Guard statements

Used to check values and perform optional binding
(similar to if/let)

If the optional binding fails or the condition is false, the
guard block is executed

Guard blocks must return or throw

Bound variables are in scope after the guard statement
is the guard statement succeeds (i.e., does not return/
throw)
Error Handling

3 kinds of errors: simple, complex, fatal
Complex errors previously handled with NSError in/out parameters and an appropriate return value (nil / false)
Swift 2.0 adds a throws keyword for func declarations, a throw keyword for throwing an error, and do/try/catch syntax for calling a func that throws
Can throw anything that adopts the ErrorType protocol (NSError adopts this protocol)

Error Handling, cont.

The try! keyword can be used to call a method that throws without do/catch, but will crash if the method throws
Catch statements must be exhaustive
Current SDK methods that take an NSError inout parameter are automatically translated to the new syntax
No finally block, as there is in Java
Deferred Statements

Code that is executed at the end of a method invocation
Executes regardless of if the method returns or throws

Protocol Extensions

Extensions can be written for protocols
Methods provided in a protocol extension are used as a default implementation and won't be present if the adopting class has an implementation
Already being adopted to add some methods to protocols in the API (for example, indexOf global function is now available on collections as a method)
Documentation

“The Swift Programming Language”