FLARE-ON CHALLENGE 9 SOLUTION

BY NICK HARBOUR

Challenge 1: Flaredle

Challenge Prompt

Welcome to Flare-On 9!

You probably won't win. Maybe you're like us and spent the year playing Wordle. We made our own version that is too hard to beat without cheating.

Play it live at: http://flare-on.com/flaredle/

7-zip password: flare

Solution

The Flaredle challenge package contains the following four files:

- Index.html – This is html page that is the primary interface to the challenge. Browsing to this file in a webserver will cause the additional three files to be loaded by the browser appropriately.
- Script.js – This is main script file backing the Flaredle app. It contains the game logic.
- Words.js – This contains a JavaScript array which provides the word list that the game uses.
- Style.css – This is the style sheet for rendering the Flaredle web page. It is not necessary to reverse engineer this file to solve this challenge.

To interact with this challenge dynamically you must use a web browser to browse to an instance of this app. For convenience, we have hosted this challenge live on the flare-on.com website here: http://flare-on.com/flaredle/

If you prefer to interact with it locally, you may launch a local web server easily using Python. Enter the directory of the Flaredle challenge files on the command line and using Python 3 type the following command:

```
python -m http.server
```

If you are using Python 2, you may instead use the following command from the Flaredle directory:

```
python -m SimpleHTTPServer
```

This will cause python to launch a simple web server on your host, listening on port 8000. You can then point your web browser to http://localhost:8000/ to access your local copy of the Flaredle app. If you are successful, you will see a full game board including both a keyboard and game grid as shown below in Figure 1.

