Objective-C Basics

CocoaHeads Presentation - September 18, 2014

What is Objective-C?

- Objective-C is a strict superset of the C programming language:
- i.e., this is a perfectly valid Objective-C program:

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    printf(“Hello World\n”);
    return 0;
}
```

  as is:

```
#import <Foundation/Foundation.h>

int main(int argc, const char *argv[])
{
    @autoreleasepool {
        NSLog(@“Hello World!”);
        return 0;
    }
}
```

- http://en.wikipedia.org/wiki/Objective

Where did Objective-C come from?

- Objective-C was created primarily by Brad Cox and Tom Love in the early 1980s
  - See http://en.wikipedia.org/wiki/Objective-C for more details
- It is strongly influenced by Smalltalk
- an example:

```
Transcript cr.
Transcript show:'Hello World'.
Transcript cr.
```

What does this have to do with Cocoa/CocoaTouch?

- Objective-C, for the foreseeable future, is the primary development language for Cocoa & Cocoa Touch
- Swift will probably displace some Objective-C but this will take a long time
- For performance, low-level C will always exist and hence Objective-C on top of that

If Objective-C is so cool, why doesn't everyone use it ?

- Being influenced by Smalltalk, it looks weird to programmers coming from other C-derived imperative languages (C++, Java, PHP, Perl, etc)
- i.e.
  calling a method in Java:
  myInstance.myMethod(arg1, arg2);

  calling a method in Objective-C:
  [myInstance my:arg1 method:arg2 signature:arg3];
- An instance method signature could be referred to as
  enumerateSubstringsInRange:options:usingBlock:
  but we would need the types of the arguments
  - (void)enumerateSubstringsInRange:(NSRange)range options:(NSStringEnumerationOptions)opts usingBlock:(void (^)(NSString *substring, NSRange substringRange, NSRange enclosingRange, BOOL *stop))block

coolness, continued

- The core of Objective-C is portable and can be compiled on multiple systems with GCC and CLANG, however …
- That’s like saying write everything in ANSI C. People don’t do it, we use libraries and frameworks to increase our productivity.
- For OS X & iOS development, we tend to use non-portable language features. In comparison, Visual C++ projects using Microsoft frameworks do not run on OS X.

more on coolness …

- So why use it ?!?!?!?!?!
- Because it is the first-class citizen of the ecosystem
- Everything else is just layered on top of it. Sometimes runtimes on top of runtimes. Do you want the best performance and earliest access to features ? Or can you make a business case for using a simpler or cross-platform tool ?
- i.e., Corona SDK - Cross-platform gaming framework exposed via Lua. (World of Warcraft scripting)

  Corona SDK games have slow load times on iOS due to embedding another language runtime into the app

  Shameless plug: Learning Gems Math 2.0
  https://play.google.com/store/apps/details?id=com.bluefireventures.learninggemsmath20&hl=en

Ok, how do I use it ?

- Path of least resistance: Go buy a modern Mac
- Xcode is a free download
- Make a new project with SHIFT-COMMAND-N
- Set the options
- Run the project with COMMAND-R
- START SMALL

A tale of two languages

- Let’s look at two projects, one in Objective-C and one in Java
- Both projects implement a Fruitstand and a number of fruits along with a main program to run a simulation
- The Objective-C project was made in Xcode, the Java project in NetBeans

Fruitstand

- A fruitstand has a couple of properties:
  - A list of operators who are working the stand
  - A collection of boxes holding various types of fruit
- We can perform a few operations on a fruit stand
  - We can put a piece of fruit into a named box
  - We can print out who is working at the stand
  - We can print out the inventory of the stand

Fruit

- A fruit has a couple of general properties:
  - fruit tends to be some color
  - fruit tends to have ripened to some stage

Types of fruit

- We generally do not eat “fruit”, we eat certain kinds of fruit like:
    - Orange
    - Strawberry
    - Banana
    - Grape

continued

- Orange
    - has a rind
- Strawberry
    - has exterior seeds
- Banana
    - has a peel
- Grape
    - can be seedless

How do we make classes and methods ?

- In Objective-C, we have a “header” file and an “implementation” file
    - These “header” files provide the “public interface” into the class
    - *** There is no such thing as actual encapsulation in Objective-C (public/private/protected access), you have to fake it
        - Perl has the same problem
- In Java, we have “class” files
    - The public members and methods of the class are introspected to provide the public interface
- Switch to code !

Fake it until you make it !

- In Objective-C, you will hear about "public" and "private" APIs
- You can poke around the runtime and find out anything you want about anything
- WARNING: Do not depend on private APIs as you will likely encounter breaking changes
- In Java, the JVM enforces the stated access methods of members and methods
- Back to the code !

Demo

iOS: ~/ios-projects/cocoaheads-20140918/presentation/DerivedData/presentation/Build/Products/Debug

```
./presentation
```

Java: ~/NetBeansProjects/presentation/build/classes

java presentation/Presentation