Precedencia de los operadores en JavaScript

La precedencia define el orden en el cual los operadores son evaluados uno respecto del otro, o en palabras más simples, la precedencia de un operador sobre otro determina qué operación se realiza en primer lugar. De esta manera, los operadores con mayor precedencia se convierten en los operandos de los operadores con menor precedencia. Esto es importante, debido a que en un principio todos los operadores se evalúan de izquierda a derecha, pero existen ciertas normas que determinan que algunos operadores se evalúen antes que otros.

Por ejemplo, en esta operación let resultado = 2 + 3 * 2 sabemos que la multiplicación debe calcularse antes que la suma, por tanto, la multiplicación tiene una precedencia más alta que la suma. Sin embargo, los paréntesis tienen la más alta prioridad, esto quiere decir que una expresión colocada entre paréntesis se evalúa primero. Así, en esta expresión let resultado = (2 + 3) * 2 se evalúa primero la suma y el resultado se multiplica por 2. Por tanto, podemos utilizar paréntesis para alterar el orden de evaluación y asegurarnos de que las expresiones se evalúen en el orden deseado.

Entre los operadores matemáticos, el operador de exponenciación tiene la segunda mayor precedencia y se evalúa después de los paréntesis (los operadores lógicos, de incremento y decremento, y otros, tienen una mayor precedencia que el operador de exponenciación y se sitúan debajo de los paréntesis, tal y como indica la tabla de abajo).

Los operadores de multiplicación, división y módulo o resto tienen la tercera mayor precedencia y se evalúan de izquierda a derecha.

Por su parte, los operadores de adición y sustracción tienen la menor precedencia y también se evalúan de izquierda a derecha.

Precedencia y Asociatividad

La asociatividad define el orden en el que un operador evalúa sus operandos. Cuando todos los operadores en una expresión son de la misma precedencia, normalmente la asociación es de izquierda a derecha, por ejemplo, en la expresión 55 + 45 + 35, la evaluación es de izquierda a derecha.

Cuando los operadores tienen diferentes niveles de precedencia, entonces el operador con la precedencia más alta va primero y la asociatividad no tiene importancia. Por ejemplo, en esta operación 2 + 3 * 2 la multiplicación tiene mayor precedencia que la suma y se ejecuta primero, a pesar de que la suma se escribe primero en la operación.

La siguiente tabla muestra un listado de los operadores de JavaScript en orden de mayor precedencia a menor precedencia, en donde 19 representa la mayor precedencia y 1 la menor precedencia:

Precedencia	Tipo de operador	Asociatividad	Operadores individuales
19	Agrupamiento	n/a	`( … )`
18	Acceso a propiedades (notación por punto)	a la izquierda	`… . …`
	Acceso a propiedades (notación por corchetes)	a la izquierda	`… [ … ]`
	`new` (con lista de argumentos)	n/a	`new … ( … )`
	Llamada a función	a la izquierda	`… ( … )`
	Encadenamiento opcional	a la izquierda	`?.`
17	`new` (sin lista de argumentos)	a la derecha	`new …`
16	Incremento sufijo	n/a	`… ++`

	Decremento sufijo		… `--`
15	NOT lógico (!)	a la derecha	`!` …
	NOT a nivel de bits (~)		`~` …
	Suma unaria (+)		`+` …
	Negación unaria (-)		`-` …
	Incremento prefijo		`++` …
	Decremento prefijo		`--` …
	`typeof`		`typeof` …
	`void`		`void` …
	`delete`		`delete` …
	`await`		`await` …
14	Potenciación (**)	a la derecha	… `**` …
13	Multiplicación (*)	a la izquierda	… `*` …
	División (/)		… `/` …
	Resto (%)		… `%` …
12	Adición (+)	a la izquierda	… `+` …
	Sustracción (-)		… `-` …
11	Desplazamiento de bits a la izquierda (<<)	a la izquierda	… `<<` …
	Desplazamiento de bits a la derecha (>>)		… `>>` …
	Desplazamiento de bits a la derecha sin signo (>>>) (en-US)		… `>>>` …
10	Menor a (<)	a la izquierda	… `<` …
	Menor o igual a (<=)		… `<=` …
	Mayor a (>)		… `>` …
	Mayor o igual a (>=)		… `>=` …
	`in`		… `in` …
	`instanceof`		… `instanceof` …
9	Igualdad (==)	a la izquierda	… `==` …
	Desigualdad (!=)		… `!=` …
	Igualdad estricta (===)		… `===` …
	Desigualdad estricta (!==)		… `!==` …
8	AND a nivel de bits (&)	a la izquierda	… `&` …

7	XOR a nivel de bits (^)	a la izquierda	… ^ …
6	OR a nivel de bits (|)	a la izquierda	… | …
5	AND lógico (&&)	a la izquierda	… && …
4	OR lógico (||)	a la izquierda	… || …
	Operador de coalescencia nula (??)	a la izquierda	… ?? …
3	Operador condicional (ternario)	a la derecha	… ? … : …
2	Asignación	a la derecha	… = …
			… += …
			… -= …
			… **= …
			… *= …
			… /= …
			… %= …
			… <<= …
			… >>= …
			… >>>= …
			… &= …
			… ^= …
			… |= …
			… &&= …
			… ||= …
			… ??= …
	yield	a la derecha	yield …
	yield*		yield* …
1	Operador coma	a la izquierda	… , …