# Interacting with the Runtime

Objective-C programs interact with the runtime system at three distinct levels: through Objective-C source code; through methods defined in the `NSObject` class of the Foundation framework; and through direct calls to runtime functions.

## Objective-C Source Code

For the most part, the runtime system works automatically and behind the scenes. You use it just by writing and compiling Objective-C source code.

When you compile code containing Objective-C classes and methods, the compiler creates the data structures and function calls that implement the dynamic characteristics of the language. The data structures capture information found in class and category definitions and in protocol declarations; they include the class and protocol objects discussed in Defining a Class and Protocols in *The Objective-C Programming Language*, as well as method selectors, instance variable templates, and other information distilled from source code. The principal runtime function is the one that sends messages, as described in Messaging. It's invoked by source-code message expressions.

## NSObject Methods

Most objects in Cocoa are subclasses of the `NSObject` class, so most objects inherit the methods it defines. (The notable exception is the `NSProxy` class; see Message Forwarding for more information.) Its methods therefore establish behaviors that are inherent to every instance and every class object. However, in a few cases, the `NSObject` class merely defines a template for how something should be done; it doesn't provide all the necessary code itself.

For example, the `NSObject` class defines a `description` instance method that returns a string describing the contents of the class. This is primarily used for debugging—the GDB `print-object` command prints the string returned from this method. `NSObject`'s implementation of this method doesn't know what the class contains, so it returns a string with the name and address of the object. Subclasses of `NSObject` can implement this method to return more details. For example, the Foundation class `NSArray` returns a list of descriptions of the objects it contains.

Some of the `NSObject` methods simply query the runtime system for information. These methods allow objects to perform introspection. Examples of such methods are the `class` method, which asks an object to identify its class; `isKindOfClass:` and `isMemberOfClass:`, which test an object's position in the inheritance hierarchy; `respondsToSelector:`, which indicates whether an object can accept a particular message; `conformsToProtocol:`, which indicates whether an object claims to implement the methods defined in a specific protocol; and `methodForSelector:`, which provides the address of a method's implementation. Methods like these give an object the ability to introspect about itself.

## Runtime Functions

The runtime system is a dynamic shared library with a public interface consisting of a set of functions and data structures in the header files located within the directory `/usr/include/objc`. Many of these functions allow you to use plain C to replicate what the compiler does when you write Objective-C code. Others form the basis for functionality exported through the methods of the `NSObject` class. These functions make it possible to develop other interfaces to the runtime system and produce tools that augment the development environment; they're not needed when programming in Objective-C. However, a few of the runtime functions might on occasion be useful

when writing an Objective-C program. All of these functions are documented in *Objective-C Runtime Reference*.

---