

## Chapter 2

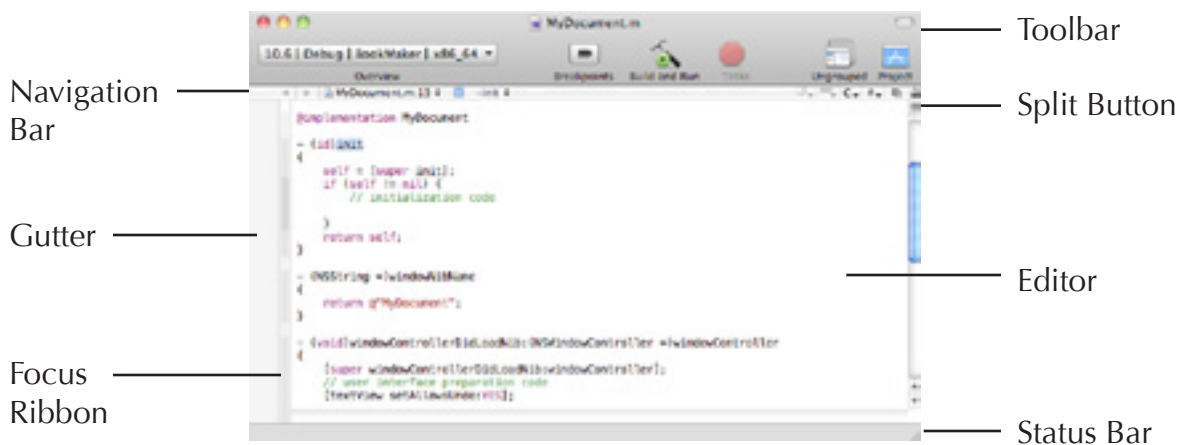
## Writing Source Code

After creating a project, the next thing you're going to do in Xcode is write source code. This chapter focuses on topics to improve your code writing experience in Xcode.

## The Editor Window

The editor window is where you write your source code. Double-clicking a source code file in the project window opens an editor window. Figure 2.1 shows a typical editor window, which contains seven parts.

- Toolbar
- Status bar
- Navigation bar
- Editor
- Gutter
- Focus ribbon
- Split button



### Figure 2.1

## Editor window

## Toolbar

The toolbar, which runs along the top of the editor window, contains commonly used controls. The Overview pop-up menu dominates the left side of the toolbar. Use the Overview menu to set the following information for the project:

- The active SDK, which is the SDK Xcode uses to build your project.
- The active build configuration, which is the build configuration Xcode uses to build your project. Each Xcode project comes with two build configurations: Debug and Release.
- The active target, which is the target Xcode creates when it builds your project.
- The active executable, which is the executable Xcode creates when it builds your project.
- The active architecture, which can be 32-bit Intel (i386), 64-bit Intel (x86\_64), or PowerPC (ppc).

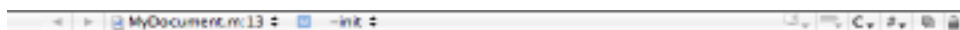
The editing mode button on the right side of the toolbar is the most interesting toolbar button for source code editing. There are two editing modes you can toggle by clicking the button: ungrouped (multiple window) and grouped (single window). The initial editing mode is ungrouped. Every time you open a source code file, the file opens in its own window. If you want to reduce window clutter, you will want to switch the editing mode to grouped. Click the editing mode button to change the editing mode to grouped. Grouped editing mode means there's only one editor window open at a time. When you open another file Xcode places the new file's contents in the window.

## Status Bar

The status bar doesn't do much when you're editing source code. It shows you the status of Xcode activities like building and debugging programs.

## Navigation Bar

The navigation bar, shown in Figure 2.2, contains controls to quickly reach areas of the current file and quickly change files. It has four different areas. On the left edge of the bar are back and forward buttons, which work similarly to an Internet browser's back and forward buttons.



