第3版
包含Windows Phone项目

Head First

C#

A Learner's Guide to
Real-World Programming
with C#, XAML, and .NET

Boss your
objects
around with
abstraction
and inheritance

Unravel the
mysteries of the
Model-View-ViewModel
(MVVM) pattern

Build a fully
functional
retro classic
arcade game

Learn how
asynchronous
programming
helped Sue keep
her users thrilled
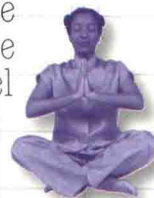
See how Jimmy used
collections and LINQ
to wrangle an unruly
comic book collection

深入浅出C# （影印版）

Head First C#

第3版

WOULDN'T IT BE DREAMY IF THERE WAS A C# BOOK THAT WAS MORE FUN THAN **MEMORIZING A PHONE BOOK?** IT'S PROBABLY NOTHING BUT A FANTASY....

Andrew Stellman &

Jennifer Greene 著

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

"*Head First C#* is a great book, both for brand new developers and developers like myself coming from a Java background. No assumptions are made as to the reader's proficiency yet the material builds up quickly enough for those who are not complete newbies—a hard balance to strike. This book got me up to speed in no time for my first large scale C# development project at work—I highly recommend it."

> — **Shalewa Odusanya, Technical Account Manager, Google**

"*Head First C#* is an excellent, simple, and fun way of learning C#. It's the best piece for C# beginners I've ever seen—the samples are clear, the topics are concise and well written. The mini-games that guide you through the different programming challenges will definitely stick the knowledge to your brain. A great learn-by-doing book!"

> — **Johnny Halife, Chief Architect, *Mural.ly***

"*Head First C#* is a comprehensive guide to learning C# that reads like a conversation with a friend. The many coding challenges keep it fun, even when the concepts are tough."

> — **Rebeca Duhn-Krahn, founding partner at Semphore Solutions**

"I've never read a computer book cover to cover, but this one held my interest from the first page to the last. If you want to learn C# in depth and have fun doing it, this is THE book for you."

> — **Andy Parker, fledgling C# programmer**

"It's hard to really learn a programming language without good engaging examples, and this book is full of them! *Head First C#* will guide beginners of all sorts to a long and productive relationship with C# and the .NET Framework."

> —**Chris Burrows, developer for Microsoft's C# Compiler team**

"With *Head First C#*, Andrew and Jenny have presented an excellent tutorial on learning C#. It is very approachable while covering a great amount of detail in a unique style. If you've been turned off by more conventional books on C#, you'll love this one."

> —**Jay Hilyard, software developer, co-author of *C# 3.0 Cookbook***

"I'd reccomend this book to anyone looking for a great introduction into the world of programming and C#. From the first page onwards, the authors walks the reader through some of the more challenging concepts of C# in a simple, easy-to-follow way. At the end of some of the larger projects/labs, the reader can look back at their programs and stand in awe of what they've accomplished."

> —**David Sterling, developer for Microsoft's Visual C# Compiler team**

"*Head First C#* is a highly enjoyable tutorial, full of memorable examples and entertaining exercises. Its lively style is sure to captivate readers—from the humorously annotated examples, to the Fireside Chats, where the abstract class and interface butt heads in a heated argument! For anyone new to programming, there's no better way to dive in."

> —**Joseph Albahari, C# Design Architect at Egton Medical Information Systems, the UK's largest primary healthcare software supplier, co-author of *C# 3.0 in a Nutshell***

"[*Head First C#*] was an easy book to read and understand. I will recommend this book to any developer wanting to jump into the C# waters. I will recommend it to the advanced developer that wants to understand better what is happening with their code. [I will recommend it to developers who] want to find a better way to explain how C# works to their less-seasoned developer friends."

—Giuseppe Turitto, C# and ASP.NET developer for Cornwall Consulting Group

"Andrew and Jenny have crafted another stimulating Head First learning experience. Grab a pencil, a computer, and enjoy the ride as you engage your left brain, right brain, and funny bone."

