Desarrollo Web en Entornos de Cliente

A02. User Defined Structures

Desarrollo Web en Entorno Cliente

Descripción

Esta hoja de ejercicios está diseñada para que los estudiantes practiquen conceptos de JavaScript, incluyendo colecciones, funciones, objetos definidos por el usuario, y más.

Ejercicio 1: Funciones Predefinidas y Definición de Funciones

Crea una función que reciba dos números como parámetros y devuelve su suma. Investiga y utiliza al menos tres funciones predefinidas del lenguaje (por ejemplo, Math.random(), parseInt(), toFixed()).

Objetivos:

- Practicar la definición y llamada de funciones.
- Familiarizarse con funciones predefinidas.

Ejercicio 2: Función Recursiva

Implementa una función recursiva que reciba un parámetro n y repita n veces el string “bauuuba”.

Objetivos:

- Comprender el concepto de recursión.

Ejercicio 3: Arrays y Operaciones Agregadas

Crea un array que contenga al menos cinco números e implementa operaciones de filtrado, reducción y recolección.

Objetivos:

- Trabajar con arrays y comprender operaciones agregadas.

Ejercicio 4: Matrices y Aplanado

Dada una matriz bidimensional de enteros, crea una función que:

- Aplane la matriz sin utilizar Array.flat().
- Calcule el promedio de los elementos de la matriz.

Objetivos:

- Manipular matrices y realizar operaciones agregadas.

Ejercicio 5: Objetos y Métodos

Define un objeto Libro con propiedades como título, autor, y paginas. Incluye un método resumen() que devuelva una descripción del libro. Crea un array de objetos Libro, recorrelo y utiliza el método en cada uno para mostrar su descripción en la página.

Objetivos:

- Practicar la creación y uso de objetos y métodos.

Ejercicio 6: Filtrado y Transformación de Datos

Crea una función filtrarYTransformar() que reciba un array de números y devuelva un nuevo array que contenga solo los números impares, multiplicados por 2. Utiliza filter() y map().

Objetivos:

- Practicar el uso de métodos de arrays como filter() y map().

- Mejorar la validación de entradas.

Ejercicio 7: Jerarquía de Clases

Define una clase Animal con propiedades como nombre y edad, y un método hablar(). Crea dos subclases: Perro y Gato, que hereden de Animal y sobreescriban el método hablar().

Objetivos:

- Introducir la herencia en la programación orientada a objetos.

Ejercicio 8: Gestión de Datos con Objetos

Crea una clase Estudiante que contenga propiedades como nombre, edad, y notas. Incluye métodos para agregar una nota, calcular el promedio y determinar si ha aprobado. Crea un objeto Estudiante y utiliza sus métodos para realizar pruebas.

Objetivos:

- Manipular objetos y crear métodos de gestión de datos.

Ejercicio 9: Cifrado ROT13

Implementa un script que tome una cadena codificada en ROT13 como entrada y devuelva la cadena decodificada. Los caracteres no alfabéticos deben permanecer sin cambios.

Objetivos:

- Aplicar estructuras de control en la manipulación de cadenas.

Ejercicio 10: Días del Mes

Crea un script que pida al usuario un número entre 1 y 12 (representando los meses del año) y devuelva el número de días en el mes correspondiente. Considera que febrero tiene 28 días y los meses de abril, junio, septiembre y noviembre tienen 30.

Objetivos:

- Practicar la validación de entradas y la lógica de decisiones.

Ejercicio 11: Simulación de Dados

Escribe un script que simule el lanzamiento de 2 dados utilizando Math.random(). Haz 36,000 lanzamientos y muestra cuántas veces ha salido cada resultado.

Objetivos:

- Practicar el uso de funciones aleatorias y bucles.

Ejercicio 12: Números Pares en un Rango

Implementa una función findPairs() que acepte dos números entre 1 y 100 y calcule todos los números pares en ese rango. Incluye validaciones para las entradas y muestra los resultados.

Objetivos:

- Trabajar con validaciones y manipulaciones de arrays.

Ejercicio 13: Reducción de Datos

Implementa una función calcularSumaPrecios() que reciba un array de objetos con propiedades nombre y precio, y devuelva la suma total de todos los precios utilizando reduce().

Objetivos:

- Practicar el uso del método reduce() en arrays.
- Aprender a manejar objetos y propiedades dentro de un array.

Ejercicio 14: Años Bisiestos

Implementa una función isLeapYear() que acepte dos años entre 2001 y 2500 y muestre todos los años bisiestos en ese rango. Incluye validaciones para las entradas.

Objetivos:

- Practicar la lógica condicional y el manejo de datos en arrays.

Ejercicio 15: Depuración y Documentación

Toma uno de los ejercicios anteriores y añade comentarios explicativos en cada parte del código. Realiza pruebas en diferentes escenarios y documenta si el comportamiento es el esperado.

Objetivos:

- Reforzar la importancia de la depuración y la documentación del código.

Ejercicio 16: Juego de Aventura

La empresa "Aventuras Fantásticas S.L." está desarrollando un juego de aventura en el que los jugadores explorarán un mundo lleno de criaturas mágicas. Necesitan crear una estructura de datos para representar a los personajes del juego.

