

# Laboratorio práctico: Crea una lista de tareas usando JavaScript



**Tiempo estimado necesario:** 30 minutos

## Lo que aprenderás

En este laboratorio de la aplicación de lista de tareas, explorarás el proceso de construir una interfaz funcional de gestión de tareas utilizando HTML, CSS y JavaScript. Sigue las instrucciones paso a paso para entender la implementación de características fundamentales, incluyendo la adición de tareas, la visualización dinámica de una lista de tareas, marcar tareas como completadas, limpiar tareas completadas e implementar una funcionalidad de búsqueda en tiempo real. A lo largo de este proceso, comprenderás conceptos clave como la manipulación del DOM, el manejo de eventos, la manipulación de arreglos para la gestión de tareas y el filtrado de tareas basado en la entrada del usuario.

## Objetivos de aprendizaje

Después de completar este laboratorio, podrás:

- **Implementación de la gestión de tareas:** Aprender el proceso de crear una interfaz funcional de gestión de tareas implementando características como agregar tareas, mostrar una lista de tareas de manera dinámica, alternar el estado de finalización de las tareas y limpiar las tareas completadas utilizando HTML, CSS y JavaScript.
- **Dominio de la manipulación del DOM:** Obtener dominio en la manipulación del Modelo de Objetos del Documento (DOM) utilizando JavaScript para crear y modificar elementos dentro de la página web de manera dinámica, lo que permite actualizaciones e interacciones en tiempo real dentro de la aplicación de lista de tareas.
- **Manejo de eventos e interacción del usuario:** Explorar la programación basada en eventos implementando oyentes de eventos para acciones del usuario, como agregar tareas, alternar la finalización y filtrar tareas según la entrada de búsqueda, fomentando una experiencia de usuario responsiva e interactiva.
- **Comprensión de los principios de front-end:** Comprender los principios fundamentales del desarrollo front-end, incluyendo consideraciones de diseño de UI, estilo CSS para el diseño y la estética, e integración de funcionalidades de JavaScript para crear una interfaz cohesiva y amigable de la aplicación de lista de tareas.

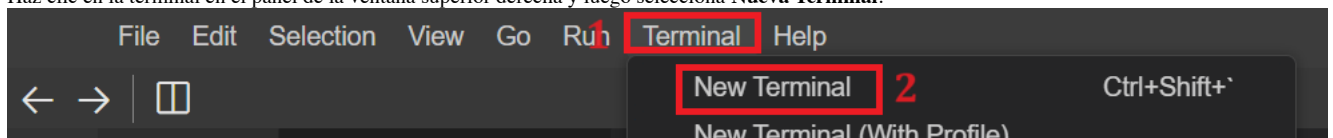
## Requisitos previos

- Conocimientos básicos de HTML.
- Comprensión básica de arreglos, métodos y propiedades de arreglos, manipulación del DOM.
- Navegador web con consola (Chrome DevTools, Firefox Console, etc.).

## Paso 1: Configuración del entorno

1. Primero, necesitas clonar tu repositorio principal en **Skills Network Environment**. Sigue los pasos dados:

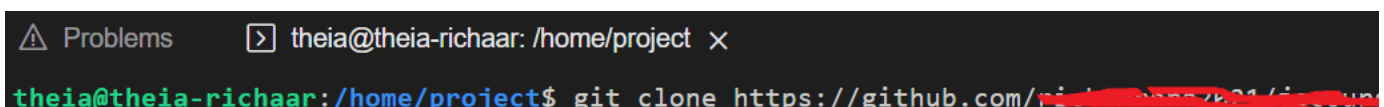
- Haz clic en la terminal en el panel de la ventana superior derecha y luego selecciona **Nueva Terminal**.



- Realiza el comando `git clone` escribiendo el comando dado en la terminal.

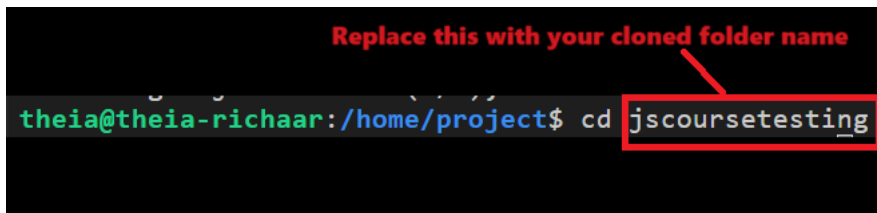
```
git clone <github-repository-url>
```

**Nota:** Pon tu propio enlace de repositorio de GitHub en lugar de `<github-repository-url>` que creaste en el primer laboratorio.



