

# Operadores en JavaScript

Las variables por sí solas son de poca utilidad. Hasta ahora, sólo se ha visto cómo crear variables de diferentes tipos y cómo mostrar su valor mediante la función `alert()`. Para hacer programas realmente útiles, son necesarias otro tipo de herramientas.

Los operadores permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables. De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.

## Incremento y Decremento

Estos dos operadores solamente son válidos para las variables numéricas y se utilizan para incrementar o decrementar en una unidad el valor de una variable.

Ejemplo:

```
var numero = 5;

++numero;

alert(numero); // numero = 6
```

El operador de incremento se indica mediante el prefijo `++` en el nombre de la variable. El resultado es que el valor de esa variable se incrementa en una unidad. Por tanto, el anterior ejemplo es equivalente a:

```
var numero = 5;

numero = numero + 1;

alert(numero); // numero = 6
```

De forma equivalente, el operador decremento (indicado como un prefijo `--` en el nombre de la variable) se utiliza para decrementar el valor de la variable:

```
var numero = 5;

-- numero;

alert(numero); // numero = 4
```

El anterior ejemplo es equivalente a:

```
var numero = 5;
```

```
numero = numero - 1;  
  
alert(numero); // numero = 4
```

Los operadores de incremento y decremento no solamente se pueden indicar como prefijo del nombre de la variable, sino que también es posible utilizarlos como sufijo. En este caso, su comportamiento es similar pero muy diferente. En el siguiente ejemplo:

```
var numero = 5;  
  
numero++;  
  
alert(numero); // numero = 6
```

El resultado de ejecutar el script anterior es el mismo que cuando se utiliza el operador ++numero, por lo que puede parecer que es equivalente indicar el operador ++ delante o detrás del identificador de la variable.

Sin embargo, el siguiente ejemplo muestra sus diferencias:

```
var numero1 = 5;  
  
var numero2 = 2;  
  
numero3 = numero1++ + numero2;  
  
// numero3 = 7, numero1 = 6  
  
var numero1 = 5;  
  
var numero2 = 2;  
  
numero3 = ++numero1 + numero2;  
  
// numero3 = 8, numero1 = 6
```

Si el operador ++ se indica como prefijo del identificador de la variable, su valor se incrementa **antes** de realizar cualquier otra operación. Si el operador ++ se indica como sufijo del identificador de la variable, su valor se incrementa **después** de ejecutar la sentencia en la que aparece.

Por tanto, en la instrucción `numero3 = numero1++ + numero2;`, el valor de `numero1` se incrementa después de realizar la operación (primero se suma y `numero3` vale 7, después se incrementa el valor de `numero1` y vale 6). Sin embargo, en la instrucción `numero3 = ++numero1 + numero2;`, en primer lugar se incrementa el valor de `numero1` y después se realiza la suma (primero se incrementa `numero1` y vale 6, después se realiza la suma y `numero3` vale 8).

## Negación (!)

Uno de los operadores lógicos más utilizados es el de la negación. Se utiliza para obtener el valor contrario al valor de la variable:

```
var visible = true;

alert(!visible); // Muestra "false" y no "true"
```

La negación lógica se obtiene prefijando el símbolo `!` al identificador de la variable. El funcionamiento de este operador se resume en la siguiente tabla:

variable ! variable	
true	false
false	true

Si la variable original es de tipo *booleano*, es muy sencillo obtener su negación. Sin embargo, ¿qué sucede cuando la variable es un número o una cadena de texto? Para obtener la negación en este tipo de variables, se realiza en primer lugar su conversión a un valor *booleano*:

- Si la variable contiene un número, se transforma en `false` si vale 0 y en `true` para cualquier otro número (positivo o negativo, decimal o entero).

- Si la variable contiene una cadena de texto, se transforma en `false` si la cadena es vacía (`""`) y en `true` en cualquier otro caso.

```
var cantidad = 0;
```

```
vacio = !cantidad; // vacio = true

cantidad = 2;

vacio = !cantidad; // vacio = false

var mensaje = "";

mensajeVacio = !mensaje; // mensajeVacio = true

mensaje = "Bienvenido";

mensajeVacio = !mensaje; // mensajeVacio = false
```

## AND (&&)

La operación lógica AND obtiene su resultado combinando dos valores booleanos. El operador se indica mediante el símbolo && y su resultado solamente es true si los dos operandos son true:

variable1	variable2	variable1 && variable2
true	true	true
true	false	false
false	true	false
false	false	false

```
var valor1 = true;

var valor2 = false;

resultado = valor1 && valor2; // resultado = false

valor1 = true;

valor2 = true;
```

```
resultado = valor1 && valor2; // resultado = true
```

