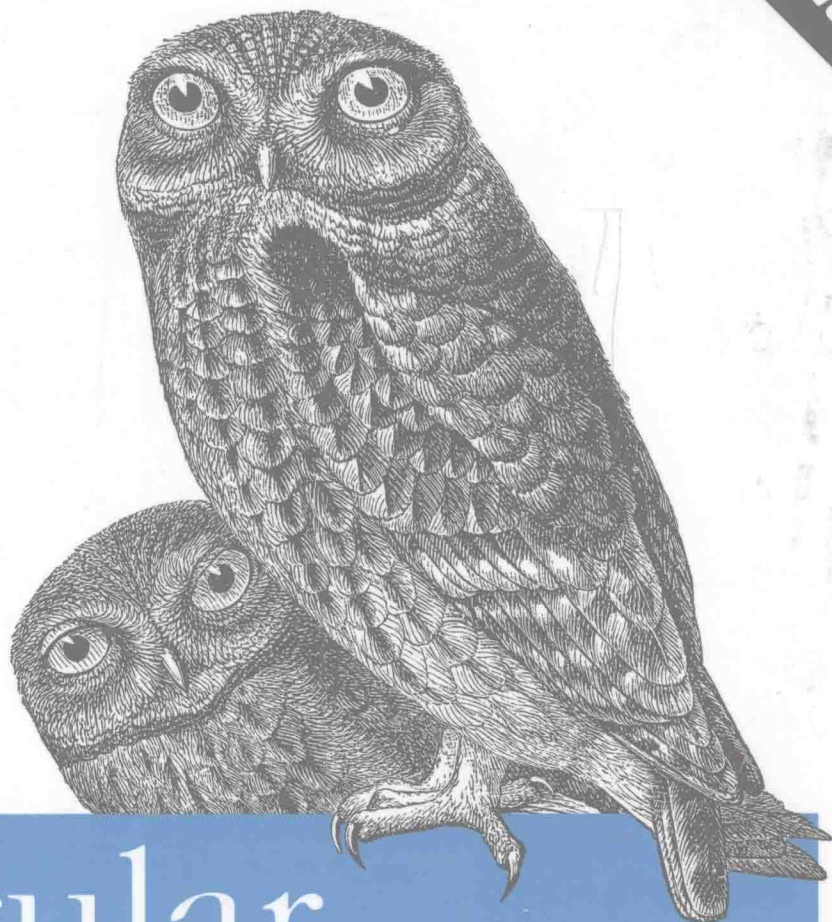


精通正则表达式(影印版)

2nd Edition



*Mastering*

# Regular Expressions

O'REILLY®

東南大學出版社

Jeffrey E. F. Friedl 著

第二版

---

精通正则表达式(影印版)  
**Mastering Regular Expressions**

*Jeffrey E. F. Friedl*

**O'REILLY®**

*Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo*

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

## 图书在版编目 (CIP) 数据

精通正则表达式: 第2版 / (美) 弗里德尔 (Friedl, E. F. J.)

著. — 影印版. — 南京: 东南大学出版社, 2005.9

书名原文: Mastering Regular Expressions, Second Edition

ISBN 7-5641-0041-9

I. 精... II. 弗... III. 正则表达式—英文 IV. TP301.2

中国版本图书馆 CIP 数据核字 (2005) 第 059125 号

江苏省版权局著作权合同登记

图字: 10-2005-083 号

©2002 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2005.  
Authorized reprint of the original English edition, 2002 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2002。

英文影印版由东南大学出版社出版 2005。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式重制。

书 名 / 精通正则表达式 (影印版)

书 号 / ISBN 7-5641-0041-9

责任编辑 / 张烨

封面设计 / Edie Freedman, 张健

出版发行 / 东南大学出版社

地 址 / 南京四牌楼 2 号 邮编 210096

印 刷 / 扬中市印刷有限公司

开 本 / 787 毫米 × 1092 毫米 16 开本 30.5 印张

版 次 / 2005 年 9 月第 1 版 2005 年 9 月第 1 次印刷

印 数 / 0001-2500 册

定 价 / 72.00 元 (册)

---

# *Preface*

This book is about a powerful tool called “regular expressions”. It teaches you how to use regular expressions to solve problems and get the most out of tools and languages that provide them. Most documentation that mentions regular expressions doesn’t even begin to hint at their power, but this book is about *mastering* regular expressions.

