

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

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 | <code>++</code> <code>--</code> | left to right |
| prefix increment and decrement, and unary | <code>++</code> <code>--</code> <code>+</code> <code>-</code> <code>~</code> <code>!</code> | right to left |
| multiplicative | <code>*</code> <code>/</code> <code>%</code> | left to right |
| additive | <code>+</code> <code>-</code> | left to right |
| shift | <code><<</code> <code>>></code> <code>>>></code> | left to right |
| relational | <code><</code> <code>></code> <code><=</code> <code>>=</code> <code>instanceof</code> | left to right |
| equality | <code>==</code> <code>!=</code> | left to right |
| bitwise AND | <code>&</code> | left to right |
| bitwise exclusive OR | <code>^</code> | left to right |
| bitwise inclusive OR | <code> </code> | left to right |
| logical AND | <code>&&</code> | left to right |
| logical OR | <code> </code> | left to right |
| ternary | <code>?:</code> | right to left |
| assignment | <code>=</code> <code>+=</code> <code>-=</code> <code>*=</code> <code>/=</code> <code>%=</code> <code>&=</code> <code>^=</code> <code> =</code> <code><<=</code> <code>>>=</code> <code>>>>=</code> | 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.