COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

DESARROLLO *WEB*

FULL STACK

NIVEL BÁSICO

Operadores & Estructura de Control en JavaScript

cymetria | tecnalia colombia

Operadores en JavaScript

Los operadores permiten realizar operaciones matemáticas, lógicas y de asignación en JavaScript. Se pueden clasificar en varios tipos:

1.1. Operadores Aritméticos

Estos operadores se utilizan para realizar cálculos matemáticos.

Operador	Descripción	Ejemplo (`let a = 10, b = 3`)	Resultado
`+`	Suma	`a + b`	`13`
`-`	Resta	`a - b`	`7`
`*`	Multiplicación	`a * b`	`30`
`/`	División	`a / b`	`3.33`
`%`	Módulo (resto de la división)	`a % b`	`1`
`**`	Exponenciación	`a ** b`	`1000`
`++`	Incremento	`a++`	`11`
`--`	Decremento	`a--`	`9`

Operadores en JavaScript

Actividad: "El Restaurante Inteligente" 🍽️💡

📌 **Objetivo:**

Los campistas aplicarán operaciones aritméticas (suma, resta, multiplicación y división) para resolver un problema realista de administración en un restaurante.

📖 **Escenario:**

Eres el gerente de un nuevo restaurante llamado "**Código Gourmet**", que acaba de abrir en la ciudad. Para garantizar su éxito, necesitas calcular los costos de los ingredientes, establecer precios justos y determinar las ganancias.

📝 Instrucciones:

- Formen equipos de 2 campista.
- Con la información dada,

deberán calcular:

- Costo de producción de cada plato (sumando los ingredientes).
- Precio de venta (agregando un 60% de margen de ganancia sobre el costo).
- Ganancias por día (suponiendo que venden 15 platos de cada tipo al día).
- Tiempo para recuperar inversión inicial (dado un costo fijo de apertura de $5,000).
- Primer equipo en calcular todo correctamente, gana una recompensa

- **datos: Plato 1: Pizza Código (Harina: $1.50, Queso: $2.00, Salsa: $1.00, Toppings: $1.50)**

Operadores en JavaScript

1.2. Operadores de Asignación

Permiten asignar valores a variables.

Operador	Ejemplo	Equivalente a
`=`	`x = 10`	`x = 10`
`+=`	`x += 5`	`x = x + 5`
`-=`	`x -= 2`	`x = x - 2`
`*=`	`x *= 3`	`x = x * 3`
`/=`	`x /= 4`	`x = x / 4`
`%=`	`x %= 2`	`x = x % 2`

Actividad: Ahorro Inteligente con Operadores de Asignación

Instrucciones:

Formen equipos de 2 campistas.

recibirán una cuenta de ahorros con:

Un saldo inicial.

Una lista de ingresos y gastos mensuales.

Los resultado deben de ser visto por pantalla.

Usando operadores de asignación, deberán calcular lo siguiente:

```
saldo = 5000;  // Saldo inicial
ingresos = 2000;
gastos = 1500;
```

Paso 1: Calcular el saldo final después de ingresos y gastos

- Utilicen `+=` para **sumar ingresos** al saldo.
- Utilicen `-=` para **restar gastos** del saldo.

Paso 2: Calcular intereses generados

- Apliquen una **tasa de interés mensual del 2%** utilizando `*=`.
- Esto simula el crecimiento del ahorro con intereses.

Paso 3: Retirar un porcentaje de emergencia

- **Retiren el 10%** del saldo total para emergencias utilizando `-=`.

Paso 4: Calcular el tiempo para alcanzar la meta

- La meta de ahorro es **$10,000**.
- Usen `/=` para calcular **cuántos meses** necesitarían si duplicaran su saldo actual cada mes.

Operadores en JavaScript

1.3. Operadores de Comparación

Se utilizan para comparar valores. Devuelven `true` o `false`.

Operador	Descripción	Ejemplo (`let a = 5, b = "5"`)	Resultado
`==`	Igualdad (valor)	`a == b`	`true`
`===`	Igualdad estricta (valor y tipo)	`a === b`	`false`
`!=`	Diferente	`a != b`	`false`
`!==`	Diferente estricto	`a !== b`	`true`
`>`	Mayor que	`a > b`	`false`
`<`	Menor que	`a < b`	`false`
`>=`	Mayor o igual que	`a >= b`	`true`
`<=`	Menor o igual que	`a <= b`	`true`

Operadores en JavaScript

Desafío: "¿Puedo Comprar Esta Casa?"

📌 **Descripción:**

Este ejercicio simula la decisión de comprar una casa en función de **tres factores clave:**

- Tu presupuesto 💰
- El precio de la casa 🏠
- Si cumples con los requisitos de crédito ✅

Usando **operadores de comparación** en JavaScript (`==` , `===` , `!=` , `!==` , `>` , `<` , `>=` , `<=`), los campistas ingresarán sus datos y el código determinará si pueden comprar la casa.

