

# Table of Contents

- [Stuff](#)
- [Holding a Program in One's Head](#)
- [How Not to Die](#)
- [News from the Front](#)
- [How to Do Philosophy](#)
- [The Future of Web Startups](#)
- [Why to Move to a Startup Hub](#)
- [Six Principles for Making New Things](#)
- [Trolls](#)
- [A New Venture Animal](#)
- [You Weren't Meant to Have a Boss](#)
- [How to Disagree](#)
- [Some Heroes](#)
- [Why There Aren't More Googles](#)
- [Be Good](#)
- [Lies We Tell Kids](#)
- [Disconnecting Distraction](#)
- [Cities and Ambition](#)
- [The Pooled-Risk Company Management Company](#)
- [A Fundraising Survival Guide](#)
- [Why to Start a Startup in a Bad Economy](#)
- [The Other Half of "Artists Ship"](#)
- [The High-Res Society](#)
- [Could VC be a Casualty of the Recession?](#)
- [After Credentials](#)
- [Keep Your Identity Small](#)
- [Startups in 13 Sentences](#)
- [What I've Learned from Hacker News](#)
- [Can You Buy a Silicon Valley? Maybe.](#)
- [Why TV Lost](#)
- [How to Be an Angel Investor](#)
- [Relentlessly Resourceful](#)
- [Five Founders](#)
- [The Founder Visa](#)
- [Why Twitter is a Big Deal](#)
- [A Local Revolution?](#)
- [Maker's Schedule, Manager's Schedule](#)
- [Ramen Profitable](#)
- [The Trouble with the Segway](#)
- [What Kate Saw in Silicon Valley](#)
- [The Anatomy of Determination](#)
- [The List of N Things](#)
- [Post-Medium Publishing](#)
- [Persuade xor Discover](#)
- [What Startups Are Really Like](#)
- [Apple's Mistake](#)
- [Organic Startup Ideas](#)

- [How to Lose Time and Money](#)
- [The Top Idea in Your Mind](#)
- [The Acceleration of Addictiveness](#)
- [The Future of Startup Funding](#)
- [What Happened to Yahoo](#)
- [High Resolution Fundraising](#)
- [Where to See Silicon Valley](#)
- [The New Funding Landscape](#)
- [What We Look for in Founders](#)
- [Tablets](#)
- [Founder Control](#)
- [Subject: Airbnb](#)
- [The Patent Pledge](#)
- [Why Startup Hubs Work](#)
- [Snapshot: Viaweb, June 1998](#)
- [Schlep Blindness](#)
- [A Word to the Resourceful](#)
- [Frighteningly Ambitious Startup Ideas](#)
- [Defining Property](#)
- [How Y Combinator Started](#)
- [Writing and Speaking](#)
- [The Top of My Todo List](#)
- [Black Swan Farming](#)
- [Startup = Growth](#)
- [The Hardware Renaissance](#)
- [How to Get Startup Ideas](#)
- [Startup Investing Trends](#)
- [Do Things that Don't Scale](#)

# Stuff

July 2007

I have too much stuff. Most people in America do. In fact, the poorer people are, the more stuff they seem to have. Hardly anyone is so poor that they can't afford a front yard full of old cars.

It wasn't always this way. Stuff used to be rare and valuable. You can still see evidence of that if you look for it. For example, in my house in Cambridge, which was built in 1876, the bedrooms don't have closets. In those days people's stuff fit in a chest of drawers. Even as recently as a few decades ago there was a lot less stuff. When I look back at photos from the 1970s, I'm surprised how empty houses look. As a kid I had what I thought was a huge fleet of toy cars, but they'd be dwarfed by the number of toys my nephews have. All together my Matchboxes and Corgis took up about a third of the surface of my bed. In my nephews' rooms the bed is the only clear space.

Stuff has gotten a lot cheaper, but our attitudes toward it haven't changed correspondingly. We overvalue stuff.

That was a big problem for me when I had no money. I felt poor, and stuff seemed valuable, so almost instinctively I accumulated it. Friends would leave something behind when they moved, or I'd see something as I was walking down the street on trash night (beware of anything you find yourself describing as "perfectly good"), or I'd find something in almost new condition for a tenth its retail price at a garage sale. And pow, more stuff.

In fact these free or nearly free things weren't bargains, because they were worth even less than they cost. Most of the stuff I accumulated was worthless, because I didn't need it.

What I didn't understand was that the value of some new acquisition wasn't the difference between its retail price and what I paid for it. It was the value I derived from it. Stuff is an extremely illiquid asset. Unless you have some plan for selling that valuable thing you got so cheaply, what difference does it make what it's "worth?" The only way you're ever going to extract any value from it is to use it. And if you don't have any immediate use for it, you probably never will.

Companies that sell stuff have spent huge sums training us to think stuff is still

valuable. But it would be closer to the truth to treat stuff as worthless.

In fact, worse than worthless, because once you've accumulated a certain amount of stuff, it starts to own you rather than the other way around. I know of one couple who couldn't retire to the town they preferred because they couldn't afford a place there big enough for all their stuff. Their house isn't theirs; it's their stuff's.

And unless you're extremely organized, a house full of stuff can be very depressing. A cluttered room saps one's spirits. One reason, obviously, is that there's less room for people in a room full of stuff. But there's more going on than that. I think humans constantly scan their environment to build a mental model of what's around them. And the harder a scene is to parse, the less energy you have left for conscious thoughts. A cluttered room is literally exhausting.

(This could explain why clutter doesn't seem to bother kids as much as adults. Kids are less perceptive. They build a coarser model of their surroundings, and this consumes less energy.)

I first realized the worthlessness of stuff when I lived in Italy for a year. All I took with me was one large backpack of stuff. The rest of my stuff I left in my landlady's attic back in the US. And you know what? All I missed were some of the books. By the end of the year I couldn't even remember what else I had stored in that attic.

And yet when I got back I didn't discard so much as a box of it. Throw away a perfectly good rotary telephone? I might need that one day.

The really painful thing to recall is not just that I accumulated all this useless stuff, but that I often spent money I desperately needed on stuff that I didn't.

Why would I do that? Because the people whose job is to sell you stuff are really, really good at it. The average 25 year old is no match for companies that have spent years figuring out how to get you to spend money on stuff. They make the experience of buying stuff so pleasant that "shopping" becomes a leisure activity.

How do you protect yourself from these people? It can't be easy. I'm a fairly skeptical person, and their tricks worked on me well into my thirties. But one thing that might work is to ask yourself, before buying something, "is this going to make my life noticeably better?"

A friend of mine cured herself of a clothes buying habit by asking herself before she bought anything "Am I going to wear this all the time?" If she couldn't convince herself that something she was thinking of buying would become one of those few things she wore all the time, she wouldn't buy it. I think that would work for any kind of purchase. Before you buy anything, ask yourself: will this be something I use constantly? Or is it just something nice? Or worse still, a mere bargain?

The worst stuff in this respect may be stuff you don't use much because it's too good. Nothing owns you like fragile stuff. For example, the "good china" so many households have, and whose defining quality is not so much that it's fun to use, but that one must be especially careful not to break it.

Another way to resist acquiring stuff is to think of the overall cost of owning it. The purchase price is just the beginning. You're going to have to *think* about that thing for years—perhaps for the rest of your life. Every thing you own takes energy away from you. Some give more than they take. Those are the only things worth having.

I've now stopped accumulating stuff. Except books—but books are different. Books are more like a fluid than individual objects. It's not especially inconvenient to own several thousand books, whereas if you owned several thousand random possessions you'd be a local celebrity. But except for books, I now actively avoid stuff. If I want to spend money on some kind of treat, I'll take services over goods any day.

I'm not claiming this is because I've achieved some kind of zenlike detachment from material things. I'm talking about something more mundane. A historical change has taken place, and I've now realized it. Stuff used to be valuable, and now it's not.

In industrialized countries the same thing happened with food in the middle of the twentieth century. As food got cheaper (or we got richer; they're indistinguishable), eating too much started to be a bigger danger than eating too little. We've now reached that point with stuff. For most people, rich or poor, stuff has become a burden.

The good news is, if you're carrying a burden without knowing it, your life could be better than you realize. Imagine walking around for years with five pound ankle weights, then suddenly having them removed.

[Spanish Translation](#)

[Russian Translation](#)

[Italian Translation](#)

[Polish Translation](#)

[Turkish Translation](#)

[French Translation](#)

[Slovak Translation](#)

[Romanian Translation](#)

[German Translation](#)

# Holding a Program in One's Head

August 2007

A good programmer working intensively on his own code can hold it in his mind the way a mathematician holds a problem he's working on. Mathematicians don't answer questions by working them out on paper the way schoolchildren are taught to. They do more in their heads: they try to understand a problem space well enough that they can walk around it the way you can walk around the memory of the house you grew up in. At its best programming is the same. You hold the whole program in your head, and you can manipulate it at will.

That's particularly valuable at the start of a project, because initially the most important thing is to be able to change what you're doing. Not just to solve the problem in a different way, but to change the problem you're solving.

Your code is your understanding of the problem you're exploring. So it's only when you have your code in your head that you really understand the problem.

It's not easy to get a program into your head. If you leave a project for a few months, it can take days to really understand it again when you return to it. Even when you're actively working on a program it can take half an hour to load into your head when you start work each day. And that's in the best case. Ordinary programmers working in typical office conditions never enter this mode. Or to put it more dramatically, ordinary programmers working in typical office conditions never really understand the problems they're solving.

Even the best programmers don't always have the whole program they're working on loaded into their heads. But there are things you can do to help:

1. **Avoid distractions.** Distractions are bad for many types of work, but especially bad for programming, because programmers tend to operate at the limit of the detail they can handle.

The danger of a distraction depends not on how long it is, but on how much it scrambles your brain. A programmer can leave the office and go and get a sandwich without losing the code in his head. But the wrong kind of interruption can wipe your brain in 30 seconds.

Oddly enough, scheduled distractions may be worse than unscheduled ones. If you know you have a meeting in an hour, you don't even start working on



something hard.

2. **Work in long stretches.** Since there's a fixed cost each time you start working on a program, it's more efficient to work in a few long sessions than many short ones. There will of course come a point where you get stupid because you're tired. This varies from person to person. I've heard of people hacking for 36 hours straight, but the most I've ever been able to manage is about 18, and I work best in chunks of no more than 12.

The optimum is not the limit you can physically endure. There's an advantage as well as a cost of breaking up a project. Sometimes when you return to a problem after a rest, you find your unconscious mind has left an answer waiting for you.

