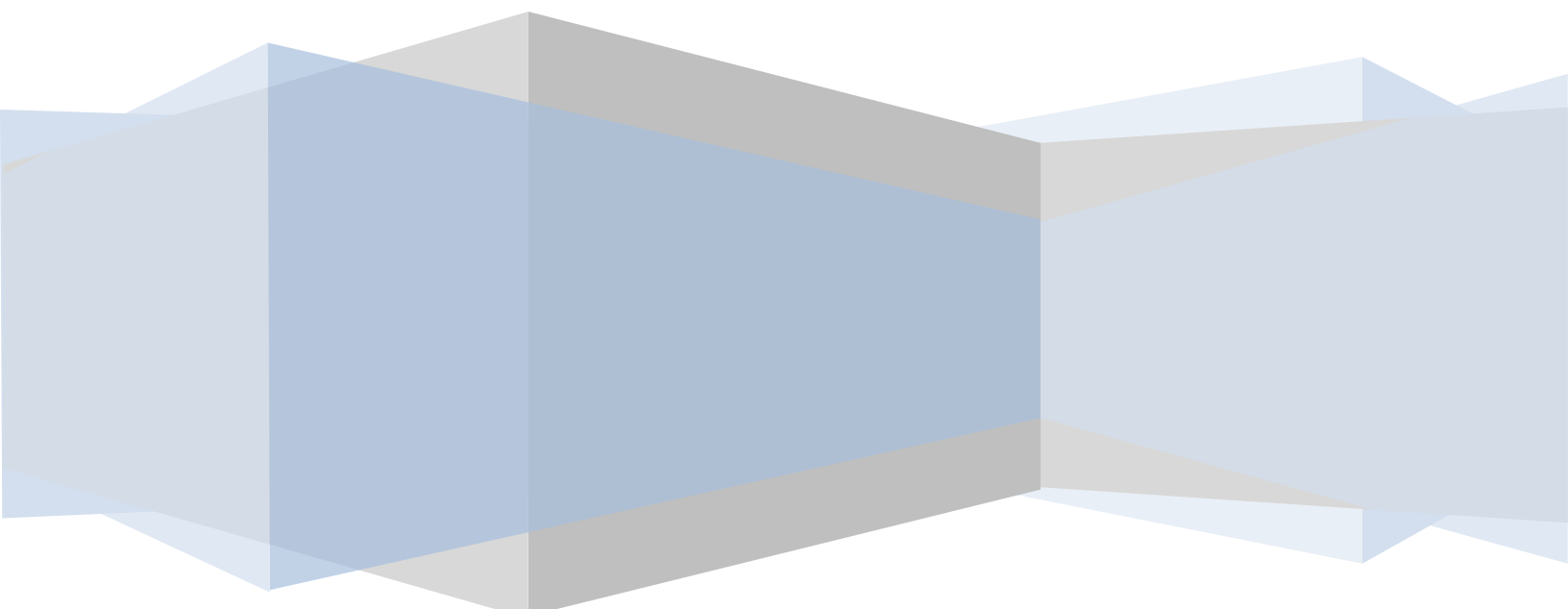


# Objective-C



# Essentials

# Objective-C 2.0 Essentials



Objective-C 2.0 Essentials – Version 1.0

© 2009 Payload Media. This eBook is provided for personal use only. Unauthorized use, reproduction and/or distribution strictly prohibited. All rights reserved.

The content of this book is provided for informational purposes only. Neither the publisher nor the author offers any warranties or representation, express or implied, with regard to the accuracy of information contained in this book, nor do they accept any liability for any loss or damage arising from any errors or omissions.

Find more eBooks online at <http://www.eBookFrenzy.com>.

## Table of Contents

Chapter 1. About Objective-C Essentials .....	13
1.1 Why are you reading this? .....	13
1.2 Supported Platforms .....	13
Chapter 2. The History of Objective-C .....	15
2.1 The C Programming Language .....	15
2.2 The Smalltalk programming Language.....	15
2.3 C meets Smalltalk .....	15
2.4 Objective-C and Apple.....	16
Chapter 3. Installing Xcode and Compiling Objective-C on Mac OS X.....	17
3.1 Installing Xcode on Mac OS X.....	17
3.2 Starting Xcode .....	19
3.3 Starting a New Project .....	20
3.4 Writing an Objective-C Application with Xcode.....	21
3.5 Compiling Objective-C from the Command Line .....	22
Chapter 4. Installing and using GNUstep and Objective-C on Windows.....	24
4.1 Downloading the GNUstep Packages.....	24
4.2 Installing MinGW and GNUstep on Windows .....	24
4.3 Running the GNUstep Shell .....	25
4.4 Testing the Installation.....	26
Chapter 5. Installing and Using GNUstep and Objective-C on Linux .....	28
5.1 Installing GNUstep on Ubuntu .....	28
5.2 Compiling Objective-C Code.....	29
Chapter 6. Building and Installing GNUstep on Linux.....	31
6.1 Installing gcc and Objective-C Support on Linux.....	31
6.2 Package Dependencies.....	31
6.3 Obtaining the GNUstep Source Code.....	32
6.4 Configuring the Build Process .....	32

