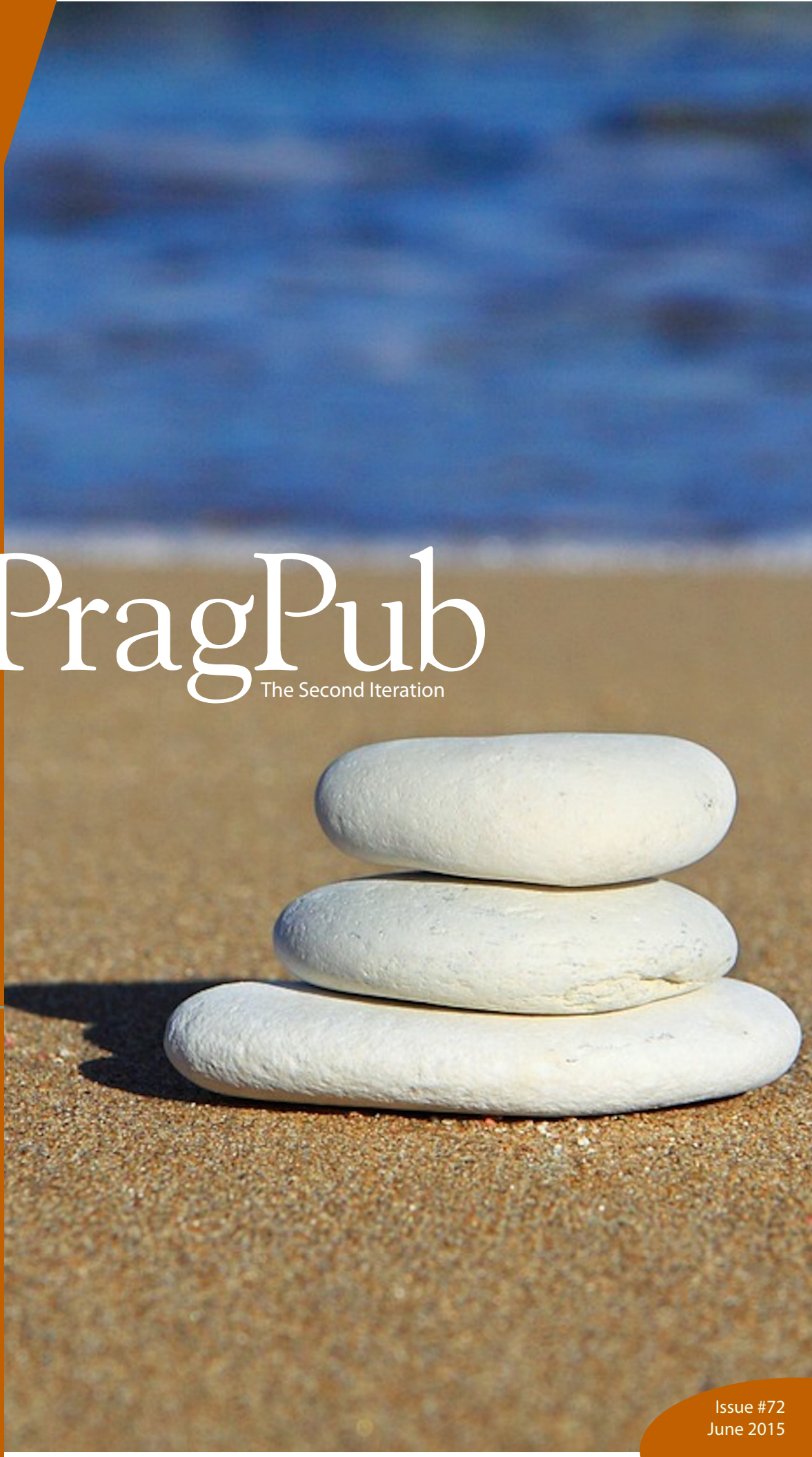
The Pragmatic Bookshelf

PragPub

The Second Iteration



IN THIS ISSUE

* Ed Burnette on Android
* Michael Bevilacqua-Linn builds a parser
* The objc.io guys on functional programming in Swift
* Douglas Engelbart, Revolutionary
* Plus Rothman and Lester on networking, Marcus Blankenship on gentle reminders, Antonio Cangiano on tech books, our monthly pub quiz, an index to the past six months of PragPub...

Issue #72
June 2015

Contents

FEATURES

Hello Again, Android! 14

by Michael Swaine (interviewer)

Ed Burnette recently wrapped up the fourth edition of Hello, Android, and we collared him for a free-ranging interview on Android, his book, and programming in general.

Paul Learns To Parse 18

by Michael Bevilacqua-Linn

In the second part of this two-part story, Michael's characters dive deeper into parsing and DSL creation.

Functional Snippets 22

by Chris Eidhof, Wouter Swierstra, and Florian Kugler

The next installment in our series on functional programming in Swift.

Unfinished Revolution 24

by Michael Swaine and Paul Freiberger

Douglas Engelbart demonstrated working models of most of the interesting user interface ideas of the past fifty years in one legendary presentation back in 1968. But his most revolutionary ideas have yet to be implemented.

DEPARTMENTS

On Tap 1

Swaine's World 3
by Michael Swaine

We follow Twitter so you don't have to.

Rothman and Lester 6
by Johanna Rothman and Andy Lester

Johanna and Andy explain how networking can help you find the right job, but only if you do it right.

New Manager's Playbook 10
by Marcus Blankenship

Marcus shows how to defuse potentially explosive situations with employees.

Antonio on Books 30
by Antonio Cangiano

Antonio looks at all the new tech books of note.

Pragmatic Bookstuff 33

Want to meet one of the Pragmatic Bookshelf authors face-to-face? Here's where they'll be in the coming months. Also, find out which are the top-selling Pragmatic Bookshelf books and what new books are coming out.

Solution to Pub Quiz 36

Our Back Pages 37

An index to issues 67 through 72 of PragPub.

Except where otherwise indicated, entire contents copyright © 2015 The Pragmatic Programmers.

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 webmaster@swaine.com. The editor is Michael Swaine (michael@pragprog.com). Visit us at http://pragprog.com for the lowdown on our books, screencasts, training, forums, and more.

ISSN: 1948-3562

On Tap

This Is the Droid You Were Looking For

You can download this issue at any time and as often as you like in any or all of our three formats: pdf[U1], mobi[U2], epub[U3]. If you subscribed through an app and want to also get the monthly file download links, just ask[U4].

Android, Python, Swift, and Revolution

It was the Woodstock of computer technology. Everybody wishes they had been there, and one suspects that a lot of people who weren't have convinced themselves that they were. In fact, there were roughly 1,000 computer professionals in the audience at that presentation in Brooks Hall in San Francisco in December of 1968. The occasion was the Joint Computer Conference, the presenter was Douglas Engelbart, and the title of the presentation was "A Research Center for Augmenting Human Intellect."

This month we profile computer revolutionary Douglas Engelbart.

And what else? Let's see...

Android is inside over a billion cell phones and other mobile devices, making it the number one platform for application developers. Ed Burnette wrote the book on Android development, and he recently released the fourth edition of Hello, Android[U5] So we collared him for a free-ranging interview on Android, his book, and programming in general.

Last year Apple introduced a new programming language, Swift. We started following it immediately, and all this year we are running a series by Chris Eidhof, Wouter Swierstra, and Florian Kugler on functional programming in Swift. This month they discuss — and demonstrate — currying, a key idea in the implementation of functional programming.

Also in this issue Michael Bevilacqua-Linn concludes his two-part series on building a parser, in which we follow the adventures of Paul and Allison as Allison teaches Paul the basics of parsing using a Python library called Parslet.

There was a time, we hear tell, when a programmer would finish school, go to work for IBM, and confidently expect to retire thirty years later with a gold watch and a comfortable pension. Things have changed a little since those halcyon days.

Today's programmer doesn't take a job for a lifetime; often she signs on for the duration of the current project, expecting to move on at its completion. Even if the job is ostensibly permanent, she's likely to decide that the fastest way to move up is to move laterally. Job Number One for a programmer today is career maintenance.

That's why we have programming career experts writing for *PragPub* every month. In this issue Johanna Rothman and Andy Lester discuss networking to find jobs: how to do it, when to do it, and why it isn't sleazy or distasteful. And what if you're managing programmers? That's a role you're likely to find yourself in at some point in your career. Marcus Blankenship has been there and done that and has some advice on how to keep little annoyances with employees from growing into chronic problems or even explosive situations.

And there's more: a puzzle, Antonio Cangiano's list of new tech books, and an index to the past six issues of *PragPub*. I hope you enjoy it.

Who Did What

photo credits: Cover: Balance [U6] by PublicDomainPictures is licensed under https://creativecommons.org/licenses/by/2.0/. Page 18: "Geek Field Trip" [U7] by Betsy Weber is licensed under https://creativecommons.org/licenses/by/2.0/. Page 22: chimney swift [U8] by Ed Schipul is licensed under https://creativecommons.org/licenses/by/2.0/ [U9]. Page 24: "1776 - Fort Moultrie" [U10] by Tom is licensed under https://creativecommons.org/licenses/by/2.0/. Page 24: Photo of Douglas Engelbart from *Fire in the Valley* courtesy of Douglas Engelbart.

editing and production: Michael Swaine

editing and research: Nancy Groth

customer support, subscriptions, submissions: webmaster@swaine.com

staff curmudgeon: John Shade (john.shade@swaine.com)

Swaine's World

Belly Up to the Bar

by Michael Swaine

"It never rains in a pub." — Traditional Irish saying

Here are a few choice bits to nosh on. Tweets. A bill of fairly interesting sites. Our Pub Quiz. Or sudoku/anagram mashup, really. That sort of thing. Cheers.

Infrequently Asked Questions

- *OH in http://irc.mozilla.org, #webdev: "Is this the proper place to complain about IE?"* — @lxt
- *I have no idea what Steampunk is except that it must be healthier than Fried Punk.* — @RickAaron
- *So Tinder is like a package manager for people?* — @iamdevloper
- *Engineer asks Quora which job offer to take. CEO replies: not ours.* — @kevinjdelaney
- *Where can I find the answers to all the security questions?* — @raganwald

The Pub Crawl

Sometimes I offer here a list of ten destinations. This month it's just one destination, but from it comes a list of ten.

- pier-to-pier network
- going to great links
- lan line
- self-phone
- motion censor goggles
- paper view
- higherarchy
- Cadillac converter
- high-bred car
- in lame man's terms
- from The Eggcorn Database [U1]

Extreme Method and Madness

- *Focus on who decides what, not who does what. #leadership @mikefulky #stareast* — @johannarothman
- *The consequences of irresponsibility should fall on the irresponsible. This may not be the whole of management, but it's a big chunk.* — @KentBeck

- *Remind me again just why was the word extreme chosen to brand the most reasonable and responsible approach to software development.* — @infinitary
- *Giving in to anger, fear, and suspicion disturbs our minds and our physical well-being, whereas trust and friendship are the basis of society.* — @DalaiLama
- *People tend to focus on the seed, when they should be focused on the soil.* — @williamlidwell
- *Maybe we should never have given it a name. Maybe we should have just done good stuff and shown what we did, explained why.* — @RonJeffries

No Business Like Show Business

- *"We don't just have our skin in the game, we have our whole body in the game." --Winklevoss on Bitcoin* — @mims
- *Demo Days, because if they called them Powerpoint Days nobody would go.* — @StartupLJackson
- *.@karaswisher just threatened to have @waltmossberg beheaded so it must be Wednesday. #CodeCon* — @inafried

The Pub Quiz

Here's this month's brain-teaser.

