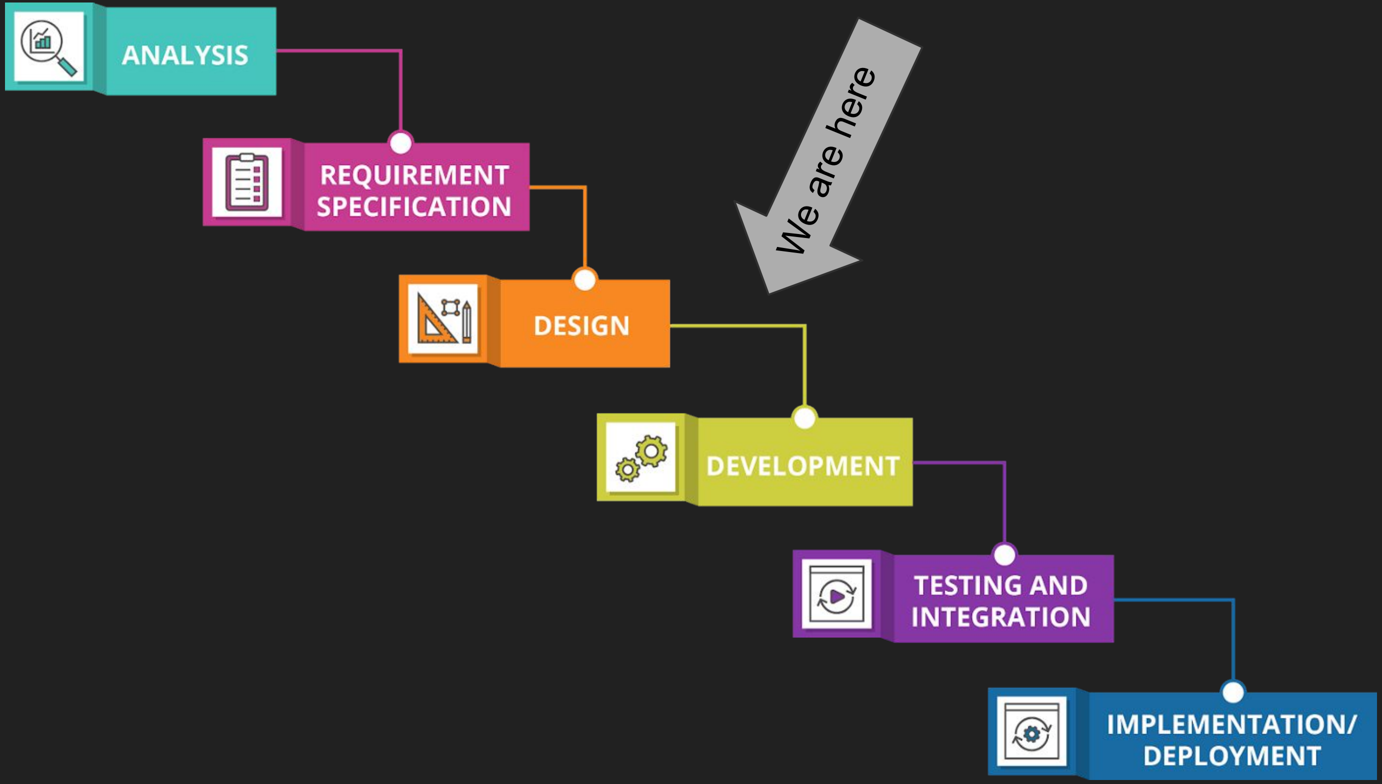
Wordle++

Jeffrey Fitzsimmons, Richard Kirk, Tyler Reinert,
Mateusz Marciniak

Requirements

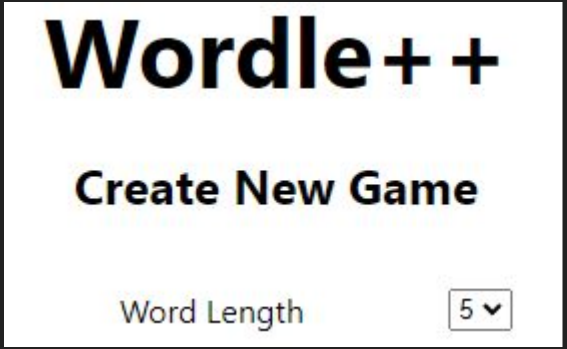
- Make Improved Wordle game
  - Add difficulty choice
  - Customizable letter of words
  - Customizable number of tries
  - Scoreboard
  - Multiplayer
  - Aesthetic elements not included in Wordle
- Resources
  - FireBase
  - JavaScript
  - Github/ git

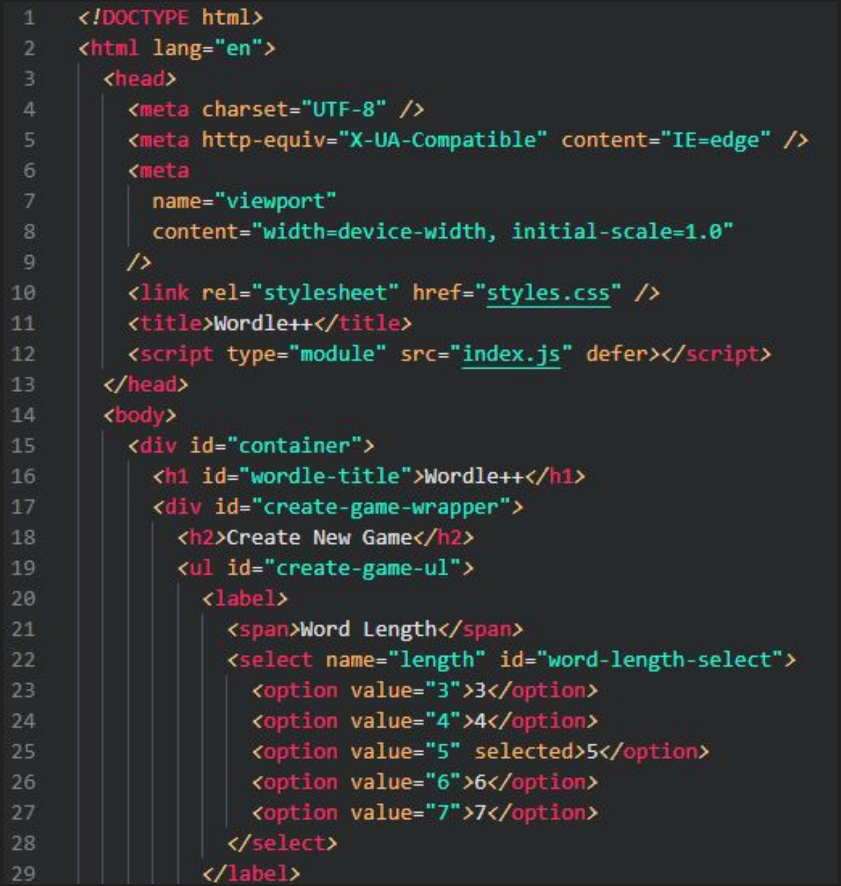
Waterfall Method:



