Examen de JavaScript 2025-26

Índice

Contexto

Vamos a hacer una pequeña aplicación de libros con persistencia en localStorage, que tendrá dos botones: "Cargar libros" y "Limpiar". El primero leerá libros de una bbdd dada por el profesor y los mostrará en el html. Y "Limpiar" vaciará la lista del html y borrará el localStorage.

Datos en: http://10.103.255.0:3000/libros
Estructura: [{ "isbn":"...", "titulo":"...", "autor":"...", "precio":19.95, "year":2021 }, …]
Trabaja en script.js, sin librerías externas, con "use strict".

Lista de funciones:

Función	Descripción
async function getBooksFromURL(URL_BOOKS) {}	Recupera los libros desde la URL indicada usando fetch (petición asíncrona).
function saveToLocalStorage(arrayBooks) {}	Guarda el array de libros en localStorage (por ejemplo, como JSON).
function getFromLocalStorage() {}	Lee los libros almacenados previamente en localStorage y los devuelve (parseando el JSON).
function renderBooks() {}	Pinta en el HTML la lista de libros (por ejemplo, dentro de un o similar).
function onLimpiarClick() {}	Limpia la lista mostrada en el HTML y borra los datos de libros del localStorage.
function calculaTotalEjemplares() {}	Calcula el número total de ejemplares/libros a partir del array de datos disponible.
function calculaPrecioTotal() {}	Calcula la suma total de los precios de todos los libros del array.
async function init() {}	Inicializa la aplicación: carga libros (de URL o

Función	Descripción
	localStorage), pinta la lista y asigna eventos.

Enunciado

A) Al iniciar la página

1. Al cargar la página se llama a la función **init()** que verificará:
   - Si existe "libros" en localStorage:
     - Llama a la función **getFromLocalStorage()** para recuperalos con JSON.parse(...) y úsalo como fuente de datos inicial.
     - “Pinta” los libros en el HTML, llamada a la función **rederBooks**(arrayBooks).
   - **Si no hay datos en** localStorage **al iniciar, muestra “No hay libros guardados”.**

B) Recuperación de datos de libros con (fetch).

1. **Al pulsar #btnCargar**, llama a la función **getBooksFromURL()** que hace fetch a la url http://10.103.255.0:3000/libros.
2. Controla errores de red y de HTTP:
   - Comprueba resp.ok;
   - Si falla, muestra un mensaje visible en la página. En <di id=”msgError”>
3. Transformación con map (obligatorio)
   Con el JSON recibido, crea un nuevo array usando map con esta forma:
   { id: isbn, titulo, autor, precio, year, igic: (precio*0.07) }.
4. Guarda datos en localStorage, hazlo en la función **saveToLocalStorage(arrayBooks)** con la clave "libros" mediante JSON.stringify(...).
5. “Pinta” la lista en el HTML, hazlo en la función **renderBooks()**.

C) Pintar los libros en lista: rederBooks()

1. Rocoge los libros de localStorage con la función **getFromLocalStorage(). Si no hay datos muestra el mensaje “No hay libros guardados”,** si los hay prosigue.
2. Vaciar “listaLibros” del DOM para evitar que se dupliquen.
3. Recorre el array de libros y muestra el resultado en una lista desordenada dentro de <ul>, que tiene por id #libros.
   - Cada libro se representará como un elemento <li> con el siguiente formato mínimo (puedes generarlo con createElement o con innerHTML controlado):

```
<li>
  <strong class="titulo">TÍTULO</strong>
  <span class="autor">Autor: AUTOR</span>
  <span class="isbn">ISBN: ISBN</span>
  <span class="year">Año: YEAR</span>
  <span class="precio">Precio: 19,95 €</span>
  <span class="igic">IGIC (7%): 1,40 €</span>
</li>
```

4. Actualizar #numEjemplares y #precioTotal. Calcula el precioTotal usando “reduce”.

D) Funciones para cálculos

1. Calcula el número total de libros que tienes y muestralo en el id #numEjemplares. Hazlo en la función **calculaTotalEjemplares().**

2. Calcula el total de Precio de los libros y muestralo en el id #precioTotal. Hazlo en la función **calculaPrecioTotal().**

E) Al pulsar el botón con id #btnLimpiar se llama a la función onLimpiarClick() que hace:

- se borra la lista renderizada
- se elimina la clave "libros" de localStorage.

