Hangman Game

> Federico R. Barca
> Javascript
> Master in Computer Science & Business Technology - IE University

Link to the game

Hangman Game

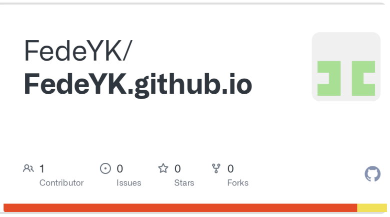
http://federicobarca.com.ar/javascript/hangman.html

Link to the GitHub Repository

FedeYK.github.io/javascript at master · FedeYK/FedeYK.github.io

Contribute to FedeYK/FedeYK.github.io development by creating an account on GitHub.

https://github.com/FedeYK/FedeYK.github.io/tree/master/javascript



FedeYK/
FedeYK.github.io

1 Contributor · 0 Issues · 0 Stars · 0 Forks

Hangman Game Description

This is an enhanced JavaScript Hangman game that challenges players to guess a word of varying length based on the selected difficulty level. The player selects individual letters from an on-screen keyboard to guess the word. The player has 7 lives, which are depleted for each incorrect guess. If the player correctly guesses the word before running out of lives, they win the game. If not, the game reveals the correct word, and the player loses.

It is a dynamic web application that interacts with the user by updating the content and visual elements in real-time without requiring a page reload. The dynamic behavior of the game is achieved through the connection between the HTML template and the `script.js` file.

HTML template

The template uses Bootstrap, a popular CSS framework, for styling and responsive design. The structure of the HTML file is divided into several parts that make up the visual elements of the game.

The connection between the HTML template and the `script.js` file is established by including the `<script src="script.js"></script>` element at the end of the `<body>` section in the HTML file.

This placement ensures that the HTML elements are fully loaded and accessible to the JavaScript code when it runs. The `script.js` file interacts with the HTML elements using their unique IDs, such as `hangman`, `word`, `keyboard`, `reset`, `difficulty`, and `message`. These IDs are used to reference the corresponding HTML elements, allowing the JavaScript code to update their properties, styles, and content dynamically.

Function & script.js file Descriptions

💡 All the code is commented and its functionallity explained on the script.js file

The `script.js` file contains the JavaScript code responsible for handling the game's logic, event listeners, and interactions with the HTML elements.

The following functions control the game’s logic.

- `getLengthByDifficulty(difficulty)`: Determines the length of the word based on the selected difficulty level (easy, medium, hard, extreme, or impossible).
- `randomWord()`: Fetches a random word of the specified length from the Random Word API and converts it to uppercase.
- `displayKeyboard()`: Creates the on-screen keyboard with buttons for each letter of the alphabet.
- `displayMaskedWord()`: Masks the selected word by replacing all its letters with underscores.
- `handleKeyPress(e)`: Handles the click event for each keyboard button, updating the displayed word and lives counter based on the player's guess.
- `disableKeyboard()`: Disables all keyboard buttons when the game is over.
- `resetGame()`: Resets the game by fetching a new word based on the selected difficulty level, re-enabling keyboard buttons, and resetting the lives counter.
- `updateHangmanImage()`: Updates the hangman image based on the number of wrong guesses.

Event listeners in the `script.js` file detect user actions, such as button clicks and difficulty selection changes. These actions trigger appropriate functions that update the game state, modify the HTML elements, and provide real-time feedback to the user, creating a dynamic and engaging gaming experience.

Ideas for Hangman 2.0

1. **Dynamic scoring system**: Create a scoring system that factors in the difficulty level, the number of correct guesses, remaining lives, and the time taken to solve the puzzle. This would encourage players to improve their skills and compete against themselves or others.
2. **Time-limited gameplay**: Introduce a countdown timer that adds an element of urgency and excitement to the game. Players must solve the puzzle before the timer runs out, or they lose. The timer duration could be adjusted based on the difficulty level for a balanced experience.
3. **Interactive animations**: Incorporate visually appealing animations for hangman image updates, button clicks, and word reveals, enhancing the overall gaming experience and making it more engaging.
4. **Custom word input**: Allow players to input their custom words or phrases, offering a personalized challenge to friends or family members. This adds variety and can be used for educational purposes, such as learning new vocabulary or practicing spelling.
5. **Global leaderboard**: Develop a leaderboard that saves and displays high scores from players worldwide, fostering a competitive environment and encouraging players to strive for better scores. The leaderboard could be filtered by difficulty level or time frame (daily, weekly, or all-time).
6. **Multiplayer mode**: Allow players to compete against each other in real-time or asynchronously, either locally or online. Players could take turns guessing or race to guess the word first.
7. **Achievements and rewards**: Introduce a system of achievements and rewards to motivate players to continue playing and improve their skills. For example, players could earn badges for winning a certain number of games or for guessing words within a certain time limit.
8. **Hints and lifelines**: Offer players the option to receive hints or use lifelines during gameplay. This could include revealing a letter in the word, removing incorrect letters from the keyboard, or even getting a definition or synonym of the word as a clue.
9. **Progressive difficulty**: Increase the difficulty of the game as players progress, such as by increasing the length of the words, reducing the number of lives, or

shortening the timer.

10. **Sound effects and background music**: Add sound effects for button clicks, correct/incorrect guesses, and game wins/losses. Background music can also enhance the overall gaming experience.
11. **Cross-platform compatibility**: Make the game compatible with various devices, such as smartphones, tablets, and computers, so that players can enjoy the game on the go or at home.
12. **Customizable user interface**: Allow players to customize the look and feel of the game by choosing different color schemes, fonts, and button styles.
13. **Language options**: Support multiple languages to attract a global audience and to provide an opportunity for players to practice their language skills.
14. **Daily challenges**: Offer daily challenges or puzzles that players can complete to earn extra