It's a sudoku, but with letters. Using only the nine letters that appear in the grid, fill in the empty cells so that every row, every column, and each of the nine 3x3 boxes contains all nine letters.

And it's also an anagram. Properly arranged, the nine letters in the puzzle will spell something appropriate.

Solution further on in the issue; don't peek!

		D		V			E	Y
					O	I		
C	Y							
S				I				D
R						O		
					D	V		I
D				S	V		R	
			E					
	E				I			V

Keep Your Eye on the Enterprise

- *We learned (again) today that acquisition is a "get out of scale free" card.* — @jayrosen_nyu

- *Remember kids, just because it's not a bubble doesn't mean a lot of people aren't going to take a bath.* — @mims
- *Friends don't let friends start companies to address infrequent use cases.* — @StartupLJackson
- *"I think that the future belongs to organizations that learn how to strategically delete code." (Michael Feathers)* — @JEG2

You Snake People Get Off My Lawn

- *A Chrome extension lets you replace "Millennial" with "Snake People."* — @elanazak
- *Over 1 year, the average Vim user saves 11 minutes in productivity. However, they lose 27 hours through evangelizing Vim to non-users.* — @iamdevloper
- *If you never hear from me again, it's because "it's on Great America Parkway off of 101" no longer computes. Damn kids and their Google Maps.* — @marick
- *@chadfowler It's almost like there's an entire world out there full of potential programmers, and the US is just a small slice of it. ;-)* — @manhattanmetric

Designs of the Times

- *All you web designers: list boxes don't scale well for long lists! Especially for those not born yesterday - scrolling through decades!* — @BobFrankston
- *Making accessible products isn't just the right thing to do; design them well and it's also extremely profitable: just ask Tim Cook. #axschat* — @NeilMilliken
- *"I'm the kind of person that when I get stuck in a rut, I decorate it." - @zeldman #aeabos* — @rachelandrew
- *I really like what you've done with the site. You should acknowledge your design team, Bric and Brac.* — @pragpub

The Punters

Here's the list of folks I followed this month: Rick Aaron, Rachel Andrew, Joshua Ballanco, Kent Beck, Reginald Braithwaite, Jonathan Chambers, Dalai Lama, Kevin Delaney, I Am Devloper, Bob Frankston, Ina Fried, James Edward Gray II, infinitary, Startup L. Jackson, Ron Jeffries, William Lidwell, Brian Marick, Neil Milliken, Christopher Mims, Jay Rosen, Johanna Rothman, Michael Swaine, Laura Thomson, and Elana Zak. Fair's fair. You can follow me at www.twitter.com/pragpub [U2]. Or visit my blog at swaine.com [U3].

- *@raganwald Programming is engineering when we want respect and art when we want freedom.* — @jon_cham

Rothman and Lester

Networking as Problem Solving

by Johanna Rothman and Andy Lester

Managing your career is a job in itself. Fortunately, Johanna and Andy have seen all the career mistakes people make and can steer you past the hidden rocks. This month Johanna and Andy explain how networking can help you find the right job, but only if you do it right.

Johanna: You're thinking about looking for a job. You know you need to meet people and discover the jobs out there. But how do you do it, especially if you happen to be shy and not want to "work a room"? You learn how to network so you can discover the jobs and the connections that can help you find the right job.

Andy: The first thing you have to do when it comes to networking on the job hunt is to get over the word "networking." A lot of people think it sounds sleazy. They think that "networking" is doing something vaguely dishonorable, like using your connections to get a job that you don't deserve.

The truth is that networking just means talking to people, and doing it with a purpose, which is to find job openings. Networking doesn't mean that you're going to get a job working for Bob just because you both know Alice. It means that talking to Alice lets you find out that Bob can help you along in your job search, whether because he has a job to fill, or because he knows someone else in the chain who can help you along in your search.

Johanna: There are several challenges when you look for jobs these days. There is no central repository of openings you can trust. Many hiring managers don't post the jobs they need filled, because they don't want a gazillion résumés. And too many HR systems (the tracking system black hole where your résumé goes to die) are keyword-based. You need to network just to show potential hiring managers that you exist.

Remember when we wrote, regarding *The Google Résumé* in the February 2015 issue of *PragPub* [U1], that you need to *solve problems* for hiring managers? When you network, you invite that conversation. If you think about it as a way to identify problems you might want to solve and a way to show potential employers how you solve those problems, you might see networking in a new light.

Andy: That's because networking is all about conversations, not asking for things. When you're talking to Alice, you don't want to say "Alice, I really need a job, do you know anyone who's hiring?" Instead, you're going to say something like "Alice, you know a lot of different companies. Do you know of any shops that can use a good Ruby programmer with a deep understanding of Oracle and PostgreSQL?" It's about talking about the needs of others and how you can help. Yes, the end goal is to get you a job, but the framing of the conversation makes all the difference.

Johanna: Let's look at why you might have chosen to speak with Alice. If you have thought about the kind of job you want in advance, you thought about the companies you want to target to find a job. Maybe Alice works at one of

those companies. If she does, you have a direct link to someone hiring at that company.

Maybe Alice doesn't work at one of those companies. In that case, you are looking for Alice to be your *loose connection* to find a job at one of the companies she knows about.

You don't have to network directly with hiring managers to find the job you want. You can — and should — develop loose connections [U2] with people across a wide spectrum of companies and jobs. Those loose connections are more likely to help you find a job than people in your tight circle of colleagues and friends.

Now when you have a conversation, you can talk about how you can help Alice or her colleagues. You prompt her — with your conversation — to consider you for jobs.

Andy: An excellent way to do this is to *not* even talk about your finding a job, but just to talk to someone about what the job market is like these days. "Alice, I know you work with a lot of different companies. Can you give me an idea about what the job market is like out there? What technologies are most in demand right now? Are there certain industries that are hurting for candidates?" Here you're just asking for Alice's knowledge, not pushing to get you a job.

I suggest that you not focus on getting considered for jobs, but on expanding your group of people. Chances are slim that any one person is going to be able to point you to a job, but the chances are very good that you can get pointed to another person in your chain of connections who can. Not everyone has a job to be filled, but everyone knows someone else who might be a better link.

Johanna: How do you find these links of people? You want to use your online networks and your in-person networks. I'll use LinkedIn as an example, because it shows you how people are connected to each other. You can use email lists to network. That's more difficult because you don't see the connections. And if you have been active in an email group for a while, that might be okay. You might already know the connections.

When you look on LinkedIn for target companies, you might look for specific jobs. I don't always recommend that. I suggest you look for companies you might be happy at. If you find a company that fits you, that might work better than looking for a specific job. Your job will change over time. With any luck, if you find a company whose culture fits your needs, you will be happy to change with the role.

Look for companies where people say something about the corporate culture that fits you. Are you agile? Look for a high collaboration culture. Do you focus on big data? Find a company that wants to use big data for something you believe in. Are you looking for someplace that will help you grow? Look for a company that talks about mentoring and coaching their staff. You start with what you need. Then you go on to how you solve problems for others. (See Manage Your Job Search [U3]) for more details on your target list.)

Andy: That reminds me of the other important point: You explicitly need to talk to your contact about what her needs are, and how you can help, even if it's only at the end of the conversation. This person is spending her time talking

to you, and it's crucial that you acknowledge that and make it clear you're willing to pay her back. Whenever I talk to someone about something job-related, at the end I will always say something like "Alice, I really appreciate your taking the time to talk to me about the job market. Is there anything that I can help you with?" I've found that this will usually stun the person you're talking to, but it will also make it clear just how much you appreciate their time, and you will stand out in their minds.

Johanna: You can start this conversation in any number of ways. Online, you can say something like this: "I've noticed you do such-and-such. Are you available for a 20-minute informational interview? I would like to do this in a new organization and I think you might have some information that could help me." You start by asking for help, and you use Andy's closing to offer help back.

If you are in a professional group meeting, you wander over to people. With luck, you know who those people are, but more likely you don't. Listen to their conversation for a minute or so. When you can, join the conversation. Don't bring up the issue of a job yet. Just have a professional conversation with people. You are building rapport right now.

As the conversation continues, you can bring up the fact that you are looking for a job and if anyone knows something, would they let you know? That's it.

If you are at a dinner meeting, make it your business to sit with people you don't know and ask about their work first. Understand what they do and if it fits with anything you would like to do. (See Job Search Trap: I can Network Only by Computer [U4])

Don't worry if they don't do anything you want to do. Exchange business cards or email addresses anyway. Now you can look them up on LinkedIn and see if they have connections into a company you might like to investigate. You can follow up your conversation with an email asking for an informational interview or, if you really hit it off, an introduction to someone they know.

Andy: I hope that reading these examples helps make clear what I said before about how networking is not sleazy. It's just talking to people to share knowledge. That's pretty much what we geeks do normally, right? In this case, it's just sharing knowledge about business and the job market with the underlying goal of helping find the next step in your job hunt.

Johanna: When you look for a position that fits more than just your technical skills, you open yourself up to new and targeted possibilities for networking. Consider how you work, what problems you like to solve, and how to approach people for your best networking options.

About the Authors

Johanna Rothman helps leaders solve problems and seize opportunities. She consults, speaks, and writes on managing high-technology product development. She enables managers, teams, and organizations to become more effective by applying her pragmatic approaches to the issues of project management, risk management, and people management. She writes the *Pragmatic Manager* email newsletter and two blogs on www.jrothman.com [U5].

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 spurred him to write his nontraditional book Land The Tech Job You Love [U6] on the new guidelines for tech job hunting. Andy is an active member of the open source community, and lives in the Chicago

area. He blogs at petdance.com [U7], tweets at @petdance, and can be reached by email at andy@petdance.com.

External resources referenced in this article:

[U1] http://theprosegarden.com/

[U2] http://www.jrothman.com/htp/agile-job-search/2012/04/the-power-of-a-loose-connection/

[U3] https://pragprog.com/book/jrsearch/manage-your-job-search

[U4] http://www.jrothman.com/htp/agile-job-search/2014/09/job-search-trap-i-can-network-only-by-computer/

[U5] http://www.jrothman.com

[U6] http://www.pragprog.com/refer/pragpub51/book/algh/land-the-tech-job-you-love

[U7] http://petdance.com

New Manager's Playbook

Defusing Frustrating Situations with Gentle Corrections

by Marcus Blankenship

You're a developer, but you're at a point in your career where you find yourself managing others. Marcus shares tips on how to be as good at managing as you are at your "real" job.

