personal-website/programming-exam/exam02/exam02.js

```
/*
    Name: Kevin Blinn
    Class: CPSC 3750
    Date Due: July 20th, 2024
    Assignment: Programming Exam #2
*/

// Wait for the DOM content to be loaded before executing the script
document.addEventListener('DOMContentLoaded', function() {

    // Fetch the vowel counts data from the PHP script
    fetch('exam02.php')
        .then(response => response.json())
        .then(data => {

            // Get references to the DOM elements
            const buttonsContainer = document.getElementById('buttonsContainer');
            const wordsContainer = document.getElementById('wordsContainer');
            const wordCountList = document.getElementById('wordCountList');
            const wordCountDropArea = document.getElementById('wordCount');
            const dropArea = document.getElementById('dropArea');
            let currentWords = [];

            // Loop through each vowel count in the data
            for (let count in data) {

                // Create a button for each vowel count and add the text
                const button = document.createElement('button');
                button.textContent = count;

                // Add an event listener to the button
                button.addEventListener('click', () => {

                    // Show words with the selected vowel count
                    showWords(data[count], count);

                });

                // Append the button to the container
                buttonsContainer.appendChild(button);
            }

            // Function to display words with the number of vowels the user selected
            function showWords(words, vowelCount) {

                // Clear the previous words in the container
                wordsContainer.innerHTML = '';

                // Update the current words and sort the words by length
                currentWords = words;
                words.sort((a, b) => a.length - b.length);

                // Loop through each word and create a word element
                words.forEach(word => {
                    const wordElement = document.createElement('div');

                    // Add the properties to the word element
                    wordElement.className = 'word';  // Add the 'word' class
                    wordElement.setAttribute('draggable', true);  // Make the ele↩
ment draggable
                    addDnDHandlers(wordElement);  // Add drag and drop event han↩
dlers

                    // Set the text content to the word
                    wordElement.textContent = word;

                    // Append the word element to the container
                    wordsContainer.appendChild(wordElement);
                });

                // Display the word count of x vowels to the user
                wordCountList.textContent = `Words that contain ${vowelCount} vow↩
els: ${words.length}`;
            }

            /* Using the drag and drop code found at https://codepen.io/retrofutur↩
istic/pen/DJWYBv */

            // Drag source element
            var dragSrcEl = null;

            // Handle the drag start event
            function handleDragStart(e) {
                // Target (this) element is the source node.
                dragSrcEl = this;
                e.dataTransfer.effectAllowed = 'move';  // Allow moving the element
                e.dataTransfer.setData('text/plain', this.textContent);  // Set the
```

```
dragged data
                this.classList.add('dragElem');  // Add the 'dragElem' class
            }

            // Handle the drag over event
            function handleDragOver(e) {
                if (e.preventDefault) {
                    e.preventDefault();  // Necessary. Allows us to drop.
                }
                this.classList.add('over');  // Add the 'over' class
                e.dataTransfer.dropEffect = 'move';  // Specify the drop effect
                return false;
            }

            // Handle the drag enter event
            function handleDragEnter(e) {
                // No action needed here, but can add visual feedback if desired
            }

            // Handle the drag leave event
            function handleDragLeave(e) {
                this.classList.remove('over');  // Remove the 'over' class
            }

            // Handle the drop event
            function handleDrop(e) {
                if (e.stopPropagation) {
                    e.stopPropagation();  // Stops some browsers from redirecting.
                }

                // Don't do anything if dropping the same element we're dragging.
                if (dragSrcEl != this) {
                    // Get the dropped word
                    const word = e.dataTransfer.getData('text/plain');
                    const dropElement = document.createElement('div');
                    dropElement.textContent = word;  // Set the text content to the
word
                    dropElement.className = 'word';  // Add the 'word' class
                    dropArea.appendChild(dropElement);  // Append the word element
to the drop area
                    updateDropAreaWordCount();  // Update the word count in the drop
area
                }
                this.classList.remove('over');  // Remove the 'over' class
                return false;
            }

            // Handle the drag end event
            function handleDragEnd(e) {
                // this/e.target is the source node.
                this.classList.remove('dragElem');  // Remove the 'dragElem' class
                this.classList.remove('over');  // Remove the 'over' class
            }

            // Add drag and drop event handlers to the element
            function addDnDHandlers(elem) {
                elem.addEventListener('dragstart', handleDragStart, false);  // Handle drag start
                elem.addEventListener('dragenter', handleDragEnter, false);  // Handle drag enter
                elem.addEventListener('dragover', handleDragOver, false);  // Handle drag over
                elem.addEventListener('dragleave', handleDragLeave, false);  // Handle drag leave
                elem.addEventListener('drop', handleDrop, false);  // Handle drop
                elem.addEventListener('dragend', handleDragEnd, false);  // Handle drag end
            }

            // Allow drop event handler to prevent default behavior
            function allowDrop(event) {
                event.preventDefault();
            }

            /* ------- END OF REFERENCE CODE ------- */


            // Update the word count in the drop area
            function updateDropAreaWordCount() {

                // Get the word count and update the text content
                const count = dropArea.getElementsByClassName('word').length;
                wordCountDropArea.textContent = `Words in Drop Area: ${count}`;

            }

            // Add event listeners for the drop area
            dropArea.addEventListener('drop', handleDrop);
            dropArea.addEventListener('dragover', allowDrop);
        });
```

```
});

```