MANNING PUBLICATIONS CO.

A Writing Guide for Manning Authors

(For convenience, both authors and readers are referred to below in the masculine form.)

1.1 ABOUT THIS GUIDE

Welcome! Whether you have already signed a publishing agreement with Manning or not may affect how you read this Guide, but in either case you should find answers here to some of your questions. You will also encounter advice on matters you may not have considered or no advice on something that interests you. We, the Editors at Manning, make an attempt here to advise all our authors, which is to say to put down some general rules of writing. Well, not quite *general* rules—those have books' worth of material to talk about—but special rules that apply to computer books we like to publish. This Guide is unfinished, a work in progress which we add to as we notice gaps and as we learn how to do things better.

If you'd like to read about good writing from the most general point of view you might consider *The Elements of Style* by Strunk and White.

1.1.1 Is adhering to this guide a requirement?

An author must retain his voice, otherwise the book will feel wooden. So we give each author a lot of freedom. But certain common features should exist across our whole line of books, features we think make good books. You will need to learn and adopt these features. This Guide describes them.

Types of Manning books

Manning has two types of books, regular and action guides. Regular Manning books are what the name implies—traditional in organization and presentation. Action guides have titles *XYZ in Action*. They include many examples to encourage self-motivated exploration of the subject. This has important advantages for the reader but these books are more difficult to write. Action guides use lots of figures, annotated code segments, tables, and other standalone elements. These elements tell a self-contained story of their own. They are visually presented as independent and written to be independent of the rest of the text.

1.1.2 Author template

We encourage authors to write using our template. This helps speed the later production of the book and gives the author a reasonable count of the pages he has written A full page of text in the template has about 415 words. Of course, text-only pages are rare and if we see such a page we will make sure it goes away. ☺

1.2 PUBLISHING JARGON AND A FEW RULES

Before we start discussing writing, you need to gain a nodding acquaintance with some publishing terms. We introduce them and present rough "rules" about optimal page counts, etc.

1.2.1 "Spread"

Open a book anywhere. The two pages you see are called a spread. Ancient rules of page design are generally formulated not in terms of a page but a spread. The left and right page columns are to be balanced and of equal length, the margins obey some ratios, and much more. Nowadays typesetting is accessible to the layman and these old rules are not broadly known. We are a traditional publisher and like those old rules, but they have been formulated for what were essentially text-only books—there are reasons, other than ignorance (although that can play a role too), that cause us to violate them at times. The first among them is that computer books contain many elements other than simple text—elements like figures, code segments, tables, etc. To be presented well, these elements require design rules of their own, rules which can conflict with spread design requirements. Anyway, design is something you should accept as important for the ultimate look and feel—and readability—of your book but should not try learning. Authors who try to master page design in order to produce the final pages generally get discouraged, and in the process often do more harm than good to their own book projects.

1.2.2 Front Matter

First impressions are important. Readers often look at the front matter (as well as at the index) before making a decision to buy a book. The front matter should be informative and well-organized, telling the reader what he can expect to find in the book. The front matter is a separate section that uses Roman numerals for page numbers. It also includes the Table of Contents.

The following components can, but do not always, have to be included. Required components are marked with an asterisk.

- Dedication
- Foreword
- * Preface
- * Acknowledgments
- * About this book
- About the author(s)

Preface. Ideally, your preface should tell the story of how you came to write your book. It should be personal, rather than an introduction to the subject matter, telling the reader how you first became interested, and then an expert, in your topic, or what specifically prompted you to write the book. It is always written in the first person. When multiple authors are involved, the first-person plural voice ("we") should be used.

For books with multiple authors, either one author can write the preface on behalf of all, or all authors can contribute their "stories" to the preface; they are then presented sequentially. If the book has a lead author, then that author should write the majority of the preface.

Generally, the preface is not "signed," that is, an author's name does not appear at the end, unless one author is speaking collectively for all the authors.

The length of the preface should be one to two pages, at most.

You can find a sample on the Manning Author's Resource page.

About this book. This section of the front matter serves as a "road map" for navigating your book and should provide a brief overview of all parts and chapters. It should contain the following sections:

- Contents of this book (a brief description of parts, chapters and appendices)
- How to use this book
- Intended audience/ Who should read this book
- Code conventions and downloads
- Software requirements

"About this book" can also include any other sections you think would be useful and informative.