—Bill Mietelski, software engineer

"Going through this *Head First C#* book was a great experience. I have not come across a book series which actually teaches you so well.…This is a book I would definitely recommend to people wanting to learn C#"

—Krishna Pala, MCP

Praise for other *Head First books*

"I feel like a thousand pounds of books have just been lifted off of my head."

—Ward Cunningham, inventor of the Wiki and founder of the Hillside Group

"Just the right tone for the geeked-out, casual-cool guru coder in all of us. The right reference for practical development strategies—gets my brain going without having to slog through a bunch of tired stale professor-speak."

—Travis Kalanick, Founder of Scour and Red Swoosh
Member of the MIT TR100

"There are books you buy, books you keep, books you keep on your desk, and thanks to O'Reilly and the Head First crew, there is the penultimate category, Head First books. They're the ones that are dog-eared, mangled, and carried everywhere. *Head First SQL* is at the top of my stack. Heck, even the PDF I have for review is tattered and torn."

— Bill Sawyer, ATG Curriculum Manager, Oracle

"This book's admirable clarity, humor and substantial doses of clever make it the sort of book that helps even non-programmers think well about problem-solving."

— Cory Doctorow, co-editor of Boing Boing
Author, *Down and Out in the Magic Kingdom*
and *Someone Comes to Town, Someone Leaves Town*

"I received the book yesterday and started to read it…and I couldn't stop. This is definitely très 'cool.' It is fun, but they cover a lot of ground and they are right to the point. I'm really impressed."

> — **Erich Gamma, IBM Distinguished Engineer, and co-author of**
> ***Design Patterns***

"One of the funniest and smartest books on software design I've ever read."

> — **Aaron LaBerge, VP Technology, ESPN.com**

"What used to be a long trial and error learning process has now been reduced neatly into an engaging paperback."

> — **Mike Davidson, CEO, Newsvine, Inc.**

"Elegant design is at the core of every chapter here, each concept conveyed with equal doses of pragmatism and wit."

> — **Ken Goldstein, Executive Vice President, Disney Online**

"Usually when reading through a book or article on design patterns, I'd have to occasionally stick myself in the eye with something just to make sure I was paying attention. Not with this book. Odd as it may sound, this book makes learning about design patterns fun.

"While other books on design patterns are saying 'Bueller… Bueller… Bueller…' this book is on the float belting out 'Shake it up, baby!'"

> — **Eric Wuehler**

"I literally love this book. In fact, I kissed this book in front of my wife."

> — **Satish Kumar**

This book is dedicated to the loving memory of Sludgie the Whale,
who swam to Brooklyn on April 17, 2007.
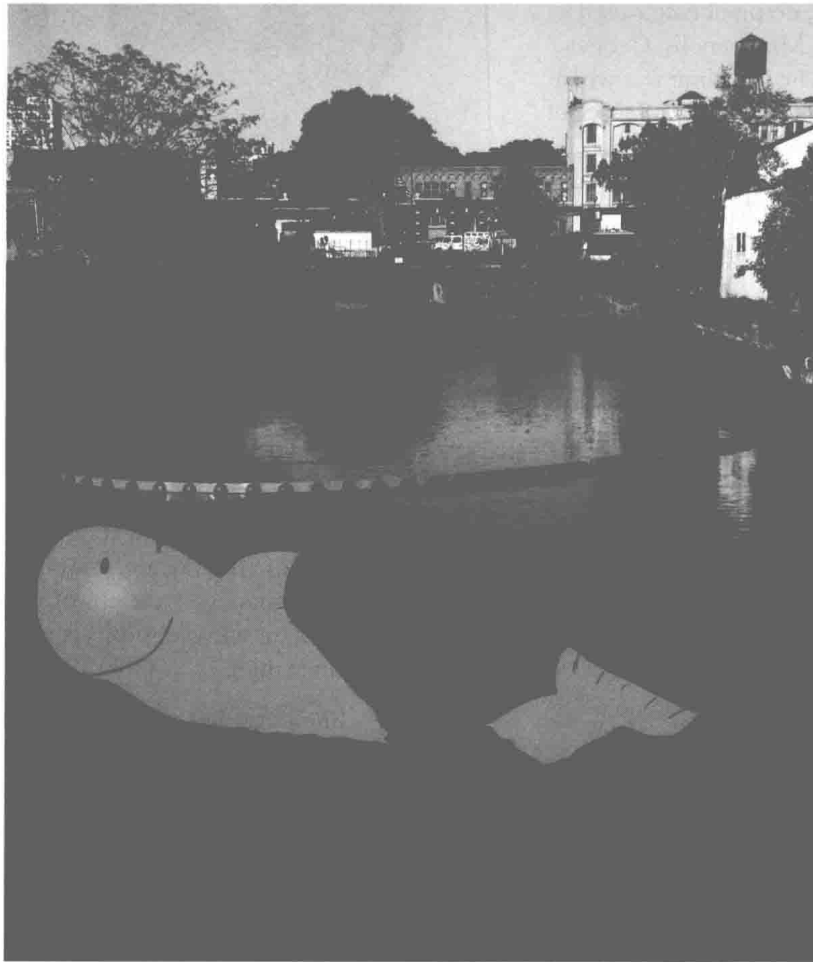
You were only in our canal for a day,
but you'll be in our hearts forever.

