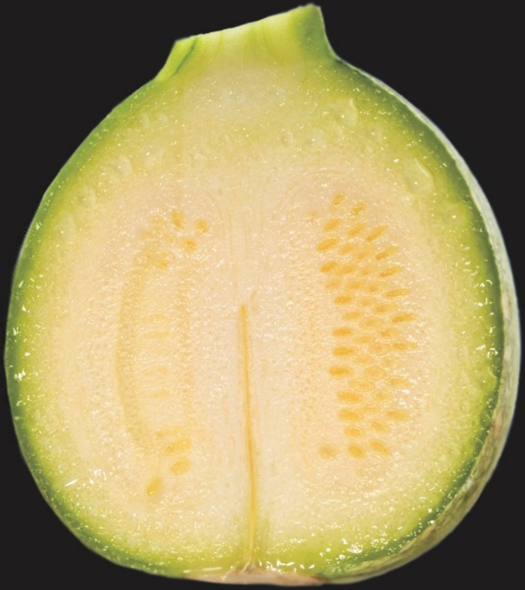
Get started building your very own iPhone and iPad apps



iPhone and iPad Apps for Absolute Beginners

Dr. Rory Lewis

Foreword by Ben Easton

Apress®

iPhone and iPad Apps for Absolute Beginners

Dr. Rory Lewis

Apress®

iPhone and iPad Apps for Absolute Beginners

Copyright © 2010 by Dr. Rory Lewis

ISBN-13 (pbk): 978-1-4302-2700-7

ISBN-13 (electronic): 978-1-4302-2701-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/info/bulksales`.

The source code for this book is available to readers at `www.apress.com`.

To Adrian and Eunice Lewis: Love you, Granny, and I miss you so much, Pops.

—Dr. Rory Lewis

Contents at a Glance

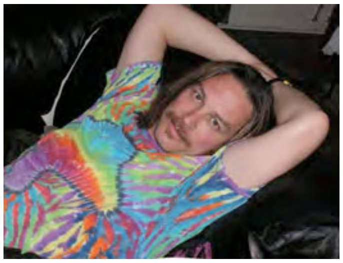
Contents

Foreword: About the Author

"Rory and I met in L.A. in 1983. He reminds me of one of my favorite film characters, Buckaroo Banzai—always going in six directions at once. If you stop him and ask what he's doing, he'll answer comprehensively and with amazing detail. Disciplined, colorful, and friendly, he has the uncanny ability to explain the highly abstract in simple, organic terms. He always accomplishes what he sets out to do, and he'll help you do the same.

Why you'll relate to Dr. Lewis

While attending Syracuse University as a computer-engineering student, Rory scrambled to pass his classes and make money to support his wife and two young daughters. In 1990, he landed a choice on-campus job as a proctor in the computer labs in the LC Smith College of Engineering. Even though he was struggling with subjects in the Electrical Engineering program, he was always there at the Help Desk. It was a daunting experience for Rory because his job was only to help his fellow students with computer lab *equipment* questions, but he invariably found his classmates asking deeper and harder questions: *"Dude, did you understand the calculus assignment? Can you help me?!"*



These students assumed that, because Rory was the proctor, he knew the answers. Afraid and full of self-doubt, he sought a way to help them without revealing his inadequacies. Rory learned to start with: *"Let's go back to the basics. Remember that last week the professor presented us with an equation…?"* By going back to the fundamentals, restating and rebranding them, Rory began to develop a technique that would, more often than not, lead to working solutions. By the time his senior year rolled around, there was often a line of students waiting at the Help Desk on the nights Rory worked.

Fast-Forward 17 Years

Picture a long-haired, wacky professor walking through the campus of the University of Colorado at Colorado Springs, dressed in a stunning contrast of old-school and drop-out. As he walks into the Engineering Building, he is greeted by students and faculty who smile and say hearty hellos, all the while probably shaking their heads at his tweed jacket, Grateful Dead t-shirt, khaki pants, and flip flops. As he walks down the hall of the Computer Science Department, there's a line of students standing outside his office. Reminiscent of the line of students that waited for him at the Help Desk in those early years as a proctor in the computer lab, they turn and greet him, *"Good morning, Dr. Lewis!"* Many of these students at UC-Colorado Springs are not even in his class, but they know that Dr. Lewis will see them and help them anyway.

Past—Present—Future

Dr. Lewis holds three academic degrees. He earned a Bachelor of Science in Computer Engineering from Syracuse University. Syracuse's LC Smith College of Engineering is one of the country's top schools. It is there that Intel, AMD, and Microsoft send their top employees to study for their PhDs.

Upon completing his BS (with emphasis on the mathematics of electronic circuitry in microprocessors), he went across the quad to the Syracuse University School of Law. During his first summer at law school, Fulbright & Jaworski, the nation's most prolific law firm, recruited Rory to work in its Austin office, where some of the attorneys specialize in high-tech intellectual-property patent litigation. As part of his clerking experience, Lewis worked on the infamous *AMD v. Intel* case; he helped assess the algorithms of the mathematics of microprocessor electrical circuitry for the senior partners.

During his second summer in law school, Skjerven, Morrill, MacPherson, Franklin, & Friel—the other firm sharing the work on the *AMD v. Intel* case—recruited Rory to work with them at their Silicon Valley branches (San Jose and San Francisco). After immersing himself in law for several years and receiving his JD at Syracuse, Lewis realized his passion was for the *mathematics* of computers, not the legal ramifications of hardware and software. He preferred a learning and creative environment rather than the fighting and arguing intrinsic in law.

After three years away from academia, Rory Lewis moved south to pursue his PhD in Computer Science at the University of North Carolina at Charlotte. There, he studied under Dr. Zbigniew W. Ras, known worldwide for his innovations in data mining algorithms and methods, distributed data mining, ontologies, and multimedia databases. While studying for his PhD, Lewis taught computer science courses to computer engineering undergraduates, as well as e-commerce and programming courses to MBA students.

Upon receiving his PhD in Computer Science, Rory accepted a tenure-track position in Computer Science at the University of Colorado at Colorado Springs, where his research is in the computational mathematics of neurosciences. Most recently, he co-wrote a grant proposal on the mathematical analysis of the genesis of epilepsy with respect to the hypothalamus. However, with the advent of Apple's revolutionary iPhone and its uniquely flexible platform—*and market*—for mini-applications, games, and personal computing tools, he grew excited and began experimenting and programming for his own pleasure. Once his own fluency was established, Lewis figured he could teach a class on iPhone apps that would include *non*-engineers. With his insider knowledge as an iPhone beta tester, he began to integrate the parameters of the proposed iPad platform into his lesson plans—even before the official release in April 2010.