- El paso anterior clonará la carpeta de tu repositorio de GitHub en la carpeta del proyecto en el explorador. También verás múltiples carpetas dentro de la carpeta clonada.
- Ahora necesitas navegar dentro de la carpeta clonada. Para esto, escribe el comando dado en la terminal:

```
cd <repository-folder-name>
```



**Nota:** Escribe el nombre de tu carpeta clonada en lugar de <repository-folder-name>. Realiza `git clone` si has cerrado sesión en **Skills Network Environment** y no puedes ver ningún archivo o carpeta después de iniciar sesión.

- Ahora selecciona la carpeta **cloned Folder Name**, haz clic derecho sobre ella y selecciona **Nueva Carpeta**. Ingresa el nombre de la carpeta como **ToDoList**. Esto creará la carpeta para ti. Luego selecciona la carpeta **ToDoList**, haz clic derecho y selecciona **Nuevo Archivo**. Ingresa el nombre del archivo como **todo\_list.html** y haz clic en Aceptar. Esto creará tu archivo HTML.
3. Crea la estructura de plantilla de un archivo HTML agregando el siguiente contenido:

```
<!DOCTYPE html>
<html>
<head>
  <title>ToDo List</title>
</head>
<body>
  <h1>ToDo List</h1>
  <input type="text" id="taskInput" placeholder="Add a new task">
  <button id="addTaskBtn">Add Task</button>
  <!-- <input type="text" id="searchInput" placeholder="Search..." -->
  <ul id="taskList">
    <!-- Tasks will be displayed here -->
  </ul>
  <button id="clearCompletedBtn">Clear Completed</button>
</body>
</html>
```

4. El código HTML anterior incluye los puntos dados:

- Este fragmento de código representa un diseño simple de página web. Incluye los siguientes elementos:
  - Un encabezado `<h1>` que indica una “Lista de Tareas”. Un campo `<input>` con el ID `taskInput` permite a los usuarios ingresar nuevas tareas, y hay un botón con el ID `addTaskBtn` presente para agregar estas tareas.
  - Visualización de la lista de tareas: Una lista desordenada `<ul>` con el ID `taskList` muestra las tareas que los usuarios agregan. Inicialmente, está vacía y está destinada a poblarse a medida que los usuarios agregan tareas.
  - Botón para limpiar completadas: Por último, hay un botón con el ID `clearCompletedBtn` destinado a eliminar las tareas completadas de la lista.

**Nota:** Cuando hayas pegado el código, guarda tu archivo.

5. Ahora selecciona la carpeta **ToDoList** nuevamente, haz clic derecho y selecciona **Nuevo Archivo**. Ingresa el nombre del archivo **todo\_list.js** y haz clic en Aceptar. Esto creará tu archivo de JavaScript.
6. Para incluir el archivo js en **todo\_list.html**, puedes usar la etiqueta de script en el archivo HTML justo antes de la etiqueta `</body>`. Puedes usar el siguiente código para incluir y guardar el archivo de script.

```
<script src="./todo_list.js"></script>
```

## Paso 2: Definición de variables para acceder a datos

1. Declara una variable llamada **taskInput** e inicialízala de la siguiente manera:

- Escribe el código dado en el archivo **todo\_list.js**.

```
const taskInput = document.getElementById("taskInput");
```

- Esta línea recupera el elemento HTML con el ID "taskInput" del documento, asignándolo a la variable constante **taskInput**. Permite acceder al campo de entrada donde los usuarios pueden ingresar nuevas tareas utilizando JavaScript en la aplicación de Todo List.

2. De manera similar, añade el código dado en el archivo **todo\_list.js**:

```
const addTaskBtn = document.getElementById("addTaskBtn");
const taskList = document.getElementById("taskList");
const clearCompletedBtn = document.getElementById("clearCompletedBtn");
```

- En las líneas de código anteriores, document.getElementById() se utiliza para recuperar elementos específicos del documento HTML por sus IDs únicos, tales como:

- addTaskBtn obtiene el elemento del botón responsable de agregar tareas.
- taskList recupera el elemento de lista desordenada donde se muestran las tareas.
- clearCompletedBtn accede al botón utilizado para limpiar las tareas completadas.