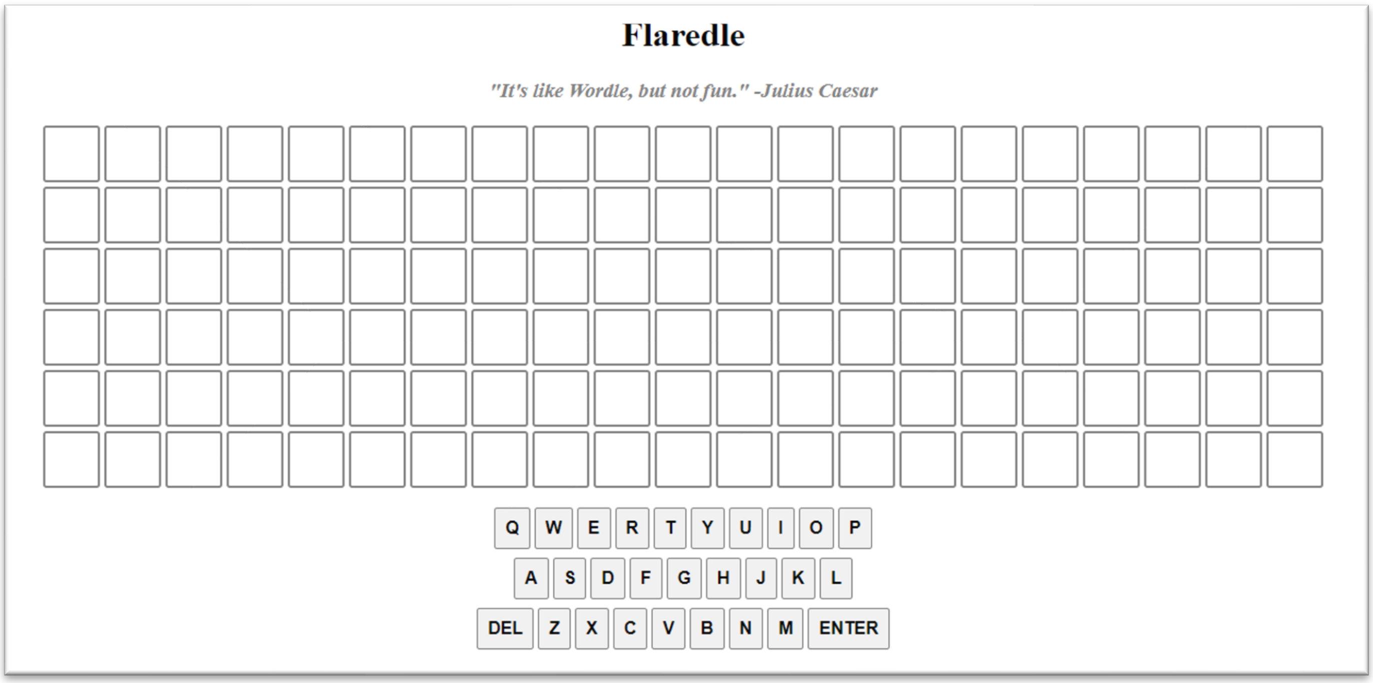


Figure 1 - Flaredle Game Screen

The index.html file contains only the code to layout the overall page, and the banner and keyboard keys. The game grid and all game logic are implemented in the file script.js. Line 65 of the index.html file, shown below, specifies that the file script.js must be loaded and executed by this page.

```
<script src="script.js" type="module"></script>
```

We will focus on the file script.js for the remainder of this document.

The Flaredle game is intended to be a variant of the popular game Wordle. In Wordle, the player must determine the 5-letter word in 6 attempts. With each attempt, the game tells the player if each letter for a given guess occurs in the correct solution, and if that letter is in the same place as it is in the correct solution. Flaredle plays in the exact same fashion, but only uses 21 letter words. The trick of it is that most people don't know many 21-letter words and playing a Wordle game based on them is futile. If you are willing to examine the game code though, there is no challenge you cannot overcome.

The first line of the file script.js is as follows:

```
import { WORDS } from "./words.js";
```

This line imports the value WORDS from the source file words.js. The contents of this source code file are the definition of an array named WORDS that is populated with 21-digit strings. The beginning 4 lines of this file are shown below:

```
export const WORDS = ['acetylphenylhydrazine',
   'aerobacteriologically',
   'alkylbenzenesulfonate',
   'aminoacetophenetidine',
```

This file contains 170 21-digit strings that it populates into the array named WORDS. Consider now the next few lines from the script.js file, lines 3 through 9:

```
const NUMBER_OF_GUESSES = 6;
const WORD_LENGTH = 21;
const CORRECT_GUESS = 57;
let guessesRemaining = NUMBER_OF_GUESSES;
let currentGuess = [];
let nextLetter = 0;
let rightGuessString = WORDS[CORRECT_GUESS];
```

The first two constants set some basic features for the game board, such as how many guesses you are allowed and how long each word is. This translates to the dimensions of the game grid (6x21 in our case). The next value is conspicuously named CORRECT_GUESS and is set to the value 57. The last line of this code fragment creates a variable named rightGuessString and sets it to the CORRECT_GUESS (57'th) record in the WORDS array.

It would be a valid approach to simply look at the words.js file at this point, find the 57th line, and attempt to input it into the game board. We will analyze the game.js file in more detail here to determine if this is a false lead.

At line 164 in script.js an event listener callback function is added for the "keyup" event. Whenever the user presses a key and releases it, this code will execute and receive information about the pressed key in the "e" parameter.

```
document.addEventListener("keyup", (e) => {

    if (guessesRemaining === 0) {
        return
    }

    let pressedKey = String(e.key)
    if (pressedKey === "Backspace" && nextLetter !== 0) {
        deleteLetter()
        return
    }

    if (pressedKey === "Enter") {
        checkGuess()
        return
    }

    let found = pressedKey.match(/[a-z]/gi)
    if (!found || found.length > 1) {
        return
    } else {
        insertLetter(pressedKey)
    }
})
```

When the user presses the "Enter" key, this function calls checkGuess(). In the Wordle and Flaredle games, you type your word and press "Enter" to register and submit you guess. This code relies on the checkGuess() function to perform all subsequent processing of the submitted guess and display its results.

The checkGuess() function begins with the following lines:

```
function checkGuess () {
    let row = document.getElementsByClassName("letter-row")[NUMBER_OF_GUESSES - 
guessesRemaining]
    let guessString = ''
    let rightGuess = Array.from(rightGuessString)
```

Here we can see that it is indeed using the rightGuessString value we previously noted to construct an array of values named rightGuess. rightGuess will be an array of individual characters derived from the string rightGuessString.