The class was a resounding success and the feedback was overwhelmingly positive, from students and colleagues alike. When approached about the prospect of converting his course into a book to be published by Apress, Dr. Lewis jumped at the opportunity. He happily accepted an offer to convert his course outlines, class notes, and videos into the book you are now holding in your hands.

Why Write This Book?

The reasons Dr. Lewis wrote this book are the same reasons he originally decided to create a class for both engineering and non-engineering majors: the challenge and the fun! According to Lewis, the iPhone and iPad are "*…some of the coolest, most powerful, and most technologically advanced tools ever made—period!*"

He is fascinated by the fact that, just under the appealing touch screen of high-resolution images and fun little icons, the iPhone and iPad are programmed in *Objective-C,* an incredibly difficult and advanced language. More and more, Lewis was approached by students and colleagues who wanted to program apps for the iPhone and would ask his opinion on their ideas. It seemed that, with every new update of the iPhone, not to mention the advent of the expanded interface of the iPad, the floodgates of interest in programming apps were thrown wider and wider. Wonderful and innovative ideas just needed the proper channel to flow into the appropriate format and then out to the world.

Generally speaking, however, the people who write books about Objective-C write for people who know Java, C#, or C++ at an advanced level. So, because there seemed to be no help for the average person who, nevertheless, has a great idea for an iPhone/iPad app, Dr. Lewis decided to launch such a class. He realized it would be wise to use his own notes for the first half of the course, and then to explore the best existing resources he could find.

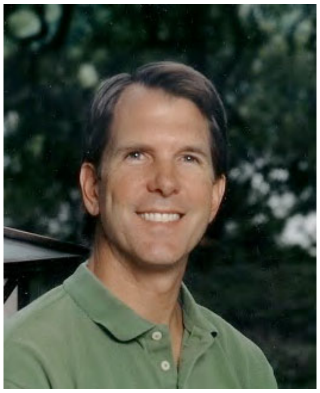
As he forged ahead with this plan, Lewis was most impressed with *Beginning iPhone 3 Development: Exploring the iPhone SDK*. This best-selling instructional book from Apress was written by Dave Mark and Jeff Lamarche. Lewis concluded that their book would provide an excellent, high-level target for his lessons…a "stepping stones" approach to comprehensive and fluent programming for all Apple's multi-touch devices.

After Dr. Lewis's course had been successfully presented, and during a subsequent conversation with a representative from Apress, Lewis happened to mention that he'd only started using that book about half-way through the semester, as he had to bring his non-engineering students up to speed first. The editor suggested converting his notes and outlines into a primer—an introductory book tuned to the less-technical programming crowd. At that point, it was only a matter of time and details—like organizing and revising Dr. Lewis's popular instructional videos to make them available to other non-engineers excited to program their own iPhone and/or iPad apps.

So, that's the story of how a wacky professor came to write this book. We hope you are inspired to take this home and begin. *Arm yourself with this knowledge and begin now to change your life!*

Ben Easton
Author, Teacher, Editor

About the Contributing Authors



Ben Easton is a graduate of Washington & Lee University and has a B.A. in Philosophy. His eclectic background includes music, banking, sailing, hang gliding, and retail. Most of his work has involved education in one form or another. Ben taught school for 17 years, mostly middle-school mathematics. More recently, his experience as a software trainer and implementer reawakened his long-time affinity for technical subjects. As a freelance writer, he has written several science fiction stories and screenplays, as well as feature articles for magazines and newsletters. Ben resides in Austin, Texas, and is currently working on his first novel.

Kyle Roucis is a student at the University of Colorado at Colorado Springs pursuing degrees in Computer Science and Game Design and Development. Kyle was Dr. Lewis' teaching assistant for CS 201, which was the class that tested all the apps and tutorial methodologies presented in this book. Kyle graded many students' attempts to write code from the lessons in this book and contributed wonderful suggestions as to how Dr. Lewis should change the way he presented certain topics. Kyle has been developing applications for the iPhone and iPod Touch since the SDK was first released in June of 2007. Most of his work has been iPhone contracting work as well as game and entertainment app development. Kyle lives in Colorado Springs and hopes to create his own game studio with an emphasis on iPhone, iPad and Mac game programming.

About the Technical Reviewer

Kristian Besley is a web developer and programmer. He currently works in education and specializes in games, interactivity, and dynamic content using mainly Flash, PHP, and .NET. He also lectures on interactive media.

Kristian has worked as a freelance producer for numerous clients including the BBC, JISC, Welsh Assembly Government, Pearson Education, and BBC Cymru.

He has written a number of books for Friends of ED, such as the *Foundation Flash* series, *Flash MX Video*, *Flash ActionScript for Flash 8*, and *Learn Programming with Flash MX*. He was also a proud contributor to the amazing *Flash Math Creativity* books and has written for *Computer Arts* magazine.

Kristian currently resides with his family in Swansea, Wales, and is a proud fluent Welsh speaker.

Acknowledgments

When I arrived in America in 1981 at the age of 20, I had no experience, money, or the knowledge to even use an American payphone. Since then it's been a wonderful road leading to this book and my life as an Assistant Professor at two University of Colorado campuses. I am such a lucky man to have met so many wonderful people.

First, to my wife, Kera, who moved mountains to help with graphics, meals, dictations, keeping me working, and sustaining a nominal level of sanity in our house. Thank you, Kera.

To my mother, Adeline, who was always there to encourage me, even in the darkest of times when I almost dropped out of Electrical Engineering. To my sister, Vivi, who keeps me grounded, and my late brother Murray, a constant reminder of how precious life is. To Keith and Nettie Lewis who helped me figure out those American payphones. To Ben Easton, Brian Bucci, and Dennis Donahue, all of whom invited me into their families when I had nobody.

A special thanks to Dr. Zbigniew Ras, my PhD advisor, who became like a father to me, and to Dr. Terry Boult, my mentor and partner in the Bachelor of Innovation program at UCCS.

Last but not least, to Clay Andres at Apress—he walked me through this process and risked his reputation by suggesting to a bunch of really intelligent people that I could author such a book as this.

Many thanks to you all.