Regular expressions are available in many types of tools (editors, word processors, system tools, database engines, and such), but their power is most fully exposed when available as part of a programming language. Examples include Java and JScript, Visual Basic and VBScript, JavaScript and ECMAScript, C, C++, C#, elisp, Perl, Python, Tcl, Ruby, PHP, sed, and awk. In fact, regular expressions are the very heart of many programs written in some of these languages.

There’s a good reason that regular expressions are found in so many diverse languages and applications: they are extremely powerful. At a low level, a regular expression describes a chunk of text. You might use it to verify a user’s input, or perhaps to sift through large amounts of data. On a higher level, regular expressions allow you to master your data. Control it. Put it to work for you. To master regular expressions is to master your data.

## *The Need for This Book*

I finished the first edition of this book in late 1996, and wrote it simply because there was a need. Good documentation on regular expressions just wasn’t available, so most of their power went untapped. Regular-expression documentation was available, but it centered on the “low-level view.” It seemed to me that they were analogous to showing someone the alphabet and expecting them to learn to speak.

## *Why I've Written the Second Edition*

In the five and a half years since the first edition of this book was published, the world of regular expressions expanded considerably. The regular expressions of almost every tool and language became more powerful and expressive. Perl, Python, Tcl, Java, and Visual Basic all got new regular-expression backends. New languages with regular expression support, like Ruby, PHP, and C#, were developed and became popular. During all this time, the basic core of the book—how to truly understand regular expressions and how to get the most from them—maintained as important and relevant as ever.

Gradually, the first edition started to show its age. It needed updating to reflect the new languages and features, as well as the expanding role that regular expressions play in today's Internet world. When I decided to update the first edition, it was with a promise to my wife that it would take no more than three months. Two years later, luckily still married, almost the entire book has been rewritten from scratch. It's good, though, that it took so long, for it brought me into 2002, a particularly active year for regular expressions. In early 2002, both Java 1.4 (with `java.util.regex`) and Microsoft's .NET were released, and Perl 5.8 was released that summer. They are all covered fully in this book.

## *Intended Audience*

This book will interest anyone who has an opportunity to use regular expressions. If you don't yet understand the power that regular expressions can provide, you should benefit greatly as a whole new world is opened up to you. This book should expand your understanding, even if you consider yourself an accomplished regular-expression expert. After the first edition, it wasn't uncommon for me to receive an email that started "*I thought I knew regular expressions until I read Mastering Regular Expressions. Now I do.*"


Programmers working on text-related tasks, such as web programming, will find an absolute gold mine of detail, hints, tips, and *understanding* that can be put to immediate use. The detail and thoroughness is simply not found anywhere else.

Regular expressions are an idea—one that is implemented in various ways by various utilities (many, many more than are specifically presented in this book). If you master the general concept of regular expressions, it's a short step to mastering a particular implementation. This book concentrates on that idea, so most of the knowledge presented here transcends the utilities and languages used to present the examples.

## *How to Read This Book*

This book is part tutorial, part reference manual, and part story, depending on when you use it. Readers familiar with regular expressions might feel that they can immediately begin using this book as a detailed reference, flipping directly to the section on their favorite utility. I would like to discourage that.

To get the most out of this book, read the first six chapters as a story. I have found that certain habits and ways of thinking can be a great help to reaching a full understanding, but such things are absorbed over pages, not merely memorized from a list.