3. **Use succinct languages.** More [powerful](#) programming languages make programs shorter. And programmers seem to think of programs at least partially in the language they're using to write them. The more succinct the language, the shorter the program, and the easier it is to load and keep in your head.

You can magnify the effect of a powerful language by using a style called bottom-up programming, where you write programs in multiple layers, the lower ones acting as programming languages for those above. If you do this right, you only have to keep the topmost layer in your head.

4. **Keep rewriting your program.** Rewriting a program often yields a cleaner design. But it would have advantages even if it didn't: you have to understand a program completely to rewrite it, so there is no better way to get one loaded into your head.
5. **Write rereadable code.** All programmers know it's good to write readable code. But you yourself are the most important reader. Especially in the beginning; a prototype is a conversation with yourself. And when writing for yourself you have different priorities. If you're writing for other people, you may not want to make code too dense. Some parts of a program may be easiest to read if you spread things out, like an introductory textbook. Whereas if you're writing code to make it easy to reload into your head, it may be best to go for brevity.
6. **Work in small groups.** When you manipulate a program in your head, your vision tends to stop at the edge of the code you own. Other parts you don't understand as well, and more importantly, can't take liberties with. So the smaller the number of programmers, the more completely a project can mutate. If there's just one programmer, as there often is at first, you can do all-encompassing redesigns.
7. **Don't have multiple people editing the same piece of code.** You never understand other people's code as well as your own. No matter how thoroughly you've read it, you've only read it, not written it. So if a piece of code is written by multiple authors, none of them understand it as well as a single author would.

And of course you can't safely redesign something other people are working on. It's not just that you'd have to ask permission. You don't even let

yourself think of such things. Redesigning code with several authors is like changing laws; redesigning code you alone control is like seeing the other interpretation of an ambiguous image.

If you want to put several people to work on a project, divide it into components and give each to one person.

8. **Start small.** A program gets easier to hold in your head as you become familiar with it. You can start to treat parts as black boxes once you feel confident you've fully explored them. But when you first start working on a project, you're forced to see everything. If you start with too big a problem, you may never quite be able to encompass it. So if you need to write a big, complex program, the best way to begin may not be to write a spec for it, but to write a prototype that solves a subset of the problem. Whatever the advantages of planning, they're often outweighed by the advantages of being able to keep a program in your head.

It's striking how often programmers manage to hit all eight points by accident. Someone has an idea for a new project, but because it's not officially sanctioned, he has to do it in off hours—which turn out to be more productive because there are no distractions. Driven by his enthusiasm for the new project he works on it for many hours at a stretch. Because it's initially just an experiment, instead of a "production" language he uses a mere "scripting" language—which is in fact far more powerful. He completely rewrites the program several times; that wouldn't be justifiable for an official project, but this is a labor of love and he wants it to be perfect. And since no one is going to see it except him, he omits any comments except the note-to-self variety. He works in a small group perforce, because he either hasn't told anyone else about the idea yet, or it seems so unpromising that no one else is allowed to work on it. Even if there is a group, they couldn't have multiple people editing the same code, because it changes too fast for that to be possible. And the project starts small because the idea *is* small at first; he just has some cool hack he wants to try out.

Even more striking are the number of officially sanctioned projects that manage to do *all eight things wrong*. In fact, if you look at the way software gets written in most organizations, it's almost as if they were deliberately trying to do things wrong. In a sense, they are. One of the defining qualities of organizations since there have been such a thing is to treat individuals as interchangeable parts. This works well for more parallelizable tasks, like fighting wars. For most of history a well-drilled army of professional soldiers could be counted on to beat an army of individual warriors, no matter how valorous. But having ideas is not very parallelizable. And that's what programs are: ideas.

It's not merely true that organizations dislike the idea of depending on individual genius, it's a tautology. It's part of the definition of an organization not to. Of our current concept of an organization, at least.

Maybe we could define a new kind of organization that combined the efforts of individuals without requiring them to be interchangeable. Arguably a market is such a form of organization, though it may be more accurate to describe a market

as a degenerate case—as what you get by default when organization isn't possible.

Probably the best we'll do is some kind of hack, like making the programming parts of an organization work differently from the rest. Perhaps the optimal solution is for big companies not even to try to develop ideas in house, but simply to [buy](#) them. But regardless of what the solution turns out to be, the first step is to realize there's a problem. There is a contradiction in the very phrase "software company." The two words are pulling in opposite directions. Any good programmer in a large organization is going to be at odds with it, because organizations are designed to prevent what programmers strive for.

Good programmers manage to get a lot done anyway. But often it requires practically an act of rebellion against the organizations that employ them. Perhaps it will help if more people understand that the way programmers behave is driven by the demands of the work they do. It's not because they're irresponsible that they work in long binges during which they blow off all other obligations, plunge straight into programming instead of writing specs first, and rewrite code that already works. It's not because they're unfriendly that they prefer to work alone, or growl at people who pop their head in the door to say hello. This apparently random collection of annoying habits has a single explanation: the power of holding a program in one's head.

Whether or not understanding this can help large organizations, it can certainly help their competitors. The weakest point in big companies is that they don't let individual programmers do great work. So if you're a little startup, this is the place to attack them. Take on the kind of problems that have to be solved in one big brain.

**Thanks** to Sam Altman, David Greenspan, Aaron Iba, Jessica Livingston, Robert Morris, Peter Norvig, Lisa Randall, Emmett Shear, Sergei Tsarev, and Stephen Wolfram for reading drafts of this.

[Japanese Translation](#)

[Simplified Chinese Translation](#)

[Portuguese Translation](#)

[Bulgarian Translation](#)

[Russian Translation](#)

# How Not to Die

August 2007

*(This is a talk I gave at the last Y Combinator dinner of the summer. Usually we don't have a speaker at the last dinner; it's more of a party. But it seemed worth spoiling the atmosphere if I could save some of the startups from preventable deaths. So at the last minute I cooked up this rather grim talk. I didn't mean this as an essay; I wrote it down because I only had two hours before dinner and think fastest while writing.)*

A couple days ago I told a reporter that we expected about a third of the companies we funded to succeed. Actually I was being conservative. I'm hoping it might be as much as a half. Wouldn't it be amazing if we could achieve a 50% success rate?

Another way of saying that is that half of you are going to die. Phrased that way, it doesn't sound good at all. In fact, it's kind of weird when you think about it, because our definition of success is that the founders get rich. If half the startups we fund succeed, then half of you are going to get rich and the other half are going to get nothing.

If you can just avoid dying, you get rich. That sounds like a joke, but it's actually a pretty good description of what happens in a typical startup. It certainly describes what happened in Viaweb. We avoided dying till we got rich.

It was really close, too. When we were visiting Yahoo to talk about being acquired, we had to interrupt everything and borrow one of their conference rooms to talk down an investor who was about to back out of a new funding round we needed to stay alive. So even in the middle of getting rich we were fighting off the grim reaper.

You may have heard that quote about luck consisting of opportunity meeting preparation. You've now done the preparation. The work you've done so far has, in effect, put you in a position to get lucky: you can now get rich by not letting your company die. That's more than most people have. So let's talk about how not to die.

We've done this five times now, and we've seen a bunch of startups die. About 10

of them so far. We don't know exactly what happens when they die, because they generally don't die loudly and heroically. Mostly they crawl off somewhere and die.

For us the main indication of impending doom is when we don't hear from you. When we haven't heard from, or about, a startup for a couple months, that's a bad sign. If we send them an email asking what's up, and they don't reply, that's a really bad sign. So far that is a 100% accurate predictor of death.

Whereas if a startup regularly does new deals and releases and either sends us mail or shows up at YC events, they're probably going to live.

I realize this will sound naive, but maybe the linkage works in both directions. Maybe if you can arrange that we keep hearing from you, you won't die.

That may not be so naive as it sounds. You've probably noticed that having dinners every Tuesday with us and the other founders causes you to get more done than you would otherwise, because every dinner is a mini Demo Day. Every dinner is a kind of a deadline. So the mere constraint of staying in regular contact with us will push you to make things happen, because otherwise you'll be embarrassed to tell us that you haven't done anything new since the last time we talked.

If this works, it would be an amazing hack. It would be pretty cool if merely by staying in regular contact with us you could get rich. It sounds crazy, but there's a good chance that would work.

A variant is to stay in touch with other YC-funded startups. There is now a whole neighborhood of them in San Francisco. If you move there, the peer pressure that made you work harder all summer will continue to operate.

When startups die, the official cause of death is always either running out of money or a critical founder bailing. Often the two occur simultaneously. But I think the underlying cause is usually that they've become demoralized. You rarely hear of a startup that's working around the clock doing deals and pumping out new features, and dies because they can't pay their bills and their ISP unplugs their server.

Startups rarely die in mid keystroke. So keep typing!

If so many startups get demoralized and fail when merely by hanging on they could get rich, you have to assume that running a startup can be demoralizing. That is certainly true. I've been there, and that's why I've never done another startup. The low points in a startup are just unbelievably low. I bet even Google had moments where things seemed hopeless.

Knowing that should help. If you know it's going to feel terrible sometimes, then when it feels terrible you won't think "ouch, this feels terrible, I give up." It feels that way for everyone. And if you just hang on, things will probably get better. The metaphor people use to describe the way a startup feels is at least a roller coaster

and not drowning. You don't just sink and sink; there are ups after the downs.

Another feeling that seems alarming but is in fact normal in a startup is the feeling that what you're doing isn't working. The reason you can expect to feel this is that what you do probably won't work. Startups almost never get it right the first time. Much more commonly you launch something, and no one cares. Don't assume when this happens that you've failed. That's normal for startups. But don't sit around doing nothing. Iterate.

I like Paul Buchheit's suggestion of trying to make something that at least someone really loves. As long as you've made something that a few users are ecstatic about, you're on the right track. It will be good for your morale to have even a handful of users who really love you, and startups run on morale. But also it will tell you what to focus on. What is it about you that they love? Can you do more of that? Where can you find more people who love that sort of thing? As long as you have some core of users who love you, all you have to do is expand it. It may take a while, but as long as you keep plugging away, you'll win in the end. Both Blogger and Delicious did that. Both took years to succeed. But both began with a core of fanatically devoted users, and all Evan and Joshua had to do was grow that core incrementally. [Wufoo](#) is on the same trajectory now.

So when you release something and it seems like no one cares, look more closely. Are there zero users who really love you, or is there at least some little group that does? It's quite possible there will be zero. In that case, tweak your product and try again. Every one of you is working on a space that contains at least one winning permutation somewhere in it. If you just keep trying, you'll find it.

