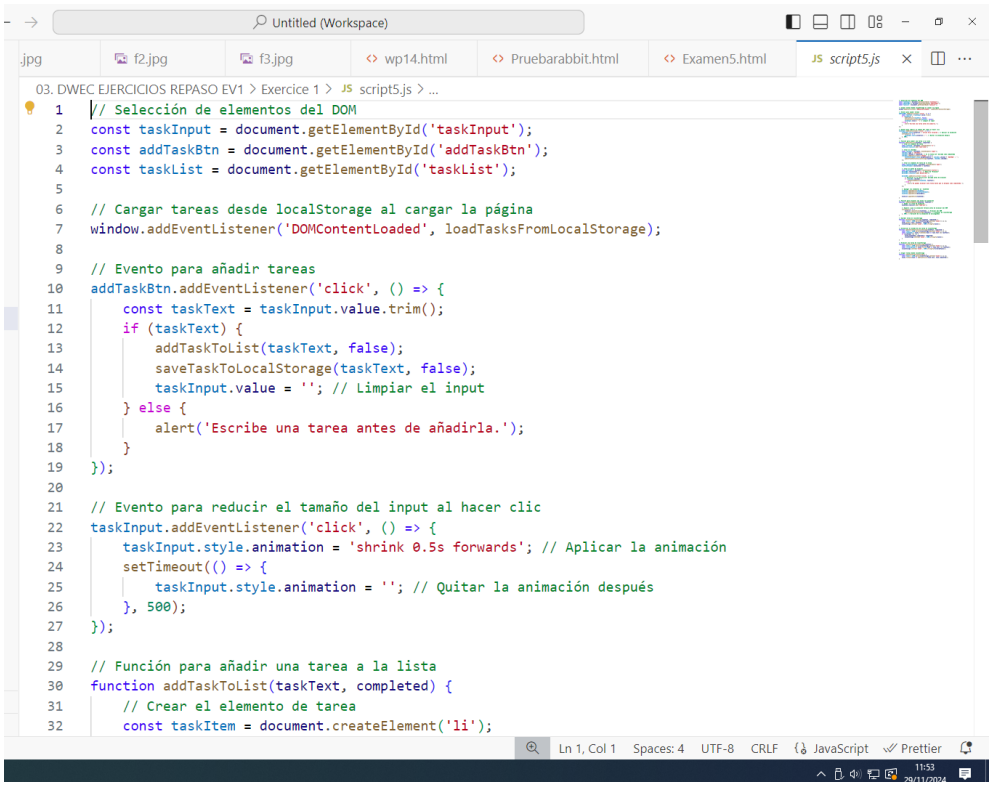
Ejercicio 1:




```
// Función para añadir una tarea a la lista
function addTaskToList(taskText, completed) {
    // Crear el elemento de tarea
    const taskItem = document.createElement('li');
    taskItem.classList.add('task-item');

    // Crear el checkbox
    const checkbox = document.createElement('input');
    checkbox.type = 'checkbox';
    checkbox.checked = completed; // Si la tarea está marcada como completada
    checkbox.addEventListener('change', () => {
        taskTextElement.style.backgroundColor = checkbox.checked ? '#d3f9d8' : '';
        updateTaskStatusInLocalStorage(taskText, checkbox.checked);
    });
    // Crear el elemento de texto de la tarea
    const taskTextElement = document.createElement('span');
    taskTextElement.textContent = taskText;

    // Crear el botón de eliminar
    const deleteBtn = document.createElement('button');
    deleteBtn.textContent = '❌'; // Emoji de "Eliminar"
    deleteBtn.classList.add('delete-btn');

    deleteBtn.addEventListener('click', () => {
        // Verificar si la casilla está marcada antes de eliminar
        if (checkbox.checked) {
            fadeOutAndRemove(taskItem, taskText);
        } else {
            alert('No puedes eliminar esta tarea hasta que la marques como completada.');
        }
    });
```

```
function addTaskToList(taskText, completed) {
    deleteBtn.addEventListener('click', () => {
        // Verificar si la casilla está marcada antes de eliminar
        if (checkbox.checked) {
            fadeOutAndRemove(taskItem, taskText);
        } else {
            alert('No puedes eliminar esta tarea hasta que la marques como completada.');
        }
    });

    // Agregar los elementos al `taskItem`
    taskItem.appendChild(checkbox);
    taskItem.appendChild(taskTextElement);
    taskItem.appendChild(deleteBtn);

    taskList.appendChild(taskItem);
}

// Función para eliminar una tarea con animación
function fadeOutAndRemove(taskItem, taskText) {
    // Añadir la clase de animación
    taskItem.classList.add('fade-out');

    // Esperar a que la animación termine antes de eliminar del DOM
    setTimeout(() => {
        taskList.removeChild(taskItem); // Eliminar del DOM
        removeTaskFromLocalStorage(taskText); // Eliminar de localStorage
    }, 500); // Duración de la animación en milisegundos
}

// Guardar tarea en localStorage
function saveTaskToLocalStorage(taskText, completed) {
```

```javascript
// Guardar tarea en localStorage
function saveTaskToLocalStorage(taskText, completed) {
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    tasks.push({ text: taskText, completed });
    localStorage.setItem('tasks', JSON.stringify(tasks));
}

// Actualizar el estado de una tarea en localStorage
function updateTaskStatusInLocalStorage(taskText, completed) {
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    const taskIndex = tasks.findIndex(task => task.text === taskText);
    if (taskIndex !== -1) {