the authors

> THANKS FOR BUYING OUR BOOK! WE REALLY LOVE WRITING ABOUT THIS STUFF, AND WE HOPE YOU GET A KICK OUT OF READING IT...

> ...BECAUSE WE KNOW YOU'RE GOING TO HAVE A **GREAT TIME** LEARNING C#.

Andrew

This photo (and the photo of the Gowanus Canal) by Nisha Sondhe

Jenny

Andrew Stellman, despite being raised a New Yorker, has lived in Minneapolis, Geneva, and Pittsburgh... *twice*. The first time was when he graduated from Carnegie Mellon's School of Computer Science, and then again when he and Jenny were starting their consulting business and writing their first book for O'Reilly.

Andrew's first job after college was building software at a record company, EMI-Capitol Records—which actually made sense, as he went to LaGuardia High School of Music & Art and the Performing Arts to study cello and jazz bass guitar. He and Jenny first worked together at a company on Wall Street that built financial software, where he was managing a team of programmers. Over the years he's been a Vice President at a major investment bank, architected large-scale real-time back end systems, managed large international software teams, and consulted for companies, schools, and organizations, including Microsoft, the National Bureau of Economic Research, and MIT. He's had the privilege of working with some pretty amazing programmers during that time, and likes to think that he's learned a few things from them.

When he's not writing books, Andrew keeps himself busy writing useless (but fun) software, playing both music and video games, practicing taiji and aikido, and owning a Pomeranian.

Jennifer Greene studied philosophy in college but, like everyone else in the field, couldn't find a job doing it. Luckily, she's a great software engineer, so she started out working at an online service, and that's the first time she really got a good sense of what good software development looked like.

She moved to New York in 1998 to work on software quality at a financial software company. She's managed a teams of developers, testers and PMs on software projects in media and finance since then.

She's traveled all over the world to work with different software teams and build all kinds of cool projects.

She loves traveling, watching Bollywood movies, reading the occasional comic book, playing PS3 games, and hanging out with her huge siberian cat, Sascha.

Jenny and Andrew have been building software and writing about software engineering together since they first met in 1998. Their first book, *Applied Software Project Management*, was published by O'Reilly in 2005. Other Stellman and Greene books for O'Reilly include *Beautiful Teams* (2009), and their first book in the Head First series, *Head First PMP* (2007).

They founded Stellman & Greene Consulting in 2003 to build a really neat software project for scientists studying herbicide exposure in Vietnam vets. In addition to building software and writing books, they've consulted for companies and spoken at conferences and meetings of software engineers, architects and project managers.

viii Check out their blog, *Building Better Software*: http://www.stellman-greene.com

Follow @AndrewStellman and @JennyGreene on Twitter

Intro

In this section, we answer the burning question: "So why *DID* they put that in a C# programming book?"

Who is this book for?

If you can answer "yes" to all of these:

(1) Do you want to **learn** C#? ⟵

(2) Do you like to tinker—do you learn by doing, rather than just reading?

(3) Do you prefer **stimulating dinner party conversation** to **dry, dull, academic lectures**?

this book is for you.

Who should probably back away from this book?

