IT Academy

by KIBERNUM

🎯 Inicio de la Clase: Recordando lo Aprendido

Antes de continuar, hagamos una pausa para recordar lo que vimos en la clase anterior.

👉 ¿Qué aprendiste en la clase anterior sobre JavaScript en el navegador y su entorno de ejecución?

💡 **Algunas ideas que pueden mencionar:**

- ¿Dónde se ejecuta JavaScript en un navegador?
- ¿Qué herramientas del navegador usaron?
- ¿Qué cosas puede hacer JavaScript en el navegador?
- ¿Qué limitaciones tiene JavaScript en un entorno web?
- ¿Cómo abrieron la consola de comandos en su navegador?

📌 Responde en una frase corta y comparte tu experiencia. ¡Vamos a repasar juntos! 🚀

🔢 Operaciones y expresiones

En JavaScript, una expresión es cualquier combinación de valores y operadores que produce un resultado.

Por ejemplo:

```
let resultado = 5 + 3;
console.log(resultado); // 8
```

Aquí, **5 + 3** es una **expresión** y el operador **+** hace la operación.

➕ Operadores aritméticos

Son los símbolos que usamos para hacer cálculos matemáticos:

Operador	Descripción	Ejemplo
+	Suma	5 + 2 (7)
-	Resta	5 - 2 (3)
*	Multiplicación	5 * 2 (10)
/	División	10 / 2 (5)
%	Módulo (resto)	10 % 3 (1)
**	Potencia (exponente)	2 ** 3 (8)

```
let suma = 5 + 2; // 7
let resta = 5 - 2; // 3
let multiplicacion = 5 * 2; // 10
let division = 10 / 2; // 5
let modulo = 10 % 3; // 1 (porque 10 dividido por 3 es 3 y sobra 1)
let potencia = 2 ** 3; // 8
```

🎭 Precedencia de operadores

Cuando hay muchas operaciones juntas, JavaScript sigue un orden de prioridad, como en matemáticas (paréntesis primero, luego multiplicación y división, y al final suma y resta).

Ejemplo:

```
let resultado = 5 + 2 * 3;
console.log(resultado); // 11, porque se hace primero 2 * 3 y luego + 5

let resultadoConParentesis = (5 + 2) * 3;
console.log(resultadoConParentesis); // 21, porque primero se hace (5 + 2) y luego * 3
```

Regla simple: Siempre usa paréntesis para asegurarte de que se ejecuta en el orden que quieres. 🎯

⏫ Operadores de incremento y decremento

Son atajos para sumar o restar 1 a una variable.

Operador	Descripción	Ejemplo
++	Incrementa en 1	x++ (x + 1)
--	Decrementa en 1	x-- (x - 1)

Ejemplo:

```
let numero = 5;
numero++; // Aumenta a 6
numero--; // Baja a 5 otra vez
```

También existen versiones antes y después:

```
let x = 5;
console.log(x++); // Imprime 5, luego aumenta a 6
console.log(++x); // Primero aumenta a 7, luego imprime 7
```

📌 Truco: Si el ++ o -- está antes (++x), primero cambia el valor y luego lo usa. Si está después (x++), primero usa el valor y luego lo cambia.

🔍 Operadores de comparación

Sirven para comparar valores. Devuelven true o false.

Operador	Significado	Ejemplo (a = 5, b = 3)
==	Igual (compara valor)	a == 5 (true)
!=	Distinto	a != b (true)
===	Igual (valor y tipo)	5 === "5" (false)
!==	Distinto (valor y tipo)	5 !== "5" (true)
>	Mayor que	a > b (true)
<	Menor que	a < b (false)
>=	Mayor o igual que	a >= 5 (true)
<=	Menor o igual que	b <= 3 (true)

Ejemplo:

```
console.log(5 == "5"); // true (porque solo compara valores)
console.log(5 === "5"); // false (porque los tipos son diferentes)
console.log(10 > 3); // true
console.log(10 < 3); // false
```

📌 **Regla de oro:** Usa **===** en lugar de **==** para evitar problemas con los tipos de datos.

📝 Cadenas de caracteres (Strings)

Una cadena de caracteres (string) es un texto que se escribe entre comillas:

```
let texto1 = "Hola";
let texto2 = 'Mundo';
```

💡 **Reglas:**

- Puedes usar **comillas dobles** " " o **simples** ' '.
- Para mezclar comillas dentro del texto, usa diferentes tipos:

```
let mensaje = "Hoy es 'viernes'";
let mensaje2 = 'El dijo: "Hola"';
```

🔗 Concatenación (unir cadenas)

Puedes unir strings con **+**:

```
let nombre = "Sofia";
let saludo = "Hola, " + nombre + "!";
console.log(saludo); // "Hola, Sofia!"
```

📌 Otra forma mejor **(Template Literals con backticks `)**:

```
let saludo2 = `Hola, ${nombre}!`;
console.log(saludo2); // "Hola, Sofia!"
```

Reto

🎮 Reto : "El Calculador Personal"

Los estudiantes deberán crear un código donde:

- Definan variables para su edad y el año actual.
- Calculen el año en que nacieron y lo impriman.
- Comparen si tienen más de 18 años y muestren true o false.

💡 Objetivos del Reto:

- Los alumnos descubren antes de que se les explique.
- Pueden experimentar sin miedo a equivocarse.
- El aprendizaje es más divertido y significativo.

IT Academy
by KIBERNUM

Expresiones Lógicas en JavaScript

🎯 ¿Qué es una expresión lógica?

En JavaScript, una expresión lógica es cualquier código que se evalúa como true o false.

```
console.log(5 > 3); // true (porque 5 es mayor que 3)
console.log(10 < 2); // false (porque 10 no es menor que 2)
```

📌 Regla simple: Si algo se puede responder con "sí" (true) o "no" (false), es una expresión lógica.

🛠 2. Operadores Lógicos en JavaScript

Operador	Significado	Ejemplo
>	Mayor que	10 > 5 ⟶ true
<	Menor que	5 < 2 ⟶ false
>=	Mayor o igual que	8 >= 8 ⟶ true
<=	Menor o igual que	3 <= 4 ⟶ true
==	Igual (compara valores)	5 == "5" ⟶ true
===	Igual (compara valores y tipo)	5 === "5" ⟶ false
!=	Diferente	10 != 2 ⟶ true
!==	Diferente (considera tipo)	10 !== "10" ⟶ true

📌 Regla de oro: Usa === y !== en vez de == y != para evitar errores.

💡 Ejemplo con === y !==

```
console.log(5 === "5"); // false (porque 5 es número y "5" es string)
console.log(5 == "5");  // true (porque compara solo el valor, no el tipo)
console.log(10 !== "10"); // true (porque uno es número y otro es string)
```

🔄 Operadores Lógicos (&&, ||, !)

Estos operadores combinan condiciones.

Operador	Nombre	Explicación	Ejemplo
&&	Y (AND)	Solo es true si ambas condiciones son true	(5 > 2 && 10 > 5) → true
||	O (OR)	Es true si al menos una de las condiciones es true.	(5 > 2 || 10 < 5) → true
!	NO (NOT)	Invierte el valor	!(5 > 2) → false

💡 Ejemplo práctico con && y ||

```
let edad = 20;
let tieneLicencia = true;

console.log(edad >= 18 && tieneLicencia); // true (es mayor de edad Y tiene licencia)
console.log(edad >= 18 || tieneLicencia); // true (tiene al menos una condición verdadera)
console.log(!(edad >= 18)); // false (porque era true y lo negamos con `!`)
```

IT Academy

by KIBERNUM