Eric Tappan

Software Systems

LKM #6: Things Not to Do in Objective C

Introduction

My intent in this project was to explore Objective C as a language, partly by trying to implement some standard problems in functional programming (Objective Th). My original goal was to explore the sensors exposed to Objective C on an iOS device, but problems involving Windows and appdev meant a late change in focus.

I took advantage of a few online resources in this project. These include Code School's introductory Objective C course[1] and the free chapter of their introductory iOS course.[2] Additionally, I took advantage of the Mac Developer Library[3] and a few sections of Objective-C Succinctly.[4] Going forward, I intend to look more closely at Succinctly if I want to further develop my skills in the language without a specific target in mind.

My primary goal in this module was to learn a new language (always interesting) and potentially give mobile development a try (sadly unrealised). I gained some insight into the former, along with experience in the many interesting ways a language may make itself inaccessible. I would be happier if I had managed to get Objective C working on any platform with its full set of features, and currently I am in the embarrassing position of not being able to effectively test the syntax of the code I present below. To the best of my knowledge it should function, but function blocks are a surprisingly recent development in Objective C and not easily attainable in Windows or Ubuntu.

[1] https://www.codeschool.com/courses/try-objective-c
[2] https://www.codeschool.com/courses/try-ios
[3] https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html#//apple_ref/doc/uid/TP40011210-CH1-SW1
[4] http://www.syncfusion.com/resources/techportal/ebooks/objective-c

Closure

Objective C is a strange language in many ways, but closures are actually straightforward. (see: *Working with Windows* and on for some caveats) As with Python, a Objective C blocks capture context from their declaring scope. For example:[5]

```
 -(void)testMethod {
     int anInteger = 42;
     void (^testBlock)(void) = ^{
         NSLog(@"Integer is: %i", anInteger);
     };
     testBlock();
 }
```

^testBlock will capture anInteger as a constant value of 42. Even if anInteger changes value, ^testBlock will always log “Integer is: 42”. As with C returning blocks has some unfortunate syntax, but it’s still a straight shot from here to our next step:[6]

```
-( void (^))makeIntPrinter: (int) anInteger {
   void (^intPrinter)(void) = ^{
      NSLog(@”Integer is: %i”, anInteger);
   };
   return intPrinter;
}
```

This will create a toy block for printing out a message with a particular integer. With some nesting we can produce a useful piece of functional programming (where here we pretend integers are The One True Type):

```
-( int (^) (int) )applyBlock: ( int (^) (int) ) block, nTimes: (int) times {
   int (^nDoer) (int) = ^(int val){
      int res = val;
      int i;
      for ( i = 0; i < times ; i++ ) {
```

[5] https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/WorkingwithBlocks/WorkingwithBlocks.html#//apple_ref/doc/uid/TP40011210-CH8-SW1
[6] http://fuckingblocksyntax.com

```
            res = block(res);
        }
        return res;
    };
    return nDoer;
}
```

And another similar problem:

```
-( int (^) (int) )applyBlock: ( int (^) (int) ) block, Until: (Bool (^) (int)) until {
    int (^nDoer) (int) = ^(int val){
        int res = val;
        int i;
        while (!until(res)) {
            res = block(res);
        }
        return res;
    };
    return nDoer;
}
```

Working with Windows

Ideally? Don't. Apple has no strong motivation to support development on Windows, so all official tools from Apple are designed with Macs in mind. Third-party solutions do exist, though they aren't necessarily so polished or featureful as XCODE.

Apple supports Clang as the official compiler for Objective C. This is great! Clang is fresh and shiny compared to GCC. While in principle it should be possible for GCC to support Objective C, recent years have seen Apple adding extensions to Objective C that aren't available with GCC. Unfortunately, this invalidates some of the more straightforward guides for getting started in Windows--Clang doesn't have a Windows installer! Luckily we aren't the only ones to think this made any kind of sense.

<http://solarianprogrammer.com/2012/03/21/clang-objective-c-windows/> Provides a walkthrough for installing Clang on Windows, whether or not you want to build from source yourself or just grab the binary.

<http://sweettutos.wordpress.com/2012/08/11/objective-c-on-windows-yes-you-can/> Provides a ("sweet") tutorial for getting a simple Hello World executable working in Windows using the open source GNUStep. There are a few small errors: don't save your makefile with the '.mak'

extension, and build using ‘make -f GNUMakefile’. If this is all you want to do you’re set with GCC, to use modern syntax and have any chance of getting help from the wider internet you’ll need Clang.

Ironically, installing Clang requires a more up-to-date version of GCC than I had available or was able to install on Windows.

Jailbreaking

p0sixspwn may actually cause problems with your iPad that prevent you from restoring to a backup. Be aware that you may need to do a complete update/recover operation which (as of this writing) will put you in the un-jailbreakable iOS 7.1.1. This is sad, and a reminder to be clear on all the risks of using weird software from the internet.

Working with Linux

<http://blog.tlensing.org/2013/02/24/objective-c-on-linux-setting-up-gnustep-clang-llvm-objective-c-2-0-blocks-runtime-gcd-on-ubuntu-12-04/> apt-get install your way to a better life. Unless you’re running in Ubuntu and can’t get the right version of libobjc2, which is impossible at the moment due to some crucially broken links.

If all else fails...

<http://www.compileonline.com/compile_objective-c_online.php> Someone, somewhere, has made something to let you do it online. For some reason there don’t appear to be free, online, editors which support New-Style Objective C. All of this would have been a lot prettier if I could use ^blocks and @properties.