HTML Breakdown

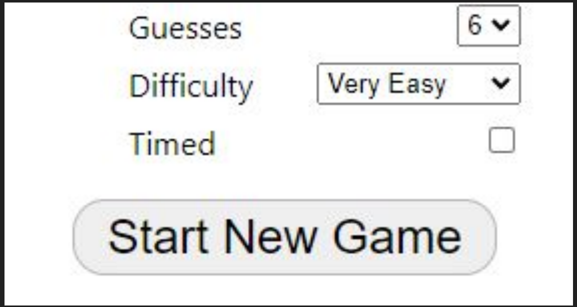
This portion of the HTML creates the title and other headers as well as the dropdown menu for choosing word length.

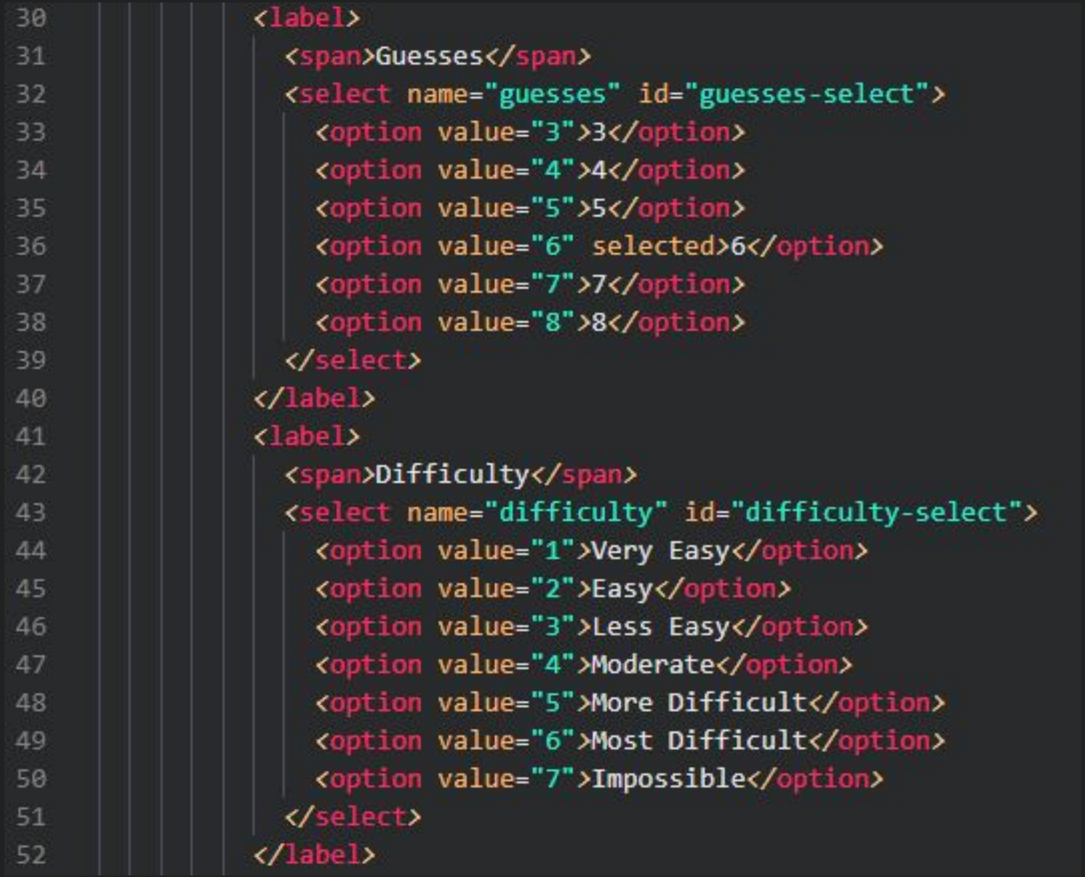



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0"
    />
    <link rel="stylesheet" href="styles.css" />
    <title>Wordle++</title>
    <script type="module" src="index.js" defer></script>
  </head>
  <body>
    <div id="container">
      <h1 id="wordle-title">Wordle++</h1>
      <div id="create-game-wrapper">
        <h2>Create New Game</h2>
        <ul id="create-game-ul">
          <label>
            <span>Word Length</span>
            <select name="length" id="word-length-select">
              <option value="3">3</option>
              <option value="4">4</option>
              <option value="5" selected>5</option>
              <option value="6">6</option>
              <option value="7">7</option>
            </select>
          </label>
```

HTML Breakdown

