# How to Become Batman

HACKERMONTHLY

Issue 12 May 2011

**Curator**
Lim Cheng Soon

**Proofreader**
Emily Griffin

**Illustrator**
Andy Fairhurst

**Printer**
MagCloud

**Contributors**

*ARTICLES*
Steve Schimmel
Mark Hughes
Mark Suster
Jason Cohen
Gabriel Weinberg
Ryan McDermott
Des Traynor
Chris Murphy
Gustavo Duarte

*COMMENTARIES*
Iain Dooley
Eric Heine
Catherine Darrow
Theo Jalba

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 Illustration:** Andy Fairhurst

# Contents

# Steve's Story: Googler 13

*By* STEVE SCHIMMEL

I WAS BORN IN 1972 to a poverty-level family in suburban Chicago. During the first few years of my life, I shared a 600 sq ft, 1-bedroom cottage with my parents, older sister, and dog. As a kid, I grew up helping my dad kill roaches and trap rats in his 1-man pest control business.

Inspired by my architect grandfather who dabbled in the stock market, I started playing the market when I was 15 years old. In 1994 at age 21, I graduated Magna Cum Laude & Dean's List from Babson College, an undergraduate business college outside of Boston. Shortly thereafter, I moved to San Francisco to find the stark reality of an uninterested job market. After a period of trivial and unsuccessful undertakings, I took to the streets out of desperation to "make something happen." My thought at the time was: "if my resume falls on the floor, nobody will bother to pick it up."

It was September, 1995. I had been in San Francisco just over a year and had nothing to show for it. I remember walking around the streets of San Francisco and seeing two individuals who made a profound impression on me. One was a panhandler who simply sat on a corner and directly asked for money. The other was a man standing on a milk crate wearing a sandwich board that said "Repent! The end of the world is coming." I was in a state of mind where I was open to anything. The things that struck me were that the first man had gotten to a point where his ego had been worn away and he was willing to simply and directly ask for what he wanted without beating around the bush. The second man believed so strongly in his convictions that he was willing to physically wear his message and present it to the world. By the end of the week, I had created a sandwich board expelling the virtues of my skills. One morning, I put on my best thrift store suit and boarded the 5am bus to the financial district with my sandwich board under my arm. I stood outside the Bank of America world headquarters, put the two-sided sign over my head and began passing out resumes. I was there for 12 hours. I passed out resumes as people rolled into work, when they went out for lunch, and as they left for home. This was one of the most humbling moments of my life. I stood out, exposed, bluntly asking for help and displaying my convictions. The response was amazing and really helped renew my faith in people. In the back of my head, I think I was expecting people to throw tomatoes at me (which my friend in New York said would have happened on Wall Street). Instead, many people took my resume and talked to me. A news crew even came.

In the end, this seemingly crazy idea lead to me getting a job as an associate equity analyst covering high-tech companies in downtown San Francisco. In my job, I was able to use my stock experience and education, but it did not take long to realize that I did not like analyzing companies. What I really wanted was to be involved with starting one.

In October of 1996 at age 24, I left my job in order to write a business plan for an idea I had relating to the relatively new phenomenon known as the commercial Internet. The idea revolved around creating a platform for pooling individual investor dollars to provide angel funding for fledgling companies and create a secondary trading market for these shares.

Unfortunately, after a few months I ran out of money and needed to look for a steady job again. I found a job opening for an entry-level, market research analyst at a small Internet company in San Francisco. Over a two-week period, I left messages on every single voice mailbox I could get at that company. I was never able to get a human on the phone or get a call back. One day, I randomly entered an extension off the main number and heard the message "Mark Goldstein's pager number is....". I knew from the website that Mark was the CEO of the company. What had started two weeks prior as a timid, pump myself up "you can do this" pep talk turned into a "somebody is going to talk to me, damn it!" When I got that pager number I called it immediately. Soon after I got a call asking "who is this?" I explained that I wanted the job his website had listed, but had not received a call back. He said that the reason nobody was there is that a few weeks earlier he had sold the company and its technology. He said he was downtown at a conference and to meet him at lunchtime. I ran over and he proceeded to tell me that he was a serial entrepreneur that would be starting another business at some point soon and to stay in touch....He emphasized that he was impressed that I was sitting across from him given the situation.

Time passed. Mark had yet to start his new company, and I was flat broke. I went to him yet again and asked for help. He simply picked up the phone and called a Venture Capitalist friend of his. The next day I was at breakfast with a VC. Two days later I was working at Netscape Communications (one of the companies responsible for the commercial Internet taking off) doing business analysis and portal deal modeling.

It was there that I honed my skills relating to understanding business models and became an expert in the Internet itself. I also proved my worth to the Netscape executives and built a reputation for myself there.

In 1999, America Online purchased Netscape and many executives left the company. One in particular was the Vice President of Business Development. He had decided to go to a very small company that was looking for venture funding so it could afford to try and build a company out of some very good technology it had developed. The company had been incorporated by two Computer Science Ph.D. candidates at Stanford University a few months prior. It only had the two founders and a few engineers working there. The Netscape VP joined the start-up in March of 1999 as employee 12 and hired me to help him build the business. In May of 1999, I joined the founding business team as lucky number 13. That company got its first and only round of Venture Capital, $25 million, a month or two later. The company was a Search Engine company that had only a few hundred thousand searches performed on it per day by at most a million users per month, mostly academics. The company had no revenue at the time. That company, now widely recognized as Google, is the world leader in search technology, bringing in billions of dollars in revenue per quarter by aiding many millions of users all over the world to find information on a daily basis.

1

2

1. Steve Schimmel, helping his dad's pest control business.
2. Steve Schimmel, doing whatever it takes to get a job.
3. Netscape CEO Jim Barksdale, Steve Schimmel, VC John Doerr, 1999
4. Steve Schimmel, Google Co-founder Sergey Brin, 2000
5. Steve Schimmel, Vice President Al Gore, 2001

During my career there, I negotiated our first $100k and $1million deals; was on the design team for the original ad program; ran a cross-functional external technology evaluation team; negotiated 3rd party technology licenses; was an all around go-to guy to just about every department that needed business help...and I founded and ran the Google Wine Club :-)

I could not have imagined a better job for myself. I was not constrained by any specific job description and was free to add value wherever it was needed. I truly got to be a Business Development Renaissance man.

When Google went public in 2004, it was one of the most successful IPOs in history. For me, that was a defining moment. I felt that I had made my impact, left my personal mark, and accomplished everything I had set out to do there. I left shortly after IPO to pursue other interests.

My confidence and follow-through, along with the chances awarded me by individuals who saw something in me and believed enough to give me a "shot," took me from poor kid to successful businessman "retired" by 32 years old. At no point did I ever compromise my integrity. I do things that I can be proud of. My colleagues and I lived by the motto of "don't be evil." It's not a gimmick. It's a philosophy of doing what's right and letting the money follow.

Having reached a level of financial success that awards me the freedom and flexibility that it has, I am now looking to share some of my knowledge and experience to benefit the next wave of those who aspire to do as I did.

...and that is my story. ■

---

Steve Schimmel was Google's 13th employee and founding business team member. He is currently an entrepreneurial advisor and angel investor and is starting a new company to unite Hollywood & Silicon Valley. Steve's blog can be found at *googler13.com*. Twitter @stevesf123.

3

4

5

## Commentary

By IAIN DOOLEY (dools)

THIS IS A story of a true hustler — and I mean that in a good way.

This guy got out there and hustled for a job. He hustled his way into Netscape by persistently calling and eventually "hacking" his way into contact with a well-connected business person.

Then when he was employed at Google, he went out there and hustled dollars that made them actually get some turnover.

It's a story we don't hear very often. It's a story about the people that make the money rather than the people that make the technology.

In a world where so much of the technology sector seems to be predicated on the idea that you build something cool, get users, and sort the "money stuff" out later, it's easy to forget that, at some point, someone's gotta get out there and actually make some goddamned money.

Having attempted to sell various technological services of my own for the past 4 years, I can whole-heartedly say that in my experience, building the technology is the easy part.

Being able to monetize it is a magical gift!

I'd also like to add that I find it pretty far fetched to refer to this success as "luck." Being a good salesman, being a good hustler, is all about being there. That's why CRM systems are such a vital sales tool — you need to make sure that every few months, you call your prospects, and if you don't sell to them, then you make an appointment to call back in 3 months, and so on.

Whether you're selling vacuum cleaners or selling your own services as an employee or contractor, you can't refer to every successful sale as "luck" — it is success based on persistent action. If anything you'd have to refer to people who hustle well and don't succeed as being unlucky, rather than the other way around.

# How to Become Batman

*By* MARK HUGHES

**Illustration:** Andy Fairhurst, *wildlifehoodoo.deviantart.com*

It depends. Which Batman, the one in the current film franchise, the one from the current monthlies, the one from the Justice League, etc., etc.?

I am going to make an assumption here, in order to best answer your question. We'll put aside the issue of Batman trained by ninjas in the films, or the question of whether in the comics Batman operates with sort-of-superpowers when interacting in stories alongside Superman and other such characters. By "become Batman" you mean the basic concept of Batman that we all could agree upon — a master of martial arts, of forensic and detective skills, of gymnastics, of science and chemistry, of history and geography, of the workings of organized crime, of criminal psychology and physiology. You mean a man with a suit offering protection against bullets and knives and electrocution, but which allows him to move as fast as an Olympian runner and acrobat.

The simple answer is, no. Unless you really boil Batman down to a very diluted level as just a really strong, fast, good fighter who can jump far and with good street smarts plus an education in crime and psychology, and who wears a mask and a lot of armor.

The genius of Batman is that it *pretends* to be realistic. It lets us convince ourselves that with enough money and training, *we* could become Batman, too. But it's still fantasy. It's just a fantasy that is more compelling and convincing—and thus more fun.

If you joined the military and became something like a Delta Force commando of the highest quality, while studying nights to get a double-major in criminal justice and psychology, with a minor in chemistry. You might also have time to take weekend courses in detective work and get a P.I. license. Then, after probably 10 years to reach all of those levels combined, you might be 28 (if you started right out of high school) and would then need to maintain your physical level while getting a job as a police officer in order to learn real crime solving and detective work on the streets and at crime scenes, to get the experience it would really take to be a master. Let's say you are so good it only takes you perhaps 3 years to become a top detective and expert in these regards. Now you are 31, and you just finished the most basic level of preparation you need to be an expert in just some of the most obvious fields required to match Batman.

Now you have to quit the force and develop a good cover story for yourself so nobody suspects that Batman might be the guy who is an expert in all of those fields Batman is a master at. You have to have made sure you lived your life never revealing your true feelings about crime and vigilantism, etc. In fact you need to cover it up unless you want to be arrested as a suspect the first time Batman comes around town. You need to spend some time doing dry runs to find your way around rooftops and fire escapes, practice running around at night in the shadows and not being seen, and presumably start practicing using your ropes and grappling hooks and other equipment you need for nightly patrols. Do some dry runs, make final preparations in case of emergencies, etc.

And you need to have been investing money and amassing a fortune the entire time, because the technology you'll need to even get close to a real-world version of Batman will cost millions of dollars. So you've done that, and now you start spending the money to get an armored suit full of electronics to communicate with assistants and have night vision and so on. You need a base of operations, so you buy one of those old used missile silos the military sells (yeah, they really do that, and it's pretty cool inside them) and turn it into a secret headquarters for the computers, monitoring equipment, car, bike and other equipment you need for your vigilante life.

Conservatively, you should probably be about 32 at this point. And you are only about to go out on your first night as Batman. Okay, it's taken longer than expected and been pretty hard. And honestly you are not quite as much a master of all fields as Batman, but at least you got the basics and are pretty well trained and smart and equipped. So off you go, looking to stop crime...

...and you're looking. And looking. Oh, wait, you hear police sirens or you get a transmission from picking up the police radio calls, there's a domestic disturbance in progress.... Well, that's not really what Batman

does, so you let that one go to the cops. Then you get another call about a robbery, ah ha! Finally Batman is going into action! You run across those rooftops, swing across to another roof — whoa, crap, that was a lot more dangerous than it looks in the comics! But you're booking it, running flat out and probably hitting, what, a good 10 miles per hour? Maybe less actually because of having to dodge things and stop at the edge of the roof to swing down again.

