DEW Examen Tema 1

Fundamentos de JavaScript Moderno y Manipulación del DOM

Objetivo del Examen

El propósito de esta prueba es evaluar su capacidad para resolver problemas utilizando la lógica de programación de JavaScript y para manipular el Document Object Model (DOM) de forma dinámica con el fin de crear interfaces de usuario interactivas. Se espera que demuestre su dominio sobre estructuras de datos, funciones y la interacción con el navegador.

Criterios de Evaluación

Las soluciones serán evaluadas según los siguientes criterios, que reflejan las buenas prácticas en el desarrollo de software :

- **Correctitud Funcional**: El código debe cumplir de manera precisa y completa con todos los requisitos funcionales descritos en cada ejercicio.
- **Calidad del Código**:
    - **Legibilidad**: El código debe ser claro, bien estructurado y fácil de entender. Se valorará el uso adecuado de indentación, espaciado y comentarios cuando sean necesarios.
    - **Nomenclatura**: Se debe utilizar una nomenclatura descriptiva y coherente para variables y funciones (por ejemplo, camelCase para variables y funciones).
    - **Simplicidad**: Se debe evitar la complejidad innecesaria y el código redundante.
- **Eficiencia y Buenas Prácticas**: Se valorará positivamente el uso de métodos modernos y apropiados para la tarea, como los métodos de array (map, filter,

reduce), en lugar de bucles tradicionales cuando el contexto lo permita, demostrando una comprensión de enfoques funcionales.

- **Uso Adecuado de Recursos**: Se permite la consulta de apuntes de JavaScript, ejercicios resueltos en clase, y documentación oficial o no oficial (páginas web) para métodos y conceptos. **No está permitida** la consulta a compañeros de clase o cualquier persona online. Tampoco se permite crear consultas en foros, aunque sí está permitido visualizar foros existentes. El uso de Inteligencia Artificial, ya sea en modo chat o integrada en el editor de código, está **estrictamente prohibido**.

Entorno de Trabajo

Se le proporcionará una carpeta que contiene una subcarpeta para cada ejercicio con los archivos necesarios para su realización. Para cada ejercicio, en su subcarpeta se presentan los siguientes archivos:

- **ejercicioX.html**: La estructura base de la página web.
- **ejercicioX.css**: Estilos visuales predefinidos para los elementos de la página.
- **datos_ejercicioX.js**: Un archivo que contiene los datos necesarios para los ejercicios.
- **ejercicioX.js**: Un archivo JavaScript donde se debe implementar lo que se pide.

Su trabajo deberá realizarse íntegramente en el archivo **ejercicioX.js**. No debe modificar ningún otro archivo (salvo **ejercicioX.html** para incluir la referencia a **ejercicioX.js**).

Forma de entrega

- Se creará una carpeta con el **nombre del alumno**.
- En dicha carpeta se pondrán cada una de las subcarpetas de los ejercicios realizados. *Si no se realiza un ejercicio, no incluir la carpeta.*

- Se comprime la carpeta y se sube a la tarea indicada.

Ejercicios

Ejercicio 1: Procesador de Textos para Nombres de Usuario

Puntuación: **2 puntos**

Descripción:

Cree una función llamada ***normalizarNombres(nombres)***. Esta función recibirá un array de strings, donde cada string es un nombre de usuarios que puede contener varios espacios en blanco al principio, al final o entre palabras, y una mezcla desordenada de mayúsculas y minúsculas.

La función debe procesar este array y devolver un nuevo array donde cada nombre de usuario cumpla con las siguientes condiciones:

- No tenga espacios en blanco al principio ni al final.
- Entre las palabras del nombre no debe haber más de un único espacio en blanco.
- Esté formateado en "formato de título", es decir, la primera letra de cada palabra debe estar en mayúscula y el resto de las letras en minúscula.

Ejemplo:

```
[ "   juan   pérez   gómez   ",  "  maría   ANTONIA  García",
"roBERTo   ",  "  ana  luisa   fernández   díaz  ",   "ELENA",  "
pedro   LÓPEZ"];
```

Debe devolver:

```
['Juan Pérez Gómez', 'María Antonia García', 'Roberto', 'Ana
Luisa Fernández Díaz', 'Elena', 'Pedro López']
```

Ejercicio 2: Análisis de Inventario

Descripción:

Implemente las funciones que operen sobre un array de productos (como el INVENTARIO proporcionado en ***datos_ejercicios2.js***). Estas funciones deben ser puras, es decir, no deben modificar el array original que reciben como parámetro.

- ***filtrarPorCategoria(productos, categoria):***

  Recibe: Un array de productos y un string con el nombre de una categoría.

  Devuelve: Un nuevo array que contiene únicamente los productos que pertenecen a la categoría especificada.

  Ejemplos usando productos = INVENTARIO:

  - Categoría = “Electrónica” devuelve [
    {id: 'SKU-001', nombre: 'Laptop Pro 15', precio: 1200.5, stock: 15, categoria: 'Electrónica', …}
    {id: 'SKU-004', nombre: 'Monitor UltraWide 34"', precio: 450, stock: 10, categoria: 'Electrónica', …},
    {id: 'SKU-006', nombre: 'Webcam HD 1080p', precio: 60, stock: 30, categoria: 'Electrónica', …}]
  - Categoría = “Hogar” devuelve []

- ***obtenerNombresDeProductos(productos):***

  Recibe: Un array de productos.

