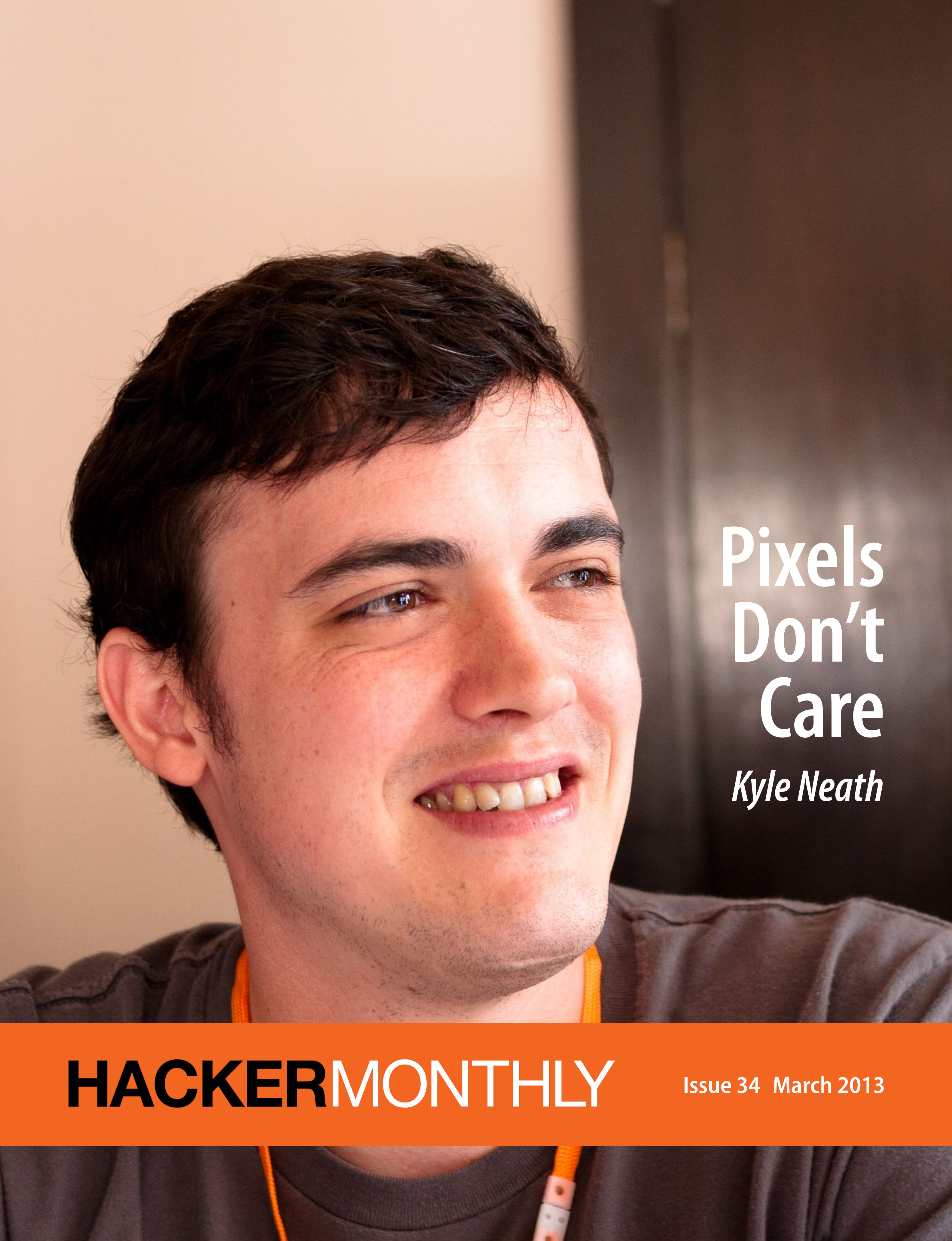


# Pixels Don't Care

*Kyle Neath*

HACKERMONTHLY

Issue 34 March 2013

HOW WOULD YOU FIX FINANCE

ADDEPAR

Engineers rebuilding the infrastructure that powers finance. ➔ careers.addepar.com

*Now you can hack on DuckDuckGo*

# DuckDuckHack

*Create instant answer plugins for DuckDuckGo*

*duckduckhack.com*

**Curator**
Lim Cheng Soon

**Contributors**
Jacques Mattheij
Kyle Neath
Jason Chen
Tom Limoncelli
Chris Zacharias
Mike Muuss
Kristian Kielhofner
Ben Kamens
Peteris Krumins
Krishnan Raman
Fat
Jacques Fuentes
Tanya Khovanova
Steven Corona
Justin Jackson

**Proofreaders**
Emily Griffin
Sigmarie Soto

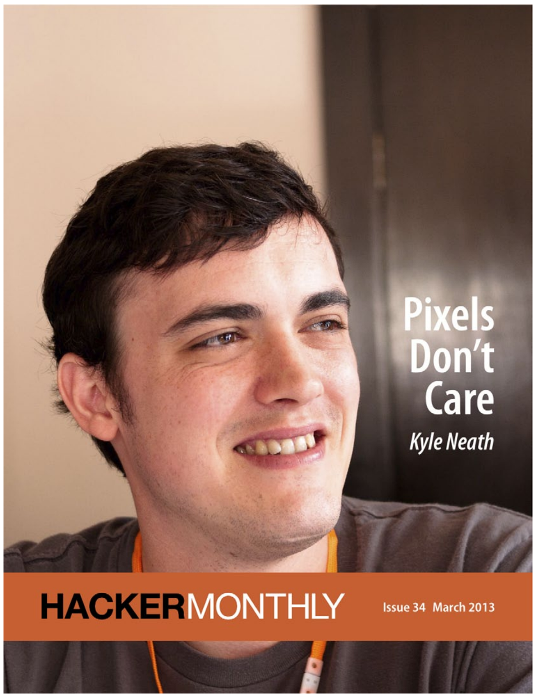
**Printer**
MagCloud

HACKER MONTHLY is the print magazine version of Hacker News — *news.ycombinator.com*, a social news website wildly popular among programmers and startup founders. The submission guidelines state that content can be "anything that gratifies one's intellectual curiosity." Every month, we select from the top voted articles on Hacker News and print them in magazine format. For more, visit *hackermonthly.com*

**Advertising**
ads@hackermonthly.com

**Contact**
contact@hackermonthly.com

**Published by**
Netizens Media
46, Taylor Road,
11600 Penang,
Malaysia.



**Cover Photo:** Andy Delcambre

# Contents

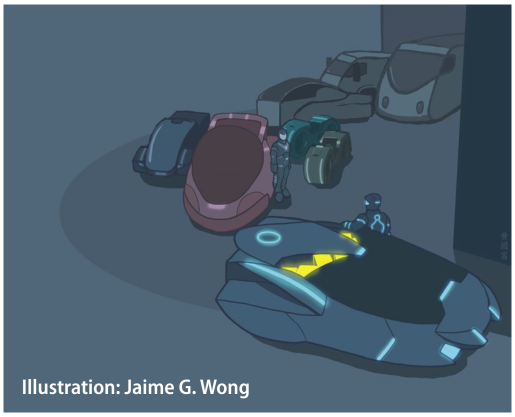
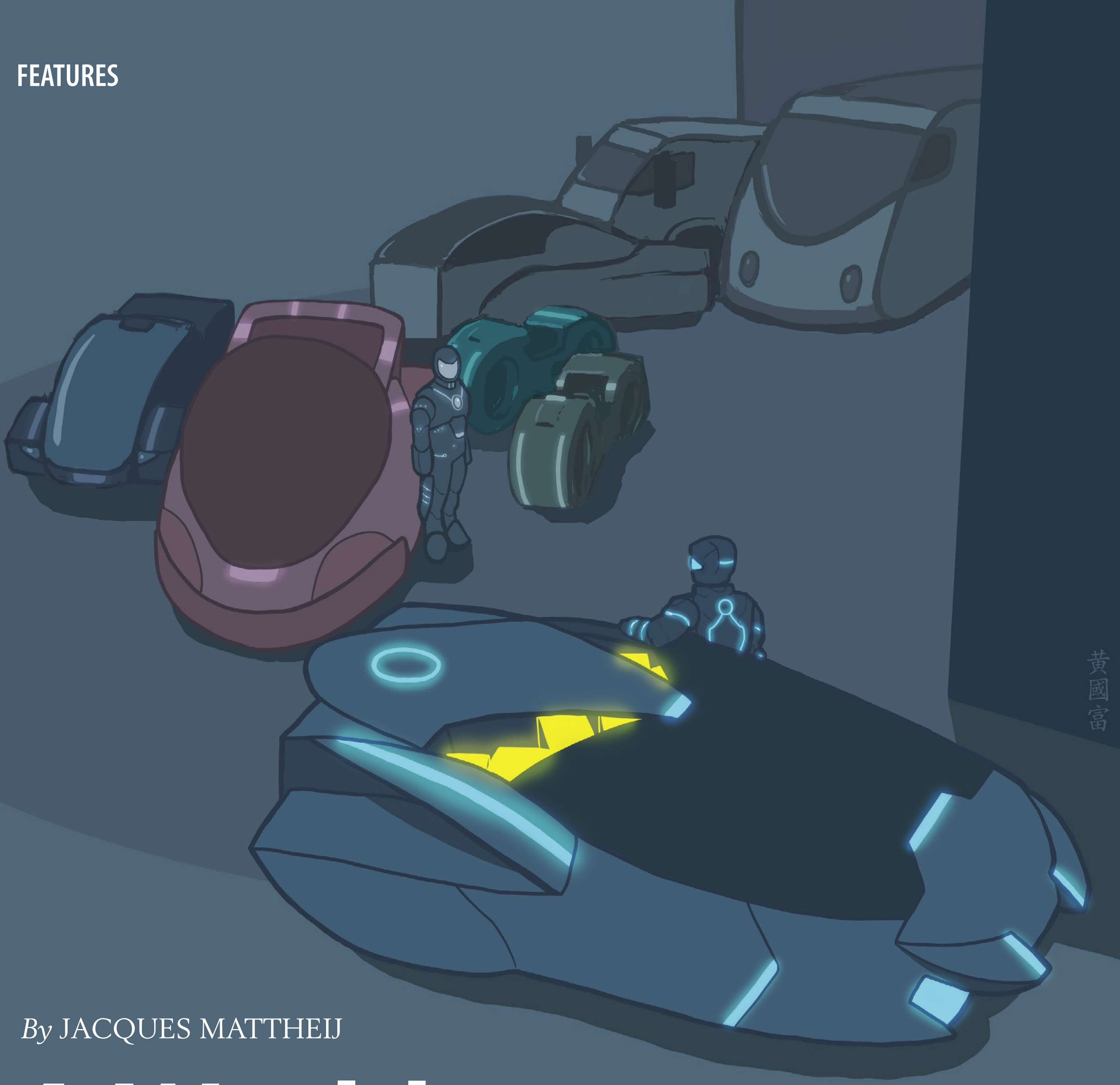


Illustration: Jaime G. Wong

For links to Hacker News dicussions, visit *hackermonthly.com/issue-34*



*By* JACQUES MATTHEIJ

# A World Without Power

The 14th of August, 2003 was an interesting day in many ways. I had just taken the family down to Toronto to spend a day there and pick up our newly minted passports at the Dutch embassy. We had some fun in the city seeing the sights, visited Niagara Falls and were on the way back on a sunny Thursday afternoon. What could possibly go wrong? We pulled into a gas station just outside of Barrie. We parked behind the person gassing up and waited our turn. The guy took his time, walked into the store to pay, got into his car and drove off. We moved up the line to the pump, I got out and put the nozzle into the fuel port, but nothing happened. This normally means that the pump hasn't been "cleared," which usually takes 10 seconds or so (an annoying little bell starts ringing in the office over the blinking light indicating the pump that needs to be cleared). After a minute of waiting, I went into the office. "Sorry bud, the power is out."

