Ejercicios resueltos

Fundamentos Funciones

Índice

Actividades....1
1.- Laboratorio de Drones Autónomos....1
2.- Estación de Energía Hiperión....2
3.- Taller de Prótesis Biónicas....2
4.- Centro de Clones: Control de Lotes....2
5.- Tráfico Orbital: Planificador Asíncrono....2
6.- Bioimpresora 3D: Catálogo de Tejidos....2
7.- Flota de Vehículos Autónomos: Ruta Óptima....2
8.- IA Médica: Triaje de Pacientes Sintéticos....2
9.- Archivo Cuántico: Compresor de Lecturas....3
10.- Mercado NeoTech: Carrito Modular....3

Actividades

1.- Laboratorio de Drones Autónomos

Implementa funciones para un CRUD de un array drones [{id, modelo, batería}]. Crear: addDrone. Leer: listDrones. Actualizar: updateBateria(id, nuevaBateria). Borrar: removeDrone(id). Usa map para obtener solo modelos y reduce para el promedio de batería.

2.- Estación de Energía Hiperión

Escribe funciones flecha para convertir energía entre unidades: julios↔megaJ, kWh↔Wh. Lee valores por argumentos o prompt. Valida números con funciones puras. Devuelve resultados con 2 decimales. Añade una función pipeline(opciones) que encadene dos conversiones.

3.- Taller de Prótesis Biónicas

Crea un array protesis [{id, tipo, precio}]. Implementa funciones calcularIGIC(precio), precioFinal, y un resumen con reduce del coste total. Agrega filtrarPorTipo(tipo) y map para obtener descripciones “ID:tipo → €precioFinal”.

4.- Centro de Clones: Control de Lotes

Usa closures para generar IDs únicos: const nextId = crearGenerador(“CL-”); Cada llamada devuelve “CL-0001”, “CL-0002”… Inserta clones en un array con addClone({id, nombre}). Muestra el listado con listClones().

5.- Tráfico Orbital: Planificador Asíncrono

Simula la secuencia de despegue con callbacks y setTimeout: chequearSistemas → cargarCombustible → acoplarMódulo → cuentaAtras → lanzamiento. Cada paso es una función que recibe callback y registra el evento en un array log. Al final, imprime el log.

6.- Bioimpresora 3D: Catálogo de Tejidos

CRUD sobre tejidos [{id, nombre, viabilidad(0–1)}]. Implementa actualizarViabilidad(id, v). Usa reduce para calcular la viabilidad media y map para devolver etiquetas “OK” si v≥0.7. Añade buscarPorNombre(query) ignorando mayúsculas/acentos.

7.- Flota de Vehículos Autónomos: Ruta Óptima

Dado vehículos [{id, consumoKWh, km}] crea funciones: coste(veh)=consumoKWh*tarifa. Usa reduce para el coste total de la flota y map para coste por km. Implementa filtrarIneficientes(umbral) y una función reporte() que imprime métricas clave.

8.- IA Médica: Triaje de Pacientes Sintéticos

Array pacientes [{id, nombre, riesgo(1–5)}]. Implementa funciones flecha para normalizar nombres, subirRiesgo(id), bajarRiesgo(id). Usa reduce para contar pacientes por nivel de riesgo y map para etiquetas “ALTO/BAJO”. Incluye removePaciente(id).

9.- Archivo Cuántico: Compresor de Lecturas

Escribe funciones puras para: normalizar(secuencia:string) → solo A,C,G,T; chunk(secuencia, n) → array de trozos; frecuencia(trozos) → objeto {trozo:conteo}. Usa map para generar trozos y reduce para la tabla de frecuencias. Imprime la más frecuente.

10.- Mercado NeoTech: Carrito Modular

Array carrito [{id, producto, unidades, precioUnit}]. Implementa addItem, updateUnidades, removeItem. Usa map para obtener subtotales y reduce para total con descuento por tramos (función calcularDescuento(total)). Muestra ticket formateado por consola.