Expresiones y operadores JS

Este capítulo describe las expresiones y los operadores de JavaScript, incluyendo los de asignación, comparación, aritméticos, lógicos, ternarios, de cadena y otros.

Operadores

JavaScript tiene los siguientes tipos de operadores. Esta sección describe los operadores y contiene información sobre la precedencia de los mismos.

JavaScript tiene ambos operadores *binarios* y *unarios*, y un operador ternario especial, el operador condicional. Un operador binario requiere dos operandos, uno antes del operando y otro después del operador.

```
operando1 operador operando2
```

Por ejemplo, **3+4** o **x*y**. Un operador unario requiere un solo operando, ya sea antes o después del operador. Por ejemplo, **x++** o **++x**.

```
operador operando
```

o

```
operando operador
```

Operador de asignación

Un operador de asignación asigna un valor a su operando izquierdo basándose en el valor de su operando derecho. El operador de asignación simple es igual (**=**), que asigna el valor de su operando derecho a su operando izquierdo. Es decir, **x = y** asigna el valor de **y** a **x**.

También hay operadores de asignación compuestos que son una abreviatura de las operaciones enumeradas en la siguiente tabla.

Nombre	Operador abreviado	Significado
Asignación	x = y	x = y
Asignación adición	x += y	x = x + y
Asignación resta	x -= y	x = x - y
Asignación multiplicación	x *= y	x = x * y
Asignación división	x /= y	x = x / y
Asignación residuo	x %= y	x = x % y
Asignación AND bit a bit	x &= y	x = x & y

Asignación de desestructuración	let [1,2,3] = num	let one = foo[0];
		let two = foo[1];
		let three = foo[2];

Operadores de comparación

Un operador de comparación compara sus operandos y devuelve un valor lógico en función de si la comparación es verdadera (**true**) o falsa (**false**). Los operandos pueden ser valores numéricos, de cadena, lógicos u objetos.

La siguiente tabla describe los operadores de comparación en términos de este código:

```
let var1 = 3;
let var2 = 4;
```

Operador	Descripción	Retorno de true
Igual (==)	Retorna true si son iguales	3 == var1 ”3” == var1 ”3” == 3
No es igual (!=)	Retorna true si “no” son iguales	var1 != 4 var2 != “3”
Estrictamente igual (===)	Retorna true si coincide en valor y tipo.	3 === var1
Desigualdad estricta (!==)	Devuelve true si estos no coinciden en tipo o valor.	var1 !== “3”
Mayor que (>)	Devuelve true si el operador izquierdo es mayor.	var2 > var1 ”12” > 2
Mayor o igual (>=)	Devuelve true si el operador izquierdo es mayor o igual.	var1 >= 3
Menor que (<)	Devuelve true si el operador izquierdo es menor.	var1 < var2 "2" < 12
Menor o igual (<=)	Devuelve true si el operador izquierdo es menor o igual.	var1 <= var2 var2 <= 5

Operadores aritméticos

Un operador aritmético toma valores numéricos (ya sean literales o variables) como sus operandos y devuelve un solo valor numérico. Los operadores aritméticos estándar son suma (**+**), resta (**-**), multiplicación (*****) y división (**/**). Estos operadores funcionan como lo hacen en la mayoría de los otros lenguajes de programación cuando se usan con números de punto flotante.

```
1 / 2; // 0.5
1 / 2 == 1.0 / 2.0; // Esto es true
```

Además de las operaciones aritméticas estándar (**+**, **-**, ***, **/**), JavaScript proporciona los operadores aritméticos enumerados en la siguiente tabla.