Let me mention some things not to do. The number one thing not to do is other things. If you find yourself saying a sentence that ends with "but we're going to keep working on the startup," you are in big trouble. Bob's going to grad school, but we're going to keep working on the startup. We're moving back to Minnesota, but we're going to keep working on the startup. We're taking on some consulting projects, but we're going to keep working on the startup. You may as well just translate these to "we're giving up on the startup, but we're not willing to admit that to ourselves," because that's what it means most of the time. A startup is so hard that working on it can't be preceded by "but."

In particular, don't go to graduate school, and don't start other projects. Distraction is fatal to startups. Going to (or back to) school is a huge predictor of death because in addition to the distraction it gives you something to say you're doing. If you're only doing a startup, then if the startup fails, you fail. If you're in grad school and your startup fails, you can say later "Oh yeah, we had this startup on the side when I was in grad school, but it didn't go anywhere."

You can't use euphemisms like "didn't go anywhere" for something that's your only occupation. People won't let you.

One of the most interesting things we've discovered from working on Y Combinator

is that founders are more motivated by the fear of looking bad than by the hope of getting millions of dollars. So if you want to get millions of dollars, put yourself in a position where failure will be public and humiliating.

When we first met the founders of [Octopart](#), they seemed very smart, but not a great bet to succeed, because they didn't seem especially committed. One of the two founders was still in grad school. It was the usual story: he'd drop out if it looked like the startup was taking off. Since then he has not only dropped out of grad school, but appeared full length in [Newsweek](#) with the word "Billionaire" printed across his chest. He just cannot fail now. Everyone he knows has seen that picture. Girls who dissed him in high school have seen it. His mom probably has it on the fridge. It would be unthinkable humiliating to fail now. At this point he is committed to fight to the death.

I wish every startup we funded could appear in a Newsweek article describing them as the next generation of billionaires, because then none of them would be able to give up. The success rate would be 90%. I'm not kidding.

When we first knew the Octoparts they were lighthearted, cheery guys. Now when we talk to them they seem grimly determined. The electronic parts distributors are trying to squash them to keep their monopoly pricing. (If it strikes you as odd that people still order electronic parts out of thick paper catalogs in 2007, there's a reason for that. The distributors want to prevent the transparency that comes from having prices online.) I feel kind of bad that we've transformed these guys from lighthearted to grimly determined. But that comes with the territory. If a startup succeeds, you get millions of dollars, and you don't get that kind of money just by asking for it. You have to assume it takes some amount of pain.

And however tough things get for the Octoparts, I predict they'll succeed. They may have to morph themselves into something totally different, but they won't just crawl off and die. They're smart; they're working in a promising field; and they just cannot give up.

All of you guys already have the first two. You're all smart and working on promising ideas. Whether you end up among the living or the dead comes down to the third ingredient, not giving up.

So I'll tell you now: bad shit is coming. It always is in a startup. The odds of getting from launch to liquidity without some kind of disaster happening are one in a thousand. So don't get demoralized. When the disaster strikes, just say to yourself, ok, this was what Paul was talking about. What did he say to do? Oh, yeah. Don't give up.



[Japanese Translation](#)

[Arabic Translation](#)

# News from the Front

September 2007

A few weeks ago I had a thought so heretical that it really surprised me. It may not matter all that much where you go to college.

For me, as for a lot of middle class kids, getting into a good college was more or less the meaning of life when I was growing up. What was I? A student. To do that well meant to get good grades. Why did one have to get good grades? To get into a good college. And why did one want to do that? There seemed to be several reasons: you'd learn more, get better jobs, make more money. But it didn't matter exactly what the benefits would be. College was a bottleneck through which all your future prospects passed; everything would be better if you went to a better college.

A few weeks ago I realized that somewhere along the line I had stopped believing that.

What first set me thinking about this was the new trend of worrying obsessively about what [kindergarten](#) your kids go to. It seemed to me this couldn't possibly matter. Either it won't help your kid get into Harvard, or if it does, getting into Harvard won't mean much anymore. And then I thought: how much does it mean even now?

It turns out I have a lot of data about that. My three partners and I run a seed stage investment firm called [Y Combinator](#). We invest when the company is just a couple guys and an idea. The idea doesn't matter much; it will change anyway. Most of our decision is based on the founders. The average founder is three years out of college. Many have just graduated; a few are still in school. So we're in much the same position as a graduate program, or a company hiring people right out of college. Except our choices are immediately and visibly tested. There are two possible outcomes for a startup: success or failure—and usually you know within a year which it will be.

The test applied to a startup is among the purest of real world tests. A startup succeeds or fails depending almost entirely on the efforts of the founders. Success is decided by the market: you only succeed if users like what you've built. And users don't care where you went to college.

As well as having precisely measurable results, we have a lot of them. Instead of doing a small number of large deals like a traditional venture capital fund, we do a large number of small ones. We currently fund about 40 companies a year, selected from about 900 applications representing a total of about 2000 people.

[1]

Between the volume of people we judge and the rapid, unequivocal test that's applied to our choices, Y Combinator has been an unprecedented opportunity for learning how to pick winners. One of the most surprising things we've learned is how little it matters where people went to college.

I thought I'd already been cured of caring about that. There's nothing like going to grad school at Harvard to cure you of any illusions you might have about the average Harvard undergrad. And yet Y Combinator showed us we were still overestimating people who'd been to elite colleges. We'd interview people from MIT or Harvard or Stanford and sometimes find ourselves thinking: they *must* be smarter than they seem. It took us a few iterations to learn to trust our senses.

Practically everyone thinks that someone who went to MIT or Harvard or Stanford must be smart. Even people who hate you for it believe it.

But when you think about what it means to have gone to an elite college, how could this be true? We're talking about a decision made by admissions officers—basically, HR people—based on a cursory examination of a huge pile of depressingly similar applications submitted by seventeen year olds. And what do they have to go on? An easily gamed standardized test; a short essay telling you what the kid thinks you want to hear; an interview with a random alum; a high school record that's largely an index of obedience. Who would rely on such a test?

And yet a lot of companies do. A lot of companies are very much influenced by where applicants went to college. How could they be? I think I know the answer to that.

There used to be a saying in the corporate world: "No one ever got fired for buying IBM." You no longer hear this about IBM specifically, but the idea is very much alive; there is a whole category of "enterprise" software companies that exist to take advantage of it. People buying technology for large organizations don't care if they pay a fortune for mediocre software. It's not their money. They just want to buy from a supplier who seems safe—a company with an established name, confident salesmen, impressive offices, and software that conforms to all the current fashions. Not necessarily a company that will deliver so much as one that, if they do let you down, will still seem to have been a prudent choice. So companies have evolved to fill that niche.

A recruiter at a big company is in much the same position as someone buying technology for one. If someone went to Stanford and is not obviously insane, they're probably a safe bet. And a safe bet is enough. No one ever measures recruiters by the later performance of people they turn down. [2]

I'm not saying, of course, that elite colleges have evolved to prey upon the weaknesses of large organizations the way enterprise software companies have. But they work as if they had. In addition to the power of the brand name, graduates of elite colleges have two critical qualities that plug right into the way large organizations work. They're good at doing what they're asked, since that's what it takes to please the adults who judge you at seventeen. And having been to an elite college makes them more confident.

Back in the days when people might spend their whole career at one big company, these qualities must have been very valuable. Graduates of elite colleges would have been capable, yet amenable to authority. And since individual performance is so hard to measure in large organizations, their own confidence would have been the starting point for their reputation.

Things are very different in the new world of startups. We couldn't save someone from the market's judgement even if we wanted to. And being charming and confident counts for nothing with users. All users care about is whether you make something they like. If you don't, you're dead.

Knowing that test is coming makes us work a lot harder to get the right answers than anyone would if they were merely hiring people. We can't afford to have any illusions about the predictors of success. And what we've found is that the variation between schools is so much smaller than the variation between individuals that it's negligible by comparison. We can learn more about someone in the first minute of talking to them than by knowing where they went to school.

It seems obvious when you put it that way. Look at the individual, not where they went to college. But that's a weaker statement than the idea I began with, that it doesn't matter much where a given individual goes to college. Don't you learn things at the best schools that you wouldn't learn at lesser places?

Apparently not. Obviously you can't prove this in the case of a single individual, but you can tell from aggregate evidence: you can't, without asking them, distinguish people who went to one school from those who went to another three times as far down the *US News* list. [3] Try it and see.

How can this be? Because how much you learn in college depends a lot more on you than the college. A determined party animal can get through the best school without learning anything. And someone with a real thirst for knowledge will be able to find a few smart people to learn from at a school that isn't prestigious at all.

The other students are the biggest advantage of going to an elite college; you learn more from them than the professors. But you should be able to reproduce this at most colleges if you make a conscious effort to find smart friends. At most colleges you can find at least a handful of other smart students, and most people have only a handful of close friends in college anyway. [4] The odds of finding

smart professors are even better. The curve for faculty is a lot flatter than for students, especially in math and the hard sciences; you have to go pretty far down the list of colleges before you stop finding smart professors in the math department.

So it's not surprising that we've found the relative prestige of different colleges useless in judging individuals. There's a lot of randomness in how colleges select people, and what they learn there depends much more on them than the college. Between these two sources of variation, the college someone went to doesn't mean a lot. It is to some degree a predictor of ability, but so weak that we regard it mainly as a source of error and try consciously to ignore it.

I doubt what we've discovered is an anomaly specific to startups. Probably people have always overestimated the importance of where one goes to college. We're just finally able to measure it.

The unfortunate thing is not just that people are judged by such a superficial test, but that so many judge themselves by it. A lot of people, probably the majority of people in America, have some amount of insecurity about where, or whether, they went to college. The tragedy of the situation is that by far the greatest liability of not having gone to the college you'd have liked is your own feeling that you're thereby lacking something. Colleges are a bit like exclusive clubs in this respect. There is only one real advantage to being a member of most exclusive clubs: you know you wouldn't be missing much if you weren't. When you're excluded, you can only imagine the advantages of being an insider. But invariably they're larger in your imagination than in real life.

So it is with colleges. Colleges differ, but they're nothing like the stamp of destiny so many imagine them to be. People aren't what some admissions officer decides about them at seventeen. They're what they make themselves.

Indeed, the great advantage of not caring where people went to college is not just that you can stop judging them (and yourself) by superficial measures, but that you can focus instead on what really matters. What matters is what you make of yourself. I think that's what we should tell kids. Their job isn't to get good grades so they can get into a good college, but to learn and do. And not just because that's more rewarding than worldly success. That will increasingly *be* the route to worldly success.

