Examen de Desarrollo de Interfaces

Tema 2. JavaScript

2º DAM

Alumno/a: _____ **Nota:** _____

Instrucciones generales:

- Duración: **2 horas y 30 minutos**.
- Crea una carpeta llamada `Examen_TuNombre` y dentro un archivo por ejercicio (ej: `ejercicio1.js`, `ejercicio2.html`, etc). Si hay varios archivos organiza en carpetas.
- Entrega en Moodle la carpeta comprimida.

1. Ejercicio 1 (2,5 puntos)

Vamos a simular un sorteo de Lotería de Navidad. Partimos de dos constantes (EL_GORDO y EL_SEGUNDO) y un array fijo misDecimos con los números que tú juegas. Debes escribir la lógica necesaria en JavaScript para completar estos pasos:

- **a)** Crea un array vacío llamado **pedreas**. Debes llenarlo con 10 números aleatorios distintos entre 0 y 99999. Es obligatorio comprobar que el número generado no esté ya dentro del array pedreas para evitar duplicados. ***(0,5 puntos)***
- **b)** Calcula cuáles son los **números anterior y posterior a EL_GORDO** para darles un premio de 2.000€. Ten en cuenta que si EL_GORDO fuera el 00000, el anterior sería el 99999. Si fuera el 99999, el posterior sería el 00000. Implementa la lógica para determinar esos dos números (anterior y posterior) correctamente y guárdalos en dos variables. ***(0,75 puntos)***
- **c)** Recorre tu array misDecimos y **comprueba cada número** siguiendo este orden (si cumple una condición, muestra el premio y pasa al siguiente número sin comprobar el resto de condiciones):
  - Si es igual a EL_GORDO: Muestra "¡Premio Gordo! El número [X] gana 4.000.000€".

- Si es igual a EL_SEGUNDO: Muestra "¡Segundo Premio! El número [X] gana 1.250.000€".
- Si es igual al número anterior o al posterior (calculados en el paso 2): Muestra "¡Aproximación! El número [X] gana 2.000€".
- Si el número está incluido dentro del array pedreas: Muestra "¡Pedrea! El número [X] gana 1.000€".
- Si la última cifra del número coincide con la última cifra de EL_GORDO: Muestra "Reintegro para el número [X]".
- En caso contrario: "El número [X] no ha sido premiado". ***(1 punto)***

2. Ejercicio 2 (2 puntos)

En este ejercicio debes construir una pequeña aplicación web para gestionar el catálogo de una agencia de superhéroes de bajo presupuesto. Partiendo de una lista inicial de héroes, deberás procesar la información para calcular sus salarios y permitir búsquedas en tiempo real.

```
const heroes = [
    { alias: "Capitán Obvio", poder: "Te dice lo que acabas de ver",
↪   nivelDestreza: 10, pideComida: true },
    { alias: "La Mujer Invisible (si nadie mira)", poder: "Se hace
↪   invisible cuando está sola", nivelDestreza: 5, pideComida: false },
    { alias: "El Hombre Microondas", poder: "Calienta la leche
↪   demasiado", nivelDestreza: 20, pideComida: true },
    { alias: "Thor-pe", poder: "Se da con el martillo en el dedo",
↪   nivelDestreza: 1, pideComida: true }
];
```

- **a)** Antes de renderizar nada, genera un nuevo array llamado **heroesContratables usando .map()**. Cada nuevo objeto debe tener todas las propiedades originales copiadas mediante el operador spread. Además, se añadirá la propiedad sueldo que será el nivelDestreza * 10 y si pideComida es true se suman 40€ al sueldo. ***(0,5 puntos)***
- **b)** Implementa un **buscador que filtre por alias**. La lista de resultados debe regenerarse en tiempo real (evento input) cada vez que el usuario escriba en la caja de texto.
  - Si el buscador está vacío, se muestran todos los héroes.
  - Si hay texto, se muestran solo los coincidentes. Al generar el HTML de los resultados, es obligatorio utilizar desestructuración de objetos para extraer las

propiedades alias, poder y sueldo en constantes antes de crear la cadena de texto. Debes utilizar un método iterador de arrays para realizar el filtrado. `<li> <strong>[Alias]</strong>: [Poder] (Precio: [Sueldo]€) </li>`. ***(1 punto)***

- **c)** Encima de la lista de resultados, añade un contenedor para mostrar al "Candidato Principal" (que será siempre **el primero de la lista filtrada**).
- Dentro de tu función de renderizado, si hay resultados, obtén el primer héroe del array utilizando desestructuración de arrays.
- Crea una función arrow llamada generarFicha que reciba dos parámetros: el nombre del héroe y una prioridad.
- El parámetro prioridad debe tener un valor por defecto de "Baja".
- Usa esa función para pintar el mensaje en el contenedor. Ejemplo: "Candidato: Thor-pe | Prioridad: Baja". ***(0,5 puntos)***

3. Ejercicio 3 (1 punto)

A partir de los archivos proporcionados `ejercicio3.html`, `main.js`, `saludo.js` y `utilidades.js`, debes convertir el proyecto en una página web funcional que utilice **módulos ES6**. Para ello, añade en `ejercicio3.html` la carga del módulo principal `main.js`, y dentro de los archivos JavaScript implementa las **exportaciones por defecto y normales (nombradas)** que se piden, así como las **importaciones correspondientes**. ***(1 punto)***

4. Ejercicio 4 (4,5 Puntos)

Desarrolla una página web sencilla para una tienda de videojuegos que utilice *json-server* como backend. La aplicación debe mostrar una **lista de productos** y un **carrito de la compra** en la parte superior, permitiendo añadir y eliminar productos del carrito mediante operaciones realizadas contra el servidor.

a) Implementa la carga de **productos** desde el endpoint `/productos`, mostrando su imagen, nombre, descripción, precio y un botón Comprar para cada uno. ***(1,25 punto)***

b) Implementa la carga del **carrito** desde el endpoint `/carrito`. Si no contiene elementos, muestra el mensaje "El carrito está vacío". ***(0,5 puntos)***

c) Al pulsar el **botón Comprar**, añade el producto al carrito siempre que no esté ya incluido, realizando un `POST` al servidor. Tras ello, actualiza visualmente el carrito. ***(1,5 punto)***

d) Para cada producto del carrito, añade un **botón Borrar** que permita eliminarlo completamente enviando un `DELETE` al endpoint correspondiente y actualizando el carrito en pantalla. ***(1,25 punto)***