# Lista de Tareas en JavaScript


IES PIO BAROJA

2º Desarrollo de Aplicaciones Web

Julio Brady Flores Nogales

## Introducción

Este proyecto es una aplicación web para gestionar una lista de tareas. Permite a los usuarios agregar, editar, eliminar y marcar tareas como completadas. La aplicación utiliza HTML, CSS y JavaScript, y almacena las tareas en el almacenamiento local del navegador.

## 1. Estructura HTML y CSS

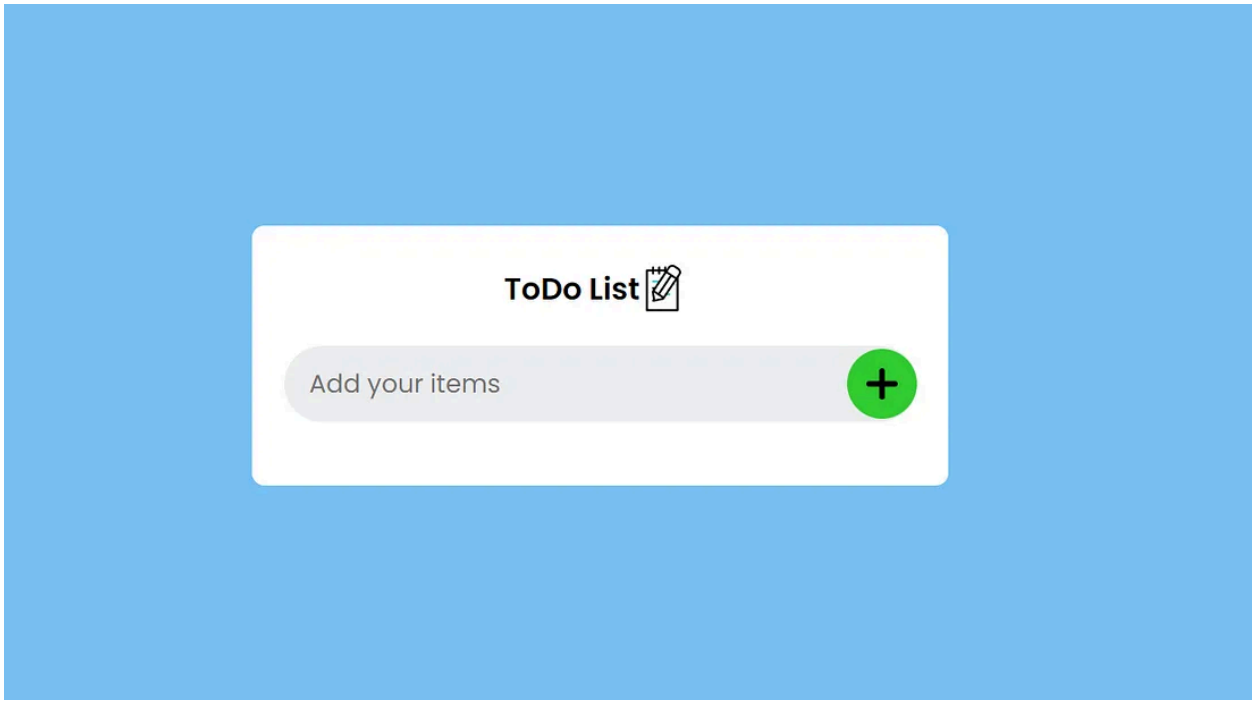
Utilizo una estructura html sencilla que se compone de:

- **Contenedor Principal:** Un `div` que agrupa todos los elementos de la lista de tareas.
- **Encabezado:** Muestra el título de la aplicación y un ícono.
- **Cuerpo de Tareas:** Un campo de entrada para agregar tareas y un botón para añadirlas.
- **Lista de Tareas:** Un ul donde se mostrarán las tareas.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aplicación de Lista de Tareas</title>
    <link rel="stylesheet" href="style.css"> <!-- Enlace a la hoja de
estilos -->
</head>
<body>
    <div class="container">
        <div class="todo-header">
            <h2>Lista de Tareas</h2>
            <img src="images/notebook.gif" height="50px" alt="Notebook">
        </div>
        <div class="todo-body">
            <input type="text" id="todoText" class="todo-input"
placeholder="Agregar tarea...">
            <img src="images/plus.png" alt="Agregar" id="AddUpdateClick"
onclick="createTodoItem();">
        </div>
        <h5 id="alert"></h5>
        <ul id="list-items" class="list-items"></ul>
```