Anyway, there you are, rooftop to rooftop, and it occurs to you that the cop cars are so far gone now that you barely hear the sirens. So you think "Hmm, no wonder the real Batman has a car, this rooftop thing looks cool but I'll never make it in time to stop a crime that isn't happening within a block or two."

And you don't — make it in time, that is. The first few nights, you keep showing up and the robberies or shootings or whatever are already over, and you realize that this makes sense because most reports about crimes are only after it happens, not while it's taking place. And you also remember that as a cop, you almost never just walked up or drove up accidentally right where a crime happened to be taking place. In fact, you were just one of several thousand cops in your city, and most of you never just stumbled right across a significant crime in progress.

By your second week, you are getting unhappy that 90% of the crimes you've even seen up-close are just pathetic junkies buying crack from another pathetic junkie selling drugs to support his/her own habit. And nothing makes you feel less like Batman than scaring sad, homeless crackheads. You tried to chase down a kid who you saw punch a lady and take her purse, but you can't really pursue that kind of thing by running on rooftops, you gotta do it the hard way by chasing him on foot down the sidewalk… in your full Batman costume, where everybody can see you. People are taking photos on cell-phones, and, yep, there's a cop car at the intersection and he saw you, and now he has his lights on and it's you he's after. Great, you have to let the kid go so you can run down an alley and climb up a fire escape to the roof to get away.

At last, week three, you get lucky: an armed robbery, right there across the street! You leap down onto the hood of their car, cape over the

**Illustration:** Andy Fairhurst, *wildlifehoodoo.deviantart.com*

windshield just like in *The Dark Knight Returns*. And a teenage kid in the passenger seat fires a shotgun though the windshield in panic, blasting your torso.

You are wearing armor, though, haha! So it merely shreds your costume and knocks you off the car onto the street, but man that hurts! And it takes your breath away just long enough for the car to speed off. You get up, angry and just in time to see everyone taking your photo again and staring at your shredded outfit. Then the police come around the corner, and you run off again but this time you are injured because although the armor stopped the slug, it still bruised you and broke a rib. You are fast, but not fast enough this time. The police draw their guns and order you to stop. You turn and grab for the smoke pellet on your belt to help hide your getaway, but unfortunately for you the cops see you reaching for something and open fire...and your suit's armor is already a mess from the shotgun blast earlier. Uh oh.

When you wake up in the ICU, your mask and costume are gone, you're in a lot of pain, but the doctors successfully removed the bullets and re-inflated your lung. The downside is the set of handcuffs trapping you in the bed. As a master detective, you can of course easily pick the lock on the cuffs to escape, but on the other hand the staph infection you caught after surgery is pretty bad and you feel like s**t. So you wait until night to sneak out — except you fall asleep on your pain meds, and wake up the next morning to the police coming to pick you up and take you to the infirmary at the state prison. Where you will spend a month recuperating until they can transfer you to the county jail for your first court appearance. During which your only comment to the judge is, "I guess it's not really possible to become Batman."

Na-na-na-na-na-na-na-na-*na*! Bat*man*! ■

Mark Hughes is a screenwriter and lifelong reader of comics. He's also a huge Batman fan, and regrets crushing the hopes and dreams of would-be superheroes around the world. :(

# Why Startups Need to Blog

*By* MARK SUSTER

BLOGS. WE ALL read them to get a sense of what is going on in the world, peeling back layers of the old world in which media was too scripted.

By definition, if you are reading this, you read blogs. But should you actually *write* one if you're a startup, an industry figure (lawyer, banker), or VC? Absolutely.

This is a post to help you figure out why you should write and what you should talk about.

### 1. Why

If you care about accessing customers, reaching an audience, communicating your vision, influencing people in your industry, marketing your services, or just plain engaging in a dialog with others in your industry, a blog is a great way to achieve this.

People often ask me why I started blogging. It really started simply enough. I was meeting regularly with entrepreneurs and offering (for better or for worse) advice on how to run a startup and how to raise venture capital from my experience in doing so at two companies. I was having the same conversations over-and-over again, and I figured I might as well just write them up and make them available for future people who might be interested. I never really expected a big audience or ever thought about it.

I had been reading Brad Feld's blog [feld.com] and Fred Wilson's blog [avc.com] for a couple of years and found them very helpful to my thinking, so I honestly just thought I was giving back to the community.

The results have been both unexpected and astounding. Within 2 years I was getting 400,000 views per month and had been voted the 2nd most respected VC in the country by an independent survey of entrepreneurs, The Funded, and sentiment analysis. I know that I have not yet earned these kudos based on investment returns (although my partners have. GRP Partners last fund is the single best performing VC fund in the US [prequin data] for its vintage year). But it speaks volumes to what people want from our industry:

- transparency
- accessibility
- authenticity
- thought leadership
- advice

I'll bet your customers, business partners, or suppliers would love similar.

**“Be a thought leader. Don’t blog to your friend, but blog to your community.”**

## 2. What

I often get the question from people, “I’d like to blog, but I don’t really know what to talk about?” or “I’m a new entrepreneur — why would I offer advice on how to run a startup?”

You wouldn’t. You shouldn’t.

Not only would it be less authentic, but if you’re a startup, it’s not immediately clear that other startup CEOs are your target market. They’re mine because I’m a VC. I care about having a steady stream of talented startup people who want to raise money thinking that they should talk to me in addition to the top others whom they’re targeting.

Whom do you want to target? Who are your customers, partners, or suppliers?

My suggestion is to blog about your industry. Think *Mint.com*. In their early days they had an enormously effective blog on the topic of personal financial management. They created a reason for their customers to aggregate on their site on a regular basis. They became both a thought leader in the space as well as a beautifully designed product. They created inbound link juice on topics that drove more traffic to their site. Type “personal financial management” into Google. *Mint.com* is the second result. Smart.

Think Magento. They are an open-source and SaaS provider of eCommerce solutions. They are the fastest growing player in the world in this space. They achieved all of this before they raised even a penny of venture capital. eCommerce is an enormously competitive search term. Yet type it into Google and the third result is Magento. Magic. They did it by creating a blog, a discussion board and hub for eCommerce advice and information.

So you developed a product for the mommy community? Blog on that topic. Do you have an application that helps mobile developers build HTML5 apps? You know your blog topic. Do you have sales productivity software? Obvious. Check out SalesCrunch posts [hn.my/salescrunch]. Blog to your community. Be a thought leader. Don’t blog to your friend (that might be a separate Tumblog or something), but blog to your community.

If you’re going to pump out regular content that is meaningful, you obviously need to blog about a topic in which you’re knowledgeable, thoughtful, and passionate. If you’re not all three of these things in your industry, then I guess you’ve got a broader problem. Honestly.

“Do a brainstorming session and create a list of 40-50 topics that interest you.”

So my biggest recommendation of “what” to blog is a series of articles that will be helpful to your community. If you’re a lawyer, blog on a topic that would be helpful to potential customers. Show that you’re a thought leader. Scott Edward Walker does an excellent job at this. It’s the only reason I know who he is. I had seen his blog and his tweets and then was interested to meet him IRL.

Do a brainstorming session and create a list of 40-50 topics that interest you. Write out the topic and maybe even the blog title. Keep the list electronically.

Struggling to come up with enough topics? Take one topic and break it up into 10 bite-sized articles. It’s probably better that way anyways. I wanted to write about the top 10 attributes of an entrepreneur. I wrote it all in one sitting and then broke it up into 10 separate posts. It kept me busy for 3 weeks! Each one ended up taking on a life of its own; as the comments flowed in for post 1, I had more thoughts to add to post 2, and so on.

## 3. Where

You need a blog. Duh. If you’re a company and if hanging it off of your company website makes sense for the link traffic, go for it. If you are head of marketing at a company and keep a more generalized blog (in addition to your company blog) so that you can influence brands and agencies, it can be separate.

I chose for my blog to be independent of my firm, GRP Partners. The reason is that I wanted to be free to say what I was thinking independently of my partners. My views don’t always represent theirs and vice-versa, even though we’re pretty like-minded (we’ve worked together for 10+ years). I chose a title that represented a brand that I wanted to emphasize: Both Sides of the Table. I chose it because I thought it would represent who I am — mostly an entrepreneur, but somebody with investment chops. I wanted to differentiate.

So. People keep asking me, “Why would you write on TechCrunch?” I guess I would have thought it was obvious. Apparently not. People say, “Aren’t you driving traffic away from your own blog?”

Facts:

- I don’t really care about total page views or uniques other than as a measure of whether I’m improving. I don’t sell ads.
- I do care about “share of mind,” which means that I want fish in the pond where the people whom I want to speak with hang out. I know a certain number hit my blog. But I’m not so arrogant (or successful) as to think they come all the time. So I take my show on the road. If I can write about a topic which I’m passionate about and double or triple the number of people who read it, that’s gold dust. That’s why I never stopped anybody from taking my feed and republishing.
- As it happens, since I began writing at TechCrunch my viewership has continued to go up, not down. I always publish on my own blog the day after it runs on TC. I want the historical post there. A large number of readers on my site get it from Feedburner or newsletter feed.

**“If you try to make everything too perfect, you’ll never hit publish.”**

- I also get a lot of inbound links from writing here. I try to make any inbound links to my blog authentic to the story. But each story has driven thousands of views.
- The majority of my traffic still comes from Twitter. TC posts = more Twitter followers = more conversion when I do write on my own blog = more Feedburner/newsletter subs = more traffic. It’s an ecosystem. Simple.

So once you have a blog, a voice, and a small following — don’t be shy about writing some guest posts for target blogs. Remember: for you that’s likely not TC — it’s the place your community hangs out.

## 4. How

**Be authentic.** Don’t try to sound too smart or too funny. Just be yourself. People will see who you are in your words. If you try to make everything too perfect, you’ll never hit publish. If you try to sound too intelligent you’ll likely be boring as shit. Most blogs are. I hate reading blowhards who try to sound like they’re smarter than the rest of us. Be open and transparent. Get inside your reader’s minds. Try to think about what they would want to know from you. In fact, ask them!

Don’t be offensive. It’s never worth it to offend great masses of people. But that doesn’t mean sitting on the fence. I have a point of view and I’m sure sometimes it rankles. But I try to be respectful about it. Sitting on the fence on all issues is also pretty boring. And don’t blog drunk. Or at least don’t hit publish ;-) Mostly, have fun. If you can’t do that, you won’t last very long.

**How do I get started?** First, you’ll need a platform. I use WordPress. Some people swear by SquareSpace. There are the new tools like Tumblr and Posterous. I’ve played with both and they’re pretty cool. They’re more lightweight and easier to use. Importantly, they’re more social. It’s much easier to build an audience in social blogging platforms the way you do in Twitter or Facebook. Then you need to decide whether to use the “hosted” version or the “installed” version. At least that’s true in WordPress. The advantage of the hosted version is that it’s easier to get started. The disadvantage is that you can’t install a lot of additional tools that use Javascript. I started with the hosted version and then migrated to an installed version so I could use Google Analytics and some other products.

You then need a URL. It’s true you can be something like msuster.typepad.com but that’s kind of lame so I wouldn’t recommend it. Just get a real URL. I think it’s important to think about what image you want to portray when you pick your URL name. It doesn’t need to be short. You’re not trying to build a consumer website. My website is a pretty long URL, but people manage to find it. Much of my traffic is through referring websites and/or social media. Some search. What are YOU trying to convey? What will be your unique positioning? Don’t just write a carbon copy of what somebody else is doing. That’s boring.

**“The #1 thing that kills 95% of blogs is that they do 5 or 6 posts in rapid succession and then peter out.”**

**So I wrote a post, now what?** Don’t blow your load on your first post. Slice it up enough to do many posts. I think most blogs are between 600-1000 words/post. Once you’re written a few posts, don’t try to make the floodgates open at once. Slowly build your audience. Make it organic. If you write good content consistently, you’ll build an audience over time.

The #1 thing that kills 95% of blogs is that they do 5 or 6 posts in rapid succession and then peter out. It’s lame to go to a blog where this happens. And then 8 months later they do the obligatory post saying, “OK, I’m going to be more committed to blogging now!” and then another 4 months go by. If you’re really not going to write that often at least don’t put dates on your posts.