## Notes

[1] Is what we measure worth measuring? I think so. You can get rich simply by being energetic and unscrupulous, but getting rich from a technology startup takes some amount of brains. It is just the kind of work the upper middle class values; it

has about the same intellectual component as being a doctor.

[2] Actually, someone did, once. Mitch Kapor's wife Freada was in charge of HR at Lotus in the early years. (As he is at pains to point out, they did not become romantically involved till afterward.) At one point they worried Lotus was losing its startup edge and turning into a big company. So as an experiment she sent their recruiters the resumes of the first 40 employees, with identifying details changed. These were the people who had made Lotus into the star it was. Not one got an interview.

[3] The *US News* list? Surely no one trusts that. Even if the statistics they consider are useful, how do they decide on the relative weights? The reason the *US News* list is meaningful is precisely because they are so intellectually dishonest in that respect. There is no external source they can use to calibrate the weighting of the statistics they use; if there were, we could just use that instead. What they must do is adjust the weights till the top schools are the usual suspects in about the right order. So in effect what the *US News* list tells us is what the editors think the top schools are, which is probably not far from the conventional wisdom on the matter. The amusing thing is, because some schools work hard to game the system, the editors will have to keep tweaking their algorithm to get the rankings they want.

[4] Possible doesn't mean easy, of course. A smart student at a party school will inevitably be something of an outcast, just as he or she would be in most [high schools](#).

**Thanks** to Trevor Blackwell, Sarah Harlin, Jessica Livingston, Jackie McDonough, Peter Norvig, and Robert Morris for reading drafts of this.

[French Translation](#)

# How to Do Philosophy



September 2007

In high school I decided I was going to study philosophy in college. I had several motives, some more honorable than others. One of the less honorable was to shock people. College was regarded as job training where I grew up, so studying philosophy seemed an impressively impractical thing to do. Sort of like slashing holes in your clothes or putting a safety pin through your ear, which were other forms of impressive impracticality then just coming into fashion.

But I had some more honest motives as well. I thought studying philosophy would be a shortcut straight to wisdom. All the people majoring in other things would just end up with a bunch of domain knowledge. I would be learning what was really what.

I'd tried to read a few philosophy books. Not recent ones; you wouldn't find those in our high school library. But I tried to read Plato and Aristotle. I doubt I believed I understood them, but they sounded like they were talking about something important. I assumed I'd learn what in college.

The summer before senior year I took some college classes. I learned a lot in the calculus class, but I didn't learn much in Philosophy 101. And yet my plan to study philosophy remained intact. It was my fault I hadn't learned anything. I hadn't read the books we were assigned carefully enough. I'd give Berkeley's *Principles of Human Knowledge* another shot in college. Anything so admired and so difficult to read must have something in it, if one could only figure out what.

Twenty-six years later, I still don't understand Berkeley. I have a nice edition of his collected works. Will I ever read it? Seems unlikely.

The difference between then and now is that now I understand why Berkeley is probably not worth trying to understand. I think I see now what went wrong with philosophy, and how we might fix it.

## Words

I did end up being a philosophy major for most of college. It didn't work out as I'd hoped. I didn't learn any magical truths compared to which everything else was mere domain knowledge. But I do at least know now why I didn't. Philosophy doesn't really have a subject matter in the way math or history or most other university subjects do. There is no core of knowledge one must master. The closest you come to that is a knowledge of what various individual philosophers have said about different topics over the years. Few were sufficiently correct that people have forgotten who discovered what they discovered.

Formal logic has some subject matter. I took several classes in logic. I don't know if I learned anything from them. [\[1\]](#) It does seem to me very important to be able to flip ideas around in one's head: to see when two ideas don't fully cover the space of possibilities, or when one idea is the same as another but with a couple things changed. But did studying logic teach me the importance of thinking this way, or make me any better at it? I don't know.

There are things I know I learned from studying philosophy. The most dramatic I learned immediately, in the first semester of freshman year, in a class taught by Sydney Shoemaker. I learned that I don't exist. I am (and you are) a collection of cells that lurches around driven by various forces, and calls itself *I*. But there's no central, indivisible thing that your identity goes with. You could conceivably lose half your brain and live. Which means your brain could conceivably be split into two halves and each transplanted into different bodies. Imagine waking up after such an operation. You have to imagine being two people.

The real lesson here is that the concepts we use in everyday life are fuzzy, and break down if pushed too hard. Even a concept as dear to us as *I*. It took me a while to grasp this, but when I did it was fairly sudden, like someone in the nineteenth century grasping evolution and realizing the story of creation they'd been told as a child was all wrong. [\[2\]](#) Outside of math there's a limit to how far you can push words; in fact, it would not be a bad definition of math to call it the study of terms that have precise meanings. Everyday words are inherently imprecise. They work well enough in everyday life that you don't notice. Words seem to work, just as Newtonian physics seems to. But you can always make them break if you push them far enough.

I would say that this has been, unfortunately for philosophy, the central fact of philosophy. Most philosophical debates are not merely afflicted by but driven by confusions over words. Do we have free will? Depends what you mean by "free." Do abstract ideas exist? Depends what you mean by "exist."

Wittgenstein is popularly credited with the idea that most philosophical controversies are due to confusions over language. I'm not sure how much credit to give him. I suspect a lot of people realized this, but reacted simply by not studying philosophy, rather than becoming philosophy professors.



How did things get this way? Can something people have spent thousands of years studying really be a waste of time? Those are interesting questions. In fact, some of the most interesting questions you can ask about philosophy. The most valuable way to approach the current philosophical tradition may be neither to get lost in pointless speculations like Berkeley, nor to shut them down like Wittgenstein, but to study it as an example of reason gone wrong.

## History

Western philosophy really begins with Socrates, Plato, and Aristotle. What we know of their predecessors comes from fragments and references in later works; their doctrines could be described as speculative cosmology that occasionally strays into analysis. Presumably they were driven by whatever makes people in every other society invent cosmologies. [3]

With Socrates, Plato, and particularly Aristotle, this tradition turned a corner. There started to be a lot more analysis. I suspect Plato and Aristotle were encouraged in this by progress in math. Mathematicians had by then shown that you could figure things out in a much more conclusive way than by making up fine sounding stories about them. [4]

People talk so much about abstractions now that we don't realize what a leap it must have been when they first started to. It was presumably many thousands of years between when people first started describing things as hot or cold and when someone asked "what is heat?" No doubt it was a very gradual process. We don't know if Plato or Aristotle were the first to ask any of the questions they did. But their works are the oldest we have that do this on a large scale, and there is a freshness (not to say naivete) about them that suggests some of the questions they asked were new to them, at least.

Aristotle in particular reminds me of the phenomenon that happens when people discover something new, and are so excited by it that they race through a huge percentage of the newly discovered territory in one lifetime. If so, that's evidence of how new this kind of thinking was. [5]

This is all to explain how Plato and Aristotle can be very impressive and yet naive and mistaken. It was impressive even to ask the questions they did. That doesn't mean they always came up with good answers. It's not considered insulting to say that ancient Greek mathematicians were naive in some respects, or at least lacked some concepts that would have made their lives easier. So I hope people will not be too offended if I propose that ancient philosophers were similarly naive. In particular, they don't seem to have fully grasped what I earlier called the central fact of philosophy: that words break if you push them too far.

"Much to the surprise of the builders of the first digital computers," Rod Brooks wrote, "programs written for them usually did not work." [6] Something similar happened when people first started trying to talk about abstractions. Much to their surprise, they didn't arrive at answers they agreed upon. In fact, they rarely

seemed to arrive at answers at all.

They were in effect arguing about artifacts induced by sampling at too low a resolution.

The proof of how useless some of their answers turned out to be is how little effect they have. No one after reading Aristotle's *Metaphysics* does anything differently as a result. [7]

Surely I'm not claiming that ideas have to have practical applications to be interesting? No, they may not have to. Hardy's boast that number theory had no use whatsoever wouldn't disqualify it. But he turned out to be mistaken. In fact, it's suspiciously hard to find a field of math that truly has no practical use. And Aristotle's explanation of the ultimate goal of philosophy in Book A of the *Metaphysics* implies that philosophy should be useful too.

## Theoretical Knowledge

Aristotle's goal was to find the most general of general principles. The examples he gives are convincing: an ordinary worker builds things a certain way out of habit; a master craftsman can do more because he grasps the underlying principles. The trend is clear: the more general the knowledge, the more admirable it is. But then he makes a mistake—possibly the most important mistake in the history of philosophy. He has noticed that theoretical knowledge is often acquired for its own sake, out of curiosity, rather than for any practical need. So he proposes there are two kinds of theoretical knowledge: some that's useful in practical matters and some that isn't. Since people interested in the latter are interested in it for its own sake, it must be more noble. So he sets as his goal in the *Metaphysics* the exploration of knowledge that has no practical use. Which means no alarms go off when he takes on grand but vaguely understood questions and ends up getting lost in a sea of words.

His mistake was to confuse motive and result. Certainly, people who want a deep understanding of something are often driven by curiosity rather than any practical need. But that doesn't mean what they end up learning is useless. It's very valuable in practice to have a deep understanding of what you're doing; even if you're never called on to solve advanced problems, you can see shortcuts in the solution of simple ones, and your knowledge won't break down in edge cases, as it would if you were relying on formulas you didn't understand. Knowledge is power. That's what makes theoretical knowledge prestigious. It's also what causes smart people to be curious about certain things and not others; our DNA is not so disinterested as we might think.

So while ideas don't have to have immediate practical applications to be interesting, the kinds of things we find interesting will surprisingly often turn out to have practical applications.

The reason Aristotle didn't get anywhere in the *Metaphysics* was partly that he set

off with contradictory aims: to explore the most abstract ideas, guided by the assumption that they were useless. He was like an explorer looking for a territory to the north of him, starting with the assumption that it was located to the south.

And since his work became the map used by generations of future explorers, he sent them off in the wrong direction as well. [8] Perhaps worst of all, he protected them from both the criticism of outsiders and the promptings of their own inner compass by establishing the principle that the most noble sort of theoretical knowledge had to be useless.

The *Metaphysics* is mostly a failed experiment. A few ideas from it turned out to be worth keeping; the bulk of it has had no effect at all. The *Metaphysics* is among the least read of all famous books. It's not hard to understand the way Newton's *Principia* is, but the way a garbled message is.

Arguably it's an interesting failed experiment. But unfortunately that was not the conclusion Aristotle's successors derived from works like the *Metaphysics*. [9] Soon after, the western world fell on intellectual hard times. Instead of version 1s to be superseded, the works of Plato and Aristotle became revered texts to be mastered and discussed. And so things remained for a shockingly long time. It was not till around 1600 (in Europe, where the center of gravity had shifted by then) that one found people confident enough to treat Aristotle's work as a catalog of mistakes. And even then they rarely said so outright.