This book tells a story, but one with many details. Once you've read the story to get the overall picture, this book is also useful as a reference. The last three chapters (covering specifics of Perl, Java, and .NET) rely heavily on your having read the first six chapters. To help you get the most from each part, I've used cross references liberally, and I've worked hard to make the index as useful as possible. (Cross references are often presented as “” followed by a page number.)

Until you read the full story, this book's use as a reference makes little sense. Before reading the story, you might look at one of the tables, such as the chart on page 91, and think it presents all the relevant information you need to know. But a great deal of background information does not appear in the charts themselves, but rather in the associated story. Once you've read the story, you'll have an appreciation for the issues, what you can remember off the top of your head, and what is important to check up on.

## *Organization*

The nine chapters of this book can be logically divided into roughly three parts. Here's a quick overview:

### **The Introduction**

- Chapter 1 introduces the concept of regular expressions.
- Chapter 2 takes a look at text processing with regular expressions.
- Chapter 3 provides an overview of features and utilities, plus a bit of history.

### **The Details**

- Chapter 4 explains the details of how regular expressions work.
- Chapter 5 works through examples, using the knowledge from Chapter 4.
- Chapter 6 discusses efficiency in detail.

### **Tool-Specific Information**

- Chapter 7 covers Perl regular expressions in detail.
- Chapter 8 looks at regular-expression packages for Java.
- Chapter 9 looks at .NET's language-neutral regular-expression package.

## The Introduction

The introduction elevates the absolute novice to “issue-aware” novice. Readers with a fair amount of experience can feel free to skim the early chapters, but I particularly recommend Chapter 3 even for the grizzled expert.

- Chapter 1, *Introduction to Regular Expressions*, is geared toward the complete novice. I introduce the concept of regular expressions using the widely available program *egrep*, and offer my perspective on how to *think* regular expressions, instilling a solid foundation for the advanced concepts presented in later chapters. Even readers with former experience would do well to skim this first chapter.
- Chapter 2, *Extended Introductory Examples*, looks at real text processing in a programming language that has regular-expression support. The additional examples provide a basis for the detailed discussions of later chapters, and show additional important thought processes behind crafting advanced regular expressions. To provide a feel for how to “speak in regular expressions,” this chapter takes a problem requiring an advanced solution and shows ways to solve it using two unrelated regular-expression-wielding tools.
- Chapter 3, *Overview of Regular Expression Features and Flavors*, provides an overview of the wide range of regular expressions commonly found in tools today. Due to their turbulent history, current commonly-used regular-expression flavors can differ greatly. This chapter also takes a look at a bit of the history and evolution of regular expressions and the programs that use them. The end of this chapter also contains the “Guide to the Advanced Chapters.” This guide is your road map to getting the most out of the advanced material that follows.

## The Details

Once you have the basics down, it’s time to investigate the *how* and the *why*. Like the “teach a man to fish” parable, truly understanding the issues will allow you to apply that knowledge whenever and wherever regular expressions are found.

- Chapter 4, *The Mechanics of Expression Processing*, ratchets up the pace several notches and begins the central core of this book. It looks at the important inner workings of how regular expression engines really work from a *practical* point of view. Understanding the details of how regular expressions are handled goes a very long way toward allowing you to master them.
- Chapter 5, *Practical Regex Techniques*, then puts that knowledge to high-level, practical use. Common (but complex) problems are explored in detail, all with the aim of expanding and deepening your regular-expression experience.

- Chapter 6, *Crafting an Efficient Expression*, looks at the real-life efficiency ramifications of the regular expressions available to most programming languages. This chapter puts information detailed in Chapters 4 and 5 to use for exploiting an engine's strengths and stepping around its weaknesses.

