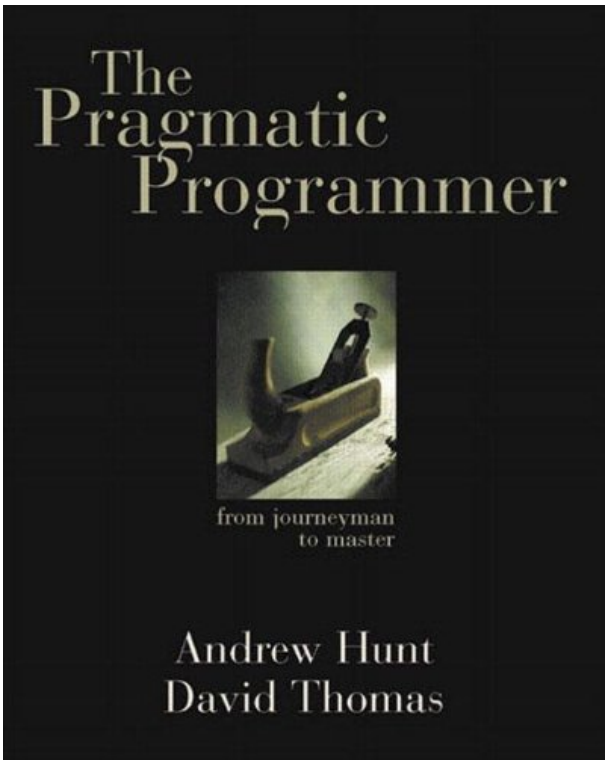
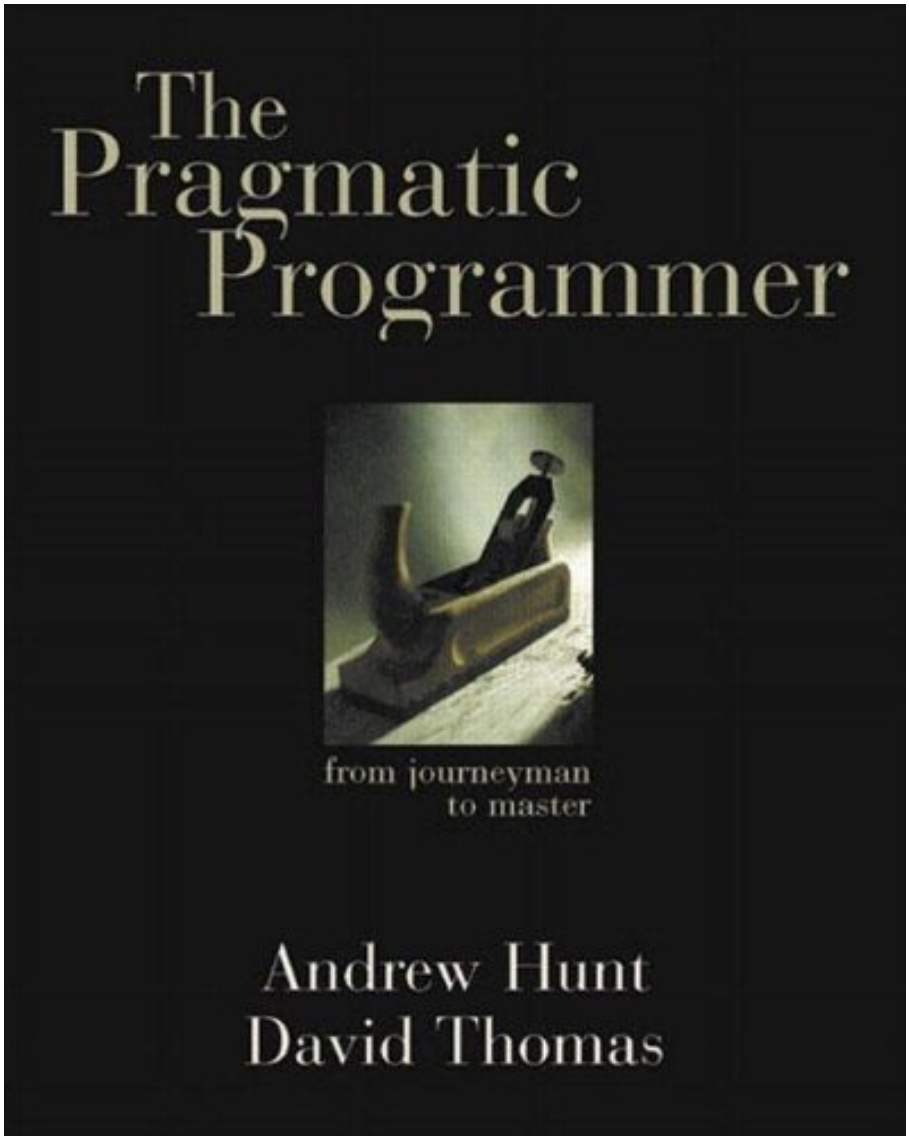
THE PRAGMATIC PROGRAMMER: FROM JOURNEYMAN TO MASTER BY ANDREW HUNT, DAVID THOMAS