La estructura de datos en esta fase del desarrollo será la de un objeto, en el que se necesitará de cada personaje las siguientes propiedades:

- Nombre
- Raza (por ejemplo, "Humano", "Elfo", "Enano")
- Nivel (comenzará en 1)
- Puntos de Vida (comenzará en 100)

Además, para hacer pruebas, se almacenarán los personajes en un array. Inicialmente habrá 2 personajes precargados en el sistema.

Se deberán crear métodos para gestionar el combate:

- Atacar: Cuando un personaje ataque, se deberá restar una cantidad de puntos de vida al enemigo (por ejemplo, 10 puntos por ataque).
- Recuperar Vida: Se deberá crear un método que permita al personaje recuperar vida. Al recuperar, se sumarán 20 puntos de vida, pero no se podrá exceder el máximo de 100 puntos.
- Subir Nivel: Cada vez que un personaje alcance 0 puntos de vida, se deberá mostrar un alerta indicando que el personaje ha sido derrotado y restablecerlo a su estado inicial (nivel 1 y 100 puntos de vida).

Ejercicio 17: Sistema de Gestión de Inventario

La empresa "Tienda de Fantasía S.L." está desarrollando un sistema de gestión de inventario para sus productos mágicos. Necesitan una estructura de datos para representar cada producto.

Crea una clase Producto con las siguientes propiedades:

- Nombre
- Categoría (por ejemplo, "Poción", "Varita", "Libro")
- Precio
- Cantidad en Stock (comenzará en 0)

Crea un método actualizarStock(cantidad) que permita aumentar o disminuir la cantidad en stock de un producto. Si la cantidad resultante es menor que 0, la cantidad debe establecerse en 0.

Implementa una clase Inventario que contenga un array de objetos Producto. Esta clase debe tener métodos para:

- Agregar un nuevo producto al inventario.
- Eliminar un producto por su nombre.
- Buscar un producto por su nombre y mostrar su información.

Implementa validaciones para asegurarte de que los datos de entrada (como el precio y la cantidad en stock) sean válidos y no provoquen errores inesperados.

Objetivos:

- Practicar el uso de clases y objetos.
- Implementar lógica de gestión de inventario y validaciones.

Ejercicio 18: Sistema de Cálculo de Descuentos

La empresa "Descuentos Mágicos S.L." quiere implementar un sistema para calcular descuentos en productos. El sistema debe permitir calcular el precio final de un producto después de aplicar uno o varios descuentos.

Crea una función calcularPrecioFinal(precioBase, descuentos) que reciba dos parámetros:

- precioBase: un número que representa el precio original del producto.
- descuentos: un array de números que representan los porcentajes de descuento a aplicar.

La función debe aplicar cada descuento secuencialmente sobre el precio base. Por ejemplo, si el precio base es 100 y se aplican descuentos del 10% y 20%, el cálculo debería ser:

- Aplicar el 10%: 100 - (100 * 0.10) = 90
- Aplicar el 20%: 90 - (90 * 0.20) = 72

La función debe validar que el precio base sea un número positivo y que cada descuento esté en el rango de 0 a 100. Si algún descuento no es válido, debe lanzar un error con un mensaje adecuado.

Además, crea una función adicional mostrarPrecioFinal(precioBase, descuentos) que llame a calcularPrecioFinal y muestre el resultado en la consola con un mensaje amigable, por ejemplo: "El precio final después de aplicar los descuentos es: X".

Objetivos:

- Practicar la creación y uso de funciones.
- Mejorar la gestión de arrays y la validación de entradas.

Entrega

En Moodle Centros, abre la tarea llamada “Entrega actividad A02” y entrega tus soluciones con la siguiente estructura y nombres:

- e1-[apellido1][apellido2]-[nombre].js
- e1-[apellido1][apellido2]-[nombre].html

También se puede entregar un repositorio de GitHub con las actividades, asegurándose de que el último commit se realice antes de la fecha de entrega.

Nota sobre validaciones y control de datos de entrada

Es fundamental realizar validaciones y controles de datos de entrada en todas las aplicaciones que desarrollemos. Los datos introducidos por los usuarios pueden variar en tipo y formato, y no siempre son confiables. Implementar validaciones adecuadas ayuda a garantizar que la entrada sea la esperada y a evitar errores en la ejecución del código.

Razones para validar los datos de entrada:

- **Evitar errores**: Los datos incorrectos o inesperados pueden causar fallos en la aplicación. Validar la entrada permite manejar estos casos de manera anticipada.

- **Mejorar la seguridad**: Las aplicaciones que no validan adecuadamente la entrada del usuario son vulnerables a ataques como inyección de código y otros tipos de abusos.
- **Mejorar la experiencia del usuario**: Al proporcionar retroalimentación inmediata sobre la entrada incorrecta, se mejora la usabilidad y la satisfacción del usuario.
- **Asegurar la integridad de los datos**: Validar los datos antes de procesarlos asegura que se mantenga la integridad de la información dentro de la aplicación.

Por lo tanto, es esencial que a lo largo de estos ejercicios, presten especial atención a las validaciones de entrada y se aseguren de que el manejo de los datos se realice de manera segura, eficiente, y que no provoquen errores inesperados.