This month Marcus shows how to defuse potentially explosive situations with employees.

Let me tell you a story.

Bob the Coder is a great guy. He shows up on time, turns in quality work, and is reliably positive and easy to work with. You're very glad he's on your team.

But there's a problem.

You like your employees to turn in a weekly report summarizing their work at 4 p.m. each Friday. Yes, it's kind of a fussy detail, but it really helps you chart progress and a course for the coming week.

Bob, however, does not see the value in this exercise.

Every week, you notice his report missing from the stack that you review each week.

When you remind him, he quickly fills it out and turns it in, but it always requires that extra effort on your part, and it's getting old.

What should you do?

This is a tough — and incredibly common — situation. Bob's a solid worker, and this report isn't a do-or-die issue. But it's really bugging you — and you're tired of having to remind him.

How do you get a better outcome without having to nag or overreact in order to get his attention?

Common Communication Mistakes

Let's take a look at some issues that probably created this uncomfortable situation. It's true that you aren't the one forgetting the report each week, but you probably have some fears or communication habits that are getting in the way of a positive resolution.

Don't ignore the problem

The absolute best time to address a problem like this one is to catch it the first or second time it happens.

This doesn't mean you have to patrol your employees like a Border Collie or overreact to every little infraction. In fact, I recommend frequent, direct, 30-second conversations with your employees to show them that you're paying attention and that you're serious.

Often, new owner-managers are afraid of coming off as heavy-handed, so it's easy to underplay small issues. If they do address an issue like our weekly report example, they usually say something like, "Hey, Bob. I know this report isn't a No. 1 priority, but it really helps. No big deal, though! Thanks."

The manager thinks they've been friendly and clear, but they've actually given the impression that it really is no big deal, so Bob doesn't change anything.

If you don't communicate that *you* know this is an important detail for Bob to address, even if you think the rule is unnecessary, he'll continue to ignore it — and you. Then other employees who had been complying often start ignoring rules they find inconvenient, and soon you've got a mini-revolt on your hands.

Don't make the problem bigger than it is

If the problem has been festering, your resentment probably has been, too. Don't blow off steam with a laundry list of problems from the past to intimidate Bob or justify the discussion.

If you start hearing yourself using the words "always" or "never," as in, "You always do this," or "You never do that," that's a big clue that you're blowing this out of proportion.

Keep things on an even keel and you'll increase the chances for a productive resolution.

Don't assume that Bob should magically "get it"

It would be great to hop into a car and have it automatically bring us to our destination — and parallel park — without any input from the driver, but that's not the way things work, at least not until driverless cars are the norm.

If you want your team to understand your company's priorities, you've got to be willing to put your hands on the steering wheel and provide direction, speed, and attention.

Talking to your team specifically about the essential pillars of company culture, what they're doing right, and what they're doing wrong will give them the kind of support that helps inform their future behavior.

The big payoff for this clear, ongoing communication is that you're investing in their knowledge, which will make your job exponentially easier down the road.

Starting Over: Effective communication for collaborative outcomes

Hit the reset button.

Even if you have a situation similar to the Bob example, one that has been simmering for months, you can still make a course correction that will get your employee's attention and reestablish your authority in a positive, gentle way.

Gentle Correction Template

1. *Have this discussion out in the open.*

Annual reviews need to take place in private but this kind of redirection needs to be at Bob's desk or in a common area.

Keeping this communication public lets everyone else within earshot hear that this is a significant issue that applies to everyone. And when the team members see a coworker have this type of discussion with you, they'll get the

idea that it happens to everyone, which takes away the apprehension of having you stop at their desk to chat.

2. *Give Bob the benefit of the doubt.*

I can't tell you the number of times I've stewed in resentment over an infraction, only to discover later that the employee had a good reason for the mistake. Being dedicated to frequent, low-intensity conversations about work performance allowed me to approach an employee and get the real story without stirring up a lot of drama.

3. *Ask an open-ended question instead of making a demand.*

In these uncomfortable situations, it's very easy to make a statement about the behavior you'd like and walk away. You may think the matter is settled when your statement was actually misunderstood or forgotten in the cloud of details that most of us have running in our heads.

When you ask a question, you immediately have an employee's attention, and they can't just smile, nod, and return to their work.

Here's a very simple, helpful script that I often use in these situations.

"Hey, Bob. It's 5:30, and I was surprised to see that your report isn't on my desk with the rest of the team's paperwork. What happened?

This phrasing lets you know you think highly of Bob (I'm surprised you didn't do what was expected) and you're assuming that he has a reasonable explanation (What happened?). This script sets both of you up for the best possible outcome.

4. *Listen.*

Often, you'll hear that they had a dentist appointment or a meeting that kept them from doing the right thing, or some other perfectly reasonable explanation. In those cases, you can just deal with the exception, and in Bob's case, ask him to get his report turned in ASAP.

If this is an ongoing problem, you might hear something unexpected that helps you develop a solution.

5. *Don't settle for a contrite reply.*

In order to get out of this discussion quickly, Bob might say, "Oh, I just forgot." If this is a chronic problem, don't let the conversation end there. Dig deeper for the reason behind the behavior so you can get to the root of the issue.

6. *Be prepared to schedule a follow-up meeting.*

If this discussion opens up a larger topic, don't hash it out right there. Set up a meeting in a few days — in private — when you can discuss the matter fully without improvising an answer on the spot.

Here's an example. If Bob says, "I hate filling out those reports. We talk about all this at our weekly meeting. I'd rather get more coding done." Your reply should be, "That's a great point, Bob. Let's talk about that more next week. I still need for you to turn in your report right now, though."

This approach keeps the meeting on topic and lets Bob (and his eavesdropping coworkers) know that this is non-negotiable. You have also clearly and gently

communicated that you are the manager and he is the employee, and employees don't just make up rules as they go along.

In the days between this conversation and the follow-up meeting, make a point of making a short list of reasons why you have this policy. Does it provide important data points for you? Does it help determine the budget for next year?

Don't get trapped into rationalizing the policy to Bob's satisfaction (it's *your* company, after all), but do come to the follow-up meeting with this information in hand so you can give Bob some context for the rule.

7. *Establish and communicate clear consequences for continued non-compliance*.

If a gentle conversation with follow-up doesn't correct the problem, request a closed-door meeting, saying, "Bob, I've reminded you about this issue X number of times, and now we need to have a focused conversation about it."

Choose to Lead, Not React

Before you escalate your response, decide in advance what the reasonable consequences should be. Get advice from fellow agency owners, if needed. Tell your employee in clear terms what is expected and what the discipline will be if those expectations are not met. Give specifics instead of meaningless general warnings like, "You need to come in on time or things are going to get serious."

When I started managing my first team of programmers, I had no idea how to respond in situations like this. I made a lot of mistakes until I discovered the value in catching these distracting, draining situations quickly.

Let me tell you that it is a huge relief to have a clear process for a quick resolution when it comes to discipline issues. No one wants to be the bad guy or get an employee in trouble, but stepping into these discussions allows you to *be a manager*.

A process like this allows you to move past insecurity around uncomfortable conversations and become a true leader. And that paves the way for better interactions with your team, your clients, and all of your business connections.

About the Author

Nearly 20 years ago I made the leap from senior-level hacker to full-on tech lead. Practically overnight I went from writing code to being in charge of actual human beings.

Without training or guidance, I suddenly had to deliver entire products on time and under budget, hit huge company goals, and do it all with a smile on my face. I share what I've learned here [U1].

External resources referenced in this article:

[U1] http://marcusblankenship.com

Hello Again, Android!

An Interview with Ed Burnette

by Michael Swaine (interviewer)

Ed Burnette recently wrapped up the fourth edition of *Hello, Android*, and we collared him for a free-ranging interview on Android, his book, and programming in general.

Ed Burnette is a software industry veteran with more than 30 years of experience as a programmer, author, and analyst. He is a founding member and senior researcher at the SAS Advanced Server Research group.

Ed is also a technical book author. He has written technical articles and books on a variety of topics, including Google Web Toolkit: Taking the Pain out of Ajax [U1], *Eclipse IDE Pocket Guide*, and Hello, Android [U2], recently released in its fourth edition.

Ed says he devotes much of his spare time to playing video games with his two sons at their home in central North Carolina. We grabbed some of that spare time to chat with him about the current state of Android, the new edition of *Hello, Android*, and the state of software development today.

ms: Greetings, Ed. Welcome to the pages of *PragPub*. We've talked before, but I don't think I've ever asked: How did you get started programming?

eb: When I was a kid I used to hang out in the local Radio Shack. I was fascinated by the electronics and computers there, especially a new computer called the TRS-80 Model I. I got one for Christmas and started going through the manual trying all the BASIC examples. Soon I was writing my own programs.

ms: Same here. Only my Model I was a birthday present. It was a challenge figuring out what you could do in 4K of RAM. I wrote sparse matrix routines, as I recall. Nerdy stuff.

eb: Figuring out what you could possibly do with it was part of the fun. One entertaining thing I discovered was that if you placed an AM radio next to the computer you could hear sounds produced by RF interference. By running different kinds of DO loops and other statements you could get tones and string them together to make music.

ms: Ah, yes. There's a famous story about Steve Dompier using that trick to make his Altair play "A Bicycle Built for Two" and "Fool on a Hill" back in 1976. Well, I think it's famous, anyway.

eb: Which I learned about later. Anyway, I was hooked.

ms: And that led to you becoming a professional programmer? Where did you go to school?

eb: At North Carolina State. I was there (up a tree) when we won the national championship in 1983 and all the students went nuts and ran screaming out to the Brickyard. Go Wolfpack!

ms: Settle down. So after that, you went to work at SAS there in North Carolina?

eb: Right. I came over from college with a group of friends and I've been there since.

One of the nice things about the company is that you can move around and do a lot of different kinds of jobs. I've done everything from device drivers to language design, from games to business software.

For a while SAS had a games division called SouthPeak Interactive; it was in the late '90s. For several years I worked on video games. Do you remember Wild Wild West? Daily Desktop? 20,000 Leagues Under the Sea? Does anybody? Well, they might not have sold very well, but we had a heck of a lot of fun making them.

ms: Why Android? Where did that come into the mix?

eb: For a while I was managing a group that did business intelligence software for Android devices (SAS BI Mobile). Then I got drafted into a team of amazingly talented folks working on a next-generation in-memory analytics server. Now, I'm working on the cloud, and I can't wait to see where it goes.

