BIT BUILD INNOVATE TRANSFORM

Taller práctico
Funciones I

JavaScript para principiantes

Instrucciones

Resuelve los siguientes ejercicios utilizando los conceptos aprendidos hasta ahora.
Puedes usar `console.log` para mostrar resultados, `alert` para mensajes, `prompt` para pedir datos al usuario y `confirm` para confirmaciones.

Este taller abordará los temas vistos en clase hasta el momento:

1. Variables, Constantes y tipos de datos
2. Concatenación de Strings
3. Arrays (Listas) y Métodos Básicos
4. Objetos y Acceso a Propiedades
5. Operadores (Aritméticos, Comparación y Lógicos)
6. Condicionales (if, else, ternarios)
7. Bucles: for, while, do-while
8. Y estará especialmente enfocado en funciones.

BIT BUILD INNOVATE TRANSFORM

Funciones Básicas + Listas

Ejercicios

1. Sumar elementos de un array:
   - Crea una función `sumarArray(array)` que reciba un array de números y devuelva la suma.
   - Ejemplo: `sumarArray([1, 2, 3]) → 6`

2. Encontrar el número mayor:
   - Función `numeroMayor(array)` que retorne el número más grande de un array.
   - Ejemplo: `numeroMayor([5, 2, 9, 1]) → 9`.

3. Contar elementos pares:
   - Función `contarPares(array)` que cuente cuántos números pares hay.
   - Ejemplo: `contarPares([3, 8, 2, 10]) → 3`.

Funciones Básicas + Listas

Ejercicios

4. Invertir un array:
   - Función `invertirArray(array)` que retorne un nuevo array invertido.
   - Ejemplo: `invertirArray(["a", "b", "c"]) → ["c", "b", "a"].`

5. Buscar un elemento:
   - Función `buscarElemento(array, elemento)` que devuelva true si el elemento existe.
   - Ejemplo: `buscarElemento(["rojo", "azul"], "azul") → true.`

Funciones + Objetos

Ejercicios

6. Calcular promedio de notas:
   - Función `promedioNotas(estudiante)` que reciba un objeto estudiante con propiedad `notas` (array) y devuelva el promedio.
   - Ejemplo:
     ```
     promedioNotas({ nombre: "Ana", notas: [80, 90, 70] }) → 80
     ```

7. Filtrar estudiantes aprobados:
   - Función `aprobados(listaEstudiantes)` que reciba un **array de objetos** (cada uno con nombre y nota) y devuelva un nuevo array con los que tengan `nota >= 60`.

Funciones + Objetos

Ejercicios

8. Agregar propiedad a objeto:
   - Función `agregarPropiedad(objeto, clave, valor)` que añada una nueva propiedad a un objeto.
   - Ejemplo:
     `agregarPropiedad({ nombre: "Luis" }, "edad", 25) → { nombre: "Luis", edad: 25 }`

9. Combinar dos objetos:
   - Función combinarObjetos(objeto1, objeto2) que una sus propiedades.
   - Ejemplo:
     `combinarObjetos({ a: 1 }, { b: 2 }) → { a: 1, b: 2 }`

Funciones + Objetos

Ejercicios

10. Contar propiedades de un objeto:
    - Función `contarPropiedades(objeto)` que retorne el número de propiedades.
    - Ejemplo: `contarPropiedades({ a: 1, b: 2 }) → 2.`

Aprendamos algo nuevo

Para poder realizar este ejercicio deberás aprender a usar un bucle navito de JavaScript que se conoce como for...in, así que tu tarea es ir a la documentación oficial en el siguiente enlace y averiguar cómo usarlo.

MDN - JavaScript - for...in
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements/for...in

Retos integradores

Ejercicios

11. Calcular área y perímetro de un rectángulo:
    - Función `calcularRectangulo(ancho, alto)` que retorne un objeto con area y perimetro.
    - Ejemplo: `calcularRectangulo(3, 4) → { area: 12, perimetro: 14 }`.

12. Buscar el estudiante con la nota más alta:
    - Función `mejorEstudiante(listaEstudiantes)` que reciba un array de objetos estudiante y devuelva el nombre del que tenga la nota más alta.

Retos integradores

Ejercicios

13. Eliminar elementos duplicados de un array:
    - Función `eliminarDuplicados(array)` que retorne un nuevo array sin duplicados.
    - Ejemplo: `eliminarDuplicados([1, 2, 2, 3]) → [1, 2, 3]`.

Aprendamos algo nuevo

Para poder realizar este ejercicio deberás aprender a usar el operador delete de JavaScript. Así que tu tarea es ir a la documentación oficial en el siguiente enlace y averiguar cómo usarlo.

MDN - JavaScript - Operador delete
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/delete

Retos integradores

Ejercicios

14. Ordenar array de números (ascendente):
    - Función `ordenarArray(array)` que ordene números sin usar **`sort()`**.
    - Pista: Usa el algoritmo **Bubble Sort**.

Aprendamos algo nuevo

Para poder realizar este ejercicio deberás implementar el algoritmo ordenamiento de burbuja, también conocido como “Bubble Sort” Así que tu tarea es aprender cómo funciona e implementarlo.

