

# PragPub

The First Iteration

## IN THIS ISSUE

- \* The Layoffs Are Coming!
- \* Why Clojure?
- \* When Things Go Wrong
- \* Pragmatic Publishing

## Contents

### FEATURES



#### The Layoffs Are Coming! ..... 9

by Andy Lester

Andy wrote the book on finding tech jobs (*Land the Tech Job You Love*). Here he tells you how to keep the job you have.



#### Why Clojure? ..... 14

interviewed by Michael Swaine

Rich Hickey created Clojure, a modern dialect of Lisp that targets the JVM. In this PragPub interview, Rich explains what that means to you.



#### When Things Go Wrong ..... 17

by Stuart Halloway

As Stuart demonstrates in this deep immersion in Clojure coding, one of Clojure's strengths is how it shines when things go wrong.



#### Pragmatic Publishing ..... 22

interviewed by Michael Swaine

Dave Thomas talks about the decisions behind Pragmatic Bookshelf's comprehensive ebook program, and how book publishing is undergoing a transformation, from electronic versus paper to direct sales versus the channel.

**Welcome to PragPub** ..... 1

by Andy Hunt and Dave Thomas

Time to start a real dialogue among highly talented, skilled professionals who care about their craft.

**Up Front** ..... 2

by Michael Swaine

Michael says hello, world.

**Choice Bits** ..... 3

Tweets, posts and chatter on job searches and revolving doors, book reviewers and bonus tracks, Google Wave and shampoo bottles, desires, donkeys, and daydreams.

**How Do We...?** ..... 5

People often ask how we do what we do. This monthly series explains...

**Swaine's World** ..... 6

by Michael Swaine

In which Mike loses his job and offers you career advice.

**The Quiz** ..... 26

A monthly diversion at least peripherally related to programming.

**Shady Illuminations** ..... 27

by John Shade

Columnist John Shade casts a jaundiced eye on Microsoft's latest attempt to out-google Google.

**Calendar** ..... 32

Author sightings and other notable events.

---

Except where otherwise indicated, entire contents copyright © 2009 The Pragmatic Programmers.

Feel free to distribute this magazine (in whole, and for free) to anyone you want. However, you may not sell this magazine or its content, nor extract and use more than a paragraph of content in some other publication without our permission.

Published monthly in PDF, mobi, and epub formats by The Pragmatic Programmers, LLC, Dallas, TX, and Raleigh, NC. E-Mail [support@pragprog.com](mailto:support@pragprog.com), phone +1-800-699-7764. The editor is Michael Swaine ([michael@pragprog.com](mailto:michael@pragprog.com)). Visit us at <http://pragprog.com> for the lowdown on our books, screencasts, training, forums, and more.

# Welcome to *PragPub*

---

by Andy Hunt and Dave Thomas



Greetings! Welcome to Iteration One of *PragPub*, our monthly excuse to hang out at the virtual water cooler with you.

*PragPub* is a look into the pragmatic world: it's a peek at what we're excited about, what our authors are excited about, and what we hope you can get excited about. It's a place where together we can share our obsessions, er, "passions."

We love learning new things. We love playing with technology, and seeing what cool things we can use it for. But it's not always the shiny new things that are important: it pays to remember how we got here in the first place. To reconsider Fred Brooks, Kristen Nygaard, or Kent Beck's thoughts under a modern microscope. To look at other industries, other professions, and learn from their successes and failures as well. To look at the full range of our lives, not just those intense hours spent on the job itself.

That's what we want to accomplish with *PragPub*. The articles in this first iteration are available in PDF, mobi and epub formats, for reading on your computer, iPhone, Kindle, Sony Reader, or other device. In future iterations, you may have other exciting options. We're counting on adapting and evolving *PragPub* to best suit your needs, according to your feedback.

Magazines have historically been a monologue. We prefer a dialogue—not a screaming mob, as you might find elsewhere on the net, but a real dialogue among highly talented, skilled professionals who care about their craft. We care, and we know you do too. After all, we can't do it alone.

So please extend a warm welcome to Michael Swaine, author and former editor-in-chief and editor-at-large for many years at *Dr. Dobb's Journal*. Michael brings his considerable experience and industry knowledge to these pages, and we're proud to have him on board, to help shepherd the dialog.

Welcome to your magazine. Welcome to *PragPub*.

-Dave & Andy, Publishers.

# Up Front

## Iteration One

by Michael Swaine

What do you think of our first issue? Help us make *PragPub* the kind of publication you'd like to read. We're eager to hear from you.<sup>[U1]</sup>



Welcome to the premier issue of *PragPub*. I'm your host, Michael Swaine.

As you may have noticed, we're calling these first few issues the First Iteration. We're firm believers in the power of agile development. Dave Thomas and Andy Hunt have built an agile publishing company, and this will be an agile magazine. That means that this is truly just the beginning—our first “Hello World.” Our aim is to create a real 21st century version of that oh-so-last-century medium, the magazine. Since nobody, least of all us, knows what that actually means, we'll be iterating (a.k.a. “making it up”) as we go along.

One thing we know is that it will emphasize the *pragmatic* in programming and life. Pragmatic Programmers get the job done, and do it well, Andy and Dave said in [The Pragmatic Programmer](#) <sup>[U2]</sup> back in 2001. That's the book that launched The Pragmatic Bookshelf line of books. As Dave and Andy explained then, *pragmatic* derives from a Greek word meaning “to do.”

This monthly publication is all about doing. You'll find articles here on programming in Clojure, how to hang on to your tech job and how we do the things we do. Upcoming articles will feature in-the-trenches Android and iPhone development and refactoring your career. Some articles will be written by our Pragmatic Bookshelf authors, but they won't be mere book excerpts, but new articles written expressly for *PragPub*—and all free.

But we'll also live-link you into our Forum at The Pragmatic Programmers site, as well as to Andy's and Dave's blogs, where you can see what's on their minds right now. Because we want your feedback, we have a suggestion box. It's called “I'd like a book that covers...;” But we want your ideas for this magazine, too. [Email us](#) <sup>[U3]</sup> or join the discussion in our Forum.

There's a vibrant community growing among our readers and authors, and we want to make it easy for you to join in.

And because there's more to the coding life than coding, we'll feature fresh content from our Pragmatic Life series authors and editors every month.

So enjoy this first monthly issue of *PragPub*. And did we mention it's free?

-Michael Swaine, editor

### External resources referenced in this article:

<sup>[U1]</sup> <mailto:michael@pragprog.com?subject=PragPub>

<sup>[U2]</sup> <http://www.pragprog.com/titles/tpp>

<sup>[U3]</sup> <mailto:michael@pragprog.com?subject=PragPub>

# Choice Bits

## Overheard on the Intertubes

Wading into the chatterstream in hip boots, we wield the Axiom of Choice with gleeful abandon.



## What's Hot

Top-Ten lists are passé—ours goes to 11. These are the top titles that folks are interested in currently, along with their rank from last month. This is based solely on direct sales from our online store.

1	4	iPhone SDK Development
2	NEW	Language Design Patterns
3	1	The RSpec Book
4	2	Agile Web Development with Rails, Third Edition
5	NEW	Agile Coaching
6	3	Programming Ruby 1.9
7	6	Programming Clojure
8	7	Writing Your First iPhone Application
9	10	Coding in Objective-C 2.0
10	NEW	Core Data
11	NEW	Hello, Android

## And Here's What Folks Are Saying...

- *This is journalism as beta.... [E]very time Google releases a beta, it is saying that the product is incomplete and imperfect. That is inevitably a call to collaborate. It is... a statement of humanity and humility: We're not perfect.* — Jeff Jarvis, [BuzzMachine](#) <sup>[U1]</sup>
- *[W]hat does the average geek do when he wants to find a job? He hits the job boards, doing simple keyword searching on CareerBuilder or Monster or one of the niche sites. But the hard truth is that almost four times as many jobs (29.0%) are filled from personal networking or hard research finding a company that's a fit. It just doesn't make sense to turn to the job boards as your primary source of finding a job.* — Andy Lester, [from his blog, The Working Geek](#) <sup>[U2]</sup>
- *Functional MRI scans showed that the area of the brain responsible for complex problem solving does not go dormant when you daydream, but in fact becomes very active.* — Andy Hunt [in his blog](#) <sup>[U3]</sup>
- *[A] revolving door gave me ideas about software process. 3 variables interact: latency, throughput, and variance.* — @KentBeck
- *Their ads don't say "Urban Spoon runs on your iPhone. It can find you nearby restaurants." They would say something like, "Say you're hungry and want to know what's nearby that you might enjoy." Their ad starts with your desire and gives you a solution....* — Daniel Steinberg [in his PragLife blog](#) <sup>[U4]</sup>
- *Associate to IT guy: So, my touchscreen hasn't been working, so I hit it harder and now there's a crack in the screen. IT guy: Your computer isn't touchscreen.* — [from Overheardintheoffice.com](#) <sup>[U5]</sup>

