Precedence	Associativity	Individual operators	Notes
18: grouping	n/a	Grouping (x)	[1]
17: access and call	left-to-right	Member access x.y	
		Optional chaining x?.y	[2]
		Computed member access x[y]	[3]
		new with argument list new x(y)	
		Function call x(y)	
	n/a	import(x)	[4]
16: new	n/a	new without argument list new x	
15: postfix operators	n/a	Postfix increment x++	
		Postfix decrement x--	[5]
14: prefix operators	n/a	Prefix increment ++x	
		Prefix decrement --x	[6]
		Logical NOT !x	
		Bitwise NOT ~x	
		Unary plus +x	
		Unary negation -x	
		typeof x	
		void x	
		delete x	[7]
		await x	
13: exponentiation	right-to-left	Exponentiation x ** y	[8]
12: multiplicative operators	left-to-right	Multiplication x * y	
		Division x / y	
		Remainder x % y	
11: additive operators	left-to-right	Addition x + y	
		Subtraction x - y	
10: bitwise shift	left-to-right	Left shift x << y	
		Right shift x >> y	
		Unsigned right shift x >>> y	
9: relational operators	left-to-right	Less than x < y	
		Less than or equal x <= y	
		Greater than x > y	
		Greater than or equal x >= y	
		x in y	
		x instanceof y	
8: equality operators	left-to-right	Equality x == y	
		Inequality x != y	
		Strict equality x === y	
		Strict inequality x !== y	
7: bitwise AND	left-to-right	Bitwise AND x & y	
6: bitwise XOR	left-to-right	Bitwise XOR x ^ y	
5: bitwise OR	left-to-right	Bitwise OR x | y	

4: logical AND	left-to-right	Logical AND x && y	
3: logical OR, nullish coalescing	left-to-right	Logical OR x || y	
		Nullish coalescing operator x ?? y	[9]
	right-to-left	Assignment x = y	
		Addition assignment x += y	
		Subtraction assignment x -= y	
		Exponentiation assignment x **= y	
		Multiplication assignment x *= y	
		Division assignment x /= y	
		Remainder assignment x %= y	
		Left shift assignment x <<= y	
		Right shift assignment x >>= y	
		Unsigned right shift assignment x >>>= y	
		Bitwise AND assignment x &= y	
		Bitwise XOR assignment x ^= y	
		Bitwise OR assignment x |= y	
		Logical AND assignment x &&= y	
		Logical OR assignment x ||= y	
		Nullish coalescing assignment x ??= y	[10]
	right-to-left	Conditional (ternary) operator x ? y : z	[11]
	right-to-left	Arrow x ⇒ y	[12]
		yield x	
		yield* x	
2: assignment and miscellaneous	n/a	Spread ...x	[13]
1: comma	left-to-right	Comma operator x, y	

Notes:

The operand can be any expression.
The "right-hand side" must be an identifier.
The "right-hand side" can be any expression.
The "right-hand side" is a comma-separated list of any expression with precedence > 1 (i.e. not comma expressions).
The operand must be a valid assignment target (identifier or property access). Its precedence means new Foo++ is (new Foo)++ (a syntax error) and not new (Foo++) (a TypeError: (Foo++) is not a constructor).
The operand must be a valid assignment target (identifier or property access).
The operand cannot be an identifier or a private property access.
The left-hand side cannot have precedence 14.
The operands cannot be a logical OR || or logical AND && operator without grouping.
The "left-hand side" must be a valid assignment target (identifier or property access).
The associativity means the two expressions after ? are implicitly grouped.
The "left-hand side" is a single identifier or a parenthesized parameter list.
Only valid inside object literals, array literals, or argument lists.