THE PRAGMATIC PROGRAMMER: FROM JOURNEYMAN TO MASTER BY ANDREW HUNT, DAVID THOMAS PDF

New updated! The **The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas** from the most effective author and publisher is now available here. This is the book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas that will make your day reading comes to be completed. When you are seeking the published book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas of this title in the book store, you may not find it. The issues can be the restricted versions The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas that are given up guide establishment.

Amazon.com Review
Programmers are craftspeople trained to use a certain set of tools (editors, object managers, version trackers) to generate a certain kind of product (programs) that will operate in some environment (operating systems on hardware assemblies). Like any other craft, computer programming has spawned a body of wisdom, most of which isn't taught at universities or in certification classes. Most programmers arrive at the so-called tricks of the trade over time, through independent experimentation. In The Pragmatic Programmer, Andrew Hunt and David Thomas codify many of the truths they've discovered during their respective careers as designers of software and writers of code.

Some of the authors' nuggets of pragmatism are concrete, and the path to their implementation is clear. They advise readers to learn one text editor, for example, and use it for everything. They also recommend the use of version-tracking software for even the smallest projects, and promote the merits of learning regular expression syntax and a text-manipulation language. Other (perhaps more valuable) advice is more light-hearted. In the debugging section, it is noted that, "if you see hoof prints think horses, not zebras." That is, suspect everything, but start looking for problems in the most obvious places. There are recommendations for making estimates of time and expense, and for integrating testing into the development process. You'll want a copy of The Pragmatic Programmer for two reasons: it displays your own accumulated wisdom more cleanly than you ever bothered to state it, and it introduces you to methods of work that you may not yet have considered. Working programmers will enjoy this book. --David Wall

Topics covered: A useful approach to software design and construction that allows for efficient, profitable development of high-quality products. Elements of the approach include specification development, customer relations, team management, design practices, development tools, and testing procedures. This approach is presented with the help of anecdotes and technical problems.

From the Publisher
As a reviewer I got an early opportunity to read the book you are holding. It was great, even in draft form. Dave Thomas and Andy Hunt have something to say, and they know how to say it. I saw what they were doing and I knew it would work. I asked to write this foreword so that I could explain why.

Simply put, this book tells you how to program in a way that you can follow. You wouldn't think that that would be a hard thing to do, but it is. Why? For one thing, not all programming books are written by programmers. Many are compiled by language designers, or the journalists who work with them to promote their creations. Those books tell you how to talk in a programming language---which is certainly important, but that is only a small part of what a programmer does.

What does a programmer do besides talk in programming language? Well, that is a deeper issue. Most programmers would have trouble explaining what they do. Programming is a job filled with details, and keeping track of those details requires focus. Hours drift by and the code appears. You look up and there are all of those statements. If you don't think carefully, you might think that programming is just typing statements in a programming language. You would be wrong, of course, but you wouldn't be able to tell by looking around the programming section of the bookstore.

In The Pragmatic Programmer Dave and Andy tell us how to program in a way that we can follow. How did they get so smart? Aren't they just as focused on details as other programmers? The answer is that they paid attention to what they were doing while they were doing it---and then they tried to do it better.

Imagine that you are sitting in a meeting. Maybe you are thinking that the meeting could go on forever and that you would rather be programming. Dave and Andy would be thinking about why they were having the meeting, and wondering if there is something else they could do that would take the place of the meeting, and deciding if that something could be automated so that the work of the meeting just happens in the future. Then they would do it.