**Figure 2.2**

Navigation bar

To the right of the back and forward buttons is the recently viewed files list. The list displays the file you're viewing and the line number. The pop-up menu next to the file name and line number shows the list of recently viewed files. The recently viewed files list helps when you edit a file, edit a few more files, and want to go back to the first file you were editing.

Next to the recently viewed files list is the function list. It shows the function you're at in the file. There's a pop-up menu containing all the functions in the file. Use the function list to reach a function in the file quickly.

At the right edge of the navigation bar are six buttons. Clicking the leftmost button lists the bookmarks you've set in the file. Clicking the second button lists the breakpoints you've set in the file.

The third button is the class navigation button. Clicking it shows any superclasses or subclasses the file has. Selecting a superclass or subclass opens the class's header file. Clicking the fourth button lists all the files the current file includes. Selecting a file from the list opens that file.

The fifth button is the Go To Counterpart button, which works with C-based languages. If the current file is an implementation file, clicking the Go To Counterpart button opens the header file for the current file. If the current file is a header file, clicking the Go To Counterpart button opens the implementation file. Clicking the sixth button locks and unlocks files. Locking a file prohibits anyone from editing the file.

## **Editor and Gutter**

The editor is where you edit source code. You've used word processors and text editors before. Xcode's editor isn't radically different from editors you've previously used. The gutter runs along the left edge of the window. Its main use is for setting breakpoints, which I will discuss in Chapter 5 "Debugging with Xcode".

## **Focus Ribbon**

The focus ribbon lets you highlight and fold blocks of code. Moving the mouse cursor over a block in the focus ribbon highlights the block of code. Clicking the focus ribbon folds the block, which means the code inside the block is hidden in the editor. When you fold a block of code, a triangle appears in the focus ribbon. Clicking the triangle unfolds the block of code in the editor.

## Split Button

The split button is wedged between the navigation bar and the vertical scroll bar. Clicking the split button splits the editor window, giving you two of everything except the toolbar and status bar. Splitting the editor window lets you look at code in two different files in the same window. If you want more than two views, click the split button again to create an additional view.

When you split the editor, a second button appears below the split button. Clicking that button reverses the split, eliminating the second editor view.

## Code Completion

Code completion tells Xcode to finish the names of your program's functions and variable names, saving you from having to type the whole name. It also helps reduce typographical errors in your code.

For code completion to work, your project must have indexing enabled. Open Xcode's Code Sense preferences and make sure the Indexing: Enable for all projects checkbox is selected. When you enable indexing, Xcode creates a project index that keeps track of all the project symbols, such as variable names, function names, and constants. Xcode uses this index for many things, one of which is code completion. If you turn off indexing, you disable code completion as well.

## Using the Completion List

To perform code completion, start typing the function or variable name and press the Esc key. A pop-up menu known as a completion list will open. The completion list contains all the possible matches. As you type more of the function or variable name, the completion list updates itself to reflect what you typed.

Select an entry from the completion list and press the Return key to complete the code. If you don't want to complete the code, press the Esc key to close the completion list.

## Automatic Suggestion

If you don't want to manually invoke the completion list to use code completion, tell Xcode to automatically suggest completions. Open Xcode's Code Sense preferences to turn on automatic suggestion.

Use the Automatically suggest pop-up menu to open the completion list automatically. Choose Immediate if you want suggestions to appear as you type. If you want to delay the suggestion, choose With Delay. Specify the amount of time that must pass before Xcode offers suggestions.

When you tell Xcode to suggest completions, one completion appears in the editor instead of a completion list. If the completion suggestion is what you want, press the Tab key. To see other completions, press Control-period. Use Control-period to cycle through the possible completions until you find the one you want.

## Customizing Code Completion

To customize Xcode's code completion behavior, open Xcode's Code Sense preferences. Xcode has two checkboxes to customize how code completion works. Selecting the Show arguments in pop-up list checkbox tells Xcode to add arguments for each function that appears in the completion list. If you don't select the checkbox, the completion list shows the function name only. Adding arguments helps when you have functions with the same name but different arguments. Without the arguments you would have no way of knowing the function you were completing.