The rightGuess array is used in this function to determine the location of specific letters within the correct guess string. The critical section of this function begins at line 109:

```
    if (guessString === rightGuessString) {
      let flag = rightGuessString + '@flare-on.com';
      toastr.options.timeOut = 0;
      toastr.options.onclick = function() {alert(flag);}
        toastr.success('You guessed right! The flag is ' + flag);
```

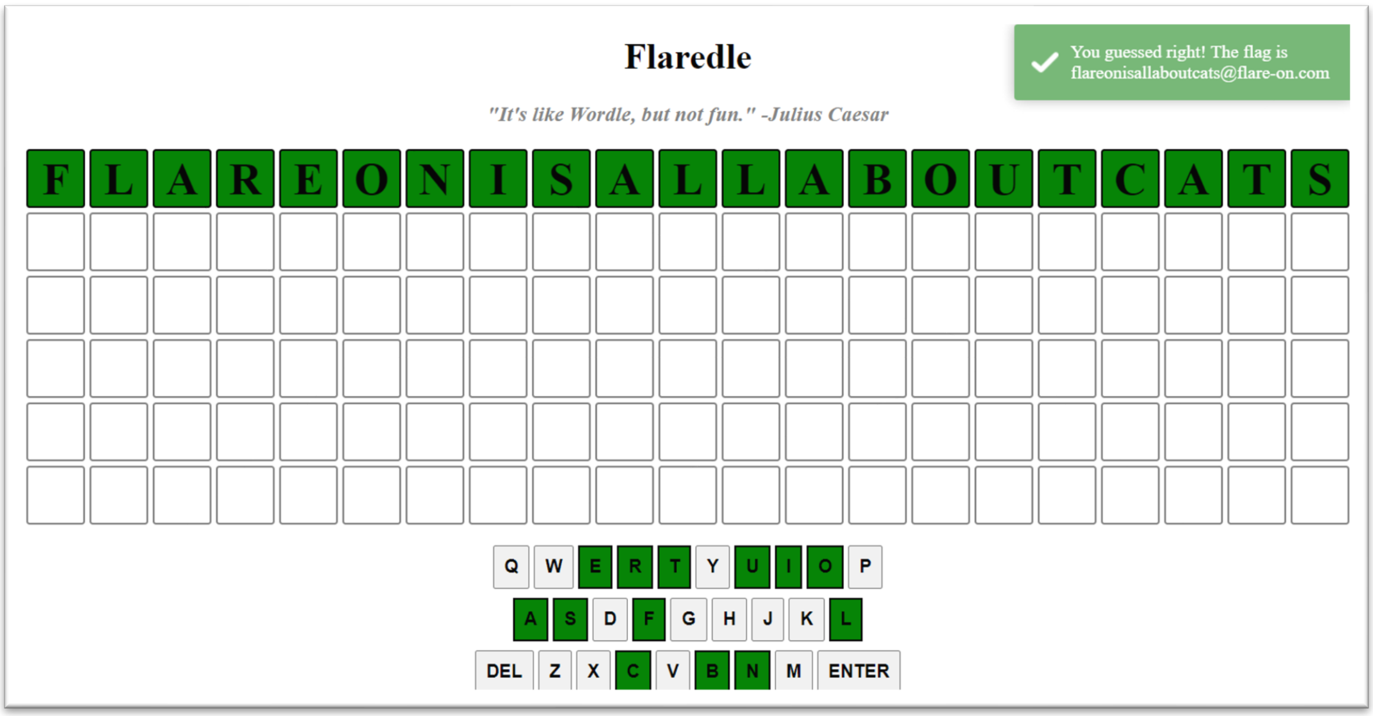
Here it is performing a simple string comparison of the string the user entered with the rightGuessString. If the strings match, then the code builds a value called "flag" which is the rightGuessString and the "@flare-on.com" strings added together, then ultimately displays "You guessed right!" to the screen along with the flag value.

Now it appears that if we know array element #57 in the WORDS array, then we know the correct answer. To find this value easily we need to keep in mind two things: each line in words.js contains one element, and the first element in the array is element 0. Therefore, to find element #57, we need to go to line 58 in the source code file because source code line numbers start from 1 instead of 0.

This is the value found on line 58 of words.js:

```
   'flareonisallaboutcats',
```

If you enter the string "flareonisallaboutcats" into the game it will present you with the victory notification including the flag value, as shown below:



Flaredle

"It's like Wordle, but not fun." -Julius Caesar

You guessed right! The flag is flareonisallaboutcats@flare-on.com

The correct flag for this challenge is:

```
flareonisallaboutcats@flare-on.com
```