This portion of the HTML creates the dropdown menus for the amount of guesses you'd like as well as the difficulty of the word.

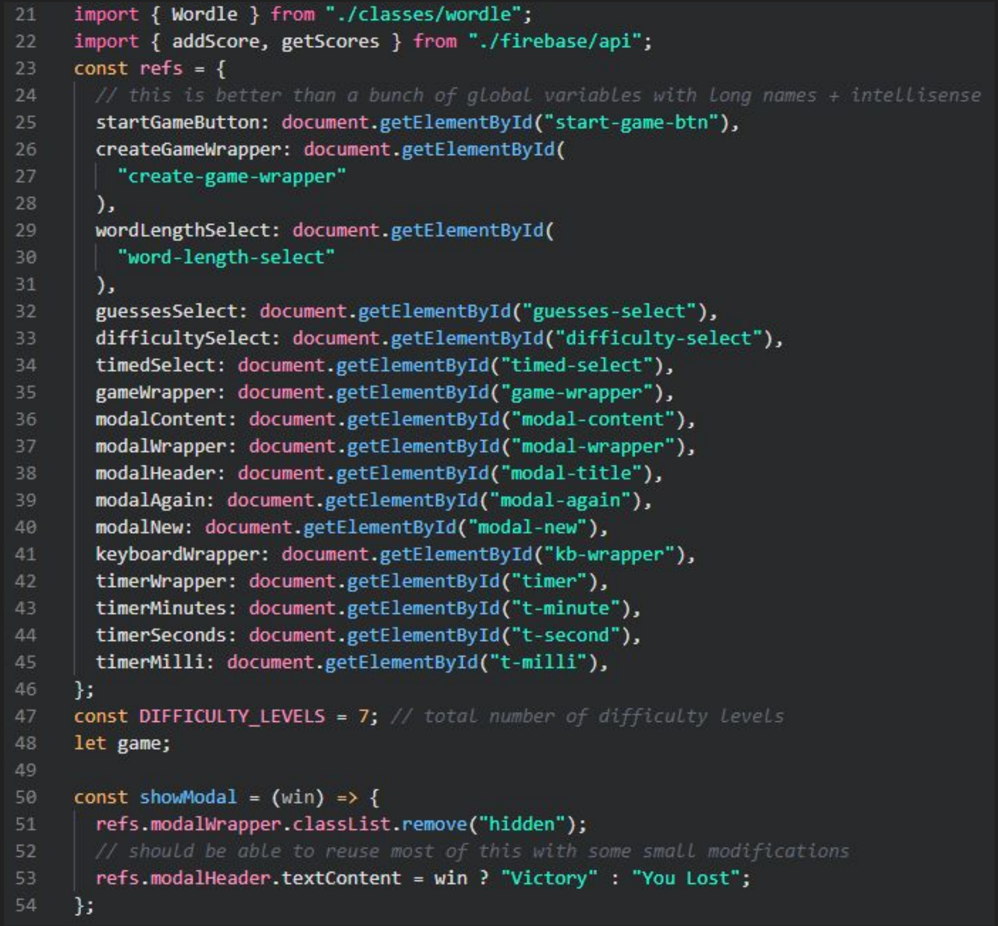

Guesses 6
Difficulty Very Easy
Timed ☐



```
<label>
  <span>Guesses</span>
  <select name="guesses" id="guesses-select">
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5</option>
    <option value="6" selected>6</option>
    <option value="7">7</option>
    <option value="8">8</option>
  </select>
</label>
<label>
  <span>Difficulty</span>
  <select name="difficulty" id="difficulty-select">
    <option value="1">Very Easy</option>
    <option value="2">Easy</option>
    <option value="3">Less Easy</option>
    <option value="4">Moderate</option>
    <option value="5">More Difficult</option>
    <option value="6">Most Difficult</option>
    <option value="7">Impossible</option>
  </select>
</label>
```

JavaScript Breakdown

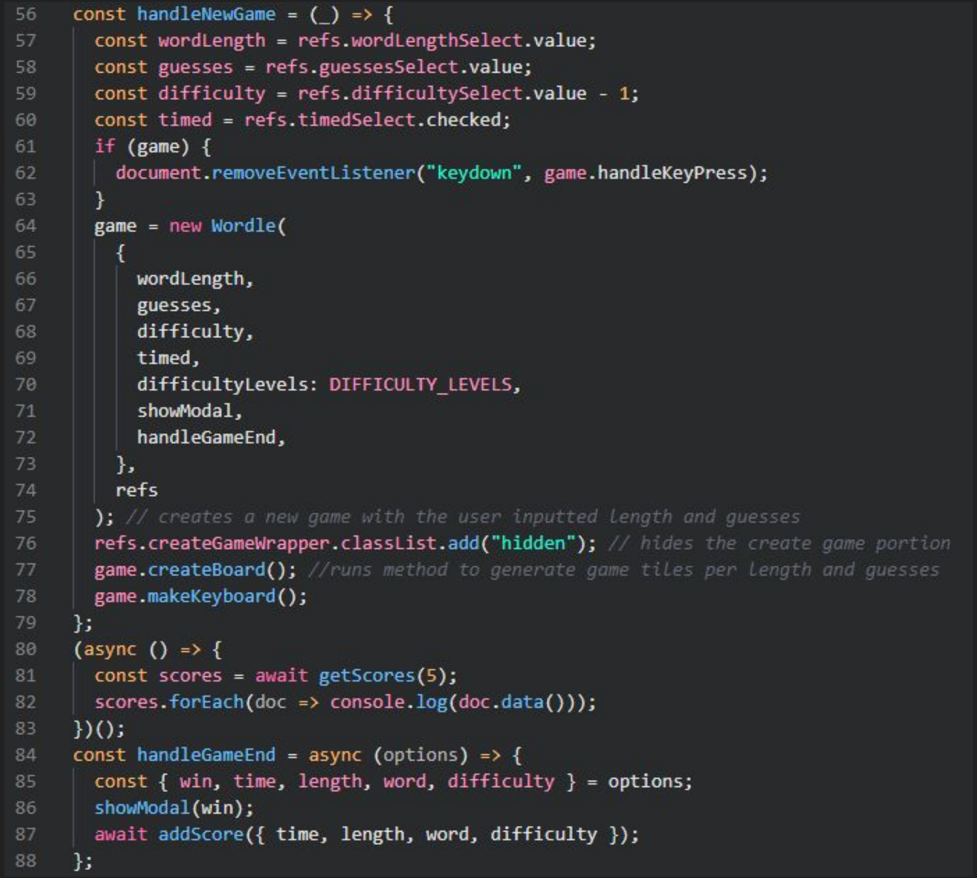
This portion of code creates the object refs, which contains a bunch of global variables that are used to determine the game settings



