97

Collective Wisdom from the Experts

97 Things Every Programmer Should Know

O'REILLY®

Edited by Kevlin Henney

Table of Contents

Introduction	0
Act with Prudence	1
Apply Functional Programming Principles	2
Ask "What Would the User Do?" (You Are not the User)	3
Automate Your Coding Standard	4
Beauty Is in Simplicity	5
Before You Refactor	6
Beware the Share	7
The Boy Scout Rule	8
Check Your Code First before Looking to Blame Others	9
Choose Your Tools with Care	10
Code in the Language of the Domain	11
Code Is Design	12
Code Layout Matters	13
Code Reviews	14
Coding with Reason	15
A Comment on Comments	16
Comment Only What the Code Cannot Say	17
Continuous Learning	18
Convenience Is not an -ility	19
Deploy Early and Often	20
Distinguish Business Exceptions from Technical	21
Do Lots of Deliberate Practice	22
Domain-Specific Languages	23
Don't Be Afraid to Break Things	24
Don't Be Cute with Your Test Data	25
Don't Ignore that Error!	26
Don't Just Learn the Language, Understand its Culture	27
Don't Nail Your Program into the Upright Position	28
Don't Rely on "Magic Happens Here"	29

Don't Repeat Yourself 30
Don't Touch that Code! 31
Encapsulate Behavior, not Just State 32
Floating-point Numbers Aren't Real 33
Fulfill Your Ambitions with Open Source 34
The Golden Rule of API Design 35
The Guru Myth 36
Hard Work Does not Pay Off 37
How to Use a Bug Tracker 38
Improve Code by Removing It 39
Install Me 40
Inter-Process Communication Affects Application Response Time 41
Keep the Build Clean 42
Know How to Use Command-line Tools 43
Know Well More than Two Programming Languages 44
Know Your IDE 45
Know Your Limits 46
Know Your Next Commit 47
Large Interconnected Data Belongs to a Database 48
Learn Foreign Languages 49
Learn to Estimate 50
Learn to Say "Hello, World" 51
Let Your Project Speak for Itself 52
The Linker Is not a Magical Program 53
The Longevity of Interim Solutions 54
Make Interfaces Easy to Use Correctly and Hard to Use Incorrectly 55
Make the Invisible More Visible 56
Message Passing Leads to Better Scalability in Parallel Systems 57
A Message to the Future 58
Missing Opportunities for Polymorphism 59
News of the Weird: Testers Are Your Friends 60
One Binary 61
Only the Code Tells the Truth 62
Own (and Refactor) the Build 63

Pair Program and Feel the Flow	64
Prefer Domain-Specific Types to Primitive Types	65
Prevent Errors	66
The Professional Programmer	67
Put Everything Under Version Control	68
Put the Mouse Down and Step Away from the Keyboard	69
Read Code	70
Read the Humanities	71
Reinvent the Wheel Often	72
Resist the Temptation of the Singleton Pattern	73
The Road to Performance Is Littered with Dirty Code Bombs	74
Simplicity Comes from Reduction	75
The Single Responsibility Principle	76
Start from Yes	77
Step Back and Automate, Automate, Automate	78
Take Advantage of Code Analysis Tools	79
Test for Required Behavior, not Incidental Behavior	80
Test Precisely and Concretely	81
Test While You Sleep (and over Weekends)	82
Testing Is the Engineering Rigor of Software Development	83
Thinking in States	84
Two Heads Are Often Better than One	85
Two Wrongs Can Make a Right (and Are Difficult to Fix)	86
Ubuntu Coding for Your Friends	87
The Unix Tools Are Your Friends	88
Use the Right Algorithm and Data Structure	89
Verbose Logging Will Disturb Your Sleep	90
WET Dilutes Performance Bottlenecks	91
When Programmers and Testers Collaborate	92
Write Code as If You Had to Support It for the Rest of Your Life	93
Write Small Functions Using Examples	94
Write Tests for People	95
You Gotta Care about the Code	96

Your Customers Do not Mean What They Say 97

Click here to download full PDF material