Ejercicio 1:

```
∠ Untitled (Workspace)

☐ f2.jpg

☐ f3.jpg

                                       Examen5.html
03. DWEC EJERCICIOS REPASO EV1 > Exercice 1 > JS script5.js > ...
  1 // Selección de elementos del DOM
       const taskInput = document.getElementById('taskInput');
       const addTaskBtn = document.getElementById('addTaskBtn');
      const taskList = document.getElementById('taskList');
       // Cargar tareas desde localStorage al cargar la página
       window.addEventListener('DOMContentLoaded', loadTasksFromLocalStorage);
       // Evento para añadir tareas
 10
       addTaskBtn.addEventListener('click', () => {
 11
          const taskText = taskInput.value.trim();
           if (taskText) {
 13
               addTaskToList(taskText, false);
               saveTaskToLocalStorage(taskText, false);
taskInput.value = ''; // Limpiar el input
 14
 16
           } else {
              alert('Escribe una tarea antes de añadirla.');
 17
 19
      });
 20
       // Evento para reducir el tamaño del input al hacer clic
       taskInput.addEventListener('click', () => {
   taskInput.style.animation = 'shrink 0.5s forwards'; // Aplicar la animación
 22
 23
           setTimeout(() => {
               taskInput.style.animation = ''; // Quitar la animación después
 25
 26
           }, 500);
      });
 27
 28
       // Función para añadir una tarea a la lista
       function addTaskToList(taskText, completed) {
 31
           // Crear el elemento de tarea
           const taskItem = document.createElement('li');
                                                             ⊕ Ln 1, Col 1 Spaces: 4 UTF-8 CRLF ( } JavaScript ✓ Prettier ♀
                                                                                                    ^ 라.ơ) 된 [전 <sup>11:53</sup> 등
```

```
29
    // Función para añadir una tarea a la lista
30 ∨ function addTaskToList(taskText, completed) {
31
        // Crear el elemento de tarea
32
         const taskItem = document.createElement('li');
33
        taskItem.classList.add('task-item');
34
35
         // Crear el checkbox
36
         const checkbox = document.createElement('input');
37
         checkbox.type = 'checkbox';
         checkbox.checked = completed; // Si la tarea está marcada como completada
38
39 ~
         checkbox.addEventListener('change', () => {
             taskTextElement.style.backgroundColor = checkbox.checked ? '#d3f9d8' : '';
40
41
             updateTaskStatusInLocalStorage(taskText, checkbox.checked);
42
         });
43
         // Crear el elemento de texto de la tarea
44
         const taskTextElement = document.createElement('span');
45
         taskTextElement.textContent = taskText:
46
47
         // Crear el botón de eliminar
48
         const deleteBtn = document.createElement('button');
         deleteBtn.textContent = 'X'; // Emoji de "Eliminar"
49
         deleteBtn.classList.add('delete-btn');
50
51
         deleteBtn.addEventListener('click', () => {
52 🗸
53
             // Verificar si la casilla está marcada antes de eliminar
             if (checkbox.checked) {
54 V
                 fadeOutAndRemove(taskItem, taskText);
55
56 V
             } else {
57
                 alert('No puedes eliminar esta tarea hasta que la marques como completada.');
58
59
         });
```

```
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) {
```

```
// 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;
```