## *Tool-Specific Information*

Once the lessons of Chapters 4, 5, and 6 are under your belt, there is usually little to say about specific implementations. However, I've devoted an entire chapter to each of three popular systems:

- Chapter 7, *Perl*, closely examines regular expressions in Perl, arguably the most popular regular-expression-laden programming language in use today. It has only four operators related to regular expressions, but their myriad of options and special situations provides an extremely rich set of programming options—and pitfalls. The very richness that allows the programmer to move quickly from concept to program can be a minefield for the uninitiated. This detailed chapter clears a path.
- Chapter 8, *Java*, surveys the landscape of regular-expression packages available for Java. Points of comparison are discussed, and two packages with notable strengths are covered in more detail.
- Chapter 9, *.NET*, is the documentation for the .NET regular-expression library that Microsoft neglected to provide. Whether using VB.NET, C#, C++, JScript, VBScript, ECMAScript, or any of the other languages that use .NET components, this chapter provides the details you need to employ .NET regular-expressions to the fullest.

## *Typographical Conventions*





When doing (or talking about) detailed and complex text processing, being precise is important. The mere addition or subtraction of a space can make a world of difference, so I've used the following special conventions in typesetting this book:

- A regular expression generally appears like `this`. Notice the thin corners which flag “this is a regular expression.” Literal text (such as that being searched) generally appears like `'this'`. At times, I'll leave off the thin corners or quotes when obviously unambiguous. Also, code snippets and screen shots are always presented in their natural state, so the quotes and corners are not used in such cases.
- I use visually distinct ellipses within literal text and regular expressions. For example `[...]` represents a set of square brackets with unspecified contents, while `[ . . . ]` would be a set containing three periods.



- Without special presentation, it is virtually impossible to know how many spaces are between the letters in “a b”, so when spaces appear in regular expressions and selected literal text, they are presented with the ‘`␣`’ symbol. This way, it will be clear that there are exactly four spaces in ‘`a␣␣␣␣b`’.

I also use visual tab, newline, and carriage-return characters. Here’s a summary of the four:



	a space character
	a tab character
	a newline character
	a carriage-return character

- At times, I use underlining or shade the background to highlight parts of literal text or a regular expression. In this example the underline shows where in the text the expression actually matches:

Because `[cat]` matches ‘`It␣indicates␣your␣cat␣is␣`’ instead of the word ‘`cat`’, we realize . . .


In this example the underlines highlight what has just been added to an expression under discussion:

To make this useful, we can wrap `[Subject|Date]` with parentheses, and append a colon and a space. This yields `[(Subject|Date) :␣]`.

- This book is full of details and examples, so to help you get the most out of it, I’ve provided an extensive set of cross references. They often appear in the text in a “123” notation, which means “see page 123.” For example, it might appear like “. . . is described in Table 8-1 ( 373).”

## Exercises

Occasionally, and particularly in the early chapters, I’ll pose a question to highlight the importance of the concept under discussion. They’re not there just to take up space; I really do want you to try them before continuing. Please. So as not to dilute their importance, I’ve sprinkled only a few throughout the entire book. They also serve as checkpoints: if they take more than a few moments, it’s probably best to go over the relevant section again before continuing on.

To help entice you to actually think about these questions as you read them, I’ve made checking the answers a breeze: just turn the page. Answers to questions marked with  are always found by turning just one page. This way, they’re out of sight while you think about the answer, but are within easy reach.

## *Links, Code, Errata, and Contacts*

I learned the hard way with the first edition that URLs change more quickly than a printed book can be updated, so rather than providing an appendix of URLs, I'll provide just one:

`http://regex.info/`

There you can find regular-expression links, many of the code snippets from this book, a searchable index, and much more. In the unlikely event this book contains an error :-), the errata will be available as well.

If you find an error in this book, or just want to drop me a note, you can contact me at `jfriedl@regex.info`.