```javascript
import { Wordle } from "./classes/wordle";
import { addScore, getScores } from "./firebase/api";
const refs = {
  // this is better than a bunch of global variables with long names + intellisense
  startGameButton: document.getElementById("start-game-btn"),
  createGameWrapper: document.getElementById(
    "create-game-wrapper"
  ),
  wordLengthSelect: document.getElementById(
    "word-length-select"
  ),
  guessesSelect: document.getElementById("guesses-select"),
  difficultySelect: document.getElementById("difficulty-select"),
  timedSelect: document.getElementById("timed-select"),
  gameWrapper: document.getElementById("game-wrapper"),
  modalContent: document.getElementById("modal-content"),
  modalWrapper: document.getElementById("modal-wrapper"),
  modalHeader: document.getElementById("modal-title"),
  modalAgain: document.getElementById("modal-again"),
  modalNew: document.getElementById("modal-new"),
  keyboardWrapper: document.getElementById("kb-wrapper"),
  timerWrapper: document.getElementById("timer"),
  timerMinutes: document.getElementById("t-minute"),
  timerSeconds: document.getElementById("t-second"),
  timerMilli: document.getElementById("t-milli"),
};
const DIFFICULTY_LEVELS = 7; // total number of difficulty levels
let game;

const showModal = (win) => {
  refs.modalWrapper.classList.remove("hidden");
  // should be able to reuse most of this with some small modifications
  refs.modalHeader.textContent = win ? "Victory" : "You Lost";
};
```

JavaScript Breakdown

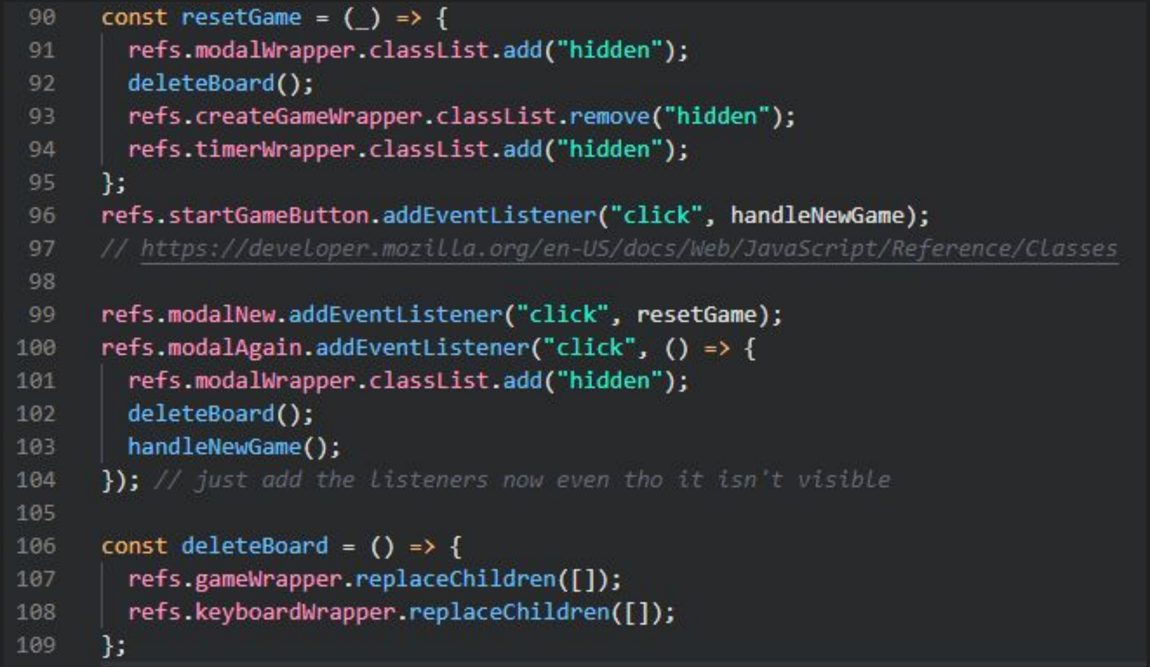
Here, the handleNewGame function is applied when the new game is started



