Javascript

Ejercicio 1:

Tabla de multiplicar personalizadaCrea una función `generarTablaMultiplicar(numero, limite)` que reciba un número y un límite. La función debe devolver un **array** con la tabla de multiplicar del número desde 1 hasta el límite.

Ejercicio 2: Contador de números pares e impares

Escribe una función `contarParesEImpares(arrayNumeros)` que reciba un array de números. La función debe devolver un **objeto** con el total de números **pares** e **impares**.

Ejercicio 3: Validación de contraseñas

Crea una función `validarContrasena(contrasena)` que reciba una cadena y devuelva `true` si cumple los siguientes requisitos:

- La contraseña tiene al menos 8 caracteres.
- Contiene al menos un número.
- Contiene al menos una letra mayúscula.

Ejercicio 4 : Simulador de inventario de tienda

Crea una función `gestionarInventario(inventario, operaciones)` que reciba dos parámetros:

1. **inventario**: un objeto donde las claves son nombres de productos y los valores son las cantidades disponibles.
2. **operaciones**: un array de objetos, donde cada objeto representa una operación con las siguientes propiedades:
    - `producto` (nombre del producto)
    - `tipo` ("añadir" o "retirar")

    - `cantidad` (número de unidades a añadir o retirar)
- `producto` (nombre del producto)
- `tipo` ("añadir" o "retirar")
- `cantidad` (número de unidades a añadir o retirar)

La función debe actualizar el inventario de acuerdo con las operaciones y devolver el inventario final. Si se intenta retirar más unidades de las disponibles, debe mostrar un mensaje de error y no modificar la cantidad de ese producto.

Ejercicio 5: Filtro de palabras

Crea una función `filtrarPalabrasPorLongitud(arrayPalabras, longitudMinima)` que reciba un array de palabras y un número. La función debe devolver un **nuevo array** con las palabras cuya longitud sea mayor o igual al número especificado.

Ejercicio 6: Generador de secuencia Fibonacci

Escribe una función `generarFibonacci(n)` que reciba un número `n` y devuelva un array con los primeros `n` números de la secuencia Fibonacci.

Ejercicio 7: Analizar notas

Crea una función `analizarNotas(notas)` que reciba un array de números representando calificaciones entre 0 y 100. La función debe devolver un objeto con la siguiente información:

- El promedio de las notas.
- La nota más alta.
- La nota más baja.
- El número de estudiantes que aprobaron (nota mayor o igual a 60).

Ejercicio 8: Contador de vocales

Escribe una función `contarVocales(cadena)` que reciba una cadena de texto y devuelva un objeto con el número de veces que aparece cada vocal (`a`, `e`, `i`, `o`, `u`).

Ejercicio 9: Simulador de dados

Crea una función `lanzarDados(cantidadDados)` que simule lanzar varios dados de 6 caras. La función debe devolver un **array** con los resultados de cada dado lanzado. Luego, crea otra función `contarResultados(arrayResultados)` que reciba el array de resultados y devuelva un **objeto** con la frecuencia de cada número (del 1 al 6).

Ejercicio 10: Números ascendentes y descendentes

Crea una función `ordenarNumeros(arrayNumeros, orden)` que reciba un array de números y una cadena `"asc"` o `"desc"`. La función debe devolver el array ordenado de forma ascendente o descendente, según el parámetro `orden`.

Ejercicio 10: Números ascendentes y descendentes

EJERCICIO 11

Hacer un programa que visualice los **10 primeros múltiplos** de un número introducido por teclado. A continuación, debe visualizar la **suma** de todos ellos.

EJERCICIO 12

Realizar un ejercicio que permita realizar la **multiplicación** de dos números pero **sin usar el operador de multiplicación** (prueba técnica).

EJERCICIO 13

Dadas tres variables enteras (`v1` , `v2` y `v3`), escribir condiciones que expresen lo siguiente:

- Todas las variables son cero.
- Todas las variables son positivas.
- Todas las variables tienen el mismo signo.
- Todos sus valores son distintos.
- Dos de sus valores coinciden.
- Como máximo dos de sus valores coinciden.
- El valor de `v2` está comprendido entre los de `v1` y `v3` .

EJERCICIO 14

Leer tres números y escribirlos en **orden decreciente**.

EJERCICIO 15

Definir una función que **analice una cadena de texto** que se le pasa como argumento. La función determina si la cadena está formada por:

- Solo mayúsculas.
- Solo minúsculas.
- Una mezcla de ambas.