If you can answer "yes" to any of these:

(1) Does the idea of writing a lot of code make you bored and a little twitchy?

(2) Are you a kick-butt C++ or Java programmer looking for a reference book?

(3) Are you **afraid to try something different**? Would you rather have a root canal than mix stripes with plaid? Do you believe that a technical book can't be serious if C# concepts are anthropomorphized?

this book is not for you.

[Note from marketing: this book is for anyone with a credit card.]

Do you know another programming language, and now you need to ramp up on C#?

Are you already a good C# developer, but you want to learn more about XAML, Model-View-ViewModel (MVVM), or Windows Store app development?

Do you want to get practice writing lots of code?

If so, then lots of people just like you have used this book to do exactly those things!

No programming experience is required to use this book... just curiosity and interest! Thousands of beginners with no programming experience have already used Head First C# to learn to code. That could be you!

We know what you're thinking.

"How can *this* be a serious C# programming book?"

"What's with all the graphics?"

"Can I actually *learn* it this way?"

And we know what your *brain* is thinking.

Your brain craves novelty. It's always searching, scanning, *waiting* for something unusual. It was built that way, and it helps you stay alive.

So what does your brain do with all the routine, ordinary, normal things you encounter? Everything it *can* to stop them from interfering with the brain's *real* job—recording things that *matter*. It doesn't bother saving the boring things; they never make it past the "this is obviously not important" filter.

How does your brain *know* what's important? Suppose you're out for a day hike and a tiger jumps in front of you, what happens inside your head and body?

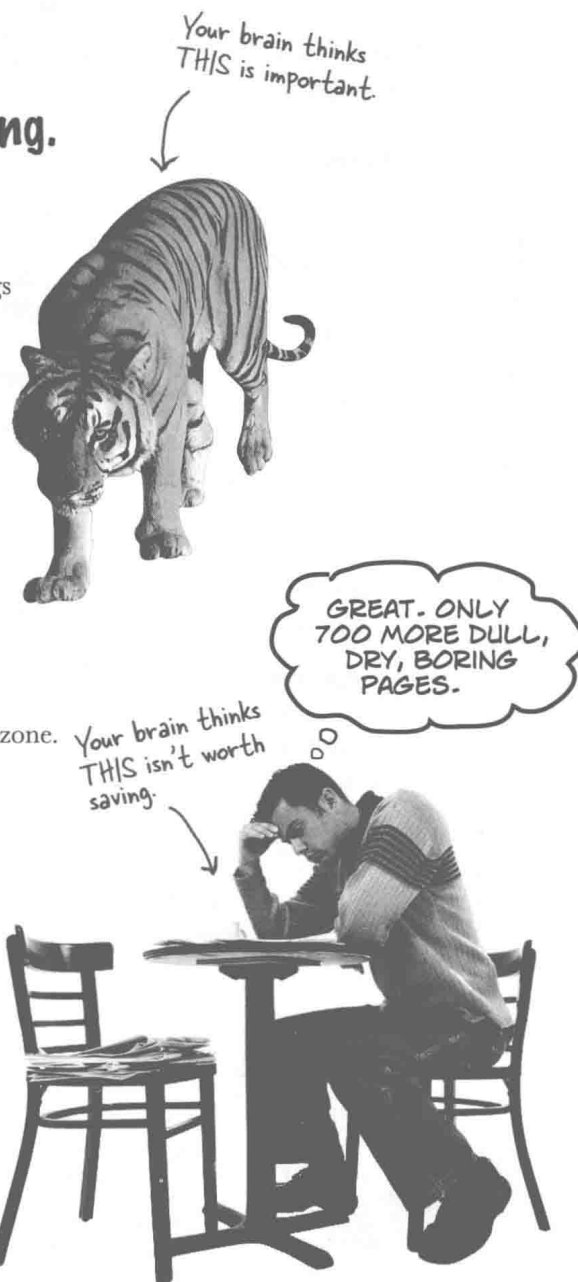Neurons fire. Emotions crank up. *Chemicals surge.*

And that's how your brain knows…

This must be important! Don't forget it!