Operador	Descripción	Ejemplo
Residuo (%)	Operador binario. Devuelve el resto entero de dividir los dos operandos.	12%5 = 2
Incremento ++	Operador unario. Agrega uno a su operando. Si se usa como operador prefijo (**++x**), devuelve el valor de su operando después de agregar uno; si se usa como operador sufijo (**x++**), devuelve el valor de su operando antes de agregar uno.	x = 3; x++ x→ vale 4
Decremento - -	Operador unario. Resta uno de su operando. El valor de retorno es análogo al del operador de incremento.	x = 3; x-- x→ vale 2
Negación unaria -	Operador unario. Devuelve la negación de su operando.	x = 3 -x → vale -3
Positivo unario +	Operador unario. Intenta convertir el operando en un número, si aún no lo es.	+"3" retorna 3. +true retorna 1.
Exponenciación **	Calcula la base a la potencia de exponente, es decir, base exponente.	2 ** 3 retorna 8. 10 ** -1 retorna 0.1.

Operadores lógicos

or ||: El operador **or**, representado por **||**, devuelve verdadero si uno de los valores combinados es verdadero.

```
var tengoEfectivo = true;
var tengoTarjeta = false;
var puedoPagar =  tengoEfectivo || tengoTarjeta;
console.log(puedoPagar); // Devuelve true, porque tengo efectivo
```

and &&: El operador **and**, representado por **&&**, devuelve verdadero solo si todos los valores combinados son verdaderos.

```
var tengoCoche = false;
var tengoCarnetDeConducir = true;
var puedoConducir = tengoCoche && tengoCarnetDeConducir;
console.log(puedoConducir); // Devuelve false, porque no tengo c
```

not !: Por último, pero no menos importante, tenemos el operador **not** de negación **!**. Se usa para negar el valor de una expresión (darle el valor opuesto).

```
!true       // => false
!false      // => true
```

Operadores de cadena

Además de los operadores de comparación, que se pueden usar en valores de cadena, el operador de concatenación (+) concatena dos valores de cadena, devolviendo otra cadena que es la unión de los dos operandos de cadena.

```
console.log('mi ' + 'cadena');
// la consola registra la cadena "mi cadena".
```

El operador de asignación abreviada **+=** también se puede utilizar para concatenar cadenas.

```
let mystring = 'alpha';
mystring += 'bet'; // se evalúa como "alphabet" y asigna este va
```

Operador condicional

El **operador condicional** es el único operador de JavaScript que toma tres operandos. El operador puede tener uno de dos valores según una condición. La sintaxis es:

```
condition ?val1 :val2
```

Si **condition** es **true**, el operador tiene el valor de **val1**. De lo contrario, tiene el valor de **val2**. Puedes utilizar el operador condicional en cualquier lugar donde normalmente utilizas un operador estándar.

```
var status = (age >= 18) ? 'adult' : 'minor';
```

Esta declaración asigna el valor "**adult**" a la variable **status** si **age** es de dieciocho años o más. De lo contrario, asigna el valor "**minor**" a **status**.

Operador coma

El **operador coma** (**,**) simplemente evalúa ambos operandos y devuelve el valor del último operando. Este operador se utiliza principalmente dentro de un bucle **for**, para permitir que se actualicen múltiples variables cada vez a través del bucle. Se considera de mal estilo usarlo en otros lugares, cuando no es necesario. A menudo, en su lugar pueden y se deben utilizar dos declaraciones independientes.

Por ejemplo, si **a** es un arreglo bidimensional con 10 elementos en un lado, el siguiente código usa el operador **coma** para actualizar dos variables a la vez. El código imprime los valores de los elementos diagonales en el array.

```
var x = [0,1,2,3,4,5,6,7,8,9]
var a = [x, x, x, x, x];
```

```
for (var i = 0, j = 9; i <= j; i++, j--)
//                                 ^
  console.log('a[' + i + '][' + j + ']= ' + a[i][j]);
```

Operadores unarios

Una operación unaria es una operación con un solo operando. Dentro de ella nos podemos encontrar delete, typeof o void.

delete : El operador **delete** elimina la propiedad de un objeto.

```
delete object.property;
delete object[propertyKey];
delete objectName[index];
delete property; // legal solo dentro de una declaración with
```

Donde **object** es el nombre de un objeto, **property** es una propiedad existente y **propertyKey** es una cadena o símbolo que hace referencia a una propiedad existente.

