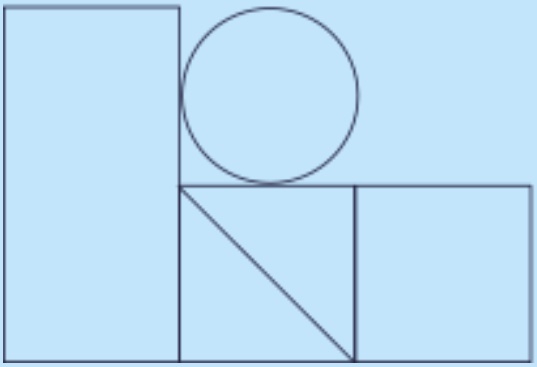
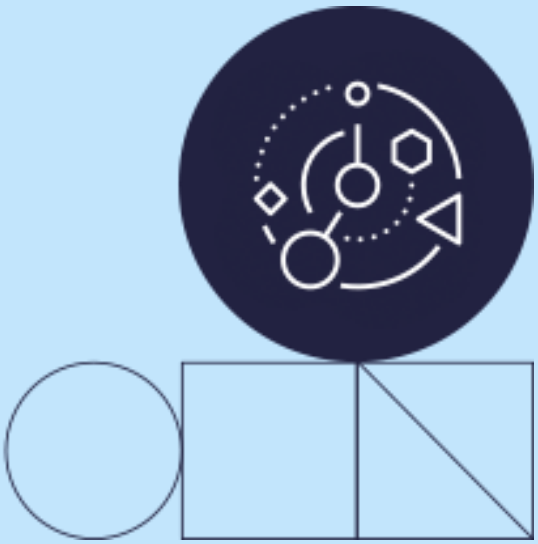
IBM SkillsBuild

IBM

# Conceptos básicos de programación

## Operadores



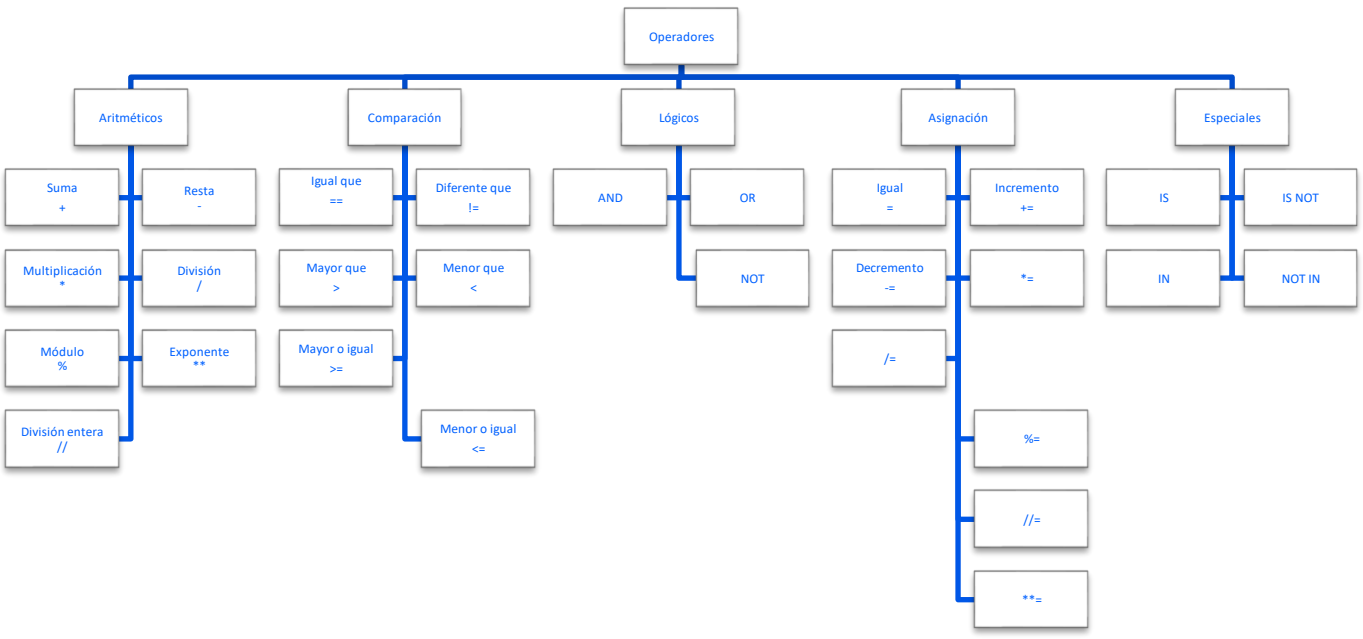


# Índice

# Introducción

Los operadores son fundamentales para ver a las variables hacer que funcione el código ya que con ellos podemos comparar, asignar, realizar funciones matemáticas, etc.