What interested me about Android was how easy it was to get started. I knew it was going to be big, but when I began learning about it I realized there was very little introductory material. So I decided to share what I was learning in a series of blogs for ZDNet.

ms: And ultimately you wrote the first edition of *Hello, Android*.

eb: In fact, those articles became the basis of my book proposal to the Pragmatic Programmers. I love working with Dave and Andy and the gang over at PragProg. They think like programmers, always looking for ways to automate and streamline the process. Other publishers make authors work in Microsoft Word.

ms: I'm going to have to ask you to watch your language.

eb: I know, yuck. With PragProg, writing a book is just like writing a program. They really make you feel like equal partners.

ms: So it's been a positive experience? Would you recommend book writing to others?

eb: Absolutely. You have to realize, though, that writing is a work of passion. You're not going to get rich doing it, and it takes a huge amount of time and effort. But that feeling you get, when you walk into a

bookstore or browse Amazon and you see your book sitting there, it's indescribable.

ms: What has changed in the Android universe since the first edition of this book?

eb: What hasn't changed? It's incredible how fast mobile computing has evolved. When I started the first book, touch screens were a new thing, so I devoted a whole chapter to trying to make sense of touch events. It was all so primitive. Now you've got classes, libraries, and services to take care of many of the details.

ms: The book itself changes dramatically with each edition. How does the current edition address these changes in the Android universe?

eb: The first edition was short and to the point, about 200 pages. Each edition after that added more and more material and the size almost doubled. With the fourth, I wanted to start fresh, so I stripped out the non-essential parts and just focused on the key concepts that all Android developers need to know. It's back down to 200 pages again, but those pages include some new topics to reflect current developments in the platform.

ms: Such as … ?

eb: Such as Fragments, animation, and Google Play Services.

ms: Presumably with the growth of the Android developer pool there's a steady stream of new-to-Android developers. What do they need to know about Android development?

eb: Absolutely. The thing that surprised me most is how many readers are approaching Android as their first development experience. So in the book I recommend a few introductory texts on learning to program that can ease them into it.

ms: So a lot of new programmers are starting out in the Android universe. Is that a good thing, do you think?

eb: I do think Android is a good choice for starting out, because it has a low barrier to entry. Programming for Android is a personal and gratifying experience. It's also a social experience, as you can share your work with millions of others with the press of a button.

ms: What advice do you have for Android developers in general?

eb: First of all, have fun. If you're having fun writing and using your program, the users of that program will be able to tell, and will enjoy it more themselves. Second, don't be afraid to experiment. I'm inspired by a friend who wrote dozens of apps, publishing them to the Play Store

one after another and getting feedback, having a ball and making quite a bit of money along the way.

ms: Gotta ask. Thoughts on Android versus iOS?

eb: I admit, at first I was a rabid Android fanboy. But now I've mellowed out quite a bit and see the advantages and disadvantages of both platforms. I like the openness of Android, but I also appreciate the quality and polish Apple puts into their products. I've programmed for both, and I own several of both and use them every day.

People get upset about features being "stolen" from one platform by the other, but I see it as a natural sharing of good ideas. Both are improved and the user comes out the winner.

ms: Do you have any more general advice for developers today?

eb: Learn, learn, and learn. I've been doing this for 30 years and I'm always learning new things. Currently I'm teaching myself how to program in Go. Enjoy yourself and share what you learn with others.

ms: We're winding down here, and I'd like to bring it back to games. You've never lost interest in games, have you?

eb: I've always loved video games, and I have a tendency to get sucked into them. There's a year of my life I can't account for when I was addicted to World of Warcraft. Now I play mobile games almost exclusively.

ms: What do you think of the recent trend towards freemium games and apps?

eb: I like that I can try new things without paying for them. But freemium games with time limits drive me nuts. There's a game I love/hate where you get 5 lives and it takes 30 minutes for one to regenerate. Unless you pay money to continue. Argh! It's a good way to make money, but what a terrible user experience. Can't we figure out a better way to do that?

And we'll leave that question as a challenge for the reader. That's our interview with Ed Burnette, author of Hello, Android [U3].

External resources referenced in this article:

[U1] https://pragprog.com/book/ebgwt/google-web-toolkit

[U2] https://pragprog.com/book/eband4/hello-android

[U3] https://pragprog.com/book/eband4/hello-android

Paul Learns To Parse

Part Two

by Michael Bevilacqua-Linn

In the second part of this two-part story, Michael's characters dive deeper into parsing and DSL creation.

This is the second in a two-part series as we follow the adventures of Paul and Allison as Allison teaches Paul the basics of parsing using a Python library called Parslet.

In the first part, Allison and Paul wrote a program that could recognize a simple query language for a book catalog, but couldn't do anything to transform that query language into something else. In this part, we'll see how we can manipulate and transform the simple query language.

Allison: Okay Paul, the code we wrote yesterday lets us recognize the query language we need to support for the catalogue, but it doesn't let us do anything interesting with it. Eventually, we'll need to translate that query language into a query to whatever database we use to store the book data, but since the database team hasn't figured that out yet, let's take a look at a simpler transformation for now.

Allison pulled up the parser code that they'd written yesterday.

```
class GenreQueryParser < Parslet::Parser
        rule(:op) { str('&') | str('|') }
        rule(:genre) { match['a-z'].repeat(1) }
        rule(:filterChain) { filter >> (op >> filter).repeat(1) }
        rule(:filter) { genre | str('(') >> filterChain >> str(')') }
        root(:filter)
end
```

Paul: The parser code we wrote yesterday would throw an error on invalid input, but on valid input it just returned the original string that got passed into it.

Allison: Right! So the first step here is to have the parser start capturing the information we'll need for further processing rather than just returning the original input. With Parslet, we can do this using the `as` method. The method can be attached to a part of our grammar that matches some input, and it takes a single symbol to use as a name for that input.

Then, Parslet will return a map where the key is the name and the value is the snippet of input. Let's modify our grammar so that we capture our operations and genres.

```
rule(:op) { (str('&') | str('|')).as(:op) }
rule(:genre) { match['a-z'].repeat(1).as(:genre) }
```

Now if we parse an `op` or a `genre`, we see that Parslet turns the original string into a map, like this:

```
irb(main):045:0> parser.op.parse("&")
=> {:op=>"&"@0}
irb(main):046:0> parser.genre.parse("foo")
=> {:genre=>"foo"@0}
```

Paul: So far, so good, but what about when we've got more than a single op or genre?

Allison: For that we can use as to name our filter chains, like this:

```
rule(:filter) {
  genre | str('(') >> filterChain.as(:filterChain) >> str(')')
}
```

Now, Parslet will return a filter chain as an array of genres and ops. Parslet does some magic here to merge together maps, based on the structure of the grammar. This is what things look like if we parse a simple filter chain:

```
irb(main):049:0> parser.filterChain.parse("foo&bar&baz")
=> [{:genre=>"foo"@0},
    {:op=>"&"@3, :genre=>"bar"@4},
    {:op=>"&"@7, :genre=>"baz"@8}]
```

Putting it all together, our parser now looks like this:

```
class GenreQueryParser < Parslet::Parser
        rule(:op) { (str('&') | str('|')).as(:op) }
        rule(:genre) { match['a-z'].repeat(1).as(:genre) }
        rule(:filterChain) { filter >> (op >> filter).repeat(1) }
        rule(:filter) {
          genre | str('(') >> filterChain.as(:filterChain) >> str(')')
        }
        root(:filter)
end
```

Now, if we parse one of our example queries, we see that it returns a map containing the query broken down into its constituent parts.

```
irb(main):050:0> parser.parse("((scifi&fantasy)|(horror&mystery))")
=> {:filterChain=>[
             {:filterChain=>[
           {:genre=>"scifi"@2}, {:op=>"&"@7, :genre=>"fantasy"@8}]},
           {:op=>"|"@16, :filterChain=>[{:genre=>"horror"@18},
           {:op=>"&"@24, :genre=>"mystery"@25}]}]}
```

Paul: Whoa, that looks complicated!

Allison: It is, but luckily Parslet has some features that help us transform from that mess into something more reasonable, the aptly named Transform. Just like a Parser takes a string and transforms it into the map we saw containing the string broken down into its parts, a Transform takes output from a Parser and helps us further manipulate it into something useful.

Let's use a Transform to translate from our parser output into a different query language format. We'll translate the ampersand and pipe in the strings and and or, and we'll translate the raw genres into the string genre, followed by an equals sign, followed by the genre name in single quotes.

We create a Transform similar to the way we'd create a parser, but we subclass the Transform class instead. Instead of matching on bits of the original string, we match on pieces of the tree that the parser produces.

Here's a rule that matches a map with a single genre, like we'd produce from a query with only one genre in it.

```
class Transformer < Parslet::Transform
        rule(:genre => simple(:genre)) { "genre='#{genre}'" }
end
```

The left hand side of the rule matches a part of the tree produced by the parser. This rule will match any map that consists of one genre, it'll then pull out the text of the genre and store a reference to it in a variable also named genre, which can be accessed by the right hand side.

Notice how we used the simple function as part of the match?

Paul: Uh-huh. I see that.

Allison: That function indicates to the transform that the value of the genre key must be a simple expression, like a string or single object, and not a compound one like a map or an array.

The right hand side, the part between the curly braces, is called the action. This part does the transformation. Here, we're returning a string with our new genre format. Let's take a look at it in action on our simplest possible input.

```
irb(main):056:0> parsed = parser.parse("foo")
=> {:genre=>"foo"@0}
irb(main):057:0> transformer.apply(parsed)
=> "genre='foo'"
```

Now we can start building out the more complex cases. Here's what the parsed version of our scifi and fantasy query looks like.

```
irb(main):058:0> parsed = parser.parse("(scifi&fantasy)")
=> {:filterChain=>[{:genre=>"scifi"@1}, {:op=>"&"@6, :genre=>"fantasy"@7}]}
```

We've already got code that'll handle a map with a single genre, now we need to add in a rule to match an op and a genre, like this snippet:

```
{:op=>"&"@6, :genre=>"fantasy"@7}
```

This looks a lot like the first rule we wrote, except we need to match both an op and a genre in the same map. When we do, the action will translate the ampersand or pipe and genre into a new string.

```
rule(:op => simple(:op), :genre => simple(:genre)) {
        sqlOp = if op == "&"; "and" else "or" end
        "#{sqlOp} genre='#{genre}'"
}
```

Next up we'll need to translate a filter chain into a new parenthesized expression. To do that, we need the sequence function, just like simple will match a simple expression, sequence will match an array. Here, we're matching all the parts of a filter chain and storing a reference to them in the filter parts variable, then joining them together inside in a new string and surrounding it in parentheses.

```
rule(:filterChain => sequence(:filterParts)) {
        "(#{filterParts.join(' ')})"
}
```

Finally, we need to handle an op and filter chain in the same map, which looks like so.

```
rule(:op => simple(:op), :filterChain => sequence(:filterParts)) {
        sqlOp = if op == "&"; "and" else "or" end
        "#{sqlOp} (#{filterParts.join(' ')})"
}
```

Putting everything together looks like this.

```
class Transformer < Parslet::Transform
        rule(:genre => simple(:genre)) { "genre='#{genre}'" }
        rule(:op => simple(:op), :genre => simple(:genre)) {
      sqlOp = if op == "&"; "and" else "or" end
      "#{sqlOp} genre='#{genre}'"
        }
        rule(:filterChain => sequence(:filterParts)) {
      "(#{filterParts.join(' ')})"
        }
        rule(:op => simple(:op), :filterChain => sequence(:filterParts)) {
      sqlOp = if op == "&"; "and" else "or" end
      "#{sqlOp} (#{filterParts.join(' ')})"
        }
end
```

Now we can run it on some of our examples and see the output in our translated language.

```
irb(main):062:0> transformer.apply(parser.parse
        ("(((scifi|horror)&fantasy)|mystery)"))
=> "(((genre='scifi' or genre='horror') and genre='fantasy')
        or genre='mystery')"
```

And that's it! Now when the database team figures out what they want to do, you should be able to take this example and use it to figure out how to translate into whatever form you'll need.