Rúbrica

Criterio (RA)	Excelente (10)	Bien (7)	Suficiente (5)	Insuficiente (0)
RA4-b (1,0 pts) Estructura en funciones definidas por el usuario (getBooksFromURL, saveToLocalStorage, getFromLocalStorage, renderBooks, onLimpiarClick, calculaTotalEjemplares, calculaPrecioTotal, init).	Define todas las funciones pedidas, con nombres y parámetros coherentes. Cada función tiene una responsabilidad clara (fetch, almacenamiento, pintado, cálculos, limpieza) y init() orquesta correctamente el flujo descrito en A–E. (1,0 puntos)	Falta 1 función menor o mezcla ligeros detalles (algún querySelector o cálculo) fuera de la función correspondiente, pero la modularización general está clara y init() gestiona casi todo el flujo. (0,7 puntos)	Faltan varias funciones o parte importante de la lógica está fuera (código suelto en el global), pero se aprecia intento de separar en funciones y se usa al menos la mitad de las funciones pedidas. (0,5 puntos)	No estructura el código en funciones (todo en un único bloque o callbacks anónimos) o ignora por completo la lista de funciones indicada. (0 puntos)
RA7-e (3,0 pts) Comunicación asíncrona con fetch y actualización dinámica del documento (apartado B). “Se ha utilizado comunicación asíncrona en la actualización dinámica del documento Web.”	getBooksFromURL(URL_BOOKS) es async, usa fetch a la URL dada, comprueba resp.ok, diferencia claramente errores de red/HTTP, muestra mensaje claro en #msgError, transforma el JSON con map a {id,isbn,…,igic}, guarda datos llamando a saveToLocalStorage, y tras la carga actualiza correctamente la lista (llamando a renderBooks o	Hace fetch a la URL y mapea bien los datos, pero el control de errores es incompleto (por ejemplo, solo catch genérico o no usa resp.ok), o el mensaje en #msgError es poco informativo, o la actualización de la lista depende de detalles poco claros (p.ej. lógica duplicada en el botón). Aun así,	Se realiza la petición y se obtienen libros, pero: el map no ajusta bien la estructura o el igic, o no se muestra mensaje de error en #msgError, o la actualización de la lista no está claramente ligada al resultado de la petición (se hace a medias). La parte asíncrona funciona de	No hay comunicación asíncrona correcta (no usa fetch, la URL es incorrecta sin justificación, o el código no llega a procesar la respuesta); no transforma los datos recibidos o no actualiza la página a partir de ellos. (0 puntos)

