The Objective-C Programming Language

by Thomas Verstraete & Philip Webster

CS 343 Introduction to Programming Languages

9 December 2013

One of the rockstars of the programming language world, having shot up from obscurity into the top tier of popular languages in less than 5 years is Objective-C. This hybrid language has become so popular because the of the popularity of Apple Computer's iPhone and has trickled out to other Apple products. The nature of the language is a clever combination of the most popular language, C, and an obscure object-oriented or OO language, SmallTalk. Objective-C's history made it as popular as it is and its the unique combination of functional and object-orientation that informs the data and control abstractions.

To understand the history of Objective-C we need to understand a little history of its parts. The C programming language was created by Dennis Richie and Ken Thompson in the late 60's and early 70's. During this time the two were working at AT&T Bell Labs and were developing the Unix Operating System [1]. At first writing unix in assembly, they created C to make this task easier. C has gone on to become one of the most popular programming languages. The other part that makes up objective C is Smalltalk. It was one of the early attempts at an object-oriented language. Created by Alen Kay and Dan Ingalls at Xerox Palo Alto Research Center[1]. C creates the backbone of Objective-C and Smalltalk informs the object-orientation of the language.

Although Smalltalk never really took off and time has relegated it to hobby projects it was the influential piece to add object-oriented paradigms to standard C [5]. In the early 1980's Brad Cox and Tom Love created Objective-C as a strict super-set of standard C with objects and message passing similar to Smalltalk [1]. It was these these features that made it attractive as a language and caught the eye of Steve Jobs and NeXT. In 1988 after Steve Jobs was forced out of Apple Computers, he started a new company called NeXT [5]. Jobs wanted to "design an entirely new generation of computer system" [1]. Developing this computer system from the ground up; hardware, software, and OS. The operating system was based on the Mach microkernel developed by Avie Tevanian. The Mach microkernel is a derivative the of the Unix operating system and lent itself to object-orientation [2].

With an object-oriented operating system you need an object-oriented programming language. NeXT decided on Objective-C because C++ was not available then and it was a well respected object-oriented language [2]. This decision was beneficial in the OO aspect, but also because of its relationship to C. With the basis of the operating system (Unix) written in C this aided the development of the OS on top of

the microkernel with Objective-C. Also with the popularity of C it was easy for programmers to learn the new syntax of Objective-C. The choice of Objective-C seemed to be perfect for NeXT, but it was Apple's next decision that brought the language into even greater popularity.

In the 1990's Apple Computers was looking to replace their outdated OS to compete with Windows. After many attempts with various partners they decided to purchase a company with a complete OS, and the choice came down to Be, Inc or NeXT [1]. The decision was made to purchase NeXT and bring Steve Jobs back to Apple Computers. Jobs regained the CEO position and the rise of Apple from the brink began. One part of it was taking the NeXT os and from it developing and releasing Mac OSX. Along with this new operating system can Objective-C as the native language [1].

Although Mac OSX came out in spring of 2001 Objective-C didn't start to see any significant usage until mid 2009 with the release of another step in Apple's current status. The first iPhone was released in 2007. The operating system for the iPhone was built with the same technology as Mac OSX and thus the use of Objective-C. However the abuility for outside developers to create applications for the iPhone didn't happen until Apple Released the SDK in spring of 2008 [4]. According to Tiobe.com a website that tracks the popularity of programming languages, Objective-C rose from obscurity at about the 38th most popular language in mid 2009 to the third most popular language today [3]. It beat out C++ to the third spot in mid 2012, taking just 3 years to get there [3].

Objective-C is the language to develop native applications on any iOS device. It continues to be the language of Mac OS X. It is a language that can be used on any system, but it is the frameworks that support this language the most. The Cocoa framework for Apple devices is the power behind the language on their platforms. It is available with other frameworks for other platforms but it is most prevalent with Apple devices.

The unique features of Objective-C do not make describing its computational paradigms, data or control abstractions clear cut. Being a strict superset of standard C with a layer of Smalltalk concepts on top, make this language a hybrid for many of the key descriptions of a language. For example C is a procedural language. However because of the additions of Object-Oriented programming from Smalltalk Objective-C is also Object-Oriented. This duality is a powerful feature that will show up in most of the data and control abstractions describe below.

The data types of Objective-C are fairly standard for any language. Of course it has all the basic primitives of standard C, like integers, floating points, chars, all signed and unsigned. It has void like C. There are some Objective-C specific primitives such as ID, Class and SEL. The ID data type is the generic type for the objects. The Class type as you might assume is the type for classes and SEL is for selectors and is the datatype to store methods [6]. Another thing Objective-C offers is an object-oriented wrapper around C primitives called NSNumber. The purpose of this is to be able to store these primitive values in object arrays. Also just like the primitives these wrappers are immutable. The data types in Objective-C are built on top of C but with additions to make it object-oriented.

The combination of C and Smalltalk paradigms are prevalent in type checking just as it is in the other aspect of Objective-C. Standard C is a statically typed language so Objective-C is also statically typed. But it couldn't be that easy. Because of the ported aspects of message passing from Smalltalk Objective-C also needs to be dynamically typed. So Objective-C is both statically and dynamically typed. How this works depends on how the syntax is written when programming.