When you select the Insert argument placeholders for completions checkbox, Xcode inserts placeholders for the function's arguments when it completes a function. If you don't select the checkbox, Xcode completes the function name only. Here's how Xcode would complete the OpenGL function `glVertex3f()` with the function name only:

```
glVertex3f
```

And here's how Xcode would complete the `glVertex3f()` function with argument placeholders:

```
glVertex3f(<#GLfloat x#>,<#GLfloat y#>,<#GLfloat z#>)
```

If you tell Xcode to insert argument placeholders, press Control-slash to move to the next argument in the function. The Control-slash combination makes it easy to replace the argument placeholders with your own arguments.

## Customizing Code Editing

Programmers have different opinions about what makes code easy to read. Some people prefer black text on a white background while others prefer white text on a black background. Some people indent code four spaces, others prefer to indent two spaces, and others prefer to indent eight spaces. Some people want to see line numbers while others don't.

Xcode has many preferences to customize your experience editing source code. The preferences for customizing source code editing are the following:

- Text Editing
- Fonts and Colors
- Indentation
- Key Bindings

## Text Editing Preferences

The text editing preferences is where you set miscellaneous text editing options. Some options you can set include the following:

- What character is generated when you press the Return key to end a line of code: a carriage return or a line feed. Mac OS (Mac OS 9, which predates Mac OS X), Unix, and Windows end lines differently. Mac OS generates a carriage return. Unix generates a line feed. Windows generates a carriage return and a line feed. Mac OS X uses Unix line endings.
- Showing the gutter on the left side of the editor window.
- Showing line numbers in the editor.
- Showing the code folding ribbon.
- The file encoding to save files, which is initially 8-bit Unicode.

## Fonts and Colors Preferences

Xcode gives different areas of your code their own text color to make reading the code easier. The Fonts and Colors preferences is where you set the text color, font, and font size for your code.

### Color Themes

Xcode comes with several color themes. Use the Color Theme pop-up menu to pick a theme. If you don't like any of Xcode's themes, choose the best matching theme and click the Duplicate button. You will be asked for a name for the theme. Use your theme to change the source code's font and color.

Below the Color Theme pop-up menu is the category list. Looking at the category list, you will see lots of categories, including comments, class names, language keywords, strings, and numbers. To change the text color, select a category and double-click the color column.

To change the font, select a category and double-click. When you look at Xcode's color themes, you'll notice each theme uses one font. If you want to change the font, select multiple categories before making the change. Changing the font for each individual category can get tedious.

## **Fonts and Colors Checkboxes**

There are four checkboxes underneath the category list.

- Use syntax-based formatting
- Color indexed symbols
- Use colors when printing
- Copy colors and fonts

Selecting the Use syntax-based formatting checkbox activates syntax coloring. If you find Xcode's text editor to be too slow, turning off syntax-based formatting may make text editing faster. You can set syntax coloring for individual files by choosing View > Syntax Coloring.

The Color indexed symbols checkbox specifies whether to use the project's Code Sense index to assign symbols to the categories for fonts and colors. If it's turned off, Xcode uses the file's programming language to assign symbols to categories. I didn't notice a difference with the checkbox selected or deselected. Deselecting the checkbox can make text editing faster.

The Use colors when printing checkbox determines whether you want to keep the syntax coloring on when printing. Printing will be faster if you turn it off.

If you select the Copy colors and fonts checkbox, Xcode copies the syntax coloring when doing cut and paste with other applications. It does not affect Xcode; you will still get syntax coloring when you cut and paste code inside Xcode if the Copy colors and fonts checkbox is not selected. Deselecting the checkbox helps if you're cutting and pasting code to a blog or a message board, and you want the fonts to match the site you're pasting your code to.

## **Indentation Preferences**

Indenting your source code makes the code easier to read. Xcode's indentation preferences let you control how Xcode indents your code. Xcode's indentation preferences contains three sections: Tabs, Line Wrapping, Syntax-Aware Indenting.