En el siguiente esquema se enumeran los operadores de comunes a casi todos los lenguajes de programación de alto nivel.



# Operadores básicos

Para explicar los operadores nos basaremos en los que contiene el lenguaje de programación Javascript.

En JavaScript disponemos de los operadores aritméticos habituales en lenguajes de programación como son suma, resta, multiplicación, división y operador que devuelve el resto de una división entre enteros (en otros lenguajes denominado operador mod o módulo de una división).

Aunque en otros lenguajes existe un operador de exponenciación para calcular potencias, en JavaScript no es así.

Las operaciones con operadores siguen un **orden de prelación o de precedencia** que determinan el orden con el que se ejecutan. Con los operadores matemáticos la multiplicación y división tienen precedencia sobre la suma y la resta. Si existen expresiones con varios operadores del mismo nivel, la operación se ejecuta de izquierda a derecha. Para evitar resultados no deseados, en casos donde pueda existir duda se recomienda el uso de paréntesis para dejar claro con qué orden deben ejecutarse las operaciones.

# Operadores aritméticos

| Operador | Función | Sintaxis |
|---|---|---|
| **+** | Suma | 2+3 |
| **-** | Resta | 3-2 |
| ***** | Multiplicación | 3*4 |
| **/** | División | 10/5 |
| **%** | Resto de la división | 11%5 valdría: 1 |

**Nota**: El operador “Resto de una división” entre enteros en otros lenguajes se denomina mod.

# Operadores incremento y decremento

| Operador | Función | Sintaxis |
|---|---|---|
| **+=** | X+=Y | X=X+Y |
| **-=** | X-=Y | X=X-Y |
| ***=** | X*=Y | X=X*Y |
| **/=** | X/=Y | X=X/Y |
| **++** | Incremento | X++ Como poner: X=X+1 |
| **--** | Decremento | X-- Como poner: X=X-1 |

**++** y **--** son sólo válidos para variables numéricas y sirven para incrementar una unidad el valor de la variable. Dependiendo de dónde se coloquen (antes o después de la variable) el resultado del cálculo puede diferir debido al momento en que se ejecuta la adición de la unidad.

Los operadores **+=**, **-=** y ***=** son formas abreviadas de escribir operaciones habituales.

Tener en cuenta que **++**, **--**, **+=**, **-=** y ***=** son expresiones que siempre se aplican sobre variables. Por ejemplo, no es válido escribir **2++** porque 2 no es una variable. Todas estas operaciones pueden sustituirse por otra equivalente más evidente. Muchos programadores prefieren no usar estos operadores porque hacen menos legible el código. A otros programadores les gusta usarlos porque les ahorra escribir.

**Importante: Respecto al operador incremento.**

No es lo mismo variable++ que ++variable.

Si ponemos variable++ primero considera el valor que tiene la variable y luego le suma 1.

Si ponemos ++variable primero le suma 1 y luego considera el valor que tiene la variable.

En el siguiente ejemplo tenemos una instrucción que debe imprimir el valor de una variable y luego sumarle 1 al valor de dicha variable. No os preocupéis si no entendéis parte de la sintaxis, son palabras de lenguaje que veremos más adelante:

```
<script>
  var numero = 5;
  document.write(numero++);
  document.write("<br>");
  document.write(numero);
</script>
```

El resultado será que nos imprimirá 5 y luego 6. Es decir, primero imprime el valor de la variable, en este caso 5, luego la añade 1 y en la siguiente impresión la variable vale 6, con lo que imprime 6.

Pero si ponemos:

```
<script>
  var numero = 5;
  document.write(++numero);
  document.write("<br>");
  document.write(numero);
</script>
```

Nos imprimirá 6 y luego 6. Ya que, primero realiza la suma de variable+1, con lo que la variable pasa a valer 6, la imprime y ya se queda con dicho valor. En la siguiente impresión vuelve a imprimir 6.