Bummer! The car we were driving was our family car — one of those Ford minivan affairs, and it wasn't exactly a paragon of fuel economy. 3.8 liter engine, still slow as could be and it drank like a fish. So when we pulled into the gas station the needle was at "empty" and the fuel warning light was on. We were stuck for the duration.

I looked at Barrie from the highway and noticed that there were lots of people in the streets, many more than you'd normally see in a place like that. Traffic lights were out. Billboards were dark. Little by little it dawned on me that this was more than just a trivial outage. I tried my cell phone; it still had a signal, so I called the island where we lived. We had a business there (a gas station, coincidentally) and I asked them if they had power. No, they didn't. Oh oh. But the generator was up and running and as far as they were concerned it was business as usual. Maybe a bit busier because some traffic from the highway pulled in on the island to gas up. I asked them to call the supplier (Wardlaw fuels in Sault Ste Marie, an awesome company to work with) to make sure they were going to be provisioned.

An hour and a half into the blackout, a truck from OPG pulled into the station. The guy in it took all the ice his truck could carry, paid cash and drove off again. Now I was getting really worried. OPG is the Ontario Power Generating Company, which runs the entire grid infrastructure. More importantly, they run the nuclear generators. If a guy with that much knowledge buys ice, he must know something that the general public doesn't know yet: that this outage is going to take a very long time — long enough that food will spoil. Nuclear power is funny in that it is excellent baseline power, but when you shut down a nuke it's going to be down for a while. They go down quite quickly, but they take forever to get back online.

I saw one guy walk around to the spot with the air hose for refilling tires. A few minutes later the same guy walks by with the length of the air hose coiled up and on his shoulder like it was the most normal thing in the world. Presumably he was going to use it to siphon fuel from one car into another. Let's hope they were at least both his cars. Another guy made off with the fire extinguisher. I've heard it said that the veneer over our society is very thin and that it won't take much to bring out the ugliness underneath, but I had not expected to see that illustrated so clearly and so quickly. Suddenly, I didn't feel safe and I did not want to wait for who knows how long next to an unfamiliar town in a vulnerable position like this with my wife and kid with me.

My mobile still worked, so I called the gas station on the island again. The mood around us continued to deteriorate from resigned to menacing; people were getting angry with the operator of the station (just some kid that was clearly not the cause of the trouble). I asked the people at our gas station on the island to please go to our house, get our other car (a Honda Civic hybrid, capable of doing 1200 km on a single tank if you were careful), load it up with fuel canisters and drive down to where we were. They said they'd get on it and that the station was being mobbed by people that wanted gas, some coming all the way from DesBarats and Sault Ste. Marie (a good 50 Km or 30 Miles away). They had already gone through one full drop (50,000 liters) and another one was on the way. Our trusty diesel was doing just fine running the pumps and half of the freezers. They had sold the food in the freezers so they could be switched off, and they had sold the other half at a discount rather than letting it go bad.

We talked to some of the people stranded around us. One group was a bunch of motorcycle enthusiasts, guys that not only drive but build their own bikes. Lots of nice metalworking there, so we talked for a bit about that. One girl that was stranded came from Toronto

**“I imagined a situation where the power was out for just a bit longer than normal and decided that we could not run that risk.”**

and was on the way to Muskoka, a popular place to spend the holidays. We waited… One of the bike guys went into town and scored a pizza. One of the pizza bakers had set up in the street and was selling pizzas for $5 as long as he had ingredients. We munched on our pizza, talked some more and waited some more.

Barrie is about 500 km from the island, and around 10 pm I saw our saviors roll into the station. Two kids with huge grins on their faces and music blaring loud from the open windows. I don’t think I was ever so glad to see my own car. The Honda was a veritable bomb, filled with as many canisters as they could cram in. We gassed up, then gassed the bikes of the bike guys (we swapped addresses; they invited us for a pig roast, which we attended, and they came to visit us on the island). The girl got enough gas to get back to Toronto, and we parceled out the remaining fuel in the same way — just enough to get as many people home as possible. No charge, obviously (people were on the whole expecting to pay the gas station with a credit card or debit card). We made a lot of friends that day :), and quite a few enemies, too (all those that we couldn’t help).

So, finally done we drove back to the island. It was the weirdest ride on Highway 17 North ever. Normally, even at night, there is light traffic there, but this time there wasn’t a single car. Just lots of wildlife, deer, moose and the occasional bear that had already taken over the highway so we went pretty slow. There weren’t any lights in the little villages on the road. When we got back to the island it was 5:30 am or so. The gas station was still open, and the pumps were still manned and going pretty much continuously. For 75 miles around there was no gas. The distribution point had huge tanks (millions of liters) that did a gravity dump into the delivery trucks, and then the truck would do a gravity dump into the underground storage tanks at the station. So none of that required power. And then the generator driven pumps would pull the fuel out of the underground tanks and deliver it to the customers just like normal.

The outage lasted 2 days and a bit. We went through our monthly amount of fuel in that time and we made the Sault Star (the local newspaper) after the power was restored. (Good thing, too, their presses were down just like everything else. We wouldn’t have been able to cope with the demand if there had been any publicity during the outage; word of mouth we could deal with, barely).

When we bought the station one of the first things I did was install that generator. I felt that what with the station being critical infrastructure for many of the islanders that we could not be dependent on Ontario’s power grid. The island was fed by cables to a step-down transformer. After that, there was a “low voltage” (not really that low, but low compared to long distance lines) distribution net on the island and every house had its own drop transformer. Outages were frequent — once every couple of months and more frequently in the winter months. Whenever lightning struck it was 50/50 that the power would go out. I imagined a situation where the power was out for just a bit longer than normal and decided that we could not run that risk. My software background makes me abhor single points of failure, so running an infrastructure business without a back-up in place seemed unwise and ran against my nature (even though the partners in the business thought it was a bit over the top, they still helped install the genny).

**“If you’re in the infrastructure business, design as if lives depend on it.”**

I didn’t have any idea that there would be such a massive power outage (there had not been one like that in decades). I also didn’t know that I was going to be one of the main beneficiaries of having a back-up. I definitely didn’t plan to get our business in the newspaper this way and did not realize how many people would end up relying on our silly little out-of-the-way gas station with its puny 2x50K liter reservoirs for a critical resource (a typical highway station has a 1/4 of a million liters). But being prepared for the occasion sure didn’t hurt. What is really scary is that apparently none of the other (much larger) stations had thought this scenario through. Critical infrastructure being down is annoying enough, but the knock-on effects are devastating.

Our house didn’t have any issues at all; we were “off the grid” by that time so our business worked uninterrupted. One thing that I was very impressed with was how well the cell phone network dealt with the outage. For at least 24 hours all the base stations worked, but after that point, they slowly dropped out one by one as their batteries ran dry.

Infrastructure is invisible, as long as it works. And only when it fails do you realize just how much we are all dependent on it and how badly we cope with such infrastructure being unavailable for any length of time. No power translates quickly into: no fuel, no water, no food and so on. If this had happened during the winter instead of a nice summer day, the results would have been much more severe.

There is no moral to this story, but I’d like to add one: if you’re in the infrastructure business, design as if lives depend on it. One day they may, even if you can’t foresee how that is possible — even if you’re just running a lousy gas station on some barely accessible island. ■

Jacques is the inventor of the live streaming webcam, founder of *camarades.com* / *ww.com* and a small time investor. He also collects insightful comments from Hacker News.

Illustration by Jaime G. Wong.

# Pixels Don’t Care

*By* KYLE NEATH

I’M SHORT.

When I was 20, I decided to try to make some extra money to pay my tuition by building websites for people. My work was good; it wasn’t phenomenal, but it was good. It was impossible for me to get work. Everything would be great until I met with a potential client. At which point they told me they’d rather hire a professional.

What they meant is that I looked too young. I didn’t really realize this was the problem until people started screwing me out of money. “You’re just a kid and you’ll get over it” I believe was the phrase my last client used to fuck me over.

Humans are really good at prejudice and intolerance.

The internet was a much different place eight years ago. Facebook wasn’t open to the public. Twitter didn’t exist. Google did not require legal names. I was just **kneath** who had a blog at *warpspire.com*. I didn’t have a picture and no one knew my age.

And the internet loved my work. I still remember the first day my blog was featured on CSSVault — it was one of the most exciting things to ever happen to me. How awesome was it that my work was highlighted as one of the best in the world? (CSSVault was quite a different beast 8 years ago, too.)

A few days later I received an email from the Art Director of a local agency asking me to come in and meet their team. And so it was that I was interviewing for a job to work on sites for the likes of Apple, Disney, HP, and RIM. Pretty fucking crazy. It felt good — it felt like validation that my work was worth paying for.

I remember the last question asked of me at the interview because it was possibly the most terrifying professional moment of my life. To paraphrase:

> *“You have no formal education and no experience with any big clients. What makes you think you could possibly be good enough to work here?”*

Through a stroke of luck, a moment of wit came upon me and I replied with the only thing my brain could grasp:

> *“I have no idea. I didn’t even know I was interviewing. Kris sent me an email asking me to come in today because he thought my work was good. Is it?”*

I never really got an answer, but I did get the job. Because my work was good. But I was given a much lower salary than my co-workers. For every hour I worked, the agency billed my time out at a 2,083% markup. To the client (who couldn't see my height), my time was worth over 20x the amount I was worth to the agency.

Looking back, I can't help but think this was discrimination. For age, for height, for whatever you will. I had no lower education than my peers, equal or better skills, and did work of the highest quality.

The physical world is harsh. I'm by all means a member of the privileged class in America by race, gender, and sexual orientation — yet a few inches of vertical height is all it took to diminish the value of my work.

At least they paid me.

About the same time, I started to get into Ruby on Rails. I wasn't really the most brilliant programmer or designer, but I could get stuff done. I was invited to hang out in the `#caboose` IRC channel. There aren't any avatars in IRC. No faces. No names. Just usernames and words.

I ended up making a lot of friends through caboose. Friends I still have today. Friends I've worked with, friends I haven't worked with. Friends who never saw my face or knew my age for almost half a decade. It just wasn't important.

We were working on code, on Photoshop documents — pixels. The pixels didn't care what we looked like. Over time we grew to respect each other. Not because of how handsome we were, but because of the things we built.

In a strange sense, it was a bit of a utopian work environment. How could the internet know you were gay? 80 years old? Hispanic? Transgender? Karl Rove? It just didn't matter. Respect was earned through actions and the words you actually said (hard to squeeze rumor out of publicly logged chat).

**“For the first time in human history, it’s possible to be represented solely through the merits of your work.”**

It took until early 2009 for me to realize the real value of this network. I was miserable at my job and I sent a long-winded email to court3nay inquiring about working with ENTP [entp.com]. ENTP was a half-product, half-consulting agency at this point comprised almost solely of caboosers. All of whom had never met me or ever heard my voice. About 30 seconds later I got a response:

> *Hey Kyle,*
>
> *That’s pretty fuckin awesome, if you’ll pardon my french.*
>
> *We’re just heading out to breakfast, I mean, an important company meeting, but I’ll get back to you today.*
>
> *Courtenay & Rick*

And then a follow up:

> *OK, I’ve talked it over with everyone (unanimous— “kyle? awesome!”)*
>
> *I think you’ll fit into our team perfectly.*

No in-person interview. No phone calls. No technical test. They were confident enough in my pixels to give me what equated to my dream job at that point in my life.