## Tabs

The tabs section controls what happens when you press the Tab key. You can specify how many spaces to move ahead when you hit the Tab key and specify whether a tab character or spaces are inserted. Inserting a tab character or spaces is the subject of intense debate.

## Line Wrapping

If you're typing a really long line of code that extends past the end of the window, you can have Xcode wrap the line. You can specify how many spaces to indent the wrapped line. Line wrapping can help if you're writing code on a smaller screen, such as a laptop.

## Syntax-Aware Indenting

If you select the Syntax-aware indenting checkbox, Xcode turns on automatic code indenting. When you type a statement that calls for indentation, such as `if`, `for`, and `while` statements, and press the Return key, Xcode indents the new line for you. You can specify the characters that trigger the indenting as well as control how Xcode indents code comments.

## Key Bindings

Use the key bindings preferences to set keyboard shortcuts for Xcode. You can set a keyboard equivalent for any menu item in the Xcode menu bar using the Menu Key Bindings tab. The Text Key Bindings tab lets you set keyboard equivalents for things like text editing, moving the cursor, and text formatting.

Xcode comes with default key binding sets for Xcode, BBEdit, CodeWarrior, and MPW. If you're moving to Xcode from BBEdit, CodeWarrior, or MPW, the key binding sets will smooth the transition to Xcode. To create a custom set of key bindings, click the Duplicate button to create a copy of one of the default sets. Use the copy to bind keys.

## Using an External Editor

There are lots of text editors available on Mac OS X. Some examples include TextMate, BBEdit, TextWrangler, SubEthaEdit, and Emacs. If you bought a text editor, you want to use it to edit your source code instead of Xcode's editor. How do you tell Xcode to use your desired text editor to edit your source code files?



Open Xcode's File Types preferences. Use the File Types preferences to tell Xcode what file types should be opened in an external editor.

Initially there will be two items: file and folder. Click the disclosure triangle next to file. You will see a hierarchy of file types. At this point you have a decision to make. Do you want to use an external editor for all text files or just for certain types of text files? If you want to use an external editor for all text files, select text so you can tell Xcode what external editor to use for text files. Otherwise, click the disclosure triangle next to text to view the text file subcategories. The two most interesting subcategories are sourcecode and scripts.

When you come across a file type you want to edit with your text editor, select the Preferred Editor column for that file type. A menu will pop up. Choose External Editor. If your text editor does not appear in the submenu, choose Other. Navigate to the location of your text editor, and click the OK button. Repeat these steps for every file type you want to edit with your text editor.

Now when you double-click a source code file in Xcode's project window, it will open in your text editor.

## Refactoring

Refactoring takes existing source code and improves it without changing the code's behavior. Why would you want to refactor your code? Suppose you've written an application that works well but the code is difficult to understand in some places. You want to make the code easier to understand, but you don't want your changes to break the application. This is the problem refactoring solves. Refactoring lets you make the code easier to understand without breaking it.

Xcode provides tools to help you refactor your code. Xcode's refactoring tools work with C and Objective-C programs. They will not work with Objective-C++ code. Xcode supports two types of refactoring: refactoring a selection of code and refactoring an entire project.

### Refactoring a Selection

To refactor a selection of code, select the code you want to refactor in Xcode's editor window and choose Edit > Refactor. The refactoring window opens, which you can see in Figure 2.3. In the upper left corner of the refactoring window is a pop-up menu with the transformations you can perform. There are seven transformations you can perform on a selection of code.

- Rename, which changes the name of a class, variable, or method.
- Extract, which creates a new method out of a piece of code.

- Create superclass, which creates a superclass for an existing class. It creates `.h` and `.m` files for the new class.
- Move up, which moves the declaration and definition of a method to its superclass.
- Move down, which moves the declaration and definition of a method to a subclass.
- Encapsulate, which creates accessors out of a variable. It also modifies the code that accesses the variable to use the accessors.
- Modernize loop, which converts `for` and `while` loops to use Objective-C 2.0's fast enumeration. Use fast enumeration to go through the items in an array.