```
const handleNewGame = (_) => {
  const wordLength = refs.wordLengthSelect.value;
  const guesses = refs.guessesSelect.value;
  const difficulty = refs.difficultySelect.value - 1;
  const timed = refs.timedSelect.checked;
  if (game) {
    document.removeEventListener("keydown", game.handleKeyPress);
  }
  game = new Wordle(
    {
      wordLength,
      guesses,
      difficulty,
      timed,
      difficultyLevels: DIFFICULTY_LEVELS,
      showModal,
      handleGameEnd,
    },
    refs
  ); // creates a new game with the user inputted length and guesses
  refs.createGameWrapper.classList.add("hidden"); // hides the create game portion
  game.createBoard(); //runs method to generate game tiles per length and guesses
  game.makeKeyboard();
};
(async () => {
  const scores = await getScores(5);
  scores.forEach(doc => console.log(doc.data()));
})();
const handleGameEnd = async (options) => {
  const { win, time, length, word, difficulty } = options;
  showModal(win);
  await addScore({ time, length, word, difficulty });
};
```

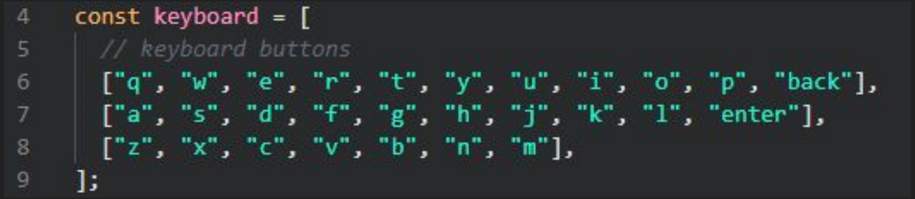
JavaScript Breakdown

This handles the game ending and the showing/hiding of the modal.

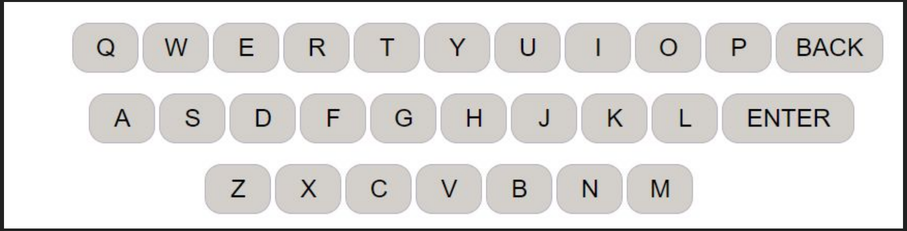


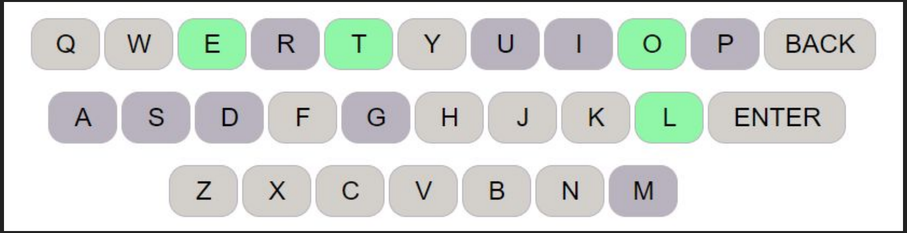
```javascript
const resetGame = (_) => {
  refs.modalWrapper.classList.add("hidden");
  deleteBoard();
  refs.createGameWrapper.classList.remove("hidden");
  refs.timerWrapper.classList.add("hidden");
};
refs.startGameButton.addEventListener("click", handleNewGame);
// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

refs.modalNew.addEventListener("click", resetGame);
refs.modalAgain.addEventListener("click", () => {
  refs.modalWrapper.classList.add("hidden");
  deleteBoard();
  handleNewGame();
}); // just add the listeners now even tho it isn't visible

const deleteBoard = () => {
  refs.gameWrapper.replaceChildren([]);
  refs.keyboardWrapper.replaceChildren([]);
};
```

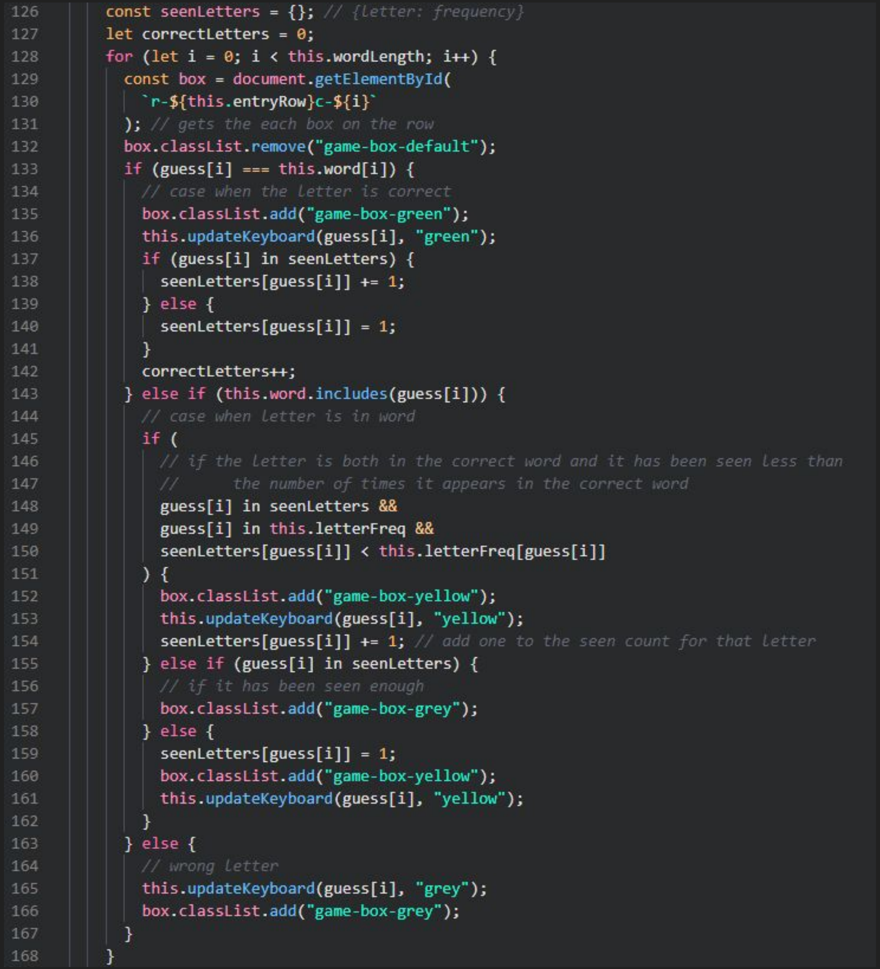
Keyboard