The publisher can be contacted at:

O'Reilly & Associates, Inc.  
1005 Gravenstein Highway North  
Sebastopol, CA 95472  
(800) 998-9938 (in the United States or Canada)  
(707) 829-0515 (international/local)  
(707) 829-0104 (fax)  
`bookquestions@oreilly.com`

For more information about books, conferences, Resource Centers, and the O'Reilly Network, see the O'Reilly web site at:

`http://www.oreilly.com`

## *Personal Comments and Acknowledgments*

Writing the first edition of this book was a grueling task that took two and a half years and the help of many people. After the toll it took on my health and sanity, I promised that I'd never put myself through such an experience again.

I've many people to thank for helping me break that promise. Foremost is my wife, Fumie. If you find this book useful, thank her; without her support and understanding, I would have never had the sanity to make it through what turned out to be almost a two year complete rewrite.

I also appreciate the support of Yahoo! Inc., where I have enjoyed slinging regular expressions for five years, and my manager Mike Bennett. His flexibility and understanding allowed this project to happen.

While researching and writing this book, many people helped educate me on languages or systems I didn't know, and more still reviewed and corrected drafts as the manuscript developed. In particular, I'd like to thank my brother, Stephen Friedl, for his meticulous and detailed reviews of the manuscript. The book is much better because of them.

I'd also like to thank William F. Maton, Dean Wilson, Derek Balling, Jarkko Hietaniemi, Jeremy Zawodny, Ethan Nicholas, Kasia Trapszo, Jeffrey Papen, Dr. Yadong Li, Daniel F. Savarese, David Flanagan, Kristine Rudkin, Shawn Purcell, Josh Woodward, Ray Goldberger, and my editor, Andy Oram. Also thanks to O'Reilly's Linda Mui for navigating this book through the pre-publication minefield and keeping the troops rallied, and Jessamyn Reed for creating the new figures this edition required.

Special thanks for providing an insider's look at Java go to Mike "madbot" McCloskey, Mark Reinhold, and Dr. Cliff Click, all of Sun Microsystems. For .NET insight, I'd like to thank David Gutierrez and Kit George, of Microsoft.

I'd like to thank Dr. Ken Lunde of Adobe Systems, who created custom characters and fonts for a number of the typographical aspects of this book. The Japanese characters are from Adobe Systems' *Heisei Mincho W3* typeface, while the Korean is from the Korean Ministry of Culture and Sports *Munhwa* typeface. It's also Ken who originally gave me the guiding principle that governs my writing: "you do the research so your readers don't have to."

For help in setting up the server for <http://regex.info>, I'd like to thank Jeffrey Papen and Peak Web Hosting (<http://www.PeakWebhosting.com/>).

---

# Table of Contents

<i>Preface</i> .....	<i>xvii</i>
<i>1: Introduction to Regular Expressions</i> .....	<i>1</i>
Solving Real Problems .....	2
Regular Expressions as a Language .....	4
The Filename Analogy .....	4
The Language Analogy .....	5
The Regular-Expression Frame of Mind .....	6
If You Have Some Regular-Expression Experience .....	6
Searching Text Files: Egrep .....	6
Egrep Metacharacters .....	8
Start and End of the Line .....	8
Character Classes .....	9
Matching Any Character with Dot .....	11
Alternation .....	13
Ignoring Differences in Capitalization .....	14
Word Boundaries .....	15
In a Nutshell .....	16
Optional Items .....	17
Other Quantifiers: Repetition .....	18
Parentheses and Backreferences .....	20
The Great Escape .....	22
Expanding the Foundation .....	23
Linguistic Diversification .....	23
The Goal of a Regular Expression .....	23

