📚 Ejercicios JavaScript Vanilla para el Gestor de Libros

🧩 1. Validación de formulario

Objetivo: Validar los datos antes de agregar un libro.

- Validar que los campos `titulo`, `autor` y `genero` no estén vacíos.
- Mostrar mensajes de error en `#errores-seccion` si hay campos inválidos.
- Evitar que se agregue el libro si hay errores.

🛠️ 2. Agregar libros a la tabla

Objetivo: Insertar dinámicamente libros en la tabla.

- Al hacer clic en "Agregar libro", añadir una fila a `#tabla-libros` con los datos del formulario.
- Incluir un botón "Eliminar" en la columna "Acciones" para borrar la fila.
- Limpiar el formulario tras añadir el libro.

🔍 3. Filtrar por género

Objetivo: Mostrar solo los libros del género seleccionado.

- Al cambiar el valor de `#filtros-genero`, mostrar solo las filas que coincidan con ese género.
- Si se selecciona "Filtrar por género", mostrar todos los libros.

📊 4. Estadísticas dinámicas

Objetivo: Actualizar los datos de estadísticas.

- Mostrar el número total de libros en `#total-libros`.
- Mostrar la distribución de géneros en `#generos-libros` (por ejemplo: Ficción: 2, Ensayo: 1…).
- Actualizar automáticamente al agregar o eliminar libros.

🧹 5. Eliminar libros

Objetivo: Gestionar la eliminación de libros.

- Al hacer clic en el botón "Eliminar" de una fila, quitarla de la tabla.
- Actualizar las estadísticas tras eliminar.

💾 6. Guardar en localStorage

Objetivo: Persistencia de datos.

- Guardar los libros en `localStorage` al agregarlos.
- Recuperar los libros al cargar la página.
- Permitir borrar todos los libros y limpiar `localStorage`.

🧪 7. Ordenar la tabla

Objetivo: Ordenar por columnas.

- Al hacer clic en los encabezados `Título`, `Autor` o `Género`, ordenar la tabla por esa columna.
- Alternar entre orden ascendente y descendente.

🧠 8. Bonus: Búsqueda por título o autor

Objetivo: Filtrar por texto.

- Añadir un campo de búsqueda.
- Mostrar solo las filas que contengan el texto en el título o autor.

👩‍🏫 Sugerencia para examen

Puedes pedir a los alumnos que implementen 3 o 4 de estos ejercicios en un tiempo limitado, evaluando:

- Uso correcto del DOM.
- Gestión de eventos.
- Manipulación de arrays y objetos.
- Buenas prácticas de código (nombres, modularidad, comentarios).

¿Quieres que te prepare una plantilla de examen con enunciados y criterios de evaluación? También puedo ayudarte a generar el archivo `script.js` con ejemplos base.

Una plantilla con los enunciados vendria bien y soluciones para luego de intentarlo poder corregirlos.

¡Perfecto! Aquí tienes una **plantilla de examen** para JavaScript vanilla basada en tu HTML del *Gestor de Libros*. Incluye:

- Enunciados claros para el alumno
- Indicaciones técnicas
- Criterios de evaluación
- Soluciones detalladas para corrección

📄 Plantilla de Examen – JavaScript Vanilla (2º DAW)

🧠 Tema: Gestor de Libros

🕑 Duración sugerida: 90 minutos

🧪 **Nivel: Intermedio**

📁 **Archivos: `index.html`, `script.js`, `estilo.css`**

📝 Enunciados

Ejercicio 1: Validación de formulario

Objetivo: Validar los campos antes de agregar un libro.

- Validar que los campos `titulo`, `autor` y `genero` no estén vacíos.
- Mostrar los errores en `#errores-seccion`.
- Evitar que se agregue el libro si hay errores.

Ejercicio 2: Agregar libro a la tabla

Objetivo: Insertar dinámicamente una fila en la tabla.

- Al hacer clic en "Agregar libro", añadir una fila con los datos del formulario.
- Incluir un botón "Eliminar" en la columna "Acciones".
- Limpiar el formulario tras añadir el libro.

Ejercicio 3: Filtrar por género

Objetivo: Mostrar solo los libros del género seleccionado.

- Al cambiar el valor de `#filtros-genero`, mostrar solo las filas que coincidan.
- Si se selecciona vacío, mostrar todos los libros.

Ejercicio 4: Estadísticas

Objetivo: Mostrar estadísticas en tiempo real.

- Actualizar `#total-libros` con el número de libros.
- Mostrar la distribución de géneros en `#generos-libros`.

✅ Criterios de Evaluación

Criterio	Puntos
Validación de campos	2
Inserción dinámica en la tabla	3
Botón de eliminación funcional	2
Filtro por género	2

Estadísticas actualizadas correctamente	3
Uso correcto de eventos y DOM	3
Código limpio y comentado	2
Total	17