# Operadores relacionales

Devuelven un booleano con el resultado de la operación que relaciona los operandos.

| Operador | Descripción |
|---|---|
| **x==y** | Igualdad débil |
| **x===y** | Igualdad estricta (sin conversión de tipo). |
| **Object.is** | Método que determina si dos valores son el mismo |
| **x!=y** | Desigualdad débil. |
| **x!==y** | sigualdad estricta (sin conversión de tipo). |
| **x>y** | x es mayor que y? |
| **x<y** | x es menor que y? |
| **x>=y** | x es mayor o igual que y? |
| **x<=y** | x es menor o igual que y? |
| **in** | Determina si un objeto tiene una propiedad dada. |
| **instance of** | Determina si un objeto es instancia de otro. |

En JavaScript disponemos de los operadores habituales en lenguajes de programación como son “es igual”, “es distinto”, menor, menor o igual, mayor, mayor o igual, and (y), or (o) y not (no). La sintaxis se basa en símbolos como veremos a continuación y cabe destacar que hay que prestar atención a no confundir == con = porque implican distintas cosas.

Disponemos además de la asignación tracional basada en =

El operador **=** es el operador de asignación y hay que tener bien claro que no sirve para realizar comparaciones. Para realizar comparaciones ha de usarse **==** (es igual a) ó **===** (es estrictamente igual a). La asignación **a = b** se lee: "asigna a a el contenido de b". Si b es una operación o expresión lógica, a almacenará el valor numérico resultado de la operación o el valor booleano resultado de evaluar la expresión lógica.

Por ejemplo **a = 3 > 5** implicará que a vale *false* porque **3 > 5** es falso.

El operador **===** se interpreta como "es estrictamente igual" y **!==** se interpreta como "no es estrictamente igual". Estos operadores resultan un poco más complejos de comprender por lo que volveremos a hablar de ellos más adelante. De momento tener en cuenta que si una variable contiene **texto1= "1"** y hacemos la comparación:

**texto1 === 1**

Obtendremos *false*, es decir, que no es igual (porque un texto no es igual a un número). Sin embargo, una comparación como **texto == 1** devolverá *true* ya que esta comparación no es estricta y trata de realizar automáticamente conversiones para comprobar si se puede establecer una equivalencia entre los dos valores. En este caso se busca el equivalente numérico del texto y luego se hace la comparación, motivo por el cual se obtiene true.

# Operadores lógicos

| Operador | Descripción |
|---|---|
| **&&** | AND Lógico |
| **\|\|** | OR Lógico |
| **!** | NOT Lógico |

Las expresiones donde se utilizan operadores lógicos y relacionales devuelven un valor booleano, es decir, verdadero (*true*) o falso (*false*).

Por ejemplo, si **a = 7** y **b = 5** la expresión **a < b** devuelve *false* (es falsa). Si **a = true** y **b = false** la expresión **a && b** devuelve *false* (es falsa porque no se cumple que a y b sean verdaderas). Si **a = true** y **b = false** la expresión **a || b** devuelve *true* porque uno de los dos operandos es verdadero. Si **a = true** la expresión **!a** devuelve *false* (el opuesto o contrario).

El operador **||** se obtiene en la mayoría de los teclados pulsando **ALT GR + 1**, es decir, la tecla ALT GR y el número 1 simultáneamente.

Los operadores **&&** y **||** se llaman operadores en cortocircuito porque si no se cumple la condición de un término no se evalúa el resto de la operación. Por ejemplo:

**(a == b && c != d && h >= k)**

Tiene tres evaluaciones: la primera comprueba si la variable **a** es igual a **b**. Si no se cumple esta condición, el resultado de la expresión es falso y no se evalúan las otras dos condiciones posteriores.

En un caso como **( a < b || c != d || h <= k)** se evalúa si **a** es menor que **b**. Si se cumple esta condición el resultado de la expresión es verdadero y no se evalúan las otras dos condiciones posteriores.

El operador **!** se recomienda no usarlo hasta que se tenga una cierta destreza en programación. Una expresión como **(!esVisible)** devuelve *false* si **(esVisible == true)**, o *true* si **(esVisible == false)**.

En general existen expresiones equivalentes que permiten evitar el uso de este operador cuando se desea.