But if you write good stuff, put in an effort, and keep going — it’s a marathon — you will see results over time.

**How do I build an audience?** If you build it, will they come? No. A blog post is just like a product. First it needs to be good. And then you need to market it. It doesn’t just happen. You should be subtle about how you market it, but market it nonetheless. If you’re too squeamish to ask for help in promoting it or to do so yourself, then you’ll never build an audience. (You’ll also likely not make it as an entrepreneur. Sorry. But that’s true.)

The obvious starting point is to email a few friends and let them know you have a new blog. Don’t be overbearing — just an email saying, “wanted to let you know about my new blog.” I also recommend you put a link to it under your email signature (in a color other than black). You also should have it be what your Twitter bio links to.

Every time I write a post, I send it out on Twitter. I try to send out the Twitter link when more people are online. Over time I’ve found out that I get better clicks at 8:30-9:30am Monday-Friday, so that’s when I tweet a lot of my stuff. I’ll frequently send two tweets — East Coast and West Coast. Not everybody sees the first one. Social media is ephemeral.

Because I’ve built my Twitter following slowly but steadily and authentically over time, I get very high click-through rates (and thus a high Klout score — currently 74). I get about 4% CTR on every tweet in the AM, and it’s actually higher because if I assume only 33% of my followers on online the CTR is closer to 12%. Interestingly, if I had sent one Tweet at 5:30am (to get East Coast time) and another at 8:30am, I get 4% CTR both times. So it’s hard to argue you shouldn’t tweet twice if you have a geographically distributed following.

How do I know my stats? I use *awe.sm* (disclosure, I’m an investor) which is the best tool I know of for tracking: it tracks each individual share behavior (it creates unique URLs for each tweet), plus it also separates out tweets from Facebook shares and from “Retweets” that come from somebody clicking on my blog, etc. It also tracks who tweeted the link so you will know who your most influential social followers are.

Make sure your blog has Tweetmeme or similar to make it easier for readers to tetweet. Also, make sure to sign up with Feedburner. That way people who want to get your blog by RSS and/or email can do so. Make sure your blog also has a Follow Me on Twitter button so people who find you can easily follow you.

“If you plan out what you want to write about in advance, then it’s really about writing.”

## 5. When

People often ask how I blog so much, since they don’t think they can do it themselves. If you write about something for which you’re both knowledgeable and passionate, I’ll bet you can pump out more than you think.

I usually blog at 10pm or on airplane flights. I never blog at work. Like you, I don’t have the time. I have board meetings, company pitches, internal partner meetings, etc. Hell, I often can’t even get to email during the day. So it comes out of TV time, which means I’m not missing anything. Occasionally if I really want to blog and I have a date or too much work, I just set my alarm for 5:30am. Yup. It’s not that hard if you make a commitment to it.

What would it mean to you and your business if you could: increase your inbound traffic, enhance your company and personal brand, and meet new influential people who suddenly know who you are? If you want these things, they are available to you for the cost of some time and effort.

If you plan out what you want to write about in advance, then it’s really about writing. Create topics, then headings to structure your article; you’ll notice on this one I started with “Why, What, Where, How,” and then I later added “When” and “What Next.”. Structure helps enormously.

I write for about 45 minutes to 1 hour in the first pass. I usually then re-read, edit, spell check, and add links. This usually takes another 20-30 minutes. I then always add an image. I think this is a nice touch. Just staring at text is a bit boring and I find that the image can add humor and/or drive people in.

## 6. What Next?

Then there are comments. You have to respond to comments. Do yourself a favor and install Disqus. It makes a huge difference in driving a comment community.

First, it’s the most fun part of blogging. It’s addictive, like Twitter. It’s where you exchange ideas with other people. It’s where your community gets to know you. It’s where you build loyalty and relationships. I have met many people in person who were first commenters on my blog. I find it frustrating if I leave comments on somebody’s blog and they never respond. If somebody found your blog and took the time to comment then they’re like a customer who should be cherished. Responses to them are like customer retention. It’s also where you’ll learn. People will tell you when you’re full of shit. ■

---

Mark Suster is a 2x entrepreneur who has gone to the Dark Side of VC. He joined GRP Partners in 2007 as a General Partner after selling his company to Salesforce. He focuses on early-stage technology companies.

# When You Want To Quit Because It's Just Not Worth It

*By* JASON COHEN

I'VE BEEN THERE. It sucks.

You know most startups fail "only" because the founders stop working on them, and often, it's because it's emotionally draining. I don't care who you are or how strong your ego is, you will have these moments — perhaps a continuous stream of moments — when you can't take it anymore.

I literally cannot remember the number of times I was so overwhelmed at Smart Bear that I almost threw in the towel. Close the bank accounts, close the doors, turn off the website, bounce the email, and just stop.

Sounds dramatic, but it's no exaggeration. You'll hit these walls, too; maybe a little commiseration will help you get through it.

Of course you expect these moments to happen at the beginning of startup life — when you're least confident, have the worst product, and the least knowledge about your customers and the market.

You see, the pain is not limited to the beginning of the venture. It's still there three years in, despite real revenue, profitability, customers arriving everyday, and a great team.

Since that is not obvious, I'd like to share a personal story.

Four years into Smart Bear I had several employees getting paid decently (which at a bootstrapped startup is hard to do!) and a product that people were buying — plus, we were doing around a million-a-year. Life was good!

I was working on my first true "enterprise sale."

I was negotiating our biggest order to date — something like $200,000. Actually, "negotiating" is the wrong word because I don't believe in price negotiation, even with enterprise sales (the one area that most people claim must include automatic discounting).

The person with whom I was negotiating wasn't the end user, nor the boss, nor boss's boss, nor anyone in that chain of command. See, big companies have entire departments devoted to dealing with vendors like you and me, and when it comes to negotiating, these departments harbor terrorists with titles like Procurement Manager or Strategic Sourcing Manager.

I say "terrorists" because they use fear tactics to get their way, yet they have no power other than fear. Imagine the worst stereotype of a salesman — the greasy used-car type, except instead of selling you something, their job is to beat a discount out of you.

Now, to be fair, many vendors do take advantage of large companies — overcharging (because "They can afford it!"), or promising one thing to the users and sneaking something else into the invoice.

But mostly it's because of the traditional enterprise sales dance, reminiscent of the lumbering mating dance of the great blue whale. The vendor asks for too much money; the client is astonished at the price. Both calmly explain that this is a deal-breaker. Then the vendor capitulates 30% but only if the client signs a three year maintenance contract (which they wanted anyway). The deal is struck.

(This tradition continues because of perverse, wasteful incentives. The vendor's salesman likes this because sometimes he gets away with a high price which pads his commission check. The Procurement Manager likes this because he can show his superiors how much money he's "saved" the company.)

So big companies need a Defender of Evil Vendors, I get that. But that's not enough for these guys; it feels to me like an attack, not a parry.

This is always how the conversation would go:

PM: What kind of discount are you offering?
**Me: We don't discount; instead we put our pricing on our website so there's no misunderstanding.**
PM: Well, I'm going to need some kind of discount. How about 30%?
**Me: As it says on our website, we don't discount.**
PM: But I'm buying 400 seats!
**Me: Yes, and we already provide a nice discount for bulk orders, which is already included on the invoice and documented on the website.**
PM: You don't understand, I always get a discount. I've done business with 47 other vendors and all of them give me at least 20% off.
**Me: There's always a first!**

So far it's actually ok — I'm the one refusing to plod through the mating ceremony, wanting to skip right to the wedding night. I expect push-back.

But here's where it gets nasty. I remember sitting there on the phone getting lambasted for my intolerable ignorance about the Way It Works. I was told "I have no business selling anything to anyone." My obstinate ignorance is a deal-breaker because of what it implies about my company in general—after all, if I don't even understand the purchasing process there's no chance in hell my software's going to work! Furthermore, despite my ignorance I'm unwilling to listen to the rules, to learn, which means there's no hope for me.

I'll never forget how this ended:

PM: OK, that's it, you give me no choice. I absolutely cannot approve this deal, and furthermore I'm recommending that we never work with your company in any capacity. At this point, even if you gave me a discount I would still reject it.

Here's where I'm supposed to unleash my intellectual fortitude. I won't capitulate, will I? I won't let this guy insult and bully me, will I? C'mon, I'm the strong-willed confident entrepreneur with the stoic well-argued voice of reason, and he's the sleezeball with the tedious day-job — surely I'll laugh as his words roll off me like water off an oiled duck's back.

Just the opposite. I felt like throwing up. He's right — who do I think I am? I'm a geek playing in the big boy's house and I don't know what the hell I'm doing. I have these naïve ideas about how the world should work and how people should treat one another, and that's just silly. And it shows. And now this guy is going back and spewing vitriol at the other folks in the company whom I actually like and worked really hard to earn their trust.

But it's over. They've seen through me. It's just a matter of time before others do, too. That's the end of deals like this.

Why am I doing this anyway? This is supposed to be fun and fulfilling but at this moment, as we say in Texas, I feel like ten tons of shit in a two-ton bag. What I like is writing code — why am I even trying to play this sales game? Why not just go get a job where I only worry about whether or not I can write code — because I sure as hell can do that — and let the natural salesmen do all this crap?

Is the money worth it? What money, we're still bootstrapping and I still don't get a regular salary. Is the promise of money worth it? Worth these feelings of inadequacy?

After days (yep, days) of fretting like this, it converted from despair to anger. Who the hell is this guy?

**“It’s about sticking through the tough parts, whatever your personal foibles or weaknesses.”**

Some asshole who isn’t good enough with money to be an accountant, too slimy to sell cars, this guy whose only skill is to be a jerk, some guy who has never had to make payroll or take a risk or put himself out there, this schmuck is going to tell me I’m the one who isn’t good enough, I’m the one who has no business selling software?

Worst of all, I’m letting him make me feel like a pile of shit!

Well, if you’re waiting for the big moment where intellectual reasoning finally defeats weak, irrational emotions, I’m sorry to disappoint you, because that moment never came. I know it’s dumb and illogical, but there it is. It’s trivial and baseless, but I still carry that experience in a corner of my thoughts. That’s how emotions work.

By the way, this guy turned out to be totally full of shit. He had, in fact, no power to stop the deal. When I finally got my main contact from that company on a conference call with the PM, the conversation was literally:

My Guy: So, what’s holding up procurement’s approval?
**PM: Nothing, just some paperwork, we’ll have it done by Friday.**

All of that angst for nothing. Son of a bitch!

Years later I was on site at this company and I finally met this guy face to face.

I still felt small.

Want to say I’m weak? Or he’s strong?

Who cares, the point is: Getting through this slog of a thing that’s a startup — or anything difficult and worthwhile — doesn’t require that you’re always confident or stoic or smart or right or wise. You don’t need to match the emotional stability you see from the big bloggers (which is mostly a façade anyway).

It’s about sticking through the tough parts, whatever your personal foibles or weaknesses.

Living through it, not beating it. I never have, to this day, “beaten” that PM, not emotionally, not if I’m being truly honest.

I’m not saying tenacity is all it takes. Just that without it, you’ll stop. It’s so easy to stop. There are so many reasons to stop.

And that — stopping — is how most little startups actually fail. ■

---

Jason is the founder of three companies, all profitable and two exist. He blogs on startups and marketing at *blog.ASmartBear.com.*

## The business book for entrepreneurs: over 30 interviews with startup founders.

Follow along as a Y Combinator alumni interviews other startup founders to understand how they deal with challenges such as finding cofounders, raising money, getting users, staying motivated, and hiring.

