Hangman Game Documentation

July 26, 2025

Introduction

This document provides a comprehensive overview of the structure, styling, and logic of a Hangman game implemented using HTML, CSS, and JavaScript. The game is designed to be interactive, responsive, and user-friendly, with a clear separation of concerns between layout, styling, and functionality.

1. HTML Structure (Layout & Components)

The HTML structure is organized into semantic and modular blocks for clarity and maintainability.

Major Sections

- **.container**: Wraps the entire application, centering content and applying global styling.
- **.title**: Displays the game title, separated for styling flexibility.
- **.alert**: Contains win/lose messages and a reset button, initially hidden.
- **.gameContainer**: Houses the main gameplay area, using flexbox to separate the hangman image and guessing UI.

Inside .gameContainer

- **.pictures**: Contains the `<img>` tag for the hangman illustration, updated based on wrong guesses.
- **.guess**: Groups the following elements:
  - **.hint**: Displays the clue for the current word.
  - **.Incorrect**: Shows the number of wrong guesses (e.g., 2 / 6).
  - **.underscores**: Displays the current state of the word with underscores and correct letters.
  - **.keyboard**: Contains custom A–Z buttons, each with the `btn` class.

2. CSS Styling & Layout

The CSS employs flexbox and responsive design to ensure a consistent user experience across devices.

Key Styling Choices

- **Flexbox Layout**: The `.gameContainer` uses `display: flex` to arrange `.pictures` and `.guess` side-by-side on desktop. A media query stacks them vertically on smaller screens.

```
.gameContainer {
    display: flex;
}
```

- **.alert**: Hidden by default (`display: none`) and shown only on win/lose conditions.
- **.btn**: Letter buttons are styled for visibility and consistent spacing.
- **.underscores h2**: Enlarged text for better readability.
- **.hint-text**: Uses a distinct font and color to stand out.

3. JavaScript – Logic & Game Flow

The JavaScript handles the game's core logic, user interactions, and state management.

Global Variables

- `const GUSSES = 6`: Maximum allowed wrong guesses.
- `let wrongGuesses = 0`: Tracks incorrect guesses.
- `let randomWord`: Stores the current word object (with `word` and `hint` properties).
- `let underScores`: References the DOM element for displaying underscores.

```
const GUSSES = 6;
let wrongGuesses = 0;
let randomWord;
let underScores;
```

Step 1: startGame()

Initializes the game and resets it on restart. It:

- Selects a random word from `words.js`.

- Sets the hint.
- Displays underscores matching the word length.
- Resets the UI (wrong guesses, buttons, image, and alert).

This ensures a clean slate for each game.

Step 2: Event Listeners on Letter Buttons

Each button listens for a click event, processes the selected letter, and disables itself to prevent double clicks.

```
buttons.forEach(function(button) {
    button.addEventListener("click", function(e) {
        let letterPressed = e.target.innerText;
        e.target.setAttribute('disabled', true);
    });
```

- **Correct Guess**: Updates underscores via `underscoresAndLetters()`. If all letters are revealed, displays "YOU WON".
- **Wrong Guess**: Increments `wrongGuesses`, updates the hangman image, and checks for loss (`wrongGuesses == GUSSES`).

Step 3: underscoresAndLetters()

Manages the display of guessed letters:

- Converts the underscore string to an array.
- Updates correct letter positions.
- Joins the array back into a string.

```
function underscoresAndLetters(letter, word, displayedWord) {
    // Implementation details
}
```

Step 4: disableAllButtons()

Disables all letter buttons after a win or loss to prevent further interaction.

Step 5: Restart Button

Triggers `startGame()` to reset the game.

```
document.querySelector('.icon').addEventListener('click', function()
    {
    startGame();
});
```

Game Flow Summary

1. Start game: Display hint and underscores.
2. Player clicks letters:
    - Correct: Fill in letters.
    - Wrong: Update image and counter.
3. Win: All letters filled.
4. Lose: 6 wrong guesses.
5. Show result and restart option.

4. JavaScript Methods & Concepts

This section explains key JavaScript methods and concepts used in the game.

1. import

Imports the words array from `words.js`.

```
import words from './words.js';
```

`export default` in `words.js` allows flexible naming.

2. Math.random() and Math.floor()

Generates a random index for word selection.

```
let randomIndex = Math.floor(Math.random() * len);
```

`Math.random()` returns a number between 0 and <1, scaled by `len` and rounded down.

3. querySelector() and querySelectorAll()

Selects DOM elements.

```
document.querySelector('.hint-text');
document.querySelectorAll('.btn');
```

`querySelector()` returns the first match, while `querySelectorAll()` returns a NodeList.

4. innerText and innerHTML

Manipulates element content.

```
hint.innerText = randomWord.hint;
underScores.innerText = s;
```

`innerText` handles plain text; `innerHTML` includes HTML tags.

5. for Loop

Iterates over a word to create underscores or check matches.

```
for (let i = 0; i < word.length; i++) {
    // Logic
}
```

6. String Methods

Manipulates strings for display.

```
displayedWord.trim().split(" ").join(" ");
```

- `trim()`: Removes leading/trailing whitespace.
- `split(" ")`: Splits a string into an array.
- `join(" ")`: Joins an array into a string.

7. addEventListener()

Attaches events to elements.

```
button.addEventListener("click", function(e) {
    // Handle click
});
```

`e.target` identifies the clicked element.

8. setAttribute() and removeAttribute()

Modifies HTML attributes.

```
e.target.setAttribute('disabled', true);
button.removeAttribute('disabled');
```

9. includes()

Checks if a letter exists in the word.

```
if (randomWord.word.includes(letterPressed)) {
    // Logic
}
```

10. forEach()

Loops over arrays or NodeLists.

```
buttons.forEach(function(button) {
    button.addEventListener("click", ...);
});
```

11. Template Literals

Embeds variables in strings.

```
image.setAttribute('src', `./images/hangman-${wrongGuesses}.svg`);
```

12. classList.add() and classList.remove()

Toggles CSS classes.

```
document.querySelector('.alert').classList.remove('hidden');
document.querySelector('.alert').classList.add('hidden');
```

13. Function Declaration

Groups reusable code.

```
function startGame() {
    // Logic
}
```

14. Variable Declarations

Uses `let` and `const` for scoping.

```
let wrongGuesses = 0;
const GUSSES = 6;
```

`let` allows reassignment; `const` prevents it, though object/array contents can change.