Here's an example:
http://www.manning.com/Neward/About.html

Acknowledgments. When multiple authors are involved, this document usually has two parts:

1. Consolidated thanks

   Includes everyone whom all the authors wish to thank. It is written using the "we" voice. (i.e., We acknowledge... Manning Publications, our outside reviewers...)

2. Personal thanks

   This gives each author an opportunity to thank specific people and is written in the third person ("he/she"). (i.e., John Doe would like to thank..... his spouse, kids, dog, boss...)

1.2.3 Chapters and parts

All chapters in a book should be of similar length. Ideally they will all be short (10–15 pages); long (30–35 pages); or in between (20–25 pages). Of course there will be exceptions, but avoid having too many exceptions. Above all

avoid chapters shorter than 10 pages or longer than 35.

"Parts" are groups of 2–7 consecutive chapters. It's best to have two or three Parts for a short book of 200 to 400 pages; more are fine for longer books, but even very long books of 1,000 pages or so should not have more than 5 Parts.

1.2.4 Chapter title and subheads

Number your chapters consecutively throughout the book—do not start from 1 in a new Part. Name chapter files using two digits as in "Chapter 03" so the files will order correctly when you have more than 10 chapters. Our templates use numbered subheads for the first and second level sections in a chapter. We call them A- and B-heads. So, section 3 in chapter 2 is an A-head with the number 2.3. The fifth B-level subsection of section 2.3 is numbered 2.3.5. C-level and lower heads do not get numbers.

1.2.5 Introduction

Whether you have a first section called "Introduction" in your chapter depends on just how much introductory material you need to cover. But even if there's little of it, and therefore you plan no such section, the chapter should start with some introductory prose. Do not make the first item in your chapter a subhead as in:

WRONG

Chapter 3 **Some Silly Topics**

3.1 Why silly topics are important

Begin silly text here…

The same holds true for headings:

WRONG

3.1 Why silly topics are important

3.1.1 Good reasons for silly topics

Begin text here...

There should always be some text following every heading, including the chapter title.

1.2.6 Summary

While a chapter Introduction is optional, a Summary is not. Summaries are useful to the reader and they are needed in every chapter. They are the place to segue from chapter to chapter. And, of course, they are the place where he will find the essential points. Remember that what's essential is obvious to you. But it's not obvious to anyone struggling to understand and learn and they should find it in the Summary.

The Summary must *not* be a rote rehash of the chapter ("We have covered this, we have seen that"). There's this common idea cruising the land that you, as an author, should "tell them what you will say, say it, and finally tell them what you said." This silly rule is anathema to us. The Summary should not tell them what you told them in any mechanical sense of the word. It should *provide a high level view of the subject and emphasize important nuances and relationships*. These the reader cannot understand at the chapter's beginning, so a Summary can do something an Introduction cannot. You should assume the reader has read the chapter and has gained a nodding acquaintance with the ideas you have covered. If you can, write it so it is useful to someone who has not read the chapter but is browsing it to decide whether it's worth reading.

1.2.7 Figures and tables

All figures and tables are numbered in the n.m style, with n the chapter number and m the consecutive figure number in that chapter. Each figure and table must have a detailed caption. We look for captions that explain the figure or table so that someone reading *only them* (i.e., someone who is browsing the chapter, not reading it) can understand them. Figure three in chapter two is "Figure 2.3" and it is followed by its caption. In the typeset version the table number plus its caption will go above the table. Figure number plus caption will go below the figure.

In addition to the caption, always refer to the numbered element in text as well: "See figure 2.3." or "Table 4.1 lists the methods and their properties..."

1.2.8 Code listings

We distinguish between short code segments, normally an integral part of the surrounding narrative, and longer pieces that have standalone value. The latter are likely to be examined by people browsing through the book without reading the surrounding text. Identify the longer pieces as "listings," number them in the n.m style, and provide brief captions. (Listing captions must be short enough to fit on one line.) Do *not* identify as listings those short code segments that *cannot* be useful in isolation from the narrative to people browsing through the book.

Listings will generally be more difficult or simply take longer to figure out. Consider the reader: what does he need to know before he starts? We found the answer to this by trial and error and were somewhat surprised by the answer. The reader needs an introduction to the code to come but he wants more. He wants to know what the code does, its results, before he dives into it. If the result is a GUI, show it before and not after the listing. If the code's results cannot be shown visually, describe them. But in every case do it before requiring the reader to dive in and struggle with the code. Knowing the results ahead of time will help him understand where the code is going.