- *I would really like to see a book on Pragmatic Django Devel. I know there are few books already out, but I love Pragmatic books—they are always a good read—and I would definitely buy one.* — Tanja Pislár
- *Seconded.* — Khurrum Maqbul from Pragmatic discussion forum “I’d like a book that covers...” [U6]
- *Perhaps getting a jump start on “Reia” . . . Ruby Syntax, Erlang VM, Concurrent goodness. Yeah its an Alpha language, and still needs work. But off to a promising start.* — Brad Hutchins from Pragmatic discussion forum “I’d like a book that covers...” [U7]
- *[I]f the shampoo bottle in the shower is turned around half the time, why isn’t the logo on both sides?* — @KentBeck
- *okay, Google Wave is a game changer.* — Brandon Zeuner via Twitter
- *Since last summer we’ve been running iPhone training courses [U8] and producing iPhone-related screencast video tutorials [U9]. Unfortunately, even in a 4-day course or a 1-hour screencast, you have to leave some stuff on the cutting room floor. Though we’re learning new iPhone development tips and tricks every day, they don’t all get brought up during a course or warrant a full-length video tutorial. But it’s good stuff we don’t want you to miss!*  
  
*So, we recently started releasing Pragmatic Studio Bonus Tracks [U10]. These are free, short screencasts focused on a single iPhone tip, trick, or technique that you can quickly apply. We plan to release a new bonus track every week or so to keep you on top of your game. Whether you’ve attended one of our courses or not, we hope you enjoy these bonus tracks all the same.*  
— Mike Clark. The Pragmatic Studio [U11] is owned and operated by Mike and Nicole Clark.
- *HR rep: Oh, yeah... I was going to do that. But then I got distracted by the festive donkeys.* — from Overheardintheoffice.com [U12]
- *I keep on buying books from the Pragmatic Programmers, and then Amazon recommends them for me, and I have to tell them I already own it.* — Andrew Grimm via Twitter
- *When the Church of the Savvy recognizes itself it’s always transfigured into the cathedral of truth.* — Jay Rosen via Twitter


#### External resources referenced in this article:

- [U1] <http://www.buzzmachine.com/2009/06/07/processjournalism/>
- [U2] <http://theworkinggeek.com/>
- [U3] <http://andy.pragprog.com/>
- [U4] [http://praglife.typepad.com/pragmatic\\_life/](http://praglife.typepad.com/pragmatic_life/)
- [U5] <http://www.overheardintheoffice.com/>
- [U6] <http://forums.pragprog.com/forums/109>
- [U7] <http://forums.pragprog.com/forums/109>
- [U8] <http://pragmaticstudio/iphone>
- [U9] <http://pragmatic.tv>
- [U10] <http://pragmaticstudio.com/screencasts>
- [U11] <http://www.pragmaticstudio.com>
- [U12] <http://www.overheardintheoffice.com/>

# How Do We...?

## How Gerbils Make Sausage

People often write in and ask us how we do things: how are the PDFs stamped? how do we ship books? what technology do we use to author books? Each month we'll answer one of these questions.



How do you add my name to the bottom of the PDFs I buy?

We refuse to hobble our electronic books with Digital Rights Management (DRM) restrictions. We feel that, just as when you buy a paper book, you should be able to use your electronic books as you see fit. You should be able to copy them to your laptop, make backups, read them on your phone or eBook reader—whatever you need. We just ask that you respect our copyright, pay for the books you use, and don't give those books to others.

To help remind readers that we're giving them responsibility for their books, we add their name to the bottom of each page of their PDFs. (We once toyed with the idea of putting their credit card number there, but prudence won out.) So how do we do that?

During the regular build process (driven by rake, of course), we produce master PDF, mobi, and epub versions of a book. These versions are checked into a subversion repository.

Over in the EC2 cloud, we have a variable number of stamping machines running. Each of these machines has a checked-out copy of the book masters.

The stamping machines use a REST interface into our order system to look for new purchases and requests for book regenerations. When a request comes along, they take the master PDF and run it through a program that adds the annotations you see (along with a few other annotations you don't see) to produce a personalized PDF. This PDF is then stored in S3. This is the PDF that you download.

We used to use some commercial software to do the stamping, but the licencing terms became ridiculous, so we've switched to using the wonderful open source [iText library](http://www.lowagie.com/iText/) <sup>[U1]</sup>. It's a Java program, so we drive it from our Ruby code using a simple `popen` call, passing stamping parameters to the stamper's standard input.

Once stamping is complete, the EC2 machine uses another REST request to tell the online store, and your download becomes available.

And how do we tell the stamping machines to update their master copies of the books when we make changes? Our book build process tells the online system, and the system tells the stampers when they next request work. They then simply issue an `svn up` and continue.

External resources referenced in this article:

<sup>[U1]</sup> <http://www.lowagie.com/iText/>



# Swaine's World

## Gerbil, Interrupted

by Michael Swaine



I was happily spinning my wheel when the call came from HR. Due to the economic downturn, the magazine I had been associated with for 25 years was ceasing print publication and all the gerbils were being set free.

I'd been running in place for so long I guess I experienced a touch of vertigo when the wheel stopped. Over the ensuing weeks, friends observed tell-tale signs: I was eating more, sleeping longer, and gnawing on the furniture. After a few weeks I did what you probably would have advised if I'd thought to ask you: I emailed Dave and Andy, we had a really nice chat, and they offered me this shiny new wheel.

Speaking of the economy, apparently there's some sort of recession going on.

Venture capital spending was off over 70 percent at the end of last year. Just since January there have been major layoffs at AMD, Amazon, AOL, Autodesk, Best Buy, Borland, Bose, Broadcom, Circuit City, Cisco, Dell, Digg, Disney, Electronic Arts, Ericsson, Expedia, Freescale, Gartner Group, Google, Hitachi, IBM, Intel, Lenovo, Logitech, Microsoft, Motorola, Nat Semi, NEC, Nokia, Oracle, PBWiki, Philips, Razorfish, SAP, Seagate, Sprint, Sun, Technorati, Teradyne, TI, Toshiba, Unisys, and Yahoo, to name a few. In the case of Microsoft and IBM, we're talking thousands of jobs.

Isn't it fortunate, then, that you're in a recession-proof industry?

At least that's what conventional wisdom says, and in fact the jobs being cut at tech firms are likely to be anything but software development jobs. If you're good, the same reliable source (conventional wisdom) says, your skills are in demand. Well, yes, but.

I mean it's great if your skills are more in demand than ever, but what if the company you work for just went belly-up? Or your client list is experiencing embarrassing shrinkage? Or your existing clients are getting stingy?

Whether or not software development is a recession-proof industry, your skills may be proof against the economic downturn. But that doesn't mean you won't suddenly find yourself having to hustle more than you're used to. Maybe you've never had to look for work. Maybe it has always come to you. And now you may have to (shudder) market yourself. I hope to convince you that tooting your own horn is not that hard, not that bad, and you're already doing it.

## The First Thing

Andy Lester identifies the first hurdle you have to get over. Don't be shy about the fact that you're looking for work. "Tell people you're available. Online, that means blogging about it and mentioning it in some online fora. It also means telling family and friends. [You don't have to say] 'Please give me a job,'

but [do] let people know. Getting the word out is the first step in working your social network for potential leads.”

But can’t you just let your work speak for you? In some ways, you can. Your work on open source projects can carry a lot of weight with savvy employers. But be a significant and visible contributor. That says you’re not just a 9-to-5er, and it speaks volumes about your accomplishments. As Andy says, “Anyone can write Struts or Nant on their résumé. Very few can write Struts committer or Nant committer.”

But you must be sociable. For some of you, that’s not a problem, but for some of us, it’s a challenge. at least you can be selective about it and expend your social energy judiciously. Professional user groups can be great for networking and meeting potential employers in a low-pressure setting. If you can speak at a tradeshow or conference, do so. It makes you an accepted expert. Got to industry afterparties. As for the after-afterparties and after-after-afterparties, you’ll have to make that call yourself.

## Work Your Social Network

As everyone knows, Facebook is now the eight largest country in the world. The recession has been very, very good to social networks. LinkedIn, which is all about professional networking, has seen a 65 percent increase in recommendations.

“Make sure you have recommendations on LinkedIn,” my friend Paul Freiburger advises. Paul has helped a lot of people refine and promote their ideas, both at McKinsey Corp. and earlier at Paul Allen’s idea incubator, Interval Research, and is now helping people refine the way they present themselves at his own company, [Shimmering Résumés](#)<sup>[U1]</sup>. “Work your LinkedIn networks. Participate in groups that share your interests. You may find people participating in these groups who are at companies that are hiring.” After passing along this advice, he reminded me to write him a LinkedIn recommendation. Nice to see the doctor taking his own medicine.

## Yes, You Need a Résumé

You need a résumé. Also a blog. And they are not the same thing. “Résumés,” Guy Kawasaki says, “are... premature and unsentimental obituaries.” And they’re treacherous: a poorly-written résumé is a strong case against hiring you. Read Andy’s book; there’s a whole chapter on résumés. Visit Freiburger’s site. Google Guy Kawasaki.

## Surviving Interviews

If you get called in for an interview, don’t freak. Be cool, be honest, and try to think like the boss. Understand the hiring process. Many interview questions are intended as opportunities for you to tell your story. Know your story. Tell it. In fact you will do well to treat every question as an opportunity to tell a story.

