Table of contents

- Section One: Introduction
    - The Development Environment
    - Why Objective-C
    - Preliminary Overview
        - Executive Overview
    - Main Overview
    - Conclusion
- Section Two: The Basics

Section One: Introduction

The Objective-C language is a simple computer language designed to enable sophisticated object-oriented programming.

Object-oriented programming, like most interesting developments, builds on some old ideas, extends them, and puts them together in novel ways. The result is many-faceted and a clear step forward for the art of programming. An object-oriented approach makes programs more intuitive to design, faster to develop, more amenable to modifications, and easier to understand. It leads not only to alternative ways of constructing programs, but also to alternative ways of conceiving the programming task.

Nevertheless, for those who have never used object-oriented programming to create applications before, object-oriented programming may present some formidable obstacles. It introduces a new way of doing things that may seem strange at first, and it comes with an extensive terminology that can take some getting used to. The terminology will help in the end, but it’s not always easy to learn. It can be difficult to get started.

That’s where this book comes in. It fully documents the Objective-C language, an object-oriented programming language based on standard C, and provides a foundation for learning about the Mac OS X Objective-C application development framework—Cocoa.

This book is also designed to help you become familiar with object-oriented programming and get over the hurdle its terminology presents. It spells out some of the implications of object-oriented design and tries to give you a flavor of what writing an object-oriented program is really like.

The book is intended for readers who might be interested in:

- Learning about object-oriented programming
- Finding out about the basis for the Cocoa application framework
- Programming in Objective-C

The Development Environment

Most object-oriented development environments consist of at least three parts:

- A library of objects
- A set of development tools
- An object-oriented programming language and support library

Cocoa is an extensive library. It includes several software frameworks containing definitions for objects that you can use “off the shelf??? or adapt to your program’s needs. These include the Foundation framework, the Application Kit framework (for building a graphical user interface), and others.

Mac OS X also includes development tools for putting together applications. There’s Interface Builder, a program that lets you design an application graphically and assemble its user interface interactively, and Xcode, a project-management program that provides graphical access to the compiler, the debugger, documentation, a program editor, and other tools.

This document is about the third component of the development environment—the programming language and its runtime environment. All Cocoa frameworks are written in the Objective-C language. To get the benefit of the frameworks, applications must use either Objective-C or a language bridged to Objective-C, such as Java.

Objective-C is defined as set of extensions to the C language. It’s designed to give C full object-oriented programming capabilities, and to do so in a simple and straightforward way. Its additions to C are few and are mostly based on Smalltalk, one of the first object-oriented programming languages.

This document both introduces the object-oriented model that Objective-C is based upon and fully documents the language. It concentrates on the Objective-C extensions to C, not on the C language itself. There are many good books available on C; this book doesn’t attempt to duplicate them.

Because this isn’t a document about C, it assumes some prior acquaintance with that language. However, it doesn’t have to be an extensive acquaintance. Object-oriented programming in Objective-C is sufficiently different from procedural programming in standard C that you won’t be hampered if you’re not an experienced C programmer.

Why Objective-C

The Objective-C language was chosen for the Cocoa framework for a variety of reasons. First and foremost, it’s an object-oriented language. The kind of functionality that’s packaged in the Cocoa frameworks can only be delivered through object-oriented techniques. This document explains the operation of the frameworks and how you can take advantage of them.

Second, because Objective-C is an extension of standard ANSI C, existing C programs can be adapted to use the software frameworks without losing any of the work that went into their original development. Since Objective-C incorporates C, you get all the benefits of C when working within Objective-C. You can choose when to do something in an object-oriented way (define a new class, for example) and when to stick to procedural programming techniques (define a structure and some functions instead of a class).

Moreover, Objective-C is a simple language. Its syntax is small, unambiguous, and easy to learn. Object-oriented programming, with its self-conscious terminology and emphasis on abstract design, often presents a steep learning curve to new recruits. A well-organized language like Objective-C can make becoming a proficient object-oriented programmer that much less difficult. The size of this book is a testament to the simplicity of Objective-C. It’s not a big book.

Compared to other object oriented languages based on C, Objective-C is very dynamic. The compiler preserves a great deal of information about the objects themselves for use at runtime. Decisions that otherwise might be made at compile time can be postponed until the program is running. This gives Objective-C programs unusual flexibility and power. For example, Objective-C’s dynamism yields two big benefits that are hard to get with other nominally object-oriented languages:

- Objective-C supports an open style of dynamic binding, a style that can accommodate a simple architecture for interactive user interfaces. Messages are not necessarily constrained by either the class of the receiver or the method selector, so a software framework can allow for user choices at runtime and permit developers freedom of expression in their design. (Terminology like “dynamic binding,??? “message,??? “class,??? “receiver,??? and “selector??? are explained in due course in this document.)

-))Objective-C’s((dynamism enables the construction of sophisticated development tools. An interface to the runtime system provides access to information about running applications, so it’s possible to develop tools that monitor, intervene, and reveal the underlying structure and activity of Objective-C applications.

Note: Using Objective-C outside of a completely Object Oriented environment severely limits the benefits

Preliminary Overview

Executive Overview

Main Overview

Conclusion

Section Two: The Basics