Really fucking crazy.

This industry we work in is magical. For the first time in human history, it’s possible to be represented (almost) solely through the merits of your work. Build something magical, push it up to GitHub under a pseudonym, and you could become one of the most sought after programmers in the world.

That’s really fucking awesome.

There’s plenty of prejudice and intolerance in our world — and in our industry. But never forget that pixels don’t care. ■

---

Kyle Neath works at GitHub in San Francisco. He loves to build beautiful things with software.

Reprinted with permission of the original author. First appeared in *hn.my/pixels* (warpspire.com)

Photo by Andy Delcambre.

stripe

Accept payments online.

# Mechanics of a Small Acquisition

*By* JASON CHEN

When a startup successfully exits, chances are it was an acquisition. Unfortunately for the founders, that acquisition was likely their first, while the acquirer has probably gone through many. This was the case with Stypi. Fortunately, my cofounder and I were lucky enough to have had access to several other acquired founders who helped us ultimately navigate our first multimillion-dollar exit for a company barely a year old. Hopefully by sharing what we learned and encountered, you can be slightly less lost, should you be faced with an acquisition of your own.

*DISCLAIMER: Every startup and founder group is unique so the data provided here may not apply in all cases. In particular, it is skewed towards acquisitions of <$25M made by much larger companies. An acquisition is important enough that you should always do your own research. Reach out to acquired founders in your space, company size range, and/or acquired by the same company that may acquire your company. If any of these apply to me, feel free to contact me at jason@stypi.com*

## First Contact

Potentials acquirers are a lot like the opposite sex: their intentions are confusing, but the prospects are exciting. Initially, you will most likely be approached by a founder or the Corporate Development department of a larger company, acting on behalf of an interested internal team. Either way, get comfortable with this person, as he/she will be setting up and facilitating your meetings and will likely be the one you eventually haggle with over terms.

Before moving forward, it is worth noting that the cost of pursuing an acquisition is nonzero — in fact, it's quite high. You and your cofounder(s) will be consumed by the process for weeks and divulge a lot of otherwise private information to potential competitors. But more importantly, rejection hurts. Believing that the finish line is just yards away and finding that it is actually miles is not going to be good for your company's morale.

## Courtship

Assuming the decision is to open your company's kimono and pursue an acquisition, you and your cofounder(s) will end up going to a series of meetings with the potential acquiring company (except at the pace of trying to decide if you want to get married by the end of the week). For larger companies, mutual NDAs will likely be signed by the first meeting. Decisions to continue the relationship are made very quickly after each meeting. This of course goes both ways — we decided to end the relationship early with more than half of interested potential acquirers.

If feelings are positive on both sides, arrangements would be made to meet again, probably with different people. Regardless of who we were meeting with, they were usually available the next or following day, which seems to suggest the priority for acquisition meetings are reasonably high. One founder, we discovered, cancelled racing Audi R8 to meet with us.

## The Term Sheet

We averaged three to four of these dates before we were invited into bed with a term sheet. The relationship is not exclusive, so it is kosher and advisable to be pursued by multiple potential acquirers in parallel. Ideally they would be lined up such that multiple term sheets would come at roughly the same time. Different companies prefer to have varying amounts of information or certainty before putting together a term sheet. In general, the time spent or saved here will be made up for during due diligence. But, for your purpose of timing the term sheets, it is not unreasonable to ask about the potential acquirer's timeline and form expectations.

Once you get your first term sheet, it's time to celebrate with your cofounder(s). If the offer is interesting enough, then it's also time to get a lawyer. Ideally other offers will soon be coming in and you can use them to get better terms from each company. While you will be negotiating the key terms, your lawyers will be taking care of less familiar ones, like escrow, indemnification, etc. They'll explain what this means, the consequences, etc., and for significant ones, they will ask how hard you want to push for more favorable terms. For these, we usually just asked them to push for market terms.

It took us roughly a week to agree on the terms and decide which company Stypi would join.

## Due Diligence and Closing

Now it's time for monogamy. When you sign a term sheet there will also be a separate exclusivity agreement that requires you to reject any other outstanding offers and, for a period of 45 days, remain faithful and not solicit other offers. There will be more meetings and every document your company has ever signed will be scrutinized (which your lawyers will want to screen before sharing) to make sure you did not misrepresent your company's position. This process should be taken seriously as either party can still back out of the deal. But the default outcome is that the deal will close and unless you are hiding something big, like a lawsuit or stealing code, there shouldn't be anything to lose sleep over.

In parallel with due diligence, weighted towards the tail end, preparations will begin for closing. Mechanically this means filling out and signing a lot of forms. A lot of time will be spent chasing down information from people connected to your company; for example, your advisors' addresses or investors' wire instructions. Many will need signatures so there might actually be some physical chasing down.

The complexity of the deal or parties involved will dictate the time frame, but this process will take roughly 3-4 weeks for small companies.

## Exit

What happens after exit is entirely up to up to you. Life can be as different or as similar to what it was before. Some founders take time off, some throw massive parties, and others buy and crash brand new Lamborghinis. For Stypi, life has not changed very much and we went straight to work the day after closing. We still have a long way to go before Stypi becomes what we envisioned and we chose an acquirer that shares and wants us to pursue that vision unencumbered. ■

Jason Chen is the founder of a YCombinator backed startup called Stypi. It was recently acquired by *Salesforce.com*, where Jason continues to work on Stypi.

# Four Unix Commands I Abuse Every Day

*By* TOM LIMONCELLI

A CO-WORKER WATCHED ME type the other day and noticed that I use certain Unix commands for purposes other than they are intended. Yes, I abuse Unix commands.

## 1 grep dot (view the contents of files prefixed by their file name)

I want to view the contents of a few files, but I want each line prepended with the file's name. My solution?

```
$ grep . *.txt
jack.txt:Once upon a time
jack.txt:there was a fellow named Jack.
lyingryan.txt:Now that "trickle down economics"
has been
lyingryan.txt:tested for 30 years and the data
shows it
lyingryan.txt:has been a total failure, candi-
dates
lyingryan.txt:still claim that cutting taxes for
lyingryan.txt:billionaires will help the econ-
omy.
market.txt:Jack went to market to sell the
family
market.txt:cow.
market.txt:He came back with a handful of magic
beans.
$
```

`grep` is a search tool. Why am I using it like a weird version of `cat`? Because `cat` doesn't have an option to prepend the file name to each line of text. And it shouldn't.

Note that `.` matches lines with at least 1 character. That is, blank lines are not included. If we change `.` (matches any 1 character) to `^` (matches the beginning of a line), then every line will be matched because every line, no matter how short or long, has a beginning! However, the period key is easier to type than the caret, at least on my keyboard. Therefore, if I don't need the blanks, I don't request them.

Example: The other day, I grabbed the `/etc/network/interfaces` file from 6 different Linux boxes. I needed to review them all. Each was copied to a file name that was the same as the hostname. `grep . *` let me view them all easily and where they come from was annotated on each line.

## 2 "more star pipe cat"

### (cat files with a header between each one)

Let's look at another way to accomplish my example of comparing 6 files. In this case, I want to print the contents of each file but separate the contents with the file name. Yes, I could do it in a loop:

```
$ for i in *.txt ; do echo === $i === ; cat $i ;
done
```

However, that takes a lot of typing.

This is where I abuse more. Are you familiar with more? It prints the contents of files but pauses every screenful to ask "More?" Pressing SPACE shows one screenful more. Pressing RETURN shows one line more.

When more was new it was very dumb. It had no search functions, you could skip forward a file but not skip back, it assumed your screen size was 24 lines long, etc. Heaven forbid you weren't on a hardware terminal fixed at 80x24; these new-fangled graphic screens with windows that could be resized confused more. Resizing your window while using more confused it even more. Another problem was that if you piped the output of more to another program things got totally confusing because those prompts were sent down the pipe. Certainly, the next program in the pipe doesn't expect to see a "More?" every 24 lines.

Luckily someone came along and created a replacement for more that fixed all of those problems. Obviously these features and bug fixes were added to more and we all benefited. No, that's not what really happened. Obviously they wrote a new program from scratch and called it "more 2.0" so we could keep typing "more" but have all those new features. No, that's not what happened. In the grand tradition of Unix having a sense of humor, this new program was called less. Thus begat funny conversations like, "Do you use more?" "No, I couldn't live without less." Or the joke: So a pager walks into a bar, and the bartender says "What are you, more or less?"

Some versions of Unix have the old traditional more and less commands. In many Unix and Unix-like systems both are the same program, but the code detects that it was run as more and goes into "more emulation mode." Either way, more gets you the old behavior with a few bugs fixed and less gets you all the cool new stuff.

If you have been using Linux for fewer than 5 years there is a good chance that you didn't know that more existed and quite possibly that you were confused why less is called less. Now you know.

Which brings us back to our story. Sometimes people get so used to typing more that they type it when they mean cat. For example they type:

```
more * | command | command2
```

when they mean:

```
cat * | command | command2
```

For example:

```
more * | grep -v bar | sort
```

Old more would send the prompts to grep, which would pass them to sort, which would get very confused. You'd have to press SPACE a number of times and, since you didn't see any output, you would usually bang on the keyboard in frustration. It's all a big mess.

less is smart enough to detect that its output is going to a pipe and would emulate cat. This is very smart.

Even smarter is that when less is emulating more instead of producing "the big mess," it acts like cat but outputs little headers for each file:

```
$ more * | cat
::::::::::::::
jack.txt
::::::::::::::
Once upon a time
there was a fellow named Jack.
::::::::::::::
lyingryan.txt
::::::::::::::
Now that "trickle down economics" has been
tested for 30 years and the data shows it
has been a total failure, candidates
still claim that cutting taxes for
billionaires will help the economy.
::::::::::::::
market.txt
::::::::::::::
Jack went to the free-market to sell the family
cow.
He came back with a handful of magic beans.
$
```

Isn't that pretty?

That works on Linux but not on *BSD. However, there's a solution that works on both. We simply take advantage of the fact that if head is given more than one file name it prints a little header in front of each file. However, we want to see the entire file, not just the first 10 that head normally shows. No worries. We assume the files are shorter than 99999 lines long and do this:

```
$ head -n 99999 *
==> jack.txt <==
Once upon a time
there was a fellow named Jack.
==> lyingryan.txt <==
Now that "trickle down economics" has been
tested for 30 years and the data shows it
has been a total failure, candidates
still claim that cutting taxes for
billionaires will help the economy.
==> market.txt <==
Jack went to market to sell the family
cow.
He came back with a handful of magic beans.
$
```

Note: You can use `head -n 0` on Linux to mean "all lines." That, however, doesn't work on FreeBSD and other Unixes. (Hey, BSD folks: can you fix that?) You can also use `tail +0`, but the header it draws is not as pretty.