```
const keyboard = [
  // keyboard buttons
  ["q", "w", "e", "r", "t", "y", "u", "i", "o", "p", "back"],
  ["a", "s", "d", "f", "g", "h", "j", "k", "l", "enter"],
  ["z", "x", "c", "v", "b", "n", "m"],
];
```



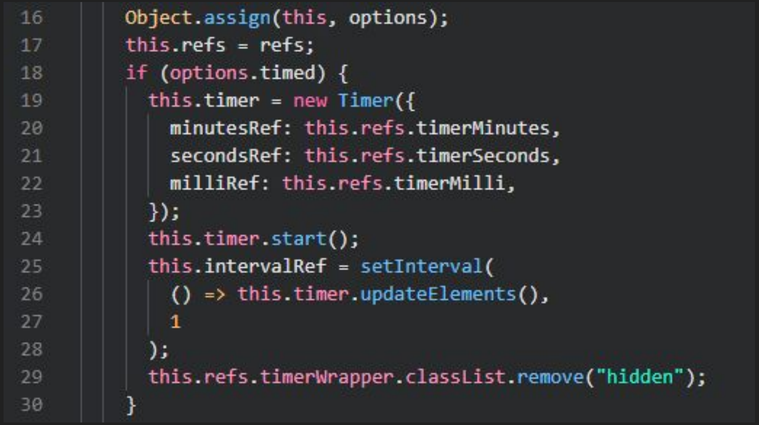




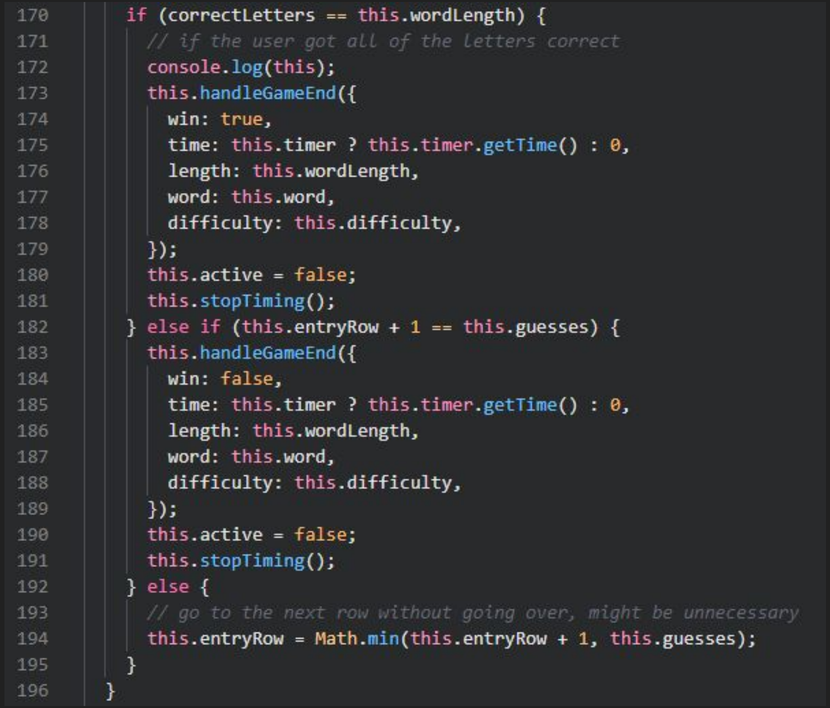
```
const seenLetters = {}; // {letter: frequency}
let correctLetters = 0;
for (let i = 0; i < this.wordLength; i++) {
  const box = document.getElementById(
    `r-${this.entryRow}c-${i}`
  ); // gets the each box on the row
  box.classList.remove("game-box-default");
  if (guess[i] === this.word[i]) {
    // case when the letter is correct
    box.classList.add("game-box-green");
    this.updateKeyboard(guess[i], "green");
    if (guess[i] in seenLetters) {
      seenLetters[guess[i]] += 1;
    } else {
      seenLetters[guess[i]] = 1;
    }
    correctLetters++;
  } else if (this.word.includes(guess[i])) {
    // case when letter is in word
    if (
      // if the letter is both in the correct word and it has been seen less than
      //       the number of times it appears in the correct word
      guess[i] in seenLetters &&
      guess[i] in this.letterFreq &&
      seenLetters[guess[i]] < this.letterFreq[guess[i]]
    ) {
      box.classList.add("game-box-yellow");
      this.updateKeyboard(guess[i], "yellow");
      seenLetters[guess[i]] += 1; // add one to the seen count for that letter
    } else if (guess[i] in seenLetters) {
      // if it has been seen enough
      box.classList.add("game-box-grey");
    } else {
      seenLetters[guess[i]] = 1;
      box.classList.add("game-box-yellow");
      this.updateKeyboard(guess[i], "yellow");
    }
  } else {
    // wrong letter
    this.updateKeyboard(guess[i], "grey");
    box.classList.add("game-box-grey");
  }
}
```

Timer

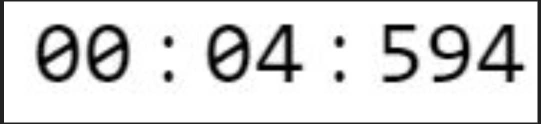
Handles the stopping and starting of the timer when a game is started/ended