But imagine you're at home, or in a library. It's a safe, warm, tiger-free zone. You're studying. Getting ready for an exam. Or trying to learn some tough technical topic your boss thinks will take a week, ten days at the most.

Just one problem. Your brain's trying to do you a big favor. It's trying to make sure that this *obviously* non-important content doesn't clutter up scarce resources. Resources that are better spent storing the really *big* things. Like tigers. Like the danger of fire. Like how you should never have posted those "party" photos on your Facebook page.

And there's no simple way to tell your brain, "Hey brain, thank you very much, but no matter how dull this book is, and how little I'm registering on the emotional Richter scale right now, I really *do* want you to keep this stuff around."

Your brain thinks THIS is important.

GREAT. ONLY 700 MORE DULL, DRY, BORING PAGES.

Your brain thinks THIS isn't worth saving.

We *think* of a "Head First" reader as a learner.

So what does it take to *learn* something? First, you have to *get* it, then make sure you don't *forget* it. It's not about pushing facts into your head. Based on the latest research in cognitive science, neurobiology, and educational psychology, *learning* takes a lot more than text on a page. We know what turns your brain on.

Some of the Head First learning principles:
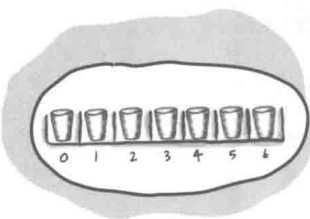
Make it visual. Images are far more memorable than words alone, and make learning much more effective (up to 89% improvement in recall and transfer studies). It also makes things more understandable. **Put the words within or near the graphics** they relate to, rather than on the bottom or on another page, and learners will be up to *twice* as likely to solve problems related to the content.

Use a conversational and personalized style. In recent studies, students performed up to 40% better on post-learning tests if the content spoke directly to the reader, using a first-person, conversational style rather than taking a formal tone. Tell stories instead of lecturing. Use casual language. Don't take yourself too seriously. Which would *you* pay more attention to: a stimulating dinner party companion, or a lecture?

Get the learner to think more deeply. In other words, unless you actively flex your neurons, nothing much happens in your head. A reader has to be motivated, engaged, curious, and inspired to solve problems, draw conclusions, and generate new knowledge. And for that, you need challenges, exercises, and thought-provoking questions, and activities that involve both sides of the brain and multiple senses.

Get—and keep—the reader's attention. We've all had the "I really want to learn this but I can't stay awake past page one" experience. Your brain pays attention to things that are out of the ordinary, interesting, strange, eye-catching, unexpected. Learning a new, tough, technical topic doesn't have to be boring. Your brain will learn much more quickly if it's not.

Touch their emotions. We now know that your ability to remember something is largely dependent on its emotional content. You remember what you care about. You remember when you *feel* something. No, we're not talking heart-wrenching stories about a boy and his dog. We're talking emotions like surprise, curiosity, fun, "what the…?", and the feeling of "I Rule!" that comes when you solve a puzzle, learn something everybody else thinks is hard, or realize you know something that "I'm more technical than thou" Bob from engineering *doesn't*.

Metacognition: thinking about thinking

If you really want to learn, and you want to learn more quickly and more deeply, pay attention to how you pay attention. Think about how you think. Learn how you learn.

Most of us did not take courses on metacognition or learning theory when we were growing up. We were *expected* to learn, but rarely *taught* to learn.

But we assume that if you're holding this book, you really want to learn how to build programs in C#. And you probably don't want to spend a lot of time. If you want to use what you read in this book, you need to *remember* what you read. And for that, you've got to *understand* it. To get the most from this book, or *any* book or learning experience, take responsibility for your brain. Your brain on *this* content.

The trick is to get your brain to see the new material you're learning as Really Important. Crucial to your well-being. As important as a tiger. Otherwise, you're in for a constant battle, with your brain doing its best to keep the new content from sticking.

> I WONDER HOW I CAN TRICK MY BRAIN INTO REMEMBERING THIS STUFF...

So just how *DO* you get your brain to treat C# like it was a hungry tiger?