Tienes todo el conocimiento requerido para lograrlo.

Wikipedia - Ordenamiento de burbuja
https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja

Retos integradores

Ejercicios

15. Validar contraseña:
    - Función `validarContraseña(contraseña)` que verifique si tiene al menos 8 caracteres, un número y una mayúscula. Retorne true o false.

Aprendamos algo nuevo

Para poder realizar este ejercicio deberás recordar el método length ¿Recuerdas que lo hemos usado para saber la longitud de una lista?

```
const frutas = [ "uva", "mora" ];
console.log( frutas.length );  // 2
```

Tambien puedes usarla para contar la longitud de un string, asi:

```
const nombre = "eva sofia";
console.log( nombre.length );  // 9
```

BIT BUILD INNOVATE TRANSFORM

Manipulación Avanzada

Ejercicios

16. Calcular factorial:
    Crea la función `factorial(n)` que calcule el factorial de un número (ej: 5! = 5*4*3*2*1).

17. Generar tabla de multiplicar:
    Crea la función `tablaMultiplicar(n)` que imprima la tabla del **`n`** del `1` al `10`.

18. Contar vocales en un string:
    - Crea la función `contarVocales(texto)` que retorne el número de vocales.
    - Ejemplo: `contarVocales("Hola") → 2.`

19. Calcular impuestos:
    - Crear la función calcularImpuestos(salarios, tasa) que reciba un array de salarios y aplique la tasa de impuesto a cada uno. Retorna un nuevo array.
    - Ejemplo: `calcularImpuestos([1000, 2000], 0.1) → [100, 200]`

Manipulación Avanzada

Ejercicios

20. Simulador de carrito de compras:
    - Función `totalCarrito(carrito)` que reciba un array de objetos `{ producto, precio, cantidad, descuento }` y calcule el total a pagar.

Aprendamos algo nuevo

JS Taller práctico

BIT BUILD INNOVATE TRANSFORM

¿Qué es un módulo?

Un **módulo** en JavaScript es un archivo independiente que contiene código (*funciones*, *objetos*, *variables*) para organizar y reutilizar lógica.

¿Qué es CommonJS?

CommonJS es un sistema de módulos usado principalmente en Vanilla JavaScript y Node.js para organizar código en archivos separados y reutilizarlos.

Conceptos clave:

- `module.exports`: Exporta funciones, objetos o valores desde un archivo.
- `require()`: Importa lo exportado por otro archivo.

Gracias al ECMAScript 2015, existe una forma moderna de usar un sistema de módulos en JavaScript que se llama **ESModule** (ECMAScript Module), pero hablaremos de ella más adelante.

¿Cómo se implementa CommonJS?

1. Creamos un archivo donde definimos: variables, constantes, objetos, listas y funciones que deseemos. En este caso solo las funciones `sumar()` y `restar()` que debemos exportar usando `module.exports`

   Este archivo se llamará `operaciones.js`

```
// Definimos funciones
function sumar(a, b) {
  return a + b;
}

function restar(a, b) {
  return a - b;
}

// Exportamos las funciones (puede ser un objeto, función, variable, etc.)
module.exports = {
  sumar,
  restar
};
```

¿Cómo se implementa CommonJS?

2. Creamos un archivo principal donde vamos a importar: variables, constantes, objetos, listas y funciones que estén definidas en un archivo diferente y que necesitemos usar en este archivo principal. En este caso solo las funciones `sumar()` y `restar()` que debemos exportar usando `module.exports`

   Este archivo se llamará `main.js`

```
// Importamos el módulo
const operaciones = require('./operaciones.js');

// Usamos las funciones importadas
console.log( operaciones.sumar(5, 3) ); //
console.log( operaciones.restar(10, 4) ); // 6
```

¿Cómo funciona CommonJS?

1. Exportación:
   - `module.exports` define qué partes del archivo estarán disponibles para otros.
   - Puedes exportar un objeto (como arriba), una función, o incluso un valor único.

2. Importación:
   - `require('./ruta')` carga el módulo desde otro archivo.
   - La ruta `./` indica que está en la misma carpeta.

Ejercicios adicionales con módulos.

Ejercicios

1. Crea un archivo llamado `funciones-lista.js` donde vas a poner todas las funciones de los 3 primeros ejercicios de este taller:
    1. Sumar elementos de un array
    2. Encontrar el número mayor
    3. Contar elementos pares

    NOTA: Verifica que estás exportando cada uno de ellos de forma correcta

2. Crea un segundo archivo llamado `main.js` donde vas a importar las funciones del archivo `funciones-lista.js` de manera que puedas invocar a las respectivas funciones para que den el resultado esperado para cada una de ellas.

    NOTA: Verifica que estás importando cada una de las funciones del archivo indicado de forma correcta

Tu yo del futuro agradecerá lo que haces en el presente

La programación no es solo entender conceptos, es entrenar la mente para pensar como un desarrollador.

Cada línea de código que escribes, cada error que corriges y cada ejercicio que resuelves, son pasos firmes hacia dominar la lógica que transforma ideas en soluciones.

¡Practica con paciencia, celebra tus avances y nunca dejes de codear!

BIT BUILD INNOVATE TRANSFORM