Preface

What *This* Book Will Do For *You*

Let me get this straight: you want to learn how to program for the iPhone or the iPad, and you consider yourself to be pretty intelligent—but whenever you read computer code or highly technical instructions, your brain seems to shut down. Do your eyes glaze over when reading gnarly instructions? Does a little voice in your head chide you, *"How about that! Your brain shut down six lines ago, but you're still scanning the page—pretending you're not as dense as you feel. Great!"*

See if you can relate to this…you're having an issue with something pretty technical and you decide to Google it and troubleshoot the problem. You open the top hit—and somebody else has asked the exact same question! You become excited as the page loads, but, alas, it's only a bulletin board (a chat site for all those geeks who yap at one another in unintelligible code). You see your question followed by…but it's too late! Your brain has already shut down, and you feel the tension and frustration as knots in your belly.

Sound familiar?

Yes? Then this book's for you! My guess is that you're probably standing in a bookstore or in the airport, checking out a magazine stand for something that might excite. Because you're reading this in some such upscale place, you can probably afford an iPhone, a Mac, a car, and plane tickets. You're probably intrigued by the burgeoning industry of handhelds and the geometric rate at which memory and microprocessors are evolving…how quickly ideas can be turned into startlingly new computing platforms, into powerful software applications, into helpful tools and clever games…perhaps even into greenbacks! And now you are wondering if you can get in on the action—using your intellect and technical savvy to serve the masses.

How do I know this about you?

Easy! Through years of teaching students to program, I know that if you're still reading this, then you're both intelligent enough and sufficiently driven to step onto the playing field of programming, especially for a device as sweet as the iPhone or as sexy as the iPad. If you identify with and feel connected to the person I've described above, then I know you. We were introduced to one another long ago.

You are an intelligent person who may have mental spasms when reading complex code—even if you have some background in programming. And even if you do have a pretty strong background in various programming languages, you are a person who simply wants an easy, on-point, no-frills strategy to learn how to program the iPhone and iPad. No problem! I can guide you through whatever psychological traffic jams you typically experience and help you navigate around any technical obstacles, real or imagined. I've done this a thousand times with my students, and my methodology will work for you, too.

The Approach I Take

I don't try and explain everything in minute detail. Nor do I expect you to know every line of code in your iPhone/iPad application at this stage. What I will do is show you, step by step, how to accomplish key actions. My approach is simultaneously comprehensive and easy-going, and I take pride in my ability to instruct students and interested learners along a wide spectrum of knowledge and skill sets.

Essentially, I will lead you, at your own pace, to a point where you can code, upload, and perhaps sell your first iPhone/iPad app, simple or complex. *Good news*: the most downloaded apps are *not* complex. The most popular ones are simple, common-sense tools for life…finding your car in a parking lot, or making better grocery lists, or tracking your fitness progress. However, when you complete this book, you may want to graduate to other books in the Apress and Friends of ED series. You have quite a few options here, and down the road I'll advise you regarding the best ways to move forward. Right now, though, you may want to read a little about me so you will feel confident in taking me on as your immediate guide in this exciting app-venture.

May you experience great joy and prosperity as you enter this amazing and magical world.

Peace!

[illegible]

Rory A. Lewis, PhD, JD

Chapter 1

Before We Get Started

This introductory chapter will make sure that you have all the required tools and accessories to proceed fully and confidently. Some of you may already be solid on these points and feel ready to jump right in. If so, you may want to jump ahead to Chapter 2 and start immediately on your first program.

It will behoove you, though, to understand why I teach certain things and skip others. For those of you who have never done it, programming in Objective-C is quite a challenge—even for my engineering students who know Java, C, and C#. Nevertheless, with the appropriate preparation and mindset you will accomplish this.

So I urge you to read on. The time you will invest in this chapter will be well worth it in peace of mind and confidence. Chapter 1 will help structure the way that your brain will file all the rich content that is to come.

Necessities and Accessories

In order to program for the iPhone and/or iPad, and to follow along with the exercises, tutorials, and examples presented in this book, you'll need to pay attention to certain minimal requirements:

- Intel-based Macintosh running Leopard (OS X … 10.5.3 or later)
  - If it was bought after 2006, you're OK.
  - You don't need the latest revved up Mac. If you haven't bought one yet, I suggest you get a basic, no-frills MacBook.
  - If you do own an older Mac, then add some RAM. Make an appointment at the Genius Bar at an Apple Store and ask them to increase the RAM as much as possible.
- Become a registered developer via the iPhone/iPad Software Development Kit (SDK).

- If you are a student, it's likely that your professor has already taken care of this, and you may already be registered under your professor's name.
- If you are not a student, then you will need to follow these steps to sign up.

1. Go to `http://developer.apple.com/programs/iphone/`, which will bring you to a page similar to the one shown in Figure 1–1. Click the Enroll Now button.

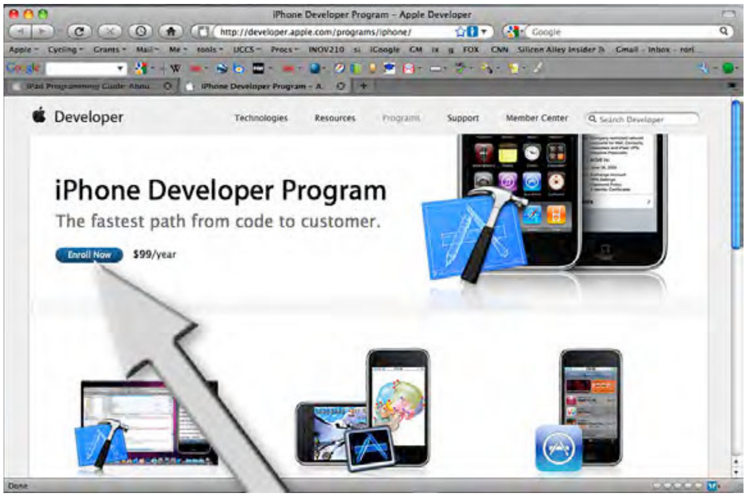
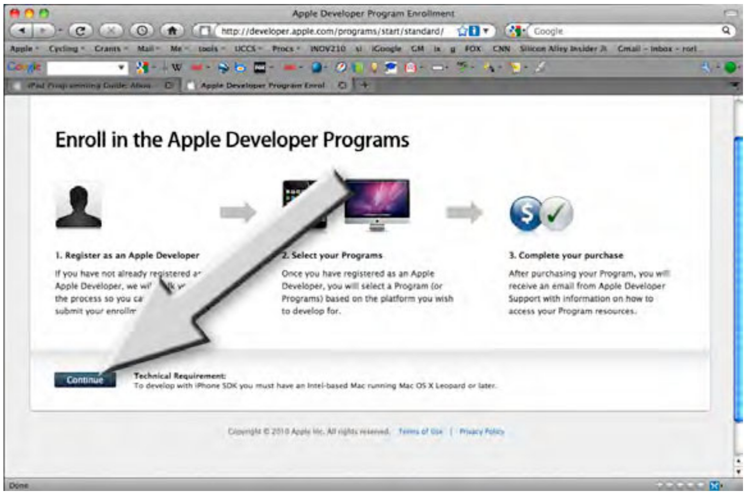

iPhone Developer Program – Apple Developer
http://developer.apple.com/programs/iphone/
Developer
Technologies
Resources
Programs
Support
Member Center
iPhone Developer Program
The fastest path from code to customer.
Enroll Now
$99/year