Marketing yourself, as Chad Fowler points out so elegantly in his book [The Passionate Programmer](#)<sup>[U2]</sup>, is not just for sluts. Selling yourself matters even

if you have a job. It matters that your bosses or clients know the value of your work. And when you're looking for work, marketing is crucial.

I'm just relaying the advice of people who know much more than I do about this stuff. After all, when I needed a job I didn't do any of this; I just emailed Dave and Andy. Do as the experts say, not as I do. But marketing yourself is just one key to thriving in a challenging economy. Next month I hope to reveal other keys. In the meantime I'd love to hear from you. What have been your experiences in the Great Recession? Are you thriving? What have you learned about thriving in challenging times? [Email me](#) <sup>[U3]</sup>. Or join the discussion in our forums.

#### About the Author

Michael Swaine is the editor of *PragPub*.

#### External resources referenced in this article:

[U1] <http://www.shimmeringresumes.com>

[U2] <http://www.pragprog.com/titles/cfcar2>

[U3] <mailto:michael@pragprog.com?subject=PragPub>

# The Layoffs Are Coming!

## How to Keep Your Tech Job

by Andy Lester

In a perfect world, your work would speak for itself, your genius would be universally recognized, and your job would never be at risk. It's not a perfect world.



The economy's in the tank. Millions of people have been laid off, and IT jobs are not exempt from the axe. Your company's undoubtedly having a sales slump, if not free-fall, and job cuts could well be coming.

Don't sweat it.

There are two types of layoffs. The huge reductions where entire departments or units are sent packing at once are beyond your control. Corporate management decides that a large part of the operation is unprofitable or can be outsourced, and out you go. This type of layoff can't be planned for. You find out one day that all twenty of you are out on the street, and there's nothing you can do to prevent it.

More likely you'll face a more selective layoff, and you won't even see it coming. Usually, a department or unit manager will be told, "You have to reduce your staff of twenty by three," and it's up to her to find the three least-profitable members in the group.

That's the layoff you can avoid, by actions that you take *now*.

## How to keep your job

Assuming you're happy with your job, your task is to keep it. I'm going to tell you how.

Your goal is to be the best programmer, the best sysadmin, the most profitable addition to the bottom line that you can. You want to be in the top tier of performers in the group, so that your boss says to herself, "There's no way I'd get rid of him."

## Treat your retention as a hire

You know that when you're working to get hired at a company, your job is to show how you will add to the company's bottom line if you are selected, often by citing examples from your past. When you're looking to keep your job, it's the same: You show the value you bring to the company, and what you're working on for the future.

How should you do this? Keep track of what it is you do every day, and how you help the company. Be consistently aware of what you're learning and how you're improving your company's bottom line. When you make things better, make sure your boss knows it.

Always look to the future. Think about how you can improve the department and the company, and talk about it. Make plans with others to make things better.

## Update your résumé

When you're tracking your value to the company, and documenting what you've done, put it in your résumé. Use your résumé as a tool to track your performance. A good résumé has work history as its headings and achievements as the supporting detail.

A good résumé tells what you did, not what you were. The "what" should have numbers associated with it. Instead of:

Yoyodyne Intl, 2006-2009

- \* Wrote application code in Perl for various projects.
- \* Worked both on teams and on solo projects.
- \* Strove to introduce new technologies to the department.

Give detail that describes the value you provided, quantifying to give the reader an idea of scope:

Yoyodyne Intl, 2006-2009

- \* Designed, coded and tested Perl applications for customer-facing web applications using Catalyst, DBIx::Class, and Moose.
- \* Worked on teams of two to eight, depending on project, as well as on solo projects.
- \* Taught myself Ruby and ran a pilot project for a simple user maintenance application using Rails. Project completion time was two weeks, instead of the expected six weeks.

Don't worry about having your résumé be too long. You can trim it later before sending it, but if you don't capture the detail up front, chances are you'll never get it back. You may also want to keep a scrap file of items that can be added to your résumé as necessary.

Don't wait until you need your résumé to spend the time to improve it. Tend to it as you go, and when it's time to harvest this critical tool for job hunting, you'll be rewarded.

## Don't keep your head down

Conventional wisdom has it that if layoffs are afoot, you want to keep your head down, not make waves, and reduce your chances of getting on someone's Bad List. This is exactly backwards. It might make sense if all workers were interchangeable, but you're not.

Keep your head up, and let others know about what improvements you've made and value you've added. It's no good to improve your backup process and reduce the number of tapes required if you keep this fact to yourself.

You have to let your boss know, but don't stop there. As you build your reputation in the organization, you'll be working with others, so let them know about your valuable work while you find ways to help them as well.

## Be valuable beyond your department

The only thing better than being a valuable asset to your boss, to your team, to your department is being valuable to those on the outside. Assuming that you're doing a good job in your department, accolades from others in the organization are music to your boss's ears.

## Don't suck up

The worst way to survive a layoff is by sucking up to the boss, or relying on a good personal relationship.

First, it won't work. The boss is not going to keep you in favor of a stronger performer just because he likes you. His job is on the line as well, and it does him no good to keep a weaker employee. Spend your time on improving yourself instead.

Second, it will backfire. Sucking up shows weakness and desperation, two traits that are never good. The boss is not stupid and will not be fooled by your sucking up. Worse, the boss may be more inclined to axe you rather than be perceived as showing favoritism.

Finally, it's degrading to you. Hold your head high and make it through tough times by your intelligence and hard work.

## Start looking for your next job

You may still lose your job. You can still minimize the impact of being laid off. Here's how.

Don't wait for the axe to fall. Start looking now for your next job. Find another job that's interesting, and assess your likelihood of landing that job. Get a feel for the market. Understand the skills wanted and opportunities offered in your job market.

You should be doing this, layoffs or not. Unless you're nearing retirement age, chances are good you're not going to stay at your current job until you retire. Since you know a job change is going to come some time, whether by your choice or your employer's, you might as well keep on top of the market.

## Make contacts who might be able to help you

If there's one part of your business portfolio that needs to be cultivated and can't be rushed, it's building a network of helpful friends and contacts.

I was fired one Tuesday at 11am. By 4pm I had two job prospects with people I knew, and one with a company that a contact referred me to. That sort of network can't be thrown together at the last minute. It came from years of working in the open source community, making connections, and keeping in touch with these valuable people.

When you get tossed on the street, it's too late to start building your contact file. Start today.

## You should be doing all these things anyway.

For everyone who's reading this but isn't expecting layoffs, listen up.

*These rules should apply to how you live your work life anyway.*

All the rules above apply every day, even without the specter of layoffs. Whenever performance reviews come around, you'll want your boss to know your value. When the boss needs top performers to work on a sexy new project, you want to be at the top of that list. When upper management starts considering whether to outsource your department's operation, you and the rest of your department should be making it clear that you provide value no other group can.

By the time you need to harvest, the best time to plant has passed, and you'll be scrambling to do what you can to catch up.

## When you get laid off anyway

If you follow the guidelines above, when the worst happens and you're out on the street, you're ready to move on. You've planted the seeds for an effective job search that puts you above most other candidates for whatever job you may apply for.

Since you've been thinking about your value to the organization in terms of impact on the bottom line, you'll be ready to analyze your skills and how they can apply to any future employer.

Since you've been working to let your boss and others know about the good work you've been doing, and what improvements you've made to your department, you'll be more comfortable with talking about these achievements in your interviews.

Since you've been keeping your résumé current with your achievements, you don't have to go through the frustrating step of being in a rush to update it and remember everything you did at the job you just left.

## The law of the farm: Why you must start now

Start now. Plant a tree, grow a farm.

The law of the farm says that you have to plant, water and tend your crops, and wait until they've grown to maturity, before they're able to be harvested. There is no short-cut for this. You can't work extra hard at planting and have the crops show up early. If you wait until autumn to plant your seeds, you'll have nothing to harvest.

When it comes to the quality of your work, and how you're perceived on the job, the law of the farm holds as well. You can't slack off for a year, then pull off a heroic super-project, and expect to be seen as a great programmer. Your users' perceptions of you as an aloof, uncaring system administrator cannot be changed in a day, a week, or even a month.

To make yourself be seen as the best member of the team, you must start today, and keep it up always.

## Don't fear the Reaper

I hope you've noticed a common thread running through this article.

Everything that you need to do when expecting layoffs is something you should be doing anyway.

You need to build contacts, improve your skills and make yourself useful outside your department even if you're *not* expecting layoffs. Partly that's because it's just good career management, but mostly because there is no such thing as job security today.

Living in fear of losing your job also lowers the quality of your job, and your enjoyment of it. If you're worried about getting canned, it means you're less likely to take chances, to make bold decisions, or to try crazy new ideas. It also means you'll enjoy your job less, and life's too short spend time at the job that you're not enjoying.

So bulk up your skills, make your achievements known, and build a network of valuable contacts. Start today, and keep at it always. Together we'll get through this rotten economy doing the work we love.

## Acknowledgments

Thanks to Pete Krawczyk for his input and insight to this article.



### About the Author

Andy Lester has developed software for more than twenty years in the business world and on the Web in the open source community. Years of sifting through résumés, interviewing unprepared candidates, and even some unwise career choices of his own have spurred him to share what he has learned in a nontraditional book on the new guidelines for tech job hunting *Land the Tech Job You Love*<sup>[U2]</sup>, as well as on his website, *The Working Geek*<sup>[U3]</sup>. Andy is an active member of the open source community, and lives in the Chicago area.

