Cheatography

Swift Programming Language Cheat Sheet by kennybatista via cheatography.com/20913/cs/3873/

Arithmetic Operators	OPERATORS (cont)	OPERATORS (cont)	Comperative Operators
Operator Purpose	Increment and Decrement	&+ Addition	Operator Purpose
+ Addition	Operators	&- Subtraction	== Equal to
- Subtraction	++ Addition	&* Multiplication	=== Identical to
* Multiplication	Subtraction	&/ Division	!= Not equal to
/ Division	If the operator is written before the	&% Remainder	!== Not identical to
% Remainder	variable, it increments the variable	Example for unsigned integers	~= Pattern match
	before returning its value.	(works similarly for signed):	> Greater than
OPERATORS	If the operator is written after the	var willOverflow = UInt8.max	< Less than
	variable, it increments the variable	// willOverflow equals 255, which is	>= Greater than or equal to
Arithmetic Operators	after returning its value.	the largest value a UInt8 can hold	<= Less than or equal to
+ Addition	Logical Operators	willOverflow = willOverflow &+ 1	
- Subtraction	! NOT	// willOverflow is now equal to 0	Assignment Operators
* Multiplication	&& Logical AND	var willUnderflow = UInt8.min	
/ Division	Logical OR	// willUnderflow equals 0, which is	Operator Purpose
% Remainder	Range Operators	the smallest value a UInt8 can hold	= Assign
Note: Unlike the remainder	< Half-open range	willUnderflow = willUnderflow &- 1	+= Addition
operator in C and Objective-C,	Closed range	// willUnderflow is now equal to 255	-= Subtraction
Swift's remainder operator can also	Bitwise Operators	Another example to show how you	*= Multiplication
operate on floating-point numbers	& Bitwise AND	can prevent dividing by zero from	/= Division
(e.g. 8 % 2.5 // equals 0.5)	Bitwise Inclusive OR	resulting in infinity:	%= Remainder
Comparative Operators	^ Exclusive OR	let $x = 1$	&= Bitwise AND
== Equal to	~ Unary complement (bit inversion)	let $y = x \& 0$	= Bitwise Inclusive OR
=== Identical to	<< Shift Left	// y is equal to 0	^= Exclusive OR
!= Not equal to	>> Shift Right	Other Operators	<<= Shift Left
!== Not identical to	Overflow and Underflow Operators	?? Nil coalescing	>>= Shift Right
~= Pattern match	Typically, assigning or	?: Ternary conditional	&&= Logical AND
> Greater than	incrementing an integer, float, or	! Force unwrap object value	= Logical OR
< Less than	double past it's range would result	? Safely unwrap object value	
>= Greater than or equal to	in a runtime error. However, if		Bitwise Operators
<= Less than or equal to	you'd instead prefer to safely		
Assignment Operators	truncate the number of available		Operator Purpose
= Assign			& Bitwise AND
+= Addition	bits, you can opt-in to have the		Bitwise Inclusive OR
-= Subtraction	variable overflow or underflow		^ Exclusive OR
*= Multiplication	using the following operators:		~ Unary complement (bit inversion)
/= Division	Operator Purpose		<< Shift Left
%= Remainder			>> Shift Right
&= Bitwise AND			
= Bitwise Inclusive OR			
^= Exclusive OR			
<= Shift Left			
>>= Shift Right			
&&= Logical AND			
= Logical OR			



By kennybatista

cheatography.com/kennybatista/

Not published yet. Last updated 10th April, 2015. Page 1 of 2.

Sponsored by Readability-Score.com Measure your website readability! https://readability-score.com

Cheatography

Swift Programming Language Cheat Sheet by kennybatista via cheatography.com/20913/cs/3873/

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

By kennybatista

cheatography.com/kennybatista/

Not published yet. Last updated 10th April, 2015. Page 2 of 2. Sponsored by Readability-Score.com Measure your website readability! https://readability-score.com