There's the slow, tedious way, or the faster, more effective way. The slow way is about sheer repetition. You obviously know that you *are* able to learn and remember even the dullest of topics if you keep pounding the same thing into your brain. With enough repetition, your brain says, "This doesn't *feel* important to him, but he keeps looking at the same thing *over* and *over* and *over*, so I suppose it must be."

The faster way is to do **anything that increases brain activity,** especially different *types* of brain activity. The things on the previous page are a big part of the solution, and they're all things that have been proven to help your brain work in your favor. For example, studies show that putting words *within* the pictures they describe (as opposed to somewhere else in the page, like a caption or in the body text) causes your brain to try to makes sense of how the words and picture relate, and this causes more neurons to fire. More neurons firing = more chances for your brain to *get* that this is something worth paying attention to, and possibly recording.

A conversational style helps because people tend to pay more attention when they perceive that they're in a conversation, since they're expected to follow along and hold up their end. The amazing thing is, your brain doesn't necessarily *care* that the "conversation" is between you and a book! On the other hand, if the writing style is formal and dry, your brain perceives it the same way you experience being lectured to while sitting in a roomful of passive attendees. No need to stay awake.

But pictures and conversational style are just the beginning.
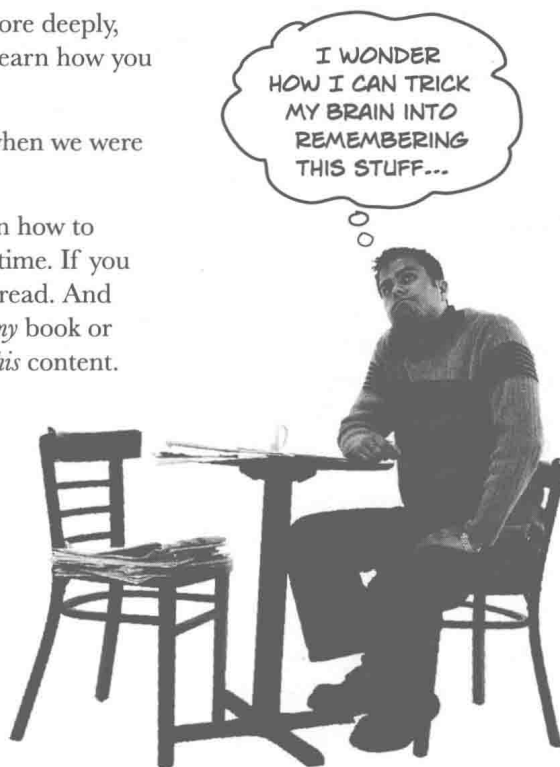
Here's what WE did:

We used *pictures*, because your brain is tuned for visuals, not text. As far as your brain's concerned, a picture really *is* worth a thousand words. And when text and pictures work together, we embedded the text *in* the pictures because your brain works more effectively when the text is *within* the thing the text refers to, as opposed to in a caption or buried in the text somewhere.

We used *redundancy*, saying the same thing in *different* ways and with different media types, and *multiple senses*, to increase the chance that the content gets coded into more than one area of your brain.

We used concepts and pictures in *unexpected* ways because your brain is tuned for novelty, and we used pictures and ideas with at least *some* *emotional* *content*, because your brain is tuned to pay attention to the biochemistry of emotions. That which causes you to *feel* something is more likely to be remembered, even if that feeling is nothing more than a little *humor*, *surprise*, or *interest*.

We used a personalized, *conversational style*, because your brain is tuned to pay more attention when it believes you're in a conversation than if it thinks you're passively listening to a presentation. Your brain does this even when you're *reading*.

We included dozens of *activities*, because your brain is tuned to learn and remember more when you *do* things than when you *read* about things. And we made the paper puzzles and code exercises challenging-yet-do-able, because that's what most people prefer.
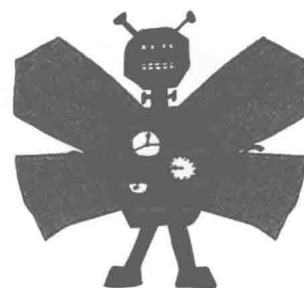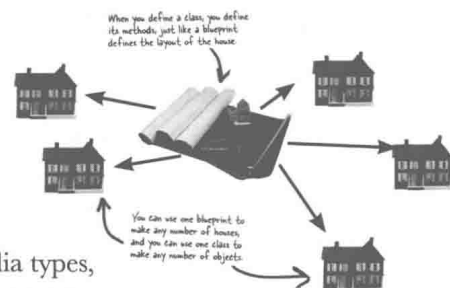
We used *multiple learning styles*, because *you* might prefer step-by-step procedures, while someone else wants to understand the big picture first, and someone else just wants to see an example. But regardless of your own learning preference, *everyone* benefits from seeing the same content represented in multiple ways.