Control Abstractions

Expressions in Objective-C are nearly identical to those in C. Infix notation is used for binary operators and prefix notation for unary and other operators. Simple arithmetic can be written as `int x = 5 + 2;`. The language has also inherited the same operator precedence as C, as shown in Figure 1.

C
++, -- (post-inc., dec.)
++, -- (pre-inc., dec.), +, - (unary), &, * (address, contents of), !, ~ (logical, bit-wise not)
* (binary), /, % (modulo division)
+, - (binary)
<<, >> (left and right bit shift)
<, <=, >, >= (inequality tests)
==, != (equality tests)
& (bit-wise and)
^ (bit-wise exclusive or)
| (bit-wise inclusive or)
&& (logical and)
|| (logical or)
?: (if…then…else)
=, +=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |= (assignment)
, (sequencing)

Figure 1. Operator Precedence in C and Objective-C [7]

Instead of functions, Objective-C uses “messages” to communicate between objects. This is done with a “mixfix” notation in the style of Smalltalk and looks like this: `[self.stepButton setEnabled:TRUE];`. Apple encourages developers to use descriptive names for the message and parameters, which can make them fairly long. Splitting the parameters across multiple lines is a common practice, for example:

```
_timer = [NSTimer scheduledTimerWithTimeInterval:interval
                                          target:self
                                        selector:@selector(autorun:)
                                        userInfo:nil
                                         repeats:YES];
```

Objective-C uses the same selection constructs as C, with the following notation for an if/else statement:

```
if (condition) {
  statement
} else {
```

```
  statement
}
```

A switch statement uses the following notation:

```
switch (expression) {
  case match1:
    statement
    break;
  case match2:
    statement
    break;
  default:
    statement
    break;
}
```

One drawback of C-style switch statements is the lack of support for matching within a range. This results in more case statements. Switch statements also “fall through” or execute the default case unless a break is inserted at the end of a case.

In addition to C’s `while`, `for`, and `do-while` flow control mechanisms, Objective-C provides `for-each` functionality for fast enumeration of collections that adhere to the NSFastEnumeration protocol. Some examples include NSArray, NSSet, NSDictionary, and of course, any class that you write conforming to NSFastEnumeration protocol. Here is sample code for how it looks in practice:

```
for (id object in array) {
  // do something with object
}
```

As touched on earlier, functions in Objective-C are called methods, which are invoked on objects by sending a message to them. Methods are declared in the class’s header file, which will look like the following short example:

```
@interface XYZPerson : NSObject
@property NSString *age;
@property (readonly) BOOL isAlive;
- (void)sayHello:(NSString *)name;
+ (void)incrementNumObjects;
```

```
@end
```

The first line indicates that we are declaring the class `XYZPerson`, which inherits from `NSObject`. The `@property` says that `age` is a publicly readable and modifiable value. If we don't wish the value to be modified, we can place `(readonly)` in front of it, which will still allow public access. The minus in front of the `sayHello` method indicates that it is an instance method, and the plus sign indicates that `incrementNumObjects` is a class method. `(void)` is the return type of the method, and any parameters with their type follow the method name.

Scope in Objective-C is the same as C, and local variables must be unique within the same scope. Method names should also be unique within a class. If using the `@property` syntax to declare object properties, the compiler creates the getter and setter methods automatically. If these need to be overridden, the accessor method should be named the same as the property (with the exception of boolean properties).

In order to avoid name collisions, best practice is to use a 3 uppercase letter prefix when creating your own classes. Apple reserves two letter prefixes for their own use. Company initials or some other descriptive letters are usually used for these prefixes.

Exceptions in Objective-C are used only for errors caused by the programmer, which is different from many other languages. Apple's documentation recommends inserting a try-catch block if you write code that may cause an exception. Other errors such as network problems and data mismatch cause `NSErrors` to be thrown. Because methods only have one return value, they may require a parameter that is a pointer to an uninitialized `NSError`. The code that followed could then check for the presence of an error.

This odd hybrid of functional and object-oriented programming has risen in popularity because of the decisions of a few select companies, such as NeXT, and Apple. Its ease of use comes mostly from his super-set of C nature making it easy for anyone with C experience to quickly pick up the language and use the powerful tools it provides. Anyone considering developing for the Mac or for iOS need to consider the learning Objective-C.

References

[1] Smyth, Neil. Objective-C 2.0 Essentials. Payload Media, 2012. ebook.

2] Hormby, Tom. “Full Circle: A Brief History of NeXT.” Low End Mac. Cobweb Publishing Inc., 5 Jul. 2005. Web. 18 Nov. 2013.

[3] TIOBE Software: Tiobe Index. Nov. 2013. Web. 28 Nov. 2013.

[4] Block, Ryan. “Live from Apple's iPhone SDK press conference.” Engadget. AOL Inc, 6 Mar. 2008. Web. 11/28/13.

[5] Welk, Richard. Cocoa. Indianapolis: Wiley Publishing, 2010. Print.

[6] “Primitives | Ry’s Objective-C Tutorial | RyPress.” *RyPress*, 2012-2013. Web. 4 Dec. 2013.

[7] Scott, Michael. “Programming Language Pragmatics.” Elsevier Inc., 2006.

[8] “Programming with Objective-C”. Apple Developer Library. Apple Inc. 2012.