3. Declara un array vacío con una variable llamada **tasks**.

```
let tasks = [];
```

## Paso 3: Definir varias funciones para acceder a los datos

1. Crea la función **addTask** incluyendo el código proporcionado en el archivo JavaScript.

```
function addTask() {
  const taskText = taskInput.value.trim();
  if (taskText !== "") {
    tasks.push({ text: taskText });
    taskInput.value = "";
    displayTasks();
  }
}
```

- El código anterior incluye:
  - La variable **taskText** para recuperar el valor ingresado en el elemento HTML taskInput por el usuario, eliminando cualquier espacio en blanco al final.
  - Una declaración condicional que utiliza un bloque if para verificar si taskText no es una cadena vacía; si no lo es, crea un nuevo objeto de tarea con el texto ingresado.
  - Adición de este nuevo objeto de tarea utilizando el método push del array a la array de tareas, representando la lista de tareas.
  - Restablecimiento del valor del campo taskInput a una cadena vacía después de agregar la tarea, limpiando la entrada para la próxima tarea.
  - Llamada a la función **displayTasks** para mostrar las tareas ingresadas, que crearás en el siguiente paso.

2. Crea la función **displayTasks** incluyendo el código dado en el archivo JavaScript.

```
function displayTasks() {
  taskList.innerHTML = "";
  tasks.forEach((task, index) => {
    const li = document.createElement("li");
    li.innerHTML = `<input type="checkbox" id="task-${index}" ${task.completed ? "checked" : ""}>
    <label for="task-${index}">${task.text}</label>`;
    li.querySelector("input").addEventListener("change", () => toggleTask(index));
    taskList.appendChild(li);
  });
}
```

- El código anterior incluye lo siguiente:
  - `taskList.innerHTML = ""`; para limpiar el contenido existente dentro del elemento `taskList` estableciendo su `innerHTML` a una cadena vacía.
  - `tasks.forEach` itera a través del array de tareas utilizando `forEach`, creando un elemento de lista `<li>` para cada tarea.
  - Construye contenido HTML para cada tarea asignándolo a `li.innerHTML`, que incluye una casilla de verificación, una etiqueta que muestra el texto de la tarea y los IDs correspondientes.
  - Luego, con la ayuda de `li.querySelector`, establece un listener de eventos para cada casilla de verificación dentro del elemento de lista `<li>`. Cuando cambia el estado de la casilla de verificación, se activa la función `toggleTask()`, que crearás en el siguiente paso.
  - Luego, agrega el nuevo elemento de lista creado que contiene los detalles de la tarea en la interfaz de la lista de tareas usando el método `appendChild`.

3. Crea la función **toggleTask** e incluye el siguiente código:

```
function toggleTask(index) {
  tasks[index].completed = !tasks[index].completed;
  displayTasks();
}
```

- Esta función **toggleTask** alterna el estado de finalización de una tarea específica en el array de tareas según el índice proporcionado.
- Ayuda seleccionando la casilla de verificación sin importar. Si está seleccionada, marcará esa tarea particular como completada.
- Para esto, necesitas llamar a una función más llamada **clearCompletedTasks**.

4. Crea una función **clearCompletedTasks**:

```
function clearCompletedTasks() {
  tasks = tasks.filter(task => !task.completed);
  displayTasks();
}
```

- En el código anterior, el método `filter` filtra el array de tareas, que tiene la lista de tareas ingresadas por los usuarios.
- `tasks.filter(task => !task.completed)`; filtra el array de tareas para recuperar solo las tareas que no están marcadas como completadas (`task.completed` es `false`), devolviendo un nuevo array excluyendo las tareas completadas.

5. Realiza `addEventListener` para los botones `addTask` y `clearCompletedTasks` para escuchar clics después de hacer clic en los botones **Agregar Tarea** y **Limpiar Completadas**.

```
addTaskBtn.addEventListener("click", addTask);
clearCompletedBtn.addEventListener("click", clearCompletedTasks);
```


6. La función llama a la función `displayTasks` para mostrar la tarea de todo ingresada después de hacer clic en el botón **Agregar Tarea**.

