REF #1 Operator Precendence List

Value	Operator	Description	Example
18	`( )`	Expression Grouping	`(100 + 50) * 3`
17	`.`	Member Of	`person.name`
17	`[]`	Member Of	`person["name"]`
17	`?.`	Optional Chaining ECMAScript 2020	`x ?. y`
17	`()`	Function Call	`myFunction()`
17	`new`	New with Arguments	`new Date("October 13, 2022")`
16	`new`	New without Arguments	`new Date()`
15	`++`	Postfix Increment	`i++`
15	`--`	Postfix Decrement	`i--`
14	`++`	Prefix Increment	`++i`
14	`--`	Prefix Decrement	`--i`
14	`!`	Logical NOT	`!(x==y)`
14	`~`	Bitwise NOT	`~x`
14	`+`	Unary Plus	`+x`
14	`-`	Unary Minus	`-x`
14	`typeof`	Data Type	`typeof x`
14	`void`	Evaluate Void	`void(0)`
14	`delete`	Property Delete	`delete myCar.color`
13	`**`	Exponentiation ECMAScript 2016	`10 ** 2`
12	`*`	Multiplication	`10 * 5`
12	`/`	Division	`10 / 5`
12	`%`	Division Remainder	`10 % 5`
11	`+`	Addition	`10 + 5`

11	`-`	Subtraction	`10 - 5`
11	`+`	Concatenation	`"John" + "Doe"`
10	`<<`	Shift Left	`x << 2`
10	`>>`	Shift Right (signed)	`x >> 2`
10	`>>>`	Shift Right (unsigned)	`x >>> 2`
9	`in`	Property in Object	`"PI" in Math`
9	`instanceof`	Instance of Object	`x instanceof Array`
9	`<`	Less than	`x < y`
9	`<=`	Less than or equal	`x <= y`
9	`>`	Greater than	`x > y`
9	`>=`	Greater than or equal	`x >= Array`
8	`==`	Equal	`x == y`
8	`===`	Strict equal	`x === y`
8	`!=`	Unequal	`x != y`
8	`!==`	Strict unequal	`x !== y`
7	`&`	Bitwise AND	`x & y`
6	`^`	Bitwise XOR	`x ^ y`
5	`|`	Bitwise OR	`x | y`
4	`&&`	Logical AND	`x && y`
3	`||`	Logical OR	`x || y`
3	`??`	Nullish Coalescing ECMAScript 2020	`x ?? y`
2	`? :`	Condition	`? "yes" : "no"`
2	`=`	Simple Assignment	`x += y`
2	`+=`	Addition Assignment	`x += y`
2	`-=`	Subtraction Assignment	`x -= y`
2	`*=`	Multiplication Assignment	`x *= y`
2	`**=`	Exponentiation Assignment	`x **= y`
2	`/=`	Division Assignment	`x /= y`

2	`%=`	Remainder Assignment	`x %= y`
2	`<<=`	Left Shift Assignment	`x <<= y`
2	`>>=`	Right Shift Assignment	`x >>= y`
2	`>>>=`	Unsigned Right Shift	`x >>>= y`
2	`&=`	Bitwise AND Assignment	`x &= y`
2	`|=`	Bitwise OR Assignment	`x |= y`
2	`^=`	Bitwise XOR Assignment	`x ^= y`
2	`&&=`	Logical AND Assignment	`x &= y`
2	`||=`	Logical OR Assignment	`x ||= y`
2	`=>`	Arrow	`x => y`
2	`yield`	Pause / Resume	`yield x`
2	`yield*`	Delegate	`yield* x`
2	`...`	Spread	`... x`
1	`,`	Comma	`x , y`