If it seems surprising that the gap was so long, consider how little progress there was in math between Hellenistic times and the Renaissance.

In the intervening years an unfortunate idea took hold: that it was not only acceptable to produce works like the *Metaphysics*, but that it was a particularly prestigious line of work, done by a class of people called philosophers. No one thought to go back and debug Aristotle's motivating argument. And so instead of correcting the problem Aristotle discovered by falling into it—that you can easily get lost if you talk too loosely about very abstract ideas—they continued to fall into it.

## The Singularity

Curiously, however, the works they produced continued to attract new readers. Traditional philosophy occupies a kind of singularity in this respect. If you write in an unclear way about big ideas, you produce something that seems tantalizingly attractive to inexperienced but intellectually ambitious students. Till one knows better, it's hard to distinguish something that's hard to understand because the writer was unclear in his own mind from something like a mathematical proof that's hard to understand because the ideas it represents are hard to understand. To someone who hasn't learned the difference, traditional philosophy seems extremely attractive: as hard (and therefore impressive) as math, yet broader in scope. That was what lured me in as a high school student.

This singularity is even more singular in having its own defense built in. When things are hard to understand, people who suspect they're nonsense generally keep quiet. There's no way to prove a text is meaningless. The closest you can get is to show that the official judges of some class of texts can't distinguish them from placebos. [10]

And so instead of denouncing philosophy, most people who suspected it was a waste of time just studied other things. That alone is fairly damning evidence, considering philosophy's claims. It's supposed to be about the ultimate truths. Surely all smart people would be interested in it, if it delivered on that promise.

Because philosophy's flaws turned away the sort of people who might have corrected them, they tended to be self-perpetuating. Bertrand Russell wrote in a letter in 1912:

Hitherto the people attracted to philosophy have been mostly those who loved the big generalizations, which were all wrong, so that few people with exact minds have taken up the subject. [11]

His response was to launch Wittgenstein at it, with dramatic results.

I think Wittgenstein deserves to be famous not for the discovery that most previous philosophy was a waste of time, which judging from the circumstantial evidence must have been made by every smart person who studied a little philosophy and declined to pursue it further, but for how he acted in response. [12] Instead of quietly switching to another field, he made a fuss, from inside. He was Gorbachev.

The field of philosophy is still shaken from the fright Wittgenstein gave it. [13] Later in life he spent a lot of time talking about how words worked. Since that seems to be allowed, that's what a lot of philosophers do now. Meanwhile, sensing a vacuum in the metaphysical speculation department, the people who used to do literary criticism have been edging Kantward, under new names like "literary theory," "critical theory," and when they're feeling ambitious, plain "theory." The writing is the familiar word salad:

Gender is not like some of the other grammatical modes which express precisely a mode of conception without any reality that corresponds to the conceptual mode, and consequently do not express precisely something in reality by which the intellect could be moved to conceive a thing the way it does, even where that motive is not something in the thing as such. [14]

The singularity I've described is not going away. There's a market for writing that sounds impressive and can't be disproven. There will always be both supply and demand. So if one group abandons this territory, there will always be others ready to occupy it.

## **A Proposal**

We may be able to do better. Here's an intriguing possibility. Perhaps we should do what Aristotle meant to do, instead of what he did. The goal he announces in the *Metaphysics* seems one worth pursuing: to discover the most general truths. That sounds good. But instead of trying to discover them because they're useless, let's try to discover them because they're useful.

I propose we try again, but that we use that heretofore despised criterion, applicability, as a guide to keep us from wondering off into a swamp of abstractions. Instead of trying to answer the question:

What are the most general truths?

let's try to answer the question

Of all the useful things we can say, which are the most general?

The test of utility I propose is whether we cause people who read what we've written to do anything differently afterward. Knowing we have to give definite (if implicit) advice will keep us from straying beyond the resolution of the words we're using.

The goal is the same as Aristotle's; we just approach it from a different direction.

As an example of a useful, general idea, consider that of the controlled experiment. There's an idea that has turned out to be widely applicable. Some might say it's part of science, but it's not part of any specific science; it's literally meta-physics (in our sense of "meta"). The idea of evolution is another. It turns out to have quite broad applications—for example, in genetic algorithms and even product design. Frankfurt's distinction between lying and bullshitting seems a promising recent example. [\[15\]](#)

These seem to me what philosophy should look like: quite general observations that would cause someone who understood them to do something differently.

Such observations will necessarily be about things that are imprecisely defined. Once you start using words with precise meanings, you're doing math. So starting from utility won't entirely solve the problem I described above—it won't flush out the metaphysical singularity. But it should help. It gives people with good intentions a new roadmap into abstraction. And they may thereby produce things that make the writing of the people with bad intentions look bad by comparison.

One drawback of this approach is that it won't produce the sort of writing that gets you tenure. And not just because it's not currently the fashion. In order to get tenure in any field you must not arrive at conclusions that members of tenure committees can disagree with. In practice there are two kinds of solutions to this problem. In math and the sciences, you can prove what you're saying, or at any rate adjust your conclusions so you're not claiming anything false ("6 of 8 subjects had lower blood pressure after the treatment"). In the humanities you can either avoid drawing any definite conclusions (e.g. conclude that an issue is a complex

one), or draw conclusions so narrow that no one cares enough to disagree with you.

The kind of philosophy I'm advocating won't be able to take either of these routes. At best you'll be able to achieve the essayist's standard of proof, not the mathematician's or the experimentalist's. And yet you won't be able to meet the usefulness test without implying definite and fairly broadly applicable conclusions. Worse still, the usefulness test will tend to produce results that annoy people: there's no use in telling people things they already believe, and people are often upset to be told things they don't.

Here's the exciting thing, though. Anyone can do this. Getting to general plus useful by starting with useful and cranking up the generality may be unsuitable for junior professors trying to get tenure, but it's better for everyone else, including professors who already have it. This side of the mountain is a nice gradual slope. You can start by writing things that are useful but very specific, and then gradually make them more general. Joe's has good burritos. What makes a good burrito? What makes good food? What makes anything good? You can take as long as you want. You don't have to get all the way to the top of the mountain. You don't have to tell anyone you're doing philosophy.

If it seems like a daunting task to do philosophy, here's an encouraging thought. The field is a lot younger than it seems. Though the first philosophers in the western tradition lived about 2500 years ago, it would be misleading to say the field is 2500 years old, because for most of that time the leading practitioners weren't doing much more than writing commentaries on Plato or Aristotle while watching over their shoulders for the next invading army. In the times when they weren't, philosophy was hopelessly intermingled with religion. It didn't shake itself free till a couple hundred years ago, and even then was afflicted by the structural problems I've described above. If I say this, some will say it's a ridiculously overbroad and uncharitable generalization, and others will say it's old news, but here goes: judging from their works, most philosophers up to the present have been wasting their time. So in a sense the field is still at the first step. [\[16\]](#)

That sounds a preposterous claim to make. It won't seem so preposterous in 10,000 years. Civilization always seems old, because it's always the oldest it's ever been. The only way to say whether something is really old or not is by looking at structural evidence, and structurally philosophy is young; it's still reeling from the unexpected breakdown of words.

Philosophy is as young now as math was in 1500. There is a lot more to discover.

## Notes

[1] In practice formal logic is not much use, because despite some progress in the last 150 years we're still only able to formalize a small percentage of statements. We may never do that much better, for the same reason 1980s-style "knowledge representation" could never have worked; many statements may have no representation more concise than a huge, analog brain state.

[2] It was harder for Darwin's contemporaries to grasp this than we can easily imagine. The story of creation in the Bible is not just a Judeo-Christian concept; it's roughly what everyone must have believed since before people were people. The hard part of grasping evolution was to realize that species weren't, as they seem to be, unchanging, but had instead evolved from different, simpler organisms over unimaginably long periods of time.

Now we don't have to make that leap. No one in an industrialized country encounters the idea of evolution for the first time as an adult. Everyone's taught about it as a child, either as truth or heresy.

[3] Greek philosophers before Plato wrote in verse. This must have affected what they said. If you try to write about the nature of the world in verse, it inevitably turns into incantation. Prose lets you be more precise, and more tentative.

[4] Philosophy is like math's ne'er-do-well brother. It was born when Plato and Aristotle looked at the works of their predecessors and said in effect "why can't you be more like your brother?" Russell was still saying the same thing 2300 years later.

Math is the precise half of the most abstract ideas, and philosophy the imprecise half. It's probably inevitable that philosophy will suffer by comparison, because there's no lower bound to its precision. Bad math is merely boring, whereas bad philosophy is nonsense. And yet there are *some* good ideas in the imprecise half.

[5] Aristotle's best work was in logic and zoology, both of which he can be said to have invented. But the most dramatic departure from his predecessors was a new, much more analytical style of thinking. He was arguably the first scientist.

[6] Brooks, Rodney, *Programming in Common Lisp*, Wiley, 1985, p. 94.

[7] Some would say we depend on Aristotle more than we realize, because his ideas were one of the ingredients in our common culture. Certainly a lot of the words we use have a connection with Aristotle, but it seems a bit much to suggest that we wouldn't have the concept of the essence of something or the distinction between matter and form if Aristotle hadn't written about them.

One way to see how much we really depend on Aristotle would be to diff European culture with Chinese: what ideas did European culture have in 1800 that Chinese culture didn't, in virtue of Aristotle's contribution?

[8] The meaning of the word "philosophy" has changed over time. In ancient times it covered a broad range of topics, comparable in scope to our "scholarship" (though without the methodological implications). Even as late as Newton's time it included what we now call "science." But core of the subject today is still what seemed to Aristotle the core: the attempt to discover the most general truths.

Aristotle didn't call this "metaphysics." That name got assigned to it because the books we now call the *Metaphysics* came after (meta = after) the *Physics* in the standard edition of Aristotle's works compiled by Andronicus of Rhodes three centuries later. What we call "metaphysics" Aristotle called "first philosophy."

[9] Some of Aristotle's immediate successors may have realized this, but it's hard to say because most of their works are lost.

[10] Sokal, Alan, "Transgressing the Boundaries: Toward a Transformative Hermeneutics of Quantum Gravity," *Social Text* 46/47, pp. 217-252.

Abstract-sounding nonsense seems to be most attractive when it's aligned with some axe the audience already has to grind. If this is so we should find it's most popular with groups that are (or feel) weak. The powerful don't need its reassurance.