BULLET POINTS

We include content for *both sides of your brain*, because the more of your brain you engage, the more likely you are to learn and remember, and the longer you can stay focused. Since working one side of the brain often means giving the other side a chance to rest, you can be more productive at learning for a longer period of time.

And we included *stories* and exercises that present *more than one point of view,* because your brain is tuned to learn more deeply when it's forced to make evaluations and judgments.

Fireside Chats

We included *challenges*, with exercises, and by asking *questions* that don't always have a straight answer, because your brain is tuned to learn and remember when it has to *work* at something. Think about it—you can't get your *body* in shape just by *watching* people at the gym. But we did our best to make sure that when you're working hard, it's on the *right* things. That *you're not spending one extra dendrite* processing a hard-to-understand example, or parsing difficult, jargon-laden, or overly terse text.
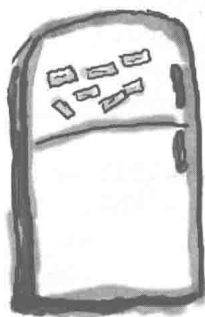
We used *people*. In stories, examples, pictures, etc., because, well, because *you're* a person. And your brain pays more attention to *people* than it does to *things*.
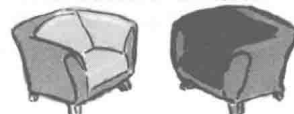
Here's what YOU can do to bend your brain into submission

So, we did our part. The rest is up to you. These tips are a starting point; listen to your brain and figure out what works for you and what doesn't. Try new things.

Cut this out and stick it on your refrigerator.

---

1 **Slow down. The more you understand, the less you have to memorize.**

Don't just *read*. Stop and think. When the book asks you a question, don't just skip to the answer. Imagine that someone really *is* asking the question. The more deeply you force your brain to think, the better chance you have of learning and remembering.

2 **Do the exercises. Write your own notes.**

We put them in, but if we did them for you, that would be like having someone else do your workouts for you. And don't just *look* at the exercises. **Use a pencil.** There's plenty of evidence that physical activity *while* learning can increase the learning.

3 **Read the "There are No Dumb Questions"**

That means all of them. They're not optional sidebars—*they're part of the core content!* Don't skip them.

4 **Make this the last thing you read before bed. Or at least the last challenging thing.**

Part of the learning (especially the transfer to long-term memory) happens *after* you put the book down. Your brain needs time on its own, to do more processing. If you put in something new during that processing time, some of what you just learned will be lost.

5 **Drink water. Lots of it.**

Your brain works best in a nice bath of fluid. Dehydration (which can happen before you ever feel thirsty) decreases cognitive function.

6 **Talk about it. Out loud.**

Speaking activates a different part of the brain. If you're trying to understand something, or increase your chance of remembering it later, say it out loud. Better still, try to explain it out loud to someone else. You'll learn more quickly, and you might uncover ideas you hadn't known were there when you were reading about it.

7 **Listen to your brain.**

Pay attention to whether your brain is getting overloaded. If you find yourself starting to skim the surface or forget what you just read, it's time for a break. Once you go past a certain point, you won't learn faster by trying to shove more in, and you might even hurt the process.

8 **Feel something.**

Your brain needs to know that this *matters*. Get involved with the stories. Make up your own captions for the photos. Groaning over a bad joke is *still* better than feeling nothing at all.

9 **Write a lot of software!**

There's only one way to learn to program: **writing a lot of code**. And that's what you're going to do throughout this book. Coding is a skill, and the only way to get good at it is to practice. We're going to give you a lot of practice: every chapter has exercises that pose a problem for you to solve. Don't just skip over them—a lot of the learning happens when you solve the exercises. We included a solution to each exercise—don't be afraid to **peek at the solution** if you get stuck! (It's easy to get snagged on something small.) But try to solve the problem before you look at the solution. And definitely get it working before you move on to the next part of the book.