## OR (||)

La operación lógica OR también combina dos valores booleanos. El operador se indica mediante el símbolo || y su resultado es true si alguno de los dos operandos es true:

variable1	variable2	variable1    variable2
true	true	true
true	false	true
false	true	true
false	false	false

```
var valor1 = true;
```

```
var valor2 = false;
```

```
resultado = valor1 || valor2; // resultado = true
```

```
valor1 = false;
```

```
valor2 = false;
```

```
resultado = valor1 || valor2; // resultado = false
```

## Matemáticos

JavaScript permite realizar manipulaciones matemáticas sobre el valor de las variables numéricas. Los operadores definidos son: suma (+), resta (-), multiplicación (\*) y división (/). Ejemplo:

```
var numero1 = 10;
```

```
var numero2 = 5;
```

```
resultado = numero1 / numero2; // resultado = 2
```

```
resultado = 3 + numero1; // resultado = 13
```

```
resultado = numero2 - 4; // resultado = 1
```

```
resultado = numero1 * numero 2; // resultado = 50
```

Además de los cuatro operadores básicos, JavaScript define otro operador matemático que no es sencillo de entender cuando se estudia por primera vez, pero que es muy útil en algunas ocasiones.

Se trata del operador “*módulo*”, que calcula el resto de la división entera de dos números. Si se divide por ejemplo 10 y 5, la división es exacta y da un resultado de 2. El resto de esa división es 0, por lo que módulo de 10 y 5 es igual a 0.

Sin embargo, si se divide 9 y 5, la división no es exacta, el resultado es 1 y el resto 4, por lo que módulo de 9 y 5 es igual a 4.

El operador módulo en JavaScript se indica mediante el símbolo %, que no debe confundirse con el cálculo del porcentaje:

```
var numero1 = 10;

var numero2 = 5;

resultado = numero1 % numero2; // resultado = 0

numero1 = 9;

numero2 = 5;

resultado = numero1 % numero2; // resultado = 4
```

Los operadores matemáticos también se pueden combinar con el operador de asignación para abreviar su notación:

```
var numero1 = 5;

numero1 += 3; // numero1 = numero1 + 3 = 8

numero1 -= 1; // numero1 = numero1 - 1 = 4

numero1 *= 2; // numero1 = numero1 * 2 = 10

numero1 /= 5; // numero1 = numero1 / 5 = 1

numero1 %= 4; // numero1 = numero1 % 4 = 1
```

## Relacionales

Los operadores relacionales definidos por JavaScript son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=).

Los operadores que relacionan variables son imprescindibles para realizar cualquier aplicación compleja, como se verá en el siguiente capítulo de [programación](#) avanzada. El resultado de todos estos operadores siempre es un valor booleano:

```
var numero1 = 3;

var numero2 = 5;

resultado = numero1 > numero2; // resultado = false

resultado = numero1 < numero2; // resultado = true

numero1 = 5;

numero2 = 5;

resultado = numero1 >= numero2; // resultado = true

resultado = numero1 <= numero2; // resultado = true

resultado = numero1 == numero2; // resultado = true

resultado = numero1 != numero2; // resultado = false
```

Se debe tener especial cuidado con el operador de igualdad (==), ya que es el origen de la mayoría de los errores de [programación](#), incluso para los usuarios que ya tienen cierta experiencia desarrollando scripts. El operador == se utiliza para comparar el valor de dos variables, por lo que es muy diferente del operador =, que se utiliza para asignar un valor a una variable:

```
// El operador "=" asigna valores

var numero1 = 5;

resultado = numero1 = 3; // numero1 = 3 y resultado = 3
```

*// El operador “==” compara variables*

```
var numero1 = 5;
```

```
resultado = numero1 == 3; // numero1 = 5 y resultado = false
```

Los operadores relacionales también se pueden utilizar con variables de tipo cadena de texto:

```
var texto1 = “hola”;
```

```
var texto2 = “hola”;
```

```
var texto3 = “adios”;
```

```
resultado = texto1 == texto3; // resultado = false
```

```
resultado = texto1 != texto2; // resultado = false
```

```
resultado = texto3 >= texto2; // resultado = false
```

Cuando se utilizan cadenas de texto, los operadores “mayor que” (>) y “menor que” (<) siguen un razonamiento no intuitivo: se compara letra a letra comenzando desde la izquierda hasta que se encuentre una diferencia entre las dos cadenas de texto. Para determinar si una letra es mayor o menor que otra, las mayúsculas se consideran menores que las minúsculas y las primeras letras del alfabeto son menores que las últimas (a es menor que b, b es menor que c, A es menor que a, etc.)