```
    </div>
    <script src="script.js"></script> <!-- Enlace al archivo JavaScript -->
</body>
</html>
```

Ello sumado a estilos css con el siguiente acabado.




# 2. Funcionalidad JavaScript

## 2.1. Inicialización y Obtención de Elementos

```
// Obtener elementos del DOM
const todoValue = document.getElementById("todoText");
const todoAlert = document.getElementById("alert");
const listItems = document.getElementById("list-items");
const addUpdate = document.getElementById("AddUpdateClick");

// Inicializar el almacenamiento local
let todo = JSON.parse(localStorage.getItem("todo-list")) || [];
```

- **Obtención de Elementos:** Se utilizan `getElementById` para obtener referencias a los elementos del DOM que se manipularán.
- **Almacenamiento Local:** Se inicializa el array todo con los datos almacenados en el almacenamiento local, o se establece como un array vacío si no hay datos.

## 2.2. Crear un Nuevo Elemento de Tarea

```
function createTodoItem() {
    if (todoValue.value.trim() === "") {
        setAlertMessage("¡Por favor, ingresa tu tarea!");
        todoValue.focus(); // Enfocar el campo de entrada
        return; // Salir de la función
    }

    // Verificar si la tarea ya existe
    let isPresent = todo.some(element => element.item === todoValue.value);
    if (isPresent) {
        setAlertMessage("¡Este elemento ya está en la lista!");
        return; // Salir de la función
    }

    // Crear un nuevo elemento de lista
    let li = document.createElement("li");
    const todoItems = `
        <input type="checkbox" class="todo-checkbox"
onchange="toggleComplete(this)" />
        <div class="todo-text">${todoValue.value}</div>
        <div>
            <img class="edit todo-controls" onclick="updateTodoItem(this)"
src="images/pencil.png" />
            <img class="delete todo-controls"
onclick="deleteTodoItem(this)" src="images/delete.png" />
        </div>`;

    li.innerHTML = todoItems; // Agregar contenido al elemento de lista
    listItems.appendChild(li); // Añadir el nuevo elemento a la lista

    // Agregar la tarea al array y actualizar el almacenamiento local
    todo.push({ item: todoValue.value, status: false });
    setLocalStorage(); // Asegúrate de que esta función esté correctamente
```

```
implementada
    setAlertMessage("¡Elemento de tarea creado con éxito!"); // Mensaje de
éxito
    todoValue.value = ""; // Limpiar el campo de entrada
}
```

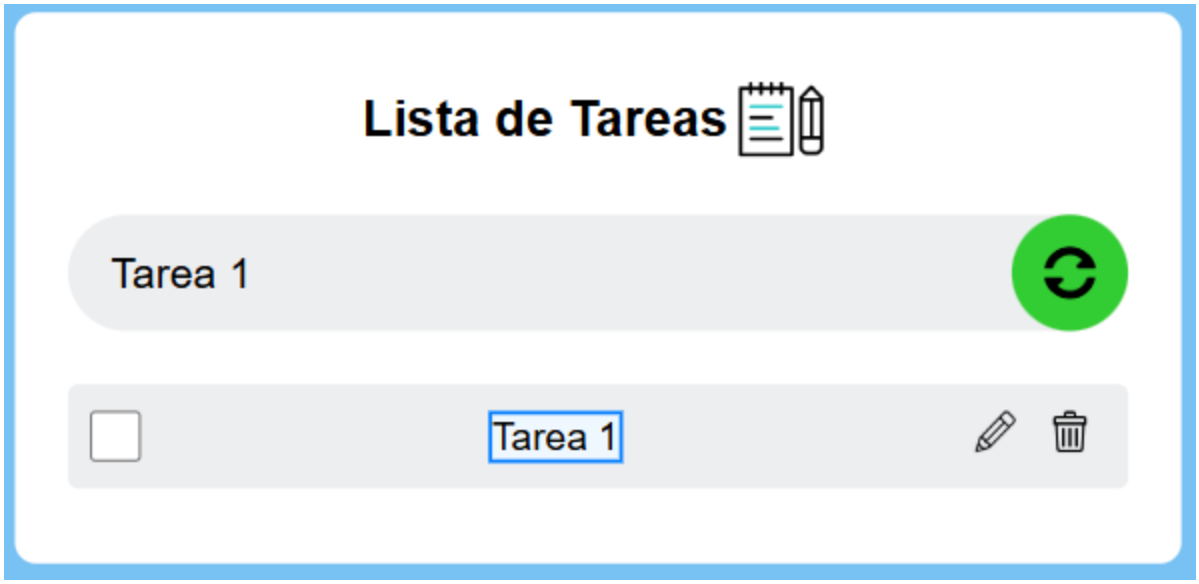
- **Validación de Entrada:** Se verifica si el campo de entrada está vacío y se muestra un mensaje de alerta si es necesario.
- **Verificación de Duplicados:** Se comprueba si la tarea ya existe en la lista para evitar duplicados.
- **Creación de Elemento:** Se crea un nuevo `li` y se añade al DOM con un checkbox, texto de tarea y botones de edición y eliminación.
- **Almacenamiento Local:** La nueva tarea se agrega al array `todo`, y se actualiza el almacenamiento local.