### External resources referenced in this article:

- [U1] <http://pragprog.com/titles/algh/>
- [U2] <http://pragprog.com/titles/algh/>
- [U3] <http://www.theworkinggeek.com>



# Why Clojure?

---

## An Interview with Rich Hickey

*interviewed by Michael Swaine*

Rich Hickey created Clojure, a modern dialect of Lisp that targets the JVM. In this Pragmatic Programmers interview, Rich explains what that means to you.



Have you taken a look at Clojure yet?

Clojure is one of the most interesting new languages to arrive on the recent scene: an elegant, clean, modern version of Lisp created for functional programming, designed for concurrency, and compiling into JVM bytecode.

Clojure addresses the issues that have held Lisp back (libraries, readability, performance) while preserving its virtues. But what's stirring all the interest in Clojure is its potential for functional programming. As Stuart Holloway points out in [Programming Clojure](#)<sup>[U1]</sup>, "Massively multi-core hardware is right around the corner, and functional languages provide a clear approach for taking advantage of it." This is the application for which Clojure was created, so we knew we had to talk with its creator, Rich Hickey.

---

ms: I really appreciate your giving this time, Rich. I want to focus on basically one question: Why Clojure? For programmers in various situations, I want to ask: What's in it for me? Why should I want to read about Clojure? Why should I invest the time to learn it? How would it benefit me to program in Clojure?

rh: Sure thing.

---

ms: Cool. So say I'm an independent software developer and while maybe I'm not personally feeling the pain of the economic crunch, I do see signs that the nature of my client list may be changing. I want to be constantly learning new languages and technologies that broaden my options. Why should Clojure be on my short list of career-enhancing technologies?

rh: If you have not yet experienced functional programming, Clojure may offer the most approachable way to do so. Some have called it "the Python of functional programming", and I'll accept anything good that implies. Its seamless access to the Java ecosystem means you'll never be at a loss for libraries, and your applications can be delivered in an accepted environment. Clojure is new, but not disconnected. The infrastructure underlying it, the JVM, is quite excellent technology.

---

ms: The argument for functional languages these days is all about multicore processors and concurrent programming. Let's say I know something about functional programming and in fact have some very specific need

for a functional language for concurrent programming. Why is Clojure that language?

rh: Clojure is designed for concurrent programming, and specifically advocates that a functional style and pervasive immutability are prerequisites for concurrency. The data structures are immutable, and the locals are not “variable.” However, Clojure also recognizes the need for state in real applications, and provides language-supported mechanisms for managing state that avoid the locks-and-deadlocks headaches commonly found in other languages. Among functional languages (Haskell, ML, etc.) Clojure is relatively unusual in being dynamically typed, and in being connected to a mainstream infrastructure.

---

ms: Let’s talk about that infrastructure. Say I’m a Java developer. I’m not afraid to learn a new language, but I’m not about to abandon the whole Java ecosystem. What does Clojure offer me?

rh: Clojure lets you do just that—learn something new and not give up your investment and knowledge. That’s true of all of the JVM languages, though—Groovy, JRuby etc. Clojure is unique there in giving you the performance of a compiled language and the flexibility of a dynamic language. Performance is closer to Java than to Python. Access to Java from Clojure, and Clojure from Java, is easy, wrapper-free, and fast.

---

ms: Clojure’s performance may be close to raw Java, but the experience of programming in Clojure is very un-Javalike.

rh: Clojure may be the most different from Java of the popular JVM languages, and it is so for a reason—we are going to have to do things differently if we are going to leverage multicore, and our large OO programs have become spaghetti. If you really want to learn something new, rather than just do what you are currently doing slightly differently, Clojure is a good choice.

---

ms: OK, imagine you’re talking to an old Lisp hacker from way back, which in fact you are. And let’s say that I’ve moved on. Performance, libraries, fitting in with the crowd, for some reason I left Lisp. Why is Clojure the reason for me to get back into it?

rh: As a Lisp dialect, Clojure offers everything you love about Lisp—interactive development, elegance, succinctness, extensibility, expressiveness. It is aimed squarely at those areas that had caused people to leave, or not be able to use, Lisp in the past, in particular the library and poor citizenship issues. It is poised to leverage the huge amount of work done in Java, and in turn be leveraged by Java programs. In addition, Clojure moves Lisp forward in incorporating some of the best ideas developed over the years—building the core library on interface-based abstractions, lazy sequences, first-class associative data structures, etc. Lisps have been called functional programming

languages, Clojure embraces that more deeply than do Common Lisp or Scheme.

---

ms: Clojure is now at version 1.0. That can mean different things. Beyond being good for exploring functional programming, is Clojure ready for prime time? Say I want to use it right now for production work. How solid is Clojure?

rh: Clojure is quite solid. It has a very small core that rarely changes, and thousands of users pounding on it. It integrates with the Java tool ecosystem, so JVM debuggers, profilers etc work right out of the box. And IDE integration efforts are well underway, with good plugins for NetBeans and IntelliJ.

---

ms: Let's try some direct comparisons. Say I've devoted some time to learning Erlang. Why should I choose Clojure over Erlang?

rh: I'm loathe to engage in us vs. them, especially with Erlang, which I quite admire. If you truly have an application for which Erlang is best suited, e.g. a low-level distributed communications app with high uptime requirements, it's hard to beat it. Clojure is more of a general-purpose language, has better compute performance, better in-process SMP concurrency support, and a much better library and interoperability story. For distribution, you can choose from many techniques and tools, including message queues, some of which are written in ... Erlang.

---

ms: All right, the same question for the new Scala developer. Why should I choose Clojure over Scala?

rh: Clojure is simpler. It is dynamic. Having fewer paradigms to support, it is more focused on being functional. Being a Lisp, it has a regular syntax, and syntactic extensions are very consistent. Not being object-oriented, Clojure libraries have a different structure, encouraging a greater use of generic data structures, yielding higher interoperability. Clojure is based on the idea that immutability and functional programming are more important contributors to program robustness than is static typing. If these values resonate with you, you will likely prefer Clojure, but I certainly expect Clojure and Scala to peacefully coexist.

---

ms: Well, you've convinced me that Clojure is worth a look. Thank you for taking the time to chat with us.

rh: Sure. This was a good exercise. Thanks for suggesting it.

Rich Hickey has over twenty years of experience in software development in a wide variety of projects from broadcast automation to database design.

**External resources referenced in this article:**

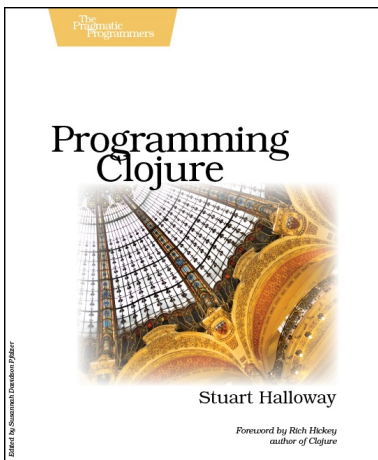
[U1] <http://www.pragprog.com/titles/shcloj/programming-clojure>

# When Things Go Wrong

## Clojure's Exceptional Handling of Exceptions

by Stuart Halloway

Stuart wrote [the book](#) [U1] on Clojure, and here he reveals one of its strengths: the error-kit Condition system, which gives you greater control and flexibility than traditional exception handling.



If you don't know about *conditions*, you should. Conditions are basically exception handling, but with greater flexibility. Many Lisps feature a condition system, and Clojure is no exception (pun inflicted by editor). Clojure's condition system is called **error-kit**. In this article, you will learn how to use **error-kit**, and why you will prefer it to plain old exception handling.

## Getting the Code

You don't need to have bought my book [Programming Clojure](#) [U2] to understand this article, but why wouldn't you want to? ;) You can follow along throughout this article by entering the code at Clojure's Read-Eval-Print Loop (REPL). To install a REPL on your local machine, download the sample code from [the book](#) [U3]. The sample code has its own home page at <http://github.com/stuarthalloway/programming-clojure> [U4].

The sample code includes a prebuilt version of Clojure, and the `clojure-contrib` library that contains **error-kit**. To launch a REPL, execute `bin/repl.sh` (Unix) or `bin/repl.bat` (Windows) from the root of the sample code project. You should see the following prompt:

```
Clojure
user=>
```

For your reference, the completed sample is included in the download at `examples/error_kit.clj`.

## A Simple Problem: Parsing Log File Entries

To see how **error-kit** handles exceptions, we'll create a simple application and perpetrate some errors. Let's write an app that parses log file entries. Our log file entries will look like this:

```
2008-10-05 12:14:00 WARN Some warning message here...
```

In this imperfect world, it is inevitable that some miscreant will pass bad data to the log file parser. To deal with this, we will define an error:

```
(use 'clojure.contrib.error-kit)
(deferror malformed-log-entry [] [msg]
  {:msg msg
   :unhandled (throw-msg IllegalArgumentException)}}
```

The error takes a single argument, a `msg` describing the problem. The `:unhandled` value defers to a normal Clojure (Java) exception in the event that a caller chooses not to handle the error. (The empty vector `[]` could contain a parent error, but we won't need that in this example.)