## 3 grep --color=always '^|foo|bar'

As you get older your eyesight gets worse. It becomes more difficult to find something in a field of text. Here's an eye test. Below is a list of recently run jobs on a Ganeti cluster.

```
$ gnt-job list
157994 success OS_DIAGNOSE
158073 running CLUSTER_VERIFY
158074 success CLUSTER_VERIFY_CONFIG
158075 success CLUSTER_VERIFY_GROUP(7ee44802-
85d3-40fb-bd36-a7e701ecea29)
158076 success CLUSTER_VERIFY_GROUP(72a2138c-
dc07-494d-bd02-ebff7916c9bc)
158077 success CLUSTER_VERIFY_GROUP(457c7377-
c83b-4fed-9ebe-a2974e2c521f)
158156 success OS_DIAGNOSE
158367 success CLUSTER_VERIFY
158368 waiting CLUSTER_VERIFY_CONFIG
158371 success CLUSTER_VERIFY_GROUP(457c7377-
c83b-4fed-9ebe-a2974e2c521f)
158432 waiting OS_DIAGNOSE
$
```

How quickly can you find which job is running? It's kind of buried in there. (The answer is job #158073.)

The most interesting jobs are the ones that are running and the ones that are waiting to run. It would be nice to have those highlighted. My first instinct was to simply use `grep` to remove the successful jobs:

```
$ gnt-job list | grep -v success
158073 running CLUSTER_VERIFY
158368 waiting CLUSTER_VERIFY_CONFIG
158432 waiting OS_DIAGNOSE
$
```

However, it is useful to see those jobs in context with all the other jobs. What I really want is to highlight the running and waiting jobs. Ah! `egrep --color=always` would color the things it finds, right? Ah, but `egrep` only shows what is found. We get:

```
$ gnt-job list | egrep --color=always
'running|waiting'
158073 running CLUSTER_VERIFY
158368 waiting CLUSTER_VERIFY_CONFIG
158432 waiting OS_DIAGNOSE
$
```

So, how can we output every line but also highlight certain words? Well . matches everything so we could use that, right? No, it matches every single character. We'd just get 100% red text (try it: `egrep . file file2`). What else does every line have? It has a beginning! We make a regular expression that matches lines with "a beginning" -or- lines with "running" -or- lines with "waiting." Every line will match and therefore be output. Since "the beginning of each line" has no length, nothing additional will be highlighted in red.

This regular expression matches any line that has a beginning or has the word "running" or has the word "waiting." The matched text will be colored red.

```
$ gnt-job list | egrep --color=always
'^|running|waiting'
157994 success OS_DIAGNOSE
158073 running CLUSTER_VERIFY
158074 success CLUSTER_VERIFY_CONFIG
158075 success CLUSTER_VERIFY_GROUP(7ee44802-
85d3-40fb-bd36-a7e701ecea29)
158076 success CLUSTER_VERIFY_GROUP(72a2138c-
dc07-494d-bd02-ebff7916c9bc)
158077 success CLUSTER_VERIFY_GROUP(457c7377-
c83b-4fed-9ebe-a2974e2c521f)
158156 success OS_DIAGNOSE
158367 success CLUSTER_VERIFY
158368 waiting CLUSTER_VERIFY_CONFIG
158371 success CLUSTER_VERIFY_GROUP(457c7377-
c83b-4fed-9ebe-a2974e2c521f)
158432 waiting OS_DIAGNOSE
$
```

Now you can easily see which jobs are running and waiting and still get the full context.

*(Note: For some reason this doesn't work on Mac OS and *BSD. However, $ matches the end of a line and works the same way.)*

I set up an alias so I can use this command all the time:

```
alias j="gnt-job list | egrep --color=always
'^|running|waiting'"
```

Note the careful use of `'` within `"`.

If you want to highlight more than just "running" and "waiting," you will need to use slightly more complex regular expressions:

Highlight starting at the word, continuing to the end of the file:

```
egrep --color=always '^|running.*$|waiting.*$'
```

Highlight the entire darn line if it has either word in it:

```
egrep --color=always '^|^.* (running|waiting)
.*$'
```

Of course, if you are typing these commands instead of using them in a script or alias, the least typing to highlight "foo" and "bar" is:

```
egrep '^|foo|bar'
```

Chances are `--color=auto` is the default and the right thing will happen. If not, add the `--color=always`.

*Note: A co-worker just pointed out that* `""` *matches every line and doesn't result in all text being highlighted. He wins for reducing the regex's to be even smaller. Just remove the ^ at the front:*

```
alias j="gnt-job list | egrep --color=always
'|running|waiting'"
```

or

```
egrep --color=always '|^.* (running|waiting)
.*$'
```

## 4 fmt -1 (split lines into individual words)

If you are not familiar with the fmt command, that's probably because you use a modern text editor like vim or emacs, which can do the formatting for you. In the old days we had to call an external command to do our formatting. Back then all Unix commands were small, single-function tools that could be combined to do great things. Now every new Unix command seems to be trying to have more features than MS-Word. But I digress.

`fmt -n` takes text as input and reformats it into nicely shaped paragraphs with no line longer than *n*. For example, `fmt -65` formats text in nice paragraphs with no line longer than 65 characters.

But what if you have a word that is longer than 65 characters? Does it truncate it? No, then you get a line with just that word on it. (Ok, I lied about "no lines longer than *n*.")

So how can we abuse this program? Simple! Suppose we have a bunch of text and want to list out the individual words one per line. Well, words that are "too long" are printed on their own line and we want every word to be printed on its own line. Therefore, why don't we tell `fmt` that all words are "too long" by saying we want the paragraphs to be formatted to be 1 character long!

```
$ fmt -1 <fraudulent.txt
Fraud
is
telling
a
lie
that
benefits
you
and
not
the
person
or
people
you
tell
it
to.
$
```

Why would you want to do that? There are plenty of situations where this is useful!

Recently, I found myself with long lines of text that mixed usernames and numbers, and I wanted to extract the names. Sure, I could have figured something out with awk or put it into a text editor and copied out the names. Instead, I did this:

```
$ fmt -1 <the_file.txt | egrep -v '^[0-9]'
fred
jane
bob
$
```

Recently, I was curious which IP addresses are mentioned on my wiki:

```
$ cat *.wiki | fmt -1 | egrep -E '[0-9]{1,3}\.
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' | sort -u
192.168.1.4
192.168.1.7
255.255.255.0#
255.255.255.240
8.3.8.1
<code>172.16.240.1
<code>172.16.240.2
```

Ok, that's not a perfect list, but I was able to do that in a few seconds rather than an hour of writing code.

A simple improvement: Transform < and > and a lot of other punctuation into spaces, then delete spaces at the end.

```
$ cat *.wiki | tr "#:@;()<>=,'-\"" ' ' | fmt -1
| egrep -E '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.
[0-9]{1,3}' | tr -d ' ' | sort -u
172.16.240.1
172.16.240.2
192.168.1.4
192.168.1.7
255.255.255.0
255.255.255.240
8.3.8.1
```

That's a lot cleaner. 8.3.8.1 is a version number, not an IP address, but this is good enough for a first pass through the list. ■

Tom is an internationally recognized author, speaker, and system administrator. His best known books include Time Management for System Administrators (O'Reilly) and The Practice of System and Network Administration (Addison-Wesley). He works at Google in NYC on the Ganeti project. [code.google.com/p/ganeti]

# Page Weight Matters

*By* CHRIS ZACHARIAS

THREE YEARS AGO, while I was a web developer at YouTube, one of the senior engineers began ranting about the page weight of the video watch page being far too large. The page had ballooned to as high as 1.2MB and dozens of requests. This engineer openly vented that "if they can write an entire Quake clone in under 100KB, we have no excuse for this!" Given that I agreed with him and was excited to find a new project, I decided to champion the cause of getting the YouTube watch page to weigh in under 100KB. On the shuttle home from San Bruno that night, I coded up a prototype. I decided to limit the functionality to just a basic masthead, the video player, five related videos, a sharing button, a flagging tool, and ten comments loaded in via AJAX. I code-named the project "Feather."

Even with such a limited set of features, the page was weighing in at 250KB. I dug into the code and realized that our optimization tools (i.e. Closure compilation) were unable to exclude code that was never actually used in the page itself (which would be an unfair expectation of any tool under the circumstances). The only way to reduce the code further was to optimize the CSS, JavaScript, and image sprites myself by hand. After three painstaking days, I had arrived at a much leaner solution. It still was not under 100KB though. Having just finished writing the HTML5 video player, I decided to plug it in instead of the far heavier Flash player. Bam! 98KB and only 14 requests. I threaded the code with some basic monitoring and launched an opt-in to a fraction of our traffic.

After a week of data collection, the numbers came back… and they were baffling. The average aggregate page latency under Feather had actually INCREASED. I had decreased the total page weight and number of requests to a tenth of what they were previously, and somehow the numbers were showing that it was taking LONGER for videos to load on Feather. This could not be possible. Digging through the numbers more and after browser testing repeatedly, nothing made sense. I was just about to give up on the project, with my world view completely shattered, when my colleague discovered the answer: geography.

When we plotted the data geographically and compared it to our total numbers broken out by region, there was a disproportionate increase in traffic from places like Southeast Asia, South America, Africa, and even remote regions of Siberia. Further investigation revealed that, in these places, the average page load time under Feather was over TWO MINUTES! This meant that a regular video page, at over a megabyte, was taking more than TWENTY MINUTES to load! This was the penalty incurred before the video stream even had a chance to show the first frame. Correspondingly, entire populations of people simply could not use YouTube because it took too long to see anything. Under Feather, despite it taking over two minutes to get to the first frame of video, watching a video actually became a real possibility. Over the week, word of Feather had spread in these areas and our numbers were completely skewed as a result. Large numbers of people who were previously unable to use YouTube were suddenly able to.

Through Feather, I learned a valuable lesson about the state of the Internet throughout the rest of the world. Many of us are fortunate to live in high bandwidth regions, but there are still large portions of the world that do not. By keeping your client side code small and lightweight, you can literally open your product up to new markets. ■

---

Chris is the cofounder of imgix. Prior to that, he worked at Google, YouTube, and Xerox where his primary focus was on high-performance web tools and infrastructure. He is a graduate of the RIT New Media program.

Reprinted with permission of the original author.
First appeared in *hn.my/weight* (chriszacharias.com)

# The Story of the PING Program

*By* MIKE MUUSS

Yes, it's true! I'm the author of `PING` for Unix. `PING` is a little thousand-line hack I wrote in an evening, which practically everyone seems to know about. :-)

I named it after the sound a sonar makes, inspired by the whole principle of echo-location. In college I had done a lot of modeling of sonar and radar systems, so the "cyberspace" analogy seemed very apt. It's exactly the same paradigm applied to a new problem domain: `PING` uses timed IP/ICMP `ECHO_REQUEST` and `ECHO_REPLY` packets to probe the "distance" to the target machine.

My original impetus for writing `PING` for 4.2a BSD Unix came from an offhand remark in July 1983 by Dr. Dave Mills while we were attending a DARPA meeting in Norway. He described some work that he had done on his "Fuzzball" LSI-11 systems to measure path latency using timed ICMP Echo packets.

In December of 1983 I encountered some odd behavior of the IP network at BRL. Recalling Dr. Mills' comments, I quickly coded up the `PING` program, which revolved around opening an ICMP style `SOCK_RAW` `AF_INET` Berkeley-style `socket()`. The code compiled just fine, but it didn't work — there was no kernel support for raw ICMP sockets! Incensed, I coded up the kernel support and had everything working well before sunrise. Not surprisingly, Chuck Kennedy (aka "Kermit") had found and fixed the network hardware before I was able to launch my very first "ping" packet. But I've used it a few times since then. *grin* If I'd known then that it would be my most famous accomplishment in life, I might have worked on it another day or two and added some more options.