1.3 How to develop an effective ToC

For the purpose of selling a book, the table of contents is its most important component. Can you imagine someone buying a book from a bookstore without being attracted by its table of contents? The ToC—a condensed summary and a kind of index of the entire book—remains important after the book has been bought, of course. Treat it respectfully.

1.3.1 A standalone document

Although it is typically a by-product of the way you write your chapter and subhead titles as you write the manuscript, the table of contents needs to read as if it had been written separately. It needs to leave a friendly impression and have continuous flow. The real estate in the table of contents is precious—don't waste it by cluttering it with useless words; or vague words; or hackneyed words. Instead, use action words indicating what the reader will know or be able to do after reading a section.

1.3.2 Should you use Parts?

Our advice to authors is to use them. Group the chapters into related clusters and take advantage of the Part titles to explain the groupings. The Part titles make reading a table of contents easier and give the reader a high level view of the structure of the book. Of course, if you are writing a short book with a simple structure, you can forget Parts.

1.3.3 Be generous with your subheads

As a rule, you should see some subheads on every spread in the book—that's in part why we expect no text-only pages in our books. You might think of A-level sections as 3–5 pages long, B-level sections often less than one page. C-level sections (which in our designs are not numbered) should be short, like B-sections.

1.4 Manning templates

You can download the Manning author's template in Microsoft Word from http://www.manning.com/msg/. We will produce your book with FrameMaker but do not recommend that you write the book in FrameMaker unless you are already comfortable with it. You will find other resources and guidelines, along with the template, on the Manning Style Guide page.

1.5 THINK BEFORE YOU WRITE

One of our advisers says this about his favorite books:

> "They're my favorites because you can open them at any point, and start reading without being too lost; and what you're reading is, as often as possible, about two things at once: one is the main matter of the book (Tibetan grammar, Snobol, etc.), the other is something more localized and interesting (loanwords from Chinese; hashing algorithms, hyphenation, backtracking parsers, closures, etc.). Moreover, each starts with fundamental concepts, and builds on them in complex and interesting ways."

We agree with him:

Foundation. Start from the fundamentals and develop from there. Even if you think your reader knows the fundamentals, it is important to do this so your story is firmly rooted in them.

Valuable details. A chapter usually covers some important area or concept. You have the full 20–30 pages of the chapter to do it in. But remember as you move toward your goal that you must bring the reader along. The best way to do that is to teach him interesting and useful details *every step* of the way.

"Diveability". Try to make it possible to dive in and read at any point. That's what readers do anyway so make it easy for them. We will say more about this when we discuss "browsables" below.

1.5.1 Attitude toward reader

Before starting on the details you should think about the "attitude" you adopt as you write. It will permeate your writing and have a large cumulative effect. As you write, check to see if you've got the "right" attitude. Here are some suggestions how to do that.

KNOW THY READER. Write for a specific reader you can imagine. Who is he? What does he do? What needs does he have? It is impossible to write a successful book without the benefit of a clear focus on the reader. This focus is a background that will allow you to advise, hand-hold, warn, amuse, and even *enchant* him.

Be a MENTOR. Think of yourself as the reader's mentor. Think of him kindly, perhaps as a younger brother. Take him by the hand and help him out. Warn him of common pitfalls. Give him your advice and insights—all the benefits of your deep knowledge of the subject. He'll love it. Above all, as a mentor, make sure the reader has the complete context in which to think about the problem at hand. Often this is easy to do if you just remember that you need to.

Be PRECISE, Be COMPLETE. Remember that the reader is your (and our) customer. Customers are demanding: make the effort to give them *complete* and *precise*

information at every step. To do so requires considerable effort and many revisions, but you must fight the all-too-natural tendency to be hazy—and lazy—about details. You know how to find those details. He doesn't.

RELAX. Authors are sometimes overwhelmed by the apparent importance of the act of writing a book. They get uptight; they write in a stiff and formal style. Fight that tendency like the devil. Give your writing a relaxed feel. Try some humor if you've got it in you (but don't if you do not). In one of our books the author adds zest to some potentially boring passages by exercising programming concepts with the aid of the seven dwarfs; or the six cardinal sins, and even combines the two; or he uses standard but, in the context, incongruous expressions such as "blood, sweat and tears." He explains programming ideas by juggling these non-programming concepts—with great effect.