Paul: Thanks, Allison!

About the Author

Michael Bevilacqua-Linn has been programming computers ever since he dragged an Apple IIGS that his parents got for opening a bank account into his fifth grade class to explain loops and variables to a bunch of pre-teenagers. He currently works for Comcast, where he builds distributed systems that power infrastructure for their next generation services, and wrote Functional Programming Patterns [U1]. He tweets occasionally at @NovusTiro [U2].

External resources referenced in this article:

[U1] http://pragprog.com/refer/pragpub51/book/mbfpp/

[U2] http://www.twitter.com/NovusTiro

Functional Snippets

Currying

by Chris Eidhof, Wouter Swierstra, and Florian Kugler

Here's the next installment in our series designed to help you get a grip on functional programming in Swift.

Swift opens up a whole new world of programming. To quote Swift's creator, Chris Lattner: " 'Objective-C without the C' implies something subtractive, but Swift dramatically expands the design space through the introduction of generics and functional programming concepts."

In this series, we'll use small snippets of Swift code to explore this new world of programming, as well as demonstrate how Swift supports the functional paradigm.

Currying

Turning a function that takes multiple arguments into a series of functions that each take one argument is called *currying*. For example, take this function with three arguments:

```
func login(email: String, pw: String, success: Bool -> ())
```

We can turn it into a series of functions where the arguments can be applied one after another using Swift's special syntax for curried functions:

```
func curriedLogin1(email: String)(pw: String)(success: Bool -> ())
```

More interestingly, we can also write a generic function that does the currying for us. The following one works on functions with three arguments, but we could write variants for any number of arguments:

```
func curry<A, B, C, R>(f: (A, B, C) -> R) -> A -> B -> C -> R {
    return { a in { b in { c in f(a, b, c) } } }
}

let curriedLogin2 = curry(login)
```

Now we can apply the arguments one by one:

```
let f = curriedLogin2("foo")("bar")
f { println("success: \($0)") }
```

You're actually using this pattern all the time without even noticing: instance methods in Swift are curried functions that take the instance as first argument.

In next month's snippet we'll show you a cool application of this.

About the Authors

Along with Daniel Eggert, Chris Eidhof and Florian Kugler created objc.io [U1], a periodical about best practices and advanced techniques for iOS and OS X development. They also wrote the book *Functional Programming in Swift* [U2], in which they explain the concepts behind functional programming and how Swift makes it easy to leverage them in a pragmatic way, in order to write clearer and more expressive code.

External resources referenced in this article:

[U1] http://www.objc.io/

[U2] http://www.objc.io/books/

Unfinished Revolution

Douglas Engelbart, Revolutionary

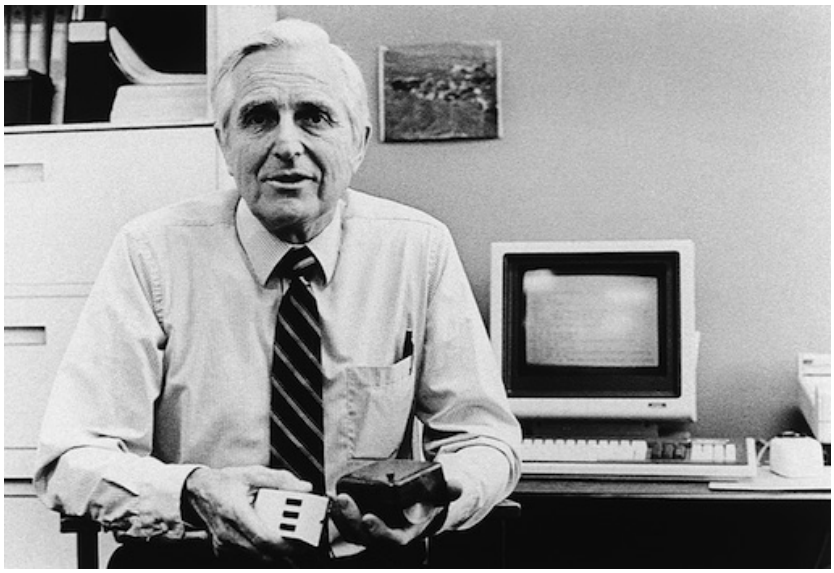
by Michael Swaine and Paul Freiberger

I don't know what Silicon Valley will do when it runs out of Doug's ideas. –Alan Kay

Portions of this article first appeared in an early edition of Fire in the Valley [U1].

It was the Woodstock of computer technology. Everybody wishes they had been there, and one suspects that a lot of people who weren't have convinced themselves that they were. In fact, there were roughly 1,000 computer professionals in the audience at that presentation in Brooks Hall in San Francisco in December of 1968. The occasion was the Joint Computer Conference, the presenter was Douglas Engelbart, and the title of the presentation was "A Research Center for Augmenting Human Intellect."

Engelbart was presenting the fruits of the past four years of work he and his team at the Augmentation Research Center (ARC) had been engaged in. ARC was part of Stanford Research Institute (SRI) and Engelbart was its founder and head.



The demo stunned the audience and became instantly legendary. People started calling it "the mother of all demos." You can watch the full video of the presentation here [U2].

To appreciate what Engelbart and his team did, and why the audience was so blown away by it, it helps to have a little context. Because the world of computing was very different in 1968.

For one thing, there was no World Wide Web. In fact, there was no Internet.

The Internet grew out of ARPAnet, the first network to implement TCP/IP, which became the technical base of the Internet. "The first two nodes of what would become the ARPANET were interconnected between Leonard Kleinrock's Network Measurement Center at the UCLA's School of

Engineering and Applied Science and Douglas Engelbart's NLS system at SRI International (SRI) in Menlo Park, California, on 29 October 1969." (Wikipedia [U3])

Right, 1969: a year *after* Engelbart's presentation.

Also, there was no email.

There were messaging tools, but these typically required sender and receiver to be using the same tool, and often to be logged into the same time-sharing system. The first email as we know it was was sent in 1971 by Ray Tomlinson. He also came up with the @ sign to indicate an address.

Three years after Engelbart's presentation.

So what did Engelbart present that was so impressive? It went something like this:

The Demo

On stage, down left in front of a large screen, Douglas Engelbart sits at an elaborate console, a microphone/earphone headset trailing a wire down his back. As he begins speaking, his face appears on the screen.

He starts by demonstrating what looks like modern word processing: editing, cutting and pasting. Uncharacteristically for the times, it is an emulation of paper: black text on a white background. It looks very modern. Now he invokes different views of the same document: this is too much information, collapse it — just show me the first line of each statement. Now make it a list, automatically numbering items. Oh, it's a shopping list, so categorize the items, give me produce and cereals and meats as sublists. Now just show me the top level, or two levels deep. He show crossreferencing. He clicks to move items around in the list.

Clicks? Yes, using a device Engelbart invented and called a "mouse." He uses it in conjunction with a one-hand keypad he also invented. He also has a standard keyboard, but hardly needs it.

Now he clicks on an underlined word to jump to a different file. To a specific *place* in that file. To a specific *view* of that place in that file. Hypertext. Sophisticated hypertext.

He moves on to collaboration, bringing in, in live online demo, a co-worker at another location. Both have microphones, cameras: they're video conferencing. They demonstrate collaborative work on a file: the co-worker views Engelbart's text on his own screen, and controls it with his own cursor, and we see it on the screen. "We're both looking at the same display," Engelbart explains. "We can talk to each other and point and maybe later, I can hand you the chalk on this black board like saying, 'Here, you control it.'

"I say, 'Now computer, do the automatic switching,' and it will bring in a camera picture from the camera mounted on his console such as the camera mounted on mine is. Hi, Bill. That's great. Now we're connected [via] audio, you can see my work, you can point at it, and I can see your face, we can talk. So let's do some collaborating."

They collaboratively examine the code of the system using the system itself. The system is implemented through a bunch of specialized little languages created using a compiler compiler. They run through some applications. User documentation. A hyperlinked glossary: click on a term to get its definition.

ARC has six computers on this system now, this thing they call NLS (for oN-Line System), with six more due to come online shortly. Engelbart and his team have been doing their work using the system itself. And that leads to his real interest.

Because, as he indicated at the beginning of his talk, the interesting part for him is not the nifty technology. It's what you do with it. What it enables. This system is "a vehicle for helping humans to operate within the domain of complex information structures."

What does he mean by complex structures? "I'm emphasizing structure because... although the content represents your concepts, there's a structure relationship between that content entities that should represent the relationship between the concepts of human thought. Alright, we know we can't do that very well with linear text, [but] inside the computer, we can represent that quite well, in fact we can represent information structures in a computer that will generally be far too complex for you to study directly." But a tool like NLS can help you navigate and think about such complex structures.

In complex systems, he explains, you can't see solutions immediately. "We need to do it evolutionary-wise because we can't be analytic enough about it at any one point, to decide what best our next thing should be. We can only decide from here, as well as we can, analyze it, where we can invest our next resources to get the most return that will increase of the effectiveness of the system we have. He calls it "boot-strapping."

And there lies the contribution he wants to make. His lofty goal is to "develop a system-oriented discipline for designing the means by which greater effectiveness is achieved." He doesn't just want to build something that makes people more efficient. He wants to build the tools that can build the tools that make people more effective. Because, he warns, our problems are getting more complex and more urgent.

The audience in Brooks Hall gave him a standing ovation.

Unfortunately, that was the high point. Funding for ARC began to dry up. Douglas's team broke up. Bob Taylorhired many of Engelbart's developers for Xerox's new research center, Xerox Palo Alto Research Center, or PARC, in 1970. In the ensuing decade, Xerox PARC implemented some of Douglas Engelbart's ideas, particularly in terms of networked computers, windowing, and his patented device, the mouse, on Xerox computers that the general public never saw. Xerox PARC was where the great ideas about personal computer technology got turned into actual metal and plastic sitting on people's desks. Mostly Xerox PARC engineers' desks.

The bulk of these particular technologies eventually did reach the market, from Xerox, from Apple, and later from Microsoft and other companies. Engelbart remains the source of some of the most powerful ideas in user interface design in computers today. In addition to the mouse, contributions to e-mail, and the first fully integrated two-way computer/video teleconference,

he pioneered multiple-window computer displays, hypermedia, groupware, and electronic publishing. But the revolution he sought has not arrived.

Recognition

It's not that he hasn't been honored for his achievements.

In 1997, Engelbart won the $500,000 Lemelson-MIT Prize, the world's largest cash award for American invention and innovation. The prize is given upon the recommendation of three review panels of leading experts representing scientific, engineering, and medical disciplines in academia and industry.

