**Introducción al paradigma de programación**

**¿Qué es una expresión?**

Una expresión es cualquier pedacito de código que produce un valor.

Pensalo como una pregunta que el programa responde con un resultado.

Ejemplos (y su valor):

5 → vale 5

"Hola" → vale "Hola"

2 + 3 → vale 5

"Hola " + "mundo" → vale "Hola mundo"

x > 10 → vale true o false (según x)

usuario.nombre → vale, por ejemplo, "Ana"

tieneHab("dash")(personaje) → vale true/false (es una llamada a función)

Entonces, si lo podes sustituir por su valor sin romper el programa, es una expresión.

**¿Qué es la evaluación?**

Evaluar una expresión es calcular su valor.

Como una máquina expendedora: pones 2 + 3 y la máquina te entrega 5.

La computadora sigue reglas de prioridad (como en matemática: primero \* y /, después + y -). El famoso orden de los operando.

Los paréntesis mandan: (3 + 4) \* 2 se evalúa primero 3 + 4.

Ejemplos paso a paso:

3 + 4 \* 2 → primero 4 \* 2 = 8, luego 3 + 8 = 11.

(3 + 4) \* 2 → primero 3 + 4 = 7, luego 7 \* 2 = 14.

**Expresión vs sentencia (statement)**

Expresión: produce un valor (una respuesta).

Sentencia: hace algo (da una instrucción), puede no producir un valor que nos interese.

Ejemplos:

2 + 3 → **expresión** (da 5).

if (x > 10) { ... } → **sentencia** (control de flujo).

let a = 5; → es una sentencia de declaración, que contiene una expresión (5) y una asignación.

En JavaScript, muchas cosas pueden usarse como expresión: incluso a = 5 devuelve 5. Pero por simplicidad, pensá a la expresión como: **expresión = valor**  / luego **sentencia = acción**.

**Operador condicional ternario**

El operador condicional ternario nos permite poder expresar una condición en su mínima expresión, ejemplo:

If(a > b) {

return a;

}else{

return b;

}

La condición anterior expresada como condicional ternario quedaría así:

**(a>b) ? a : b**

**Expresiones que combinan otras expresiones**

Podés encastrar expresiones como bloques LEGO:

("Hola " + persona.nombre).toUpperCase()

persona.nombre → da un string (expresión)

"Hola " + (ese string) → otro string

.toUpperCase() sobre el resultado → otro string

Todo el conjunto sigue siendo una gran expresión que produce un valor.

**Evaluación con funciones**

Llamar a funciones también produce valores:

sumar(2, 3) // 5

esMayorQue(18)(edad) // true/false (función que devuelve función)

Ojo: una función puede devolver un valor y además tener efectos (ej.: console.log).

En estilo funcional, preferimos que solo devuelva (sin efectos colaterales).

**Cortocircuito (&&, ||, ??)**

JavaScript a veces se detiene temprano, ¿Qué quiero decir con esto?:

A && B → si A es falso, no evalúa B (ya sabemos que todo será falso).

A || B → si A es verdadero, no evalúa B (ya alcanza con A).

A ?? B → si A no es null/undefined, no evalúa B.

Esto importa cuando B llama a funciones (¡podrían tener efectos o ser costosas!).

**Ejercicios:**

**A. Evaluación paso a paso**  
Averiguá el valor final sin correr código:

1. 10 - 2 \* 3 → **?**
2. (10 - 2) \* 3 → **?**
3. ("Hola " + "José").length → **?**
4. true && (3 \* 2) > 5 → **?**

**B. Cortocircuito**  
Evalúa la parte derecha:

* false && carisimo() → ¿se llama a carisimo()?
* true || carisimo() → ¿se llama a carisimo()?

*Te doy las soluciones: A) 4, 24, 8, true y B) no y no.)*