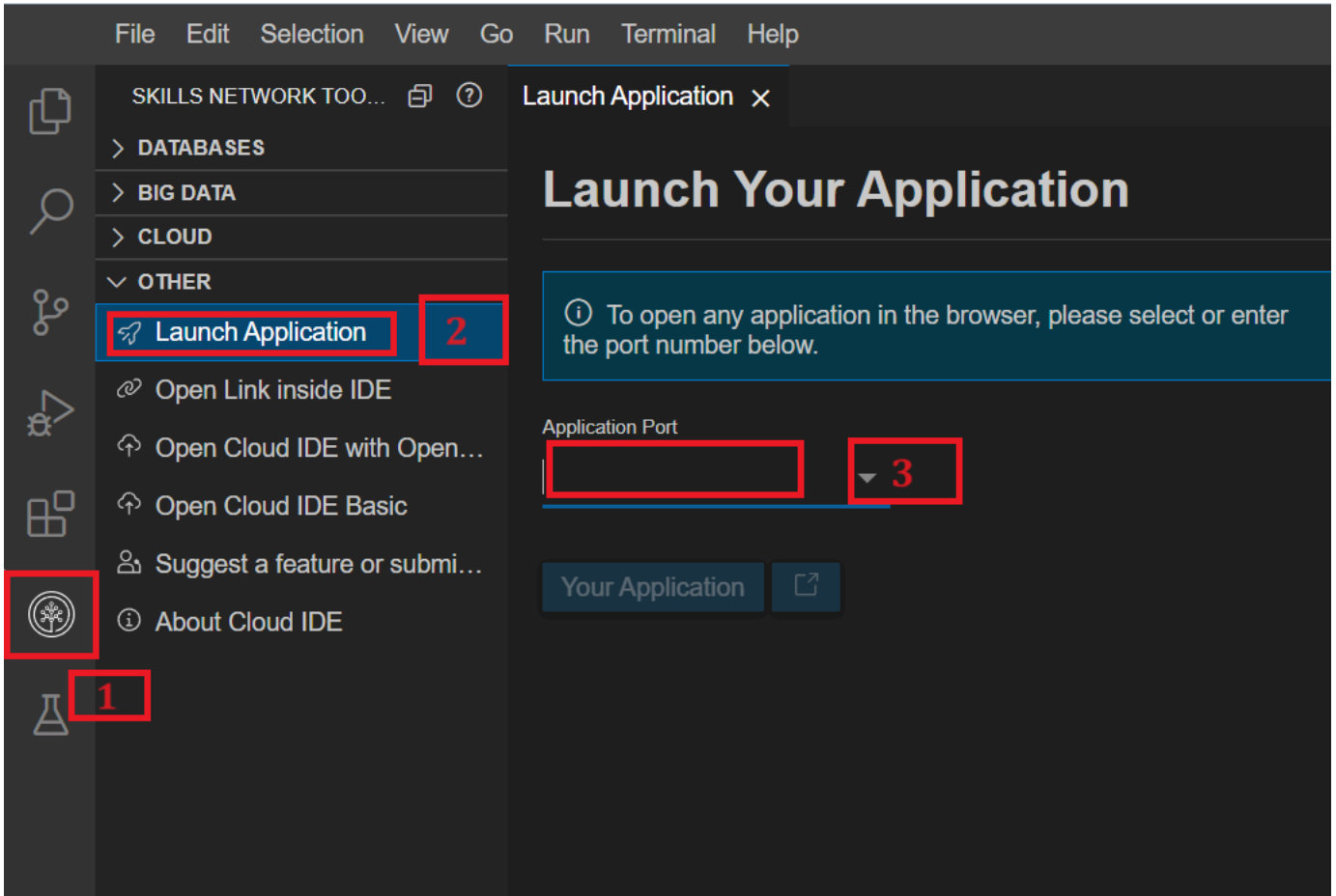
```
displayTasks();
```

## Paso 4: Verifica la salida

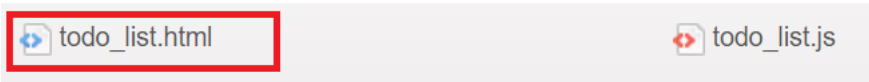
1. Para ver tu página HTML, haz clic derecho en el archivo **todo\_list.html** después de seleccionarlo, luego selecciona “Abrir con Live Server”.
  - o El servidor debería iniciar en el puerto 5500, indicado por una notificación en la esquina inferior derecha.