If you rename a class, two checkboxes will be enabled. Selecting the first checkbox renames related KVC/Core Data members of the class. KVC stands for key-value coding. Selecting the second checkbox renames related files. If you rename a method or an instance variable, the first checkbox will be enabled.

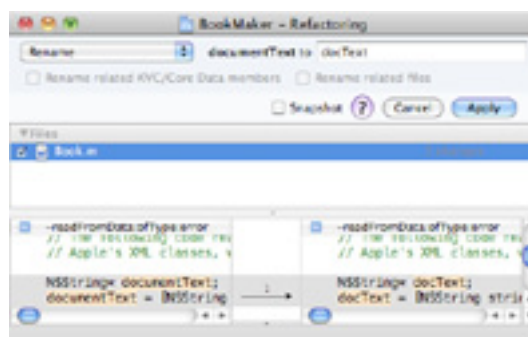
The transformations that are available to you depend on the code you select. If you select a variable name, the Extract transformation will be unavailable. If you select ten lines of code, the Rename transformation will be unavailable.

Clicking the Preview button lets you see the changes that will occur in your code. A list of files affected by the refactoring will appear after clicking the Preview button. Selecting a file shows you the changes that will occur in the file.

Clicking the Snapshot checkbox tells Xcode to take a snapshot of the project before doing the refactor. If you don't like the changes, you can go back to the snapshot. Refer to Chapter 6, "Version Control", for more information about project snapshots.

## Refactoring a Project

Choose `Edit > Convert to Objective-C 2.0` to refactor an entire project. The options for refactoring a project are more limited than the options for refactoring a selection. You can convert your Objective-C code to Objective-C 2.0. You have two options when converting a project to Objective-C 2.0: modernize loops and use properties. Modernizing loops converts `for` and `while` loops to use fast enumeration to go through the items in an array.



**Figure 2.3**

Refactoring window

Properties are a replacement for accessors. Suppose you have the following member of a class:

```
NSString* title;
```

Without properties you would declare two functions to get and set the value of the title member.

```
- (NSString*)title;
- (void)setTitle:(NSString*)titleName;
```

With properties you declare title as a property and use dot syntax to get and set the value of title.

```
@property (copy) NSString* title;

self.title = @"Xcode Tools Sensei";
```

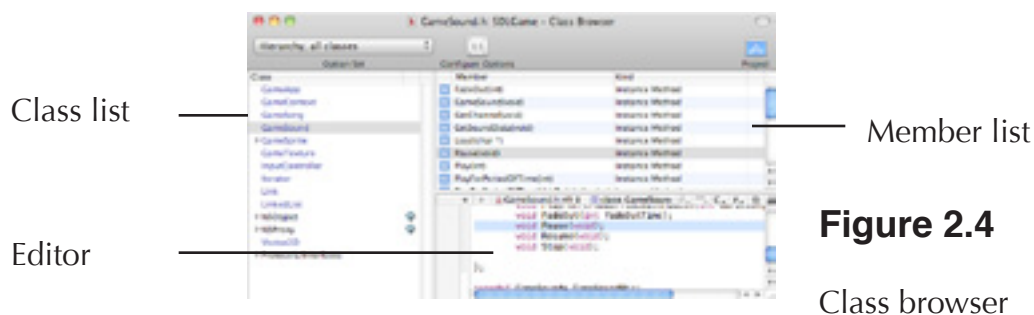
If you select the Use properties checkbox, Xcode will create properties for the data members of your Objective-C classes.

Objective-C 2.0 code will not run on any version of Mac OS X prior to 10.5. If you want to support older versions of Mac OS X, don't convert your code to Objective-C 2.0.

## Using Xcode's Class Browser

For those of you using object-oriented languages like C++ and Objective-C, Xcode has a class browser to examine your classes' members. Choose Project > Class Browser to open the class browser, which you can see in Figure 2.4. The class browser has three areas.

- Class list
- Member list
- Editor



**Figure 2.4**

Class browser

If no classes appear in the browser, you may need to index your project. To index your project, select the project name from the Groups and Files list and click the Info button. The project's inspector will open. Click the General tab. Click the Rebuild Code Sense Index button to create the index.

