

## **Operator Precedence**

**Operator precedence** determines the order in which the operators in an expression are evaluated.

For eg -

int 
$$x = 3 * 4 - 1$$
;

In the above example, the value of x will be 11, not 9. This happens because the precedence of \* operator is higher than - operator. That is why the expression is evaluated as (3 \* 4) - 1 and not 3 \* (4 - 1).

## **Operator Precedence Table**

Operators	Precedence	
postfix increment and decrement	++	
prefix increment and decrement, and unary	++ + - ~ !	
multiplicative	* / %	
additive	+-	
shift	<< >> >>>	
relational	< > <= >= instanceof	
equality	== !=	
bitwise AND	&	
bitwise exclusive OR	^	
bitwise inclusive OR		
logical AND	&&	
logical OR		
ternary	?:	
assignment	=   =   *=   /=   %=   &=   =   =	

**Associativity of Operators** 



If an expression has two operators with similar precedence, the expression is evaluated according to its **associativity** (either left to right, or right to left).

Operators	Precedence	Associativity
postfix increment and decrement	++	left to right
prefix increment and decrement, and unary	++ + - ~ !	right to left
multiplicative	* / %	left to right
additive	+ -	left to right
shift	<< >>>>	left to right
relational	< > <= >= instanceof	left to right
equality	== !=	left to right
bitwise AND	&	left to right
bitwise exclusive OR	^	left to right
bitwise inclusive OR		left to right
logical AND	&&	left to right
logical OR		left to right
ternary	?:	right to left
assignment	= \( += \( -= \( *= \) /= \\ \%= \( &= \) \( ^= \) \(  = \) \( <<= \) \( >>= \) \( >>= \) \( >>= \)	right to left

**Note -** These notes are just for a quick glance. We don't have to memorize them all at once. Most of these rules are very logical and we have been following them in a lot of instances already.