Table of Contents

About the Author XVII
About the Technical Reviewer XIX
Preface XXI
Who This Book Is For XXI
Who This Book Is (Probably) Not For XXII
How to Read This Book XXII
What's In and What Not XXIII
The HyperSpec XXIV
Which Implementation XXIV
Source Code XXV
The Index XXV
Typographical Conventions XXV
Acknowledgements XXVI
1. Symbols and Packages 1
1-1. Understanding the Role of Packages and the Symbol Nomenclature . 1
1-2. Making Unique Symbols 5
1-3. Making Symbols Inaccessible 9
How Can We Fix This? 11
1-4. Avoiding Name Conflicts 12
When Name Conflicts Do *Not* Occur 15
1-5. Using Symbols As Stand-Ins for Arbitrary Forms 15
1-6. Searching for Symbols by Name 17
1-7. Iterating Through All Symbols of a Package 19
What To Do If You Don't Like LOOP 20
1-8. Understanding COMMON LISP's Case (In)Sensitivity 21
Style Hint: Don't Use CamelCase! 25
1-9. Using Symbols As String Substitutes 26
So, What Should You Use? 27
1-10. "Overloading" of Standard COMMON LISP Operators 28
2. Conses, Lists, and Trees 31
2-1. Understanding Conses 31
List Access 34
Testing Whether Something Is a Cons or a List 35
2-2. Creating Lists 37
Converting Vectors to Lists 38
2-3. Transposing a Matrix 39

2-4. Using List Interpolation 40
2-5. Adding Objects to the End of a List 42
The Tail Wagging the List 44
2-6. "Splicing" into a List 45
2-7. Detecting Shared Structure in Lists 49
Isolating the Non-Shared Part 51
2-8. Working with Trees 51
More Complicated Trees 54
COMMON LISP's Standard Tree Functions 55
2-9. Working with Stacks 55
2-10. Implementing a Queue 56
2-11. Destructuring and Pattern Matching 58
3. Strings and Characters 61
3-1. Getting the ASCII Code of a Character 61
The Other Way Around 63
The Limit 63
3-2. Naming Characters 63
3-3. Using Different Character Encodings 65
3-4. Comparing Strings or Characters 67
Internationalization 70
3-5. Escaping Characters in String Literals and Variable Interpolation . . 71
Is It Still a Literal? 72
3-6. Controlling Case 72
What About Unicode? 74
3-7. Accessing or Modifying a Substring 75
3-8. Finding a Character or a Substring Within a String 77
3-9. Trimming Strings 77
3-10. Processing a String One Character at a Time 79
3-11. Joining Strings 81
3-12. Reading CSV Data 83
4. Numbers and Math 87
4-1. Using Arbitrarily Large Integers 87
4-2. Understanding Fixnums 89
4-3. Performing Modular Arithmetic 92
Efficiency Considerations 93
4-4. Switching Bases 94
4-5. Performing Exact Arithmetic with Rational Numbers 96
Various Ways of Converting Numbers to Integers 98
How Not to Use `FLOOR` and Friends 99
Converting Floating-Point Numbers to Rationals and Vice Versa 99
Mixing Rationals and Floats 101
4-6. Controlling the Default Float Format 102
4-7. Employing Arbitrary Precision Floats 104
4-8. Working with Complex Numbers 106

