The Swift Programming Language: Quick Reference Sheet

Variables	
let	Keyword to declare a constant
var	Keyword to declare a variable

Types	
Int	e.g., 123
Bool	e.g., True
String	e.g., "Hello"
Character	e.g., 'H'
Double	e.g., 34.22156358
Float	e.g., 1.2

Collections	
array	Same types in ordered list.
set	Same types in unordered list.
dictionary	Key-Value pair with no order.

Classes	
class	Keyword to declare a class.
class <*name*>	Classes can have variables and methods.

Structs	
struct	Keyword to declare a struct.
struct <*name*>	Classes can have variables and methods.

Functions	
func	Keyword to declare a function.
func <*name*>()−> <*rtntype*>	You must type the return value.
func <*name*>(<*a*>: <*type*>)	You must type the parameters.
func <*name*>(_<*a*>: <*type*>)	You can use underscore, before the argument name, to avoid having to label the parameter in the call.
func <*name*>()−> (<*type*>, <*type*>)	You can return multiple values as a tuple.
inout	Keyword to use when you want to copy by reference.
func <*name*>(<*a*>: inout <*type*>)	inout variables require the reference in the call.
mutating	Keyword to use when you want modify the structure.

Properties	
lazy	Keyword to make a property lazy.
var <*name*>: <*type*> { get {} set {} }	Allows for computed properties.
willSet(val)	Will execute when the value is about to change.
didSet	Will execute after the value changed.

Initialization and Deinitialization	
init() {}	Keyword used with objects to initialize values. Similar to constructor.
deinit {}	Keyword used when you want something to happen when the object is destroyed. Similar to deconstructor.

Extensions	
extension	Keyword used to extend a type.
extension <*type*>{}	Allows you to create new variables and methods and attach them to the type.
extension <*type*>, <*Prtcl*>{}	You can chain on protocols to give further functionality to types.

Protocols	
protocol	Keyword for creating a protocol.
protocol <*name*>	Protocols allow to you blueprint variables and methods you want to be implemented in the object.
class <*class*>	You can attach a protocol to a class, struct, or extension.

Generics	
func someFunction<*T*>	Use <*T*> to allow you to use generic types.
func someFunction<*T*> (<*a*>:*T*)	Use T in your parameters to give it the same generic type.