Now, let's write a `parse-log-entry` function:

```
(defn parse-log-entry [entry]
  (or
    (next (re-matches #"(\d+-\d+-\d+) (\d+:\d+:\d+) (\w+) (.*)" entry))
    (raise malformed-log-entry entry)))
```

The first argument to `or` uses a regular expression to crack a log entry. If the log entry is not in the correct format, the second argument to `or` will *raise* an error. Try it with a valid log entry:

```
(parse-log-entry
  "2008-10-05 12:14:00 WARN Some warning message here...")
-> ("2008-10-05" "12:14:00" "WARN" "Some warning message here...")
```

Of course, we could do more than just return a simple sequence, but since we are focused on the error case we'll keep the results simple.

What happens with a bad log line?

```
(parse-log-entry "some random string")
-> java.lang.IllegalArgumentException: some random string
```

An unhandled error is converted into a Java exception, and propagates as normal.

## The Problem with Exceptions

So why wouldn't we simply throw and catch exceptions? The problem is one of context. At the point of an exception, you know the most intimate details about what went wrong. But you do *not* know the broader context. How does the calling subsystem or application want to deal with this particular kind of error? Since you do not know the context, you throw the exception back out to someone who does.

At some higher level, you have enough context to know what to do with the error, but by the time you get there, you have lost the context to continue. The stack has unwound, partial work has been lost, and you are left to pick up the pieces. Or, more likely, to give up on the application-level task that you started.

## The Solution: Conditions

Conditions provide a way to have your cake and eat it too. At some high-level function, you pick a *strategy* for dealing with the error, and register that strategy as a *handler*. When the lower-level code hits the error, it can then pick a handler *without unwinding the call stack*. This gives you more options. In particular, you can choose to cope with the problem and continue.

Let's say that you are processing some log files that include some garbage lines, and that you are content to skip past these lines. You can use `with-handler` to execute the code with a handler that will replace bad lines with, for example, a simple `nil`.

```
(defn parse-or-nil [logseq]
  (with-handler
    (vec (map parse-log-entry logseq))
    (handle malformed-log-entry [msg]
      (continue-with nil))))
```

The call to `continue-with` will replace any malformed log entries with `nil`. Despite the structural similarity, this is *not at all* like a catch block. The `continue-with` is specified by an outer calling function (`parse-or-nil`) and will execute inside an inner, called function (`parse-log-entry`).

To test `parse-or-nil`, create a few top level vars, one with a good sequence of log entries, and one with some corrupt entries:

```
(def good-log
  ["2008-10-05 12:14:00 WARN Some warning message here..."
   "2008-10-05 12:14:00 INFO End of the current log..."])
(def bad-log
  ["2008-10-05 12:14:00 WARN Some warning message here..."
   "this is not a log message"
   "2008-10-05 12:14:00 INFO End of the current log..."])
```

The `good-log` will parse without any problems, of course:

```
(parse-or-nil good-log)
-> [("2008-10-05" "12:14:00" "WARN" "Some warning message here...")
    ("2008-10-05" "12:14:00" "INFO" "End of the current log...")]
```

When parsing hits an error in `bad-log`, it substitutes a `nil` and moves right along:

```
(parse-or-nil bad-log)
-> [("2008-10-05" "12:14:00" "WARN" "Some warning message here...")
    nil
    ("2008-10-05" "12:14:00" "INFO" "End of the current log...")]
```

OK, but what if you wanted to do more than just return `nil`? Maybe the original API signals an error, but doesn't do any logging. No problem, just impose your own logging from without:

```
(defn parse-or-warn [logseq]
  (with-handler
    (vec (map parse-log-entry logseq))
    (handle malformed-log-entry [msg]
      (continue-with (println "****warning****: invalid log: " msg))))))
```

Now, parsing the `bad-log` will log the problem.

```
(parse-or-warn bad-log)
****warning****: invalid log: this is not a log message
-> [("2008-10-05" "12:14:00" "WARN" "Some warning message here...")
    nil
    ("2008-10-05" "12:14:00" "INFO" "End of the current log...")]
```

Of course a production-quality solution would use a real logging API, but you get the idea. Slick, huh?

## Make Life Simple For Your Callers

It gets even better.

If you know in advance some of the strategies your callers might want to pursue in dealing with an error, you can name those strategies at the point of a possible error, and then let callers select a strategy *by name*. The `bind-continue` form takes the name of a strategy, an argument list, and a form to implement the strategy.

So, continuing with our log example, you might choose to provide explicit `skip` and `log` strategies for dealing with a parse error:

```
(defn parse-or-continue [logseq]
  (let [parse-log-entry
        (fn [entry]
          (with-handler (parse-log-entry entry)
            (bind-continue skip [msg]
              nil)
            (bind-continue log [msg]
              (println "****invalid log: " msg)))))]
    (vec (map parse-log-entry logseq))))
```

parse-or-continue has no continue-with block, so a bad log entry will default to a Java exception:

```
(parse-or-continue bad-log)
-> java.lang.RuntimeException: java.lang.IllegalArgumentException:
    this is not a log message
```

Callers of parse-or-continue can select a handler strategy with the continue form. Here, the call selects the skip strategy:

```
(with-handler (parse-or-continue bad-log)
  (handle malformed-log-entry [msg] (continue skip msg)))
-> [("2008-10-05" "12:14:00" "WARN" "Some warning message here...")
   nil
   ("2008-10-05" "12:14:00" "INFO" "End of the current log...")]
```

And here it selects the log strategy:

```
(with-handler (parse-or-continue bad-log)
  (handle malformed-log-entry [msg] (continue log msg)))
****warning****: invalid log: this is not a log message
-> [("2008-10-05" "12:14:00" "WARN" "Some warning message here...")
   nil
   ("2008-10-05" "12:14:00" "INFO" "End of the current log...")]
```

Notice the continue forms pass an argument to the bound continues. In these examples we just passed the error message, but the parameter list could be used to implement arbitrary coordination between continue calls and bound continue forms. This is powerful.

## Laziness and Errors

Most Clojure data structures are *lazy*, which means that they are evaluated only as needed. To make these lazy structures play nicely with conditions (or even plain old exceptions, for that matter), you have to install your handlers around the code that actually *realizes* the collection, not around the code that *creates* the collection.

This can be confusing at the REPL. Can you spot the problem below?

```
(with-handler
  (map parse-log-entry bad-log)
  (handle malformed-log-entry [msg]
    (continue-with nil)))
-> java.lang.IllegalArgumentException: this is not a log message
```

The code above is trying to add a handler, but it isn't working. Stepping through the sequence of events will show why:

1. The "with-handler" block sets a handler.
2. The "map" creates a lazy sequence.

3. The “handler” block exits, returning the lazy sequence to the REPL.
4. The REPL realizes the sequence to print it, but by now the handler is gone. Oops.

In the earlier examples we avoided this problem by explicitly realizing the sequence with calls to `vec`. Here’s the takeaway: In your own applications, make sure to install handlers around *realization*, not instantiation.

## Wrapping Up

Traditional exception handling gives you two points of control: the point of failure, and the handler. With a condition system, you have an all-important third point of control. Handlers can make *continues* available at the point of failure. Low-level functions can then raise an error, and higher-level functions can deal with the error *at the point it occurred, with full context*.

If you have ever found a large project staggering under the weight of exception handling code, you might want to consider giving conditions a shot.

## Notes

The choice of log file parsing for the example was inspired by a [similar example](#) [U5] in Peter Seibel’s excellent book [Practical Common Lisp](#) [U6].

Copyright 2009 Stuart Halloway. Used with permission.



### About the Author

Stuart Halloway is a co-founder and CEO of Relevance, Inc. Relevance provides development, consulting, and training services based around agile methods and leading-edge technologies such as Ruby and Clojure. In addition to *Programming Clojure*, Stuart has authored several other books, including *Component Development for the Java Platform* and *Rails for Java Developers*.

### External resources referenced in this article:

- [U1] <http://www.pragprog.com/titles/shcloj/programming-clojure>
- [U2] <http://www.pragprog.com/titles/shcloj/programming-clojure>
- [U3] <http://www.pragprog.com/titles/shcloj/programming-clojure>
- [U4] <http://github.com/stuarthalloway/programming-clojure>
- [U5] <http://gigamonkeys.com/book/beyond-exception-handling-conditions-and-restarts.html>
- [U6] <http://gigamonkeys.com/book/>



# Pragmatic Publishing

---

## Dave Thomas Talks about Ebooks, Agility, and the Future of Books

*interviewed by Michael Swaine*

For our inaugural issue (or First Iteration) it seemed appropriate to talk with one of the Pragmatic Programmers. Dave Thomas was willing to chat about the decisions behind Pragmatic Bookshelf's comprehensive ebook program and about the revolutionary changes going on in publishing.



A revolution has been going on in publishing, and current economic conditions seem to have accelerated it. At the end of March, *Bookseller* reported that the printing industry was in its worst state in 18 years.

Newspapers are dropping like flies. Clay Shirky points out that newspapers weren't blindsided by the internet; it's just that none of the perfectly reasonable strategies they came up with worked, because there literally is no new model for newspapers to replace the old one.

