allowed:
Answers and allowed guesses will need to be loaded into a Trie. Guessing will just involve checking if the submitted word is in the Trie before going any further. If we cared about performance, we could do this asynchronously while the user is typing in their guess. But this is actually pretty fast.

Comparison/Scoring:
My best idea for scores is that a guess will require 2 passes for processing.

The first pass will determine if a letter is used or not. Unused letters will have a score of 0, used letters will have a score of 1.

The second pass will determine if a letter is in the right place compared to the original word. Letters that match will get another value added.

The final array would look like this:

[0, 1, 0, 2, 1]

Where a 0 is gray, a 1 is yellow, and a 2 is green.

A basic loop can handle running sequences of games.

Can use cookies to persist stats between sessions (dont want to bother with that though so I wont)

For processing into probabilities:

We will need to filter out all words that contain gray letters. [can be optimized]
We need to filter in all words that have green letters in the right spot.
We need to filter out all words that have yellow letters in the wrong spot.

This will give us a candidate pool of words that match for a guess.

Note on optimization: We can filter out all the gray letters by separating all the words into a hashmap. The hashmap will contain 26 entries, each key being a letter. The value of the key will be all words that contain at least one occurrence of that letter in them. There will be overlap between each list. On our first guess we can just merge the sets as our gray pass. This should yield us a much smaller set of letters to do subsequent searches on.