INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

TAREA #3:

"Lista de Tareas (To Do App)"

Integrantes:

Anastasio Salas Juan Carlos

Cabrera Duran Itzel Amayramy

Materia:

Tecnologías de Aplicaciones Web

Horario:

14:00 - 15:00 hrs.

Cd. Madero, Tamaulipas

DEJANDO HUELLA

TAREA NO. 3: Lista de Tareas (To Do App)

INSTRUCCIONES:

- **Funcionalidad:** Permite a los usuarios agregar, editar, marcar como completadas y eliminar tareas.
- **Punto clave:** Es un excelente proyecto para practicar la manipulación del DOM y el manejo del estado. Puedes implementar el almacenamiento local (localStorage) para guardar las tareas entre sesiones, sin necesidad de backend.
- **Lenguajes por utilizar:** HTML 5, CSS3 y Javascript.

DESARROLLO:

Concepto del Proyecto

El proyecto es una aplicación web de Lista de Tareas (To-Do List) desarrollada con tecnologías nativas (Vanilla JS, HTML y CSS).

Su principal diferenciador y funcionalidad es la gestión de estados mediante la separación física de las tareas en dos secciones distintas:

- **Tareas Pendientes:** Donde aparecen las tareas nuevas o no finalizadas.
- **Tareas Completadas:** Donde se mueven las tareas al marcarlas como hechas.

Además, cuenta con persistencia de datos local, lo que asegura que la información no se pierda al recargar la página.

Análisis de Funciones

A continuación, se describen las funciones y bloques de código en su orden jerárquico de ejecución y dependencia.

1. Inicialización (loadData)

Esta es la primera función que tiene efecto al cargar la página, ya que se invoca inmediatamente en la última línea del script.

Función: loadData()

Props/Parámetros: Ninguno.

Propósito: Restaurar el estado anterior de la aplicación.

Proceso:

1. Lee el almacenamiento local (localStorage) buscando las claves 'pendingData' y 'completedData'.
2. Inserta el HTML guardado directamente dentro de los contenedores de lista correspondientes (pendingListContainer y completedListContainer), recuperando así todas las tareas tal cual estaban.

2. Creación de Tareas (AddTask)

Esta función es activada por el usuario al hacer clic en el botón "Add" (definido en el HTML onclick="AddTask()").

Función: AddTask()

Props/Parámetros: Ninguno (lee directamente el estado del DOM).

Propósito: Generar una nueva tarea válida e insertarla en la interfaz.

Proceso:

1. **Validación:** Verifica si el campo de texto (inputBox) está vacío. Si lo está, muestra una alerta.
2. **Creación DOM:**
    - Crea un elemento <li> y le asigna el texto escrito por el usuario.
    - Inserta este <li> en la lista de Pendientes (pendingListContainer).
    - Crea un elemento con el símbolo \u00d7 (×), que servirá como botón de eliminar, y lo añade dentro del <li>.
3. **Limpieza:** Borra el texto del campo de entrada (inputBox.value = '').
4. **Persistencia:** Llama a saveData() para guardar el cambio.

3. Interacción y Gestión de Estado (Event Listener Global)

En lugar de tener funciones individuales para cada tarea, el proyecto utiliza un único "Event Listener" en el objeto document para detectar clics (Delegación de eventos).

Función:

(Callback anónimo dentro de document.addEventListener)

Props/Parámetros: e (el evento del clic).

Propósito: Manejar las acciones de completar, descompletar y eliminar tareas.

Proceso:

1. **Identificación del Objetivo:** Evalúa qué elemento fue clickeado (e.target).

2. **Caso 1: Clic en una Tarea (LI):**
    - Alterna la clase CSS .checked en el elemento.
    - **Lógica de Movimiento:**
        - Si la tarea adquiere la clase checked (se completa), se mueve físicamente al contenedor completedListContainer.
        - Si se le quita la clase checked (se desmarca), regresa a pendingListContainer.
    - Llama a saveData().
3. **Caso 2: Clic en Eliminar (SPAN):**
    - Identifica al padre del icono (el <li>) y lo elimina completamente del DOM (remove()).
    - Llama a saveData().

4. Persistencia (saveData)

Es una función auxiliar llamada por AddTask y por el Event Listener cada vez que hay un cambio.

Función: saveData()

Props/Parámetros: Ninguno.

Propósito: Guardar el estado actual del HTML en el navegador.

Proceso:

1. Toma todo el contenido HTML interno (innerHTML) de la lista de pendientes y lo guarda en localStorage bajo la clave 'pendingData'.
2. Toma todo el contenido HTML interno de la lista de completadas y lo guarda bajo la clave 'completedData'.