## Browsing Classes

The class list runs down the left side of the window. Classes you wrote appear in blue text. Classes defined in another framework, such as the built-in Cocoa classes, appear in black text. If a class has subclasses, that class will have a disclosure triangle next to it. Click the disclosure triangle to see the subclasses.

Selecting a class from the class list will do two things. First, Xcode fills the member list with the class's members. Second, Xcode opens the class's header file in the editor. Selecting a method from the member list takes you to the method's declaration in the header file. Double-clicking a class or member opens the class's header in a separate editor.

If you're writing Cocoa or iPhone programs, you'll notice the Cocoa and Cocoa Touch classes have a book icon. Clicking the book icon next to a class in the browser displays the documentation for the class. Clicking the book icon for a Cocoa or Cocoa Touch class member takes you to the documentation for the member.

## Customizing What the Class Browser Shows

You can customize what appears in the class browser by clicking the Configure Options button in the class browser toolbar. Doing so will open the class browser configuration window, shown in Figure 2.5. Looking at the figure, you can see two columns of controls. The left column deals with what appears in the class list, and the right column deals with what appears in the member list. Read the next two sections to learn what you can customize for the class and member lists.



**Figure 2.5**

Class browser  
configuration window

At the top of the configuration window is a pop-up menu containing saved configurations. A saved configuration saves your settings so you don't have to configure the class browser every time you use it. The class browser comes with four saved configurations, and you can add configurations to the list. Click the Add button to create a saved configuration. Name the configuration, use the configuration window to modify the class browser settings, and click the OK button. The configuration you created now appears in the Option Set pop-up menu in the class browser window.

## **Customizing What Appears in the Class List**

In the Class List Display Settings section there's a radio button group asking you whether you want the classes in your project to appear in a hierarchical outline or a flat list. In a hierarchical outline, classes with subclasses have a disclosure triangle, which you click to see the subclasses. In a flat list the classes appear in alphabetical order.

Below the radio button group are three pop-up menus. The first pop-up menu lets you decide whether you want only classes from your project, only classes from external frameworks (such as Cocoa), or both types of classes to appear in the class browser. The second pop-up menu lets you decide whether you want only classes, only protocols and interfaces, or both to appear in the browser. Protocols and interfaces are different names for the same thing. Objective-C uses the term protocol, and Java uses the term interface. A protocol is a collection of method declarations. These declarations are not part of a class. Any class can implement the methods in the protocol.

The third pop-up menu applies only to Objective-C programs. It lets you determine how to show Objective-C categories. Categories let you extend existing classes by adding methods to them, which is something you can't do in C++ and Java. By using categories you can add methods to the built-in Cocoa classes. You can show categories as subclasses of the class you're extending, subclasses for root classes, or show them merged into the class itself.

## **Customizing What Appears in the Member List**

In the Member List Display Settings section there's a checkbox asking you if you want to show inherited members of a class. If you select this checkbox, the inherited members of the class appear in gray text in the member list. The non-inherited members appear in black text to distinguish them.

Below the checkbox are two pop-up menus. The first pop-up menu asks whether you want to show methods, data, or both in the member list. Initially the class browser shows methods only.

The second pop-up menu asks whether you want to show the instances, show the class, or both. Instances are the class's data members and non-static member functions. If you choose class only, only static functions will appear in the member list. In Objective-C code, only class methods will appear if you choose class only; no instance methods will appear in the member list. Because most methods in classes are non-static, you'll want to see instances in the class browser.

## Reading Developer Documentation

When you're writing source code that uses Apple's technologies, you occasionally need to consult Apple's developer documentation. You don't have to leave Xcode to read developer documentation. Choose Help > Developer Documentation to open Xcode's documentation window. Most of your interaction with the documentation window takes place in the toolbar. The documentation window toolbar has four elements.