4-9. Parsing Numbers 109
4-10. Testing Whether Two Numbers Are Equal 111
Don't Ever Use EQ with Numbers! 112
4-11. Computing Angles Correctly 113
4-12. Calculating Exact Square Roots 115
5. Arrays and Vectors 117
5-1. Working with Multiple Dimensions 117
5-2. Understanding Vectors and Simple Arrays 119
5-3. Obtaining the Size of an Array 120
5-4. Providing Initial Contents 121
A Warning About Identical Objects 122
5-5. Treating Arrays As Vectors 123
5-6. Making the Length of Vectors Flexible 125
5-7. Adjusting Arrays 127
5-8. Using an Array As a "Window" into Another Array 129
5-9. Restricting the Element Type of an Array 131
Upgrading Element Types 133
5-10. Copying an Array 134
A Warning About Object Identity 135
6. Hash Tables, Maps, and Sets 137
6-1. Understanding the Basics of Hash Tables 137
Why Does GETHASH Return Two Values? 139
How Many Entries Does the Hash Table Have? 140
6-2. Providing Default Values For Hash Table Lookups 140
6-3. Removing Hash Table Entries 142
6-4. Iterating Through a Hash Table 143
Don't Rely on Any Order! 146
Don't Modify While You're Iterating! 147
Can't This Be More Concise, Please? 147
6-5. Understanding Hash Table Tests and Defining Your Own 148
What Is SXHASH For? 152
6-6. Controlling Hash Table Growth 152
6-7. Getting Rid of Hash Table Entries Automatically 155
6-8. Representing Maps As Association Lists 158
Combining Lookup and Manipulation 161
Why Would Anybody Prefer Alists over Hash Tables? 162
6-9. Representing Maps As Property Lists 163
When to Prefer Plists over Alists 165
The Plist of a Symbol 166
6-10. Working with Sets 166
Representing Sets As Hash Tables 169
Representing Sets As Bit Patterns 169
7. Sequences and Iteration 171
7-1. Filtering a Sequence 171

7-2. Searching a Sequence 172
7-3. Sorting and Merging Sequences 175
7-4. Mixing Different Sequence Types 177
7-5. Re-Using a Part of a Sequence 177
7-6. Repeating Some Values Cyclically 179
Alternatives 181
7-7. Counting Down 182
7-8. Iterating over "Chunks" of a List 184
7-9. Closing over Iteration Variables 186
7-10. "Extending" Short Sequences in Iterations 187
7-11. Breaking out of LOOP 188
7-12. Making Sense of the MAP... Zoo 191
The Sequence Variants 194
7-13. Defining Your Own Sequence Types 194
7-14. Iterating with ITERATE 196
7-15. Iterating with SERIES 200
What the Example Does 201
8. The Lisp Reader 203
8-1. Employing the Lisp Reader for Your Own Code 203
Why READ Is Potentially Dangerous 205
What READ Doesn't Do 205
The Optional Arguments to READ 206
Go Wild! 206
8-2. Troubleshooting Literal Object Notation 206
This Also Applies to Strings! 208
8-3. Evaluating Forms at Read Time 208
What to Look Out For 210
Alternatives 210
8-4. Embedding Literal Arrays into Your Code 211
The Usual Warning 212
8-5. Understanding the Different Ways to Refer to a Function 213
8-6. Repeating Something You Already Typed 214
They Don't Only *Look* Identical, They *Are* Identical! 216
8-7. Safely Experimenting with Readtables 216
Temporarily Switching to Standard IO Syntax 218
8-8. Changing the Syntax Type of a Character 219
The Six Syntax Types 220
How to Actually Change the Syntax Type 222
Some Things Never Change 222
8-9. Creating Your Own Reader Macros 223
What Reader Macro Functions Do 224
8-10. Working with Dispatching Macro Characters 226
8-11. Preserving Whitespace 228