The folks at Berkeley eagerly took back my kernel modifications and the `PING` source code, and it's been a standard part of Berkeley Unix ever since. Since it's free, it has been ported to many systems since then, including Microsoft Windows 95 and WindowsNT. You can identify it by the distinctive messages it prints, which look like this:

```
PING vapor.arl.army.mil (128.63.240.80): 56 data bytes
64 bytes from 128.63.240.80: icmp_seq=0 time=16 ms
64 bytes from 128.63.240.80: icmp_seq=1 time=9 ms
64 bytes from 128.63.240.80: icmp_seq=2 time=9 ms
64 bytes from 128.63.240.80: icmp_seq=3 time=8 ms
64 bytes from 128.63.240.80: icmp_seq=4 time=8 ms
^C
----vapor.arl.army.mil PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 8/10/16
```

In 1993, ten years after I wrote `PING`, the USENIX association presented me with a handsome scroll, pronouncing me a joint recipient of The USENIX Association 1993 Lifetime Achievement Award, which was presented to the Computer Systems Research Group, University of California at Berkeley 1979-1993.

**“From my point of view, PING is not an acronym standing for Packet InterNet Grouper; it’s a sonar analogy.”**

“Presented to honor profound intellectual achievement and unparalleled service to our Community. At the behest of CSRG principals we hereby recognize the following individuals and organizations as CSRG participants, contributors and supporters.” Wow!

From my point of view, `PING` is not an acronym standing for Packet InterNet Grouper; it’s a sonar analogy. However, I’ve heard second-hand that Dave Mills offered this expansion of the name, so perhaps we’re both right. Sheesh, and I thought the government was bad about expanding acronyms! :-)

Phil Dykstra added ICMP Record Route support to `PING`, but in those early days few routers processed them, making this feature almost useless. The limitation on the number of hops that could be recorded in the IP header precluded this from measuring very long paths.

I was insanely jealous when Van Jacobson of LBL used my kernel ICMP support to write `TRACEROUTE`. He realized that he could get ICMP Time-to-Live Exceeded messages when pinging by modulating the IP time to life (TTL) field. I wish I had thought of that! :-) Of course, the real `traceroute` uses UDP datagrams because routers aren’t supposed to generate ICMP error messages for ICMP messages.

The best `PING` story I’ve ever heard was told to me at a USENIX conference, where a network administrator with an intermittent Ethernet had linked the `PING` program to his vocoder program, in essence writing:

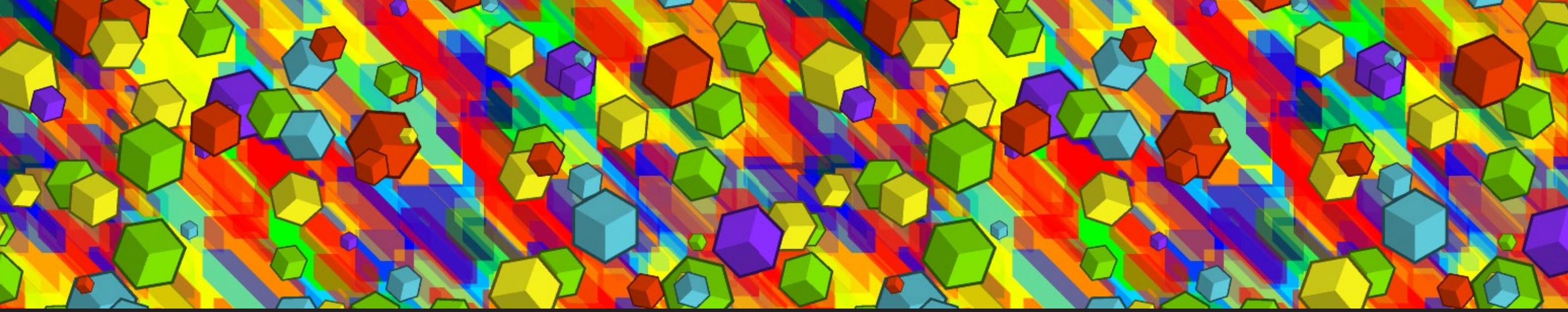
```
ping goodhost | sed -e 's/.*/
ping/' | vocoder
```

He wired the vocoder’s output into his office stereo and turned up the volume as loud as he could stand. The computer sat there shouting “Ping, ping, ping...” once a second, and he wandered through the building wiggling Ethernet connectors until the sound stopped. And that’s how he found the intermittent failure. ■

---

The late Mike Muuss was the architect of BRL-CAD, a substantial third-generation CSG solid modeling system, and the author of ping, ttcp, and assorted other network goodies.

First appeared in *hn.my/ping*



# Packets of Death

*By* KRISTIAN KIELHOFNER

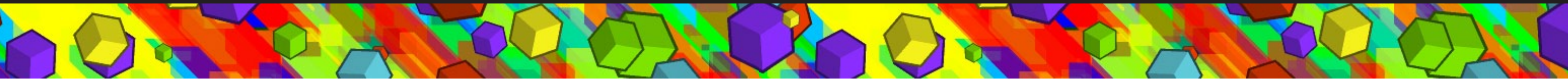


Illustration: Patrick Hoesly

PACKETS OF DEATH. I started calling them that because that's exactly what they are.

Star2Star has a hardware OEM that has built the last two versions of our on-premise customer appliance. I'll get more into this appliance and the magic it provides in another post. For now let's focus on these killer packets.

About a year ago, we released a refresh of this on-premise equipment. It started off simple enough, pretty much just standard Moore's Law stuff. Bigger, better, faster, cheaper. The new hardware was 64-bit capable, had 8X as much RAM, could accommodate additional local storage, and had four Intel (my preferred Ethernet controller vendor) gigabit Ethernet ports. We had (and have) all kinds of ideas for these four ports. All in all it was pretty exciting.

This new hardware flew through performance and functionality testing. The speed was there and the reliability was there. Perfect. After this extensive testing we slowly rolled the hardware out to a few beta sites. Sure enough, problems started to appear.

All it takes is a quick Google search to see that the Intel 82574L Ethernet controller has had at least a few problems, including, but not necessarily limited to, EEPROM issues, ASPM bugs, and MSI-X quirks. We spent several months dealing with each and every one of these and eventually, we thought we were done.

We weren't. It was only going to get worse.

I thought I had the perfect software image (and BIOS) developed and deployed. However, that's not what the field was telling us. Units kept failing. Sometimes a reboot would bring the unit back, but usually it wouldn't. When the unit was shipped back, however, it would work when tested.

Wow. Things just got weird.

The weirdness continued and I finally got to the point where I had to roll up my sleeves. I was lucky enough to find a very patient and helpful reseller in the field to stay on the phone with me for three hours while I collected data. This customer location, for some reason or another, could predictably bring down the Ethernet controller with voice traffic on their network.

Let me elaborate on that for a second. When I say "bring down" an Ethernet controller I mean BRING DOWN an Ethernet controller. The system and Ethernet interfaces would appear fine and then after a random amount of traffic the interface would report a hardware error (lost communication with PHY) and lose link. Literally the link lights on the switch and interface would go out. It was dead.

Nothing but a power cycle would bring it back. Attempting to reload the kernel module or reboot the machine would result in a PCI scan error. The interface was dead until the machine was physically powered down and powered back on. In many cases, for our customers, this meant a truck roll.

While debugging with this very patient reseller I started stopping the packet captures as soon as the interface dropped. Eventually, I caught on to a pattern: the last packet out of the interface was always a 100

Trying provisional response, and it was always a specific length. Not only that, I ended up tracing this (Asterisk) response to a specific phone manufacturer's INVITE.

I got off the phone with the reseller, grabbed some guys, and presented my evidence. Even though it was late in the afternoon on a Friday, everyone did their part to scramble and put together a test configuration with our new hardware and phones from this manufacturer.

We sat there, in a conference room, and dialed as fast as our fingers could. Eventually we found that we could duplicate the issue! Not on every call and not on every device, but every once in a while we could crash the Ethernet controller. However, every once in a while we couldn't at all. After a power cycle, we'd try again and hit it. Either way, as anyone who has tried to diagnose a technical issue knows, the first step is duplicating the problem. We were finally there.

Believe me, it took a long time to get here. I know how the OSI stack works. I know how software is segmented. I know that the contents of a SIP packet shouldn't do anything to an Ethernet adapter. It just didn't make any sense.

Between packet captures on our device and packet captures from the mirror port on the switch we were finally able to isolate the problem packet. It turns out it was the received INVITE, not the transmitted 100 Trying! The mirror port capture never saw the 100 Trying hit the wire.

Now we needed to look at this INVITE. Maybe the userspace daemon processing the INVITE was the problem. Maybe it was the transmitted 100 Trying. One of my colleagues suggested we shut down the SIP software and see if the issue persisted. No SIP software running, no transmitted 100 Trying.

First, we needed a better way to transmit the problem packet. We isolated the INVITE transmitted from the phone and used tcpreplay [tcpreplay.synfin.net] to play it back on command. Sure enough, it worked. Now, for the first time in months, we could shut down these ports on command with a single packet. This was significant progress and it was time to go home, which really meant it was time to set this up in the lab at home!

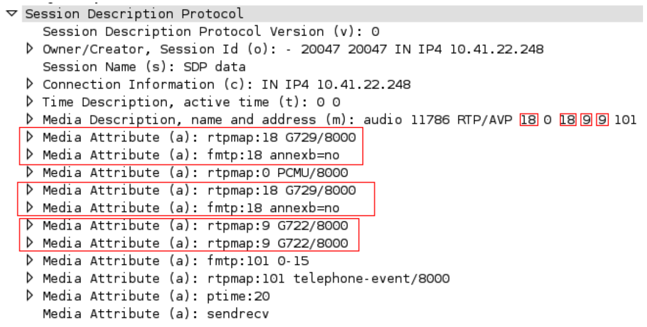
Before I go any further I need to give another shout out to an excellent open source piece of software I found. Ostinato [code.google.com/p/ostinato/] turns you into a packet ninja. There's literally no limit to what you can do with it. Without Ostinato, I could have never gotten beyond this point.

With my packet Swiss army knife in hand I started poking and prodding. What I found was shocking.

It all starts with a strange SIP/SDP quirk. Take a look at this SDP:

```
v=0
o=- 20047 20047 IN IP4 10.41.22.248
s=SDP data
c=IN IP4 10.41.22.248
t=0 0
m=audio 11786 RTP/AVP 18 0 18 9 9 101
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:0 PCMU/8000
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:9 G722/8000
a=rtpmap:9 G722/8000
a=fmtp:101 0-15
a=rtpmap:101 telephone-event/8000
a=ptime:20
a=sendrecv
```

Wireshark picture:



```
▽ Session Description Protocol
    Session Description Protocol Version (v): 0
  ▷ Owner/Creator, Session Id (o): - 20047 20047 IN IP4 10.41.22.248
    Session Name (s): SDP data
  ▷ Connection Information (c): IN IP4 10.41.22.248
  ▷ Time Description, active time (t): 0 0
  ▷ Media Description, name and address (m): audio 11786 RTP/AVP 18 0 18 9 9 101
  ▷ Media Attribute (a): rtpmap:18 G729/8000
  ▷ Media Attribute (a): fmtp:18 annexb=no
  ▷ Media Attribute (a): rtpmap:0 PCMU/8000
  ▷ Media Attribute (a): rtpmap:18 G729/8000
  ▷ Media Attribute (a): fmtp:18 annexb=no
  ▷ Media Attribute (a): rtpmap:9 G722/8000
  ▷ Media Attribute (a): rtpmap:9 G722/8000
  ▷ Media Attribute (a): fmtp:101 0-15
  ▷ Media Attribute (a): rtpmap:101 telephone-event/8000
  ▷ Media Attribute (a): ptime:20
    Media Attribute (a): sendrecv
```

Yes, I saw it right away too. The audio offer is duplicated and that's a problem, but again, what difference should that make to an Ethernet controller?!? Well, if nothing else it makes the Ethernet frame larger...

