Created by: Philip Thomas Barkow iOS Mandatory Week 16

Closure & Weak vs. Strong

Closure

In Swift functions are just named Closures. Closures are in it essence all about enclosed functionality that can be used in our code. Closures capture data in the context they are declared in. Now what that means is that we can for instance declare a Closure as a param to a function and use the result of the previously defined Closure in another function. A Closure can be an unnamed function in case we choose to define it as part of a variable or constant declaration like in the following example:

A key feature of using Closures is our ability to use positional arguments within the body of our Closure. For instance we could declare a Closure with a param of type String called ‘name’ and retrieve the value of the param called ‘name’ with $0 as it is the first positional argument. By supporting the last mentioned, Closures allows us to write more shorthanded code as well.

Weak vs. Strong

When talking about strong/weak in swift we are talking about declaration of variables or properties of a class. How we choose to declare this when declaring variables or properties in a class can have significance for our application in the sense that memory leaks may be introduced if considerations of wether to declare as weak or strong has not been done. If you do not declare for instance a variable as being weak, with the prefix ‘weak’ before the variable name, it is implicit that it is strong. If a strong variable is being used somewhere in your application, you will not be able to unset the variable and free it from memory, thus causing a memory leak as it is being held in memory. If we in comparison declare a variable as weak it will be released when it is no longer used in our application. Also strong/weak is a way to describe how strong the reference to a variable or class for instance is.

An example of weak/strong could be a class called Order, who has a var called *product* of the type Product. Now imagine Product having an optional var called *order* of type Order. If we were to declare these as strong we would leave out the weak prefix to the naming of our variables. On later assignment of these properties to one another we would establish a strong reference to one another. By doing so we are not able to release the one or the other from memory because they are tightly coupled. So in this case it would make sense to for instance change the declaration of *order* as weak as it then allows us to release it from memory by for instance setting it to nil.