9. Printing 231
9-1. Using the Printing Primitives 231
Printing Objects So That They Can Be Read Back in Again . . . 235
Shortcuts 236
9-2. Printing to and *into* Strings 237
9-3. Printing NIL As a List 239
9-4. Extending FORMAT Control Strings Over More Than One Line 240
9-5. Using Functions As FORMAT Controls 241
9-6. Creating Your Own FORMAT Directives 243
9-7. Recursive Processing of FORMAT Controls 245
9-8. Controlling How Your Own Objects Are Printed 247
9-9. Controlling the Pretty Printer 249
9-10. Printing Long Lists 253
9-11. Pretty-Printing Compound Objects 257
Using the Pretty Printer from FORMAT 260
9-12. Modifying the Pretty Printer 262
10. Evaluation, Compilation, Control Flow 265
10-1. Comparing Arbitrary Lisp Objects 265
Comparing State 266
Constants 269
10-2. Using Constant Variables as Keys in CASE Macros 269
10-3. Using Arbitrary Variable Names for Keyword Parameters 271
Keyword Names Don't Have to Be Keywords 272
Keyword Dames Don't Have to Be Constant 273
10-4. Creating "Static Local Variables," Like in C 273
10-5. "Preponing" the Computation of Values 275
10-6. Modifying the Behavior of Functions You Don't Have the Source Of 278
10-7. Swapping the Values of Variables (or Places) 280
10-8. Creating Your Own Update Forms for "Places" 283
Using DEFSETF 285
Using DEFINE-SETF-EXPANDER 286
So, Which One Do I Use? 291
Using DEFINE-MODIFY-MACRO 291
Multiple-Valued Places 293
10-9. Working with Environments 294
10-10. Commenting Out Parts of Your Code 299
Some Notes About ; and #| 302
How ;, #|, and Others Are Implemented 302
11. Concurrency 303
11-1. Managing Lisp Processes 304
Escape Hatches 307
Threads Are Expensive 308
11-2. Accessing Shared Resources Concurrently 308
Locks 312

Atomic Operations 313
More Problems 314
11-3. Using Special Variables in Concurrent Programs 317
Per-Thread Initial Bindings 319
Variables That Are Always Global 319
11-4. Communicating with Other Threads 320
Alternatives 322
11-5. Parallelizing Algorithms Without Threads and Locks 322
What the Example Does 325
Fine-Tuning 326
Ptrees 326
Alternatives 330
11-6. Determining the Number of Cores 330
12. Error Handling and Avoidance 333
12-1. Checking Types at Run Time 333
Alternatives 335
12-2. Adding Assertions to Your Code 336
Disabling Assertions in "Production Code" 338
12-3. Defining Your Own Conditions 338
How Conditions Are Printed 340
12-4. Signaling a Condition 341
Condition Designators 343
12-5. Handling Conditions 344
Ignoring Errors 349
12-6. Providing and Using Restarts 350
Visible Restarts 354
Predefined Restarts 355
12-7. Getting Rid of Warning Messages 356
12-8. Protecting Code from Non-Local Exits 357
"WITH-" Macros 360
13. Objects, Classes, Types 361
13-1. Defining Types 361
Compound Type Specifiers 363
Derived Types 365
13-2. Using Classes As Types 366
13-3. Writing Methods for Built-In Classes 367
13-4. Providing Constructors for Your Classes 369
13-5. Marking Slots As "Private" 372
13-6. Changing the Argument Precedence Order 374
13-7. Automatically Initializing Slots on First Usage 376
13-8. Changing and Redefining Classes on the Fly 377
Objects Changing Their Class 380
Redefining Classes 381
13-9. Making Your Objects Externalizable 383

13-10. Using and Defining Non-Standard Method Combinations 385
Rolling Your Own 388
Arbitrarily Complex Method Combinations 389
13-11. Extending and Modifying CLOS 391
What the example does 393
14. I/O: Streams and Files 397
14-1. Redirecting Streams 397
Other Ways to Do It 398
Synonym Streams 399
14-2. Flushing an Output Stream 400
14-3. Determining the Size of a File 401
14-4. Reading a Whole File at Once 402
Alternatives 404
14-5. Sending Data to Two Streams in Parallel 404
Synonym Streams 406
14-6. Sending Data to "/dev/null" 407
14-7. Pretending a String Is a Stream 408
More Details 410
14-8. Concatenating Streams 411
14-9. Processing Text Files Line by Line 412
What Happens at the End of a Line? 413
What Happens at the End of the File? 414
14-10. Working with Binary Data 415
Reading or Writing Several Bytes at Once 416
You Might Get Bigger Chunks Than You Asked For 417
14-11. Reading "Foreign" Binary Data 418
Floating-Point Values 421
14-12. Using Random Access I/O 422
Different Characters May Have Different Lengths 424
14-13. Serializing Lisp Objects 425
Shared Structure 427
Is It Readable? 428
Can This Be Done Faster, Please? 429
What About JSON or Other Formats? 430
14-14. Customizing Stream Behavior 430
15. Pathnames, Files, Directories 435
15-1. Getting and Setting the Current Directory 435
Shortcuts and Deviations 437
15-2. Testing Whether a File Exists 437
What About Directories? 439
15-3. Creating a Directory 439
Implementation-Specific Alternatives 441
What Might Go Wrong 441
15-4. Finding Files Matching a Pattern 442