Books are also being hurt by the internet and the economy. HarperCollins cut Collins, Random House is cutting staff in waves, and Borders looks doomed.

At the same time, actual newspaper readership (print plus online) is up, and some book publishers in some markets are doing very well. It was in this unsettled context that the Pragmatic Bookshelf recently began producing all of its books in every popular electronic format: .mobi (e.g., Amazon Kindle), .epub (e.g., Apple iPhone), and .pdf in addition to print. And if you buy one, you've got them all: you have purchased a license to the content. You can purchase the ebook for your iPhone and then when you get an Amazon Kindle, you can download that version, too, because you already own it.

It seemed like an ideal time to chat with Dave Thomas, one of the two owners of the Pragmatic Bookshelf, about ebooks, agility, and the future of publishing.

---

**ms:** These are interesting times to be in publishing. Clay Shirky has described this as being inside a revolution, and says that the inside of a revolution feels like chaos, like constant change. Are we inside a revolution? And if so, how do you adapt to constant change?

**dt:** We are inside a revolution. It's like being in a whirlwind. And no, you can never adapt to constant change, but you can deal with it. The trick is you have to be agile, you have to be able to refocus quickly. And we know something about that: to be able to refocus quickly you need to have all the mechanical stuff, all the non-intellectual stuff, automated. That's something we work on all the time, to be as automated as possible.

---

**ms:** Although many book publishers are struggling, at least with books there is a plausible path forward, unlike the situation with newspapers. I'm referring to ebooks. You have embraced electronic publishing since the start and now all your books are available in multiple ebook formats. And it seemed like you were able to convert them to the new formats almost overnight.

dt: Because we built our company on agility, we're in a comfortable position. All our books are authored in custom markup so all our content can be easily repurposed for multiple platforms, Kindle for example. It wasn't literally overnight, though; it took a few days. But that's because I did it all myself, one person, working on it part-time. And although we use custom markup—automating as much as possible—we converted each book individually to give each one its individual treatment. At this point we think all the books look pretty good on all supported devices.

---

ms: Any time you move content to a new medium there are unanticipated consequences. A recent *New York Times* article raises an interesting challenge for electronic books: once you're on the platform, more immediately gratifying activities like games are just a click away from the book. Reading has always had to compete with distractions, but with ebooks it has to compete more aggressively, it seems.

dt: I disagree that games and such are competition for ebooks. Or rather, I totally agree when it comes to fiction. I would be afraid if I were publishing novels. But for our kind of publishing—nonfiction, reference oriented—games and the like are not competition at all. They address different needs. I have a time when I want to learn and a time when I want to play. No, the competition for the ebook isn't games, it's the Web and free content.

---

ms: Or even not-free content. In addition to new ways of presenting the contents of books, we are seeing new ways of marketing books, including self-published, print-on-demand books, like Author Solutions and Lulu. You can publish and sell a book on Amazon now for a few hundred dollars. If you have a recognized expertise, a website, and a following, you can sell on your own site. The odds are long but the startup costs are low, at least apart from the sizable investment of your own time.

dt: Lulu and other self-publishing options are a step up from simply selling on the Web. So why is that not a threat to ebooks? That really comes down to the question of what publishers offer. We think about that a lot. Most authors need help to get a book out. Help organizing their material. Help finding their voice. They need a coach, they need encouragement to get it finished. That's something that a publisher can offer. Also, we're a place where people come to buy books. When it works right, it's a mutually beneficial relationship. A good publisher's reputation is built on the work of its authors, and the authors benefit from that reputation.

---

ms: So how significant are ebooks? Are they the future of books, or of major categories of books?

dt: That's difficult to answer. I have mixed feelings about the fact that as an industry we have started using this term ebook. I can see why we do: it's a comfortable metaphor. But it leads us into some strange decisions. For example, when we're reading on screen, we're clicking

through a document page by page. Why? Why would we want to replicate one of the worst features of print? I'm concerned that the word "book" in "ebook" implicitly limits where we think we can go.

---

ms: So where might we go? Do you see something new that eventually replaces the ebook of today?

dt: Taking existing content and repurposing it for ebook readers is the first baby step. We need to look beyond that. But asking what will replace the ebook is looking at it the wrong way around. The fundamental issue we have to address is not what will happen to the book but what will happen to content. The real question is what we want to create in terms of content.

---

ms: So how do you go about answering that?

dt: Listen to the reader. The two keys to dealing with change are being agile and listening to your customers. In the past five years control has passed from supplier to consumer, in publishing and many other markets. That's something that was talked about and it's really happening, and it's a wonderful position to be in. Well, maybe in the Chinese curse sense of wonderful. But it can be wonderful if you embrace it. The old model was the publisher gets to decide what we read. The new model is you listen to what your readers want and try to produce content for them. You succeed by listening to your customers and delivering what they want.

---

ms: And what do you find that readers want?

dt: One thing we're learning that readers want is customizable content. When we get a new video game in my family, I want to start playing right away, but my kids want to spend the first half-hour setting things up: the colors, their characters' names, and so on. They really enjoy customizing their environment—that's what's important to them. You need to deliver content that the customer can adapt to their situation. Some people want books, some people want audio presentation. Different people want different delivery of the same content. That's why we're doing screencasts, seminars, podcasts, this magazine, in addition to books on paper and in a variety of electronic formats. But that's really just the beginning. We will be customizing the content—and enabling readers to customize the content—even more as we listen to our readers and determine what they want and need.

---

ms: How do you do that?

dt: How do you take an idea and create multiple implementations of it? It's a problem. Customizability is hard. Most publishers can envisage incredibly rich content models. The problem is how do you get them made? Right now we can cobble something together with duct tape and Ruby scripts, but that's not a solution. The challenge is that writing itself is hard. The average book is a labor of love. You tell an author

that he also has to produce a podcast and courseware and you'll drastically reduce the number of books that even get started.

---

ms: So what's your strategy?

dt: Again, you have to trust your readers. They are the ones who know what they need and want. Publishers traditionally have tried to tell them what they need. We won't do that. They say we don't want DRM, well neither do we, so no DRM. We sell direct so we have that relationship with the customer, so we can have a conversation. And we use that conversation to inform our various experiments with different kinds of content delivery.

---

ms: I'm learning that you mean that pretty literally. You really engage with your readers. You do your own support. What if what readers want is unreasonable or unrealistic?

dt: You have to draw a fine balance between being reactive and being conservative. We play with a lot of stuff that never sees the light of day. We look for the long-term value as opposed to the short-term gain.

---

ms: It seems to be working.

dt: It seems to be. Last year we celebrated our fifth year in publishing. We're a young company, but we're growing every year, even in this economy. Publishers are struggling all around us and we're going from strength to strength. We're having a whole lot of fun. I just wish there were about three times as many hours in the day to do all the things we want to do.

# The Quiz

## This Month: A is for APL

So you think you know programming languages? Challenge yourself with this month's puzzle. Answer next month.



Below is a simple acrostic. On the left you'll find 6 examples of programming languages. Write the first letter of the name of the language in the box at the right. When you're done, the right column will spell the name of a seventh language. See how far you can get before you start searching.

<pre>Process Class Creator; Begin   While true do begin     Activate New Consumer(Time);     Hold(Uniform(5, 15, 1));   End While; End of Creator;</pre>	
<pre>&lt;Any T, Any U&gt; T first((T,U) tuple) {   (T t, U u) = tuple;   return t; }</pre>	
<pre>PROC null.farm(CHAN OF ADDR.TASK.STREAM from.farm,                 CHAN OF ADDR.RESULT.STREAM to.farm ) PAR   from.farm ? CASE no.more.task.packets   to.farm ! no.more.result.packets :</pre>	
<pre>LET start() = VALOF \$( FOR i = 1 TO 5 DO writef("%n! = %i4*n", i, fact(i))   RESULTIS 0 \$) AND fact(n) = n=0 -&gt; 1, n*fact(n-1)</pre>	
<pre>PROCEDURE speak*( VAR bird : Birds.Bird ); BEGIN   WITH bird : Cuckoos.Cuckoo DO     bird.sound := "Cuckoo!";     bird : Ducks.Duck DO     bird.sound := "Quack!";   ELSE     bird.sound := "Tweet!";   END; END setSound;</pre>	
<pre>HAI CAN HAS STDIO? IM IN YR LOOP UPPIN YR NUM TIL BOTHSAEM NUM AN 10   VISIBLE SUM OF NUM AN 1 IM OUTTA YR LOOP KTHXBYE</pre>	

## Answer to Last Month's Quiz

# Shady Illuminations

## Microsoft Buys a Verb

by John Shade

Can Microsoft really challenge Google on its own turf? And why would they even try? John Shade casts a jaundiced eye at Bing, Wolfram Alpha, and other attempts to transcend Google.



Microsoft is an agile company.

You doubt me? I can understand that. I'm pretty sketchy at the best of times. You probably figure that being agile is one of those lean and hungry things, while Microsoft is more of a fat and bloated thing. You get no argument from me. After the first billion dollars or so, any company can pretty much forget about being described as lean, even by its most loyal sycophants. But I'm standing my ground on hunger: no matter how huge and bloated Microsoft gets, it always stays hungry. Hunger got inside Microsoft when it was just a greedy leer in Bill Gates's eye. Microsoft has hungry DNA. Hungry, paranoid, and quintessentially nerdy DNA.