- Back and Forward buttons that work like the Back and Forward buttons in a web browser.
- A Home button that opens a documentation set. Read the "Browsing Documentation" section for more information.
- A Bookmarks button that contains documents you have bookmarked. Read the "Bookmarks" section for more information.
- A search field. Read the "Searching Documentation" section for more information.

When you open Xcode's documentation window for the first time, you get the Xcode Quick Start page, which is strange because the Help menu has a menu item for the Xcode Quick Start page. To start reading documentation you can either browse or search. Use the Home button to browse. Use the search field to search.

## Browsing Documentation

To browse documentation, click the Home button. A menu opens with a menu item for each documentation set you've installed. Most of you have at least two documentation sets installed: the Mac OS X set and the Developer Tools set. If you've installed the iPhone SDK, there should be a third set containing iPhone documentation.

With Mac OS X and iPhone documentation, the documentation is grouped four ways: by resource type, by topic, by framework, and by tool. On the left side of the documentation window is a list of category links containing the available resource types, topics, frameworks, and tools. Click a category link to show the documents for that category. Figure 2.6 provides an example of what the documentation window looks like after clicking a link. The Developer Tools documentation set has no list of links. It contains a list of all documents on Apple's developer tools.



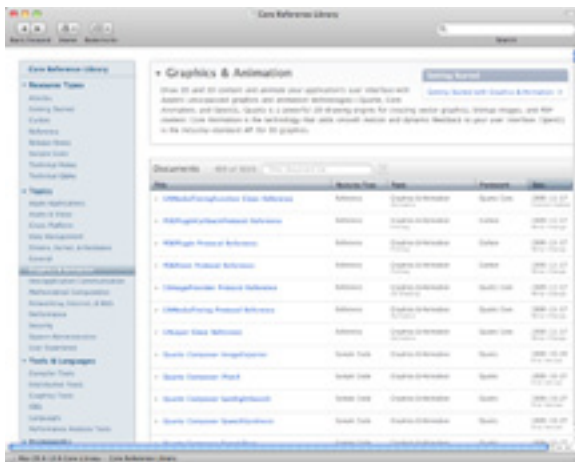
Some categories contain a lot of documents. If you click the Sample Code link under Resource Types, you will find over 750 entries. Use the search field above the list of documents to filter what appears in the documentation window.

## Searching Documentation

Searching the documentation is pretty simple. Enter what you want to search for in the search field on the documentation window toolbar and press the Return key. The search results appear on the left side of the documentation, divided by API, Title, and Full Text. Figure 2.7 shows an example of the documentation window after performing a search.

After performing the initial search, options appear under the toolbar to filter the search. The first filter deals with the search term: Contains, Prefix, or Exact. The initial value is Contains, which means the search results show every document that contains the search term. If you click Prefix, the search results show every document that starts with the search term. If you click Exact, the search results show every document that is an exact match for the search term. Clicking Exact reduces the search results greatly.

The second filter is the documentation sets to search. Initially Xcode searches all document sets. If you're an iPhone developer, you may want to limit searches to the iPhone documentation. Use the pull-down menu to specify the documentation sets to search. The third filter is the programming languages: C, C++, JavaScript, and Objective-C. Initially Xcode searches for all four languages. Use the pull-down menu to limit the languages to search.



### Figure 2.6

Documentation  
window browsing



### Figure 2.7

Documentation  
window searching

## Bookmarks

Adding bookmarks lets you quickly reach documents so you don't have to waste time navigating the documentation. To bookmark the page you're reading, choose Edit > Add to Bookmarks. You can also click the Bookmarks button in the documentation window's toolbar and choose Add Bookmark.

Clicking the bookmarks button opens a menu that contains a list of all the documents you bookmarked. Choose a document from the menu to read it.

Clicking the Bookmarks button and choosing Manage Bookmarks opens a sheet that contains all your documentation bookmarks. Double-clicking a bookmark lets you rename the bookmark. Selecting a bookmark and clicking the minus button removes that bookmark. Clicking the + button adds the page you're reading to the bookmarks list.