[11] Letter to Ottoline Morrell, December 1912. Quoted in:

Monk, Ray, *Ludwig Wittgenstein: The Duty of Genius*, Penguin, 1991, p. 75.

[12] A preliminary result, that all metaphysics between Aristotle and 1783 had been a waste of time, is due to I. Kant.

[13] Wittgenstein asserted a sort of mastery to which the inhabitants of early 20th century Cambridge seem to have been peculiarly vulnerable—perhaps partly because so many had been raised religious and then stopped believing, so had a vacant space in their heads for someone to tell them what to do (others chose Marx or Cardinal Newman), and partly because a quiet, earnest place like Cambridge in that era had no natural immunity to messianic figures, just as European politics then had no natural immunity to dictators.

[14] This is actually from the *Ordinatio* of Duns Scotus (ca. 1300), with "number" replaced by "gender." Plus ca change.

Wolter, Allan (trans), *Duns Scotus: Philosophical Writings*, Nelson, 1963, p. 92.

[15] Frankfurt, Harry, *On Bullshit*, Princeton University Press, 2005.

[16] Some introductions to philosophy now take the line that philosophy is worth studying as a process rather than for any particular truths you'll learn. The philosophers whose works they cover would be rolling in their graves at that. They



hoped they were doing more than serving as examples of how to argue: they hoped they were getting results. Most were wrong, but it doesn't seem an impossible hope.

This argument seems to me like someone in 1500 looking at the lack of results achieved by alchemy and saying its value was as a process. No, they were going about it wrong. It turns out it is possible to transmute lead into gold (though not economically at current energy prices), but the route to that knowledge was to backtrack and try another approach.

**Thanks** to Trevor Blackwell, Paul Buchheit, Jessica Livingston, Robert Morris, Mark Nitzberg, and Peter Norvig for reading drafts of this.

[French Translation](#)

# The Future of Web Startups

October 2007

*(This essay is derived from a keynote at FOWA in October 2007.)*

There's something interesting happening right now. Startups are undergoing the same transformation that technology does when it becomes cheaper.

It's a pattern we see over and over in technology. Initially there's some device that's very expensive and made in small quantities. Then someone discovers how to make them cheaply; many more get built; and as a result they can be used in new ways.

Computers are a familiar example. When I was a kid, computers were big, expensive machines built one at a time. Now they're a commodity. Now we can stick computers in everything.

This pattern is very old. Most of the turning points in economic history are instances of it. It happened to steel in the 1850s, and to power in the 1780s. It happened to cloth manufacture in the thirteenth century, generating the wealth that later brought about the Renaissance. Agriculture itself was an instance of this pattern.

Now as well as being produced by startups, this pattern is happening *to* startups. It's so cheap to start web startups that orders of magnitudes more will be started. If the pattern holds true, that should cause dramatic changes.

## **1. Lots of Startups**

So my first prediction about the future of web startups is pretty straightforward: there will be a lot of them. When starting a startup was expensive, you had to get the permission of investors to do it. Now the only threshold is courage.

Even that threshold is getting lower, as people watch others take the plunge and survive. In the last batch of startups we funded, we had several founders who said they'd thought of applying before, but weren't sure and got jobs instead. It was only after hearing reports of friends who'd done it that they decided to try it themselves.

Starting a startup is hard, but having a 9 to 5 job is hard too, and in some ways a worse kind of hard. In a startup you have lots of worries, but you don't have that feeling that your life is flying by like you do in a big company. Plus in a startup you could make much more money.

As word spreads that startups work, the number may grow to a point that would now seem surprising.

We now think of it as normal to have a job at a company, but this is the thinnest of historical veneers. Just two or three lifetimes ago, most people in what are now called industrialized countries lived by farming. So while it may seem surprising to propose that large numbers of people will change the way they make a living, it would be more surprising if they didn't.

## **2. Standardization**

When technology makes something dramatically cheaper, standardization always follows. When you make things in large volumes you tend to standardize everything that doesn't need to change.

At Y Combinator we still only have four people, so we try to standardize everything. We could hire employees, but we want to be forced to figure out how to scale investing.

We often tell startups to release a minimal version one quickly, then let the needs of the users determine what to do next. In essence, let the market design the product. We've done the same thing ourselves. We think of the techniques we're developing for dealing with large numbers of startups as like software. Sometimes it literally is software, like [Hacker News](#) and our application system.

One of the most important things we've been working on standardizing are investment terms. Till now investment terms have been individually negotiated. This is a problem for founders, because it makes raising money take longer and cost more in legal fees. So as well as using the same paperwork for every deal we do, we've commissioned generic angel paperwork that all the startups we fund can use for future rounds.

Some investors will still want to cook up their own deal terms. Series A rounds, where you raise a million dollars or more, will be custom deals for the foreseeable future. But I think angel rounds will start to be done mostly with standardized agreements. An angel who wants to insert a bunch of complicated terms into the agreement is probably not one you want anyway.

## **3. New Attitude to Acquisition**

Another thing I see starting to get standardized is acquisitions. As the volume of startups increases, big companies will start to develop standardized procedures

that make acquisitions little more work than hiring someone.

Google is the leader here, as in so many areas of technology. They buy a lot of startups— more than most people realize, because they only announce a fraction of them. And being Google, they're figuring out how to do it efficiently.

One problem they've solved is how to think about acquisitions. For most companies, acquisitions still carry some stigma of inadequacy. Companies do them because they have to, but there's usually some feeling they shouldn't have to—that their own programmers should be able to build everything they need.

Google's example should cure the rest of the world of this idea. Google has by far the best programmers of any public technology company. If they don't have a problem doing acquisitions, the others should have even less problem. However many Google does, Microsoft should do ten times as many.

One reason Google doesn't have a problem with acquisitions is that they know first-hand the quality of the people they can get that way. Larry and Sergey only started Google after making the rounds of the search engines trying to sell their idea and finding no takers. They've *been* the guys coming in to visit the big company, so they know who might be sitting across that conference table from them.

#### **4. Riskier Strategies are Possible**

Risk is always proportionate to reward. The way to get really big returns is to do things that seem crazy, like starting a new search engine in 1998, or turning down a billion dollar acquisition offer.

This has traditionally been a problem in venture funding. Founders and investors have different attitudes to risk. Knowing that risk is on average proportionate to reward, investors like risky strategies, while founders, who don't have a big enough sample size to care what's true on average, tend to be more conservative.

If startups are easy to start, this conflict goes away, because founders can start them younger, when it's rational to take more risk, and can start more startups total in their careers. When founders can do lots of startups, they can start to look at the world in the same portfolio-optimizing way as investors. And that means the overall amount of wealth created can be greater, because strategies can be riskier.

#### **5. Younger, Nerdier Founders**

If startups become a cheap commodity, more people will be able to have them, just as more people could have computers once microprocessors made them cheap. And in particular, younger and more technical founders will be able to start startups than could before.

Back when it cost a lot to start a startup, you had to convince investors to let you

do it. And that required very different skills from actually doing the startup. If investors were perfect judges, the two would require exactly the same skills. But unfortunately most investors are terrible judges. I know because I see behind the scenes what an enormous amount of work it takes to raise money, and the amount of selling required in an industry is always inversely proportional to the judgement of the buyers.

Fortunately, if startups get cheaper to start, there's another way to convince investors. Instead of going to venture capitalists with a business plan and trying to convince them to fund it, you can get a product launched on a few tens of thousands of dollars of seed money from us or your uncle, and approach them with a working company instead of a plan for one. Then instead of having to seem smooth and confident, you can just point them to Alexa.

This way of convincing investors is better suited to hackers, who often went into technology in part because they felt uncomfortable with the amount of fakeness required in other fields.

## **6. Startup Hubs Will Persist**

It might seem that if startups get cheap to start, it will mean the end of startup hubs like Silicon Valley. If all you need to start a startup is rent money, you should be able to do it anywhere.

This is kind of true and kind of false. It's true that you can now *start* a startup anywhere. But you have to do more with a startup than just start it. You have to make it succeed. And that is more likely to happen in a startup hub.

I've thought a lot about this question, and it seems to me the increasing cheapness of web startups will if anything increase the importance of startup hubs. The value of startup hubs, like centers for any kind of business, lies in something very old-fashioned: face to face meetings. No technology in the immediate future will replace walking down University Ave and running into a friend who tells you how to fix a bug that's been bothering you all weekend, or visiting a friend's startup down the street and ending up in a conversation with one of their investors.

The question of whether to be in a startup hub is like the question of whether to take outside investment. The question is not whether you *need* it, but whether it brings any advantage at all. Because anything that brings an advantage will give your competitors an advantage over you if they do it and you don't. So if you hear someone saying "we don't need to be in Silicon Valley," that use of the word "need" is a sign they're not even thinking about the question right.

And while startup hubs are as powerful magnets as ever, the increasing cheapness of starting a startup means the particles they're attracting are getting lighter. A startup now can be just a pair of 22 year old guys. A company like that can move much more easily than one with 10 people, half of whom have kids.

We know because we make people move for Y Combinator, and it doesn't seem to be a problem. The advantage of being able to work together face to face for three months outweighs the inconvenience of moving. Ask anyone who's done it.

The mobility of seed-stage startups means that seed funding is a national business. One of the most common emails we get is from people asking if we can help them set up a local clone of Y Combinator. But this just wouldn't work. Seed funding isn't regional, just as big research universities aren't.

Is seed funding not merely national, but international? Interesting question. There are signs it may be. We've had an ongoing stream of founders from outside the US, and they tend to do particularly well, because they're all people who were so determined to succeed that they were willing to move to another country to do it.

The more mobile startups get, the harder it would be to start new silicon valleys. If startups are mobile, the best local talent will go to the real Silicon Valley, and all they'll get at the local one will be the people who didn't have the energy to move.

This is not a nationalistic idea, incidentally. It's cities that compete, not countries. Atlanta is just as hosed as Munich.

## **7. Better Judgement Needed**

If the number of startups increases dramatically, then the people whose job is to judge them are going to have to get better at it. I'm thinking particularly of investors and acquirers. We now get on the order of 1000 applications a year. What are we going to do if we get 10,000?

That's actually an alarming idea. But we'll figure out some kind of answer. We'll have to. It will probably involve writing some software, but fortunately we can do that.

Acquirers will also have to get better at picking winners. They generally do better than investors, because they pick later, when there's more performance to measure. But even at the most advanced acquirers, identifying companies to buy is extremely ad hoc, and completing the acquisition often involves a great deal of unnecessary friction.