Figure 1–1. *Click the Enroll Now button.*

2. Click the Continue button as illustrated in Figure 1–2.


Apple Developer Program Enrollment
http://developer.apple.com/programs/start/standard/
Continue

Enroll in the Apple Developer Programs

1. Register as an Apple Developer

If you have not already registered a[illegible] Apple Developer, we wi[illegible] the process so you ca[illegible] submit your enrollm[illegible]

2. Select your Programs

Once you have registered as an Apple Developer, you will select a Program (or Programs) based on the platform you wish to develop for.

3. Complete your purchase

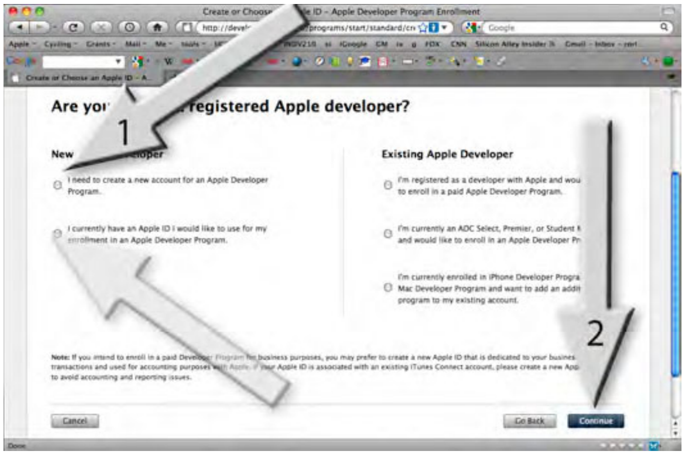
After purchasing your Program, you will receive an email from Apple Developer Support with information on how to access your Program resources.

Technical Requirement:
To develop with iPhone SDK you must have an Intel-based Mac running Mac OS X Leopard or later.

Copyright © 2010 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

Figure 1–2. *Click the Continue button.*

3. Most people reading this book will select the "I need to create a new account for …" option (arrow 1 in Figure 1–3). Next, click the Continue button as illustrated by arrow 2 in Figure 1–3. (If you already have an existing account, then you have been through this process before; go ahead with the process beginning with the "I currently have an Apple ID ..." option, and I'll meet you at step 6, where we will log onto the iPhone/iPad development page and download the SDK.)



Are yoı registered Apple developer?

New eloper

I need to create a new account for an Apple Developer Program.

I currently have an Apple ID I would like to use for my enrollment in an Apple Developer Program.

Existing Apple Developer

I'm registered as a developer with Apple and wou to enroll in a paid Apple Developer Program.

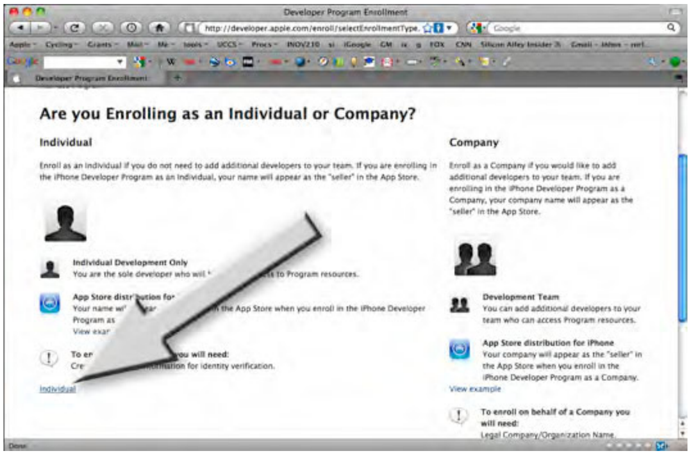
I'm currently an ADC Select, Premier, or Student M and would like to enroll in an Apple Developer Pr

I'm currently enrolled in iPhone Developer Progra Mac Developer Program and want to add an addi program to my existing account.

Note: If you intend to enroll in a paid Developer Program for business purposes, you may prefer to create a new Apple ID that is dedicated to your busines transactions and used for accounting purposes with Apple. If your Apple ID is associated with an existing iTunes Connect account, please create a new App to avoid accounting and reporting issues.

Figure 1–3. *Click the "I need to create an Apple ID …" option to proceed.*

4. You are probably going to be enrolling as an individual, so click the Individual link as illustrated in Figure 1–4. If you are enrolling as a company, click the Company option to the right and follow the appropriate steps; I'll meet you at step 6.


Developer Program Enrollment
http://developer.apple.com/enroll/selectEnrollmentType.
Individual
View example

Are you Enrolling as an Individual or Company?

Individual

Enroll as an individual if you do not need to add additional developers to your team. If you are enrolling in the iPhone Developer Program as an individual, your name will appear as the "seller" in the App Store.

Individual Development Only
You are the sole developer who will s to Program resources.

App Store distr bution for
Your name wi ar the App Store when you enroll in the iPhone Developer Program as
View exar

To er ou will need:
Cre mation for identity verification.

Company

Enroll as a Company if you would like to add additional developers to your team. If you are enrolling in the iPhone Developer Program as a Company, your company name will appear as the "seller" in the App Store.

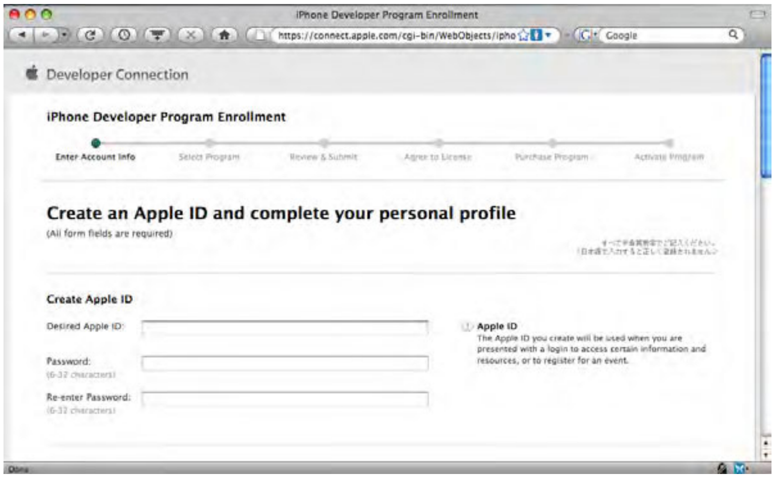
Development Team
You can add additional developers to your team who can access Program resources.