Yes, nerdy. I hate to tell you this, but as long as there is a Microsoft you will never get rid of the popular stereotype of a computer nerd. Microsoft makes the stereotype true. Microsoft as a company is killer smart, socially inept, and wears orange socks.

This is mere common knowledge.

But agile, you ask? Yes, agile. The Microsoft agility mantra is Agility through Paranoia. Bill Gates—or the spirit of Bill Gates that is the twisted soul of Microsoft—has always been motivated by the certainty that someday some bright young hacker will come along and redefine the market, rearchitect the platform, rewrite the rulebook, move the cheese, or somehow change some fundamental something and rip the rightful riches from Microsoft's jewel-encrusted belly.

Technically it's always *two* bright young hackers. Andreessen and Bina, Filo and Yang, Page and Brin. Why two? Think Gates and Allen: Microsoft itself was founded by two bright young hackers who changed the game, so they know how the game-changing game is played. That's the Microsoft corporate view of pair programming, as a matter of fact: some pair of programmers somewhere is at this very moment plotting our destruction. You probably didn't know that.

## What Microsoft Wants

So I ask myself, what does Microsoft, in all its bloated nerdy paranoid agility, want? Easy, it wants what Google has. It wants a verb.

The verb "to google" is in the OED. *The OED!* People who've never used The Google talk glibly about googling their acquaintances. Google has attained to the holy pantheon of Verbed Brands. It's up there with xerox and slashdot and twitter and tivo on Brand Olympus. Even Apple and Sony aren't verbs. Once you're verbed, you're forever. You can't buy cred like that.

Unless you're Microsoft.

Microsoft would like to buy a verb. Microsoft has never had a verb. Nobody *words* a letter or *excels* a budget. Some people use *powerpoint* as an epithet, but it's not the same. Microsoft wants to buy a verb, and the verb it wants to buy is *bing*. If Microsoft has its way you will soon be binging left and right. You'll tell your friends to just bing it, you'll assure your boss or client, hey no problem, I can bing that. You'll confess to spending all afternoon binging. You'll become a hardcore binger.

Bing, as you know unless you've been living under a hype-blocking rock, is the name Microsoft has given to the latest version of its Live Search technology. I liked Live Search. The name, I mean, not the search tool. Live Search was a straightforward name; it had no personality, but it had character. But Live Search suffered from two problems. First, Google *owns* search. Second, Google owns the *word* for search. Live Search? Is that something you use to google things? See? It doesn't work. It's quixotic to try to compete with the company that owns the category, but it's flat-out stupid to try to compete with the company that *owns the word* for the category.

But Microsoft can't walk away from search any more than it could walk away from the eyeball battlefield of the 1990s that was hilariously miscalled the browser "market," and for the exact same reason. The hive mind that is the Microsoft brain trust lives in mortal fear of those bright young hackers who change the rules of the game. And Page/Brin is the new Andreessen/Bina. In the 1990s the emerging center of the galaxy was the browser window; in the 2000s it's the search engine results page. The SERP.

So Microsoft has to compete with Google but it can't compete with Google. What's the solution? Easy: redefine the category. Declare search dead and christen its replacement. Break a bottle over its bow and call it Bing.

Here's how it's intended to work: Google owns search and the name for search, but search is just a service. The SERP, though, is concrete. It's the internet's prime real estate. That's what you need to own, and if you can peel that away from Google, you win. So you just need to bribe people to come to your SERP and somehow get them to stay. Then you monetize the heck out of it. Flog those eyeballs for all they're worth.

OK, you see the flaw in this plan, I suppose. To get people to hang around on your binging SERP, you've got to make it sticky. Well, even I know how to accomplish that: the page just needs to be extremely well designed, focused with laserlike intensity on function, rich, simple, and elegant. That's all. And you just know that Microsoft's natural inclination is to chintz it up with five flavors of gingerbread and dress it in orange socks. If anyone can create a non-sticky SERP, it's Microsoft.

## The BitTorrent of Search

To see how you might go about end-running around Google with a better SERP, take a look at [Cuil](#) [11]. I'm sure Microsoft did. Cuil indexes massive amounts of data, analyzes the context of discovered search terms, and presents the results as a sort of newspaper front page because, hey, nothing says 2009 like a newspaper. Cuil usually figures out that your search term has several

meanings and offers the opportunity to dig deeper in any of these meanings in a sidebar, sort of like a Wikipedia disambiguation page.

The key is to claim that you're doing more than just search. Semantic Knowledge Discovery through Relevance-Intuiting Neural Network Algorithms. I just made that up, but it's more or less the template. Feel free to steal it. Chances are, I did. [Yebol](#)<sup>[U2]</sup>, in fact, promises smarter search through neural networks. [Wowd](#)<sup>[U3]</sup> wants to be the BitTorrent of search. [Hakia](#)<sup>[U4]</sup> and [Clusty](#)<sup>[U5]</sup> do clustering: grouping semantically-related results into categories for further search. So does Cuil, it seems to me. So does Bing. So, in fact, does Google, but we're not talking about Google here.

The one thing you absolutely must do is to refer to Google's SERP as "ten blue links." Because, you know, [blue is so 2008](#)<sup>[U6]</sup>.

Or if you're a super-genius you can skip search entirely and just *compute* the answers people are looking for. That's what [Wolfram Alpha](#)<sup>[U7]</sup> claims to do. After reinventing science, super-genius and Mathematica language developer Stephen Wolfram retreated to his secret lab in an undisclosed location at 100 Trade Center Drive in Champaign, Illinois. Wolfram is so brilliant that he powers light bulbs, so you just knew he was working on some radical project destined to stun the world.

Now after seven years he has emerged, and the world is well and truly gobsmacked. Turns out the polymathematician inventor of A New Kind of Science has been working on a search engine. Or rather, A New Kind of Google. Or maybe A New Kind of Interface to Wikipedia.

Wolfram Alpha is the eponymous answer engine, capable, according to its inventor, of parsing English-language queries and not merely looking up but actually *computing* the answers using the awesome computational power of Stephen Wolfram's brain channeled through Mathematica functions and crunched on multiple supercomputers, ultimately to be displayed as glorious Gif images.

As soon as Wolfram Alpha, or WA, as I like to call it, went online, I was there to poke it with some pointed questions.

---

**Me:** [What do they call a quarter pounder with cheese in France?](#)

**WA:** Wait, wait, I know this.

---

**Me:** [I'm wai-ting.](#)

**WA:** Assuming any type of McDonald's Quarter Pounder | Use McDonald's Quarter Pounder, plain or McDonald's Quarter Pounder, with cheese instead.

---

**Me:** [With cheese, please.](#)

**WA:** McDonald's Quarter Pounder, with cheese: serving size 1 sandwich (185 g), total calories 460, total fat 24 g, saturated fat 9 g, trans fat 1 g.



---

Me: Argh. And in France they call it—?

WA: France: country, calling code +33.

---

Me: Maybe we should try something simpler. Try this: how many ounces per pound?

WA: Result: 0.0625.

Not in my kitchen it isn't.

## Bing: Google, Embraced and Extended

So how terrible is Bing? Now, now that's not a healthy attitude. Just look where that kind of cynicism has gotten me. The fact is, Bing is "much better than expected," to quote one reviewer. That's the kind of treatment you come to expect if you're Microsoft: "We assumed it would be crap, but it's not half bad." Bing is not half bad. Here are some of the areas where Bing is clearly superior to Google:

- \* Shopping. For example, when you're looking for things to buy, Bing has a cashback program. (Microsoft will bribe you to use Bing.)
- \* Travel advice. Bing gives good advice on airline travel using the Farecast service Microsoft bought. (Bing excels at promoting Microsoft properties.)
- \* Video. When Bing finds a video, it doesn't just give you one of those blue links, it plays the video for you right there in the SERP. (Testing the video copyright waters for the rest of us.)
- \* Protecting you from your nasty self. If you live in India, you won't be troubled with inappropriate sexual offers because Bing won't let you search for "sex." (That's what Craigslist is for.)

Danny Sullivan, the Seymour Hersh of search engines, complains that Bing clutters up its fine collation of travel, shopping, and local results with paid listings. I think Danny misses the point. Microsoft spent eighty million dollars promoting Bing. (And that was without Jerry Seinfeld.) It's got to get that eighty million back some way. That's why Microsoft has redesigned MSN to funnel visitors into Bing. Because they're not going to [www.bing.com](http://www.bing.com). And it's probably why they're hawking all that Bing bling. Although Microsoft being Microsoft, I'm not sure whether Bing coffee mugs are intended as a way for Microsoft to make money off Bing or to spend money on Bing.

Because Microsoft isn't stopping at a mere eighty million dollars. The company's sitting on umpty billion in cash and doesn't know what to do with it. Trying to take on Google in search is really a brilliant idea if your problem is how to burn through a few billion fast. So maybe they plan to broadcast those Bing tchotchkes on the breeze like AOL CDs.