STARTUPS
OPEN SOURCED

NEW

stories to inspire & educate

Grooveshark Reddit GitHub Foursquare Airbnb Weebly Greplin AppSumo Wufoo LittleAppFactory MixPanel LikeALittle Djangy Divvyshot Justin.TV Blippy Bump WePay DailyBooth Gobble KISSMetrics Omnisio Cloudkick Noteleaf OneLlama Octopart Crowdbooster Listia Hipmunk Indinero OrangeQC Camino Real One

*Jared Tame*

## What people are saying

"This is great! Having interviewed over 300 founders, I'm impressed - if you want to understand how some of today's most impressive startup founders got to where they are, this book will deliver."
- Andrew Warner, Mixergy.com

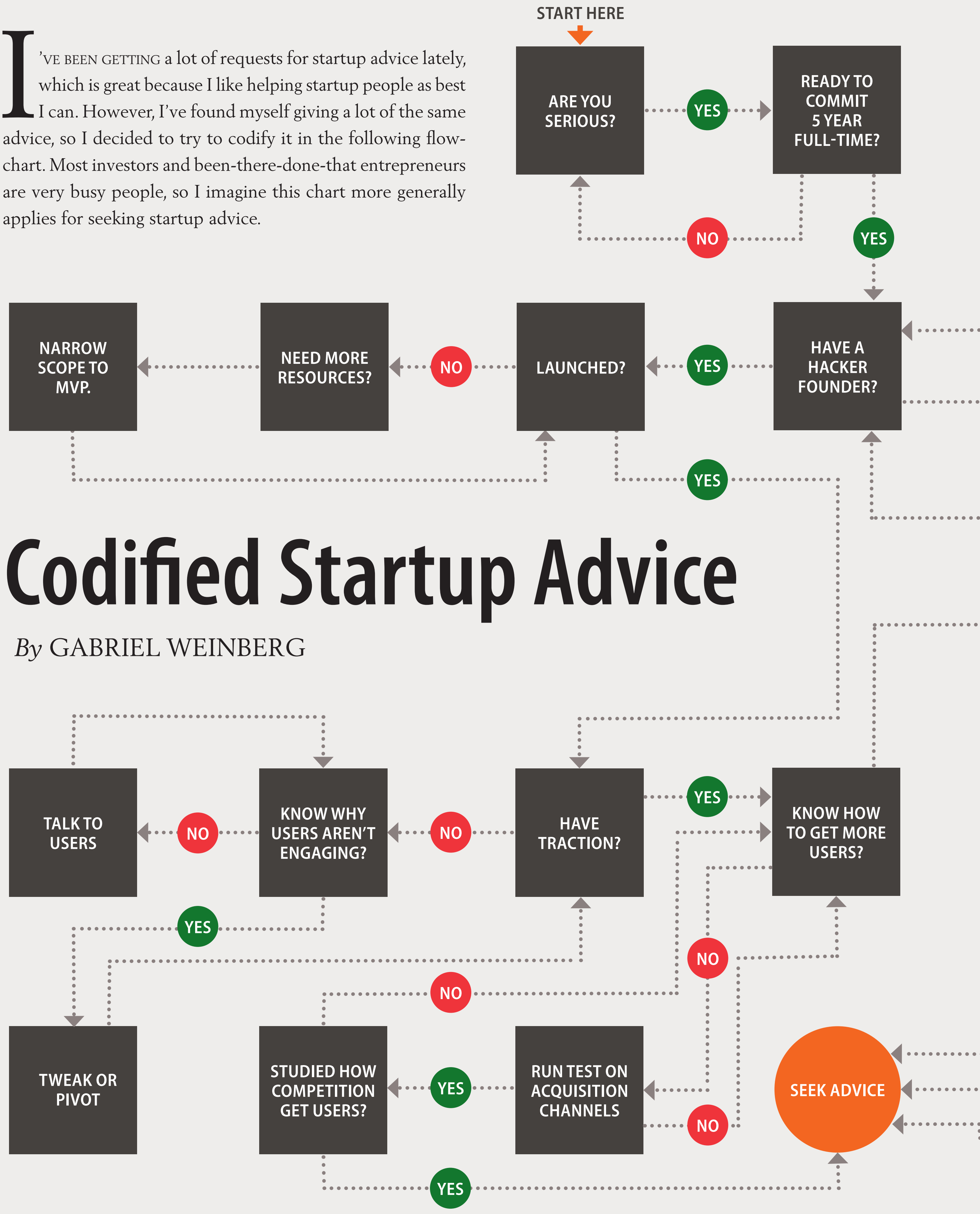
"One of the richest collections of authentic and inspiring stories."
- Avichal Garg, BluePrint Labs cofounder

"I think this book will end up causing more students to try to become entrepreneurs."
- Niniane Wang, Cofounder Sunfire Offices

**Special discount for Hacker Monthly readers!**

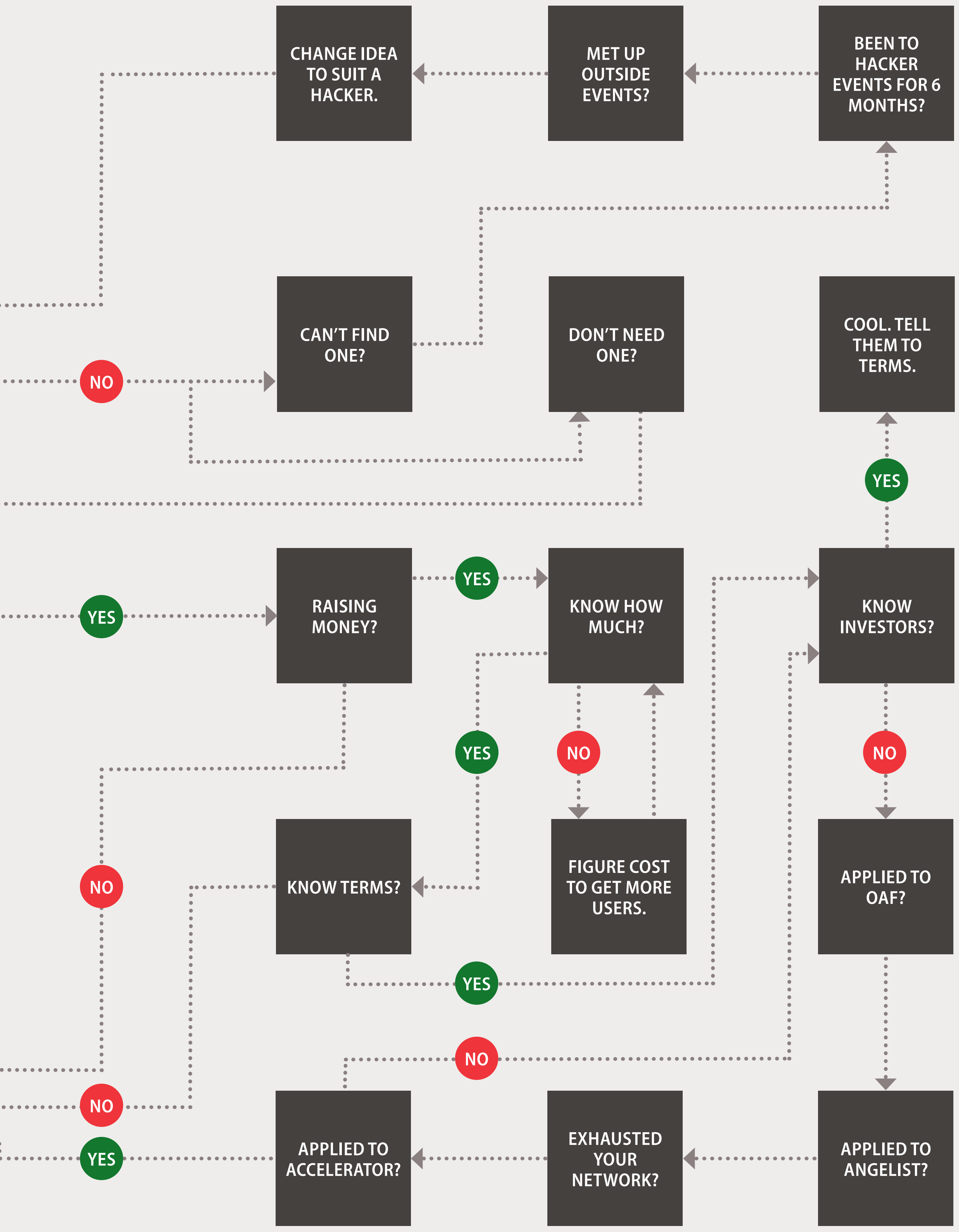
First 250 May edition readers receive 15% off at **StartupsOpenSourced.com**, just use coupon code **"hmonthly511"**

I've been getting a lot of requests for startup advice lately, which is great because I like helping startup people as best I can. However, I've found myself giving a lot of the same advice, so I decided to try to codify it in the following flowchart. Most investors and been-there-done-that entrepreneurs are very busy people, so I imagine this chart more generally applies for seeking startup advice.



# Codified Startup Advice

*By* GABRIEL WEINBERG


CHANGE IDEA TO SUIT A HACKER.
MET UP OUTSIDE EVENTS?
BEEN TO HACKER EVENTS FOR 6 MONTHS?
CAN'T FIND ONE?
DON'T NEED ONE?
COOL. TELL THEM TO TERMS.
NO
YES
RAISING MONEY?
YES
KNOW HOW MUCH?
KNOW INVESTORS?
YES
NO
NO
YES
KNOW TERMS?
FIGURE COST TO GET MORE USERS.
APPLIED TO OAF?
NO
YES
NO
NO
YES
APPLIED TO ACCELERATOR?
EXHAUSTED YOUR NETWORK?
APPLIED TO ANGELIST?

# My Fellow Geeks, We Need to Have a Talk

*By* RYAN MCDERMOTT

MY FELLOW NERDS, geeks, hackers, designers, makers, builders, and DIYers, there is something very, very wrong with our culture right now.

We're jackasses to one another.

No, we're not! Right? Geeks help each other out! Well, sometimes we do, but most of the time, we're the most abrasive, critical, non-cooperative community of people I've ever encountered. How many websites are there like the daily wtf? Or clients from hell? Or photoshop disasters?

How many blog posts have been written about how everybody is doing everything wrong! Stop using comic sans, god dammit! What are you, illiterate? "Grammar nazis" are engrained into our culture, and disregarding something somebody has said because of minor misspelling is a common, accepted, and even expected practice.

"Tables? What is this, the 1990s? Ha ha ha!"

"This design looks like MySpace gorged itself on Friendster and vomited all over Geocities!"

"You're using the default hashing algorithm in mysql instead of bcrypt? You should probably give up and see if they're hiring down at the local concrete crushing factory because you, sir, have absolutely no business whatsoever touching, much less programming, a computer."

"God I hate the arduino. It's not real hacking. Using the arduino is no different than going down to target and just buying whatever it is that you're trying to build. Arduino is for idiots who can't actually program because they're too stupid to figure out how to hook a parallel cable into a bread board. God, kids these days are fucking idiots."

These are all embellished caricatures of comments I've actually seen.

What the hell, guys? Why is this attitude so common? And it extends beyond just criticizing other designers/hackers/makers. Why does every single nerd I meet just hate "hipsters"? Or "bros"?

Are we all back in high school again?

I want to share the experiences I've had with other communities, specifically sports people. I've shared this before, so if you've already heard it, please excuse me. When I was about 16 years old, I was a huge (literally, I was physically huge) nerd. I'm not sure if it was because of the tiny school that I went to, but somehow, I managed to befriend some skateboarders. After a few times going with them to the local skatepark and helping them film a "sponsor me" video, I decided that I should learn to skateboard myself, so I bought a board.

“Instead of outlining all of the ways that your peers are terrible at programming, give people constructive criticisms.”

This was probably hilarious to watch. A big huge nerd who was certainly more comfortable sitting behind a python interpreter than in front of a skate ramp was hopelessly rolling around in circles in the parking lot.

Except nobody told me that I sucked at skateboarding, or that my form was terrible, or that I should give up on it. In fact quite the opposite. One day at the skatepark I was sitting off to the side just watching everybody else and kind of wishing that I wasn't there. One of my best friends, Steve, came up to me to ask what I was doing.

“Oh, man, I suck at this. I'm just going to practice at home or something. I don't want to get in anybody's way.”

“What? Dude, you look like a weird-o just sitting over here, and you're not going to learn anything by just staring at that thing. If I ever catch you sitting on this bench again, you're not invited to the skatepark anymore.” (There were probably quite a few more vulgarities, but this was the gist of it)

I have never seen this attitude in the geek community. It's always been “You're doing it wrong, and you should give up because you suck at it,” or “if you're not using $hip_new_language, then you're a loser.”

Guys, why do we do this? Most of us were nerds when we were younger, and this attitude of “you're not cool enough to be in the $cool_designers or $cool_programmers club” is exactly the type of stuff we had to deal with. It's the high-school lunch room all over again.