15-5. Splitting a Filename into its Component Parts . . . 445
15-6. Renaming a File . . . 447
Implementation-Specific Alternatives . . . 449
Don't Expect "Move" Behavior! . . . 449
15-7. Deleting a File . . . 450
What Does "Success" Mean Anyway? . . . 451
15-8. Deleting a Directory . . . 452
15-9. Copying a File . . . 453
15-10. Processing the Contents of a Directory Recursively . . . 454
The CL-FAD Library . . . 455
15-11. Getting the Pathname a Stream Is Associated With . . . 456
15-12. Dealing with Symbolic Links . . . 457
What If I *Want* the Symlinks? . . . 459
15-13. Navigating a Directory Tree . . . 459
15-14. Figuring Out (Source) File Locations Programmatically . . . 461
15-15. Understanding Logical Pathnames . . . 463
What Exactly Are Logical Pathnames? . . . 467
So, Maybe Logical Pathnames Aren't Totally Useless... . . . 467
16. Developing and Debugging . . . 469
16-1. Embracing Lisp's Image-Based Development Style . . . 469
The Role of the Source Code . . . 471
16-2. Deciding Which IDE to Use . . . 471
A Brief History of Emacs (As Seen from Lisp) . . . 474
Alternatives (?) . . . 475
16-3. Debugging with the Debugger . . . 475
Entering the Debugger Intentionally . . . 478
Without SLIME . . . 479
Logging Backtraces . . . 480
16-4. Tracing Functions . . . 481
Graphical Tracing . . . 483
16-5. Stepping Through Your Code . . . 484
16-6. Acquiring Information About Functions, Macros, and Variables . . . 486
Accessing the HyperSpec . . . 488
Cross-Reference Information . . . 489
16-7. Inspecting and Modifying (Compound) Objects . . . 489
The SLIME Inspector . . . 492
16-8. Browsing Your Lisp Image . . . 492
Alternatives . . . 493
16-9. "Undoing" Definitions . . . 494
16-10. Distinguishing Your IDE's Streams . . . 497
16-11. Utilizing the REPL's Memory . . . 499
IDE History Features . . . 500
16-12. Recording Your Work . . . 501

17. Optimization 503
17-1. Understanding the Importance of the Right Algorithms 504
17-2. Deciding If and Where to Optimize 505
Instrumentation 508
Statistical Profiling 510
Some Things Cannot Be Profiled 512
Where to Go From Here 512
A Warning About Empirical Data 513
What Does TIME Do? 513
CPUs Can Deceive You 513
17-3. Asking the Compiler to Optimize 515
Implementation-Defined Optimize Qualities 517
Can I Have This Global, Please? 517
Compilers Aren't Wizards 518
17-4. Obtaining Optimization Hints from the Compiler 518
17-5. Helping the Compiler by Providing Type Information 522
Generic Operations 525
Boxing 525
How to Declare Types 526
The Scope of Type Declarations 527
Declaring the Return Type of Forms 528
Type Inference 528
Pitfalls 529
17-6. Reducing "Consing" 529
"Consing" and the Heap 532
Reusing Data Structures 533
Destructive Functions 534
"Hidden" Consing 535
"Tuning" the Garbage Collector 536
17-7. Using the Stack Instead of the Heap 536
Multiple Values 540
17-8. Optimizing Recursive Functions 542
17-9. Helping the Compiler with Alternative Implementation Strategies . 544
17-10. Avoiding Repeated Computations 547
17-11. Avoiding Function Calls 549
Alternatives That Aren't Really Alternatives 553
The NOTINLINE Declaration and Compiler Macros 553
17-12. Utilizing the Disassembler 553
17-13. Switching to Machine Code 556
Inline Assembly Code 557
17-14. Optimizing Array Access 558
17-15. Comparing Different Implementations 560