But DON'T RELAX TOO MUCH. Sometimes authors try to relax their writing by using the casual and imprecise language of verbal communication. That never works either. Because body language and voice color aid verbal communication, an impoverished language can succeed in transmitting clear meaning verbally. In a book, body language and sound cues are absent: words and pictures communicate alone. They have to be precise and crystal clear at every step.

REMEMBER WHAT'S IMPORTANT. And finally, the most important advice: "It's the code, stupid." The code—its quality, conciseness, reusability and its novelty—is what makes computer books a success. More on this in the section "Writing code."

1.6 WRITING TEXT

How can we tell good writing? Is there a process that helps us write well? We don't know the answer to either, really. How can we decide what is a good painting or a way to live a good life. These are questions of the same kind—basically unanswerable. And yet, we have to deal with them: text is what a book primarily is and our authors need all the help we can give them writing it.

We summarize here ideas we've gathered over the years, but we suspect—we know, really—that there's much more to be said. Below is the best advice we can give you at this point. The sections focus on features of good writing like conciseness, steady pace, and so on. They don't give you a formula on how to achieve them. You will need to find your own way.

1.6.1 Be concise

A common writing problem is verboseness; another is uneven pace. Make your writing concise. Give it a steady pace. Most early drafts are verbose and their pace is unsteady—yours will be too. You must make a deliberate effort to solve these problems as you revise.

Humans may well be *genetically* verbose. We have been optimized by evolution to communicate with other humans verbally, not in writing. In evolutionary terms, writing is a recent invention that has had little chance to feed back into our genes. Our speech is rich in unnecessary words like "you know," "umm," "like," "kind of." This is true in all human languages. Whether the verboseness of normal speech is an evolutionary adaptation that has real benefits or is simply an undesirable side effect of the acts of thinking and speaking, it is a fact of life that contributes to the difficulty of your task: there is no benefit to verboseness in writing.

Most of us have had modest training and little experience writing, so it's not a surprise that the way we write is to transcribe on paper the voice we hear in our heads. We write as we speak, and it shows: the result tends to be meandering prose choked with unneeded words.

So, if verboseness is natural, how do we overcome it? Simple:

RULE *Write a first draft without worrying about verboseness, then revise.*

When you start, focus on getting your story down on paper and don't worry about being concise. Then, as you revise, focus on cutting and reducing so the meaning flows easily. Rigorously cut words like *very, much, most, but, and, however, nevertheless, like, tend,...* Consider these words and others like them *forbidden*. Take pleasure in every such word you can toss out and yet keep the meaning clear and flowing smoothly. Keep cutting and reducing mercilessly. Find one word to replace several—in the process you will be forced to find the *right* word. Replace whole paragraphs with shorter versions. To help you do this we have some advice. Cut…

- the first sentence of every paragraph
- the first paragraph of every chapter and
- the first chapter of every manuscript.

While we do not mean these rules entirely literally, they point at a very real and common problem.

In speech, when we start a new topic we use words to play for time while our mind wraps around the subject. These words give us time to warm up. When we write, we often do the same, starting from a needlessly distant starting point. What the reader needs is prose without the warm-up, prose that's well focused. The reader will read your book faster than you can write it. Most of the time he will not need those warm-ups nor quite as distant a starting point. *Cut the warm-ups and begin with the first meaty sentence in your first draft.* Sometimes you will need to go as far as cutting an entire paragraph and even an entire chapter. Your first paragraphs won't get to the point quite fast enough. Neither will your first chapter, which you will be writing *without* knowing what's in the rest of your book (since it does not exist yet). Most of the time, you should consider tossing that chapter when you revise.

But as you are reducing the text remember the main point: the point of cutting is not just conciseness for its own sake. It is to make your meaning easy to follow. That's your goal, cutting is your means.

In Appendix A on page 19 we look at an example of reducing the verboseness of a paragraph.

1.6.2 Go steady

We see many cases of unsteady pace. As he writes, a writer's mind is focused on how to explain the subject, on getting the facts right, on remembering important details. It is difficult for him to simultaneously keep an eye on the pace. Subject experts tend to unknowingly assume background knowledge the typical reader may not possess. Whether he is conscious of it or not, when an expert skips such knowledge he effectively causes his writing to surge forward, only to slow down later as he starts to cover something in detail. When you are revising or rewriting your first draft remember to focus not on the facts but on the reader.

