Two websites that will be helpful for our class today!

**[rudd](https://ganyceveningcourses.slack.com/team/rudd" \t "/team/rudd)***[6:02 PM]*[http://practicalswift.com/2014/06/14/the-swift-standard-library-list-of-built-in-functions/](http://practicalswift.com/2014/06/14/the-swift-standard-library-list-of-built-in-functions/" \t "_blank)

[practicalswift.com](http://practicalswift.com/" \t "http://practicalswift.com)

**[practicalswift](https://practicalswift.com/author/practicalswift/" \t "https://practicalswift.com/author/practicalswift/)**

**[Swift Standard Library: Documented and undocumented built-in functions in the Swift standard library – the complete list with all 74 functions](http://practicalswift.com/2014/06/14/the-swift-standard-library-list-of-built-in-functions/" \t "http://practicalswift.com/2014/06/14/the-swift-standard-library-list-of-built-in-functions/)**

Swift has 74 built-in functions but only seven of them are documented in the Swift book (“The Swift Programming Language”). The rest remain undocumented.  
This article lists all built-in Swift functions – both documented and undocumented ones. The definition used for “built-in function” used in this article is a function available in Swift without importing any modules (such as Foundation, etc.) or referencing any classes.  
Let’s start with the seven documented built-in functions mentioned in the Swift book along with the page number on which the function was first mentioned:  
// assert mentioned on page 55   
assert(true)   
// countElements mentioned on page 79   
countElements("foo") == 3   
// enumerate mentioned on page 94   
for (i, j) in enumerate(["A", "B"]) {   
   // "0:A", "1:B" will be printed   
   println("\(i):\(j)")   
}   
// min mentioned on page 246   
min(8, 2, 3) == 2   
// print mentioned on page 85   
print("Hello ")   
// println mentioned on page 4   
println("World")   
// sort mentioned on page 14   
for i in sort(["B", "A"]) {   
   // "A", "B" will be printed   
   println(i)   
}  
Now on to the most useful undocumented functions …  
abs(signedNumber): Returns the absolute value of a given signed number. Trivial but not documented.abs(-1) == 1   
abs(-42) == 42   
abs(42) == 42contains(sequence, element): Returns true if a given sequence (such as an array) contains the specified element.var languages = ["Swift", "Objective-C"]   
contains(languages, "Swift") == true   
contains(languages, "Java") == false   
contains([29, 85, 42, 96, 75], 42) == truedropFirst(sequence): Returns a new sequence (such as an array) without the first element of the sequence.var languages = ["Swift", "Objective-C"]   
var oldLanguages = dropFirst(languages)   
equal(oldLanguages, ["Objective-C"]) == truedropLast(sequence): Returns a new sequence (such as an array) without the last element of the sequence passed as argument to the function.var languages = ["Swift", "Objective-C"]   
var newLanguages = dropLast(languages)   
equal(newLanguages, ["Swift"]) == truedump(object): Dumps the contents of an object to standard output.var languages = ["Swift", "Objective-C"]   
dump(languages)   
// Prints:   
// ▿ 2 elements   
//   - [0]: Swift   
//   - [1]: Objective-Cequal(sequence1, sequence2): Returns true if sequence1 and sequence2 contain the same elements.var languages = ["Swift", "Objective-C"]   
equal(languages, ["Swift", "Objective-C"]) == true   
var oldLanguages = dropFirst(languages)   
equal(oldLanguages, ["Objective-C"]) == truefilter(sequence, includeElementClosure): Returns a the elements from sequence that evaluate to true by includeElementClosure.for i in filter(1...100, { $0 % 10 == 0 }) {   
   // 10, 20, 30, ...   
   println(i)   
   assert(contains([10, 20, 30, 40, 50, 60, 70, 80, 90, 100], i))   
}find(sequence, element): Return the index of a specified element in the given sequence. Or nil if the element is not found in the sequence.var languages = ["Swift", "Objective-C"]   
find(languages, "Objective-C") == 1   
find(languages, "Java") == nil   
find([29, 85, 42, 96, 75], 42) == 2indices(sequence): Returns the indices (zero indexed) of the elements in the given sequence.equal(indices([29, 85, 42]), [0, 1, 2])   
for i in indices([29, 85, 42]) {   