But wait, there were plenty of successful Ethernet frames in these packet captures. Some of them were smaller, some were larger. No problems with them. It was time to dig into the problem packet. After some more Ostinato-fu and plenty of power cycles, I was able to isolate the problem pattern (with a problem frame).

Warning: we're about to get into some hex.

The interface shutdown is triggered by a specific byte value at a specific offset. In this case, the specific value was hex 32 at 0x47f. Hex 32 is an ASCII 2. Guess where the 2 was coming from.

```
a=ptime:20
```

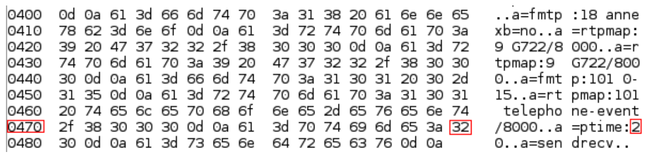
All of our SDPs were identical (including ptime, obviously). All of the source and destination URIs were identical. The only difference was the Call-IDs, tags, and branches. Problem packets had just the right Call-ID, tags, and branches to cause the "2" in the ptime to line up with 0x47f.

BOOM! With the right Call-IDs, tags, and branches (or any random garbage) a "good packet" could turn into a "killer packet" as long as that ptime line ended up at the right address. Things just got weirder.

While generating packets I experimented with various hex values. As if this problem couldn't get any weirder, it does. I found out that the behavior of the controller depended completely on the value of this specific address in the first received packet to match that address. It broke down to something like this:

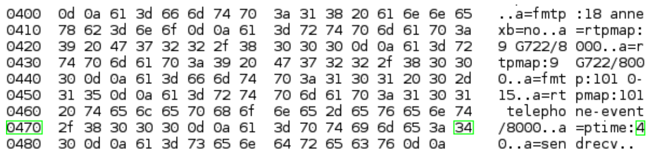
```
Byte 0x47f = 31 HEX (1 ASCII) - No effect
Byte 0x47f = 32 HEX (2 ASCII) - Interface shutdown
Byte 0x47f = 33 HEX (3 ASCII) - Interface shutdown
Byte 0x47f = 34 HEX (4 ASCII) - Interface inoculation
```

Bad:



```
0400   0d 0a 61 3d 66 6d 74 70  3a 31 38 20 61 6e 6e 65   ..a=fmtp :18 anne
0410   78 62 3d 6e 6f 0d 0a 61  3d 72 74 70 6d 61 70 3a   xb=no..a =rtpmap:
0420   39 20 47 37 32 32 2f 38  30 30 30 0d 0a 61 3d 72   9 G722/8 000..a=r
0430   74 70 6d 61 70 3a 39 20  47 37 32 32 2f 38 30 30   tpmap:9  G722/800
0440   30 0d 0a 61 3d 66 6d 74  70 3a 31 30 31 20 30 2d   0..a=fmt p:101 0-
0450   31 35 0d 0a 61 3d 72 74  70 6d 61 70 3a 31 30 31   15..a=rt pmap:101
0460   20 74 65 6c 65 70 68 6f  6e 65 2d 65 76 65 6e 74    telepho ne-event
0470   2f 38 30 30 30 0d 0a 61  3d 70 74 69 6d 65 3a 32   /8000..a =ptime:2
0480   30 0d 0a 61 3d 73 65 6e  64 72 65 63 76 0d 0a      0..a=sen drecv..
```

Good:



```
0400   0d 0a 61 3d 66 6d 74 70  3a 31 38 20 61 6e 6e 65   ..a=fmtp :18 anne
0410   78 62 3d 6e 6f 0d 0a 61  3d 72 74 70 6d 61 70 3a   xb=no..a =rtpmap:
0420   39 20 47 37 32 32 2f 38  30 30 30 0d 0a 61 3d 72   9 G722/8 000..a=r
0430   74 70 6d 61 70 3a 39 20  47 37 32 32 2f 38 30 30   tpmap:9  G722/800
0440   30 0d 0a 61 3d 66 6d 74  70 3a 31 30 31 20 30 2d   0..a=fmt p:101 0-
0450   31 35 0d 0a 61 3d 72 74  70 6d 61 70 3a 31 30 31   15..a=rt pmap:101
0460   20 74 65 6c 65 70 68 6f  6e 65 2d 65 76 65 6e 74    telepho ne-event
0470   2f 38 30 30 30 0d 0a 61  3d 70 74 69 6d 65 3a 34   /8000..a =ptime:4
0480   30 0d 0a 61 3d 73 65 6e  64 72 65 63 76 0d 0a      0..a=sen drecv..
```

When I say "no effect" I mean it didn't kill the interface, but it didn't inoculate the interface either (more on that later). When I say the interface shutdown, well, remember my description of this issue - the interface went down. Hard.

With even more testing I discovered this issue with every version of Linux I could find, FreeBSD, and even when the machine was powered up complaining about missing boot media! It's in the hardware; the OS has nothing to do with it. Wow.

To make matters worse, using Ostinato, I was able to craft various versions of this packet - an HTTP POST, ICMP echo-request, etc. Pretty much whatever I wanted. With a modified HTTP server configured to generate the data at byte value (based on headers, host, etc) you could easily configure an HTTP 200 response to contain the packet of death - and kill client machines behind firewalls!

I know I've been pointing out how weird this whole issue is. The inoculation part is by far the strangest. It turns out that if the first packet received contains any value (that I can find) other than 1, 2, or 3, the interface becomes immune from any death packets (where the value is 2 or 3). Also, valid ptime attributes are defined in multiples of 10 - 10, 20, 30, 40. Depending on Call-ID, tag, branch, IP, URI, etc. (with this buggy SDP), these valid ptime attributes line up perfectly. Really, what are the chances?!?

All of a sudden it hsd become clear why this issue was so sporadic. I'm amazed I tracked it down at all. I've been working with networks for over 15 years and I've never seen anything like this. I doubt I'll ever see anything like it again. At least I hope I don't...

I was able to get in touch with two engineers at Intel and send them a demo unit to reproduce the issue. After working with them for a couple of weeks they determined there was an issue with the EEPROM on our 82574L controllers.

They were able to provide new EEPROM and a tool to write it out. Unfortunately, we weren't able to distribute this tool and it required unloading and reloading the e1000e kernel module, so it wouldn't be preferred in our environment. Fortunately (with a little knowledge of the EEPROM layout), I was able to work up some bash scripting and ethtool magic to save the "fixed" eeprom values and write them out on affected systems. We now have a way to detect and fix these problematic units in the field. We've communicated with our vendor to make sure this fix is applied to units before they are shipped to us. What isn't clear, however, is just how many other affected Intel Ethernet controllers are out there.

I guess we'll just have to see... ■

Kristian Kielhofner is the co-founder and CTO of Star2Star Communications, creators of the world's most reliable business communications solution. Since creating AstLinux in 2004, Kristian has spent most of his time working on various technologies that interest him.

# I Don't Understand

*By* BEN KAMENS

I'VE NOTICED THAT the most mature and accomplished developers I've worked with are also those who most frequently say "I don't understand" when they're listening to a technical explanation. This has been the case with coworkers both at Fog Creek and at Khan Academy.

In one way, it's counterintuitive. Shouldn't the senior devs already know everything? But it makes a lot of sense. Those who are most secure in their own abilities are the most comfortable to admit when they haven't fully wrapped their minds around something. Newer devs assume that their confusion is their own fault. They don't want to interrupt others due to their own perceived shortcomings.

New devs should try to really get just how common it is to not fully understand a technical problem. Most tech stacks have crossed the threshold of one person being able to hold the entire codebase in their head, especially at companies that are hiring. And once that threshold is crossed, you'll start hearing from people about the new JavaScript rendering framework or the latest MapReduce pipeline or a bug in the deploy script or a proposal for a new caching pattern, and a little voice in your head is going to start saying, "Wait…I don't get it."

"I don't understand" is the perfect response. You're not insulting anybody. You're not showing weakness. You're building a culture of respect for how smart everybody is, because you know that after a few minutes of explanation you will get it.

Either that or you'll find a bug. I like to think of "I don't understand" as a kind of reverse rubber ducking. Except in this version, the duck comes alive and quacks and stomps and "I don't understand"s all over your keyboard while forcing you to explain various things.

It's most said by the best, decades after they've become a master. We newer devs should follow their lead and get rid of any stigma associated with those words. ■

Ben is lead dev at Khan Academy, where he started volunteering after watching one of Sal's talks and feeling left with no choice but to help. Ben was previously VP of Engineering at Fog Creek Software, where he spent five and a half years learning how to push bits around with small, fast teams.

# My Favorite Regex of All Time

*By* PETERIS KRUMINS

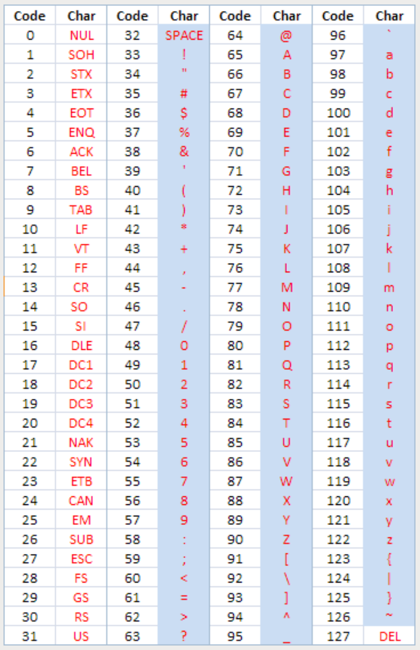
I THOUGHT I'D SHARE my favorite regex of all time:

```
[ -~]
```

Any ideas what this regexp matches?

It matches all ASCII characters from the space to the tilde. What are those characters? They're all printable characters!

Take a look at the ASCII table. The printable characters start at the space and end at the tilde:



| Code | Char | Code | Char | Code | Char | Code | Char |
|---|---|---|---|---|---|---|---|
| 0 | NUL | 32 | SPACE | 64 | @ | 96 | ` |
| 1 | SOH | 33 | ! | 65 | A | 97 | a |
| 2 | STX | 34 | " | 66 | B | 98 | b |
| 3 | ETX | 35 | # | 67 | C | 99 | c |
| 4 | EOT | 36 | $ | 68 | D | 100 | d |
| 5 | ENQ | 37 | % | 69 | E | 101 | e |
| 6 | ACK | 38 | & | 70 | F | 102 | f |
| 7 | BEL | 39 | ' | 71 | G | 103 | g |
| 8 | BS | 40 | ( | 72 | H | 104 | h |
| 9 | TAB | 41 | ) | 73 | I | 105 | i |
| 10 | LF | 42 | * | 74 | J | 106 | j |
| 11 | VT | 43 | + | 75 | K | 107 | k |
| 12 | FF | 44 | , | 76 | L | 108 | l |
| 13 | CR | 45 | - | 77 | M | 109 | m |
| 14 | SO | 46 | . | 78 | N | 110 | n |
| 15 | SI | 47 | / | 79 | O | 111 | o |
| 16 | DLE | 48 | 0 | 80 | P | 112 | p |
| 17 | DC1 | 49 | 1 | 81 | Q | 113 | q |
| 18 | DC2 | 50 | 2 | 82 | R | 114 | r |
| 19 | DC3 | 51 | 3 | 83 | S | 115 | s |
| 20 | DC4 | 52 | 4 | 84 | T | 116 | t |
| 21 | NAK | 53 | 5 | 85 | U | 117 | u |
| 22 | SYN | 54 | 6 | 86 | V | 118 | v |
| 23 | ETB | 55 | 7 | 87 | W | 119 | w |
| 24 | CAN | 56 | 8 | 88 | X | 120 | x |
| 25 | EM | 57 | 9 | 89 | Y | 121 | y |
| 26 | SUB | 58 | : | 90 | Z | 122 | z |
| 27 | ESC | 59 | ; | 91 | [ | 123 | { |
| 28 | FS | 60 | < | 92 | \ | 124 | \| |
| 29 | GS | 61 | = | 93 | ] | 125 | } |
| 30 | RS | 62 | > | 94 | ^ | 126 | ~ |
| 31 | US | 63 | ? | 95 | _ | 127 | DEL |

[ -~] matches all printable ASCII characters

I love this. ■

Peteris Krumins is 28 years old and he's from Riga, Latvia. He loves blogging and writing books about programming.