That is just the way Dave and Andy think. That meeting wasn't something keeping them from programming. It was programming. And it was programming that could be improved. I know they think this way because it is tip number two: Think About Your Work.

So imagine that these guys are thinking this way for a few years. Pretty soon they would have a collection of solutions. Now imagine them using their solutions in their work for a few more years, and discarding the ones that are too hard or don't always produce results. Well, that approach just about defines pragmatic. Now imagine them taking a year or two more to write their solutions down. You might think, That information would be a gold mine. And you would be right.

The authors tell us how they program. And they tell us in a way that we can follow. But there is more to this second statement than you might think. Let me explain.

The authors have been careful to avoid proposing a theory of software development. This is fortunate, because if they had they would be obliged to warp each chapter to defend their theory. Such warping is the tradition in, say, the physical sciences, where theories eventually become laws or are quietly discarded. Programming on the other hand has few (if any) laws. So programming advice shaped around wanna-be laws may sound good in writing, but it fails to satisfy in practice. This is what goes wrong with so many methodology books.

I've studied this problem for a dozen years and found the most promise in a device called a pattern language. In short, a pattern is a solution, and a pattern language is a system of solutions that reinforce each other. A whole community has formed around the search for these systems.

This book is more than a collection of tips. It is a pattern language in sheep's clothing. I say that because each tip is drawn from experience, told as concrete advice, and related to others to form a system. These are the characteristics that allow us to learn and follow a pattern language. They work the same way here.

You can follow the advice in this book because it is concrete. You won't find vague abstractions. Dave and Andy write directly for you, as if each tip was a vital strategy for energizing your programming career. They make it simple, they tell a story, they use a light touch, and then they follow that up with answers to questions that will come up when you try.

And there is more. After you read ten or fifteen tips you will begin to see an extra dimension to the work. We sometimes call it QWAN, short for the quality without a name. The book has a philosophy that will ooze into your consciousness and mix with your own. It doesn't preach. It just tells what works. But in the telling more comes through. That's the beauty of the book: It embodies its philosophy, and it does so unpretentiously.

So here it is: an easy to read---and use---book about the whole practice of programming. I've gone on and on about why it works. You probably only care that it does work. It does. You will see. --Ward Cunningham

From the Author
Other books by the Pragmatic Programmers:
* Pragmatic Project Automation (0974514039)
* Pragmatic Unit Testing in C# with Nunit (0974514020)
* Pragmatic Unit testing in Java with Junit (0974514012)
* Pragmatic Version Control Using CVS (0974514004)
* Programming Ruby 2nd Edition (0974514055)

THE PRAGMATIC PROGRAMMER: FROM JOURNEYMAN TO MASTER BY ANDREW HUNT, DAVID THOMAS PDF

New upgraded! The **The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas** from the very best writer as well as author is currently offered here. This is the book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas that will make your day reading becomes finished. When you are seeking the published book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas of this title in guide store, you may not find it. The troubles can be the minimal versions The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas that are given up guide establishment.

Reviewing routine will certainly consistently lead individuals not to completely satisfied reading *The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas*, an e-book, ten book, hundreds publications, and also a lot more. One that will make them really feel pleased is completing reviewing this publication The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas and obtaining the notification of guides, after that discovering the various other following e-book to check out. It continues an increasing number of. The moment to complete reviewing a book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas will be always different relying on spar time to invest; one example is this The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas

Now, exactly how do you recognize where to get this book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas Never mind, now you might not visit the book shop under the bright sunlight or night to look the publication The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas We below consistently assist you to find hundreds type of publication. One of them is this publication entitled The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas You may go to the link page offered in this collection and after that opt for downloading and install. It will certainly not take even more times. Simply link to your web gain access to and also you can access guide The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas on the internet. Certainly, after downloading and install The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas, you could not print it.