```
> console.log(JSON.parse(localStorage.getItem("todo-list")));
  ▼ [{…}] i
    ▶ 0: {item: 'Tarea 1', status: false}
      length: 1
    ▶ [[Prototype]]: Array(0)
< undefined
```

## 2.3. Funciones de Edición y Eliminación

```
function updateTodoItem(e) {
    const itemText =
e.parentElement.parentElement.querySelector("div").innerText; // Obtener el
texto de la tarea
    todoValue.value = itemText; // Establecer el texto en el campo de
entrada
    updateText = e.parentElement.parentElement.querySelector("div"); //
Guardar referencia al texto actualizado
    addUpdate.setAttribute("onclick", "UpdateOnSelectionItems()"); //
Cambiar la función del botón
    addUpdate.setAttribute("src", "/images/refresh.png"); // Cambiar la
imagen del botón a refrescar
    todoValue.focus(); // Enfocar el campo de entrada
}
```

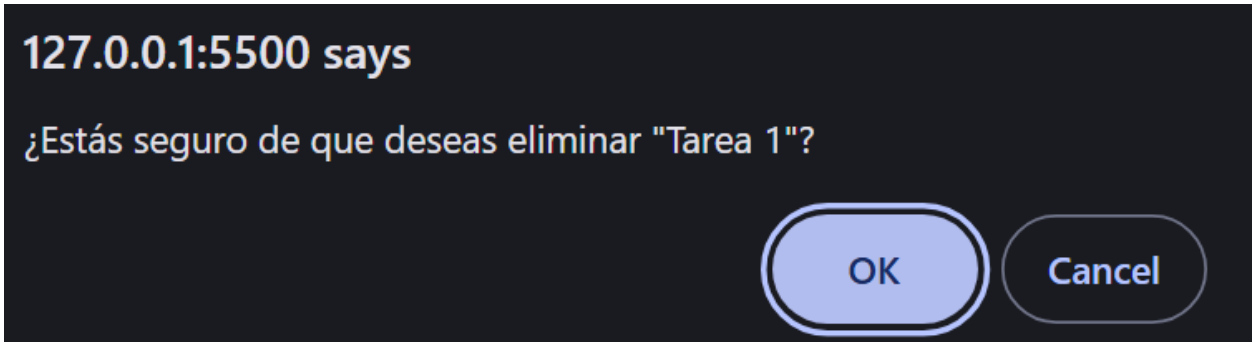
- **Obtener Texto:** Se obtiene el texto de la tarea que se desea editar y se establece en el campo de entrada.
- **Actualizar Referencia:** Se guarda una referencia al elemento que se está editando para actualizarlo posteriormente.
- **Cambiar Función del Botón:** Se cambia la función del botón "**+**" para que al hacer clic se guarde la tarea actualizada.



## 2.4. Eliminar una Tarea

```
function deleteTodoItem(e) {
    const deleteValue =
e.parentElement.parentElement.querySelector(".todo-text").innerText.trim();
// Obtener el texto de la tarea
    if (confirm(`¿Estás seguro de que deseas eliminar "${deleteValue}"?`))
{
        todo = todo.filter(element => element.item.trim() !== deleteValue);
// Comparar sin espacios
        setLocalStorage(); // Actualizar el almacenamiento local
        e.parentElement.parentElement.remove(); // Eliminar el elemento de
la lista
        setAlertMessage("¡Elemento de tarea eliminado con éxito!"); //
Mensaje de éxito
    }
}
```

- **Confirmación de Eliminación:** Se muestra un cuadro de confirmación antes de eliminar la tarea.
- **Filtrar Tarea:** Se utiliza `filter` para eliminar la tarea del array todo.
- **Actualizar DOM:** Se elimina el elemento de la lista del DOM y se actualiza el almacenamiento local.




```
Intentando eliminar: Tarea 1
Tareas después de eliminar: ▼ [] i
                              length: 0
                            ▶ [[Prototype]]: Array(0)
console.log(JSON.parse(localStorage.getItem("todo-list")));
▼ [] i
    length: 0
  ▶ [[Prototype]]: Array(0)
```