EJERCICIO 16

Realizar un programa que me permita crear cualquier **tabla de multiplicar** por pantalla. El programa pedirá un número y calculará su tabla completa (hasta el 10). Ejemplo:

```
5 x 1 = 5
5 x 2 = 10
...
```

EJERCICIO 17

Crear un script que **pinte n cuadrados** de colores alternos. Se debe pedir:

- El número de cuadrados.
- El lado de cada cuadrado.
- El color inicial.Debe colocarlos **en fila** o **en columna**.

EJERCICIO 18

Hacer un programa que pida **10 números** y devuelva la **suma** de solo los números **impares**.

EJERCICIO 19

Pedir un número, pintar una lista no numerada (`<ul>`), donde:

- El **primer elemento** es el número pedido.
- Los siguientes **5 elementos** son el **doble del anterior**.
- No se puede usar `document.write` ni funciones del DOM.

EJERCICIO 21

Dado el siguiente modelo de datos de nóminas:

```
const nominas = [
  { id: 1, nombre: 'Juan', bruto: 30000, irpf: 19, pagas: 1
2 },
  { id: 2, nombre: 'Manu', bruto: 20000, irpf: 15, pagas: 1
4 },
  { id: 3, nombre: 'Ruben', bruto: 22000, irpf: 17, pagas:
16 },
  { id: 4, nombre: 'Danny', bruto: 19000, irpf: 14, pagas:
12 },
];
```

Realizar un programa que calcule y añada:

- Sueldo neto anual: `bruto - (bruto * irpf) / 100`
- Sueldo neto mensual: `netoAnual / pagas`

EJERCICIO 22

Crear un **listado de productos**. Cada producto tiene:

- `name`: String
- `price`: Number
- `stock`: Boolean
- `category`: String (lácteos, carnes, frutas, verduras, pescados).

Acciones a realizar:

1. Crear un array con **10 productos**.

2. Filtrar productos con precio entre **1 y 3 euros**.
3. Filtrar por **categoría**.
4. Filtrar por **disponibilidad de stock**.
5. Filtrar productos entre **1 y 3 euros** que tengan **stock**.

EJERCICIO 23

Hacer una función que **llene un array** con números aleatorios entre **0 y 50**. Pintar el array, pero **excluir las decenas** (10, 20, 30, 40, 50). **No se puede usar** `document.write`.

EJERCICIO 24

Crear un **array de nombres** de diferentes longitudes. Dividir el array en **sub-arrays** donde cada sub-array contenga nombres de la misma longitud.

EJERCICIO 25

Crear una lista de alimentos con la siguiente información:

- `nombreAlimento`
- `categoría`
- `calorías`

Separar los alimentos en dos listas:

- **Saludables**: menos de 15 calorías (pintarlos en verde).
- **No saludables**: 15 calorías o más (pintarlos en rojo).

EJERCICIO 26

Dada la siguiente base de datos:

```
const base_datos = [
  { id: 1, name: "ma+nz*anas", stock: true },
  { id: 2, name: "a((na((car**dos", stock: false },
  { id: 3, name: "a$$$$g**uac&ate", stock: true },
  { id: 4, name: "p$$***iña$", stock: true },
```

```
  { id: 5, name: "c&ar$$n***e", stock: false },
  { id: 6, name: "f*r*ambues%sa$s", stock: true },
  { id: 7, name: "c&e&&re&&eal&es", stock: false },
  { id: 8, name: "q &&**ues***o", stock: true },
  { id: 9, name: "p$oll$o$", stock: false },
  { id: 10, name: "a%%%g**u++ua", stock: true },
  { id: 11, name: "ve*r*du*r*as&&", stock: true }
];
```

Realizar un programa que:

1. Elimine los caracteres especiales de los nombres.
2. Devuelva solo los productos que tengan **stock**.

EJERCICIO 27

Dado un array de **vuelos** con `origen` y `destino`, realizar un programa que:

1. Añada las horas de salida y llegada.
2. Permita completar con al menos **10 vuelos**.
3. Funciones necesarias para:
   - Mostrar todos los vuelos.
   - Mostrar vuelos de un **destino** específico.
   - Mostrar vuelos que **llegan más tarde** que una hora específica.

EJERCICIO 28

Escribir un programa que tome cualquier **frase de texto** y muestre solo los caracteres que **no sean vocales**.