📝 Instrucciones:

1 El asesor ingresará:

- Su presupuesto disponible (número). Input
- El precio de la casa (número). Input
- Si tiene crédito aprobado (true o false). Select
- Valor del Crédito Aprobado (número). Input

2 El programa comparará los valores usando operadores de comparación y dará una respuesta.

- ✅ `>=` → Verifica si el presupuesto es suficiente para la casa
- ✅ `&&` → Comprueba si ambas condiciones son verdaderas (dinero + crédito)
- ✅ `<` → Evalúa si el presupuesto es menor que el precio de la casa
- ✅ `!` → Niega el valor del crédito (`true` o `false`)

Operadores Logicos

Operador	Nombre	Descripción	Ejemplo	Resultado
&&	AND (Y)	Devuelve true si ambas condiciones son verdaderas	true && true	true
	AND (Y)	Devuelve false si al menos una condición es falsa	true && false	false
	AND (Y)		false && false	false
||	OR (O)	Devuelve true si **al menos una** condición es verdadera	true && false	Devuelve true si al menos una condición es verdadera
	OR (O)	Devuelve false solo si todas las condiciones son falsas	false	
!	NOT (NO)	Invierte el valor de una condición	!true	false
			!false	true

📌 Estructura de control

1️⃣ Ejemplo de `if`

🔹 **Caso:** Un usuario ingresa su edad, y si es mayor de edad (18 o más), se le muestra un mensaje.

javascript Copy Edit

```javascript
let edad = parseInt(prompt("Ingresa tu edad:"));

if (edad >= 18) {
    alert("✅ Eres mayor de edad, puedes ingresar.");
}
```

💡 Explicación:

- Solo se ejecuta el código dentro del `if` si la condición es verdadera.
- Si la edad es menor de 18, **no pasa nada**.

📌 Estructura de control

2️⃣ Ejemplo de `if else`

🔹 **Caso:** Validamos si un número ingresado es par o impar.

javascript Copy Edit

```javascript
let numero = parseInt(prompt("Ingresa un número:"));

if (numero % 2 === 0) {
    alert("✅ El número es par.");
} else {
    alert("❌ El número es impar.");
}
```

💡 **Explicación:**

- Si el número es divisible por 2 (`% 2 === 0`), es par.
- Si no, el código dentro del `else` se ejecuta.

📌 Estructura de control

3 Ejemplo de `if else if else`

🔹 **Caso:** Un sistema de calificaciones que evalúa si un estudiante aprueba, saca buena nota o falla.

javascript Copy Edit

```javascript
let calificacion = parseFloat(prompt("Ingresa tu calificación (0-100):"));

if (calificacion >= 90) {
    alert("🎉 ¡Excelente! Sacaste una A.");
} else if (calificacion >= 70) {
    alert("👍 Aprobaste, sigue esforzándote.");
} else {
    alert("❌ Reprobaste, necesitas estudiar más.");
}
```

💡 **Explicación:**

- Primero verifica si la calificación es **90 o más** → "¡Excelente!"
- Si **no cumple la primera condición**, pero es **70 o más** → "Aprobaste".
- Si **ninguna de las anteriores es verdadera**, significa que la calificación es menor a 70 → "Reprobaste".

Actividad

🔷 Actividad: "Acceso Inteligente al Parque de Diversiones"

📌 Objetivo: Utilizar operadores lógicos (`&&` , `||` , `!`) junto con estructuras de control de flujo (`if` , `else if` , `else`) para determinar si una persona puede ingresar a un parque de diversiones según ciertas condiciones.

Instrucciones:

1. Cada estudiante ingresará:
   - Su **edad**
   - Si tiene un **pase VIP** (`true` o `false`)
   - Si viene acompañado por un **adulto** (`true` o `false`)
2. Se evaluarán las condiciones de acceso:
   - Si la persona **tiene pase VIP**, entra automáticamente.
   - Si **tiene 18 años o más**, puede ingresar libremente.
   - Si tiene entre **12 y 17 años**, puede entrar **solo si viene con un adulto.**
   - Si tiene menos de **12 años**, **no puede ingresar** por seguridad.
3. Mostrar un mensaje con el resultado en consola.

Estructura de control Switch

2.2. Switch (alternativa a `if else`)

Es útil cuando hay muchas opciones.

Ejemplo:

javascript Copy Edit

```javascript
let dia = "lunes";

switch (dia) {
    case "lunes":
        console.log("Inicio de semana");
        break;
    case "viernes":
        console.log("Fin de semana pronto");
        break;
    default:
        console.log("Día normal");
}
```

Estructura de control Switch

🔷 Actividad: "Planificador de la Semana" 📅

📌 **Objetivo:** Utilizar `switch` para mostrar un **plan de actividades diarias**, según el día de la semana ingresado por el usuario.

📝 Instrucciones:

1. El usuario ingresará un día de la semana (`lunes`, `martes`, etc.).
2. El programa mostrará las actividades programadas para ese día.
3. Si el usuario ingresa un día inválido, se mostrará un mensaje de error.
4. Se utilizará `switch` para manejar los diferentes casos.