So I have a challenge for you: for the next 30 days, be more like my friend Steve. Instead of outlining all of the ways that your peers are terrible at programming because they're not doing manual memory management, or that your customers are illiterate morons and how dare they have the audacity to question your work, give people constructive criticisms. If their design is bad, tell them what they can do to improve it. If there code is bad, offer to help them patch it and make it better. If there spelling or grammar is off, just let it go.

And please, stop it with the irrational hatred of “hipsters.” Most “hipsters” that I know love geek culture and would be elated at the opportunity to have somebody show them around a laser cutter. ■

Ryan is an independent developer living in Phoenix, Arizona. He currently spends most of his time working on thingist.com, a social website for keeping and sharing lists.

## Commentary

*By* ERIC HEINE (sophacles)

I'M REMINDED OF something a wise man once told me (HNified a bit):

In every pairwise conversation there are 6 people:

1. Alice
2. Bob
3. Who Alice thinks she is.
4. Who Bob thinks he is.
5. Who Bob thinks Alice is
6. Who Alice things Bob is.

Perception plays an absurdly large part in communication, as do nonverbal cues. Frequently we adjust our message based on feedback we get from the listener. Those lacking "social graces" or communicating in just text on the internet don't get these cues, so the message comes out "harsher".

Complicating this, there is a lot of baggage each person attaches to words, phrases and general styles of questioning/commenting. So one person's harsh may be another's "in to it".

One example of all this I have experienced:

One time at a vendor show, me and some colleagues were in a small demo, presented by a sales guy and a few engineers from the company. During the Q&A, I started questioning the engineer pretty intensely with questions like:

Does it do $X? Why not? Do you plan on adding it?

(These are actually pretty neutral questions)

Then about another aspect I was really into some possibilities of:

Can I use it for $Y? Can I make $Y happen by this? What happens if I do $Z? How about if I work around that limitation like this and get $Y + $Z effectively?

(these are not neutral questions, they are me geeking out)

So after the demo some people thought I broke the engineer and ripped him a new one with the second set of questions, because I was rapid fire asking questions towards a goal. One engineer thought it was a fun "play with an idea time". The other engineer thought I was severely criticizing his work.

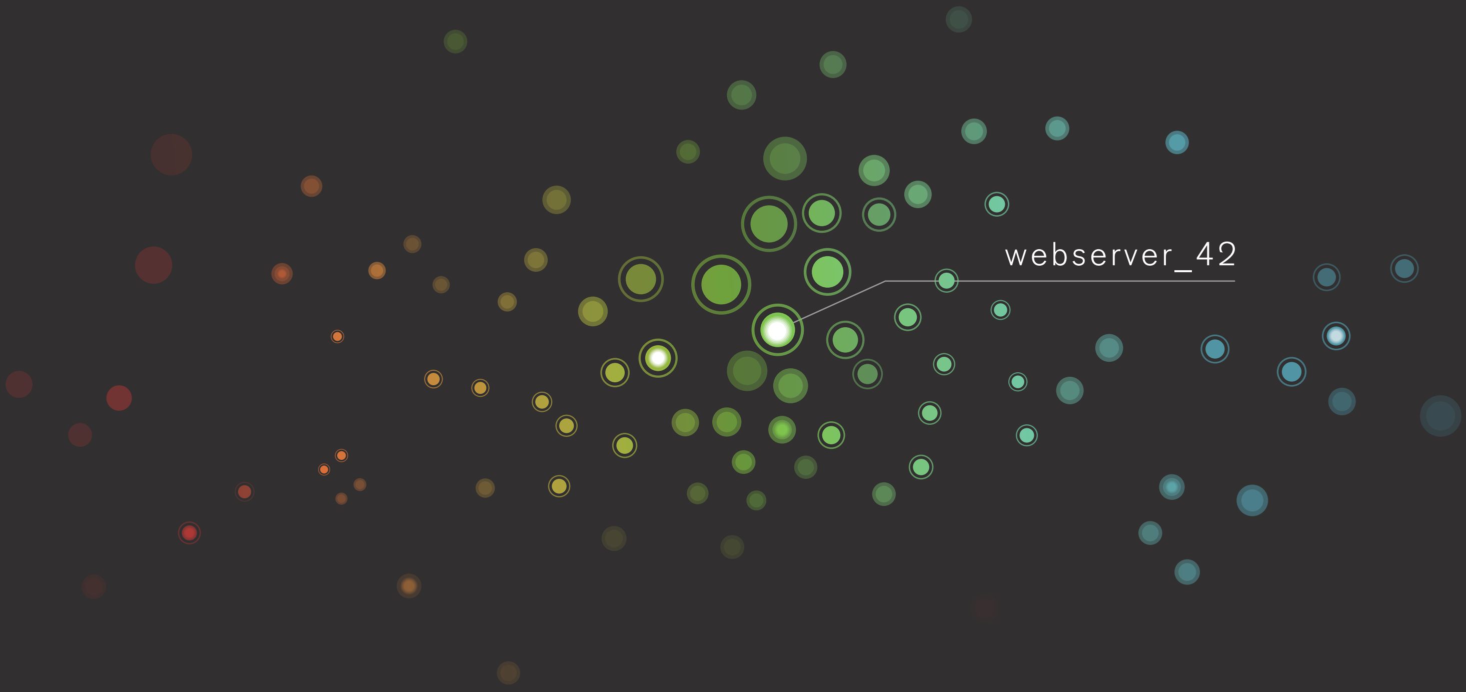
The sales guy and several of the audience members thought I was being unduly harsh by asking about the feature $X. Apparently this was a contentious issue that I knew nothing about. The engineers and others thought nothing of those questions.

Similarly: I frequently get frustrated when people wrap up valid criticism in fake nice BS. I don't want to hear "great thing, what if instead you did this". I really would rather just hear "What about this other method? Why not use that?" or even "Dude, 10s of googling would have shown you the flaws in that". Because an honest self assessment includes the fact that I don't know everything, and that many (most) of the things I come up with have also been thought of by other people, who may have found flaws in that reasoning.

I guess my point is there is a lot more than just "nerds are mean to each other" going on.

These are your servers

These are your servers on Cloudkick



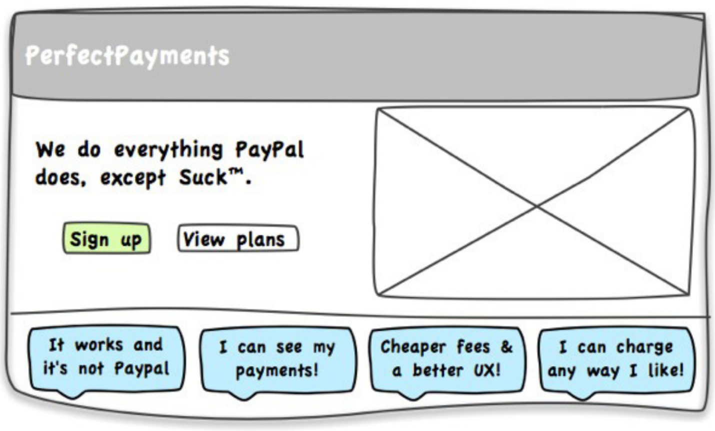
Any questions?

cloudkick.com
415.779.5425

support for 8 clouds + dedicated hardware

cloudkick
the best way to manage the cloud

# Designing Your Sign-Up Page

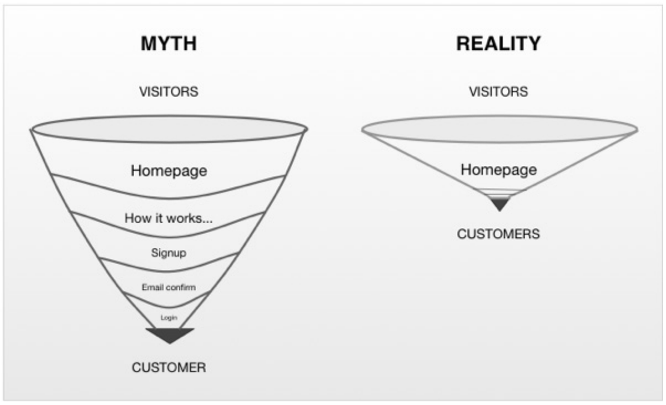

PerfectPayments
We do everything PayPal does, except Suck™.
Sign up
View plans
It works and it's not Paypal
I can see my payments!
Cheaper fees & a better UX!
I can charge any way I like!

*By* DES TRAYNOR

Turning a visitor into a user of your application is difficult. Turning a user into a customer is even harder. Much has been written about designing for sign-up, focusing on funnels, metrics, cost-per-acquisition, etc. A lot of the guides focus on the assumption that lost customers are a result of poor form design, bad layout choices, and visual design-related blunders. Unfortunately, that's very rarely the case.

## Sign-Up Funnels: Myth and Reality

In some transactions you lose customers at each step in the process. In retail stores, this can be measured. Gap knows that the more customers who try things on, the more they'll sell, so they encourage their visitors into changing rooms. They know that by having an assistant on standby with a size up, a size down, and a different colour, they're more likely to close the sale. They know how to attract and acquire customers.



The only equivalent of this is shopping carts where you can measure conversions by watching customers move from one step to the next. It's a lovely idea that a purchase works in a perfectly measurable funnel: "View Product -> Add to Cart -> Go to Checkout -> Enter Details -> Confirm Order -> Success". That's true for some cases, but most of the time there's five tabs open looking at different prices/charges/delivery dates/refund policies/taxes, etc. And the purchase might happen once they get home, or when they get paid, in a separate visit, possibly recorded as a separate visitor. That said, shopping cart funnels still offer heaps of information that should inform design decisions.

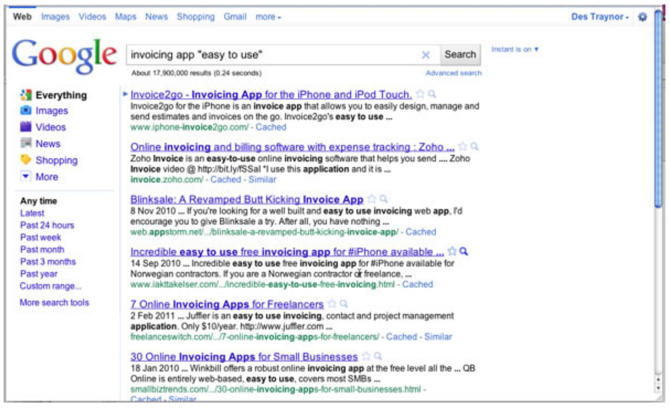
The problem is that none of this works well when designing marketing sites for a web app about to launch. There are different forces at play.

## The Cheapest App Money Can Buy

Web applications are rarely a commodity. Commodity web apps are things like file format conversions, URL shorteners, Twitter pic uploaders, or File-hosting sites. They're disposable, one-off transactions and the user doesn't really care what URL they get out of the exchange. They're tough rackets to be in.

Users aren't looking for the cheapest app. They want the fastest, most reliable, best supported app. The only thing that matters when designing your sign-up page content is making sure that it serves their needs. Are you convincing users that your product does something useful for them? Does it make them rich, make them laugh, pique their interest, or get them laid? [hn.my/content]

If you offer me an invoicing solution and print "easy to use" everywhere on your site, it means nothing to me. Just like everyone thinks they have a good sense of humour, everyone thinks their software looks good and is easy to use.