18. Libraries . . . 563
18-1. Organizing Your Code . . . 563
Components . . . 564
Dependencies . . . 565
How Does ASDF Find System Definitions? . . . 567
Additional Information . . . 568
What About the Names? . . . 568
Advanced Usage of ASDF . . . 568
18-2. Employing Open Source Libraries . . . 569
Installing QUICKLISP . . . 570
Using QUICKLISP for Your Own Code . . . 571
18-3. Creating a Project's Skeleton Automatically . . . 571
18-4. Avoiding Wheel-Reinvention . . . 572
18-5. Using Libraries to Write Portable Code . . . 574
18-6. Utilizing Regular Expressions . . . 576
Regex Syntax . . . 577
Scanners . . . 578
Convenience Features . . . 579
Modifying and Dissecting Strings . . . 581
More Information . . . 581
18-7. Obtaining Data via HTTP . . . 581
Parsing HTML . . . 583
18-8. Creating Dynamic Web Sites with Lisp . . . 584
Generating HTML . . . 588
Web Frameworks . . . 589
19. Interfacing with Other Languages . . . 591
19-1. Calling C Functions from Lisp . . . 592
How the FFI Finds and Loads Shared Libraries . . . 594
How the FFI Calls C Functions . . . 596
How the FFI Converts Between C and Lisp Types . . . 598
The "`stdcall` Problem" . . . 599
19-2. Working with C Pointers . . . 600
Typed Pointers . . . 602
19-3. Accessing and Generating C Arrays . . . 604
Giving C Access to Lisp Arrays . . . 606
19-4. Handling C Structs and Unions . . . 607
Passing Structs by Value . . . 611
19-5. Converting Between Lisp and C Strings . . . 611
19-6. Calling Lisp Functions from C . . . 614
19-7. Generating FFI Code Automatically . . . 615
19-8. Embedding C in COMMON LISP . . . 617
19-9. Calling C++ from Lisp . . . 618
Automating the Process . . . 620
ECL and CLASP . . . 622

19-10. Using JAVA from Lisp 622
Alternatives 625
19-11. Reading and Writing JSON 628
19-12. Reading and Writing XML 631
19-13. Using PROLOG from COMMON LISP 634
20. Graphical User Interfaces 637
20-1. Using a Web Browser as the GUI for Your Lisp Program 638
What the Example Does 641
20-2. Building Applications with the *"Lisp Toolkit"* 642
What the Example Does 645
20-3. Creating COMMON LISP GUIs Through JAVA 646
What the Example Does 648
A Better Example 648
20-4. Using CAPI to Build Graphical User Interfaces 652
What the Example Does 655
20-5. Using Lisp on Mobile Devices 657
Alternatives 660
21. Persistence 661
21-1. Serializing Your Data 662
21-2. Accessing Relational Databases 664
21-3. Keeping Your Database in RAM 668
21-4. Using a Lisp Object Database 672
22. The World Outside 677
22-1. Accessing Environment Variables 677
The Windows Registry and Application-Specific Settings 679
22-2. Accessing the Command-Line Arguments 679
22-3. Querying Your Lisp for Information About Its Environment 680
22-4. Delivering Stand-Alone Executables 682
Delivering Programs with the Commercial Lisps 685
22-5. Customizing Your Lisp 685
Saving an Image 687
22-6. Running External Programs 688
22-7. Embedding Lisp 692
Creating Shared Libraries with LISPWORKS 694
Embedding ABCL in a JAVA Program 695
22-8. Measuring Time 696
22-9. Working with Dates and Times 698
The LOCAL-TIME Library 701
22-10. Working with the Garbage Collector 703
Finalizers 705
Index 709