```
      Object.assign(this, options);
      this.refs = refs;
      if (options.timed) {
        this.timer = new Timer({
          minutesRef: this.refs.timerMinutes,
          secondsRef: this.refs.timerSeconds,
          milliRef: this.refs.timerMilli,
        });
        this.timer.start();
        this.intervalRef = setInterval(
          () => this.timer.updateElements(),
          1
        );
        this.refs.timerWrapper.classList.remove("hidden");
      }
```

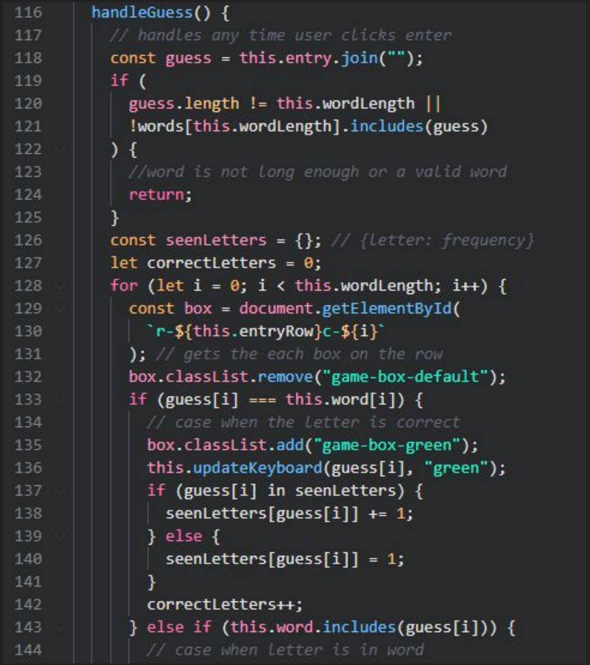


```
      if (correctLetters == this.wordLength) {
        // if the user got all of the letters correct
        console.log(this);
        this.handleGameEnd({
          win: true,
          time: this.timer ? this.timer.getTime() : 0,
          length: this.wordLength,
          word: this.word,
          difficulty: this.difficulty,
        });
        this.active = false;
        this.stopTiming();
      } else if (this.entryRow + 1 == this.guesses) {
        this.handleGameEnd({
          win: false,
          time: this.timer ? this.timer.getTime() : 0,
          length: this.wordLength,
          word: this.word,
          difficulty: this.difficulty,
        });
        this.active = false;
        this.stopTiming();
      } else {
        // go to the next row without going over, might be unnecessary
        this.entryRow = Math.min(this.entryRow + 1, this.guesses);
      }
    }
```

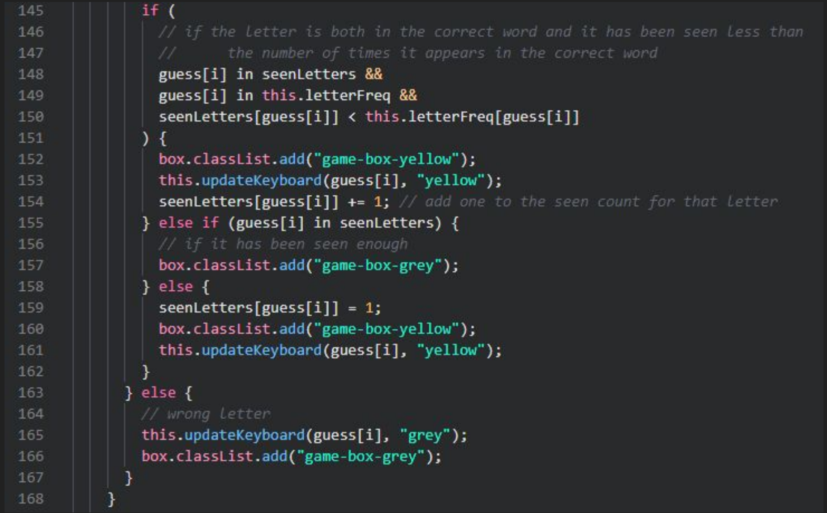


00:04:594

Guessing Words



```javascript
handleGuess() {
  // handles any time user clicks enter
  const guess = this.entry.join("");
  if (
    guess.length != this.wordLength ||
    !words[this.wordLength].includes(guess)
  ) {
    //word is not long enough or a valid word
    return;
  }
  const seenLetters = {}; // {letter: frequency}
  let correctLetters = 0;
  for (let i = 0; i < this.wordLength; i++) {
    const box = document.getElementById(
      `r-${this.entryRow}c-${i}`
    ); // gets the each box on the row
    box.classList.remove("game-box-default");
    if (guess[i] === this.word[i]) {
      // case when the letter is correct
      box.classList.add("game-box-green");
      this.updateKeyboard(guess[i], "green");
      if (guess[i] in seenLetters) {
        seenLetters[guess[i]] += 1;
      } else {
        seenLetters[guess[i]] = 1;
      }
      correctLetters++;
    } else if (this.word.includes(guess[i])) {
      // case when letter is in word
```

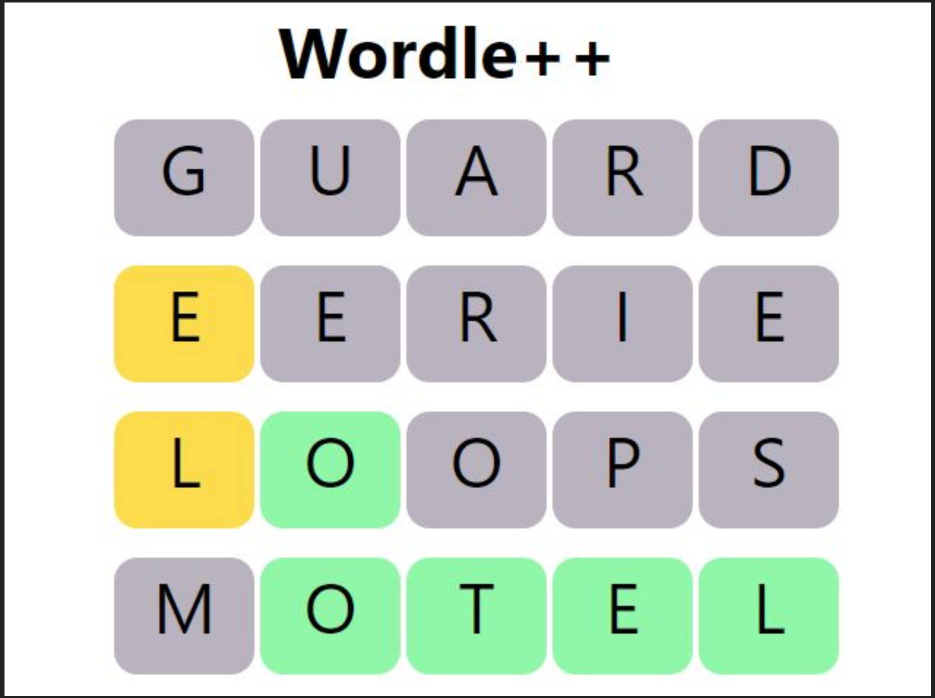