In 1998, his peers and fans honored him in "Engelbart's Unfinished Revolution," a day-long Festschrift commemorating the thirtieth anniversary of "The Mother of All Demos." Industry leaders and fans, 1500 hundred of them, came to praise Engelbart and, just as happened at that earlier presentation, the entire audience gave him a standing ovation.

He has been honored. But he wasn't understood. He wasn't listened to. In the last decades of his life, he wasn't funded.

Ted Nelson was at the Festschrift. He said, "This prophet-without-honor thing is all completely true. The lack of appreciation of Doug is like you stand next to the Empire State Building and you don't know how tall it is. You just know it's taller than you and a lot of people resent that."

Although moved by the show of support, Engelbart couldn't help but reflect that he had not yet solved the problem he set for himself back in 1951: augmenting human intellect to deal with the complexity of life.

His ideas

Back in 1977, when the Xerox PARC researchers began honing some of his ideas, Engelbart kept working on the ideas behind these ideas. His computer devices were only part of the plan; the true plan, more revolutionary than his inventions, was about developing new systems to expand mental scope. These systems didn't have to be technological. The key to getting more intelligence out of any organization, Engelbart believed, was sharing information and techniques.

Engelbart's real plan was motivated by an insight about complexity and levels of tools. Here's a rough sketch of the idea: The problems facing humanity getting more complicated and more urgent. But the tools and techniques we have used in the past to solve problems are inappropriate to present-day problems. So we are now dependent on a higher-level class of tools: methods to improve our problem-solving tools and techniques. But beyond that, we either have passed or will soon pass the point where our incremental, ad-hoc methods for improving our tools and techniques are up to the task. We need to move to the next level. We need tools to improve how we improve our tools.

It comes down to types of information.

Engelbart drew distinctions among kinds of information, based on types of work.

A-work is the primary mission of an organization, like building cars. It requires knowledge that is widespread and common in an industry.

B-work includes techniques for improving A-work. The problems here are the same for all companies, but their solutions give a company a competitive edge, so firms usually protect them as proprietary information.

C-work is knowledge about improving the improvement process itself. One example is systems for responding to customer complaints. Engelbart believed that it was crucial to share this kind of knowledge. Because it is widely useful and because it is high-level knowledge, its spread can enormously benefit society as a whole.

Having identified his target, he moved on to identifying methods for sharing it. Actually, he began inventing methods for sharing it. That was the motivation behind all his inventions. E-mail, hypertext documents, and shared-screen teleconferencing were remarkable breakthroughs, but to Engelbart they were merely tools for achieving the real goal: sharing C-work knowledge.

Much of what's gone on in the personal computer revolution, Engelbart thought, doesn't do much to advance that cause. The concept of groupware, for example, is really just about sharing a document. It doesn't get into how people can work together within a complex hypertext-linked repository of documents. The personal computer has delivered a lot of power to the individual, but hasn't done very much to help people work together in new ways.

In Engelbart's view, the tragedy of the personal computer revolution was this decade-long detour away from the crucial goal of connecting people in new, more empowering ways. He had hoped to see the computer change the very nature of collaborative work, with work happening in shared virtual spaces like electronic whiteboards that everyone could write on and annotate in parallel. But everywhere he looked he saw office workers grinding away in the relative seclusion of their offices or cubicles, limiting their information sharing to e-mailed messages and static files.

Today, though, the trend has shifted toward sharing information and technology. The Internet and the World Wide Web are bringing people together in unexpected ways, hinting at new modes of collaboration, new models of wall-less libraries open to all. Opensource software development is both a model of how people can work together productively and a test bed for collaborative techniques.

But will they be the techniques Engelbart thought we need? He believed that corporations in the same market niche should work together on pilot projects in which high-performance teams can experiment with ways to get bootstrapping processes started. Is this naive? Did Engelbart underestimate the pressures to compete?

Possibly so, but his principles have an interesting resonance with the open-source software movement. When proponents of the free sharing of software ideas and code talk about how quickly problems can be fixed in this open atmosphere and how rapidly technology can advance when anyone is free to contribute to the most promising projects, what are they talking about but bootstrapping?

We may get there yet. But the goal that Engelbart set for himself nearly 50 years ago always eluded him. There were many successes, a few high points of acknowledgment, off-again-on-again funding, and much disappointment. His revolution remains unfinished.

About the Authors

Michael Swaine is an author, editor, and publisher. His writing has appeared in publications ranging from *The Farmer's Almanac* and *The Whole Earth Software Catalog* and *Encyclopedia Britannica* to an Italian comic strip, but he is best known for his work in technology publishing. He is an in-demand columnist for computer publications, having appeared regularly in a dozen magazines. He has launched and managed a number of magazines, and was editor-in-chief of the legendary magazine for software developers, *Dr. Dobb's Journal*. Currently he edits books for programmers and is the editor of *PragPub*.

Paul Freiberger is an award-winning author, consultant, and business executive. His books include *Fuzzy Logic*, which won the *Los Angeles Times* Book Prize. He has been a radio and print journalist. He produced reports for NPR programs, including All Things Considered and Morning Edition. He is a former business technology columnist for the *San Jose Mercury News* and a former reporter for the *San Francisco Examiner*, where he covered Silicon Valley business for seven years. He was also the West Coast head of corporate communications at McKinsey & Company. He was a member of the research staff for Interval Research, the think tank and research lab created by Paul Allen, where he supervised teams of engineers and consultants in developing new technologies and products. He is President of Shimmering Resumes [U4], a career counseling business, which has helped thousands of job seekers to improve their careers.

Michael and Paul wrote Fire in the Valley: the Birth and Death of the Personal Computer [U5].

External resources referenced in this article:

[U1] https://pragprog.com/book/fsfire/fire-in-the-valley

[U2] https://archive.org/details/XD300-23_68HighlightsAResearchCntAugHumanIntellect

[U3] http://en.wikipedia.org/wiki/Internet#History

[U4] http://www.shimmeringresumes.com/

[U5] https://pragprog.com/book/fsfire/fire-in-the-valley

Antonio on Books

Thirty New Books

by Antonio Cangiano

Antonio looks at all the new tech books of note.

Antonio Cangiano is the author of the excellent Technical Blogging [U1] and you can subscribe to his reports on new books in technology and other fields here [U2].

Teaching your kids to code might not guarantee them a career in the future, but it certainly can't hurt. Whether you feel that coding will be the new literacy in the decades to come or not, I would argue that learning to code will provide them with an invaluable asset. Namely, the ability to approach problems in an analytical manner.

It will also provide them with the tools needed to analyze information and automate workflows, almost regardless of what career they ultimately end up purusing. As a parent, you could do a lot worse than teaching your children to code, especially if you make it a fun, bonding experience.

Here is where this month's pick comes in. *Teach Your Kids to Code: A Parent-Friendly Guide to Python Programming* by Dr. Payne is a parent's (and teacher's) guide to how to go about teaching kids to program, specifically in Python, which is an excellent language from a didactical perspective.

As the title implies, this is more of a guide for parents (especially those with some programming experience under their belt) than a book for kids. And it should be noted that due to the somewhat mathematical nature of the examples, the material covered within will suit high schoolers best of all.

Our Staff Pick: *Teach Your Kids to Code: A Parent-Friendly Guide to Python Programming* [U3] • By *Bryson Payne* • ISBN: *1593276141* • Publisher: *No Starch Press* • Publication date: *May 1, 2015* • Binding: *Paperback* • Estimated price: *$16.93*

DevOps: A Software Architect's Perspective [U4] • By *Len Bass, Ingo Weber, Liming Zhu* • ISBN: *0134049845* • Publisher: *Addison-Wesley Professional* • Publication date: *May 28, 2015* • Binding: *Hardcover* • Estimated price: *$32.59*

Marketing Data Science: Modeling Techniques in Predictive Analytics with R and Python [U5] • By *Thomas W. Miller* • ISBN: *0133886557* • Publisher: *Pearson FT Press* • Publication date: *May 22, 2015* • Binding: *Hardcover* • Estimated price: *$58.35*

Automate the Boring Stuff with Python: Practical Programming for Total Beginners [U6] • By *Albert Sweigart* • ISBN: *1593275994* • Publisher: *No Starch Press* • Publication date: *May 1, 2015* • Binding: *Paperback* • Estimated price: *$19.29*

Big Data: Principles and best practices of scalable realtime data systems [U7] • By *Nathan Marz, James Warren* • ISBN: *1617290343* • Publisher: *Manning Publications* • Publication date: *May 10, 2015* • Binding: *Paperback* • Estimated price: *$27.00*

Xcode 6 Start to Finish: iOS and OS X Development (2nd Edition) (Developer's Library) [U8] • By *Fritz Anderson* • ISBN: *0134052773* • Publisher:

Addison-Wesley Professional • Publication date: *May 16, 2015* • Binding: *Paperback* • Estimated price: *$26.57*

Ruby on Rails Tutorial: Learn Web Development with Rails (3rd Edition) (Addison-Wesley Professional Ruby Series) [U9] • By *Michael Hartl* • ISBN: *0134077709* • Publisher: *Addison-Wesley Professional* • Publication date: *May 9, 2015* • Binding: *Paperback* • Estimated price: *$30.14*

Professional AngularJS [U10] • By *Valeri Karpov, Diego Netto* • ISBN: *1118832078* • Publisher: *Wrox* • Publication date: *May 4, 2015* • Binding: *Paperback* • Estimated price: *$29.03*

Make: JavaScript Robotics: Building NodeBots with Johnny-Five, Raspberry Pi, Arduino, and BeagleBone [U11] • By *Backstop Media, Rick Waldron* • ISBN: *1457186950* • Publisher: *Maker Media, Inc* • Publication date: *May 8, 2015* • Binding: *Paperback* • Estimated price: *$16.70*

Patterns, Principles, and Practices of Domain-Driven Design [U12] • By *Scott Millett, Nick Tune* • ISBN: *1118714709* • Publisher: *Wrox* • Publication date: *May 4, 2015* • Binding: *Paperback* • Estimated price: *$37.48*

Learn Android Studio: Build Android Apps Quickly and Effectively [U13] • By *Adam Gerber, Clifton Craig* • ISBN: *1430266015* • Publisher: *Apress* • Publication date: *May 11, 2015* • Binding: *Paperback* • Estimated price: *$40.03*

Hello, Android: Introducing Google's Mobile Development Platform [U14] • By *Ed Burnette* • ISBN: *1680500376* • Publisher: *Pragmatic Bookshelf* • Publication date: *May 14, 2015* • Binding: *Paperback* • Estimated price: *$21.16*

Crafting the InfoSec Playbook: Security Monitoring and Incident Response Master Plan [U15] • By *Jeff Bollinger, Brandon Enright, Matthew Valites* • ISBN: *1491949406* • Publisher: *O'Reilly Media* • Publication date: *May 24, 2015* • Binding: *Paperback* • Estimated price: *$33.21*