6.5	Building GNUstep .....	33
6.6	Testing the Objective-C and GNUstep Installation .....	34
6.7	Compiling Objective-C Code.....	35
Chapter 7. Objective-C 2.0 Data Types .....		37
7.1	int Data Type .....	38
7.2	char Data Type .....	39
7.2.1	Special Characters/Escape Sequences.....	39
7.3	float Data Type .....	40
7.4	double Data Type .....	40
7.5	id Data Type.....	40
7.6	BOOL Data Type .....	40
7.7	Objective-C Data Type Qualifiers .....	41
7.7.1	long.....	41
7.7.2	long long.....	41
7.7.3	short .....	41
7.7.4	signed / unsigned.....	42
7.8	Summary .....	42
Chapter 8. Working with Variables and Constants in Objective-C.....		43
8.1	What is an Objective-C Variable.....	43
8.2	What is an Objective-C Constant?.....	44
8.3	Type Casting Objective-C Variables.....	45
Chapter 9. Objective-C Operators and Expressions .....		47
9.1	What is an Expression? .....	47
9.2	The Basic Assignment Operator .....	47
9.3	Objective-C Arithmetic Operators.....	48
9.4	Compound Assignment Operators.....	49
9.5	Increment and Decrement Operators.....	49
9.6	Comparison Operators.....	50

9.7	Boolean Logical Operators .....	51
9.8	The Ternary Operator.....	52
9.9	Bitwise Operators.....	53
9.9.1	Bitwise AND .....	54
9.9.2	Bitwise OR.....	54
9.9.3	Bitwise XOR.....	55
9.9.4	Bitwise Left Shift .....	55
9.9.5	Bitwise Right Shift .....	56
9.10	Compound Bitwise Operators.....	57
Chapter 10.	Objective-C 2.0 Operator Precedence .....	58
10.1	An Example of Objective-C Operator Precedence .....	58
10.2	Objective-C Operator Precedence and Associativity .....	59
10.3	Overriding Operator Precedence .....	61
Chapter 11.	Commenting Objective-C Code.....	62
11.1	Why Comment your Code? .....	62
11.2	Single Line Comments .....	62
11.3	Multi-line Comments .....	63
11.4	Summary .....	64
Chapter 12.	Objective-C Flow Control with <i>if</i> and <i>else</i> .....	65
12.1	Using the if Statement .....	65
12.2	Using if ... else ... Statements .....	66
12.3	Using if ... else if ... Statements .....	67
12.4	Summary .....	67
Chapter 13.	The Objective-C <i>switch</i> Statement.....	68
13.1	Why Use a switch Statement? .....	68
13.2	Using the switch Statement Syntax .....	69
13.3	A switch Statement Example .....	70
13.4	Explaining the Example .....	71

13.5	Combining case Statements.....	72
Chapter 14.	Objective-C Looping - The <i>for</i> Statement .....	73
14.1	Why Use Loops? .....	73
14.2	Objective-C Loop Variable Scope .....	75
14.3	Creating an Infinite <i>for</i> Loop .....	76
14.4	Breaking Out of a <i>for</i> Loop .....	76
14.5	Nested <i>for</i> Loops .....	77
14.6	Breaking from Nested Loops.....	77
14.7	Continuing <i>for</i> Loops .....	78
14.8	Using <i>for</i> Loops with Multiple Variables .....	79
Chapter 15.	Objective-C Looping with <i>do</i> and <i>while</i> Statements.....	81
15.1	The Objective-C while Loop.....	81
15.2	Objective-C <i>do ... while</i> loops.....	82
15.3	Breaking from Loops .....	82
15.4	The <i>continue</i> Statement.....	83
Chapter 16.	An Overview of Objective-C Object Oriented Programming.....	85
16.1	What is an Object? .....	85
16.2	What is a Class? .....	85
16.3	Declaring an Objective-C Class Interface .....	85
16.4	Adding Instance Variables to a Class.....	86
16.5	Define Class Methods.....	87
16.6	Declaring an Objective-C Class Implementation .....	88
16.7	Declaring, Initializing and Releasing a Class Instance .....	90
16.8	Calling Methods and Accessing Instance Data.....	91
16.9	Creating the Program Section .....	91
16.10	Bringing it all Together .....	92
16.11	Structuring Object-Oriented Objective-C Code.....	95
Chapter 17.	Writing Objective-C Class Methods .....	98