La cuarta forma es legal solo dentro de una declaración **with**, para eliminar una propiedad de un objeto, y también para las propiedades del objeto global.

Si el operador **delete** tiene éxito, elimina la propiedad del objeto. Intentar acceder a él después dará como resultado **undefined**. El operador **delete** devuelve **true** si la operación es posible; devuelve **false** si la operación no es posible.

```
x = 42; // implícitamente crea window.x
var y = 43;
var myobj = {h: 4}; // crea un objeto con la propiedad h
```

```
delete x;       // devuelve true (se puede eliminar si se crea
delete y;       // devuelve false (no se puede borrar si se dec
delete Math.PI; // devuelve false (no se pueden eliminar propie
delete myobj.h; // devuelve true (puede eliminar propiedades de
```

typeof: El operador **typeof** devuelve los siguientes resultados para estas variables.

```
var myFun = new Function('5 + 2');
var shape = 'round';
var size = 1;
var foo = ['Apple', 'Mango', 'Orange'];
var today = new Date();

typeof myFun;       // devuelve "function"
typeof shape;       // devuelve "string"
typeof size;        // devuelve "number"
typeof foo;         // devuelve "object"
typeof today;       // devuelve "object"
typeof doesntExist; // devuelve "undefined"

typeof true; // devuelve "boolean"
typeof null; // devuelve "object"
```

void: El operador void es una evaluación sin retorno

```
void (expression)
void expression
```

El operador **void** especifica una expresión que se evaluará sin devolver un valor. **expression** es una expresión de JavaScript para evaluar. Los paréntesis que

rodean la expresión son opcionales, pero es un buen estilo usarlos.

Operadores relacionales

operador in: devuelve **true** si la propiedad especificada está en el objeto especificado.

```
propNameOrNumber in objectName
```

Donde **propNameOrNumber** es una expresión de cadena, numérica o de símbolo que representa un nombre de propiedad o índice de arreglo, y **objectName** es el nombre de un objeto.

```
// Arrays
var trees = ['redwood', 'bay', 'cedar', 'oak', 'maple'];
0 in trees;        // devuelve true
3 in trees;        // devuelve true
6 in trees;        // devuelve false
'bay' in trees;    // devuelve false (debes especificar el núme
                   // no el valor en ese índice)
'length' en trees; // devuelve true (la longitud es una propied

// objetos integrados
'PI' in Math;          // devuelve true
var myString = new String('coral');
'length' in myString;  // devuelve true

// Objetos personalizados
var mycar = { make: 'Honda', model: 'Accord', year: 1998 };
'make' in mycar;  // devuelve true
'model' in mycar; // devuelve true
```

instanceof: El **operador instanceof** devuelve **true** si el objeto especificado es del tipo de objeto especificado.

Utiliza **instanceof** cuando necesites confirmar el tipo de un objeto en tiempo de ejecución. Por ejemplo, al detectar excepciones, puedes ramificar a diferentes controladores según el tipo de excepción lanzada.

Por ejemplo, el siguiente código usa **instanceof** para determinar si **theDay** es un objeto **Date**. Debido a que **theDay** es un objeto **Date**, las instrucciones de la expresión **if** se ejecutan.

```
var theDay = new Date(1995, 12, 17);
if (theDay instanceof Date) {
  // instrucciones a ejecutar
}
}
```

Operador de agrupación

El operador de agrupación **( )** controla la precedencia de la evaluación en las expresiones. Por ejemplo, puedes redefinir la multiplicación y la división primero, luego la suma y la resta para evaluar la suma primero.

```
let a = 1;
let b = 2;
let c = 3;

// precedencia predeterminada
a + b * c     // 7
// evaluado por omisión así
a + (b * c)   // 7

// ahora prevalece sobre la precedencia
```

```
// suma antes de multiplicar
(a + b) * c   // 9

// que es equivalente a
a * c + b * c // 9
```