App Store distribution for iPhone
Your company will appear as the "seller" in the App Store when you enroll in the iPhone Developer Program as a Company.

To enroll on behalf of a Company you will need:
Legal Company/Organization Name.

Figure 1–4. *Click the Individual option.*

5. From here you will enter all your information as shown in Figure 1–5 and pay your fee of $99.00 for the Standard Program. This provides all the tools, resources, and technical support you will need. (If you're reading this book, you really do not want to buy the Enterprise program at $299, as it is for commercial in-house applications.) After paying, save your Apple ID and Username; then receive and interact with your confirmation email appropriately.


iPhone Developer Program Enrollment

https://connect.apple.com/cgi-bin/WebObjects/ipho

Developer Connection

iPhone Developer Program Enrollment

Enter Account Info — Select Program — Review & Submit — Agree to License — Purchase Program — Activate Program

Create an Apple ID and complete your personal profile
(All form fields are required)

Create Apple ID

Desired Apple ID:

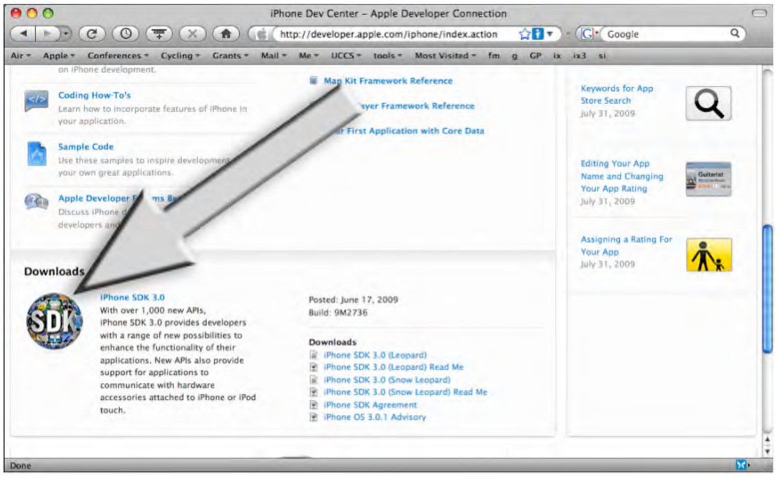
Password:
(6-32 characters)

Re-enter Password:
(6-32 characters)

Apple ID
The Apple ID you create will be used when you are presented with a login to access certain information and resources, or to register for an event.

Figure 1–5. *Enter all your information accordingly.*

6. Use your Apple ID to log into the main iPhone/iPad development page. Scroll down to the bottom of the page and download the SDK as illustrated in Figure 1–6. Extract the necessary icons onto your dock. Included with the Apple SDK that you've now downloaded is Apple's integrated development environment (IDE). This is a programming platform that contains a suite of tools, sub-applications, and boilerplate code that all enable us to do our jobs more easily. We will use Xcode, Interface Builder, and the iPhone/iPad Simulator extensively, so I advise you to bring these icons to your dock to save yourself tons of time searching for them.


iPhone Dev Center – Apple Developer Connection

http://developer.apple.com/iphone/index.action

on iPhone development.

Coding How-To's
Learn how to incorporate features of iPhone in your application.

Sample Code
Use these samples to inspire development your own great applications.

Apple Developer F ms B
Discuss iPhone d developers and

Map Kit Framework Reference
yer Framework Reference
r First Application with Core Data

Keywords for App Store Search
July 31, 2009

Editing Your App Name and Changing Your App Rating
July 31, 2009

Assigning a Rating For Your App
July 31, 2009

Downloads

iPhone SDK 3.0
With over 1,000 new APIs, iPhone SDK 3.0 provides developers with a range of new possibilities to enhance the functionality of their applications. New APIs also provide support for applications to communicate with hardware accessories attached to iPhone or iPod touch.

Posted: June 17, 2009
Build: 9M2736

Downloads
- iPhone SDK 3.0 (Leopard)
- iPhone SDK 3.0 (Leopard) Read Me
- iPhone SDK 3.0 (Snow Leopard)
- iPhone SDK 3.0 (Snow Leopard) Read Me
- iPhone SDK Agreement
- iPhone OS 3.0.1 Advisory

Done

Figure 1–6. *Having logged in as a Registered Apple Developer, you can now scroll down to the bottom of the page and download the SDK.*

7. Bring Xcode to your dock. by choosing Macintosh HD ➤ Developer ➤ Applications ➤ Xcode.app and dragging it onto your dock as illustrated in Figure 1–7. In the same way, bring Interface Builder to your dock by choosing Macintosh HD ➤ Developer ➤ Applications ➤ Interface Builder.app and dragging it. Finally, bring the iPhone/iPad Simulator to your dock by choosing Macintosh HD ➤ Developer ➤ Platforms ➤ iPhone/iPad Simulator Platform and dragging it.

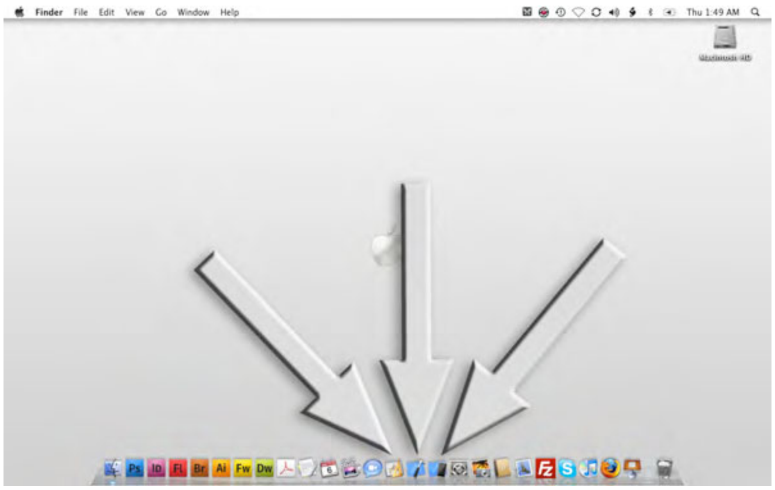


Figure 1–7. *Xcode, Interface Builder, and the iPhone/iPad Simulator—locked and loaded, ready to roll!*

NOTE: Whenever I say "iPhone" or "iPad," I am referring to any iPhone or iPad OS device. This includes the iPod touch.