THE PRAGMATIC PROGRAMMER: FROM JOURNEYMAN TO MASTER BY ANDREW HUNT, DAVID THOMAS PDF

-- Ward Cunningham Straight from the programming trenches, The Pragmatic Programmer cuts through the increasing specialization and technicalities of modern software development to examine the core process--taking a requirement and producing working, maintainable code that delights its users. It covers topics ranging from personal responsibility and career development to architectural techniques for keeping your code flexible and easy to adapt and reuse. Read this book, and youll learn how to *Fight software rot; *Avoid the trap of duplicating knowledge; *Write flexible, dynamic, and adaptable code; *Avoid programming by coincidence; *Bullet-proof your code with contracts, assertions, and exceptions; *Capture real requirements; *Test ruthlessly and effectively; *Delight your users; *Build teams of pragmatic programmers; and *Make your developments more precise with automation. Written as a series of self-contained sections and filled with entertaining anecdotes, thoughtful examples, and interesting analogies, The Pragmatic Programmer illustrates the best practices and major pitfalls of many different aspects of software development. Whether youre a new coder, an experienced programm

- Sales Rank: #8101 in Books
- Color: Black
- Brand: Hunt, Andrew/ Thomas, David
- Published on: 1999-10-30
- Original language: English
- Number of items: 1
- Dimensions: 9.00" h x 1.00" w x 7.30" l, 1.41 pounds
- Binding: Paperback
- 352 pages

Amazon.com Review
Programmers are craftspeople trained to use a certain set of tools (editors, object managers, version trackers) to generate a certain kind of product (programs) that will operate in some environment (operating systems on hardware assemblies). Like any other craft, computer programming has spawned a body of wisdom, most of which isn't taught at universities or in certification classes. Most programmers arrive at the so-called tricks of the trade over time, through independent experimentation. In The Pragmatic Programmer, Andrew Hunt and David Thomas codify many of the truths they've discovered during their respective careers as designers of software and writers of code.

Some of the authors' nuggets of pragmatism are concrete, and the path to their implementation is clear. They advise readers to learn one text editor, for example, and use it for everything. They also recommend the use of version-tracking software for even the smallest projects, and promote the merits of learning regular expression syntax and a text-manipulation language. Other (perhaps more valuable) advice is more light-hearted. In the debugging section, it is noted that, "if you see hoof prints think horses, not zebras." That is, suspect everything, but start looking for problems in the most obvious places. There are recommendations for making estimates of time and expense, and for integrating testing into the development process. You'll want a copy of The Pragmatic Programmer for two reasons: it displays your own accumulated wisdom more

cleanly than you ever bothered to state it, and it introduces you to methods of work that you may not yet have considered. Working programmers will enjoy this book. --David Wall

Topics covered: A useful approach to software design and construction that allows for efficient, profitable development of high-quality products. Elements of the approach include specification development, customer relations, team management, design practices, development tools, and testing procedures. This approach is presented with the help of anecdotes and technical problems.

From the Publisher
As a reviewer I got an early opportunity to read the book you are holding. It was great, even in draft form. Dave Thomas and Andy Hunt have something to say, and they know how to say it. I saw what they were doing and I knew it would work. I asked to write this foreword so that I could explain why.

Simply put, this book tells you how to program in a way that you can follow. You wouldn't think that that would be a hard thing to do, but it is. Why? For one thing, not all programming books are written by programmers. Many are compiled by language designers, or the journalists who work with them to promote their creations. Those books tell you how to talk in a programming language---which is certainly important, but that is only a small part of what a programmer does.

What does a programmer do besides talk in programming language? Well, that is a deeper issue. Most programmers would have trouble explaining what they do. Programming is a job filled with details, and keeping track of those details requires focus. Hours drift by and the code appears. You look up and there are all of those statements. If you don't think carefully, you might think that programming is just typing statements in a programming language. You would be wrong, of course, but you wouldn't be able to tell by looking around the programming section of the bookstore.

In The Pragmatic Programmer Dave and Andy tell us how to program in a way that we can follow. How did they get so smart? Aren't they just as focused on details as other programmers? The answer is that they paid attention to what they were doing while they were doing it---and then they tried to do it better.

Imagine that you are sitting in a meeting. Maybe you are thinking that the meeting could go on forever and that you would rather be programming. Dave and Andy would be thinking about why they were having the meeting, and wondering if there is something else they could do that would take the place of the meeting, and deciding if that something could be automated so that the work of the meeting just happens in the future. Then they would do it.

That is just the way Dave and Andy think. That meeting wasn't something keeping them from programming. It was programming. And it was programming that could be improved. I know they think this way because it is tip number two: Think About Your Work.