# Tom, Dick & Harry

*By* KRISHNAN RAMAN

ONE DAY, THE bean counter said to the boss, "Unless you fire one employee, we are not going to be profitable this year."

The boss loudly said, "Find the oldest guy who has been at this company for the longest time and is still not making six figures. We'll fire him!"

Impeccable management logic here: The guy has been with the company for the longest time. The guy is the oldest. He is still not making six figures. Obviously a sure loser. Fire this loser and save money!

Three programmers, Tom, Dick and Harry, overheard their boss and got to work immediately.

The employee database obviously looks like this:

```
case class Employee(age:Int, tenure:Int,
salary:Int, name:String)

def rnd(min:Int,max:Int) = math.round(min+math.
random*(max-min)).toInt

val empDB = for(employees<-1 to 10000) yield
Employee( rnd(25,35),rnd(1,6),rnd(85000,150000),
"emp"+employees)
```

So we see there are 10,000 employees at this company.

They are between 25 and 35 years old.

They have been with the company anywhere from 1 to 6 years.

They make between $85,000 to $150,000.

Tom was a PHP/JS programmer. He worked on rapidly prototyped web-apps and didn't give a rat's ass about efficiency.

Tom reasoned: This is just a nested sort. Let me first sort on the tenure; that will tell me who has been with the company for the longest time. Then I'll sort on age; that tells me who the oldest bloke is. Then I'll sort on salary; that tells me who is making the least money. That should be it!

In Scala, Tom's code would look like:

```
scala> empDB.sortBy(e=>(e.tenure,e.age,-e.
salary)).reverse.head
res0: Employee = Employee(35,6,85079,emp7435)
```

Dick was a back-end systems programmer. He thought deep and hard about algorithms and Big O performance and such.

So Dick thought: There's no need to sort anything. We just want the oldest guy with the longest tenure who makes the least salary. That just an `O(n)` problem, NOT an `O(nlog(n))` problem. We just need the `min` tuple.

In Scala, Dick's code looks like this:

```
scala> empDB.minBy(e=>(-e.tenure,-e.age,e.salary))
res1: Employee = Employee(35,6,85079,emp7435)
```

Harry was a math major who went about looking for mathematical abstractions in the simplest of code. He took the day off and thought long and hard about this problem while lounging in the bathtub.

Suddenly, it occurred to Harry: Hey, this is a monoid!

Naked Harry happily danced the mathematician's Eureka Dance and set about coding.

In Scala, Harry's code looks like:

```
case class Monoid(e:Set[Employee]) {
  def plus(a:Employee,b:Employee)= {
  if(a.age > b.age) a else
  if(b.age > a.age) b else
  if(a.tenure > b.tenure) a else
  if(b.tenure > a.tenure) b else
  if (a.salary > b.salary) b else
  if(b.salary > a.salary) a else
  if( math.random > 0.5) a else b
}
  val identity = Employee(0,0,0,"id")
  val theUnluckyOne = e.foldLeft(identity)((a,b)=>plus(a,b))
}

scala> Monoid(empDB.toSet).theUnluckyOne
res2: Employee = Employee(35,6,85079,emp7435)
```

In scalaz, it would look like this:

```
import scalaz.Monoid

case class EmpMonoid(e:Set[Employee]) extends Monoid[Employee] {

  override val zero:Employee = Employee(0,0,0,"emp0")

  override def append(a:Employee, b: =>Employee):Employee = {
  if(a.age > b.age) a else
  if(b.age > a.age) b else
  if(a.tenure > b.tenure) a else
  if(b.tenure > a.tenure) b else
  if (a.salary > b.salary) b else
  if(b.salary > a.salary) a else
  if( math.random > 0.5) a else b
 }

 val theUnluckyOne:Employee =  e.foldLeft(zero)((a:Employee, b:Employee)=>append(a,b))
}

scala> EmpMonoid( empDB.toSet).theUnluckyOne
res12: Employee = Employee(35,6,85079,emp7435)
```

The next day, Tom, Dick and Harry met with the boss.

They happily announced, "We've found the unlucky bastard! He is employee number 7435. He has been with the company for 6 years. He is 35 years old. He makes $85,079. Fire him and we'll be profitable again!"

The boss was very impressed, but he wanted to know how they came up with this solution.

Tom said, "This is just a nested sorting problem."

Dick said, "There is no need to sort at all! It's just a min-tuple problem. Takes `O(n)` to find the right answer."

Harry said, "You are both trivially right. The most important insight is that this is a monoid!"

"But what is a monoid ?" the boss asked.

"Well, you have a set with an associative plus and an identity. That's a monoid!" said Harry.

Nobody understood anything, so Harry said, "Think of the employee database as a set of employees. I claim two employees can be added!"

"What do you mean? How do you add an employee to another employee?" they asked.

"Well," Harry said, "If you are asked to add two employees, return the guy who is older. But if both of them are the same age, then return the guy who has been with the company the longest. But if both guys have been working for us for the same number of years, then return the guy with the lowest salary. But if they both earn the same as well, then just randomly pick one over the other."

"And how does all that work?" they asked.

"Well, let's define an identity employee. If you add something to this identity, you get back the thing you added. So we create a dummy employee who has been with us for 0 years, so that everybody else has been here longer than the identity. Then the addition works out."

"OK. I still don't see how all this works," said the boss.

"Well, if we seed a catamorphism with the identity employee and aggregate over the set, we collapse to the unlucky employee 7435, who has been here the longest, is the oldest and makes the least money," Harry triumphantly announced.

Everybody stared at Harry, absolutely speechless.

Long story short: Harry was immediately fired. Dick was made the CTO and Tom was appointed as Dick's secretary. ■

Krishnan Raman is a data scientist at Twitter. He has contributions to several Twitter open source Scala libraries including Scalding, Bijections, Algebird. He works on big data & machine learning algorithms.

**Bonus Problem:** What happened to employee number 7435?

Solution at the bottom of this column.

**Solution:** Employee number 7435 was the boss! You can't fire the boss!! He took the least amount of salary home for tax avoidance purposes. The bulk of his earnings came from equity.

# What I Learned Building Twitter Bootstrap

*By* FAT

I DIDN'T THINK I'D learn anything building Bootstrap.

Actually, I was pretty confident I wouldn't learn anything.

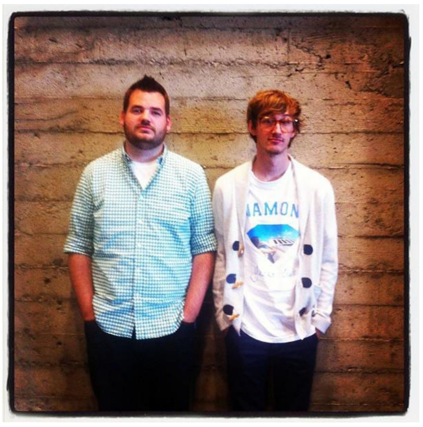
As a technical challenge, Bootstrap just isn't very interesting. It's trying to provide a collection of components — components like modals, tooltips, and grids that have been around the Web forever — to people who don't spend a lot of time writing frontend code.

## That's It

There's nothing groundbreaking there.

I knew it would enable people to move faster and build more beautiful products, and that was great. I also knew that Bootstrap would be relatively successful — it being something I think really needed to exist and it being released under Twitter's name.

But what I didn't anticipate was the extent of its popularity and, in turn, what its popularity meant.



## 40,000 Stars

Today, Bootstrap is really fucking popular. However, what people often don't realize is that despite its popularity, and despite it being a "Twitter" project, it isn't actually maintained by a team at Twitter (nor was it ever).

Bootstrap is (and has always been) maintained by two nerds who like to write code together. Just two nerds.

What's more is that this wasn't something we did in the office — there was no "Bootstrap team" and no 20% time. It was just Mark and me hacking in our free time. And this is significant because what building Bootstrap has taught me more than anything else is that's all I really care about.

## Writing Code with the Homies

Sitting on my bed alone, stressed out, hammering through 50+ issues a night isn't why I wanted to release Bootstrap with Mark; or Ratchet with Dave and Connor; or Ender with Dustin; or Hogan with Rob; or Bower with Alex.

Getting together and creating something with your friends is amazing, and for me, it's easily one of the most fun and rewarding things I do. I love it. And that's why I've done it and will continue to do it.

The trick is not losing sight of this. ■

---

Fat is a software engineer at Obvious Corp. He is also the co-author of Bootstrap, Ratchet, Bower, and a number of other open source projects.

Reprinted with permission of the original author. First appeared in *hn.my/bootstrap* (medium.com)

Photo by Olivia Watkins.

# A Letter to My Daughter, Augusta, in Ruby

*By* JACQUES FUENTES

I WANTED TO CREATIVELY express my affection for my daughter, Augusta, in the way I know best. I chose Ruby for its flexibility and elegance. My hope is to introduce her to its boundless beauty someday soon using this composition.

```
require "./love"

a_letter to: Augusta do
  twas(only: 16.months.ago) { The::Universe << You.to(OurFamily) }
  life.has :been => %w(i n c r e d i b l y).zip(*"wonderful!").ever_since
  We::Wish.we_could { experience these_moments: over & over }
  You.will always_be: Loved, and: Cherished
  until Infinity.ends do; Forever.(); end
end
```

This is a real, working, program, which outputs Augusta, we <3 you! when executed. Be sure to read the `love.rb` file, which supports the letter's syntax. I tried to keep it symmetrical and legible so that the source closely resembles the letter's content.

```
Augusta = Awesome = true and Loved = Cherished = true

Infinity = (+1.0 / 0)..(-1.0 / 0)
def Infinity.ends; false; end
Forever = -> { puts "Augusta, we \033[31m<3\033[0m you!"; sleep 5 }
Incredible = :wunderbar!
%w(We The).map { |const| self.class.const_set const, Module.new do; end }
OurFamily = :the_number_one_most_important_thing # not breakfast
```

```ruby
def a_letter(*to); yield Augusta; end

class Numeric
  def method_missing(*); instance_eval { self }; end
end

def twas(as_if_it_were = {}, &re)
  memories = as_if_it_were.fetch :only, 1.day.ago
  re.call memories
end

class You
  class << self
    def will(always_be_loved); end
    def to(us = OurFamily); end
  end
end

class The::Universe < Infinity.class
  def self.<<(you); end
end

def life
  Class.new do
    def self.has(since); since.fetch :been, Incredible; end
  end
end

class Array
  def ever_since; end
end

class String
  def each; self.chars; end
end

module We
  class Wish
    def self.we_could(&blk)
      klass = Module.new do
        def self.experience(these_moments_for); Infinity; end
      end.instance_eval &blk
    end
  end
end

def over(and_over = Infinity); end

trap :INT, :IGNORE # Forever and ever!
```

Jacques Fuentes is a father, husband, hacker, autodidact, classical music addict, friend of dogs, and last but not least: seriously envious of full-blown unix beards. He began teaching myself web development in early 2008 which quickly transformed into a passion for composing elegant code and solving problems simply.