```javascript
      if (
        // if the letter is both in the correct word and it has been seen less than
        //      the number of times it appears in the correct word
        guess[i] in seenLetters &&
        guess[i] in this.letterFreq &&
        seenLetters[guess[i]] < this.letterFreq[guess[i]]
      ) {
        box.classList.add("game-box-yellow");
        this.updateKeyboard(guess[i], "yellow");
        seenLetters[guess[i]] += 1; // add one to the seen count for that letter
      } else if (guess[i] in seenLetters) {
        // if it has been seen enough
        box.classList.add("game-box-grey");
      } else {
        seenLetters[guess[i]] = 1;
        box.classList.add("game-box-yellow");
        this.updateKeyboard(guess[i], "yellow");
      }
    } else {
      // wrong letter
      this.updateKeyboard(guess[i], "grey");
      box.classList.add("game-box-grey");
    }
  }
```

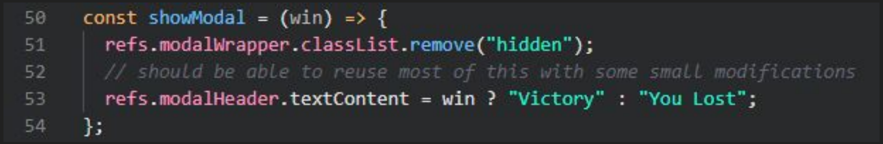
These bits of the code are working with the boxes for each letter and identifying if they are in the word or not, and if they are in the right position in the word or not.

Guessing Words

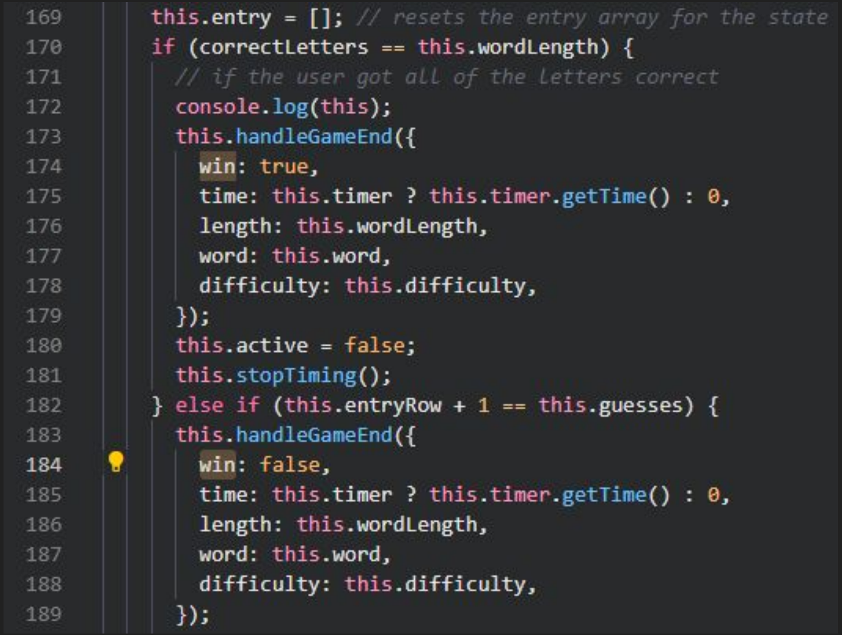
This is the result of the code in the last slide



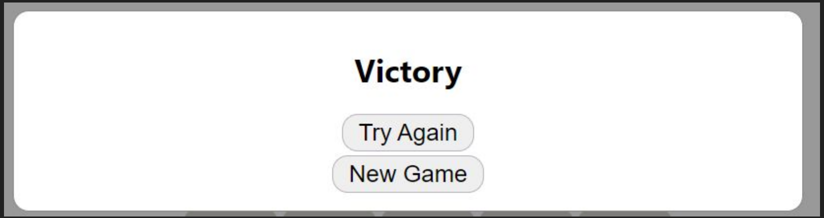
End Screen

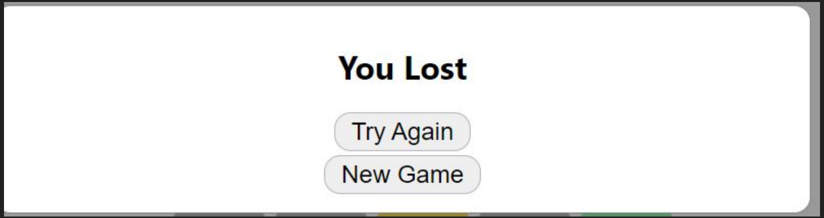


```
const showModal = (win) => {
  refs.modalWrapper.classList.remove("hidden");
  // should be able to reuse most of this with some small modifications
  refs.modalHeader.textContent = win ? "Victory" : "You Lost";
};
```



```
      this.entry = []; // resets the entry array for the state
      if (correctLetters == this.wordLength) {
        // if the user got all of the letters correct
        console.log(this);
        this.handleGameEnd({
          win: true,
          time: this.timer ? this.timer.getTime() : 0,
          length: this.wordLength,
          word: this.word,
          difficulty: this.difficulty,
        });
        this.active = false;
        this.stopTiming();
      } else if (this.entryRow + 1 == this.guesses) {
        this.handleGameEnd({
          win: false,
          time: this.timer ? this.timer.getTime() : 0,
          length: this.wordLength,
          word: this.word,
          difficulty: this.difficulty,
        });
```





FireBase

Built with Firebase

Using Firebase for a scoreboard and possibly a multiplayer mode