About 17,900,000 results (0.24 seconds)

Invoice2go - Invoicing App for the iPhone and iPod Touch.
Invoice2go for the iPhone is an **invoice app** that allows you to easily design, manage and send estimates and invoices on the go. Invoice2go's **easy to use** ...
www.iphone-invoice2go.com/ - Cached

Online invoicing and billing software with expense tracking : Zoho ...
Zoho **Invoice** is an **easy-to-use** online **invoicing** software that helps you send .... Zoho **Invoice** video @ http://bit.ly/fSSal "I use this **application** and it is ...
invoice.zoho.com/ - Cached - Similar

Blinksale: A Revamped Butt Kicking Invoice App
8 Nov 2010 ... If you're looking for a well built and **easy to use invoicing** web **app**, I'd encourage you to give Blinksale a try. After all, you have nothing ...
web.appstorm.net/.../blinksale-a-revamped-butt-kicking-invoice-app/ - Cached

Incredible easy to use free invoicing app for #iPhone available ...
14 Sep 2010 ... Incredible **easy to use** free **invoicing app** for #iPhone available for Norwegian contractors. If you are a Norwegian contractor or freelance. ...
www.iakttakelser.com/.../incredible-easy-to-use-free-invoicing.html - Cached

7 Online Invoicing Apps for Freelancers
2 Feb 2011 ... Juffler is an **easy to use invoicing**, contact and project management **application**. Only $10/year. http://www.juffler.com ...
freelanceswitch.com/.../7-online-invoicing-apps-for-freelancers/ - Cached - Similar

30 Online Invoicing Apps for Small Businesses
18 Jan 2010 ... Winkbill offers a robust online **invoicing app** at the free level all the ... QB Online is entirely web-based, **easy to use**, covers most SMBs ...
smallbiztrends.com/.../30-online-invoicing-apps-for-small-businesses.html -
Cached - Similar

## Good Content Sells

Content is king on marketing pages, yet often they're the most content scarce pages on the web. A fancy tilted screenshot and a big red button doesn't convince me of anything, except your ability to rotate images. Here's an incomplete brain dump of questions you need to answer if you're selling invoicing.

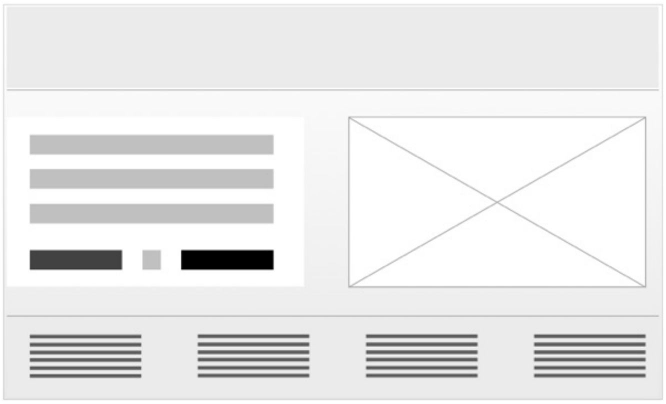
Your app looks simple to use, but is it powerful enough to handle my set-up? Does your software know about the taxes/rules about how invoices are handled in my country? Will I be your first serious customer, or do have experience dealing with firms of my size? Do other firms like mine use your software? How long have you been running? How do I know you won't wrap things up in a few months? Can I trust you guys? Can I talk to you guys? How do I know you are legit? Do you offer good support?

Begging your visitors to take your free trial is often the wrong approach. A free trial costs time and doesn't answer all the questions. Screencasts are good, but they're usually not enough.

**“This goes beyond “Content is King” and isn’t really about design. It’s about the ability to sell.”**

Bear in mind also that invoicing is a well-defined problem. People know what to expect of invoicing software. It gets harder when you’re pitching a solution to an unknown problem, or re-defining an existing problem. Take FlowApp [getflow.com], for example. Flow aims to change the way I work. This means Flow needs to convince me that they know how I work, convince me there is a problem with it, explain how they solve it, why it works, who it’s aimed at, and then go ahead and answer all the other questions I listed earlier. No wonder they’ve yet to launch a marketing site. This is hard stuff.

## What You Can Include



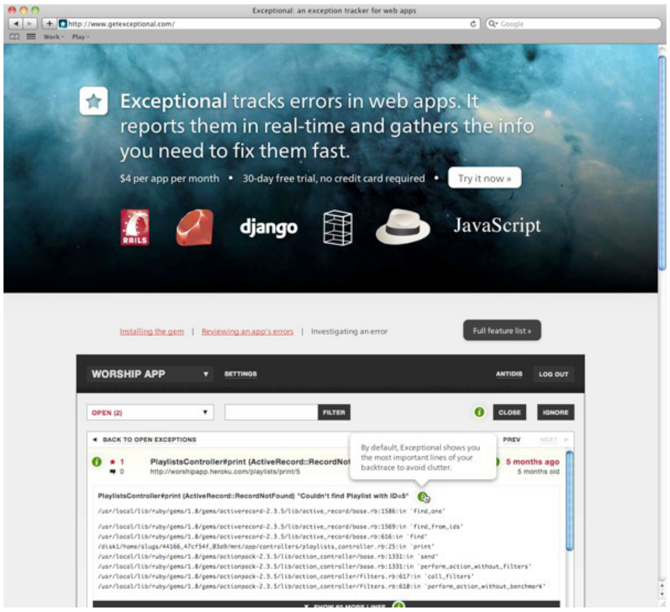
Before you open “Ye Olde Web App” template and routinely drop in the obvious components, think about how you would sell this to someone. What sort of information pushes people over the line. If you were trying to impress me at a conference, what would you say? Easy to use? Heard that before. Convenient? I’d hope so. You need more than that to attract interest, here are some ideas…

- What interesting figures can you aggregate (100,000 hours billed, 2,000 companies managed, 3.6 Terrabytes of data secured)
- Who’s currently using it, and for what?
- Who is the team behind the application?
- How long has the application been worked on?
- What significant changes has the app been through while alive? What is the story behind the application?
- How can you be contacted? Can you be called? How good is your support?
- How secure is my information?

“But many popular web apps don’t do this!” you might say. Firstly, well established web apps are feeding off their recommendations and the established reputation of their creators. When you’re just getting started, things are different. You might not have an audience yet, so unlike the big names, you need to win trust and respect. You might get the benefit of the doubt, but you can’t rely on it.

Secondly, many of the big name web apps have content heavy homepages. Look at Highrise or Basecamp, Mailchimp, Campaign Monitor, they’re not scrimping on information. This goes beyond “Content is King” and isn’t really about design. It’s about the ability to sell. Even when your product is stunning and sells itself, you still need to sell me on your company, your support, your features, your future. That’s why it doesn’t surprise me to see companies like 37Signals continue to add content such as the Yes page [highrisehq.com/yes], or the customer support happiness page [smiley.37signals.com]. There will be more to come.

## The Exceptional Homepage



Exceptional tracks errors in web apps. It reports them in real-time and gathers the info you need to fix them fast.

$4 per app per month • 30-day free trial, no credit card required •

By default, Exceptional shows you the most important lines of your backtrace to avoid clutter.

```
PlaylistsController#print (ActiveRecord::RecordNot
http://worshipapp.heroku.com/playlists/print/5

PlaylistsController#print (ActiveRecord::RecordNotFound) "Couldn't find Playlist with ID=5"
/usr/local/lib/ruby/gems/1.8/gems/activerecord-2.3.5/lib/active_record/base.rb:1586:in `find_one'
/usr/local/lib/ruby/gems/1.8/gems/activerecord-2.3.5/lib/active_record/base.rb:1569:in `find_from_ids'
/usr/local/lib/ruby/gems/1.8/gems/activerecord-2.3.5/lib/active_record/base.rb:616:in `find'
/disk1/home/slugs/44166_47cf54f_03a9/mnt/app/controllers/playlists_controller.rb:25:in `print'
/usr/local/lib/ruby/gems/1.8/gems/actionpack-2.3.5/lib/action_controller/base.rb:1331:in `send'
/usr/local/lib/ruby/gems/1.8/gems/actionpack-2.3.5/lib/action_controller/base.rb:1331:in `perform_action_without_filters'
/usr/local/lib/ruby/gems/1.8/gems/actionpack-2.3.5/lib/action_controller/filters.rb:617:in `call_filters'
/usr/local/lib/ruby/gems/1.8/gems/actionpack-2.3.5/lib/action_controller/filters.rb:610:in `perform_action_without_benchmark'
```

The Exceptional marketing site [getexceptional.com] has gone through many revisions over the past three years. One lesson we've learned is that the more useful information we can give visitors, the better our conversion rate. The numbers back this up. Each piece of content is there to answer a question. Our wall of logos lets you know that we are for real, and have 6,000 people relying on us. Our status site lets you know take performance seriously. Our blog lets you see our customers and what they use us for. Our features page details every single thing the app can do for you. Our screenshots offer tooltips to explain what you're looking at. The point being every piece of content is there to answer a question, and bring you one step closer to sign-up. When we discuss the site, it's from a "What else would persuade people to sign-up, if they knew about it" approach. Our last addition was the row of supported languages & frameworks, and again we're seeing positive results. Allan Branch of Less Accounting reported that adding a phone number increased conversions by 1.8%. We'll look at that next.

## Metrics For Marketing Pages

Metrics are great for telling the what, but not the why. No matter how many Google Analytics tutorials you follow, you're never going to find the killer regex that checks for "user actually being interested in the app". Your best bet there is to start finding people who you know should be interested in your software, try to sell to them, and find out what works and what doesn't. If you see a lot of inertia, lots of "I can't be bothered", then you have two choices. Either target new consumers in the market (i.e. the people who have no solution at present) or identify a new feature that users will pay for. Be wary of the latter tactic though. The world is full of people would would buy it if….

As I've said before, the truth with funnels and A/B tests is that they're of little value during the early days of a web app, when traffic isn't significant. I've seen many A/B test junkies wait a long time for customer "B" to even show up.

When you don't have the volume, go for the personal approach. When you can no longer go personal, then analyse the volume. At every step you need to ask yourself "Is every single thing on my website selling the product?" and "Is there anything else I can include that will help?"

Looking through Mixergy interviews [mixergy.com] with successful founders, you could be forgiven for thinking you needed a popular blog to be able to release an app successfully. The correlation here isn't coincidental. A popular blog is an indication that the writers can sell things, whether it's their credo, depth of thought, technical skills, or opinions about business. It's surely no surprise that if they can sell themselves, they can also sell their software.

The thing is, we're all salesmen, and whether we like it or not, we're always selling. We just don't wear the shiny shoes. ■

---

Des Traynor is the User Experience Lead at Contrast. In this role he works primarily with start-ups helping them define a product strategy, identify their customers, and design solutions to attract and delight them. Des regularly writes his thoughts about his experience in design and the business of web applications on the Contrast blog [contrast.ie/blog], He can be found on Twitter as @destraynor.

Reprinted with permission of the original author. First appeared in *hn.my/signup.*

# Advanced Git Techniques

*By* CHRIS MURPHY

## Finding That Issue

### "The Pickaxe"

So you're doing a code review on a piece of the program you don't normally touch, and you notice that there's a new property of your `Person` class. Since when did we start tracking `social_security_number` for our workout app? The pickaxe is part of the internal `gitdiffcore`, but you won't find it by name in the git commands we all use. Instead, pass in the `-S` string with a term to search for. In our case, let's search for that new variable:

```
$ git log-Ssocial_security_number
```

This will show you each time that the string "social_security_number" appeared or disappeared from the repository history. It can be very useful, as long as you know what string you're looking for. The pickaxe is one of a few different transformations that the `diff` uses, but it's really the only one I've used. I'm not even sure if the others, other than order, are intended for the end-user.

You probably know the `-p` flag for `git log`, which shows the `git diff` inline with the log messages. You can combine that with the pickaxe so that you can see a little context while you search. If you use the pickaxe for presenting changes, you might find the `--pickaxe-all` switch useful. That'll show you the diff for all of the changes in the commits that the pickaxe finds — not just the actual lines that the pickaxe recognized. Try them out:

```
$ git log -Ssomething -p
$ git log -Ssomething -p --pickaxe-all
```

Pretty powerful stuff.

### Git Blame

What if you don't know exactly what you're looking for, but you know the file you want to look at? Git has a tool just for you: `git blame`. Try it out:

```
$ git blame <file>
```

This will show you who is responsible for each of the lines in that file. In other words, the last commit that touched the line. This is really useful, but don't use it as an excuse to yell at people. Unless they did something really awful. Okay, not even then.

By the way, this is one of two times that I sometimes like to use GUIs for git. The lines for `git blame` can be pretty long (even if you modify the output with switches). Just say:

```
$ git gui blame
```

That'll do the exact same thing, except in a nice GUI interface, which will make it easier to see the whole line and navigate between commits.