1 Lunes:

- 🧘 Clase de yoga o meditación
- 📋 Planificación de la semana
- 🍏 Inicio de hábitos saludables
- 📚 Leer un libro o artículo interesante
- 💻 Organizar pendientes laborales o académicos

2 Martes:

- 🏋️ Sesión de ejercicio en el gimnasio
- 🎬 Noche de películas o series
- 📖 Estudio o capacitación en línea
- 🍽️ Cena ligera y relajante
- 🚶 Paseo corto para despejar la mente

3 Miércoles:

- 📚 Aprender algo nuevo (cursos, tutoriales, lectura)
- 🍻 Salida con amigos o familia
- 🎨 Actividad creativa (dibujar, escribir, tocar un instrumento)
- 🚴 Caminata o actividad al aire libre
- 📊 Revisión de metas semanales

Operadores en JavaScript

2.3. Bucles (`for` , `while` , `do while`)

Se utilizan para ejecutar repetidamente un bloque de código.

Ciclo `for` **(cuando sabemos cuántas veces iterar)**

javascript Copy Edit

```javascript
for (let i = 0; i < 5; i++) {
    console.log("Iteración número " + i);
}
```

Operadores en JavaScript

Ciclo `while` (cuando no sabemos cuántas veces iterar)

javascript

Copy Edit

```javascript
let i = 0;
while (i < 5) {
    console.log("Número " + i);
    i++;
}
```

Ciclo `do while` (se ejecuta al menos una vez)

javascript

Copy Edit

```javascript
let j = 0;
do {
    console.log("Valor de j: " + j);
    j++;
} while (j < 5);
```

Actividad Dinámica - ¡Adivina el Número Secreto!

Los estudiantes crearán un juego en el que el usuario debe adivinar un número secreto.
Se usará un bucle para repetir la solicitud hasta que el usuario acierte.

1. **Selección de Intentos** 🎲
   - El usuario puede elegir entre **1 y 10 intentos** antes de iniciar el juego.
2. **Generación de Número Secreto** 🎲
   - Se elige un número aleatorio entre **1 y 20** al iniciar el juego.
3. **Ingreso de Número** ⌨️
   - El usuario introduce su número en un **input** y confirma su elección con un botón.
4. **Pistas en Tiempo Real** 🔍
   - Si el número ingresado es incorrecto, el juego indica:
     - **"Muy alto"** si el número es mayor al secreto.
     - **"Muy bajo"** si el número es menor al secreto.
5. **Límite de Intentos** ⏳
   - El usuario tiene la cantidad de intentos seleccionados para adivinar el número.
   - Si se agotan los intentos, el juego finaliza y muestra el número secreto.
6. **Mensaje de Victoria o Derrota** 🏆❌
   - **Si acierta:** Aparece **"¡Felicidades, ganaste!"**.
   - **Si pierde:** Se muestra **"¡Se acabaron los intentos! El número era X"**.
7. **Interfaz Amigable y Dinámica** 🎨
   - Diseño claro y atractivo con botones y mensajes interactivos.
8. **Opción de Reinicio** 🔄
   - Se puede volver a jugar seleccionando una nueva cantidad de intentos y presionando **"Iniciar Juego"**.
9. **Validación de Entrada** ⚠️
   - No permite ingresar números fuera del rango **(1-20)** para evitar errores.

Crear Un numero Aleatorio

```
numeroSecreto = Math.floor(Math.random() * 20) + 1;
```

Try - catch

📌 **Concepto:**

`try - catch` se usa en JavaScript para manejar errores sin detener la ejecución del programa.

- `try` : Se coloca el código que podría generar un error.
- `catch` : Si ocurre un error en `try` , este bloque lo captura y ejecuta el código dentro de él.

```
function dividir(a, b) {
    try {
        if (b === 0) {
            throw new Error("No se puede dividir entre cero."); // Lanzamos un error
        }
        let resultado = a / b;
        console.log(`El resultado de la división es: ${resultado}`);
    } catch (error) {
        console.error(`❌ Error: ${error.message}`);
    } finally {
        console.log("Operación finalizada.");
    }
}

// Pruebas
dividir(10, 2);  // ✅ Resultado: 5
dividir(8, 0);   // ❌ Error: No se puede dividir entre cero.
```

📌 Operadores en JavaScript

🎯 Actividad: Calculadora Sencilla con Manejo de Errores (Try-Catch)

📌 Objetivo:

Crear una calculadora en JavaScript que permita sumar, restar, multiplicar y dividir dos números ingresados por el usuario. La calculadora debe prevenir errores como:

- Ingreso de letras o campos vacíos.
- División por cero.
- Cualquier otro error inesperado.

Para ello, usaremos `try-catch` para manejar las excepciones y mostrar mensajes adecuados al usuario.

🎯 Objetivo de la Actividad:

✅ Aplicar estructuras de control `try-catch` para manejar errores en JavaScript.
✅ Usar `switch` para seleccionar la operación a realizar.
✅ Validar entradas del usuario para evitar errores en cálculos.
✅ Mejorar la experiencia del usuario, evitando bloqueos por errores inesperados.