Cheatography

Swift Programming Language Cheat Sheet

by kennybatista via cheatography.com/20913/cs/3873/

Arithmetic Operators

Operator Purpose
+ Addition
- Subtraction
* Multiplication
/ Division
% Remainder

OPERATORS

Arithmetic Operators
+ Addition
- Subtraction
* Multiplication
/ Division
% Remainder
Note: Unlike the remainder operator in C and Objective-C, Swift's remainder operator can also operate on floating-point numbers (e.g. 8 % 2.5 // equals 0.5)
Comparative Operators
== Equal to
=== Identical to
!= Not equal to
!== Not identical to
~= Pattern match
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to
Assignment Operators
= Assign
+= Addition
-= Subtraction
*= Multiplication
/= Division
%= Remainder
&= Bitwise AND
|= Bitwise Inclusive OR
^= Exclusive OR
<<= Shift Left
>>= Shift Right
&&= Logical AND
||= Logical OR

OPERATORS (cont)

Increment and Decrement Operators
++ Addition
-- Subtraction
If the operator is written before the variable, it increments the variable before returning its value.
If the operator is written after the variable, it increments the variable after returning its value.
Logical Operators
! NOT
&& Logical AND
|| Logical OR
Range Operators
..< Half-open range
... Closed range
Bitwise Operators
& Bitwise AND
| Bitwise Inclusive OR
^ Exclusive OR
~ Unary complement (bit inversion)
<< Shift Left
>> Shift Right
Overflow and Underflow Operators
Typically, assigning or incrementing an integer, float, or double past it's range would result in a runtime error. However, if you'd instead prefer to safely truncate the number of available bits, you can opt-in to have the variable overflow or underflow using the following operators:
Operator Purpose

OPERATORS (cont)

&+ Addition
&- Subtraction
&* Multiplication
&/ Division
&% Remainder
Example for unsigned integers (works similarly for signed):
var willOverflow = UInt8.max
// willOverflow equals 255, which is the largest value a UInt8 can hold
willOverflow = willOverflow &+ 1
// willOverflow is now equal to 0
var willUnderflow = UInt8.min
// willUnderflow equals 0, which is the smallest value a UInt8 can hold
willUnderflow = willUnderflow &- 1
// willUnderflow is now equal to 255
Another example to show how you can prevent dividing by zero from resulting in infinity:
let x = 1
let y = x &/ 0
// y is equal to 0
Other Operators
?? Nil coalescing
?: Ternary conditional
! Force unwrap object value
? Safely unwrap object value

Comperative Operators

Operator Purpose
== Equal to
=== Identical to
!= Not equal to
!== Not identical to
~= Pattern match
> Greater than
< Less than
>= Greater than or equal to
<= Less than or equal to

Assignment Operators

Operator Purpose
= Assign
+= Addition
-= Subtraction
*= Multiplication
/= Division
%= Remainder
&= Bitwise AND
|= Bitwise Inclusive OR
^= Exclusive OR
<<= Shift Left
>>= Shift Right
&&= Logical AND
||= Logical OR

Bitwise Operators

Operator Purpose
& Bitwise AND
| Bitwise Inclusive OR
^ Exclusive OR
~ Unary complement (bit inversion)
<< Shift Left
>> Shift Right

Increment and Decrement Operators

Operator	Purpose
++	Addition
--	Subtraction

-If the operator is written before the variable, it increments the variable before returning its value.

-If the operator is written after the variable, it increments the variable after returning its value.

Logical Operators

Operator	Purpose
!	NOT
&&	Logical AND
||	Logical OR

Range Operators

Operator	Purpose
..<	Half-open range
...	Closed range