1.6.3 Focus on reader

Take a distance from your thoughts and view the subject from the reader's point of view. Have you given him all the facts he needs to follow your point? Thinking from his perspective, and assuming he does not know the subject, you will inevitably find your text skips over assumptions, makes unrelated transitions, and is difficult to follow. Ask yourself: What are the assumptions you have made? What background knowledge must the reader have? Then revise so the flow is continuous and steady and *starts from the beginning.*

Begin gradually

At the beginning of the book and of any new topic, begin gradually. Review the big picture. Give the reader the context in which the knowledge you plan to cover fits. Tell him what he needs to know to understand what's coming. This will help you get on the same wavelength with him.

You will find yourself struggling with the question of how *much* to say about the prerequisites. You cannot possibly cover everything in detail—other books are devoted to that. But you should not ignore the issue and use terms some may not understand. Again, conflicting requirements will require that you find a balance by exhibiting a fine sense for your reader's needs. Often the best is to review the concepts briefly the first time they are introduced. Say something that will help them feel comfortable with the terms—a mnemonic for the word, or a simple, catchy reason the concept is important. If there are a lot of terms the best may be to write a brief overview and segregate the whole thing into an introductory chapter called something like, "What you need to know first."

However you decide to handle it, at the beginnings, ramp up gradually. The book should start with a ramp-up chapter or two. A chapter should ramp up via an introduction. Many sections will need to ramp up gradually.

Is this in conflict with the 'cutting rule,' above? Not really. That rule suggests that you cut material you wrote in an effort to focus your own thinking. The reader needs ramp-up writing focused on him.

Prerequisite knowledge is by definition outside the focus of your book and must be left out. But, you must review it so the reader knows what it will take to understand the book. By first reading about stuff he knows he relaxes and learns to trust you. It has the effect of bringing him along with you as your story takes off. But you have to walk a fine line. You have to make the introductions concise and interesting. If you repeat what he knows in ways he's seen before, you will bore him; if you skip too much, you will fail to bring him along.

RULE: *Boring your reader is a cardinal sin.*

You must cover well-known material in a fresh and interesting way. *If you catch yourself writing something you have seen elsewhere*, *drop it like a hot potato*: the reader has most likely seen it too. He will immediately begin to doubt you. A trivial example: how many times have you read in introductions to XML that XML has evolved from SGML. That statement, while obviously true, has been repeated so many times it can only do harm—it gives the reader nothing new to catch his interest and attention and bores him with a tired old fact. You should fight the inclination to say the obvious or the hackneyed and should review background information in a fresh manner; try to shed light on it from the point of view of the main focus of the book.

One of our books has two ramp-up chapters that have received a lot of compliments. The author is an advanced practitioner of martial arts. Apparently, once they have passed a test of high proficiency and become masters, martial arts practitioners are required to return to the basic exercises and practice them again. The idea is that masters will perform them with a deepened appreciation of the value of simplicity. This is the mental framework in which our author (Damian Conway, *Object Oriented Perl*) wrote the first two chapters of his book. Appreciative readers often say, "Gee, I once knew that was the way to do it, but had long forgotten it." When written this way, an introductory chapter can be a marvelous way to start a book.

The "crutch"

Whenever you discuss an important or difficult concept start with a "crutch" which motivates the topic through a concrete example. Too many authors happily launch into a new and difficult topic with nary a word of motivation. The crutch is a special kind of introduction, one that describes a scenario you might want to solve and shows how it can be difficult to do so in the absence of the to-be-described concept. Thereafter, the scenario serves as the running context in which the reader can better

understand the discussion of the concept. Here are two examples of a crutch, from two different Perl books:

Example 1. Suppose that you wanted to create a subroutine to return the items that two arrays have in common. This might come up in a program that had to compare a master list of files with a list of files that had changed, in preparation for saving the changes. How could you pass two lists to one subroutine?

You couldn't just pass the lists one right after the other, like this:

```
@master_list = qw(a b c d e f g);_
@change_list = qw(b c f);_
$intersect = get_intersect @master_list @change_list;
```

Because of flattening, `get_intersect` wouldn't receive two arguments. It would get ten, one for each element of both arrays.

```
(a b c d e f g b c f) not (a b c d e f g) (b c f)
```