17.1	Instance and Class Methods.....	98
17.2	Creating a New Class Method .....	98
17.3	The @interface Section .....	99
17.4	The @implementation Section .....	99
17.5	The main() Function .....	100
Chapter 18.	Objective-C - Data Encapsulation, Synthesized Accessors and Dot Notation .....	103
18.1	Data Encapsulation.....	103
18.2	Synthesized Accessor Methods.....	103
18.3	Direct Access to Encapsulated Data.....	105
18.4	Objective-C and Dot Notation .....	105
18.5	Controlling Access to Instance Variables .....	106
Chapter 19.	Objective-C Inheritance .....	108
19.1	Inheritance, Classes and Subclasses.....	108
19.2	An Objective-C Inheritance Example .....	108
19.3	Extending the Functionality of a Subclass.....	111
19.4	Overriding Inherited Methods .....	112
Chapter 20.	Pointers and Indirection in Objective-C.....	115
20.1	How Variables are Stored.....	115
20.2	An Overview of Indirection .....	116
20.3	Indirection and Objects .....	119
20.4	Indirection and Object Copying.....	119
Chapter 21.	Objective-C Dynamic Binding and Typing with the id Type.....	120
21.1	Static Typing vs Dynamic Typing .....	120
21.2	Dynamic Binding.....	121
21.3	Polymorphism .....	123
Chapter 22.	Objective-C Variable Scope and Storage Class .....	124
22.1	Variable Scope.....	124
22.2	Block Scope .....	124

22.3	Function Scope .....	126
22.4	Global Scope.....	128
22.5	File Scope.....	131
22.6	Variable Storage Class .....	131
Chapter 23. An Overview of Objective-C Functions .....		133
23.1	What is a Function?.....	133
23.2	How to Declare an Objective-C Function .....	133
23.3	The main() Function .....	134
23.4	Calling an Objective-C Function .....	134
23.5	Function Prototypes.....	135
23.6	Function Scope and the static Specifier .....	138
23.7	Static Variables in Functions .....	138
Chapter 24. Objective-C Enumerators.....		141
24.1	Why Use Enumerators .....	141
24.2	Declaring an Enumeration.....	141
24.3	Creating and Using an Enumeration .....	142
24.4	Enumerators and Variable Names .....	142
Chapter 25. An Overview of the Objective-C Foundation Framework.....		144
25.1	The Foundation Framework.....	144
25.2	Including the Foundation Headers.....	144
25.3	Finding the Foundation Framework Documentation .....	145
Chapter 26. Working with String Objects in Objective-C .....		146
26.1	Strings without NSString .....	146
26.2	Declaring Constant String Objects .....	147
26.3	Creating Mutable and Immutable String Objects .....	148
26.4	Getting the Length of a String.....	149
26.5	Copying a String.....	149
26.6	Searching for a Substring .....	151

26.7	Replacing Parts of a String.....	152
26.8	String Search and Replace .....	153
26.9	Deleting Sections of a String .....	153
26.10	Extracting a Subsection of a String .....	153
26.11	Inserting Text into a String .....	154
26.12	Appending Text to the End of a String .....	154
26.13	Comparing Strings.....	155
26.14	Checking for String Prefixes and Suffixes .....	156
26.15	Converting to Upper or Lower Case .....	157
26.16	Converting Strings to Numbers .....	158
26.17	Converting a String Object to ASCII .....	159
Chapter 27.	Understanding Objective-C Number Objects .....	160
27.1	Creating and Initializing NSNumber Objects.....	160
27.2	Getting the Value of a Number Object .....	161
27.3	Comparing Number Objects.....	162
27.4	Getting the Number Object Value as a String.....	163
Chapter 28.	Working with Objective-C Array Objects .....	165
28.1	Mutable and Immutable Arrays .....	165
28.2	Creating an Array Object.....	165
28.3	Finding out the Number of Elements in an Array .....	166
28.4	Accessing the Elements of an Array Object .....	166
28.5	Accessing Array Elements using Fast Enumeration.....	167
28.6	Adding Elements to an Array Object.....	167
28.7	Inserting Elements into an Array.....	168
28.8	Deleting Elements from an Array Object .....	169
28.9	Sorting Array Objects .....	170
Chapter 29.	Objective-C Dictionary Objects.....	171
29.1	What are Dictionary Objects? .....	171