### Git Bisect

Bisect is an awesome idea: a combination of binary search, interactive, and testing that can only result in happiness. Suppose a user reports a bug in your program, and you figure out how to reproduce it. You write a test for it, and sure enough, it fails on current build. You have no idea when the bug was introduced, but bisect is here to help. If you happen to know a version where the bug didn't happen, you can tell bisect to start there. If not, just tell it the beginning of your history.

```
$ git bisect start
$ git bisect good <sha>
$ git bisect bad master
```

Git will do all the hard work for you. You just have to run your tests at each point that git prompts you, and tell it whether it passed or not:

```
$ git bisect bad (or good)
```

Thanks to the power of binary search, you'll find it pretty quickly. I haven't found a need to use this feature in anger, but in my simple tests, it worked like a charm. You don't even need to have a bug to test it out — just lie to git about what is good and what is bad.

If you have the ability to run your tests and get a Unix return code (0 for success, non-zero for failure), git bisect run will actually do all of the work for you!

## Branch Management

When you're making good use of git branching, you'll notice that the easiest thing to do is to just merge in your branch to master and move on. But if you do a lot of branching, and you're constantly pulling in changes from other branches, that'll become really ugly, really quick. It may be worthwhile to prevent all of those merge commits, depending on your team strategy. If you want to do this, git (as always) has plenty of ways to help.

### Cherry Pick

git cherry-pick is useful for bringing in a selection of commits from any branch. I use it when there are selected bug fixes or feature adds in another branch that I need to pull in to my working branch.

As with all git commands, you can tell git which commit you want to cherry-pick in a variety, but to be honest, I almost always explicitly call out the sha-1. Use git log or gitk to find the sha-1, then:

```
$ git cherry-pick <sha-1>
```

Here again we see the value of keeping your commits logically separated. Otherwise, a simple one-liner turns into manual file editing, which is just asking for mistakes. Why not let git do the work for you?

### Rebase (onto)

You already know about rebasing from Git 102 [hn.my/git102]. In that case, we just rebased the master branch, but we can do so much more. If you decide you don't want a merge commit to show in master, you can do the following in your feature branch:

```
$ git rebase master
$ git checkout master
$ git merge feature-branch
```

Since you already did the work to rebase the changes from master with your changes in feature-branch, when you merge into master, it'll be a simple, fast-forward merge. No merge commit.

The next step is to use rebase across multiple branches. If you often float back and forth between fixes, this one is for you. Let's see what we can do with --onto:

```
git rebase [-i | --interactive] [options]
[--onto <newbase>] <upstream> [<branch>]
```

You feed git rebase --onto three things: the place where git should play commits onto and the two places to compare commits from. So if you do a simplified version of the man page's example:

```
$ git rebase master~5 master~3 master
```

In our case, master looks like this:

```
A-B-C-D-E-F (F is HEAD)
```

Git looks at master and master~3 (three parents from the HEAD) and figures out what commits are in master, but not in master~3. In this case, that would be D, E, and F. It applies those commits onto your first argument, which is master~5 (commit A). Using git's friendly message terminology, it rewinds HEAD to master~5 (it actually does exactly what a `reset --hard` would do), then plays back commits D, E, and F. The newly rebased master looks like this:

```
A-D-E-F
```

Put simply, `rebase` killed B and C from your tree. That's pretty awesome. Now we can tackle the more complex case of working around branches.

Look at the example from the man page [hn.my/rebase] with `topicA` branched from master and `topicB` branched from `master`.

```
                          H---I---J topicB
                         /
                E---F---G  topicA
               /
  A---B---C---D  master
```

The page says to run `git rebase --onto` master `topicA topicB` to get commits H, I, and J applied to `master`. Since `topicB` was branched from `topicA`, it has all of the commits in the picture in its history. But `topicA` doesn't have any of the commits of `topicB`, so `rebase` will find that H, I, and J need to be applied to your target, which is `master`.

You should note that the man page's previous example with topic and next is actually identical. The only difference is that `master` has moved on since you branched next. This demonstrates that the commits are played back onto `master`'s HEAD (since that's what you told it to do) — it has nothing to do with where the first branch (next or `topicA`) branched from.

## Recovering from Mistakes

### git reset

Sometimes, you just can't gracefully back out of a messed up manual `merge`, `rebase`, or plain ol' corrupted working tree.

```
$ git reset --hard <something>
```

That'll reset your current branch, index, and working tree (for less than all three, try `--mixed` or `--soft`) to whatever you tell it. Most often, I'm running it against `origin/some-branch`, to reset to the last pushed state.

Ok, so you probably already knew that one. But what about the untracked working tree files that get spewed around sometimes (p4merge on Windows leaves .orig lying around all.the.time.)?

### git clean -fd

`clean` will kill off all of those pesky files that you haven't added to git, and want gone. Since git is so safety-conscious, you have to force it with `-f`, and you probably want to tell it to get rid of the directories, too, with `-d`. With `reset` and `clean`, you should never have to `rm -rf` and `re-clone` again.

### The reflog

All this playing around with hard resets, rebases, and whatnot might scare you. What happens if you hard reset two days of work? No matter how careful you are, this will happen eventually. Well, git hasn't really destroyed everything permanently, at least until the garbage collector comes around. If you accidentally trash something, just type `git reflog` to see a list of the last operations. Here's an example:

```
$ git reset --hard HEAD~2
$ git reflog
7298e1e HEAD@{0}: HEAD~2: updating HEAD
ca9164c HEAD@{1}: Oh man this is the best algorithm
ever. It sorts in constant time!!
$ git reset --hard ca9164c
```

*phew*. You obviously shouldn't rely on this, and if you're digging into the reflog every day, you should probably re-evaluate your git strategy. But it's great to have that safety net there when you need it!

### Filter-branch

One last tip. If you're working in an open-source environment (pushing to GitHub, for example), you'll want to be careful not to commit any sensitive information. As we've seen, any commit is part of the repo forever. That is, unless you rewrite history. That's what `filter-branch` is there for. As you might imagine, GitHub has an excellent article [help.github.com/removing-sensitive-data] to show you how to protect yourself.

By the way, anything that's hanging around in `reflog` waiting to be `gc`'d won't make it out if you push to GitHub, so don't worry about that. You can still kick off `gc` manually if necessary. ■

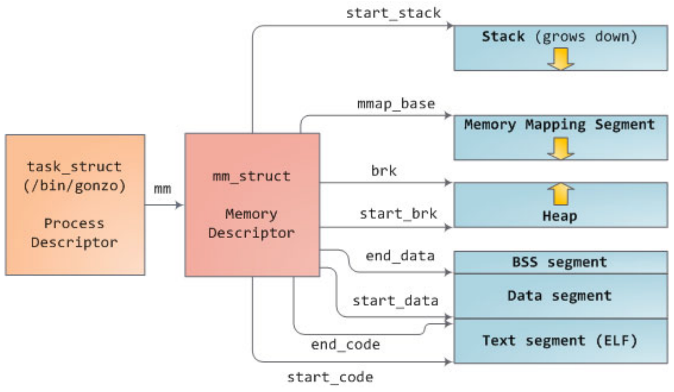
Chris Murphy lives near Boston and is a graduate of Bowdoin College. He is a displaced Python and Java man trying to write C# by day, and loves sports, cooking, and learning.

Rediscovering mathematics:
You do the math
Shai Simonson
MAA

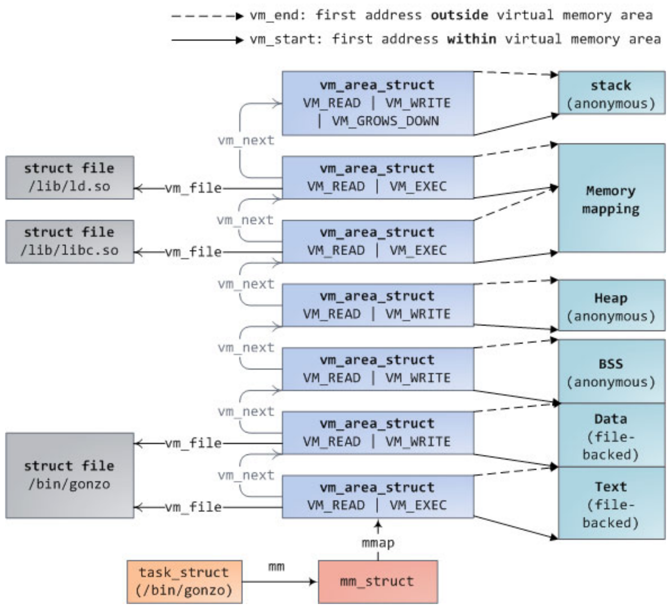
# How The Kernel Manages Your Memory

*By* GUSTAVO DUARTE

After examining the virtual address layout [hn.my/virtual] of a process, we turn to the kernel and its mechanisms for managing user memory. Here is gonzo:



Linux processes are implemented in the kernel as instances of task_struct, the process descriptor. The mm field in task_struct points to the memory descriptor, mm_struct, which is an executive summary of a program's memory. It stores the start and end of memory segments as shown above, the number of physical memory pages used by the process (rss stands for Resident Set Size), the amount of virtual address space used, and other tidbits. Within the memory descriptor we also find the two work horses for managing program memory: the set of virtual memory areas and the page tables. Gonzo's memory areas are shown next:



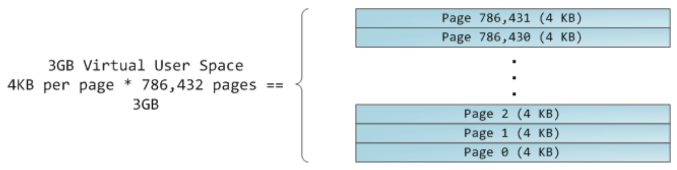
vm_end: first address outside virtual memory area
vm_start: first address within virtual memory area
vm_area_struct VM_READ | VM_WRITE | VM_GROWS_DOWN
vm_area_struct VM_READ | VM_EXEC
vm_area_struct VM_READ | VM_EXEC
vm_area_struct VM_READ | VM_WRITE
vm_area_struct VM_READ | VM_WRITE
vm_area_struct VM_READ | VM_WRITE
vm_area_struct VM_READ | VM_EXEC
vm_next
vm_file
struct file /lib/ld.so
struct file /lib/libc.so
struct file /bin/gonzo
stack (anonymous)
Memory mapping
Heap (anonymous)
BSS (anonymous)
Data (file-backed)
Text (file-backed)
mmap
task_struct (/bin/gonzo)
mm
mm_struct

Each virtual memory area (VMA) is a contiguous range of virtual addresses; these areas never overlap. An instance of vm_area_struct fully describes a memory area, including its start and end addresses, flags to determine access rights and behaviors, and the vm_file field to specify which file is being mapped by the area, if any. A VMA that does not map a file is anonymous. Each memory segment above (e.g., heap, stack) corresponds to a single VMA, with the exception of the memory mapping segment. This is not a requirement, though it is usual in x86 machines. VMAs do not care which segment they are in.

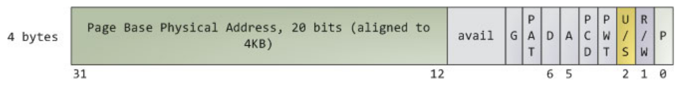
A program's VMAs are stored in its memory descriptor both as a linked list in the `mmap` field, ordered by starting virtual address, and as a red-black tree rooted at the `mm_rb` field. The red-black tree allows the kernel to search quickly for the memory area covering a given virtual address. When you read file `/proc/pid_of_process/maps`, the kernel is simply going through the linked list of VMAs for the process and printing each one.

In Windows, the `EPROCESS` block is roughly a mix of `task_struct` and `mm_struct`. The Windows analog to a VMA is the Virtual Address Descriptor, or VAD; they are stored in an AVL tree. You know what the funniest thing about Windows and Linux is? It's the little differences.