What you need for this book:

We wrote this book using **Visual Studio Express 2012 for Windows 8** and **Visual Studio Express 2012 for Windows Desktop**. All of the screenshots that you see throughout the book were taken from those two editions of Visual Studio, so we recommend that you use them. You can also use Visual Studio 2012 Professional, Premium, Ultimate or Test Professional editions, but you'll see some small differences (but nothing that will cause problems with the coding exercises throughout the book).

SETTING UP VISUAL STUDIO 2012 EXPRESS EDITIONS

★ You can download **Visual Studio Express 2012 for Windows 8** *for free* from Microsoft's website. It installs cleanly alongside other editions, as well as previous versions: http://www.microsoft.com/visualstudio/eng/downloads

Click the "Install Now" link to launch the web installer, which automatically downloads and installs Visual Studio.

You'll also need to generate a product key, which is free for the Express editions (but requires you to create a Microsoft.com account).
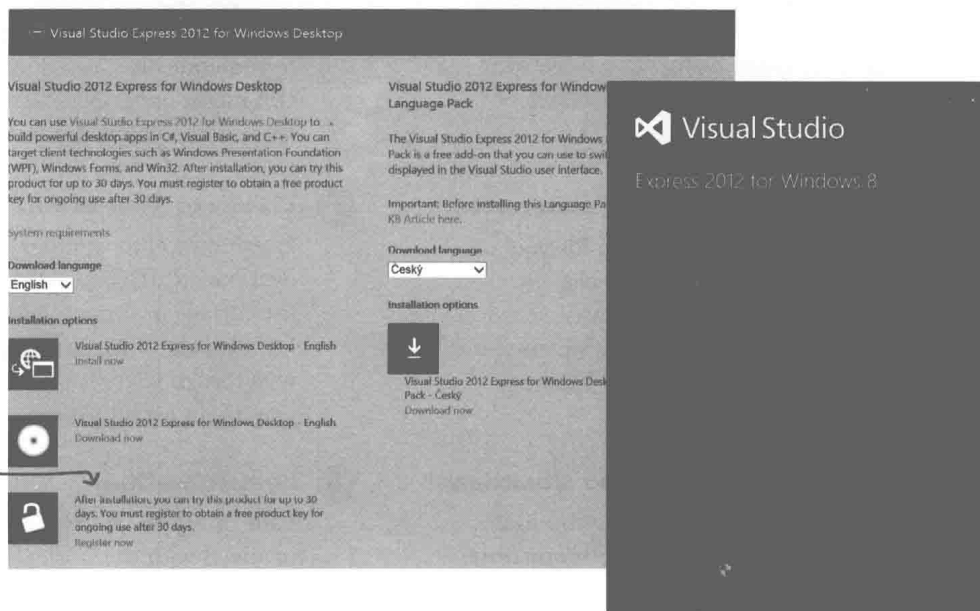
★ Once you've got it installed, you'll need to do the same thing for **Visual Studio Express 2012 for Windows Desktop**.

What to do if you don't have Windows 8 or can't run Visual Studio 2012

Many of the coding exercises in this book require Windows 8. But we definitely understand that some of our readers may not be running it—for example, a lot of professional programmers have office computers that are running operating systems as old as Windows 2003, or only have Visual Studio 2010 installed and cannot upgrade it. **If you're one of these readers, don't worry**—you can still do *almost* every exercise in this book. Here's how:

★ The exercises in chapters 3 through 9 the first two labs do not require Windows 8 at all. You'll even be able to do them using Visual Studio 2010 (and even 2008), although the screenshots may differ a bit from what you see.

★ For the rest of the book, **you'll need to build Windows Presentation Foundation (WPF) desktop apps** instead of Windows 8 apps. We've put together a PDF that you can download from the Head First Labs website (*http://headfirstlabs.com/hfcsharp*) to help you out with this. *Flip to leftover #11 in the appendix to learn more.*