29.2	Creating Dictionary Objects .....	171
29.3	Initializing and Adding Entries to a Dictionary Object .....	171
29.4	Getting an Entry Count.....	172
29.5	Accessing Dictionary Entries .....	173
29.6	Removing Entries from a Dictionary Object.....	173
Chapter 30.	Working with Directories in Objective-C .....	175
30.1	The Objective-C NSFileManager, NSFileHandle and NSData Classes .....	175
30.2	Understanding Pathnames in Objective-C.....	175
30.3	Creating an NSFileManager Instance Object .....	176
30.4	Identifying the Current Working Directory .....	176
30.5	Changing to a Different Directory .....	176
30.6	Creating a New Directory .....	177
30.7	Deleting a Directory .....	177
30.8	Renaming or Moving a Directory .....	178
30.9	Getting a Directory File Listing .....	178
30.10	Getting the Attributes of a File or Directory .....	179
Chapter 31.	Working with Files in Objective-C.....	181
31.1	Creating an NSFileManager Instance .....	181
31.2	Checking if a File Exists.....	181
31.3	Comparing the Contents of Two Files .....	181
31.4	Checking if a File is Readable/Writable/Executable/Deletable .....	182
31.5	Moving/Renaming a File .....	183
31.6	Copying a File .....	183
31.7	Removing a File .....	183
31.8	Creating a Symbolic Link .....	184
31.9	Reading and Writing Files with NSFileManager .....	184
31.10	Working with Files using the NSFileHandle Class.....	185
31.11	Creating an NSFileHandle Object .....	185

31.12	NSFileHandle File Offsets and Seeking .....	186
31.13	Reading Data from a File .....	187
31.14	Writing Data to a File .....	188
31.15	Truncating a File .....	189
Chapter 32. Constructing and Manipulating Paths with NSStringUtilities .....		190
32.1	The Anatomy of a Path.....	190
32.2	Finding a Temporary Directory .....	190
32.3	Getting the Current User's Home Directory .....	191
32.4	Getting the Home Directory of a Specified User.....	191
32.5	Extracting the Filename from a Path .....	191
32.6	Extracting the Filename Extension.....	192
32.7	Standardizing a Path.....	192
32.8	Extracting the Components of a Path .....	193
Chapter 33. Copying Objects in Objective-C.....		195
33.1	Objects and Pointers .....	195
33.2	Copying an Object in Objective-C using the <NSCopying> Protocol.....	195
33.3	<NSCopying> Protocol and copyWithZone Method Implementation .....	196
33.4	Performing a Deep Copy .....	198
Chapter 34. Using Objective-C Preprocessor Directives.....		202
34.1	The #define Statement.....	202
34.2	Creating Macros with the #define Statement.....	203
34.3	Changing the Objective-C Language with #define .....	203
34.4	Undefining a Definition with #undef.....	205
34.5	Conditional Compilation.....	205
34.6	The #import Directive .....	207

## Chapter 1. About Objective-C Essentials

---

### 1.1 Why are you reading this?

On the surface this sounds like an odd opening sentence for a programming book. After all, if this were a book about JavaScript or PHP I'd be safe in assuming that you planned to develop some kind of web site or web application. Similarly, if this were a Visual Basic book it'd be a good bet that you had plans to write a Windows application. Indeed, had I asked this question a few years ago, I could have guessed with a reasonable level of confidence that you wanted to learn Objective-C in order to develop some software to run on Apple's Mac OS X operating system. Now, however, there is a greater likelihood that you plan to develop an application to run on the iPhone.

The iPhone, after all, runs a special version of Mac OS X. Given that Objective-C is the programming language of choice for this operating system it should come as no surprise that before you can develop iPhone applications you first need to learn how to program in Objective-C.

The objective of this book is to teach the skills necessary to program in Objective-C using a style that is easy to follow, rich in examples and accessible to those who have never used Objective-C before. Topics covered include the fundamentals of Objective-C such as variables, looping and flow control. Also included are details of object oriented programming, working with files and memory and the Objective-C Foundation framework.

Those who have developed using other programming languages such as C, C++, C# or Java will find much about Objective-C that is familiar. That said, there are aspects of the language syntax that are unique to Objective-C. Even experienced programmers should therefore expect to spend some time transitioning to this increasingly popular programming language before embarking on a major development project.

Whatever your background and experience, we have worked hard to make this book as useful and helpful as possible as you traverse the Objective-C learning curve.