What I Won't Teach You

With your Xcode, Interface Builder, and iPhone/iPad Simulator tools installed and ready to access easily, you're ready to roll. But wait! You need to know where we're going.

First, though, let me say something about where we won't be going—what I will *not* be covering. I will not attempt to teach you how every line of code works. Instead, I will take a subsystem approach, indicating which pieces or sections of code will serve you in which situations.

While this book is designed to impart to you, the reader and programmer, a comprehensive understanding and ability, we will be dealing in molecules rather than atoms or subatomic particles. The emphasis will be on how to recognize general attributes, behaviors, and relationships of code so that you need not get bogged down in the symbol-by-symbol minutiae. I will get you to a place where you can choose those areas in which you may want to specialize.

Computer Science: A Broad and Diverse Landscape

Consider this analogy: suppose that the iPhone/iPad is a car. Most of us drive cars in the same way that we use computers. Just as I would not attempt to teach you how every part of the car works if I were giving you driving lessons, I would not—and will not—approach iPhone and iPad programming with fundamental computer engineering as the first step.

Even great mechanics who work on cars every day rarely know the fundamental physics and electronics behind the modern internal combustion engine, not to mention all the auxiliary systems; they can drive a car, diagnose what's wrong with it when it needs servicing, and use their tools and machines (including computers) to repair and tune it optimally. Similarly, clever programmers who create the apps for the iPhone and iPad rarely know the fundamental coding and circuit board designs at the root of the Apple platforms. But they can use these devices, they can envision a new niche in the broad spectrum of applications needs, and they can use their tools and applications—residing on their desktops and laptops—to design, code, and deliver them to the market.

To continue with this analogy, programming the iPhone or iPad is like playing with the engine of your car—customizing it to do the things you want it to do. Apple has designed a computing engine every bit as fantastic as a V8 motor. Apple has also provided a pretty cool chassis in which we can modify and rebuild our computing engine. There are restrictions on how we can "pimp" our iPhone/iPad cars, and, for those of you who have never pimped a car, I will demonstrate how to maximize creative possibilities while honoring these restrictions.

I’m going to show you, without too much detail, how to swap oil filters, tires, seats, and windows to convert it into an off-road car, a hot rod, a racing car, or a car that can get us through the jungle. When you’ve mastered this book, you will know how to focus on and modify the engine, the transmission, the steering, the power train, the fuel efficiency, or the stereo system of the car.

Why Purgatory Exists In Objective-C

My Assumption: you’ve never worked on a car, and you’ve never gotten grease on your hands, and you want to pimp one of the world’s most powerful automobiles—with a complex V8 engine. I’m going to show you exactly how to do this, and we’re going to have fun doing it!

First, you need to know a little about how we even came to have the souped-up car with the V8—that is, the iPad. In 1971, Steve Jobs and Steve Wozniak met, and five years later they formed Apple, producing one of the first commercially successful personal computers. In 1979, Jobs visited Xerox PARC (Palo Alto Research Center), and secured the Xerox Alto's features into their new project called the *Lisa.* Although the Alto was not a commercial product, it was the first personal computer to use the desktop metaphor and graphical user interface (GUI). The Lisa was the first Apple product with a mouse and a GUI.

In early 1985, Jobs lost a power struggle with the Board of Directors at Apple, resigned from the company, and founded NeXT, which eventually bought out Apple in 1997. During his time at NeXT, Steve Jobs changed some critical features of the code on the Macintosh (Mac) to talk in a new language, a very intense but beautiful language called Objective-C. The power of this language was in its ability to efficiently use objects. Rather than reprogramming code that was used in one portion of the application, Objective-C *reused* these objects. Jobs’ brain was on overdrive at the time, and this incredible code took this new language of Objective-C to new heights. His inspiration was fused into the guts of the Mac by creating a metalanguage we call Cocoa. A metalanguage is a language used to analyze or define another language. As I’ve indicated, Objective-C is a very challenging beast, and you can think of Cocoa as the linguistic taming of the beast, or at least the caging of the beast.

As an “absolute beginner” to the world of programming, you cannot be expected to be concerned with the subtleties of coding language distinctions. I am simply giving you an overview here so that you will have a rough historical context in which to place your own experience. The main point I’m making here is that Objective-C and Cocoa are very powerful tools, and both are relevant to the programming of the iPhone/iPad.

Houston, We Have a Problem

This is the essence of the challenge that intrigued me, and led to the design of my original course. How can one teach non-engineering students, perhaps like you, something that even the best engineering students struggle with? At the university level, we typically have students first take introductory programming classes, and then proceed to introductory object-oriented programming, such as C# or C++.

That being said, we are going to dive *head on* into Objective-C! At times, I'm going to put blindfolds onto you; at other times, I'm going to cushion the blows. There will be times when you may need to reread pages or rewind video examples a few times— so that you can wrap your head around a difficult concept.

How We'll Visit Purgatory Every Now and Again

There are specific places in my courses where I know that half the class will immediately get it, a quarter will have to sweat over it before they get it, and the remaining quarter will struggle and give up. This third group will typically transfer out of engineering and take an easier curriculum. I know where these places are, and I'm not going to tell you. I'll repeat that. I will not tell you.

Don't worry, I won't allow you to disturb a hornet's nest (of Objective-C issues) and get stung to death. Nor will I mark off those concepts that you may find difficult. I'm not going to explain this now. Just accept it! If you just relax and follow my lead, you'll get through this book with flying colors.

When you do find yourself in one of those tough spots, persevere. You can always reread the section or rewind the video examples. In this iPhone and iPad programming adventure, it won't serve you to skip it. There are only about three of these critical areas in the book, and I've made them as easy as I can. There are also blogs and discussion boards you can access in order to discuss problems and share your thoughts with others.

Looking Forward … *Beginning iPhone 3 Development: Exploring the iPhone SDK*

Down the line, some of you may want to continue your iPhone and iPad programming adventure by reading Dave Mark and Jeff Lamarche's book, *Beginning iPhone 3 Development* (Apress, 2009). Remember the analogy of becoming a mechanic for an automobile with a V8 engine mounted on a basic chassis? Their book presumes that the readers know what a carburetor is, know what a piston is, and that they can mount racing tires and super fly rims on their friends' pimped-up wheels.