   // 0, 1, 2   
   println(i)   
}join(separator, sequence): Returns the elements of the supplied sequence separated by the given separator.join(":", ["A", "B", "C"]) == "A:B:C"   
var languages = ["Swift", "Objective-C"]   
join("/", languages) == "Swift/Objective-C"map(sequence, transformClosure): Returns a new sequence with the transformClosure applied to all elements in the supplied sequence.equal(map(1...3, { $0 \* 5 }), [5, 10, 15])   
for i in map(1...10, { $0 \* 10 }) {   
   // 10, 20, 30, ...   
   println(i)   
   assert(contains([10, 20, 30, 40, 50, 60, 70, 80, 90, 100], i))   
}max(comparable1, comparable2, etc.): Returns the largest of the arguments given to the function.max(0, 1) == 1   
max(8, 2, 3) == 8maxElement(sequence): Returns the largest element in a supplied sequence of comparable elements.maxElement(1...10) == 10   
var languages = ["Swift", "Objective-C"]   
maxElement(languages) == "Swift"minElements(sequence):  Returns the smallest element in a supplied sequence of comparable elements.minElement(1...10) == 1   
var languages = ["Swift", "Objective-C"]   
minElement(languages) == "Objective-C"reduce(sequence, initial, combineClosure): Recursively reduce the elements in sequence into one value by running the combineClosure on them with starting value of initial.var languages = ["Swift", "Objective-C"]   
reduce(languages, "", { $0 + $1 }) == "SwiftObjective-C"   
reduce([10, 20, 5], 1, { $0 \* $1 }) == 1000reverse(sequence): Returns the elements of the given sequence reversed.equal(reverse([1, 2, 3]), [3, 2, 1])   
for i in reverse([1, 2, 3]) {   
   // 3, 2, 1   
   println(i)   
}startsWith(sequence1, sequence2): Return true if the starting elements sequence1 are equal to  the of sequence2.startsWith("foobar", "foo") == true   
startsWith(10..100, 10..15) == true   
var languages = ["Swift", "Objective-C"]   
startsWith(languages, ["Swift"]) == trueBelow is the full list of all 74 built-in functions in Swift. The functions covered above are the ones I think are useful on a day-to-day basis, but perhaps I’ve missed some functions from the list below that deserves coverage. If so, let me know in the comments section and please include a short code snippet to show how to use the function.  
Happy Swifting!  
abs(...)   
advance(...)   
alignof(...)   
alignofValue(...)   
assert(...)   
bridgeFromObjectiveC(...)   
bridgeFromObjectiveCUnconditional(...)   
bridgeToObjectiveC(...)   
bridgeToObjectiveCUnconditional(...)   
c\_malloc\_size(...)   
c\_memcpy(...)   
c\_putchar(...)   
contains(...)   
count(...)   
countElements(...)   
countLeadingZeros(...)   
debugPrint(...)   
debugPrintln(...)   
distance(...)   
dropFirst(...)   
dropLast(...)   
dump(...)   
encodeBitsAsWords(...)   
enumerate(...)   
equal(...)   
filter(...)   
find(...)   
getBridgedObjectiveCType(...)   
getVaList(...)   
indices(...)   
insertionSort(...)   
isBridgedToObjectiveC(...)   
isBridgedVerbatimToObjectiveC(...)   
isUniquelyReferenced(...)   
join(...)   
lexicographicalCompare(...)   
map(...)   
max(...)   
maxElement(...)   
min(...)   
minElement(...)   
numericCast(...)   
partition(...)   
posix\_read(...)   
posix\_write(...)   
print(...)   
println(...)   
quickSort(...)   
reduce(...)   
reflect(...)   
reinterpretCast(...)   
reverse(...)   
roundUpToAlignment(...)   
sizeof(...)   
sizeofValue(...)   
sort(...)   
split(...)   
startsWith(...)   
strideof(...)   
strideofValue(...)   
swap(...)   
swift\_MagicMirrorData\_summaryImpl(...)   
swift\_bufferAllocate(...)   
swift\_keepAlive(...)   
toString(...)   
transcode(...)   
underestimateCount(...)   
unsafeReflect(...)   
withExtendedLifetime(...)   
withObjectAtPlusZero(...)   
withUnsafePointer(...)   
withUnsafePointerToObject(...)   
withUnsafePointers(...)   
withVaList(...)  
Twitter: @practicalswift

**[rudd](https://ganyceveningcourses.slack.com/team/rudd" \t "/team/rudd)***[6:02 PM]*[http://kpbp.github.io/swiftcheatsheet/#variables](http://kpbp.github.io/swiftcheatsheet/" \l "variables" \t "_blank)

A quick cheat sheet and reference guide for Apple's Swift language. This guide intends to cover all the key features of Swift, including Strings, Arrays and Dictionaries.