EJERCICIO 29

Dado el siguiente array, devuelve una lista que contenga los valores de la propiedad .name y cambia el nombre a 'Anacleto' en caso de que empiece por 'A'.

```
const users = [{id: 1, name: 'Abel'}, {id:2, name: 'Juli
a'},{id:3, name: 'Pedro'}, {id:4, name: 'Amanda'}];
```

EJERCICIO 30

Dado el siguiente array, haz la media de las notas de todos los examenes

```
const exams = [
    {name: 'Abel Cabeza Román', score: 5},
    {name: 'Maria Aranda Jimenez', score: 1},
    {name: 'Cristóbal Martínez Lorenzo', score: 6},
    {name: 'Mercedez Regrera Brito', score: 7},
    {name: 'Pamela Anderson', score: 3},
    {name: 'Enrique Perez Lijó', score: 6},
    {name: 'Pedro Benitez Pacheco', score: 8},
    {name: 'Ayumi Hamasaki', score: 4},
    {name: 'Robert Kiyosaki', score: 2},
    {name: 'Keanu Reeves', score: 10}
];
```

EJERCICIO 31

Filtrado avanzado de pacientes

Partiendo de un **array de 10 pacientes**, cada uno con los siguientes datos:

- **nombre**: String
- **apellidos**: String
- **edad**: Number
- **enfermedad**: String (diagnóstico de la enfermedad)
- **dni**: String

Desarrolla un programa en **JavaScript** que cumpla con las siguientes funcionalidades:

1. **Filtrar por rango de edades**:
    - El programa debe permitir especificar una **edad mínima** y una **edad máxima** para filtrar a los pacientes dentro de ese rango.

2. **Filtrar por diagnóstico de enfermedad**:
    - Debe ser posible especificar un **diagnóstico** (enfermedad) para filtrar únicamente a los pacientes que coincidan con ese diagnóstico.
3. **Funcionalidad combinada**:
    - El programa debe permitir combinar ambos filtros, es decir, filtrar por **rango de edades** y **diagnóstico de enfermedad** simultáneamente.
4. **Salida de datos**:
    - El programa debe mostrar los pacientes filtrados en un **formato legible**, mostrando **nombre completo**, edad, diagnóstico y DNI.

Ejemplo del array de pacientes:

```
javascript
Copiar código
const pacientes = [
  { nombre: "Juan", apellidos: "Pérez García", edad: 25, en
fermedad: "Gripe", dni: "12345678A" },
  { nombre: "María", apellidos: "López Martínez", edad: 40,
enfermedad: "Diabetes", dni: "87654321B" },
  { nombre: "Ana", apellidos: "González Ruiz", edad: 30, en
fermedad: "Hipertensión", dni: "45678912C" },
  { nombre: "Pedro", apellidos: "Martín López", edad: 50, e
nfermedad: "Gripe", dni: "32165498D" },
  { nombre: "Laura", apellidos: "Hernández Torres", edad: 2
0, enfermedad: "Hipertensión", dni: "74125896E" },
  { nombre: "Carlos", apellidos: "Sánchez Morales", edad: 6
0, enfermedad: "Diabetes", dni: "96385274F" },
  { nombre: "Lucía", apellidos: "Fernández Díaz", edad: 35,
enfermedad: "Asma", dni: "15975368G" },
  { nombre: "David", apellidos: "Ramírez Gómez", edad: 45,
enfermedad: "Hipertensión", dni: "85296314H" },
  { nombre: "Sofía", apellidos: "Vargas Torres", edad: 55,
enfermedad: "Gripe", dni: "74136982I" },
  { nombre: "Marta", apellidos: "Navarro Ruiz", edad: 28, e
nfermedad: "Asma", dni: "36925874J" }
```

```
];
```

Requisitos adicionales:

- La lógica debe estar encapsulada en funciones reutilizables.
- El programa debe permitir ejecutar los filtros de forma separada o combinada.
- La salida debe ser **clara y organizada** (puede imprimirse por consola en formato de tabla o lista).
- deberas pintar el listado de pacientes en el html sin usar document.write usando funciones del DOM

Ejercicio 32: Pintar elementos en diferentes secciones

Crear una array de numeros vacio. Llenar ese array con 50 numeros aleatorios del 1 al 100

Pintar esos numeros dentro de las secciones del html (div) los impares dentro de la seccion impares y los pares dentro de la seccion pares.

Ejercicio 33: Pintar elementos activos

Crear un array vacio de productos.
Crear un funcion que me permita meter productos, id, title, price, quantity, active (boolean) dentro del array
Pintar todos los producto dentro de un section del dom, que tenga html y css, solo los producto que esten activos.