The solution is to pass references. A *reference* is…

Example 2. Suppose we wanted to implement a subroutine called `listdir` that provides the functionality of our operating system's directory listing command (i.e., `dir` or `ls`). Such a subroutine might take arguments specifying which files to list, what type of files to consider, whether to list hidden files, what details of each file should be reported, whether files and directories should be listed separately, how to sort the listing, whether directories should be listed recursively, how many columns to use, and whether the output should be pages or just dumped. But we certainly don't want to have to specify every one of those nine parameters every time we call `listdir`:

```
Listdir("*", "any", 1, 1, 0, 0, "alpha", 4, 1);
```

Even if we arranged things so that specifying an undefined value for an argument selects a default behavior for that argument, the call is no easier to code and no more readable:

```
Listdir(undef, undef, 1, 1, undef, undef, undef, 4, 1);
```

Some programming languages provide a mechanism for naming the arguments passed to a subroutine. This facility is especially useful when implementing a subroutine like `listdir`, where there are many potential parameters, but only a few of them may be needed for a particular call.

Perl supports named arguments in a cunning way…

Two little things

1. The narrative should include a little "reward" or "encouragement" for the reader upon completing a concept; perhaps just a "That's it! Now you know everything about X."
2. The narrative refers to the reader as "you", e.g. "you'll get a run-time error."

1.6.4 Writing code

The first thing programmers tend to look at in a book is the code. And the code is what they return to many times after they have read the text once. Think deeply about your code and make it rich in meaning. Make it teach as much as possible in as few lines as possible. Show your reader code he has never seen before, code he will learn from in waves as he unearths deeper layers of ideas and techniques.

We distinguish between two kinds of code examples. The first occurs within a narrative and may have little or no value in isolation from the surrounding text. The second has standalone value—it can be understood and it teaches how to do something all by itself. This second kind of code example should be separated as a listing, given a title, and annotated (unless it is less than about half a page).

1.6.5 Code annotations: two styles

1. Stand-alone annotations that sit next to the code only (these should be about 10 words in length).
2. Footnoted annotations. A short annotation sits next to the code (also about 10 words long at most) and the rest of the explanation is contained in a bulleted footnote following the code listing.

Listing 2.1 Product XML document example

```
<?xml version="1.0"?>

<product-catalog>
  <product sku="123456" name="The Product">
    <description locale="en_US">
      An excellent product.
    </description>
    <description locale="es_MX">
      Un producto excellente.
    </description>
    <price locale="en_US" unit="USD">
      99.95
    </price>
    <price locale="es_MX" unit="MXP">
      9999.95
    </price>
  </product>
</product-catalog>
```

Defines a product with SKU=123456 and the name "The Product" — **Standalone annotation**

❶ **Lists descriptions and prices for this product in the U.S. and Mexico** — **Footnoted annotation**

❶ Shows a catalog containing a single product. The product information includes its name, SKU number, description, and price. Note that the document contains multiple price and description nodes, each of which is specific to a locale.

Annotation footnote

The first type of annotation does not get a cue-ball with a number. The second type does get a number—and alerts the reader that more explanation follows. The reader then goes to the footnote with the corresponding number to read more.

As you prepare your annotations, please keep in mind that the numbering conventions are different for stand-alone annotations and for annotations that have footnotes.

1.7 WRITING THE "BROWSABLES"

Manning terminology recognizes two types of chapter content: the narrative and the browsables. We use "narrative" in a perfectly conventional way but "browsable" is our invention. Another piece of private terminology we use is "crutch." These and more are discussed below.

1.7.1 Browsables

Standalone pieces of the manuscript that can be understood on their own are *browsables*. They are visually identifiable as separate and they draw the reader's eye. Each of them contains a complete, self-standing piece of information. Browsables are common things such as a table, definition box, annotated code segment, figure, etc. Browsables are the types of information a "quick study" will find useful—and, possibly, sufficient. Someone who knows the topic and just needs the specifics. This might be a confident reader; or a reader in a hurry; or someone who has read the chapter before and is now returning to it looking for information.

You might imagine the browsable pieces as the writing remaining on a whiteboard after a technical discussion. What would you see? Code segments with words and arrows pointing to pieces of the code; pictures with words and arrows; definitions written out in full; tables with comments attached to them, etc. Of course you are writing a book, not leading a whiteboard discussion, so the browsables are *complete pieces of information*, not sketches. They should be useful even if read in isolation from anything else in the chapter. The 'quick study' may scan the browsable pieces in a chapter and never read the rest.

