Hangman Game JS Code Explanation

The provided code is a JavaScript implementation of the Hangman game. Let's go through the code and explain its functionality:

## Word Selection:

The word array contains pairs of words and their corresponding descriptions. Each pair represents a word that can be chosen for the game and its associated clue or description.

## Game Keyboard:

The keyboardAlphabets string represents all the alphabets in uppercase. These alphabets will be used to create the game keyboard.

## Game Memory:

The variables select, wordLeft, and fail are used to keep track of the game state.

select stores the index of the selected word from the word array.

wordLeft is an array that stores the remaining letters to be guessed in the selected word.

fail keeps track of the number of incorrect guesses made by the player.

## Web-page onload:

The code inside the window.addEventListener("load", () => {...}) function is executed when the web page finishes loading.

It sets up an event listener on the moveKeyboard element to handle touch movement and adjust the position of the keyboard accordingly.

The createKeyboard() function is called to generate the initial game keyboard.

## Start Game:

The startGame() function is called when the player clicks the "Play" button.

It hides the home screen (#home) and result screen (#result) by adding the "h" class to their respective elements.

Then, it calls the newGame() function to start a new game.

## New Game:

The newGame() function is responsible for resetting the game state and starting a new game.

It calls the clearKeyboard() and clearPlayer() functions to reset the keyboard and player visuals.

Finally, it calls the createWord() function to select a new word for the game.

## Clear Keyboard:

The clearKeyboard() function resets the game keyboard by removing the "data" attribute from all the keyboard buttons.

## Clear Player:

The clearPlayer() function resets the player visuals by setting the appropriate "data" attributes of various elements to their initial values.

It also clears the fail variable and the wordLeft array.

## Get New Word:

The createWord() function selects a new word from the word array and sets up the placeholder elements to display the word's letters.

It clears the letterContainer element and creates a new placeholder element for each letter in the selected word.

The placeholder elements have IDs in the format l${i} and CSS classes (l or ls) for styling purposes.

It also populates the wordLeft array with the unique letters from the selected word.

## Create Keyboard:

The createKeyboard() function generates the game keyboard by dynamically creating buttons for each alphabet letter.

It clears the keyboard element and creates a button element for each letter in the keyboardAlphabets string.

Each button has a class of "b" and an event listener to handle the button click.

## Game Check (Button Click):

The bTas(button) function is called when a keyboard button is clicked.

It checks if the button has already been clicked (button.getAttribute("data") === "").

If the button is not clicked, it retrieves the letter from the button, checks if it exists in the selected word using the isExist(letter) function, and updates the game state accordingly.

If the selected word is fully guessed, it calls the gameEnd(true) function to end the game with a win.

Otherwise, if the guessed letter is incorrect, it calls the showNextFail() function to show the next part of the hangman.

## If Letter "X" Exists:

The isExist(letter) function checks if the provided letter exists in the wordLeft array (remaining letters to be guessed).

If the letter exists, it removes it from the wordLeft array, updates the corresponding placeholder element with the guessed letter, and returns true.

If the letter doesn't exist, it returns false.

## Show Next Fail Drawing:

The showNextFail() function increments the fail variable by one and updates the hangman visuals accordingly.

Depending on the value of fail, it sets the "data" attribute of different elements to control their visibility.

If the fail reaches 10, it means the player lost the game, and the gameEnd(false) function is called to end the game with a loss.

## Type Word:

The typeWord(letter) function updates the corresponding placeholder elements with the guessed letter.

It iterates over the selected word and replaces the placeholder elements with the letter if they match the guessed letter.

## Game Result:

The gameEnd(isWin) function displays the game result on the result screen (#result).

It sets the "data" attribute of the result screen to indicate whether the player won (isWin is true) or lost (isWin is false).

Depending on the result, it updates the title and message elements with appropriate text.

## Show Hint:

The hint() function is called when the player clicks the hint button.

It displays the hint text associated with the selected word on the hint screen (#hint).

## Exit Hint:

The hintExit() function is called when the player clicks the exit button on the hint screen.

It hides the hint screen by setting its display to "none".

## Get HTML Element by ID:

The gId(id) function is a helper function to retrieve an HTML element by its ID using document.getElementById(id).

It simplifies the code by providing a shorter alias for the getElementById function.

Overall, this code handles the logic and interactivity of the Hangman game, including word selection, keyboard creation, game state management, and result display.