: The Swift Programming Language 5.7 : Quick Reference Guide : **Basic Operators** :

Terminology

UNARY OPERATORS operate on a single target.
UNARY PREFIX OPERATORS
appear immediately before their target.
UNARY POSTFIX OPERATORS
appear immediately after their target.
BINARY INFIX OPERATORS
appear between their two targets.
(a ? b : c) is Swift's
only TERNARY CONDITIONAL OPERATOR.
OPERANDS are values affected by operators.

Assignment Operator

```
a = b // Updates the value of a with the value of b.
let (x, y) = (1, 2) // 1 is assigned to x,
and 2 is assigned to y in this tuple.
if x = y { CODE } // Returns an error because = does not
return a value.
```

Arithmetic Operators

(All number types are supported.)

```
1 + 2      // Equals 3.
5 - 3      // Equals 2.
2 * 3      // Equals 6.
10.0 / 2.5 // Equals 4.0.
"hello, " + "world" // Performs String concatenation.
```

Remainder Operator

(Also Known As : Modulo.)

```
9 % 4 // Equals 1.
a % b // The sign of a is used.
```
The sign of b is ignored.

Unary Minus Operator

```
let three = 3
let minusThree = -three // Equals -3.
```

Unary Plus Operator

```
let minusSix = -6
let alsoMinusSix = +minusSix // Equals -6.
```

Compound Assignment Operators

```
a += 2 // Is shorthand for a = a + 2.
```
Does not return a value.

Comparison Operators

(Returns a Bool value: true or false)

```
a == b // Is a equal to b?
a != b // Is a not equal to b?
a > b  // Is a greater than b?
a < b  // Is a less than b?
a >= b // Is a greater than or equal to b?
a <= b // Is a less than or equal to b?

objectReferenceA === objectReferenceB
// Refers to the same object instance?
objectReferenceA !== objectReferenceB
// Does not refer to the same object instance?
```

Commonly used by if.
Works on tuples of the same type and number of values
up to seven elements.

Ternary Conditional Operator

```
(question ? answer1 : answer2) // Is shorthand for
"if question {answer1} else {answer2}".
```

Nil-Coalescing Operator

```
(a ?? b) // Is shorthand for "a != nil ? a! : b"
```

NOTE: a! // Force unwrap optional a

Range Operators

Closed Range Operator

```
1...5 // 1, 2, 3, 4, 5
```

Half-Open Range Operator

```
1..<5 // 1, 2, 3, 4 (Useful with arrays.)
```

One-Sided Ranges

```
2... // From index 2 to the end of the array.
...2 // From index 0 to index 2.
..<2 // From index 0 to index 1.
```

Logical Operators (Bool)

```
!a     // Logical NOT : Inverts the value of a
a && b // Logical AND : Returns true if both a and b are true
a || b // Logical OR  : Returns true if a or b is true
```

Compound logical operators are left-associative.
(Leftmost subexpressions are evaluated first.)

Explicit Parentheses

Use explicit parenthesis to clarify code.