

PHP Programming

COMP203TH

Lecture: 7

PHP Operator Precedence and Associativity

E.g. in maths, you must have learned about BODMAS → that specifies the order in which a calculator or a computer performs a sequence of mathematical operations: Brackets, Order, Division, Multiplication, Addition and Subtraction.

Similarly, PHP follows a similar set of rules when determining which operators have precedence over others.

The **precedence of an operator** specifies how tightly it binds two expressions together.

e.g. in the expression $1 + 5 * 3$, the answer is 16 and not 18, because the multiplication (*) operator has a higher precedence than the addition (+) operator.

When **operators have equal precedence** their **associativity** decides how the operations are grouped.

e.g.

- “-” is left-associative, so $1 - 2 - 3$ is grouped as $(1 - 2) - 3$ and evaluates to -4.
- “=” is right-associative, so $\$a = \$b = \$c$ is grouped as $\$a = (\$b = \$c)$.

Precedence and associativity are two characteristics of operators that determine the evaluation order of sub-expressions in absence of brackets.

Parentheses always have the highest precedence, so wrapping an expression in parentheses will force PHP to evaluate it first.

Below table lists the operators in order of precedence with the highest-precedence ones at the top. Operators on the same line have equal precedence, in which case associativity decides grouping.

Associativity	Operator	Remarks
Right	**	exponentiation
n/a	++ --	increment/decrement
n/a	!	logical operator
left	*/ %	arithmetic
left	+ - .	arithmetic and string
left	<< >>	bitwise
non-associative	< <= > >==	comparison
left	&	bitwise
left	^	bitwise
left	 	bitwise
left	&&	logical
left	 	logical
left	?:	ternary
right	= += -=	all assignment operators
left	and	logical
left	xor	logical
left	or	logical

Note: in the above table observe the precedence level of “and” and “or”, “&&” and “||”.