In other words, they assume that you understand the fundamentals of object-oriented programming: that you know what objects, loops, and variables are, and that you are familiar with the Objective-C programming language.

On the other hand, I assume that you don't know, for example, what a "class" is, or what a "member" or "void" is. I imagine that you have no idea how memory management works on an iPhone/iPad and, furthermore, that you never had an interest – *until now* – in understanding an array, or an SDK.

What You Will Learn

When students start a challenging class, I have found that it works wonders to have them create something real cool, and with relative ease. At each stage of this process, I will typically present an example that you can read, see, and digest right away. Later on, we will return to analyze some of the early steps and go into more detail. I will explain how we accomplished some task or action the first time *without even knowing it*. Then, by comparing the first time through with subsequent modifications, you will learn how to tweak the program a little here, a little there. This way, you'll stay on track—motivated and inspired to absorb the next new batch of tricks, lessons, and methods.

Creating Cool and Wacky Apps: Why I Teach This Way

You've heard the bit about how we best remember things: doing is better than seeing, which is better than hearing, and so on. Well, I know that students love humor—and guess what! We remember funny stories and lessons much better than we remember dull and boring ones. I have found that, without exception, when students work on code that is fun and wacky, they tend to spend much more time solving it.

The more we apply ourselves mentally toward the solution of a problem, the more neural connections are made in our brains. The more neurons we connect, the more we remember and—most importantly—the less apt we are to waste time on ineffective methods.

The more time we spend on a particular topic, the more chance there is that you will experience gut feelings about whether a particular methodology for solving a project is on track or not. So, as we proceed, be aware that I am employing humor to burn computer science and Objective-C concepts and methods into your brain without your exerting any conscious effort.

It is common for my students to contact me after receiving a difficult homework assignment. First, they'll send me a tweet asking if they can Skype me. One particular night, I was playing chess with a colleague when I received a tweet asking if I were available. "Of course," I responded. I warned my colleague, also a professor at the University of North Carolina, that students whom he knew were about to appear on Skype. When they buzzed in, sure enough: four of my electrical engineering students, wide-eyed and smiling. "*Hey, Dr. Lewis, we finally got it, but Dude! The last method you assigned---.*"

When we finished our conversation, and I turned off my Mac, it was 12:30 am. My colleague asked, "Rory, I never called a professor this late in the evening—much less *after midnight*! Shouldn't they ask these questions during office hours?!" He was probably right, but after thinking about it for a minute I replied, "I'm just happy that they're working on my wacky assignment!" As we set up the next chess game, he murmured something about how I might be comfortable in the insanity ward.

The point is that I want you to read this entire book. I want you to work all the examples and to feel elation as you complete each assignment! I have done everything I can to make this book enjoyable. If you choose to engage with the ideas contained herein, this book will change your life!

By the way, successfully navigating these lessons will make you a certified geek. Everybody around you will sense your growing ability and will witness your transformation; as a result, they will seek you out to request that you write apps for them.

Evangelizing to Your Grandmother … What You Coded Is Crucial!

It's important that you not let complex code turn you inside-out. Just two minutes ago, a student walked into my office—so confused that he couldn't even tell me what it was he didn't know. He said something like, "My second order array worked fine in-line, but not as a class or a method." I said, "No, that's too complex! Here's an easier way of saying it …"

I described how he had a long line of "stuff" going in one end and being spat out the other – and it worked really well. But, when he put it in a *method*, he couldn't see the start of the long line of stuff; when he put it in a *class*, he couldn't see *any* of the stuff!"

"Wow! I know what I did wrong, Dr. Lewis. Thank you!" Now, as I type this, he's explaining it to his two buddies who came in yesterday and tried to ask the same question. Don't worry, the confusion that drove these questions – such as the distinctions between "classes" and "methods," and other coding entities, will be covered later in this book. All in good time!

If you can keep your feet on the ground and transform complex things into simpler ideas, then you can remember them—and master them. Grasp this concept, and you will be able to convert your far out ideas into code—and who knows where that will take you! This is why I am so determined to impart to you the ability to convert things your grandmother wants to be able to do into iPhone and iPad programming language.

How Does This All Work?

Before we start our first program in Chapter 2, it's critical that you are able to step back and know where we've been, where we are now, and where we will go next. Looking at Figure 1–8, you can see a gray strip containing two icons that represent Mac OSX and the SDK, which includes Interface Builder and Xcode. These will be explained in detail later; 90 percent of this book deals with the items in this strip.

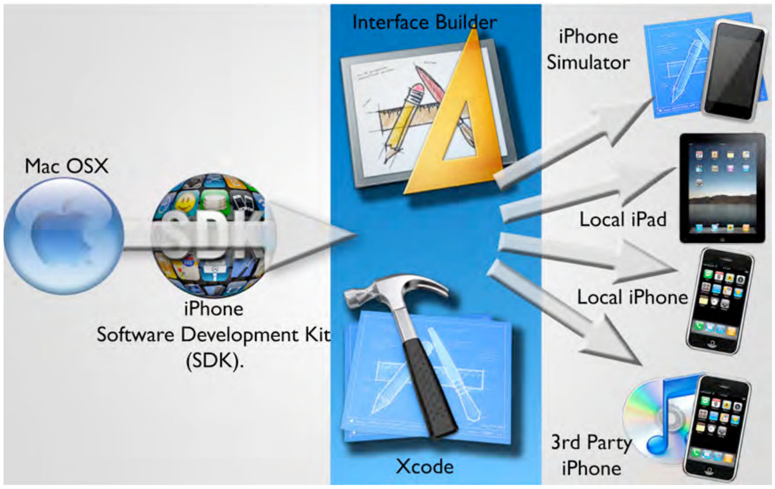


Figure 1–8. *The iPhone and iPad app programming landscape. Mac OS X 10.5.3 or later is housing the iPhone and iPad SDK. I will teach you how to use the SDK's Xcode and the Interface Builder to create apps. Once you create an app, there are four ways to run it: using iPhone/iPad Simulator, with your iPhone attached to your Mac, with your iPad attached to your Mac, or downloaded via iTunes to a third-party iPhone/iPad.*

The blue band in the middle is where we are presently. To the left of this area is where you've been. You have a Mac (purchased after 2006 and running Mac OS X 10.5.3 or higher), and we've just walked through the process of downloading the iPhone and iPad SDK (Figures 1–1 thru 1–6). We have also extracted Interface Builder, Xcode, and the iPhone/iPad Simulator and positioned them onto your dock (Figure 1–7). That's where we are now.