So imagine that these guys are thinking this way for a few years. Pretty soon they would have a collection of solutions. Now imagine them using their solutions in their work for a few more years, and discarding the ones that are too hard or don't always produce results. Well, that approach just about defines pragmatic. Now imagine them taking a year or two more to write their solutions down. You might think, That information would be a gold mine. And you would be right.

The authors tell us how they program. And they tell us in a way that we can follow. But there is more to this second statement than you might think. Let me explain.

The authors have been careful to avoid proposing a theory of software development. This is fortunate, because if they had they would be obliged to warp each chapter to defend their theory. Such warping is the

tradition in, say, the physical sciences, where theories eventually become laws or are quietly discarded. Programming on the other hand has few (if any) laws. So programming advice shaped around wanna-be laws may sound good in writing, but it fails to satisfy in practice. This is what goes wrong with so many methodology books.

I've studied this problem for a dozen years and found the most promise in a device called a pattern language. In short, a pattern is a solution, and a pattern language is a system of solutions that reinforce each other. A whole community has formed around the search for these systems.

This book is more than a collection of tips. It is a pattern language in sheep's clothing. I say that because each tip is drawn from experience, told as concrete advice, and related to others to form a system. These are the characteristics that allow us to learn and follow a pattern language. They work the same way here.

You can follow the advice in this book because it is concrete. You won't find vague abstractions. Dave and Andy write directly for you, as if each tip was a vital strategy for energizing your programming career. They make it simple, they tell a story, they use a light touch, and then they follow that up with answers to questions that will come up when you try.

And there is more. After you read ten or fifteen tips you will begin to see an extra dimension to the work. We sometimes call it QWAN, short for the quality without a name. The book has a philosophy that will ooze into your consciousness and mix with your own. It doesn't preach. It just tells what works. But in the telling more comes through. That's the beauty of the book: It embodies its philosophy, and it does so unpretentiously.

So here it is: an easy to read---and use---book about the whole practice of programming. I've gone on and on about why it works. You probably only care that it does work. It does. You will see. --Ward Cunningham

From the Author
Other books by the Pragmatic Programmers:
* Pragmatic Project Automation (0974514039)
* Pragmatic Unit Testing in C# with Nunit (0974514020)
* Pragmatic Unit testing in Java with Junit (0974514012)
* Pragmatic Version Control Using CVS (0974514004)
* Programming Ruby 2nd Edition (0974514055)

Most helpful customer reviews

10 of 10 people found the following review helpful.
Inconsistent, but some gems.
By James Leibert
The good:
- The advice on metaprogramming is probably the strongest part of the book. I really liked the clarity of the examples. The key idea is that you should program the structure and methods in code, and provide the detailed implementation (business rules, algorithms etc.) as data.
- The advice on code generators, little languages and plain text interfaces is solid advice worth reiterating and the sections on these are good. However, I preferred the treatment offered by Jon Bentley in his really excellent Programming Pearls books
- The section on automation is a healthy reminder that programmers should have a "do it once, or automate" philosophy
- The section on contract-driven development is worthy and echoes current practice. Jon Bentley's version is a more theoretical, conceptual approach, less polluted by particular industry practices.
- The sections on test-driven development is fine, but the ideas are much better handled by Kent Beck's

books

The not so good:
- The chapters on Pragmatism, Specification, Team Work and Discipline fit nicely with the general themes of the book, but say very little of practical value. If you are interested in operating in a team environment, this isn't the book you are looking for.
- The section on programming methodologies is so wishy-washy it's hard to know what advice if any to take away from it
- References to specific resources are somewhat out of date (the book was written 15 years ago)

The Pragmatic Programmer is worth the price, but if you are thinking of buying this book because you are a relatively new programmer and are looking for advice, I would strongly suggest first reading the much better books:
Jon Bentley's Programming Pearls and More Programming Pearls
Kernighan and Pike's The Practice of Programming
Kent Beck's Extreme Programming Explained

Another book you might be interested in is Susan Lammers, Programmers at Work. Although it is very old (1989), its interesting that most of the programmers interviewed discuss exactly the same themes picked up in The Pragmatic Programmer.

In case you were also thinking about buying a copy of Code Complete, which is also often recommended to new programmers, also hold off - its far too long and a great deal of it feels like a padded-out outline.