Hands-On Mobile App Testing: A Guide for Mobile Testers and Anyone Involved in the Mobile App Business [U16] • By *Daniel Knott* • ISBN: *0134191714* • Publisher: *Addison-Wesley Professional* • Publication date: *May 28, 2015* • Binding: *Paperback* • Estimated price: *$32.59*

The Gourmet iOS Developer's Cookbook: Even More Recipes for Better iOS App Development [U17] • By *Erica Sadun* • ISBN: *0134086228* • Publisher: *Addison-Wesley Professional* • Publication date: *May 15, 2015* • Binding: *Paperback* • Estimated price: *$22.76*

Securing SQL Server, Third Edition: Protecting Your Database from Attackers [U18] • By *Denny Cherry* • ISBN: *0128012757* • Publisher: *Syngress* • Publication date: *May 7, 2015* • Binding: *Paperback* • Estimated price: *$38.96*

Enterprise Cybersecurity: How to Build a Successful Cyberdefense Program Against Advanced Threats [U19] • By *Scott E. Donaldson, Stanley G. Siegel, Chris K. Williams, Abdul Aslam* • ISBN: *1430260823* • Publisher: *Apress* • Publication date: *May 20, 2015* • Binding: *Paperback* • Estimated price: *$59.70*

Scratch For Kids For Dummies [U20] • By *Derek Breen* • ISBN: *1119014875* • Publisher: *For Dummies* • Publication date: *May 4, 2015* • Binding: *Paperback* • Estimated price: *$16.99*

Coding with JavaScript For Dummies [U21] • By *Chris Minnick, Eva Holland* • ISBN: *1119056071* • Publisher: *For Dummies* • Publication date: *May 26, 2015* • Binding: *Paperback* • Estimated price: *$20.66*

Hello Web App [U22] • By *Tracy Osborn* • ISBN: *0986365912* • Publisher: *Tracy Osborn* • Publication date: *May 4, 2015* • Binding: *Paperback* • Estimated price: *$19.95*

Adventures in Arduino [U23] • By *Becky Stewart* • ISBN: *1118948475* • Publisher: *Wiley* • Publication date: *May 4, 2015* • Binding: *Paperback* • Estimated price: *$13.78*

NumPy Cookbook - Second Edition [U24] • By *Ivan Idris* • ISBN: *1784390941* • Publisher: *Packt Publishing - ebooks Account* • Publication date: *May 29, 2015* • Binding: *Paperback* • Estimated price: *$44.99*

Inside the Machine: An Illustrated Introduction to Microprocessors and Computer Architecture [U25] • By *Jon Stokes* • ISBN: *1593276680* • Publisher: *No Starch Press* • Publication date: *May 25, 2015* • Binding: *Paperback* • Estimated price: *$36.67*

A First Course in Network Theory [U26] • By *Ernesto Estrada, Philip Knight* • ISBN: *0198726465* • Publisher: *Oxford University Press* • Publication date: *May 19, 2015* • Binding: *Paperback* • Estimated price: *$40.02*

Big Data and The Internet of Things: Enterprise Information Architecture for A New Age [U27] • By *Robert Stackowiak, Art Licht, Venu Mantha, Louis Nagode* • ISBN: *1484209877* • Publisher: *Apress* • Publication date: *May 6, 2015* • Binding: *Paperback* • Estimated price: *$35.74*

The Hacker's Guide to Python [U28] • By *Julien Danjou* • ISBN: *1304819248* • Publisher: *Lulu.com* • Publication date: *May 4, 2015* • Binding: *Paperback* • Estimated price: *$45.49*

Apache Oozie: The Workflow Scheduler for Hadoop [U29] • By *Mohammad Kamrul Islam, Aravind Srinivasan* • ISBN: *1449369928* • Publisher: *O'Reilly Media* • Publication date: *May 24, 2015* • Binding: *Paperback* • Estimated price: *$32.59*

Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects [U30] • By *S. P. T. Krishnan, Jose Ugia Gonzalez* • ISBN: *1484210050* • Publisher: *Apress* • Publication date: *May 22, 2015* • Binding: *Paperback* • Estimated price: *$39.32*

Level Up Your Web Apps With Go [U31] • By *Mal Curtis* • ISBN: *0992461294* • Publisher: *SitePoint* • Publication date: *May 7, 2015* • Binding: *Paperback* • Estimated price: *$22.63*

C++ Recipes: A Problem-Solution Approach [U32] • By *Bruce Sutherland* • ISBN: *1484201582* • Publisher: *Apress* • Publication date: *May 15, 2015* • Binding: *Paperback* • Estimated price: *$36.43*

Pragmatic Bookstuff

Here's what's up with the Pragmatic Bookshelf and its authors.

Here's what's happening at Pragmatic Bookshelf.

What's Hot

1	NEW	Real-World Kanban
2	8	Text Processing with Ruby
3	NEW	Pragmatic Scala
4	7	Programming Elixir
5	1	Clojure Applied
6	10	Agile Web Development with Rails 4
7	NEW	Hello, Android
8	NEW	Programming Ruby 1.9 & 2.0
9	NEW	XSLT Jumpstarter
10	3	iOS 8 SDK Development
11	2	Ruby Performance Optimization

Who's Where When

And here's what the authors are up to:

2015-04-29 **Estimates and Continuous Delivery**
Seb Rose (author of The Cucumber for Java Book [U1])
CoDeOSL, Oslo [U2]

2015-05-07 **Extensions Programming Workshop**
Chris Adamson (author of iOS 8 SDK Development [U3])
CocoaConf Portland [U4]

2015-05-07 **Talk: Building Learning into Team Life**
Rachel Davies (author of Agile Coaching [U5])
Agile Manchester, UK [U6]

2015-05-08 **What Makes You a Great Test Leader?**
Johanna Rothman (author of Behind Closed Doors [U7] , Manage It! [U8] , Manage Your Project Portfolio [U9] , Hiring Geeks That Fit [U10] , and Manage Your Job Search [U11])
Testing & Quality Leadership Summit, Orlando, FL [U12]

2015-05-08 **Core Image: The Most Fun API You're Not Using**
Chris Adamson
CocoaConf Portland [U13]

2015-05-09 **Revenge of the 80s: Cut/Copy/Paste, Undo/Redo, and More Big Hits**
Chris Adamson
CocoaConf Portland [U14]

2015-05-14 **#NoEstimates does not mean no estimates**
Seb Rose
SDD, London [U15]

2015-05-15 **Workshop on Coaching teams through change**
Rachel Davies
The Big DO, Oxford, UK [U16]

2015-06-06 **Arcade Arduino - Learn how to turn an Arduino into a video game console using an OLED display and a joystick shield.**
Maik Schmidt (author of Raspberry Pi: A Quick-Start Guide (2nd edition) [U17] and Arduino: A Quick-Start Guide, Second Edition [U18])
Maker Faire Hannover (Germany) [U19]

2015-06-11 **Making Apple Devices Work Together Most of us have more than one iOS device. Whether it's an iPhone and iPad, iPhone and Apple Watch, or iPad and MacBook, these devices no longer live in isolation.**
Jeffrey R. Kelley (author of Developing for Apple Watch [U20])
AltConf [U21]

2015-06-15 **Talks**
Andrew Hunt (author of Practices of an Agile Developer [U22] , Pragmatic Thinking and Learning [U23] , and Learn to Program with Minecraft Plugins (2nd edition) [U24])
NDC Oslo [U25]

2015-06-18 **Agile 1's Game**
Seb Rose
NDC, Oslo [U26]

2015-06-25 **Beyond Code: Repository Mining with Clojure**
Adam Tornhill (author of Your Code as a Crime Scene [U27])
EuroClojure 2015, Barcelona, Spain [U28]

2015-06-25 **You Got Your Idris in My C++! A First Look at Denotational Design**
Ian Dees (author of Cucumber Recipes [U29] and Seven More Languages in Seven Weeks [U30])
Open Source Bridge, Portland [U31]

2015-06-29 **Recycling tests in TDD, #NoEstimates**
Seb Rose
SPA, London [U32]

2015-06-29 **Workshop on coaching**
Rachel Davies
SPA2015, London, UK [U33]

2015-07-20 **Mini-Munging: The Joy of Small Data. Big data is trendy, but most data is small. What's a useful toolkit for processing and manipulating data on a scale that doesn't require us to spin up vast clusters of servers, and where does Ruby fit in?**
Rob Miller (author of Text Processing with Ruby [U34])
Brighton Ruby Conference 2015 [U35]

2015-07-21 **Design Your Agile Project**
Johanna Rothman
Uberconf, Denver, CO [U36]

2015-07-22 **Tuning Your Agile Team**
Johanna Rothman
Uberconf, Denver, CO [U37]

2015-08-01 **Agile Teaching - Some things I've learned teaching Ruby to beginners.**
Mark Sobkowicz (author of Learn Game Programming with Ruby [U38])
Burlington Ruby Conference, Burlington, VT [U39]

2015-09-08 **Treat your Code as a Crime Scene**
Adam Tornhill
Swanseacon, Swansea, Wales, UK [U40]

2015-09-24 **Lead organizer**
Alex Miller (author of Clojure Applied [U41])
Strange Loop - St. Louis, MO [U42]

2015-10-07 **Full day Workshop: Code as a Crime Scene**
Adam Tornhill
GOTO Copenhagen, Denmark [U43]

What's Happening

But to really be in the know, you need to subscribe to the weekly newsletter. It'll keep you in the loop, it's a fun read, and it's free. All you need to do is create an account on pragprog.com [U44] (email address and password is all it takes) and select the checkbox to receive newsletters.

Solution to Pub Quiz

Here's the solution to this month's pub quiz:

I	O	D	C	V	R	S	E	Y
E	S	R	D	Y	O	I	V	C
C	Y	V	I	E	S	D	O	R
S	V	Y	O	I	E	R	C	D
R	D	I	V	C	Y	O	S	E
O	C	E	S	R	D	V	Y	I
D	I	C	Y	S	V	E	R	O
V	R	O	E	D	C	Y	I	S
Y	E	S	R	O	I	C	D	V

Rearranging the letters from any row, column, or smaller square leads to DISCOVERY.

Our Back Pages

An index to issues 67 through 72 of PragPub.

We're thinking about publishing an index to recent issues every so often. Like every six months. We'll make it available online, too. Here's an index to the first six months of 2015. Title, subtitle, author, summary, topics.

January 2015

A Testing Framework in Elixir: Transitioning from Ruby on Rails to Elixir • Bruce Tate • Bruce goes all in with Elixir. • elixir, testing, ruby

The Voyage to Altair: The Lost Computer • Michael Swaine and Paul Freiberger • Ed Roberts was facing bankruptcy, his mail-order hobby electronics firm crushed in the calculator wars of the 1970s. His response was to take a very big risk on an obviously crazy idea. • history

Turning APIs into Resources: An Exploration of Resource Oriented Computing • Tom Geudens • Resource Oriented Computing lets you build apps the way the Web is constructed. Here Tom shows how to turn APIs into resources, and why. • api, web

Functional Snippets: Decomposing Arrays • Chris Eidhof, Wouter Swierstra, and Florian Kugler • Launching a new series on functional programming in Swift. • swift, language

On Tap • Michael Swaine • What's in this issue.