Then again, Microsoft doesn't have to outrun the bear. You know the old joke:

Two lawyers are hiking through the woods and spot an unfriendly-looking bear. The first lawyer pulls a pair of sneakers out of his briefcase (in this joke, lawyers carry briefcases while hiking through the woods, OK?) and puts them

on. The second lawyer stares at him and says, “You’re crazy! Bears can run like 35 miles an hour! You’ll never be able to outrun that bear!”

“I don’t have to outrun the bear,” the first lawyer says. “I only have to outrun you.”

It wouldn’t have to be two lawyers, of course, but almost any joke is improved by putting a lawyer in it. OK, just to be repulsively obvious, lawyer number two is Yahoo. And Bing did indeed outrun the second lawyer during Bing’s honeymoon period.

Google’s response to all this sincerest form of flattery? Why, Google Squared, of course. If they’re going to raise the pot we’ll double down, Google says, mixing its card-playing metaphors. Google Squared is a Google Labs project that present search results in tabular form because the dazzling success of Wolfram Alpha and Cuil and Bing and the rest has convinced the Google gang that what you really want on your SERP is structured search result data, and, well, putting it in a spreadsheet makes it structured, right? Right.

As God is my witness, I thought turkeys could fly.

Looking at early tests of this project, I have to conclude that this turkey is not the exception to the no-fly rule. In fact the only explanation I can see is that Google Squared is snarkware. It’s Google’s way of making fun of the competition. They’re such a fun-loving crowd.

But the one shining and enduring truth of all search engines is this: most of what they give you is irrelevant, useless, or wrong. For all its computational power, Wolfram Alpha doesn’t know how many ounces there are in a pound. Clusty and Hakia and Bing and Cuil know, but they get other things wrong. Google gets it right with its first blue link: “1 pound = 16 ounces.” Yay, Google. But for its second blue link it quotes Yahoo Answers: “I thought it was 12, but it may be 16, I don’t know.” Woohoo, Yahoo.

You rarely get such refreshing honesty from a search engine.

#### About the Author

John Shade was born in Montreux, Switzerland in 1962. Subsequent internment in a series of obscure institutions failed to enlighten him so much as a foot-candle. Today he frets away the idle hours wondering if you got the light bulb joke.

#### External resources referenced in this article:

- [U1] <http://www.cuil.com>
- [U2] <http://www.yebol.com>
- [U3] <http://www.wowd.com>
- [U4] <http://www.hakia.com>
- [U5] <http://clusty.com>
- [U6] <http://www.pantone.com/pages/Pantone/Pantone.aspx?pg=20540>
- [U7] <http://www.wolframalpha.com>

# Calendar

Going to OSCON in July? Maybe we'll see you there.  
Or at another of these upcoming events.



- July 9-12 **FutureRuby Conference** <sup>[U1]</sup>  
Toronto, Ontario
- July 15-17 **Advanced Ruby Studio** <sup>[U2]</sup>  
Reston, VA \*SOLD OUT\*
- July 20-24 **OSCON** <sup>[U3]</sup>  
San Jose, CA
- July 15 **Deadline for former IBM executive Papermaster to testify under penalty of perjury that he has not used or disclosed IBM confidential information, and that he does not intend to do so.**  
Southern District of New York
- Wed Jul 22 **Sessions: Just Enough C For Open Source Projects; Effective Job Interviewing from Both Sides of the Desk**  
Andy Lester, author of [Land the Tech Job You Love](#) <sup>[U4]</sup>  
OSCON 2009, San Jose, CA
- Wed Jul 22 **Session: Testing iPhone Apps with Ruby and Cucumber**  
Ian Dees, author of [Scripted GUI Testing With Ruby](#) <sup>[U5]</sup>  
OSCON, San Jose, CA
- Wed Jul 22 **Session: Practical Object-Oriented Models in SQL**  
Bill Karwin, author of [SQL Antipatterns](#)  
OSCON, San Jose, CA
- July 24-25 **Rails Underground** <sup>[U6]</sup>  
London
- Fri Jul 24 **Rails Underground**  
Maik Schmidt, author of [Enterprise Recipes with Ruby and Rails](#) <sup>[U7]</sup>  
London, UK
- July 30-Aug 1 **RubyRX / AgileRX** <sup>[U8]</sup>  
Philadelphia, PA
- Thu Jul 30 **AgileRX** <sup>[U9]</sup>  
Esther Derby, author of [Behind Closed Doors: Secrets of Great Management](#) <sup>[U10]</sup>; [Agile Retrospectives: Making Good Teams Great](#) <sup>[U11]</sup>  
Philadelphia, PA
- Thu Jul 30 **RubyRX/AgileRX** <sup>[U12]</sup> **This combo conference in Philly has the best of both Ruby and Agile speakers!**  
Jared Richardson, author of [Ship It!](#) <sup>[U13]</sup>  
Philadelphia, PA
- Aug 4-7 **iPhone SDK Studio** <sup>[U14]</sup>  
Reston, VA
- Aug 13-16 **Filemaker Developer Conference 2009** <sup>[U15]</sup>  
San Francisco, CA
- Aug 19-21 **Ruby on Rails Studio** <sup>[U16]</sup>  
Denver, CO
- Thu Aug 20 **Biz Conference**  
Esther Derby, author of [Behind Closed Doors: Secrets of Great Management](#) <sup>[U17]</sup>; [Agile Retrospectives: Making Good Teams Great](#) <sup>[U18]</sup>  
Amelia Island, Florida

- Aug 24-28 **Agile 2009** <sup>[U19]</sup>  
Chicago
- Mon Aug 24 **Agile Coaching workshop, Agile2009 Conference**  
Rachel Davies and Liz Sedley, authors of *Agile Coaching*  
Chicago, IL
- Mon Aug 24 **Agile 2009**  
Esther Derby, author of *Behind Closed Doors: Secrets of Great Management* <sup>[U20]</sup>;  
*Agile Retrospectives: Making Good Teams Great* <sup>[U21]</sup>  
Chicago, IL
- Mon Aug 24 **Sessions: *Idea Factory* (with David Carlton) and *Eight Guiding Values***  
Brian Marick, author of *Everyday Scripting With Ruby* <sup>[U22]</sup> and the upcoming  
RubyCocoa book  
Agile 2009
- Aug 25-28 **iPhone SDK Studio** <sup>[U23]</sup>  
Denver, CO
- Aug 25 **Linux turns 18. Free krill for all!**
- Thu Sep 3 **AgileRX**  
Esther Derby, author of *Behind Closed Doors: Secrets of Great Management* <sup>[U24]</sup>;  
*Agile Retrospectives: Making Good Teams Great* <sup>[U25]</sup>  
Reston, VA
- Fri Sep 4 **RubyRX/AgileRX**  
Jared Richardson, author of *Ship It!* <sup>[U26]</sup>  
Washington, DC
- Sept. 9-10 **Gov 2.0 Summit** <sup>[U27]</sup>  
Washington, DC
- Sept 22-24 **EmTech09** <sup>[U28]</sup>  
Cambridge, MA
- Sept 28-30 **NanoTech Europe** <sup>[U29]</sup>  
Berlin

#### External resources referenced in this article:

- <sup>[U1]</sup> <http://futuresruby.com/>
- <sup>[U2]</sup> <http://pragmaticstudio.com/>
- <sup>[U3]</sup> <http://en.oreilly.com/oscon2009>
- <sup>[U4]</sup> <http://www.pragprog.com/titles/algh>
- <sup>[U5]</sup> <http://www.pragprog.com/titles/idgtr>
- <sup>[U6]</sup> <http://www.rails-underground.com/>
- <sup>[U7]</sup> <http://www.pragprog.com/titles/msenr>
- <sup>[U8]</sup> <http://www.nfjsone.com/conference/philadelphia/2009/07/index.html>
- <sup>[U9]</sup> <http://nfjsone.com>
- <sup>[U10]</sup> <http://www.pragprog.com/titles/rdbcd>
- <sup>[U11]</sup> <http://www.pragprog.com/titles/dlret>
- <sup>[U12]</sup> <http://nfjsone.com>
- <sup>[U13]</sup> <http://www.pragprog.com/titles/prj>
- <sup>[U14]</sup> <http://pragmaticstudio.com/>
- <sup>[U15]</sup> <http://www.filemaker.com/developers/devcon/index.html>
- <sup>[U16]</sup> <http://pragmaticstudio.com/>
- <sup>[U17]</sup> <http://www.pragprog.com/titles/rdbcd>
- <sup>[U18]</sup> <http://www.pragprog.com/titles/dlret>
- <sup>[U19]</sup> <http://agile2009.agilealliance.org/>
- <sup>[U20]</sup> <http://www.pragprog.com/titles/rdbcd>
- <sup>[U21]</sup> <http://www.pragprog.com/titles/dlret>

- [U22] <http://www.pragprog.com/titles/bmsft>
- [U23] <http://pragmaticstudio.com/>
- [U24] <http://www.pragprog.com/titles/rdbcd>
- [U25] <http://www.pragprog.com/titles/dlret>
- [U26] <http://www.pragprog.com/titles/prj>
- [U27] <http://www.gov2summit.com/>
- [U28] <http://www.technologyreview.com/emtech/>
- [U29] <http://www.nanotech.net/>