In Chapter 2, we will start using Xcode and Interface Builder to turn you into a bona-fide geek! We're going to run all the programs we make by compiling them to one of several possible locations, the icons for which are to the right of the central blue area. The primary location will be the iPhone/iPad Simulator. The secondary locations will be your local iPhone and/or your local iPad. Lastly, we could use iTunes to upload your iPhone and/or iPad App to the App Store where people can purchase it or download it for free. This is where we are going.

The two central objects in Figure 1–8, as you now know, are where we will spend the vast majority of our time within this book. We'll be using Xcode to type in code, just like the serious geeks do. I'll show you how to operate all its features such as file management, compilation, debugging, and error reporting. Interface Builder is the cool way Apple allows us to drag and drop objects onto our iPhone/iPad apps. If you want a button, for instance, you simply drag and drop it where you want it to be located on the virtual iPhone or iPad.

Essentially, we'll use Xcode to manage, write, run, and debug your app—to create the content and functionality. We'll use Interface Builder to drag and drop items onto your interface until it looks like the colorful and cool application you envisioned—to give it the style, look, and feel that suits your artistic tastes.

After we integrate all the interface goodies with the code we wrote in Xcode, we might get advanced and tweak the parameters dealing with memory management and efficiency. But that's jumping too far ahead in our story.

Our Road Map: Using Xcode and Interface Builder

Very often authors of programming books do the same old thing. They first present a very simple, ubiquitous "Hello World" application and then throttle the user with intense code that loses a great many readers and students straight away. Utilizing Objective-C (being run in Cocoa) along with the iPhone and iPad SDK, I've had to really rethink this introductory process. I have identified three challenges here.

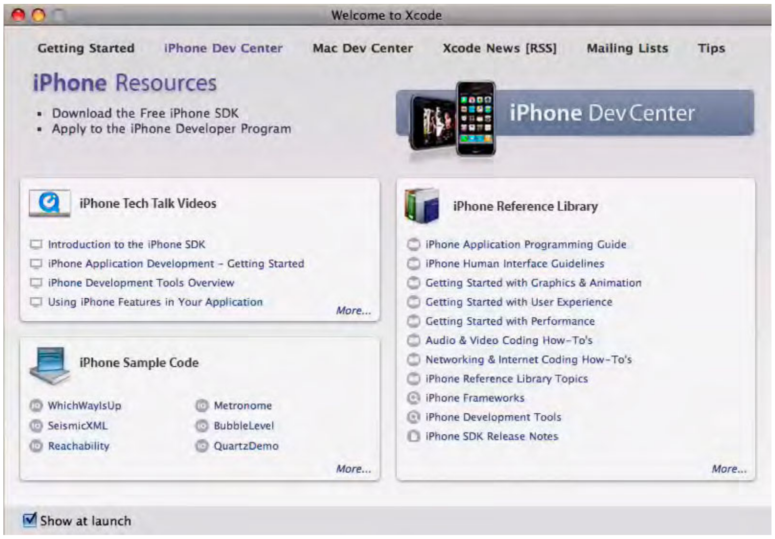
- Teaching you "Hello World" and then going into advanced technologies and APIs would be counter-productive.
- It makes no sense to randomly choose one of the many ways to say Hello to the world from your iPhone or iPad. They are all going to be necessary to have in your toolkit at a later date.
- Trying to write a simple "Hello World" application in Objective-C is more involved than the beginner is ready for, unless we break up the process into stages or layers.

My solution to overcoming these issues is simple. I'll show you how to say Hello to the world from your iPhone/iPad in not one, not two, but quite a few different ways. Each time, we'll go a little bit deeper, and we'll have a blast as we do so.

Each time you travel down the road into the land of Xcode, you are immediately asked what type of vehicle you'd like to drive. A Jeep? A race car? A convertible? By focusing on basics, I am going to show you how to "drive" *in Xcode*. The objective here will be to gain competence and confidence in whatever style of vehicle we must access. So, let's take a look at exactly what these different vehicles have to offer. Here I would like you to follow along with me.

Getting Ready For Your First iPhone/iPad Project

Assuming that you have already downloaded the SDK and installed Interface Builder, Xcode, and the iPhone/iPad Simulator, open up your Mac and click the Xcode icon on your dock. Your screen should look similar to Figure 1–9. Up pops the Welcome to Xcode window; it includes all your iPhone and iPad resources.



iPhone Resources

- Download the Free iPhone SDK
- Apply to the iPhone Developer Program

iPhone Tech Talk Videos

- Introduction to the iPhone SDK
- iPhone Application Development – Getting Started
- iPhone Development Tools Overview
- Using iPhone Features in Your Application

iPhone Sample Code

- WhichWayIsUp
- SeismicXML
- Reachability
- Metronome
- BubbleLevel
- QuartzDemo

iPhone Reference Library

- iPhone Application Programming Guide
- iPhone Human Interface Guidelines
- Getting Started with Graphics & Animation
- Getting Started with User Experience
- Getting Started with Performance
- Audio & Video Coding How-To's
- Networking & Internet Coding How-To's
- iPhone Reference Library Topics
- iPhone Frameworks
- iPhone Development Tools
- iPhone SDK Release Notes

Figure 1–9. *After clicking the Xcode icon, you will see the "Welcome to Xcode" screen. Keep the "Show at Launch" option checked.*

Look at the bottom-left corner of the pop-up window and make sure you keep the Show at Launch option checked. There are many valuable resources here that you will find handy. I suggest that, after you have completed Chapter 4, you take a little time and explore these resources—give them a test drive, so to speak. This practice will open all kinds of creative doors for you.

Without actually starting a new project, let's walk up to the showroom floor and check out some of the models we might be driving. To open a new project in Xcode, enter Command + Shift + N simultaneously. This three-key shortcut, depicted in Figure 1–10 as (⌘⇧N), will open a window that showcases the different types of vehicles that you can drive in the land of Xcode.

Figure 1–10 displays the six vehicle models: Navigation-based Application, Open GL ES Application, Tab Bar Application, Utility Application, View-based Application, and Window-based Application.

Early on, most of our travel in Xcode will be by one of the latter two styles shown. Switching back to computer terms, View-based Application and Window-based Application are the structures we will utilize in the basic development cycle for the iPhone/iPad. It is here that we will access cool gadgets and components.

Don't worry, I haven't forgotten our goal of creating a simple "Hello World" application. We will say Hello to the world while using a number of the six options, and you will become familiar with each.