🧩 Soluciones orientativas (`script.js`)

```
js
const formulario = document.getElementById("libro-formulario");
const tabla = document.querySelector("#tabla-libros tbody");
const erroresSeccion = document.getElementById("errores-seccion");
const filtroGenero = document.getElementById("filtros-genero");
const totalLibros = document.getElementById("total-libros");
const generosLibros = document.getElementById("generos-libros");

let libros = [];

formulario.addEventListener("submit", (e) => {
  e.preventDefault();
  erroresSeccion.innerHTML = "";

  const titulo = document.getElementById("titulo").value.trim();
  const autor = document.getElementById("autor").value.trim();
  const genero = document.getElementById("genero").value;

  let errores = [];
  if (!titulo) errores.push("El título es obligatorio.");
  if (!autor) errores.push("El autor es obligatorio.");
  if (!genero) errores.push("El género es obligatorio.");

  if (errores.length > 0) {
    erroresSeccion.innerHTML = `<ul>${errores.map(e => `<li>${e}</li>`).join("")}</ul>`;
    return;
  }

  const libro = { titulo, autor, genero };
  libros.push(libro);
  actualizarTabla();
  actualizarEstadisticas();
  formulario.reset();
});

function actualizarTabla() {
  tabla.innerHTML = "";
  libros.forEach((libro, index) => {
```

```
    const fila = document.createElement("tr");
    fila.innerHTML = `
      <td>${libro.titulo}</td>
      <td>${libro.autor}</td>
      <td>${libro.genero}</td>
      <td><button onclick="eliminarLibro(${index})">Eliminar</button></td>
    `;
    tabla.appendChild(fila);
  });
}

function eliminarLibro(index) {
  libros.splice(index, 1);
  actualizarTabla();
  actualizarEstadisticas();
}

filtroGenero.addEventListener("change", () => {
  const generoSeleccionado = filtroGenero.value;
  tabla.innerHTML = "";
  libros
    .filter(libro => !generoSeleccionado || libro.genero === generoSeleccionado)
    .forEach((libro, index) => {
      const fila = document.createElement("tr");
      fila.innerHTML = `
        <td>${libro.titulo}</td>
        <td>${libro.autor}</td>
        <td>${libro.genero}</td>
        <td><button onclick="eliminarLibro(${index})">Eliminar</button></td>
      `;
      tabla.appendChild(fila);
    });
});

function actualizarEstadisticas() {
  totalLibros.textContent = `Número de libros: ${libros.length}`;
  const conteo = {};
  libros.forEach(libro => {
    conteo[libro.genero] = (conteo[libro.genero] || 0) + 1;
  });
  const distribucion = Object.entries(conteo)
    .map(([genero, cantidad]) => `${genero}: ${cantidad}`)
    .join(", ");
  generosLibros.textContent = `Distribución de géneros: ${distribucion || "N/A"}`;
```

🧪 Características técnicas incluidas

Técnica ES6	Aplicación en el código
`use strict`	Activado al inicio del script
`const` y `let`	Declaración de variables
Arrow functions	Todas las funciones son flecha
Spread operator (`...`)	Al agregar libros y al filtrar
`map`	Para renderizar tabla y errores
`filter`	Para eliminar y filtrar libros
`reduce`	Para calcular estadísticas

¿Quieres que te prepare también una versión con persistencia en `localStorage`, o añadir ordenación por columnas como extensión?

persistencia con localStorage

¡Vamos allá! Para añadir **persistencia con** `localStorage` al Gestor de Libros, te muestro cómo integrar esta funcionalidad en tu archivo `script.js`, manteniendo el estilo moderno con `use strict`, ES6, `map`, `spread operator`, y funciones limpias.

💾 Persistencia con localStorage (ES6 + buenas prácticas)

🔧 1. Guardar libros en localStorage

Cada vez que se agregue o elimine un libro, actualizamos `localStorage`.

```
js
const guardarLibrosEnLocalStorage = (lista) => {
  localStorage.setItem('libros', JSON.stringify(lista));
};
```

📥 2. Cargar libros al iniciar

Al cargar la página, recuperamos los libros guardados y los mostramos.

```
js
const cargarLibrosDesdeLocalStorage = () => {
  const datos = localStorage.getItem('libros');
  if (datos) {
    libros = JSON.parse(datos);
```

```
    renderTabla(libros);
    actualizarEstadisticas(libros);
  }
};
```

🧼 3. Borrar todos los libros

Puedes añadir un botón en el HTML para borrar todos los libros y limpiar `localStorage`.

html
```
<button id="btn-borrar-todo">Borrar todos los libros</button>
```

Y en el JS:

js
```
const btnBorrarTodo = document.getElementById('btn-borrar-todo');
btnBorrarTodo.addEventListener('click', () => {
  libros = [];
  localStorage.removeItem('libros');
  renderTabla(libros);
  actualizarEstadisticas(libros);
});
```

🧩 4. Integración completa en el flujo

Actualiza estas funciones para incluir persistencia:

js
```
formulario.addEventListener('submit', (e) => {
  e.preventDefault();
  erroresSeccion.innerHTML = '';

  const titulo = document.getElementById('titulo').value.trim();
  const autor = document.getElementById('autor').value.trim();
  const genero = document.getElementById('genero').value;

  const errores = validarFormulario({ titulo, autor, genero });
  if (errores.length > 0) {
    erroresSeccion.innerHTML = `<ul>${errores.map(e => `<li>${e}</li>`).join('')}</ul>`;
    return;
  }

  libros = [...libros, { titulo, autor, genero }];
  guardarLibrosEnLocalStorage(libros);
  formulario.reset();
  renderTabla(libros);
  actualizarEstadisticas(libros);
```

```
});

const eliminarLibro = (index) => {
  libros = libros.filter((_, i) => i !== index);
  guardarLibrosEnLocalStorage(libros);
  renderTabla(libros);
  actualizarEstadisticas(libros);
};
```

🚀 5. Ejecutar al cargar la página

Al final del script:

```
js
cargarLibrosDesdeLocalStorage();
```

¿Quieres que te prepare una versión completa del archivo `script.js` con todo integrado y comentado para entregar como plantilla? También puedo ayudarte a añadir ordenación por columnas o búsqueda por texto como extensión.