A Few More Examples .....	23
Regular Expression Nomenclature .....	27
Improving on the Status Quo .....	30
Summary .....	32
Personal Glimpses .....	33
<b>2: <i>Extended Introductory Examples</i> .....</b>	<b>35</b>
About the Examples .....	36
A Short Introduction to Perl .....	37
Matching Text with Regular Expressions .....	38
Toward a More Real-World Example .....	40
Side Effects of a Successful Match .....	40
Intertwined Regular Expressions .....	43
Intermission .....	49
Modifying Text with Regular Expressions .....	50
Example: Form Letter .....	50
Example: Prettifying a Stock Price .....	51
Automated Editing .....	53
A Small Mail Utility .....	53
Adding Commas to a Number with Lookaround .....	59
Text-to-HTML Conversion .....	67
That Doubled-Word Thing .....	77
<b>3: <i>Overview of Regular Expression Features and Flavors</i> .....</b>	<b>83</b>
A Casual Stroll Across the Regex Landscape .....	85
The Origins of Regular Expressions .....	85
At a Glance .....	91
Care and Handling of Regular Expressions .....	93
Integrated Handling .....	94
Procedural and Object-Oriented Handling .....	95
A Search-and-Replace Example .....	97
Search and Replace in Other Languages .....	99
Care and Handling: Summary .....	101
Strings, Character Encodings, and Modes .....	101
Strings as Regular Expressions .....	101
Character-Encoding Issues .....	105
Regex Modes and Match Modes .....	109
Common Metacharacters and Features .....	112
Character Representations .....	114

Character Classes and Class-Like Constructs .....	117
Anchors and Other “Zero-Width Assertions” .....	127
Comments and Mode Modifiers .....	133
Grouping, Capturing, Conditionals, and Control .....	135
Guide to the Advanced Chapters .....	141
<b>4: The Mechanics of Expression Processing .....</b>	<b>143</b>
Start Your Engines! .....	143
Two Kinds of Engines .....	144
New Standards .....	144
Regex Engine Types .....	145
From the Department of Redundancy Department .....	146
Testing the Engine Type .....	146
Match Basics .....	147
About the Examples .....	147
Rule 1: The Match That Begins Earliest Wins .....	148
Engine Pieces and Parts .....	149
Rule 2: The Standard Quantifiers Are Greedy .....	151
Regex-Directed Versus Text-Directed .....	153
NFA Engine: Regex-Directed .....	153
DFA Engine: Text-Directed .....	155
First Thoughts: NFA and DFA in Comparison .....	156
Backtracking .....	157
A Really Crummy Analogy .....	158
Two Important Points on Backtracking .....	159
Saved States .....	159
Backtracking and Greediness .....	162
More About Greediness and Backtracking .....	163
Problems of Greediness .....	164
Multi-Character “Quotes” .....	165
Using Lazy Quantifiers .....	166
Greediness and Laziness Always Favor a Match .....	167
The Essence of Greediness, Laziness, and Backtracking .....	168
Possessive Quantifiers and Atomic Grouping .....	169
Possessive Quantifiers, <code>?</code> , <code>+</code> , <code>*</code> , <code>++</code> , and <code>{m,n}+</code> .....	172
The Backtracking of Lookaround .....	173
Is Alternation Greedy? .....	174
Taking Advantage of Ordered Alternation .....	175

NFA, DFA, and POSIX .....	177
“The Longest-Leftmost” .....	177
POSIX and the Longest-Leftmost Rule .....	178
Speed and Efficiency .....	179
Summary: NFA and DFA in Comparison .....	180
Summary .....	183
<b>5: Practical Regex Techniques .....</b>	<b>185</b>
Regex Balancing Act .....	186
A Few Short Examples .....	186
Continuing with Continuation Lines .....	186
Matching an IP Address .....	187
Working with Filenames .....	190
Matching Balanced Sets of Parentheses .....	193
Watching Out for Unwanted Matches .....	194
Matching Delimited Text .....	196
Knowing Your Data and Making Assumptions .....	198
Stripping Leading and Trailing Whitespace .....	199
HTML-Related Examples .....	200
Matching an HTML Tag .....	200
Matching an HTML Link .....	201
Examining an HTTP URL .....	203
Validating a Hostname .....	203
Plucking Out a URL in the Real World .....	205
Extended Examples .....	208
Keeping in Sync with Your Data .....	208
Parsing CSV Files .....	212
<b>6: Crafting an Efficient Expression .....</b>	<b>221</b>
A Sobering Example .....	222
A Simple Change—Placing Your Best Foot Forward .....	223
Efficiency Verses Correctness .....	223
Advancing Further—Localizing the Greediness .....	225
Reality Check .....	226
A Global View of Backtracking .....	228
More Work for a POSIX NFA .....	229
Work Required During a Non-Match .....	230
Being More Specific .....	231
Alternation Can Be Expensive .....	231

