# HTML-CSS-JAVASCRIPT

# Javascript Implementation in the Web-App

## `index.jsp`

### Theme Toggle Functionality

```
<script>
    // Function to toggle between light and dark themes
    function toggleTheme() {
        const body = document.body;
        const themeToggleBtn = document.getElementById('theme-toggle-btn');

        // Toggle the 'light' class in the body to switch themes
        body.classList.toggle('light');

        // Check if the current theme is light or dark and update the button
text accordingly
        const isLightTheme = body.classList.contains('light');
        themeToggleBtn.innerHTML = isLightTheme ? '<i class="moon-icon"
color="#333">Dark</i>' : '<i class="sun-icon">Light</i>';
    }
</script>
```

### Theme Toggle Button Event Listener

```
<button class="moon-sun" id="theme-toggle-btn" onclick="toggleTheme()">
    <i class="sun-icon">Light</i>
</button>
```

The theme toggle button includes an `onclick` attribute, calling the `toggleTheme` function when clicked. This enables users to switch between light and dark themes interactively.

### Script Initialization

```
<script>
    // Event listeners and functions are defined here for script initialization
    // ...
</script>
```

The script tag at the end of the `index.jsp` file initializes the functions defined earlier. It ensures that all the functions are available and ready for use when the document is loaded.

## `hangulLearning.js`

### Word Generation Function

```
// Function to generate a random Hangul word based on user settings
function generateWord() {
    const length = document.getElementById("length_slider").value;
    const difficulty =
document.querySelector('input[name="difficulty"]:checked').value;

    // Display difficulty and length
    document.getElementById("radio_d").innerText = "Difficulty: " + difficulty +
" | Length: " + length;

    // Make a request to the servlet with the selected length and difficulty
    const servletUrl = "HangulServlet?length_slider=" + length + "&difficulty="
+ difficulty;

    // Example using fetch API
    fetch(servletUrl)
        .then(response => response.json())
        .then(data => {
            // Display the Hangul word
            document.getElementById("hangulWord").innerText = "Hangul Word: " +
data.koreanWord;
            document.getElementById("wordDisplay").style.display = "block";

            // Store the correct romanized word in a data attribute
            document.getElementById("wordDisplay").setAttribute("data-correct-
romanized", data.romanizedWord);
        })
        .catch(error => console.error('Error:', error));
}
```

This function is responsible for generating a random Hangul word based on user settings. It uses the Fetch API to make a request to the server, retrieves the JSON response containing the Korean and Romanized words, and updates the HTML content to display the Hangul word.

### Word Verification Function

```
// Function to check user input against the correct Romanized word
function checkWord() {
    const userInput = document.getElementById("romanizedInput").value;
```

```
    const correctRomanizedWord =
document.getElementById("wordDisplay").getAttribute("data-correct-romanized");

    // Compare user input with the correct Romanized word
    // For simplicity, you can do a case-insensitive comparison
    if (userInput.toLowerCase() === correctRomanizedWord.toLowerCase()) {
        alert("Correct!");
    } else {
        alert("Incorrect. Try again.");
    }
}
```

The `checkWord` function compares the user's input with the correct Romanized word. It provides user feedback through an alert, indicating whether the input is correct or incorrect.

## Fetch API Usage

```
fetch(servletUrl)
    .then(response => response.json())
    .then(data => {
        // Update HTML content with the generated Hangul word
        document.getElementById("hangulWord").innerText = "Hangul Word: " +
data.koreanWord;
        document.getElementById("wordDisplay").style.display = "block";

        // Store the correct Romanized word in a data attribute
        document.getElementById("wordDisplay").setAttribute("data-correct-
romanized", data.romanizedWord);
    })
    .catch(error => console.error('Error:', error));
```

This code block demonstrates the usage of the Fetch API to make an asynchronous request to the server (`servletUrl`). It handles the JSON response, updating the HTML content with the generated Hangul word and storing the correct Romanized word in a data attribute.

## DOM Manipulation

```
document.getElementById("hangulWord").innerText = "Hangul Word: " +
data.koreanWord;
document.getElementById("wordDisplay").style.display = "block";
document.getElementById("wordDisplay").setAttribute("data-correct-romanized",
data.romanizedWord);
```

These lines of code manipulate the Document Object Model (DOM) to update the content and style of specific HTML elements. They set the Hangul word text, display the word container,

and store the

## Event Handling

```
<button onclick="generateWord()">Generate Word</button>
```

The "Generate Word" button includes an `onclick` attribute, triggering the `generateWord` function when the button is clicked. This allows users to request a new Hangul word interactively.

These explanations provide an in-depth understanding of the JavaScript code in both `index.jsp` and `hangulLearning.js`. Use these insights to further customize or enhance the functionality of your Hangul Learning App.