Examples of browsables

Browsables take various forms. We describe these in more detail later. But what kinds of content should you consider putting into the browsable form? Here are a few examples to get you thinking:

- Show the "big picture" in the form of a diagram that shows how everything fits together;
- Illustrate how the concepts work through several lines of code, annotated so the main points are immediately identified;

- Present rules of syntax that might require a lot of words through an economical device we call a "hedgehog," a brief code segment (usually not more than a short line) with annotations placed around the code in a hedgehog-like arrangement;
- State definitions and principles tersely and abstractly and present them in a specially designed text format (equivalent to a text box);
- Put any factual text with special cases and/or relationships (that often require lots of words) into something more economical and clearer: a table.

1.7.2 How to write a browsable

The "browsables" present *the facts* and *only the facts*—the "what" but not the "why." The reader actively searches through the book to find what he's interested in and unless he is a beginner who knows nothing about the topic, the browsables will tend to tell him much of what he needs.

When writing the browsables remember that they will not be read in order; so try to make them independent of previous parts of the book.

A list of browsables

Here are some suggestions on the most useful browsables.

Annotated code. For a computer book this is the most important type. The more advanced programmers will go straight to the code, look at it, and figure out what they need just from that. Write the annotated code segment so the advanced reader can get away with doing just that. Manning has two annotation styles and you will generally use one or the other:

- Line annotations—set within the code listing
- Word annotations—set next to (and sometimes following) the code

Hedgehog diagram. If you are explaining some syntax, use a short piece of code that allows you to show the syntax in place and define the rules through annotations of the example spread around it in a hedgehog arrangement. As a rule, write any statement of rules of syntax in the form of a hedgehog diagram *instead* of prose.

Table. Things you want to present systematically, so they include conditions or special cases, relationships, multiple examples or values, are candidates for a table. A table must have a long, detailed caption. The caption should say enough to make the table stand alone," i.e., be understandable (and interesting) without reading the narrative.

Definition. An important definition should be presented as a visually separate item—a text box. Whether or not the defined concept is important enough to deserve to stand out visually, its definition must be written properly. Authors commonly "define" a concept by stating what it does.

That's wrong. Instead, a definition must state what the concept *is*. A definition *must* identify the concept's location in the conceptual framework of which it is a part. The way to do that is to identify the broader concept (*genus*) of which the concept is a special case *and* identify one or more of its distinguishing features (*differentiae*). For example, we define an insulator by identifying the genus in which it belongs—the genus of all materials—and then specifying what sets it apart from other materials:

An insulator is a material that does not conduct electricity.

A common mistake is skipping the genus, as in "a conductor conducts electricity." Yes, it does, but so does my hand. Is my hand a conductor? Well, not really because when we say "conductor" we mean a type of material, which a hand is not. This ambiguity does not arise when the genus is properly identified: a conductor is a material that conducts electricity. For a complete discussion of this topic see
http://www.manning.com/Kovitz/Chapter15.html#definitions

Figures. So many computer books are impoverished visually, we sometimes wonder whether the programming community has a cultural bias against pictures. And then, the few illustrations you do find in books are often overly complicated, requiring a large effort for the reader to decipher. We expect Manning authors to do better. Our rule is:

Every important concept should be illustrated.

Do not try to make the picture complex. In fact, try *not* to make the picture complex. The simpler, the better. A picture showing two boxes connected with an arrow, plus a few words, is effective. Our typesetters will make the picture pretty and small and place it on the page so the text flows around it. That way the benefits are gained without wasted space and without the eye traveling too far from the text it is reading. Letting such a simple picture take up half a page insults the reader's intelligence and esthetic sense. Making it small is esthetically appealing.

Pictures

- help explain the concept(s)
- emphasize the concept(s)
- help the reader navigate the book

Figure captions are an integral part of the figure and they need to be detailed enough so that someone browsing the book can understand the point of the illustration without reading the surrounding narrative. People who look through a book first look at the illustrations and read their captions. Captions are a great opportunity to draw the reader—and the bookstore browser—into your book. And, exactly the same applies to tables.

Screen shots. They play the usual role of illustrating how something looks on the screen. We like to keep our screen shots small, so plan for the important parts to be visible. That means you must crop the image so only the necessary parts are shown. Cut out the windows frames unless they are really needed—they are tiresome. Who wants to see them in a book? Also, remember that the image will be printed in black ink, the colors all becoming shades of gray. This makes features of the image even harder to distinguish so look for strong contrasts between light and dark.

A detailed caption is recommended, nay, required.

