Chapter 8

A Quick Tour of the Foundation Kit

You’ve already seen that Objective-C is a pretty nifty language, and we haven’t even finished exploring all the features it has to offer. For now, we’re going to take a quick side trip and have a look at Cocoa’s Foundation framework. Although strictly part of Cocoa and not built in to Objective-C, the Foundation framework is so important that we thought it worth exploring in this book.

As you saw in Chapter 2, Cocoa is actually composed of two different frameworks: Foundation and Application Kit. The Application Kit has all the user interface objects and high-level classes. You’ll get a taste of the AppKit (as the cool kids call it) in Chapter 14.

Cocoa’s Foundation framework has a bunch of useful low-level, data-oriented classes and types. We’ll be visiting a number of these, such as NSString, NSArray, NSEnumerator, and NSNumber. Foundation has more than a hundred classes, all of which you can explore by looking at the documentation installed with Xcode. These documents live at /Developer/ADC Reference Library/documentation/index.html.

Before we continue, here’s a note about the projects for this chapter and for the rest of this book. We’ll still be making Foundation tool projects, but we’ll leave in the boilerplate code, which follows (slightly reformatted to fit on this page):

#import <Foundation/Foundation.h>

int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool
    = [[NSAutoreleasePool alloc] init];
Take a look through this code. main() starts by creating (via alloc) and initializing (via init) an NSAutoreleasePool. The pool is drained at the end. This is a sneak preview of Cocoa memory management, which we’ll discuss in the next chapter. For now, please just nod, smile, and leave the NSAutoreleasePool stuff in there. If you take it out, you won’t hurt yourself, but you’ll get some very strange messages when you run your programs.

Some Useful Types

Before digging into real live Cocoa classes, let’s take a look at some structs that Cocoa provides for our benefit.

Home on the Range

The first structure is NSRange:

typedef struct _NSRange {
    unsigned int location;
    unsigned int length;
} NSRange;

This structure is used to represent a range of things, usually a range of characters in a string or a range of items in an array. The location field holds the starting position of the range, and length is the number of elements in the range. For the string “Objective-C is a cool language”, the word “cool” can be described by the range that starts at location 17 and has length 4. location can have the value NSNotFound to indicate that the range doesn’t refer to anything, probably because it’s uninitialized.

You can make a new NSRange in three different ways. First, you can assign the field values directly:

NSRange range;
range.location = 17;
range.length = 4;