### 1.2 Supported Platforms

After all this talk about Mac OS X and the iPhone, it is important to note that Objective-C is not confined to Apple's operating systems. In fact, Objective-C is available on a wide range of platforms including Linux, NetBSD, OpenBSD, FreeBSD, Solaris and Windows in the form of the open source *GNUstep* environment. This means that anyone with access to a GNUstep supported platform can learn Objective-C for free, though if your ultimate objective is to

develop for the iPhone, you will at some point need access to an Intel based Mac computer system.

Perhaps one key advantage to using a Mac OS X system for learning Objective-C comes in the form of access to Apple's Xcode development environment. Other than references to Xcode in early chapters, however, the remainder of this book is intended to be as platform agnostic as possible.

## Chapter 2. The History of Objective-C

---

Before learning the intricacies of a new programming language it is often worth taking a little time to learn about the history and legacy of that language. In this chapter of *Objective-C 2.0 Essentials* we will provide a brief overview of the origins of Objective-C and the business history that ultimately led to it becoming the programming language of choice for both Mac OS X and the iPhone.

### 2.1 The C Programming Language

Objective-C is based on a programming language called, quite simply, *C*. The origins of the *C* programming language can be traced back nearly 40 years to two engineers named Dennis Ritchie and Ken Thompson working at what is now known as AT&T Bell Labs. At the time, the two were working on developing the UNIX operating system on PDP-7 and PDP-11 systems. After attempts to write this operating system using assembly language (essentially using sequences of instruction codes understood by the processor), it was decided that a higher level, more programmer friendly programming language was required to handle the complexity of an operating system such as UNIX. The first attempt was a language called *B*. The *B* language, which was based on a language called *BCPL*, was found to be lacking. Taking the next initial from the *BCPL* name, the *C* language was created and subsequently used to write much of the UNIX operating system kernel and infrastructure. As far as we can tell, *C* was so successful that new languages named *P* and *L* never needed to be created.

### 2.2 The Smalltalk programming Language

The *C* programming language is what is known as a *procedural* language. As such, this means that it lacks features such as object oriented programming. Object oriented programming advocates the creation of small, clearly defined code objects that can be assembled and reused to create more complex systems.

An early attempt at an object-oriented programming language was developed by a team including Alan Kay (who later went to work for Apple) and Dan Ingalls at Xerox PARC (Palo Alto Research Center) in the 1970s. This language is known as Smalltalk.

### 2.3 C meets Smalltalk

An interesting history lesson so far, but what does this have to with Objective-C? Well, in the 1980s, two developers named Brad Cox and Tom Love extended the *C* programming language to support the object oriented features of Smalltalk. This melding of languages ultimately culminated in the creation of Objective-C. Objective-C was subsequently adopted by the Free Software Foundation and released under the terms of the GNU Public License (GPL).

## 2.4 Objective-C and Apple

To understand how Objective-C, a language based on two 40 year old programming languages, ended up being the language of choice on Mac OS X and the latest cutting edge smart phones from Apple it is necessary to move away from technology for a while and talk about business.

In the 1980s Steve Jobs and Steve Wozniak founded Apple Computer. After many years of success, Steve Jobs hired a marketing wizard from PepsiCo called John Sculley to help take Apple to the next level of business success. To cut a long story short, a boardroom battle ensued and Steve Jobs got pushed out of the company (for the long version of the story pick up a used copy of John Sculley's book *Odyssey: From Pepsi to Apple*) leaving John Sculley in charge.

After leaving Apple, Jobs started a new company called NeXT to design an entirely new generation of computer system. The operating system developed by NeXT to run on these computers was called NeXTstep. In order to develop NeXTstep, NeXT licensed Objective-C. NeXT subsequently joined forces with Sun Microsystems to create a standardized version of NeXTstep named OPENstep which the Free Software Foundation then adopted as GNUstep.

During the 1990s, John Sculley left Apple and a procession of new CEOs came and went. During this time, Apple had been losing market share and struggling to come out with a new operating system to replace the aging Mac OS. After a number of failed attempts and partnerships, it was eventually decided that rather than try to write a new operating system, Apple should acquire a company that already had one. During Gil Amelio's brief reign as CEO, a shortlist of two companies was drawn up. One was a company called Be, Inc. founded by a former Apple employee named Jean-Louis Gassée, and the other was NeXT.

Ultimately, NeXT was selected and Steve Jobs once again joined Apple. In another boardroom struggle (another long story as outlined in Gil Amelio's book *On the Firing Line: My 500 Days at Apple*) Steve Jobs pushed out Gil Amelio and once again became CEO of the company he had founded all those years ago.

The rest, as they say, is history. NeXTStep formed the foundation of what became Mac OS X, bringing with it Objective-C. Mac OS X was subsequently modified to provide the operating system for the spectacularly successful iPhone.