Actividad de repaso Evaluación 1	
UD 02-06	Duración: 2h

Deberás desarrollar una aplicación web para gestionar tareas. La aplicación permitirá realizar las siguientes operaciones:

1. **Formulario para agregar tareas:**
    - Crea un formulario con un campo de texto en el cual los usuarios puedan ingresar el nombre de una tarea.
    - El formulario debe incluir un botón para agregar la tarea a una lista.
    - Al agregar una tarea, esta debe aparecer en la lista de tareas en la página, acompañada de los siguientes botones: "Marcar como completada", "Eliminar tarea" y "Editar tarea".
2. **Definición de clases en JavaScript:**
    - Crea una clase Tarea que tenga las siguientes **propiedades**:
        - id: un identificador único para cada tarea (puedes generar el id de manera automática usando el tiempo de creación o un contador incremental).
        - texto: el texto o descripción de la tarea.
        - completada: un valor booleano que indica si la tarea está completada o no.
    - La clase debe tener los siguientes **métodos**:
        - marcarCompletada(): Este método cambiará el valor de la propiedad completada de la tarea. Cuando una tarea esté

marcada como completada, su texto debe cambiar de color para reflejar su estado.

- eliminar(): Este método eliminará la tarea de la lista de tareas.
- editar(nuevoTexto): Este método permitirá cambiar el texto de la tarea. El usuario debe poder editar el contenido de la tarea directamente desde la lista.

3. Manipulación dinámica del DOM:
    - Utiliza JavaScript para crear y agregar dinámicamente los elementos HTML que representen la lista de tareas.
    - Cada tarea deberá estar representada por un div que contenga:
        - El texto de la tarea.
        - Un botón para marcar la tarea como completada.
        - Un botón para eliminar la tarea.
        - Un botón para editar la tarea (cuando se presione este botón, el texto de la tarea debe cambiar a un campo de texto editable).
    - Los elementos deben generarse dinámicamente a partir de las instancias de la clase Tarea. Los botones de cada tarea deben estar asociados a sus respectivos métodos (marcarCompletada, eliminar, editar).
4. Gestión de tareas con localStorage:
    - La aplicación debe ser capaz de guardar las tareas en el almacenamiento local utilizando localStorage.
    - Cada vez que el usuario agregue, marque como completada, edite o elimine una tarea, esos cambios deben reflejarse en el almacenamiento local.

- Al recargar la página, las tareas almacenadas en localStorage deben cargarse automáticamente en la interfaz, respetando su estado (completada o no) y su texto.
- Asegúrate de que, si el usuario elimina una tarea, esta también se elimine de localStorage.

5. Clasificación de tareas:
   - Implementa la funcionalidad para clasificar las tareas por su estado: "Todas", "Pendientes" y "Completadas".
   - Crea tres botones de filtro en la parte superior de la página:
     - **Todas**: Muestra todas las tareas, independientemente de su estado.
     - **Pendientes**: Muestra solo las tareas no completadas.
     - **Completadas**: Muestra solo las tareas que han sido marcadas como completadas.
   - El filtro debe actualizarse dinámicamente sin necesidad de recargar la página, y debe reflejarse en el listado de tareas visibles.

Consideraciones adicionales:

- **Validación del formulario:** El formulario de ingreso de tareas debe incluir una validación básica para evitar que se agreguen tareas vacías. Si el usuario intenta agregar una tarea sin texto, debe mostrar un mensaje de error.
- **Resistencia a la recarga:** Al recargar la página, las tareas deben conservarse, con su estado de "completado" o "no completado", y deben estar ordenadas según el filtro seleccionado en el momento de la recarga.
- **Interacción y flujo de usuario:** Asegúrate de que la experiencia de usuario sea fluida. Por ejemplo, los botones deben estar habilitados sólo cuando corresponda (por ejemplo, no permitir marcar como completada una tarea ya completada).