9 of 9 people found the following review helpful.
Too broad and rather outdated
By Gunnar Lium
This book reads like a braindump of all the topics the authors thought could be relevant, with from a few paragraphs to few pages of details on each. It would benefit from much tighter editing, and just a lot fewer topics.

Also, I find it rather annoying how they attempt to establish "pragmatic programmer" as a _thing_. Being pragmatic has it's virtues, but the way they constantly write "a pragmatic programmers does X and Y" is for the most part not at all related to pragmatism, but rather common sense and/or being professional. A pragmatic writer would skip the whole pragmatic programmer agenda, and focus on the content instead.

Furthermore, this book has not aged well. Quite of lot ot the principles are well established today, and many topics are either too focused on specific software, or just plain absurd in their obviousness. (eg. you should use a code editor with syntax highlighting, because that makes it easier to see when you write something wrong).

I would recommended reading "Clean Coder" by Robert C. Martin a hundred times more than this book, and having read that (and perhaps it's companion "Clean Code"), I don't think you would pick up anything new and valuable from this book.

0 of 0 people found the following review helpful.
Programming Meets Self-Improvement
By William P Ross
This book discusses ways in which you can improve yourself as a software engineer. The variety of topics is decent, ranging from general tips, tools, programming, and projects. The book was published in 1999, so it's

quite old at the point, but most of the tips are still relevant today.

One chapter I enjoyed was Chapter 5 which is titled "Your Knowledge Portfolio". It discusses how your knowledge is like an expiring asset, because technologies grow old quickly. Many of the points stress how you invest in yourself regularly to learn new technologies. An analogy is given to investing that "Diversification is the key to long-term success". In the context of programming it means not being cornered into a niche technology because you were not learning.

Some of the tips many programmers will be familiar with such as DRY (Don't Repeat Yourself). The authors provide a balanced opinion of many topics from their experience. While many of the stories are anecdotal, it is to be expected in this sort of book.

I also give a lot of credit to this being one of the first books to describe the meta-view perspective of programming. The authors want to help you become better at your craft.

See all 338 customer reviews...

THE PRAGMATIC PROGRAMMER: FROM JOURNEYMAN TO MASTER BY ANDREW HUNT, DAVID THOMAS PDF

You could conserve the soft data of this book **The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas** It will certainly rely on your extra time and activities to open up and also review this publication The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas soft file. So, you might not hesitate to bring this publication The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas everywhere you go. Simply include this sot data to your gizmo or computer disk to let you read every single time and anywhere you have time.

Amazon.com Review
Programmers are craftspeople trained to use a certain set of tools (editors, object managers, version trackers) to generate a certain kind of product (programs) that will operate in some environment (operating systems on hardware assemblies). Like any other craft, computer programming has spawned a body of wisdom, most of which isn't taught at universities or in certification classes. Most programmers arrive at the so-called tricks of the trade over time, through independent experimentation. In The Pragmatic Programmer, Andrew Hunt and David Thomas codify many of the truths they've discovered during their respective careers as designers of software and writers of code.

Some of the authors' nuggets of pragmatism are concrete, and the path to their implementation is clear. They advise readers to learn one text editor, for example, and use it for everything. They also recommend the use of version-tracking software for even the smallest projects, and promote the merits of learning regular expression syntax and a text-manipulation language. Other (perhaps more valuable) advice is more light-hearted. In the debugging section, it is noted that, "if you see hoof prints think horses, not zebras." That is, suspect everything, but start looking for problems in the most obvious places. There are recommendations for making estimates of time and expense, and for integrating testing into the development process. You'll want a copy of The Pragmatic Programmer for two reasons: it displays your own accumulated wisdom more cleanly than you ever bothered to state it, and it introduces you to methods of work that you may not yet have considered. Working programmers will enjoy this book. --David Wall

Topics covered: A useful approach to software design and construction that allows for efficient, profitable development of high-quality products. Elements of the approach include specification development, customer relations, team management, design practices, development tools, and testing procedures. This approach is presented with the help of anecdotes and technical problems.

From the Publisher
As a reviewer I got an early opportunity to read the book you are holding. It was great, even in draft form. Dave Thomas and Andy Hunt have something to say, and they know how to say it. I saw what they were doing and I knew it would work. I asked to write this foreword so that I could explain why.