        tasks[taskIndex].completed = completed;
        localStorage.setItem('tasks', JSON.stringify(tasks));
    }
}

// Eliminar una tarea de localStorage
function removeTaskFromLocalStorage(taskText) {
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    const filteredTasks = tasks.filter(task => task.text !== taskText);
    localStorage.setItem('tasks', JSON.stringify(filteredTasks));
}

// Cargar tareas desde localStorage
function loadTasksFromLocalStorage() {
    const tasks = JSON.parse(localStorage.getItem('tasks')) || [];
    tasks.forEach(task => addTaskToList(task.text, task.completed));
}
```

Ejercicio 2

```
// Array de imágenes para las tarjetas (pares de imágenes)
const images = [
    'f1.jpg',
    'f2.jpg',
    'f3.jpg',
    'f4.jpg',
    'f1.jpg',
    'f2.jpg',
    'f3.jpg',
    'f4.jpg'
];

// Barajar las tarjetas
function shuffle(array) {
    for (let i = array.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [array[i], array[j]] = [array[j], array[i]];
    }
}
shuffle(images);

// Crear el tablero
const board = document.getElementById('board');
let firstCard = null; // Primera tarjeta seleccionada
let secondCard = null; // Segunda tarjeta seleccionada
let lockBoard = false; // Evitar múltiples clics durante la comparación

images.forEach((image, index) => {
    const card = document.createElement('div');
    card.classList.add('card');
    card.dataset.image = image;
```

```
images.forEach((image, index) => {
    const card = document.createElement('div');
    card.classList.add('card');
    card.dataset.image = image;

    const img = document.createElement('img');
    img.src = image;

    card.appendChild(img);
    board.appendChild(card);

    card.addEventListener('click', () => {
        if (lockBoard || card.classList.contains('revealed') || card.classList.contains('matched')) {
            return false;
        }

        // Mostrar la tarjeta
        card.classList.add('revealed');

        // Verificar si es la primera o segunda tarjeta seleccionada
        if (!firstCard) {
            firstCard = card;
        } else {
            secondCard = card;
            lockBoard = true; // Bloquear el tablero temporalmente

            // Comparar las dos tarjetas seleccionadas
            if (firstCard.dataset.image === secondCard.dataset.image) {
                // Si coinciden, mantenerlas visibles
                firstCard.classList.add('matched');
                secondCard.classList.add('matched');
                resetBoard();
            } else {
                // Si no coinciden, ocultarlas después de un breve intervalo
                setTimeout(() => {
                    firstCard.classList.remove('revealed');
                    secondCard.classList.remove('revealed');
                    resetBoard();
                }, 1000);
            }
        }
    });
});

// Reiniciar el estado del tablero
```

```
// Reiniciar el estado del tablero
function resetBoard() {
    [firstCard, secondCard] = [null, null];
    lockBoard = false;
}
```