## Quick Help

Xcode's documentation window is fine for reading longer pieces of documentation, but when you want a quick explanation of a class or method, there's Quick Help. Quick Help works best with Cocoa, Cocoa Touch and Core Foundation. It also helps when you're examining code examples you haven't written.

### Invoking Quick Help

There are two ways to invoke Quick Help. First, choosing Help > Quick Help opens the Quick Help window. Moving the cursor inside a symbol in an editor fills the Quick Help window with information about that symbol. Examples of symbols are classes, functions, and variables.

Using the first option keeps the Quick Help window open all the time. If you don't want the Quick Help window open all the time, there is a second option. Option-double-clicking a symbol will open the Quick Help window when you want to use it.

If the Quick Help window displays the message "Symbol not found" when you've selected something in a file, make sure the file's corresponding project is open. If you open the Organizer, look at a file without opening the file's corresponding project, and try to get information on a symbol, nothing will appear in Quick Help.



## The Quick Help Window

At the top of the Quick Help window, shown in Figure 2.8, is the name of the symbol. If the symbol is a method, the class name is in parentheses. Two buttons are in the upper right corner of the window. Clicking the book button opens the documentation for the symbol. Command-Option-double-clicking a symbol in the editor also opens the documentation. Clicking the header button opens the header file in Xcode. Command-double-clicking a symbol in the editor also opens the header file.

Underneath the symbol is Quick Help's information on the symbol. Quick Help provides the following information:

- Abstract, which is the description of the symbol.
- Declaration. Most classes won't have a declaration. The Declaration section for methods shows the return type and the arguments the method takes. The Declaration section for data structures shows the members of the data structure.
- Sample Code, which contains links to sample code that uses the symbol.
- Related Documents, which contains links to any additional documentation about the symbol.
- Related API, which contains links to related method calls. Many entries won't have related API.
- Availability, which tells you the versions of Mac OS X or iPhone OS that can use the symbol.

Use Xcode's documentation preferences to customize the information that appears in the Quick Help window.

## Reading man Pages

Man pages contain documentation about the Unix operating system, which is the base of Mac OS X. By reading man pages you can learn more about Unix commands, tools, and function calls as well as command-line tools that ship with Xcode.



**Figure 2.8**

Quick Help window

To read a man page, choose Help > Open man Page. A dialog box opens. Enter the name of the page you want to open and click the OK button. The contents of the man page should appear in the documentation window. The name of the page you want to open corresponds to the command, tool, or function call you want to read. If you want to read `gdb`'s man page, enter `gdb` in the text field.

If you get an error message saying “No man Page Found”, do the search again. Click the search string radio button before clicking the OK button. Sometimes the information you want may not have its own page, but the information may be part of another man page. Doing a search string can find the information you need.

## Updating Documentation

Xcode's documentation is constantly changing so the documentation that was installed when you installed Xcode may not be up to date. Staying current is not too difficult as long as you have a broadband Internet connection.

Open Xcode's preferences and click the Documentation button in the toolbar. Click the Check and Install button to update the documentation. If you want Xcode to automatically install updates for you, select the Check for and install updates automatically checkbox.

Underneath the Check and Install button is a list of available documentation sets. Xcode initially installs the developer tools library, the Mac OS X library for the version of Mac OS X you're running, and the iPhone OS library for the iPhone SDK you've installed. Documentation sets you have not installed have a Get button next to them. Click the Get button to install that set.

Xcode's documentation viewer is not limited to Apple's documentation. If you have third-party documentation you want to install, click the Add Documentation Set Publisher button. The documentation must be an Atom or RSS feed. Atom feeds start with `http://` while RSS feeds start with `feed://`. Use the Feed URL text field to enter the location of the feed you want to install.

After adding a documentation feed, click the Get button to install it. The documentation set appears in the list of sets when you click the Home button. You can find third-party documentation sets you've installed in the following location:

`/Library/Developer/Shared/Documentation/DocSets`

If you want to delete a documentation set, select it from Apple's documentation preferences, right-click, and choose Reveal In Finder. Drag the documentation set to the Trash.