Introducing Objective-C and Xcode

For the most part, all computer languages perform the typical tasks that any computer needs to do—store information, compare information, make decisions about that information, and perform some action based on that decision. Objective-C is a language to make these tasks easier to accomplish and understand. The real trick with Objective-C (actually the trick with any C language) is understanding the symbols and keywords used to accomplish those tasks. This chapter introduces you Objective-C and Xcode—from Objective-C’s humble beginnings as an extension to the C language to using Xcode (a tool to build programs using Objective-C) to build programs. By the end of this chapter, you will understand what Objective-C is and know how to write a simple application using Xcode.

A Brief History of Objective-C

Objective-C is really a combination of two languages: the C language and a lesser-known language called Smalltalk. Back in the 1970s, several very bright engineers from Bell Labs created a language they named C that made it easy to port their pet project, the UNIX operating system, from one machine to another. Prior to C, people had to write programs in assembly languages. The problem with assembly languages is that each is specific to its machine, so moving software from one machine to another was nearly impossible. The C language, created by Brian Kernighan and Dennis Ritchie, solved this problem by providing a language that wrote out the assembly language for whatever machine it supported, a kind of Rosetta Stone for early computer languages. Because of its portability, C quickly became the de facto language for many types of computers, early PCs especially.

Fast-forward to the early 1980s, and the C language is on its way to becoming one of the most popular languages of the decade. Right around this time, an engineer from a company called Stepstone was mixing the C language with another up-and-coming language called Smalltalk. The C Language is typically referred to as a procedural
language, that is, a language that uses procedures to divide up processing steps. Smalltalk, on the other hand, was something entirely different. It was an object-oriented programming language. Instead of processing things procedurally, it used programming objects to get its work done. This new superset of the C language became known as “C with Objects” or more commonly, Objective-C.

In 1985, Brad Cox sold the Objective-C language and trademark to NeXT Computer, Inc. NeXT was the brainchild of Steve Jobs, who had been fired from his own company, Apple Computer, that very same year. NeXT used the Objective-C language to build the NeXTSTEP operating system and its suite of development tools. In fact, the Objective-C language gave NeXT a competitive advantage with all of its software. Programmers using NeXTSTEP and Objective-C could write more-functional programs faster than those writing in the traditional C language. While the hardware part of NeXT computers never really took off, the operating system and tools did. Quite interestingly, NeXT was purchased by Apple Computer in late 1996 with the intention of replacing its aging operating system, which had been in existence since the first Macintosh was developed in 1984. Four years after the acquisition, what had been NeXTSTEP reemerged as Mac OS X—with Objective-C still at the heart of the system.

Understanding C Language Basics

Even though Objective-C integrates a great object-oriented language, at the heart of Objective-C is C. Here is the most basic “Hello World” program written in the C language:

```c
int main(void)
{
    if (printf("Hello World") == 0)
        return 0;
    else
        return 1;
}
```

Let’s dissect this a bit. Every program must start somewhere, right? Well, for Objective-C and C, main is the name of the procedure (which is often called a function in C) that is called first.

```c
int main(void)
```

It must be called “main”, not “Main,” “MAIN,” or anything else. C and Objective-C are case-sensitive languages meaning that main and Main are entirely different names.

Functions (and main is a function) all share the following syntax when they are declared:

```c
return-type functionName ( argument-list )
```