**Q3: Undo-Redo Functionality for a Code Editor**: Create an **undo-redo feature** using two stacks to track changes made in a code editor. As the user performs actions (e.g., writing, deleting text), track each action and allow them to undo or redo changes.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Undo-Redo Code Editor</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #f5f5f5;

}

.editor-container {

background: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

text-align: center;

width: 500px;

}

textarea {

width: 100%;

height: 200px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

font-family: monospace;

font-size: 14px;

margin-bottom: 10px;

}

button {

padding: 10px 20px;

margin: 5px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:disabled {

background-color: #ccc;

cursor: not-allowed;

}

button:hover:not(:disabled) {

background-color: #0056b3;

}

</style>

</head>

<body>

<div class="editor-container">

<h1>Code Editor</h1>

<textarea id="editor" placeholder="Start typing here..."></textarea>

<div>

<button id="undo" disabled>Undo</button>

<button id="redo" disabled>Redo</button>

</div>

</div>

<script>

const editor = document.getElementById('editor');

const undoButton = document.getElementById('undo');

const redoButton = document.getElementById('redo');

const undoStack = [];

const redoStack = [];

// Save the current state of the editor

function saveState(stack) {

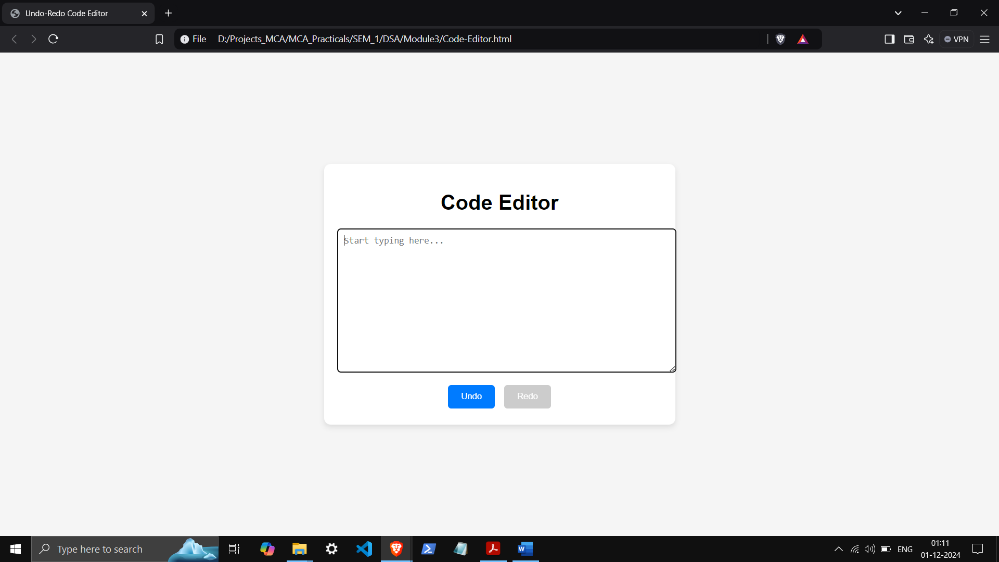
stack.push(editor.value);

}

// Undo the last action

function undo() {

if (undoStack.length > 0) {

 redoStack.push(editor.value); // Save current state to redo stack

const previousState = undoStack.pop();

editor.value = previousState;

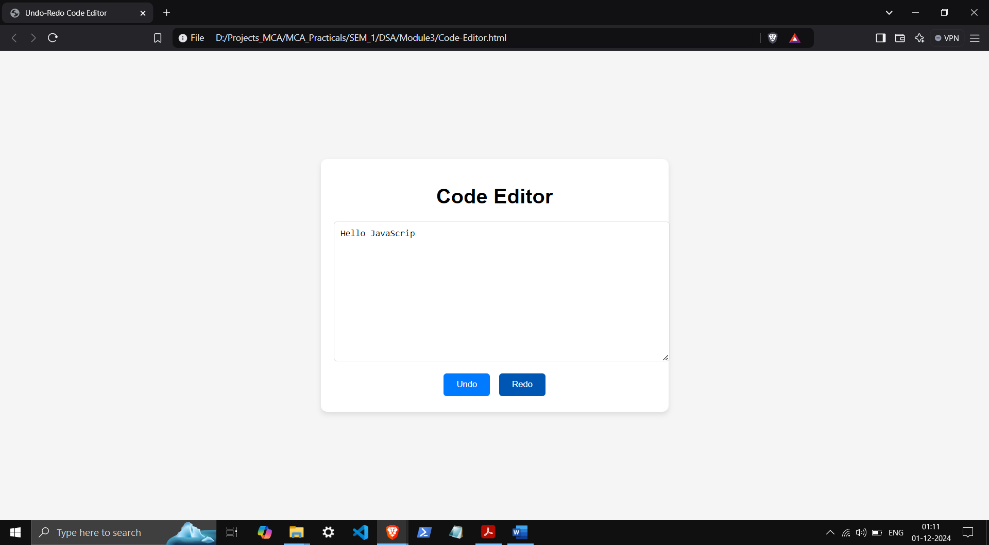
}

updateButtons();