I think acquirers may eventually have chief acquisition officers who will both identify good acquisitions and make the deals happen. At the moment those two functions are separate. Promising new startups are often discovered by developers. If someone powerful enough wants to buy them, the deal is handed over to corp dev guys to negotiate. It would be better if both were combined in one group, headed by someone with a technical background and some vision of what they wanted to accomplish. Maybe in the future big companies will have both a VP of Engineering responsible for technology developed in-house, and a CAO responsible for bringing technology in from outside.

At the moment, there is no one within big companies who gets in trouble when they buy a startup for \$200 million that they could have bought earlier for \$20 million. There should start to be someone who gets in trouble for that.

## **8. College Will Change**

If the best hackers start their own companies after college instead of getting jobs, that will change what happens in college. Most of these changes will be for the better. I think the experience of college is warped in a bad way by the expectation that afterward you'll be judged by potential employers.

One change will be in the meaning of "after college," which will switch from when one graduates from college to when one leaves it. If you're starting your own company, why do you need a degree? We don't encourage people to start startups during college, but the best founders are certainly capable of it. Some of the most successful companies we've funded were started by undergrads.

I grew up in a time where college degrees seemed really important, so I'm alarmed to be saying things like this, but there's nothing magical about a degree. There's nothing that magically changes after you take that last exam. The importance of degrees is due solely to the administrative needs of large organizations. These can certainly affect your life—it's hard to get into grad school, or to get a work visa in the US, without an undergraduate degree—but tests like this will matter less and less.

As well as mattering less whether students get degrees, it will also start to matter less where they go to college. In a startup you're judged by users, and they don't care where you went to college. So in a world of startups, elite universities will play less of a role as gatekeepers. In the US it's a national scandal how easily children of rich parents game college admissions. But the way this problem ultimately gets solved may not be by reforming the universities but by going around them. We in the technology world are used to that sort of solution: you don't beat the incumbents; you redefine the problem to make them irrelevant.

The greatest value of universities is not the brand name or perhaps even the classes so much as the people you meet. If it becomes common to start a startup after college, students may start trying to maximize this. Instead of focusing on getting internships at companies they want to work for, they may start to focus on working with other students they want as cofounders.

What students do in their classes will change too. Instead of trying to get good grades to impress future employers, students will try to learn things. We're talking about some pretty dramatic changes here.

## **9. Lots of Competitors**

If it gets easier to start a startup, it's easier for competitors too. That doesn't erase

the advantage of increased cheapness, however. You're not all playing a zero-sum game. There's not some fixed number of startups that can succeed, regardless of how many are started.

In fact, I don't think there's any limit to the number of startups that could succeed. Startups succeed by creating wealth, which is the satisfaction of people's desires. And people's desires seem to be effectively infinite, at least in the short term.

What the increasing number of startups does mean is that you won't be able to sit on a good idea. Other people have your idea, and they'll be increasingly likely to do something about it.

## **10. Faster Advances**

There's a good side to that, at least for consumers of technology. If people get right to work implementing ideas instead of sitting on them, technology will evolve faster.

Some kinds of innovations happen a company at a time, like the punctuated equilibrium model of evolution. There are some kinds of ideas that are so threatening that it's hard for big companies even to think of them. Look at what a hard time Microsoft is having discovering web apps. They're like a character in a movie that everyone in the audience can see something bad is about to happen to, but who can't see it himself. The big innovations that happen a company at a time will obviously happen faster if the rate of new companies increases.

But in fact there will be a double speed increase. People won't wait as long to act on new ideas, but also those ideas will increasingly be developed within startups rather than big companies. Which means technology will evolve faster per company as well.

Big companies are just not a good place to make things happen fast. I talked recently to a founder whose startup had been acquired by a big company. He was a precise sort of guy, so he'd measured their productivity before and after. He counted lines of code, which can be a dubious measure, but in this case was meaningful because it was the same group of programmers. He found they were one thirteenth as productive after the acquisition.

The company that bought them was not a particularly stupid one. I think what he was measuring was mostly the cost of bigness. I experienced this myself, and his number sounds about right. There's something about big companies that just sucks the energy out of you.

Imagine what all that energy could do if it were put to use. There is an enormous latent capacity in the world's hackers that most people don't even realize is there. That's the main reason we do Y Combinator: to let loose all this energy by making it easy for hackers to start their own startups.



## A Series of Tubes

The process of starting startups is currently like the plumbing in an old house. The pipes are narrow and twisty, and there are leaks in every joint. In the future this mess will gradually be replaced by a single, huge pipe. The water will still have to get from A to B, but it will get there faster and without the risk of spraying out through some random leak.

This will change a lot of things for the better. In a big, straight pipe like that, the force of being measured by one's performance will propagate back through the whole system. Performance is always the ultimate test, but there are so many kinks in the plumbing now that most people are insulated from it most of the time. So you end up with a world in which high school students think they need to get good grades to get into elite colleges, and college students think they need to get good grades to impress employers, within which the employees waste most of their time in political battles, and from which consumers have to buy anyway because there are so few choices. Imagine if that sequence became a big, straight pipe. Then the effects of being measured by performance would propagate all the way back to high school, flushing out all the arbitrary stuff people are measured by now. That is the future of web startups.

**Thanks** to Brian Oberkirch and Simon Willison for inviting me to speak, and the crew at Carson Systems for making everything run smoothly.

[Japanese Translation](#)

# Why to Move to a Startup Hub



October 2007

After the last [talk](#) I gave, one of the organizers got up on the stage to deliver an impromptu rebuttal. That never happened before. I only heard the first few sentences, but that was enough to tell what I said that upset him: that startups would do better if they moved to Silicon Valley.

This conference was in London, and most of the audience seemed to be from the UK. So saying startups should move to Silicon Valley seemed like a nationalistic remark: an obnoxious American telling them that if they wanted to do things right they should all just move to America.

Actually I'm less American than I seem. I didn't say so, but I'm British by birth. And just as Jews are ex officio allowed to tell Jewish jokes, I don't feel like I have to bother being diplomatic with a British audience.

The idea that startups would do better to move to Silicon Valley is not even a nationalistic one. <sup>[1]</sup> It's the same thing I say to startups in the US. Y Combinator alternates between coasts every 6 months. Every other funding cycle is in Boston. And even though Boston is the second biggest startup hub in the US (and the world), we tell the startups from those cycles that their best bet is to move to Silicon Valley. If that's true of Boston, it's even more true of every other city.

This is about cities, not countries.

And I think I can prove I'm right. You can easily reduce the opposing argument ad what most people would agree was absurdum. Few would be willing to claim that it doesn't matter at all where a startup is—that a startup operating out of a small agricultural town wouldn't benefit from moving to a startup hub. Most people could see how it might be helpful to be in a place where there was infrastructure for

startups, accumulated knowledge about how to make them work, and other people trying to do it. And yet whatever argument you use to prove that startups don't need to move from London to Silicon Valley could equally well be used to prove startups don't need to move from smaller towns to London.

The difference between cities is a matter of degree. And if, as nearly everyone who knows agrees, startups are better off in Silicon Valley than Boston, then they're better off in Silicon Valley than everywhere else too.

I realize I might seem to have a vested interest in this conclusion, because startups that move to the US might do it through Y Combinator. But the American startups we've funded will attest that I say the same thing to them.

I'm not claiming of course that every startup has to go to Silicon Valley to succeed. Just that all other things being equal, the more of a startup hub a place is, the better startups will do there. But other considerations can outweigh the advantages of moving. I'm not saying founders with families should uproot them to move halfway around the world; that might be too much of a distraction.

Immigration difficulties might be another reason to stay put. Dealing with immigration problems is like raising money: for some reason it seems to consume all your attention. A startup can't afford much of that. One Canadian startup we funded spent about 6 months working on moving to the US. Eventually they just gave up, because they couldn't afford to take so much time away from working on their software.

(If another country wanted to establish a rival to Silicon Valley, the single best thing they could do might be to create a special visa for startup founders. US immigration policy is one of Silicon Valley's biggest weaknesses.)

If your startup is connected to a specific industry, you may be better off in one of its centers. A startup doing something related to entertainment might want to be in New York or LA.

And finally, if a good investor has committed to fund you if you stay where you are, you should probably stay. Finding investors is hard. You generally shouldn't pass up a definite funding offer to move. [\[2\]](#)

In fact, the quality of the investors may be the main advantage of startup hubs. Silicon Valley investors are noticeably more aggressive than Boston ones. Over and over, I've seen startups we've funded snatched by west coast investors out from under the noses of Boston investors who saw them first but acted too slowly. At this year's Boston Demo Day, I told the audience that this happened every year, so if they saw a startup they liked, they should make them an offer. And yet within a month it had happened again: an aggressive west coast VC who had met the founder of a YC-funded startup a week before beat out a Boston VC who had known him for years. By the time the Boston VC grasped what was happening, the deal was already gone.

Boston investors will admit they're more conservative. Some want to believe this comes from the city's prudent Yankee character. But Occam's razor suggests the truth is less flattering. Boston investors are probably more conservative than Silicon Valley investors for the same reason Chicago investors are more conservative than Boston ones. They don't understand startups as well.

West coast investors aren't bolder because they're irresponsible cowboys, or because the good weather makes them optimistic. They're bolder because they know what they're doing. They're the skiers who ski on the diamond slopes. Boldness is the essence of venture investing. The way you get big returns is not by trying to avoid losses, but by trying to ensure you get some of the big hits. And the big hits often look risky at first.

Like Facebook. Facebook was started in Boston. Boston VCs had the first shot at them. But they said no, so Facebook moved to Silicon Valley and raised money there. The partner who turned them down now says that "may turn out to have been a mistake."

Empirically, boldness wins. If the aggressive ways of west coast investors are going to come back to bite them, it has been a long time coming. Silicon Valley has been pulling ahead of Boston since the 1970s. If there was going to be a comeuppance for the west coast investors, the bursting of the Bubble would have been it. But since then the west coast has just pulled further ahead.

West coast investors are confident enough of their judgement to act boldly; east coast investors, not so much; but anyone who thinks east coast investors act that way out of prudence should see the frantic reactions of an east coast VC in the process of losing a deal to a west coast one.

In addition to the concentration that comes from specialization, startup hubs are also markets. And markets are usually centralized. Even now, when traders could be anywhere, they cluster in a few cities. It's hard to say exactly what it is about face to face contact that makes deals happen, but whatever it is, it hasn't yet been duplicated by technology.

Walk down University Ave at the right time, and you might overhear five different people talking on the phone about deals. In fact, this is part of the reason Y Combinator is in Boston half the time: it's hard to stand that year round. But though it can sometimes be annoying to be surrounded by people who only think about one thing, it's the place to be if that one thing is what you're trying to do.