Swaine's World • We Follow Twitter so You Don't Have To • Michael Swaine • Here's what our editor was paying attention to this past month on Twitter.

Also, a puzzle sort of tangentially related to computing. Or technology. Or science. Whatever Mike comes up with, really.

Rothman and Lester: What Not to Say in an Interview • Andy Lester and Johanna Rothman • You'd never make any of these job interview blunders, right? Um, are you sure? • career

New Manager's Playbook: How to Manage Your Boss • Marcus Blankenship • This month Marcus takes on the critical challenge of managing up. • career

Antonio on Books • Antonio Cangiano • Antonio Looks at all the new tech books of note. • books

Pragmatic Bookstuff • Want to meet one of the Pragmatic Bookshelf authors face-to-face? Here's where they'll be in the coming months. Also, find out which are the top-selling Pragmatic Bookshelf books and what new books are coming out. • books

Shady Illuminations: Self-Driving Cars • John Shade • John wonders if smart cars are a dumb idea.

February 2015

Creating Elixir Test Data With Blacksmith: Transitioning from Ruby on Rails to Elixir • Bruce Tate • Bruce gets all testy with Elixir. • elixir, language, testing

The Voyage to Altair: Going for Broke • Michael Swaine and Paul Freiberger • An excerpt from Mike and Paul's history of the personal computer, Fire in the Valley, and part two of a series. • history

Building Shootduino: Recreational Programming on the Arduino • Maik Schmidt • Maik was looking for a fun one-month project. So he built a retro space shooter game for the Arduino single-board computer. And he shares all the details here. • games, graphics, arduino

Functional Snippets: Function Composition • Chris Eidhof, Wouter Swierstra, and Florian Kugler • The next installment in our new series on functional programming in Swift. • swift, language

On Tap • Michael Swaine • What's in this issue.

Swaine's World • We Follow Twitter so You Don't Have To • Michael Swaine • Here's what our editor was paying attention to this past month on Twitter. Also, a puzzle sort of tangentially related to computing. Or technology. Or science. Whatever Mike comes up with, really.

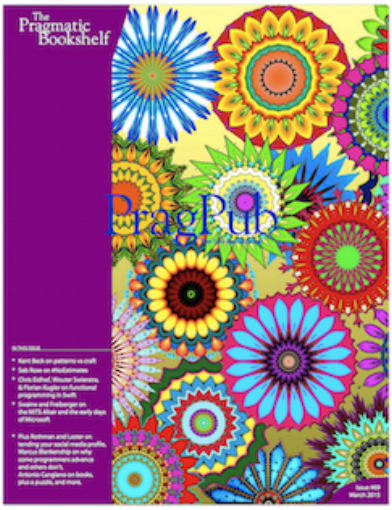
Rothman and Lester: The Google Résumé • Andy Lester and Johanna Rothman • Andy and Johanna explain why Google's job interview process probably means nothing to you. • career

New Manager's Playbook: No Surprises • Marcus Blankenship • Everyone hates annual reviews. Marcus explains how to make them work. • career

Antonio on Books • Antonio Cangiano • Antonio Looks at all the new tech books of note. • books

Pragmatic Bookstuff • Want to meet one of the Pragmatic Bookshelf authors face-to-face? Here's where they'll be in the coming months. Also, find out which are the top-selling Pragmatic Bookshelf books and what new books are coming out. • books

Shady Illuminations: Putting the Happy in Haptic • John Shade • You'd better get started if you want to build this sex toy before Valentine's Day.



March 2015

Patterns Enhance Craft: Thoughts on Patterns • Kent Beck • Kent has been through the patterns-vs-craft fight and come out on the other side. • agile, patterns

#NoEstimates Does Not Mean "No Estimates": It Means You're Asking the Wrong Question • Seb Rose • Sometimes the reason you're not getting the answers you want is that you're asking the wrong question. "How can I deliver better estimates?" is probably the wrong question. • agile, estimation

The Voyage to Altair: Rapid Reentry • Michael Swaine and Paul Freiberger • An excerpt from Mike and Paul's history of the personal computer, Fire in the Valley, and part three of a series. • history

Functional Snippets: A Functional Quicksort • Chris Eidhof, Wouter Swierstra, and Florian Kugler • The next installment in our new series on functional programming in Swift. • swift, language

On Tap • Michael Swaine • What's in this issue.

Swaine's World • We Follow Twitter so You Don't Have To • Michael Swaine • Here's what our editor was paying attention to this past month on Twitter. Also, a puzzle sort of tangentially related to computing. Or technology. Or science. Whatever Mike comes up with, really.

Rothman and Lester: What Your Social Media Profiles Should Look Like, and Why They Should be Different • Andy Lester and Johanna Rothman • Johanna and Andy look at the various social media platforms for presenting yourself to the world, and explain how to use them to best advantage. • career

New Manager's Playbook: Passed Over: Why Great Programmers Don't Get Promoted • Marcus Blankenship • For some reason, you just can't seem to make the leap from programmer to management, and you don't know why. Marcus can help. • career

Antonio on Books • Antonio Cangiano • Antonio Looks at all the new tech books of note. • books

Pragmatic Bookstuff • Want to meet one of the Pragmatic Bookshelf authors face-to-face? Here's where they'll be in the coming months. Also, find out which are the top-selling Pragmatic Bookshelf books and what new books are coming out. • books

April 2015

A Timely Essay on Apple Watch App Development: Know What the Apple Watch Can — and Should Do Before You Even Think about Creating an App • Jeff Kelley • Wait! Don't build that Apple Watch app yet. In this article, Jeff walks you through the questions you need to answer in conceiving of, designing, and producing an Apple Watch app. • mobile, watch

Rule #1 for Distributed Teams: The 2015 Edition • Jeff Langr • Distributed development teams face some daunting challenges. Jeff has been there and knows how to solve them. • agile

Meet the Social Side of Your Codebase: Approaching Your Code as a Crime Scene • Adam Tornhill • You can read the structure of the organization that produced a piece of code in the code's history — which will reveal clues to solving deep development problems. • agile, career

Functional Snippets: Flattening and Mapping Arrays • Chris Eidhof, Wouter Swierstra, and Florian Kugler • The next installment in our new series on functional programming in Swift. • swift, language

On Tap • Michael Swaine • What's in this issue.

Swaine's World • We Follow Twitter so You Don't Have To • Michael Swaine • Here's what our editor was paying attention to this past month on Twitter. Also, a puzzle sort of tangentially related to computing. Or technology. Or science. Whatever Mike comes up with, really.

Rothman and Lester: How to Deal with Recruiters without Going Crazy • Andy Lester and Johanna Rothman • Why recruiters are the way they are, and what to do about it. • career

New Manager's Playbook: Making the Transition to Manager • Marcus Blankenship • I thought I was getting a promotion. Why do I feel like I've lost something important? • career

Antonio on Books • Antonio Cangiano • Antonio Looks at all the new tech books of note. • books

Pragmatic Bookstuff • Want to meet one of the Pragmatic Bookshelf authors face-to-face? Here's where they'll be in the coming months. Also, find out which are the top-selling Pragmatic Bookshelf books and what new books are coming out. • books

Shady Illuminations: Are You Ready for the Diamond Age? • John Shade • Can 3D printers clean up our beaches? And why does Slate hate metal detecting?

May 2015

Unicursal Mazes and Space-Filling Curves: Cool Code and a Playful Product • Jamis Buck • You want to exercise your mind and develop new intuitions? Create a maze! • games, graphics, algorithms

Paul Learns To Parse: Part One • Michael Bevilacqua-Linn • Coming up with user stories is a technique for aligning your product development with actual use cases and user needs. What could we learn from telling developer stories? Michael explores. • agile, algorithms, language

Unfinished Revolution: Ted Nelson, Revolutionary • Michael Swaine and Paul Freiberger • This month and next we're profiling a couple of true revolutionaries. This month it's Ted Nelson, the Thomas Paine of the personal computer revolution. • history

Functional Snippets: Reduce • Chris Eidhof, Wouter Swierstra, and Florian Kugler • The next installment in our new series on functional programming in Swift. • swift, language

On Tap • Michael Swaine • What's in this issue.

Swaine's World • We Follow Twitter so You Don't Have To • Michael Swaine • Here's what our editor was paying attention to this past month on Twitter. Also, a puzzle sort of tangentially related to computing. Or technology. Or science. Whatever Mike comes up with, really.

Rothman and Lester: Six Real-World Lessons • Andy Lester and Johanna Rothman • They didn't teach you these lessons about programming in school. • career

New Manager's Playbook: Leadership • Marcus Blankenship • When you make the jump to running your own business, it's not about what you can produce anymore. It's about how you can lead. • career

Antonio on Books • Antonio Cangiano • Antonio Looks at all the new tech books of note. • books

Pragmatic Bookstuff • Want to meet one of the Pragmatic Bookshelf authors face-to-face? Here's where they'll be in the coming months. Also, find out which are the top-selling Pragmatic Bookshelf books and what new books are coming out. • books

Shady Illuminations: The Unexamined Life • John Shade • John wishes the Watch wouldn't.

June 2015

Hello Again, Android!: An Interview with Ed Burnette • Michael Swaine • Ed Burnette recently wrapped up the fourth edition of *Hello, Android*, and we collared him for a free-ranging interview on Android, his book, and programming in general. • android, mobile

Paul Learns To Parse: Part Two • Michael Bevilacqua-Linn • Coming up with user stories is a technique for aligning your product development with actual use cases and user needs. What could we learn from telling developer stories? Michael explores. In the second part of this two-part story, Michael's characters dive deeper into parsing and DSL creation. • agile, algorithms, language

Unfinished Revolution: Douglas Engelbart, Revolutionary • Michael Swaine and Paul Freiberger • This month and last month we're profiling a couple of true revolutionaries. This month it's Douglas Engelbart. • history

Functional Snippets: Reduce • Chris Eidhof, Wouter Swierstra, and Florian Kugler • The next installment in our new series on functional programming in Swift. • swift, language

On Tap • Michael Swaine • What's in this issue.

Swaine's World • We Follow Twitter so You Don't Have To • Michael Swaine • Here's what our editor was paying attention to this past month on Twitter. Also, a puzzle sort of tangentially related to computing. Or technology. Or science. Whatever Mike comes up with, really.

Rothman and Lester: Networking as Problem Solving • Andy Lester and Johanna Rothman • Johanna and Andy explain how networking can help you find the right job, but only if you do it right. • career

New Manager's Playbook: Defusing Frustrating Situations with Gentle Corrections • Marcus Blankenship • Marcus shows how to defuse potentially explosive situations with employees. • career

Antonio on Books • Antonio Cangiano • Antonio Looks at all the new tech books of note. • books

Pragmatic Bookstuff • Want to meet one of the Pragmatic Bookshelf authors face-to-face? Here's where they'll be in the coming months. Also, find out which are the top-selling Pragmatic Bookshelf books and what new books are coming out. • books

Index to Contents of January-June 2015 *PragPub*.