Simply put, this book tells you how to program in a way that you can follow. You wouldn't think that that would be a hard thing to do, but it is. Why? For one thing, not all programming books are written by programmers. Many are compiled by language designers, or the journalists who work with them to promote their creations. Those books tell you how to talk in a programming language---which is certainly important, but that is only a small part of what a programmer does.

What does a programmer do besides talk in programming language? Well, that is a deeper issue. Most programmers would have trouble explaining what they do. Programming is a job filled with details, and keeping track of those details requires focus. Hours drift by and the code appears. You look up and there are all of those statements. If you don't think carefully, you might think that programming is just typing statements in a programming language. You would be wrong, of course, but you wouldn't be able to tell by looking around the programming section of the bookstore.

In The Pragmatic Programmer Dave and Andy tell us how to program in a way that we can follow. How did they get so smart? Aren't they just as focused on details as other programmers? The answer is that they paid attention to what they were doing while they were doing it---and then they tried to do it better.

Imagine that you are sitting in a meeting. Maybe you are thinking that the meeting could go on forever and that you would rather be programming. Dave and Andy would be thinking about why they were having the meeting, and wondering if there is something else they could do that would take the place of the meeting, and deciding if that something could be automated so that the work of the meeting just happens in the future. Then they would do it.

That is just the way Dave and Andy think. That meeting wasn't something keeping them from programming. It was programming. And it was programming that could be improved. I know they think this way because it is tip number two: Think About Your Work.

So imagine that these guys are thinking this way for a few years. Pretty soon they would have a collection of solutions. Now imagine them using their solutions in their work for a few more years, and discarding the ones that are too hard or don't always produce results. Well, that approach just about defines pragmatic. Now imagine them taking a year or two more to write their solutions down. You might think, That information would be a gold mine. And you would be right.

The authors tell us how they program. And they tell us in a way that we can follow. But there is more to this second statement than you might think. Let me explain.

The authors have been careful to avoid proposing a theory of software development. This is fortunate, because if they had they would be obliged to warp each chapter to defend their theory. Such warping is the tradition in, say, the physical sciences, where theories eventually become laws or are quietly discarded. Programming on the other hand has few (if any) laws. So programming advice shaped around wanna-be laws may sound good in writing, but it fails to satisfy in practice. This is what goes wrong with so many methodology books.

I've studied this problem for a dozen years and found the most promise in a device called a pattern language. In short, a pattern is a solution, and a pattern language is a system of solutions that reinforce each other. A whole community has formed around the search for these systems.

This book is more than a collection of tips. It is a pattern language in sheep's clothing. I say that because each tip is drawn from experience, told as concrete advice, and related to others to form a system. These are the characteristics that allow us to learn and follow a pattern language. They work the same way here.

You can follow the advice in this book because it is concrete. You won't find vague abstractions. Dave and Andy write directly for you, as if each tip was a vital strategy for energizing your programming career. They make it simple, they tell a story, they use a light touch, and then they follow that up with answers to questions that will come up when you try.

And there is more. After you read ten or fifteen tips you will begin to see an extra dimension to the work. We

sometimes call it QWAN, short for the quality without a name. The book has a philosophy that will ooze into your consciousness and mix with your own. It doesn't preach. It just tells what works. But in the telling more comes through. That's the beauty of the book: It embodies its philosophy, and it does so unpretentiously.

So here it is: an easy to read---and use---book about the whole practice of programming. I've gone on and on about why it works. You probably only care that it does work. It does. You will see. --Ward Cunningham

From the Author
Other books by the Pragmatic Programmers:
* Pragmatic Project Automation (0974514039)
* Pragmatic Unit Testing in C# with Nunit (0974514020)
* Pragmatic Unit testing in Java with Junit (0974514012)
* Pragmatic Version Control Using CVS (0974514004)
* Programming Ruby 2nd Edition (0974514055)

New updated! The **The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas** from the most effective author and publisher is now available here. This is the book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas that will make your day reading comes to be completed. When you are seeking the published book The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas of this title in the book store, you may not find it. The issues can be the restricted versions The Pragmatic Programmer: From Journeyman To Master By Andrew Hunt, David Thomas that are given up guide establishment.