Benchmarking .....	232
Know What You're Measuring .....	234
Benchmarking with Java .....	234
Benchmarking with VB.NET .....	236
Benchmarking with Python .....	237
Benchmarking with Ruby .....	238
Benchmarking with Tcl .....	239
Common Optimizations .....	239
No Free Lunch .....	240
Everyone's Lunch is Different .....	240
The Mechanics of Regex Application .....	241
Pre-Application Optimizations .....	242
Optimizations with the Transmission .....	245
Optimizations of the Regex Itself .....	247
Techniques for Faster Expressions .....	252
Common Sense Techniques .....	254
Expose Literal Text .....	255
Expose Anchors .....	255
Lazy Versus Greedy: Be Specific .....	256
Split Into Multiple Regular Expressions .....	257
Mimic Initial-Character Discrimination .....	258
Use Atomic Grouping and Possessive Quantifiers .....	259
Lead the Engine to a Match .....	260
Unrolling the Loop .....	261
Method 1: Building a Regex From Past Experiences .....	262
The Real "Unrolling-the-Loop" Pattern .....	263
Method 2: A Top-Down View .....	266
Method 3: An Internet Hostname .....	267
Observations .....	268
Using Atomic Grouping and Possessive Quantifiers .....	268
Short Unrolling Examples .....	270
Unrolling C Comments .....	272
The Freeflowing Regex .....	277
A Helping Hand to Guide the Match .....	277
A Well-Guided Regex is a Fast Regex .....	279
Wrapup .....	280
In Summary: Think! .....	281



<b>7: Perl</b> .....	<b>283</b>
Regular Expressions as a Language Component .....	285
Perl's Greatest Strength .....	286
Perl's Greatest Weakness .....	286
Perl's Regex Flavor .....	286
Regex Operands and Regex Literals .....	288
How Regex Literals Are Parsed .....	292
Regex Modifiers .....	292
Regex-Related Perlisms .....	293
Expression Context .....	294
Dynamic Scope and Regex Match Effects .....	295
Special Variables Modified by a Match .....	299
The <code>qr/-/</code> Operator and Regex Objects .....	303
Building and Using Regex Objects .....	303
Viewing Regex Objects .....	305
Using Regex Objects for Efficiency .....	306
The Match Operator .....	306
Match's Regex Operand .....	307
Specifying the Match Target Operand .....	308
Different Uses of the Match Operator .....	309
Iterative Matching: Scalar Context, with <code>/g</code> .....	312
The Match Operator's Environmental Relations .....	316
The Substitution Operator .....	318
The Replacement Operand .....	319
The <code>/e</code> Modifier .....	319
Context and Return Value .....	321
The Split Operator .....	321
Basic Split .....	322
Returning Empty Elements .....	324
Split's Special Regex Operands .....	325
Split's Match Operand with Capturing Parentheses .....	326
Fun with Perl Enhancements .....	326
Using a Dynamic Regex to Match Nested Pairs .....	328
Using the Embedded-Code Construct .....	331
Using <code>local</code> in an Embedded-Code Construct .....	335
A Warning About Embedded Code and <code>my</code> Variables .....	338
Matching Nested Constructs with Embedded Code .....	340
Overloading Regex Literals .....	341
Problems with Regex-Literal Overloading .....	344