Criterio (RA)	Excelente (10)	Bien (7)	Suficiente (5)	Insuficiente (0)
	equivalente). Flujo correcto al pulsar #btnCargar. (3,0 puntos)	funciona en condiciones normales. (2,1 puntos)	forma básica. (1,5 puntos)	
RA3-g (2,0 pts) Uso de almacenamiento en cliente para persistir libros: lectura/escritura en localStorage como JSON (apartados A, B, C, E). “Se han utilizado «cookies» para almacenar información y recuperar su contenido.” (adaptado a localStorage).	saveToLocalStorage(arrayBooks) guarda correctamente con JSON.stringify bajo la clave "libros". getFromLocalStorage() comprueba existencia, hace JSON.parse seguro y devuelve un array válido. init() usa esos datos al arrancar y muestra “No hay libros guardados” cuando procede. onLimpiarClick() elimina la clave y sincroniza la interfaz. Uso consistente del mismo formato de datos en toda la app. (2,0 puntos)	Guarda y lee de localStorage usando JSON, pero con pequeños fallos: no controla bien el caso “no existe clave”, el mensaje de “No hay libros guardados” no siempre se actualiza, o hay algún detalle menor de sincronización (por ejemplo, borra localStorage pero no el mensaje, o al revés). (1,4 puntos)	Usa localStorage de forma básica (guardar y/o leer), pero: guarda en formato no JSON o inconsistente, o el flujo de inicio no se apoya realmente en getFromLocalStorage, o la limpieza no borra todos los datos. Aun así, se aprecia persistencia mínima entre recargas. (1,0 puntos)	No utiliza localStorage, lo hace de forma incorrecta (ej. intenta guardar objetos sin serializar y no los recupera), o el resultado no permite persistir realmente los libros entre recargas. (0 puntos)
RA6-c (2,0 pts) Acceso y modificación del DOM: pintar lista de libros, mensajes y limpieza de la interfaz (apartados A, C, E). “Se ha creado y verificado un código que acceda a la estructura del documento… Se han creado nuevos elementos de la estructura y modificado elementos ya existentes.”	Vacía correctamente #libros antes de pintar. Recorre el array y genera con la plantilla indicada (título, autor, isbn, year, precio formateado, IGIC formateado) usando createElement y/o innerHTML controlado. Actualiza mensajes (“No hay libros guardados”) de forma coherente según haya o no datos. onLimpiarClick() limpia visualmente la lista y deja la interfaz en estado inicial. (2,0 puntos)	Genera y muestra la lista de libros correctamente, pero le falta algún detalle de formato (por ejemplo, alguna clase o parte del texto) o la gestión de mensajes no es completa (a veces no muestra “No hay libros guardados”, o no lo quita cuando hay datos). La limpieza de la interfaz funciona casi siempre. (1,4 puntos)	Muestra la lista de alguna forma (aunque no respete del todo la plantilla) y accede al DOM para vaciar o volver a pintar, pero con errores visibles: duplicados, no vacía antes, formato confuso o mensajes poco claros. (1,0 puntos)	No consigue mostrar la lista en #libros de forma funcional, o nunca la vacía, o el DOM no refleja el estado real de los datos (por ejemplo, sigue mostrando libros borrados o no muestra los cargados). (0 puntos)
RA4-c (1,5 pts) Uso de arrays y métodos de recorrido (map, reduce, etc.) para transformar datos y calcular totales (apartados B, C, D). “Se han reconocido las características del lenguaje relativas a la creación y uso de arrays. Se han creado y utilizado arrays.”	Usa map para transformar el JSON en el nuevo array con estructura { id, titulo, autor, precio, year, igic } y reduce (u otro método adecuado) para calcular correctamente #numEjemplares y #precioTotal. Cálculos correctos y formateo razonable (decimales, símbolo €). (1,5 puntos)	Usa arrays y métodos de recorrido (forEach, map, reduce o combinaciones) para transformar y calcular, pero con pequeños errores: igic mal redondeado, total correcto pero usando un bucle tradicional en vez de reduce, o algún detalle de formato menor. (1,05 puntos)	Trabaja con arrays de forma básica (for/forEach) y calcula al menos uno de los totales (ejemplares o precio), aunque el uso de map/reduce sea parcial o incorrecto. La lógica aritmética es en general correcta. (0,75 puntos)	No utiliza arrays de forma adecuada (trabaja solo con valores sueltos, re-pide siempre al servidor, o los totales son incorrectos o inexistentes). (0 puntos)
RA6-e (0,5 pts) Asociación de eventos del modelo: carga inicial y botones #btnCargar y #btnLimpiar. “Se han asociado acciones a los eventos del modelo.”	Usa DOMContentLoaded (o equivalente) para llamar a init(). Asocia correctamente el click de #btnCargar a getBooksFromURL/flujo de carga y el click de #btnLimpiar a onLimpiarClick(). No hay efectos colaterales (múltiples registros, etc.). (0,5 puntos)	Asocia los eventos principales, pero con detalles mejorables: init() se llama de otra forma pero consistente, o se mezclan funciones en el manejador de click en vez de delegar en las funciones diseñadas, aunque el comportamiento general es correcto. (0,35 puntos)	Hay intentos de asociar los eventos (algún addEventListener o manejador inline) pero el flujo no está del todo bien: falta algún botón o se llama a las funciones incorrectamente. (0,25 puntos)	No asocia eventos relevantes (la app no reacciona a los botones, o solo funciona recargando manualmente desde consola). (0 puntos)