The 4GB virtual address space is divided into pages. x86 processors in 32-bit mode support page sizes of 4KB, 2MB, and 4MB. Both Linux and Windows map the user portion of the virtual address space using 4KB pages. Bytes 0-4095 fall in page 0, bytes 4096-8191 fall in page 1, and so on. The size of a VMA must be a multiple of page size. Here's 3GB of user space in 4KB pages:



The processor consults page tables to translate a virtual address into a physical memory address. Each process has its own set of page tables; whenever a process switch occurs, page tables for user space are switched as well. Linux stores a pointer to a process' page tables in the `pgd` field of the memory descriptor. To each virtual page there corresponds one page table entry (PTE) in the page tables, which in regular x86 paging is a simple 4-byte record shown below:

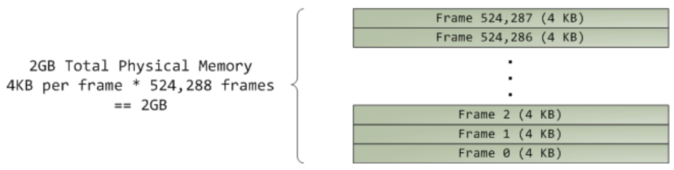


Linux has functions to read and set each flag in a PTE. Bit P tells the processor whether the virtual page is present in physical memory. If clear (equal to 0), accessing the page triggers a page fault. Keep in mind that when this bit is zero, the kernel can do whatever it pleases with the remaining fields. The R/W flag stands for read/write; if clear, the page is read-only. Flag U/S stands for user/supervisor; if clear, then the page can only be accessed by the kernel. These flags are used to implement the read-only memory and protected kernel space we saw before.

Bits D and A are for dirty and accessed. A dirty page has had a write, while an accessed page has had a write or read. Both flags are sticky: the processor only sets them, they must be cleared by the kernel. Finally, the PTE stores the starting physical address that corresponds to this page, aligned to 4KB. This naive-looking field is the source of some pain, for it limits addressable physical memory to 4 GB. The other PTE fields are for another day, as is Physical Address Extension.

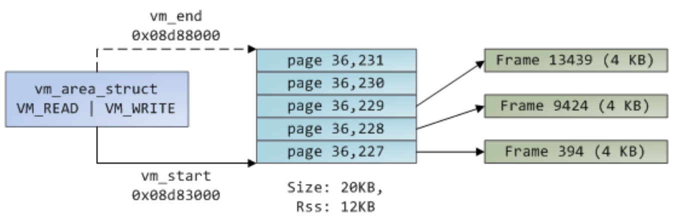
A virtual page is the unit of memory protection because all of its bytes share the U/S and R/W flags. However, the same physical memory could be mapped by different pages, possibly with different protection flags. Notice that execute permissions are nowhere to be seen in the PTE. This is why classic x86 paging allows code on the stack to be executed, making it easier to exploit stack buffer overflows (it's still possible to exploit non-executable stacks using `return-to-libc` and other techniques). This lack of a PTE no-execute flag illustrates a broader fact: permission flags in a VMA may or may not translate cleanly into hardware protection. The kernel does what it can, but ultimately the architecture limits what is possible.

Virtual memory doesn't store anything, it simply maps a program's address space onto the underlying physical memory, which is accessed by the processor as a large block called the physical address space. While memory operations on the bus are somewhat involved, we can ignore that here and assume that physical addresses range from zero to the top of available memory in 1-byte increments. This physical address space is broken down by the kernel into page frames. The processor doesn't know or care about frames, yet they are crucial to the kernel because the page frame is the unit of physical memory management. Both Linux and Windows use 4KB page frames in 32-bit mode; here is an example of a machine with 2GB of RAM:



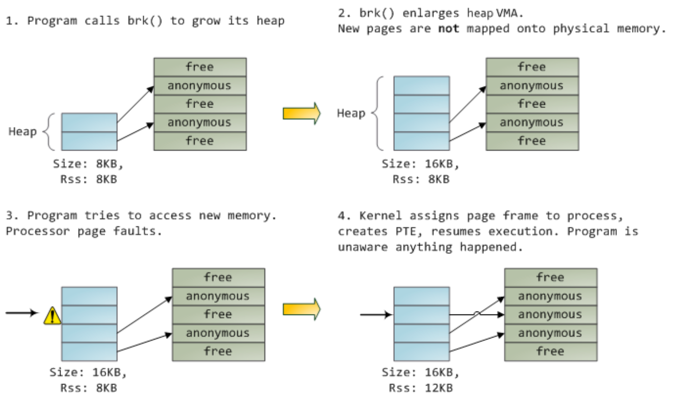
In Linux each page frame is tracked by a descriptor and several flags. Together these descriptors track the entire physical memory in the computer; the precise state of each page frame is always known. Physical memory is managed with the buddy memory allocation technique, hence a page frame is free if it's available for allocation via the buddy system. An allocated page frame might be anonymous, holding program data, or it might be in the page cache, holding data stored in a file or block device. There are other exotic page frame uses, but leave them alone for now. Windows has an analogous Page Frame Number (PFN) database to track physical memory.

Let's put together virtual memory areas, page table entries and page frames to understand how this all works. Below is an example of a user heap:



Blue rectangles represent pages in the VMA range, while arrows represent page table entries mapping pages onto page frames. Some virtual pages lack arrows; this means their corresponding PTEs have the Present flag clear. This could be because the pages have never been touched or because their contents have been swapped out. In either case access to these pages will lead to page faults, even though they are within the VMA. It may seem strange for the VMA and the page tables to disagree, yet this often happens.

A VMA is like a contract between your program and the kernel. You ask for something to be done (memory allocated, a file mapped, etc.), the kernel says "sure," and it creates or updates the appropriate VMA. But it does not actually honor the request right away, it waits until a page fault happens to do real work. The kernel is a lazy, deceitful sack of scum; this is the fundamental principle of virtual memory. It applies in most situations, some familiar and some surprising, but the rule is that VMAs record what has been agreed upon, while PTEs reflect what has actually been done by the lazy kernel. These two data structures together manage a program's memory; both play a role in resolving page faults, freeing memory, swapping memory out, and so on. Let's take the simple case of memory allocation:


1. Program calls brk() to grow its heap
Heap
free
anonymous
free
anonymous
free
Size: 8KB,
Rss: 8KB
2. brk() enlarges heap VMA.
New pages are not mapped onto physical memory.
Heap
free
anonymous
free
anonymous
free
Size: 16KB,
Rss: 8KB
3. Program tries to access new memory.
Processor page faults.
free
anonymous
free
anonymous
free
Size: 16KB,
Rss: 8KB
4. Kernel assigns page frame to process,
creates PTE, resumes execution. Program is
unaware anything happened.
free
anonymous
anonymous
anonymous
free
Size: 16KB,
Rss: 12KB

When the program asks for more memory via the brk() system call, the kernel simply updates the heap VMA and calls it good. No page frames are actually allocated at this point, and the new pages are not present in physical memory. Once the program tries to access the pages, the processor page faults and do_page_fault() is called. It searches for the VMA covering the faulted virtual address using find_vma(). If found, the permissions on the VMA are also checked against the attempted access (read or write). If there's no suitable VMA, no contract covers the attempted memory access and the process is punished by Segmentation Fault.

When a VMA is found the kernel must handle the fault by looking at the PTE contents and the type of VMA. In our case, the PTE shows the page is not present. In fact, our PTE is completely blank (all zeros), which in Linux means the virtual page has never been mapped. Since this is an anonymous VMA, we have a purely RAM affair that must be handled by do_anonymous_page(), which allocates a page frame and makes a PTE to map the faulted virtual page onto the freshly allocated frame.

Things could have been different. The PTE for a swapped out page, for example, has 0 in the Present flag but is not blank. Instead, it stores the swap location holding the page contents, which must be read from disk and loaded into a page frame by do_swap_page() in what is called a major fault. ■

Gustavo Duarte founded his first start up as a freshman in high school, building a web-based stock market analysis tool in Brazil. He sold that company at 18 and emigrated to the US, and now divides his time between the two countries developing software, authoring technical material, and riding snow and waves. He can be reached at *gustavo@duartes.org*.

## Prototype Like A Pro Using Tools You Already Know

Design User Interfaces and Clickable Mockups For Web, Mobile & Desktop Applications In 30 Minutes Or Less Using Your Favorite Presentation Tool

**Keynotopia** user interface libraries contain thousands of native vector components for prototyping with Apple Keynote, Microsoft PowerPoint and OpenOffice Impress.

To create your mockup screens, simply copy UI elements from the libraries onto your slides, add hyperlinks to make the interface clickable, then export the prototype as an interactive PDF file and test it on your mobile device, or send it to your clients, managers and team members to get their feedback.

Keynotopia UI libraries include components for prototyping iPhone, iPad, Android, Windows Phone, BlackBerry, Facebook, OS X, Windows 7 and web 2.0 apps.

It works great with all recent versions of Keynote, PowerPoint and OpenOfice.

KEYNOTOPIA

**Get 20% Off This Week Only With Coupon HM411 at Keynotopia.com**

# Why You Should Never Ask Permission to Clean Up Code

*By* COLIN DEVROE

"CAN I TAKE some time to clean up this code? It is horrendous." The answer to this question should always be "yes." However, often we find ourselves up against walls in the form of budgets, time, due dates, and expectation, and so the typical "powers that be" at companies often veto the request. My advice to you, dear developer, is to never ask for permission for things you know are vital to your work.

You know your work environment better than I do, so perhaps you can ask this question and immediately have the full support of your team. Sad to say that many aren't so fortunate. They'll ask their boss if they can take some time to clean up their code, make it efficient and extensible and, while the boss may recognize the need for such tasks, ultimately the boss will simply say "maybe we can do that later."

Why is this the typical reaction? Because bosses don't have to read, edit, and support the code.

This is folly and every developer knows it. Bosses, (if you're reading this) putting off a few hours worth of code clean-up now will only turn into many hours or days in the future. So by allowing your developers time to do this much-needed code maintenance, you're actually saving your company money. But don't worry — they're not going to ask you for permission anymore. They're just going to do it. ■

Colin Devroe is the Director, Product Management for *Viddler.com*. He enjoys art, writing, traveling, and all forms of whiskey. You can follow him on Twitter as @cdevroe.

Reprinted with permission of the original author.
First appeared in *hn.my/maintenance*.

## Commentary

*By* CATHERINE DARROW (Dove)

I AGREE TO AN extent. It can be easy to fool yourself about what constitutes good code — in the sense of making a product better or making work easier. Sometimes bad code is better left as is. Even working totally unconstrained, I prefer not to refactor something unless I have a pressing reason in mind.

My rule of thumb is this: as a programmer and an employee, I am professionally bound to produce quality software efficiently. If I know I can complete an assignment faster (or in equal time, but leaving behind a better code base) by rewriting something, building a tool, fixing something architectural . . . I will silently do it. No point in asking permission. It's in my charter.

On the other hand, if I want to take a lot of time to re-architect something — an order of magnitude more than it would take to just do whatever it was that brought me there — at that point, it's a strategic decision and management deserves to know about it.

The way I see it, management has no right to require me to produce an unprofessional product in my day to day work. And I have no right to force management to use engineering considerations only in strategic decisions.

*By* THEO JALBA (theoj)

THEN AGAIN, REFACTORING without permission could be a really bad idea. You need to ask yourself a few questions before you proceed.

Do you have thorough unit tests for the code that you are trying to refactor? If not, be aware that there is no way to know for sure that your refactoring won't break the functionality of the code.

Suppose it breaks the code. Have you thought about the operational impact on clients and the financial costs?

Let's say the costs are low. How big and political is your organization? What kind of trouble will you find yourself in? As the hysteria rises, will you be fed to the dogs over this?

How bureaucratic is your company, and how many people do you need to interact with to fix a functionality breakage? The more people you will need to interact with, the more damage you will do to yourself and your reputation. Others will resent working in panic mode to clean up after you (now widely known as the "rogue" programmer).

## Dream. Design. Print.

MagCloud, the revolutionary new self-publishing web service by HP, is changing the way ideas, stories, and images find their way into peoples' hands in a printed magazine format.

HP MagCloud capitalizes on the digital revolution, creating a web-based marketplace where traditional media companies, upstart magazine publishers, students, photographers, designers, and businesses can affordably turn their targeted content into print and digital magazine formats.

Simply upload a PDF of your content, set your selling price, and HP MagCloud takes care of the rest—processing payments, printing magazines on demand, and shipping orders to locations around the world. All magazine formatted publications are printed to order using HP Indigo technology, so they not only look fantastic but there's no waste or overruns, reducing the impact on the environment.

Become part of the future of magazine publishing today at www.magcloud.com.

## 25% Off the First Issue You Publish

Enter promo code **HACKER** when you set your magazine price during the publishing process.

Please contact promo@magcloud.com with any questions.

MAGCLOUD