Examen de JavaScript, Cursos E-Learning

Ejercicio 0: Configuración del Proyecto (0.5 puntos)

Requisitos de partida:

- Crea un proyecto utilizando **Vite**.
- Levanta una API en el puerto **3500** a través de un script llamado `examen-javascript` utilizando el fichero `db.json`.
- Crea un fichero **.env** para las variables globales.

Ejercicio 1: Gestión de Cursos en la API (5 puntos)

a) Filtrar Cursos por Nivel (1 puntos)

Implementa una función llamada `getCourses(level)` que reciba como parámetro un `level` (por ejemplo, `"avanzado"` o `"intermedio"`) y devuelva un array con los cursos que correspondan a ese nivel.

b) Crear un Nuevo Curso en la API (1 puntos)

Implementa una función llamada `createCourse(data)` que reciba un objeto `data` con la información de un nuevo curso y lo añada a la API.

Requisitos:

- La función debe recibir un objeto `data` con las propiedades `title`, `instructor`, `level`, `duration`, `rating`, y `tags`.
- Debe retornar una Promesa que devuelva el curso recién creado, incluyendo el `id` generado por la API.
- Si `data` es inválido (por ejemplo, si falta alguna propiedad o algún tipo no es el correcto), genera un error manejado por la función.

c) Actualización de Etiquetas en Cursos (2 puntos)

Crea una función `updateCourseLevel()` que recorra todos los cursos en la API y, según la duración de cada uno, añada una etiqueta (`tag`) en `tags`.

Requisitos:

1. Para cada curso:
    - Si la duración (`duration`) es menor o igual a 30 minutos, agrega `"Express"` a `tags`.
    - Si es mayor, agrega `"Extenso"` a `tags`.
2. Retorna una Promesa que devuelva un array con los cursos actualizados.

> ***Nota***: Evita `forEach` con `async/await`. Usa `for...of` para secuencial o `Promise.all` para paralelo y mejorar el rendimiento.

d) Media de los Cursos (1 punto)

Crear una función `getAverageRating()` que le pasemos como parámetro un array de cursos (el id de los cursos), devolviendo el `rating` promedio de todos los cursos de esa lista usando `reduce()`

Ejercicio 2: Cursos en Progreso (1.75 puntos)

Implementa una función `removeCourseInProgress(studentId, courseId)` que elimine un curso en progreso de un estudiante en la API.

- Si el `studentId` existe, elimina el `courseId` de su progreso.
- Cada vez que elimines un curso, guarda en `localStorage` bajo la clave `BackupProgress` un objeto que contenga `studentId` y `courseId` eliminados. Si ya existe información previa, añade el nuevo registro sin sobreescribir el anterior.

Ejercicio 3: Listado de Cursos Completados por Estudiante (1.75 puntos)

Instrucciones:

Crea una función llamada `getCompletedCourses(studentId)` que devuelva un `MAP` con todos los cursos que un estudiante ha completado.

Requisitos:

- Recibir como parámetro el `studentId` para identificar al estudiante en la API.
- Un curso se considera completado cuando el número de lecciones en `completedLessons` del estudiante coincide con el número total de lecciones en `lessons` del curso.
- La funciño debe devolver un Map con los cursos completados por el alumno con ese `studentId`, teniendo como clave la `couseId` y cuyo contenido sea un objeto como el que se muestra a continuación:

```
{
  courseTitle: "Título del curso",
  lastAccess: "Fecha del último acceso en formato ISO",
  completedLessons: número de lecciones completadas
}
```

- Si el estudiante no ha completado ningún curso, devuelve un `Map` vacío.

Ejercicio 4: Verificación de las Funciones (1 punto)

Crea una función llamada `init()` en `main.js` que realice un test o comprobación de cada ejercicio realizado.

Nota:

> - Asegúrate de que todas las funciones se comporten como se espera y documenta todas las funciones utilizando **JSDoc**.
> - Añade test para comprobar que tus funciones controlan todo lo que se pide en el examen.
> - Recuerda añadir tu `nombre` a cada fichero .js que entregues con la sintaxis de **JSDoc**.
> - Recuerda gestionar toda la **Asincronía** controlando los `posibles errores` que puedan ocurrir
> - Este examen cubre todos los resultados de aprendizaje de Javascript sin incluir DOM ni POO.