  Devuelve: Un nuevo array que contiene únicamente los nombres (strings) de esos productos.

  Ejemplos usando productos = INVENTARIO:

  > Devuelve: ['Laptop Pro 15', 'Mouse Inalámbrico Ergonómico', 'Teclado Mecánico RGB', 'Monitor UltraWide 34"', 'Silla de Oficina Premium', 'Webcam HD 1080p']

- ***calcularValorTotalInventario(productos):***

  Recibe: Un array de productos.

  Devuelve: Un número que representa el valor total del inventario. Este valor se calcula sumando el resultado de multiplicar el precio por el stock de cada producto en el array.

  Ejemplos usando productos = INVENTARIO:

  > Devuelve: 35326.25

- ***filtrarPorEtiquetas(productos, …etiquetas):***

  Recibe: Un array de productos y un número indeterminados de strings que son etiquetas (tags)..

  Devuelve: Un nuevo array con los productos que tienen al menos todas las etiquetas se se incluyen como parámetro.

Ejemplos usando productos = INVENTARIO:

- etiquetas = “oficina” devuelve [
  {id: 'SKU-002', nombre: 'Mouse Inalámbrico Ergonómico', precio: 25, stock: 50, categoria: 'Accesorios', …},
  {id: 'SKU-004', nombre: 'Monitor UltraWide 34"', precio: 450, stock: 10, categoria: 'Electrónica', …},
  {id: 'SKU-005', nombre: 'Silla de Oficina Premium', precio: 300, stock: 20, categoria: 'Mobiliario', …}]
- etiquetas = “oficina” y “ergonómico” devuelve [
  {id: 'SKU-002', nombre: 'Mouse Inalámbrico Ergonómico', precio: 25, stock: 50, categoria: 'Accesorios', …},
  {id: 'SKU-005', nombre: 'Silla de Oficina Premium', precio: 300, stock: 20, categoria: 'Mobiliario', …}]
- etiquetas = “oficina”, “ergonómico” y “monitor” devuelve [].
- sin etiquetas = devuelve todos los productos.

Ejercicio 3: Renderizado del Catálogo de Productos

Puntuación: **3 puntos**

Descripción:

Crea cada uno de los apartados que se muestran a continuación:

1. Una función llamada ***renderizarCatalogo(contenedor, productos)***. Esta función será la responsable de mostrar los productos en la página web.

   Recibe: Un elemento del DOM (el contenedor) y un array de productos.

Funcionalidad:

- La función debe primero limpiar completamente cualquier contenido existente dentro del contenedor.
- Luego, debe iterar sobre el array de productos. Por cada producto, debe crear dinámicamente una estructura HTML de "tarjeta de producto" y añadirla al contenedor.
- La estructura HTML para cada tarjeta de producto debe ser la siguiente:

```
<article class="producto">
  <h3 class="producto-nombre">[Nombre del Producto]</h3>
  <p class="producto-precio">Precio: $[Precio del Producto]</p>
  <p class="producto-stock">Stock: unidades</p>
</article>
```

2. Una función ***mostrarTodos()*** que llama a la función anterior para que se muestre todos los productos del inventario dado en la división con id = “catalogo”. De esta forma, al pulsar el botón “***Ver todos***” se debería ejecutar.
3. Crea una función ***categoríasDesplegables(selector, categorias)*** que muestre inserta en el campo desplegable (*select)* que se le pasa en ***selector*** las categorías que se pasan como array en ***categorias***.
4. Usando la función anterior inicia desde el principio el campo desplegable que está dentro del div con id=”filtro” con las categorías de los productos del inventario. No se deben repetir las categorías y deben mostrarse ordenadas alfabéticamente.
5. Cuando se elige una categoría en el desplegable y se pulsa el botón “filtrar”, se deben actualizar los productos visibles para que se muestren solo los que pertenecen a dicha categoría.

Ejercicio 4: Modificaciones del DOM

Puntuación: 3 puntos

Descripción:

Dado el html se deben realizar las siguientes **funciones** que leerán o cambiarán el DOM:

1. Función que devuelve un array con todos los títulos de los artículos.
   Para este html debe devolver:
   `['Asistente Virtual con IA en JavaScript', 'Plataforma de E-learning con WebAssembly', 'Red Social Descentralizada con Web3', 'Visualizador de Datos 3D con Three.js']`

   `No debe mostrar 'GraphQL', 'Progressive Web Apps (PWA)'`

2. Función que modifica el estilo de todos los títulos de artículos para que incluyan una clase que se pasa como parámetro. Puedes probar con la clase “elegante” que ya está incluida en el css y mostrará los títulos de los artículos en amarillo.

3. Función que devuelve un array solo con los títulos de los artículos marcados con una clase concreta que se pasa como parámetro. Es decir, de la clase “destacados”.
   Para este html debe devolver:
   `['Plataforma de E-learning con WebAssembly', 'Visualizador de Datos 3D con Three.js']`

4. Función que recibe dos strings, el primero es el nombre del dominio a modificar y el segundo el dominio por el que se sustituye. La función cambia el dominio de todos los enlaces. Puedes probar sustituyendo el dominio “servidor.es” a “dew.org”.

5. Función que cambia el orden en el que se muestran los artículos. Es decir, el primero será el último.

6. Función que elimina los artículos de una clase concreta. Por ejemplo “destacados”.