}

// Redo the last undone action

function redo() {

 if (redoStack.length > 0) {

undoStack.push(editor.value); // Save current state to undo stack

const nextState = redoStack.pop();

editor.value = nextState;

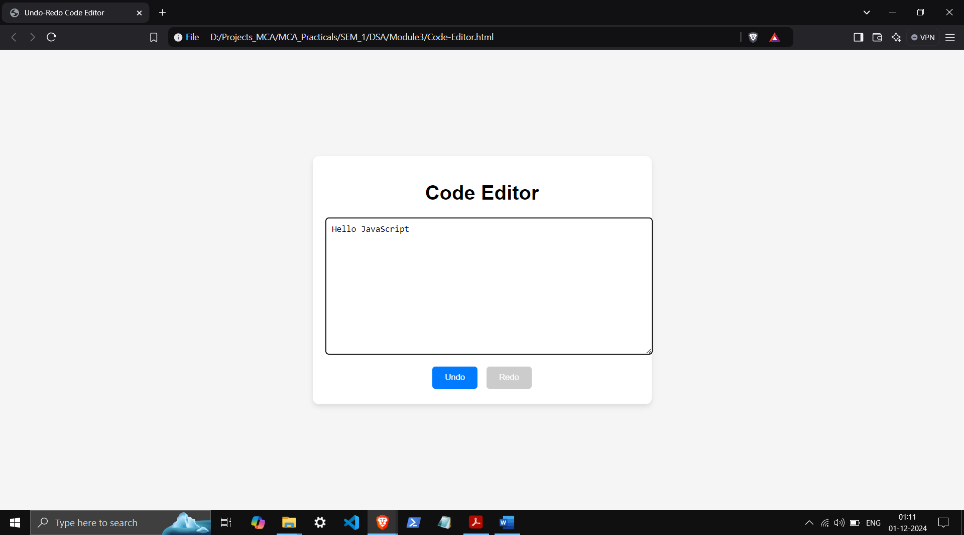
}

updateButtons();

}

// Update the state of undo and redo buttons

function updateButtons() {

 undoButton.disabled = undoStack.length === 0;

redoButton.disabled = redoStack.length === 0;

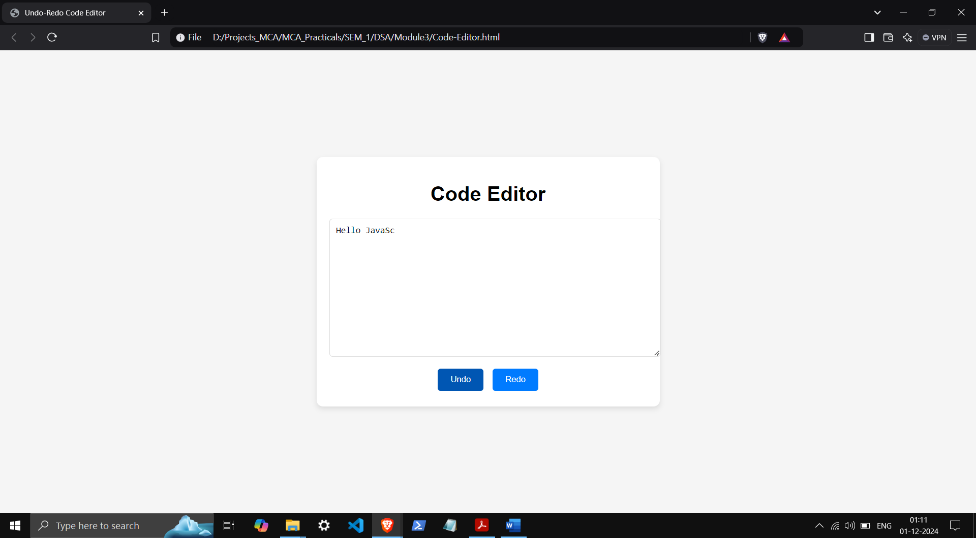
}

// Track changes in the editor

editor.addEventListener('input', () => {

saveState(undoStack); // Save current state to undo stack

redoStack.length = 0; // Clear redo stack on new input

 updateButtons();

});

// Attach event listeners to buttons

undoButton.addEventListener('click', undo);

redoButton.addEventListener('click', redo);

// Save the initial state

saveState(undoStack);

updateButtons();

</script>

</body>

</html>