2. Haz clic en el botón de Skills Network a la izquierda (consulta el número 1). Se abrirá la “Caja de herramientas de Skills Network”. A continuación, haz clic en “Iniciar aplicación” (consulta el número 2). Desde allí, ingresa el número de puerto 5500 en el número 3 y haz clic en este botón .



3. Se abrirá tu navegador predeterminado donde verás el nombre de la carpeta **cloned-folder-name**. Haz clic en esa carpeta **cloned-folder-name**. Después de hacer clic, verás múltiples nombres de carpetas, entre esos nombres de carpetas haz clic en la carpeta **ToDoList**. Verás archivos relacionados con esta carpeta donde nuevamente harás clic en el archivo **todo\_list.html** como se muestra a continuación.



3. Se abrirá la página HTML y mostrará la salida como se muestra a continuación:

## To-Do List

4. Luego, ingresa cualquier tarea en el cuadro de entrada y haz clic en el botón **Agregar tarea**; verás la tarea en la página principal como se muestra a continuación.

# To-Do List

- ☐ Milk
- ☐ Bread

5. Luego puedes seleccionar esa tarea particular haciendo clic en la casilla de verificación y en el botón **Eliminar completados**. Verás que la tarea ya no es visible en la página principal.

# To-Do List

- ☒ Milk
- ☐ Bread

## Paso 5: Realizar comandos de Git

1. Realiza `git add` para agregar los últimos archivos y carpetas en el entorno de git.

```
git add --a
```

- o Asegúrate de que la terminal tenga la ruta como sigue:



2. Luego, realiza `git commit` en la terminal. Al realizar `git commit`, la terminal puede mostrar un mensaje para configurar tu `git config --global` para `user.name` y `user.email`. Si es así, entonces también necesitarás realizar el comando `git config` para `user.name` y `user.email` como se indica a continuación.

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

Luego realiza `git commit` como se indica:

```
git commit -m "mensaje"
```

3. A continuación, realiza `git push` simplemente escribiendo el comando dado en la terminal.

```
git push origin
```

- Después del comando de push, el sistema te pedirá que ingreses tu nombre de usuario y contraseña. Ingresa el nombre de usuario de tu cuenta de GitHub y la contraseña que creaste en el primer laboratorio. Después de ingresar las credenciales, todas tus últimas carpetas y archivos se enviarán a tu repositorio de GitHub.

## Tarea de práctica

1. En esta tarea de práctica, necesitas incluir un botón para borrar todas las tareas a la vez.
2. Para esto, necesitas crear un botón llamado **Borrar Todas las Tareas** en el archivo HTML.
3. Luego, crea una función para limpiar la lista de tareas, que has almacenado en el array **tasks** en el laboratorio, y llama a esta función en el momento en que se haga clic en el botón **Borrar Todas las Tareas**.

**Sugerencia:** puedes vaciar todo el array cuando el usuario haga clic en el botón.

No olvides realizar los comandos `git add`, `git commit` y `git push` nuevamente después de completar la tarea.

## Resumen

1. **Estructura HTML:** Se crearon elementos HTML esenciales como el título, encabezado de la lista de tareas, campo de entrada y botones para agregar tareas y limpiar las completadas. Se configuró una lista vacía para mostrar las tareas.
2. **Definición de variables:** Para capturar elementos HTML usando `document.getElementById`, se declararon variables. Se inicializó un arreglo vacío de tareas para gestionar las tareas.
3. **Gestión de tareas:** `addTask` recopiló la entrada del usuario, actualizó las tareas y refrescó la lista mostrada a través de `displayTasks`. Esta función generó dinámicamente elementos HTML para cada tarea. `toggleTask` y `clearCompletedTasks` gestionaron la finalización y eliminación de tareas.
4. **Manejo de eventos:** Se establecieron oyentes de eventos para los botones “Agregar tarea” y “Limpiar completadas”, activando las funciones respectivas. Inicialmente, se mostraron las tareas existentes a través de `displayTasks`. Esta configuración formó la base para una lista de tareas dinámica, permitiendo la adición, visualización, finalización y limpieza de tareas de manera fluida.

© IBM Corporation. Todos los derechos reservados.