Demo SPA-Juguetería

Desarrollar una aplicación web de una sola página (SPA) que permita gestionar un inventario de juguetes. La aplicación debe permitir listar, buscar, crear y borrar juguetes.

Estructura del proyecto.

- **estilos.css**: Estilos básicos para los componentes de la aplicación (navegación, buscador, listado y formulario de propiedades).
- js/**datos.js**: Contiene un array de objetos con datos de prueba de los juguetes.
- js/**juguete.js**: Define la clase Juguete con las propiedades jugueteId, nombre, marca y precio.
- js/**jugueteria.js**: Define la clase Juguetería que maneja la lógica de la aplicación, incluyendo métodos para iniciar la aplicación, obtener juguetes, crear y borrar juguetes, y generar el HTML para los diferentes componentes.

El fichero **datos.js** contiene datos de prueba de juguetes.

Vistas del proyecto.

- Inicio: cuenta con la navegación, el buscador y el listado.
- Listado: cuenta con la navegación y el listado.
- Propiedades: cuenta con la navegación y las propiedades.

El menú de navegación cuenta con dos opciones: “Inicio” y “Listado”.

Clase **Juguete** tiene las siguientes características:

Propiedades

- **jugueteId**: get, privado.
- **nombre**: get, set.
- **marca**: get, set.
- **precio**: get, set.

Métodos

- **constructor(jugueteId, nombre, marca, precio)**: Inicializa un objeto Juguete, dar valores por defecto para el caso de creación de un nuevo juguete.
- **generarHTMLPropiedades()**: Devuelve una cadena HTML con el formulario de propiedades asociado a la instancia.

Clase **Juguetería** tiene las siguientes características:

Propiedades

- **contenedor**: privado. Referencia al contenedor donde se renderiza la aplicación.
- **juguetes**: privado. Lista de juguetes (clase Juguete).
- **contador**: privado. Almacena el jugueteId máximo.

Métodos

- **constructor()**: Carga los datos de prueba, inicializa la lista juguetes y el contador.
- **iniciarApp(selector)**: Inicializa contenedor, si no lo encuentra muestra un alert informativo. Carga la vista de inicio.
- **obtenerJuguetes(filtro)**: Devuelve una lista de juguetes filtrada por nombre, si no hay filtro se devuelve la lista completa.
- **obtenerJuguete(jugueteId)**: Devuelve el juguete con el identificador indicado.
- **crearJuguete(nuevo)**: Crea un nuevo juguete asignándole un identificador único (variable contador) y lo añade a la colección de juguetes.
- **borrarJuguete(jugueteId)**: Elimina el juguete con el identificador indicado de la colección de juguetes.
- **navegarInicio()**: privado, compone la vista de inicio y la inserta en el contenedor, comprueba los eventos.
- **navegarListadoJuguetes()**: privado, compone la vista de listado de juguetes y la inserta en el contenedor, comprueba los eventos.
- **navegarPropiedades(juguete)**: privado, compone la vista de propiedades a partir del juguete y la inserta en el contenedor, comprueba los eventos.
- **asignarEventos()**: privado, para cada componente asocia los eventos específicos de dicho componente. Si la función se hace demasiado grande crea funciones auxiliares para cada componente.
- **generarHTMLNavegacion()**: Devuelve la cadena HTML con la barra de navegación.
- **generarHTMLBuscador()**: Devuelve la cadena HTML con el formulario del buscador.
- **generarHTMLListado(listaJuguetes)**: Devuelve la cadena HTML con el listado de juguetes.

Maquetado de referencia (Index.html)

```
<div id="app">
    <!-- Navegación -->
    <nav data-componente="navegacion" class="jg-navegacion">
        <ul>
            <li><a href="#" data-destino="inicio">Inicio</a></li>
            <li><a href="#" data-
destino="listadojuguetes">Listado</a></li>
        </ul>
    </nav>

    <!-- Buscador -->
    <form data-componente="buscador" name="jg-buscador">
        <input type="text" id="jg-buscador-filtro" placeholder="Buscar
por nombre..." />
        <button type="submit">Buscar</button>
    </form>

    <!-- Listado de Juguetes -->
    <div data-componente="listado" class="jg-tabla">
        <div class="jg-tabla-fila jg-cabecera">
            <div>Nombre</div><div>Marca</div><div>Precio</div>
        </div>
```

```
        <!-- Filas de juguetes -->
    </div>

    <!-- Formulario de Propiedades -->
    <form data-componente="propiedades" name="jg-formulario">
        <input type="hidden" id="jugueteid" name="jugueteid"
value="1">
        <div>
            <label for="nombre">Nombre del juguete:</label>
            <input type="text" id="nombre" name="nombre"
placeholder="Ej. Camión de bomberos" required>
        </div>
        <div>
            <label for="marca">Marca:</label>
            <input type="text" id="marca" name="marca"
placeholder="Ej. Hot Wheels" required>
        </div>
        <div>
            <label for="precio">Precio:</label>
            <input type="number" id="precio" name="precio"
placeholder="Ej. 19.99" step="0.01" required>
        </div>
        <div>
            <button type="submit">Guardar</button>
        </div>
    </form>
</div>
```

NOTA: El maquetado sugerido no incluye los botones de “nuevo”, “ver” o “borrar”, debes incluirlos y pensar en su código para poder enlazar las distintas acciones.

Estilos de referencia (estilos.css)

```
.jg-navegacion{
    list-style-type: none;
    padding: 0;
    margin: 0;

    li {
        display: inline-block;
        a {
            text-decoration: none;
            /* color: inherit; */
        }
    }
}

.jg-tabla-fila {
    display: flex;
    flex-direction: row;
    justify-content: space-evenly;

    div {
        flex: 1;
        outline: 1px solid black;
        padding: .5em;
    }
```

```
}

.jg-tabla-fila.jg-cabecera {
    background-color: azure;
    text-align: center;
    font-weight: bolder;
    color: green;
}
```

Recomendaciones

1. Lee y comprende el enunciado, hazte un esquema en papel con las partes.
2. Crea la estructura del proyecto, no codifiques.
3. Arranca juguetería, que cargue los datos de prueba en la clase prueba.
4. Para cada vista crea su plantilla y luego crea el método que la genera. Como tienes datos puedes probarlo por consola.
5. Viene la parte complicada, estructura la lógica del proyecto, define los métodos que dibujan cada vista de uno en uno. Recuerda que cada vista tiene varios componentes, en este caso sigue el mismo principio e impleméntalos de uno en uno.
6. Después de probar que se ven las vistas, debemos añadir los eventos. Sigue el mismo truco, ve troceando en partes pequeñas y resuelve cada una de manera independiente.
7. Añade comentarios útiles con lo que hace cada pieza, que te permitan tener claro cual es el objetivo y alcance de cada pieza. Si es necesario añade información del contexto.