# My IQ

*By* TANYA KHOVANOVA

When I came to the US, I heard about Mensa — the high IQ society. My IQ had never been tested, so I was curious. I was told there was a special IQ test for non-English speakers and that my fresh immigrant status and lack of English knowledge was not a problem. I signed up.

There were two tests. One test had many rows of small pictures, and I had to choose the odd one out in each row. That was awful. The test was English-free, but it wasn't culture-free. I couldn't identify some of the pictures at all. We didn't have such things in Russia. I remember staring at a row of tools that could as easily have been from a kitchen utensil drawer as from a garage tool box. I didn't have a clue what they were.

But the biggest problem was that the idea of crossing the odd object out seemed very strange to me in general. What is the odd object out in this list?

*Cow, hen, pig, sheep.*

The "right" answer is supposed to be hen, as it is the only bird. But that is not the only possible correct answer. For example, pig is the only one whose meat is not kosher. And, look, sheep has five letters while the rest have three.

Thus, creative people get fewer points. That means IQ tests actually measure how standard and narrow your mind is.

The second test asked me to continue patterns. Each page had a three-by-three square of geometric objects. The bottom right corner square, however, was empty. I had to decide how to continue the pattern already established by the other eight squares by choosing from a set of objects they provided.

This test is similar to continuing a sequence. How would you continue the sequence 1,2,3,4,5,6,7,8,9? The online database of integer sequences has 1479 different sequences containing this pattern. The next number might be:

- 10, if this is the sequence of natural numbers;
- 1, if this is the sequence of the digital sums of natural numbers;
- 11, if this the sequence of palindromes;
- 0, if this is the sequence of digital products of natural numbers;
- 13, if this is the sequence of numbers such that 2 to their powers doesn't contain 0;
- 153, if this is the sequence of numbers that are sums of fixed powers of their digits;
- 22, if this is the sequence of numbers for which the sum of digits equals the product of digits; or
- any number you want.

Usually when you are asked to continue a pattern the assumption is that you are supposed to choose the simplest way. But sometimes it is difficult to decide what the testers think is the simplest way. Can you replace the question mark with a number in the following sequence: 31, ?, 31, 30, 31, 30, 31, … You might say that the answer is 30 as the numbers alternate, or you might say that the answer is 28 as these are the amount of days for each month.

Towards the end of my IQ test, the patterns were becoming more and more complicated. I could have supplied several ways to continue the pattern, but my problem was that I wasn't sure which one was considered the simplest.

When I received my results, I barely made it to Mensa. I am glad that I am a member of the society of people who value their brains, but it bugs me that I might not have been creative enough to fail their test. ■

Tanya Khovanova is a research affiliate at MIT and a freelance mathematician. Her current interests lie in recreational mathematics including puzzles, magic tricks, combinatorics, number theory, geometry, and probability theory. Her website is located at *tanyakhovanova.com*, her highly popular math blog at *blog.tanyakhovanova.com* and her Number Gossip website at *numbergossip.com*

# MEET MANDRILL

By MailChimp

Mandrill is a new way to send transactional, triggered, and personalized emails.
It's also the world's largest species of monkey.

MANDRILL.COM

# How I Automated the Boring Parts of Life

*By* STEVEN CORONA

I HAVE A BAD habit: waiting until the last minute to do things. For instance, last month I had to fly to New York to run a 193 mile relay race. I knew about the trip for almost 6 months, but I didn't buy plane tickets until 3 days before, at 4x the cost. That is true dedication to the art of procrastination. You can call me somewhat of an expert, and it's one of my biggest shortcomings.

It's not because I'm lazy — in fact, the exact opposite is true; I'm incredibly productive. I move fast, but I single-task. I don't bounce around with a million things at once because of the incredible cost of context switching. Spending time planning trips, picking flights, and buying tickets just doesn't really seem important to me until it's 3 days away.

Paying $700 bucks for plane tickets, though, when they should have cost $200, isn't the best move, so I've spent the past month trying to get a handle on my procrastination.

## Fixing procrastination without willpower

Self-discipline and willpower — two words that aren't the solution. You only have a fixed amount of self-discipline and after it's been used up, you need to wait for it to recharge. If you "fix" a personality defect by brute-forcing with self-discipline, you'll be back in the same boat a week later.

I broke the cycle of procrastination and willpower exhaustion by automating the things I was putting off. Yup, I threw money at the problem, and it was cheaper than you think.

## Get a personal assistant

Hold off your judgment for a second. I'm still new to the assistant thing, but it easily falls into the 10 most effective life changes I've ever made. Within the first 24 hours of having an assistant, I delegated 2 tasks that had been marinating in my todo list for months. You know, the thing you say you're going to finally get done every time Monday rolls around? Well last Monday, it finally got done, and all I had to do was send an email. Mind blowing.

Total cost? $25 DOLLARS A MONTH. The service I use is FancyHands [fancyhands.com] and it paid for itself on the first day.

Think it's ridiculous? So did I, but it ended up changing my entire workflow. For example, I just had them book an AirBnB place for a trip next month. Next month! This is coming from a guy that was buying plane tickets days before a trip.

If it doesn't end up being useful you can just have your assistant email me about how much I suck; might as well get your monies worth.

**"You can automate almost all of the boring parts of your life, for less than 25 bucks and half an hour on Amazon."**

### Amazon Subscribe & Save and Emergency Deodorant

Maybe you've heard of it, maybe you haven't, but Amazon lets you set up reoccurring orders that get shipped automatically. It takes like 5 minutes to set up and has saved me so much frustration. I used to be the guy running to Target on the way out to buy a little thing I like to call "emergency deodorant." I've found myself standing in the deodorant aisle nonchalantly applying some Old Spice before taking it up to the register more times than I can count.

It was surprisingly hard to come up with ideas for using Subscribe and Save, so if you need some inspiration, check out the list of things Amazon automatically sends me. [hn.my/amzsteve] Fair warning: It's hilarious to open your door to a UPS guy carrying a 40-pack of toilet paper, IN THE ORIGINAL PACKAGING.

### Automate the things you hate

Even though I really enjoy writing, I despise proofreading and editing to the point where I rather just not write at all so I don't have to deal with the proofreading part. Nothing kills my flow more than having to re-read what I just wrote 1000 times.

Even though the book I'm working on has been an overall success, one of my biggest failures with the project has been not proofreading it. I'd fall into the cycle of just not writing, so I wouldn't have to proofread.

Instead of fighting it with willpower, I automated it. How? I tested both Mechanical Turk and Fiverr as proofreading platforms. Fiverr cost a total of $50 to have 62 pages edited. Just to give you an idea: editing 62 pages is my idea of a personal hell, but for less than a dollar/page, all of my hesitation, doubt, and negativity vanished.

### Try it out today

You can automate almost all of the boring parts of your life, today, for less than 25 bucks and half an hour on Amazon. And make sure to have your assistant email me telling me how much time I saved you. ■

---

Steve is the CTO at Twitpic and author of Scaling PHP Book [scalingphpbook.com]. He blogs about hacking his life and his startup hustle stories at *stevecorona.com*

# Things I've Quit Doing at My Desk

*By* JUSTIN JACKSON

Photo: Brendon Burton

We need to think of our desks as workstations.

In reality, we do all sorts of things at our desks that aren't really work-related (or affect our ability to produce our best work).

Here are things I'm trying to quit doing at my desk:

1 **Thinking:** Nobody does their best thinking sitting at their desk. When you reflect on your biggest "Ah Ha!" moments, how many of them occurred while you were staring at a screen? If you're like me, your best thinking happens when you're not at your desk: taking a walk, going and asking another person for help, drinking a coffee, in the shower, etc. Your desk is for executing; do your thinking elsewhere.

2 **Socializing:** When I sit down at my desk, I want to be in work mode. I want to prioritize my most important tasks, and then complete them with the fastest velocity possible. Socializing while I'm at my desk sullies the purity of the workstation. This is why the water cooler is actually a brilliant social construct: when you want to hang out, you can get up from your workstation and go to the socialstation. I think every office should have a socialstation, a place (or time in the morning) where team members can hang out and talk informally.

3 **Procrastinating:** Check Facebook, check Twitter, go on YouTube, check email, mindlessly read blog posts. I think that breaks, and downtime, are important in a work day. But again, I think maintaining the purity of my desk as a place where I work is important. If I need some "mindless" time, I think it's better to walk away from my desk and have a place and time limit where I do that. It's also important that we catch ourselves when procrastinating and ask ourselves, Why? Are we procrastinating because we're tired? Hungry? Bored? Are we stuck on a problem? Are we just feeling lethargic and need to get up and move around? Figure out what's at the source of your mindless net browsing, and deal with the problem.

4 **Sitting:** For the past 18 months I've been using a standing desk. I've realized that the best part isn't that I'm standing all day; it's that I'm not sitting. A standing desk allows you to stand, sit, lean, and put one leg up while you're at your workstation. Even better, I've felt more freedom to just walk away when I'm faced with a problem and need to do some thinking (or when I'm tired and need a break).

Many writers maintain a private writing hut. The hut has one purpose: it's the place they go to write. They don't do anything else there. Once they can't write any more, they go do something else. I think we need to think of our desks in the same way; these are places where we get work done. ■

---

Justin Jackson is a Product Manager at Industry Mailout [industrymailout.com], and the host of the Product People podcast [productpeople.tv]. He lives in Vernon, British Columbia in Canada. Follow him on Twitter @mijustin or visit his blog: *justinjackson.ca*

```
{
    join: 'Intensive Online Bootcamp',
    learn: 'Web Development',
    goto: 'http://www.gotealeaf.com'
}
```

Tealeaf Academy
an online school for developers

Learn Ruby on Rails | Level up Skills | Launch Products | Get a Job

# MEMSET®
HOSTING

Rent your IT infrastructure from Memset and discover the incredible benefits of cloud computing.

## MINISERVER™
CLOUD COMPUTE

From £0.015p/hour
to 4 x 2.9 GHz Xeon cores
31 GBytes RAM
2.5TB RAID(1) disk

## MEMSTORE™
CLOUD STORAGE

£0.07p/GByte/month or less
99.999999% object durability
99.995% availability guarantee
**RESTful API, FTP/SFTP and CDN Service**

SCAN THE CODE FOR MORE INFORMATION

MEMSET®
HOSTING

**CarbonNeutral®** hosting

THE BRITISH ASSESSMENT BUREAU ISO9001 UKAS MANAGEMENT SYSTEMS 4307
ISO 9001: Quality

THE BRITISH ASSESSMENT BUREAU ISO14001 UKAS MANAGEMENT SYSTEMS 4307
ISO 14001: Environmental

THE BRITISH ASSESSMENT BUREAU ISO27001 THE BRITISH ASSESSMENT BUREAU
ISO 27001: Security

PC PRO Excellence AWARDS 2011 WINNER BEST WEB HOST

Best Managed Service WINNER The ISPAs 2011 www.ispaawards.org.uk

Find out more about us at
www.memset.com

or chat to our sales team on
0800 634 9270.