Information maps. An *information map* is a table-like representation of a list, a procedure, or a set of rules and requirements. It's not a conventional table, though. It transforms list-like information into a structured table in which the number of steps is reduced and related substeps are clustered into related steps. For example

Step	Description
1	Grumpy opens the door and leaves
2	Bashful follows him
3	Sleepy is the last to leave
4	They walk and sing, "Hi-ho, Hi-ho …"

These four steps can be grouped into two:

Step	Description
1. Dwarfs leave for work	Grumpy first He opens door, steps out and walks Bashful and Sleepy follow They fall in line behind Grumpy
2. They walk	Single-file and sing in unison: "Hi-ho, Hi-ho …"

Information maps effectively take typesetting to another level, but they are not just a matter of visual presentation: they require writing that expressly organizes and groups flat lists into fewer items of more complex, related information. This should not be strange to anyone who has heard of the advantages of object-oriented programming over procedural programming. Take a look at the examples at www.infomap.com. In the left column of this page find the heading "The Method" and then link to "Demos." Take a look at some of the before and after demos to get an idea of how to use information maps.

Minimal pairs. Now here's a beautiful technique. Suppose you're trying to explain the 'step' attribute in the following Java code. Put the code and its output into two side-by-side columns as follows:

SOURCE:	SOURCE:
<jx:forEach var="c" status="s" items="$customers" step="**1**"> <jx:expr value="$s.index"/> <jx:expr value="$c"/> </jx:forEach>	<jx:forEach var="c" status="s" items="$customers" step="**2**"> <jx:expr value="$s.index"/> <jx:expr value="$c"/> </jx:forEach>
OUTPUT: 1. Andrew Aaronson 2. Brian Bournelli 3. Charles Colworth 4. Daniel Devoe 5. Erica Ellbridge	OUTPUT: 1. Andrew Aaronson 3. Charles Colworth 5. Erica Ellbridge

... with appropriate differences bolded, "**1**" vs. "**2**" in the source code. Such a minimal pair might be more valuable in cases where the differences are more subtle, but this example makes it clear how to use it. You can set one up easily in a table with one row and two columns.

1.8 RESOURCES FOR AUTHORS

Books

- *The Chicago Manual of Style* (14th Edition)
- *Webster's Tenth New Collegiate Dictionary*
- *Wired Style* (from the editors of Wired magazine)
- Elbow, Peter, *Writing with Power,* (2nd Edition), Oxford University Press, Oxford, 1998.
- Mager, Robert F., *The How to Write a Book Book*, Atlanta, Georgia: The Center for Effective Performance, Inc., 1991.

Web sites

- StudioB, http://www.studiob.com
- Infomap, http://www.infomap.com
- Online Paradigm Writing Assistant, http://www.powa.org/
- Merriam-Webster Online, http://www.m-w.com/
- yourDictionary.com, http://www.yourdictionary.com/
- InstantWeb Online Computing Dictionary,
- http://www.instantweb.com/d/dictionary/index.html

APPENDIX A

Reducing words (13%)

This Appendix shows an example of revising a short piece of text to make it shorter. We picked a Warranty Disclaimer we found on the laptop on which this was being written. You might think this was an easy target for our editorial scissors, but it turned out to be quite the opposite—in trying to reduce a warranty so it's easy to understand we were under a major constraint: not to cut anything lawyers might consider important. What made that hard was that we had no idea what the lawyers might think. Anyway, here it is.

BEFORE. The SimCity 2000 Urban Renewal Kit (SCURK from here on out) does many things that the designers of SimCity 2000 did not anticipate. As a result, lots of changes you can make to your graphics or saved cities in SCURK may have unpredictable effects when actually loaded back into SimCity 2000. We have taken every feasible precaution to protect against serious complications, such as crashing SimCity 2000 or confusing the simulator, but cannot guarantee that everything you do will work perfectly or even at all. *(85 words, 427 characters)*

AFTER. SimCity 2000 inevitably has errors, or "bugs," which cause it to behave unpredictably. Changes you make to the graphics or the cities can cause a SimCity 2000 file to misbehave. We have tried to avoid serious bugs which could crash SimCity 2000 or confuse the simulator, but cannot guarantee that this product will work perfectly, or even at all, when your changes are included. *(64 words, 315 characters)*

Notice that we have not dramatically reduced the word count, just 13%. The attempt at reduction did result in a to-the-point feel with simpler and more meaningful words replacing some vague jargon in the original. We feel the revised version is considerably easier to understand and, of course, that was the goal.