I was talking recently to someone who works on search at Google. He knew a lot of people at Yahoo, so he was in a good position to compare the two companies. I asked him why Google was better at search. He said it wasn't anything specific Google did, but simply that they understood search so much better.

And that's why startups thrive in startup hubs like Silicon Valley. Startups are a

very specialized business, as specialized as diamond cutting. And in startup hubs they understand it.

## Notes

[1] The nationalistic idea is the converse: that startups should stay in a certain city because of the country it's in. If you really have a "one world" viewpoint, deciding to move from London to Silicon Valley is no different from deciding to move from Chicago to Silicon Valley.

[2] An investor who merely seems like he will fund you, however, you can ignore. Seeming like they will fund you one day is the way investors say No.

**Thanks** to Sam Altman, Jessica Livingston, Harjeet Taggar, and Kulveer Taggar for reading drafts of this.



[Comment](#) on this essay.

[Japanese Translation](#)

# Six Principles for Making New Things



February 2008

The fiery reaction to the release of [Arc](#) had an unexpected consequence: it made me realize I had a design philosophy. The main complaint of the more articulate critics was that Arc seemed so flimsy. After years of working on it, all I had to show for myself were a few thousand lines of macros? Why hadn't I worked on more substantial problems?

As I was mulling over these remarks it struck me how familiar they seemed. This was exactly the kind of thing people said at first about Viaweb, and Y Combinator, and most of my essays.

When we launched Viaweb, it seemed laughable to VCs and e-commerce "experts." We were just a couple guys in an apartment, which did not seem cool in 1995 the way it does now. And the thing we'd built, as far as they could tell, wasn't even software. Software, to them, equalled big, honking Windows apps. Since Viaweb was the first web-based app they'd seen, it seemed to be nothing more than a website. They were even more contemptuous when they discovered that Viaweb didn't process credit card transactions (we didn't for the whole first year). Transaction processing seemed to them what e-commerce was all about. It sounded serious and difficult.

And yet, mysteriously, Viaweb ended up crushing all its competitors.

The initial reaction to [Y Combinator](#) was almost identical. It seemed laughably lightweight. Startup funding meant series A rounds: millions of dollars given to a small number of startups founded by people with established credentials after months of serious, businesslike meetings, on terms described in a document a foot thick. Y Combinator seemed inconsequential. It's too early to say yet whether Y Combinator will turn out like Viaweb, but judging from the number of imitations, a lot of people seem to think we're on to something.

I can't measure whether my essays are successful, except in page views, but the reaction to them is at least different from when I started. At first the default reaction of the Slashdot trolls was (translated into articulate terms): "Who is this guy and what authority does he have to write about these topics? I haven't read the essay, but there's no way anything so short and written in such an informal style could have anything useful to say about such and such topic, when people with degrees in the subject have already written many thick books about it." Now there's a new generation of trolls on a new generation of sites, but they have at least started to omit the initial "Who is this guy?"

Now people are saying the same things about Arc that they said at first about Viaweb and Y Combinator and most of my essays. Why the pattern? The answer, I realized, is that my m.o. for all four has been the same.

Here it is: I like to find (a) simple solutions (b) to overlooked problems (c) that actually need to be solved, and (d) deliver them as informally as possible, (e) starting with a very crude version 1, then (f) iterating rapidly.

When I first laid out these principles explicitly, I noticed something striking: this is practically a recipe for generating a contemptuous initial reaction. Though simple solutions are better, they don't seem as impressive as complex ones. Overlooked problems are by definition problems that most people think don't matter. Delivering solutions in an informal way means that instead of judging something by the way it's presented, people have to actually understand it, which is more work. And starting with a crude version 1 means your initial effort is always small and incomplete.

I'd noticed, of course, that people never seemed to grasp new ideas at first. I thought it was just because most people were stupid. Now I see there's more to it than that. Like a contrarian investment fund, someone following this strategy will almost always be doing things that seem wrong to the average person.

As with contrarian investment strategies, that's exactly the point. This technique is successful (in the long term) because it gives you all the advantages other people forgo by trying to seem legit. If you work on overlooked problems, you're more likely to discover new things, because you have less competition. If you deliver solutions informally, you (a) save all the effort you would have had to expend to make them look impressive, and (b) avoid the danger of fooling yourself as well as your audience. And if you release a crude version 1 then iterate, your solution can benefit from the imagination of nature, which, as Feynman pointed out, is more powerful than your own.

In the case of Viaweb, the simple solution was to make the software run on the server. The overlooked problem was to generate web sites automatically; in 1995, online stores were all made by hand by human designers, but we knew this wouldn't scale. The part that actually mattered was graphic design, not transaction processing. The informal delivery mechanism was me, showing up in jeans and a t-

shirt at some retailer's office. And the crude version 1 was, if I remember correctly, less than 10,000 lines of code when we launched.

The power of this technique extends beyond startups and programming languages and essays. It probably extends to any kind of creative work. Certainly it can be used in painting: this is exactly what Cezanne and Klee did.

At Y Combinator we bet money on it, in the sense that we encourage the startups we fund to work this way. There are always new ideas right under your nose. So look for simple things that other people have overlooked—things people will later claim were "obvious"—especially when they've been led astray by obsolete conventions, or by trying to do things that are superficially impressive. Figure out what the real problem is, and make sure you solve that. Don't worry about trying to look corporate; the product is what wins in the long term. And launch as soon as you can, so you start learning from users what you should have been making.

[Reddit](#) is a classic example of this approach. When Reddit first launched, it seemed like there was nothing to it. To the graphically unsophisticated its deliberately minimal design seemed like no design at all. But Reddit solved the real problem, which was to tell people what was new and otherwise stay out of the way. As a result it became massively successful. Now that conventional ideas have caught up with it, it seems obvious. People look at Reddit and think the founders were lucky. Like all such things, it was harder than it looked. The Reddits pushed so hard against the current that they reversed it; now it looks like they're merely floating downstream.

So when you look at something like Reddit and think "I wish I could think of an idea like that," remember: ideas like that are all around you. But you ignore them because they look wrong.



# Trolls

February 2008

A user on Hacker News recently posted a [comment](#) that set me thinking:

Something about hacker culture that never really set well with me was this 💎 the nastiness. ... I just don't understand why people troll like they do.

I've thought a lot over the last couple years about the problem of trolls. It's an old one, as old as forums, but we're still just learning what the causes are and how to address them.

There are two senses of the word "troll." In the original sense it meant someone, usually an outsider, who deliberately stirred up fights in a forum by saying controversial things. [1] For example, someone who didn't use a certain programming language might go to a forum for users of that language and make disparaging remarks about it, then sit back and watch as people rose to the bait. This sort of trolling was in the nature of a practical joke, like letting a bat loose in a room full of people.

The definition then spread to people who behaved like assholes in forums, whether intentionally or not. Now when people talk about trolls they usually mean this broader sense of the word. Though in a sense this is historically inaccurate, it is in other ways more accurate, because when someone is being an asshole it's usually uncertain even in their own mind how much is deliberate. That is arguably one of the defining qualities of an asshole.

I think trolling in the broader sense has four causes. The most important is distance. People will say things in anonymous forums that they'd never dare say to someone's face, just as they'll do things in cars that they'd never do as pedestrians 💎 like tailgate people, or honk at them, or cut them off.

Trolling tends to be particularly bad in forums related to computers, and I think that's due to the kind of people you find there. Most of them (myself included) are more comfortable dealing with abstract ideas than with people. Hackers can be abrupt even in person. Put them on an anonymous forum, and the problem gets worse.

The third cause of trolling is incompetence. If you disagree with something, it's

easier to say "you suck" than to figure out and explain exactly what you disagree with. You're also safe that way from refutation. In this respect trolling is a lot like graffiti. Graffiti happens at the intersection of ambition and incompetence: people want to make their mark on the world, but have no other way to do it than literally making a mark on the world. [2]

The final contributing factor is the culture of the forum. Trolls are like children (many *are* children) in that they're capable of a wide range of behavior depending on what they think will be tolerated. In a place where rudeness isn't tolerated, most can be polite. But vice versa as well.

There's a sort of Gresham's Law of trolls: trolls are willing to use a forum with a lot of thoughtful people in it, but thoughtful people aren't willing to use a forum with a lot of trolls in it. Which means that once trolling takes hold, it tends to become the dominant culture. That had already happened to Slashdot and Digg by the time I paid attention to comment threads there, but I watched it happen to Reddit.

News.YC is, among other things, an experiment to see if this fate can be avoided. The site's [guidelines](#) explicitly ask people not to say things they wouldn't say face to face. If someone starts being rude, other users will step in and tell them to stop. And when people seem to be deliberately trolling, we ban them ruthlessly.

Technical tweaks may also help. On Reddit, votes on your comments don't affect your karma score, but they do on News.YC. And it does seem to influence people when they can see their reputation in the eyes of their peers drain away after making an asshole remark. Often users have second thoughts and delete such comments.

One might worry this would prevent people from expressing controversial ideas, but empirically that doesn't seem to be what happens. When people say something substantial that gets modded down, they stubbornly leave it up. What people delete are wisecracks, because they have less invested in them.

So far the experiment seems to be working. The level of conversation on News.YC is as high as on any forum I've seen. But we still only have about 8,000 uniques a day. The conversations on Reddit were good when it was that small. The challenge is whether we can keep things this way.

I'm optimistic we will. We're not depending just on technical tricks. The core users of News.YC are mostly refugees from other sites that were overrun by trolls. They feel about trolls roughly the way refugees from Cuba or Eastern Europe feel about dictatorships. So there are a lot of people working to keep this from happening again.

## Notes

[1] I mean forum in the general sense of a place to exchange views. The original Internet forums were not web sites but Usenet newsgroups.

[2] I'm talking here about everyday tagging. Some graffiti is quite impressive (anything becomes art if you do it well enough) but the median tag is just visual spam.

[Russian Translation](#)

# A New Venture Animal



March 2008, rev May 2013

*(This essay grew out of something I wrote for myself to figure out what we do. Even though Y Combinator is now 3 years old, we're still trying to understand its implications.)*

I was annoyed recently to read a description of Y Combinator that said "Y Combinator does seed funding for startups." What was especially annoying about it was that I wrote it. This doesn't really convey what we do. And the reason it's inaccurate is that, paradoxically, funding very early stage startups is not mainly about funding.

Saying YC does seed funding for startups is a description in terms of earlier models. It's like calling a car a horseless carriage.

When you scale animals you can't just keep everything in proportion. For example, volume grows as the cube of linear dimension, but surface area only as the square. So as animals get bigger they have trouble radiating heat. That's why mice and rabbits are furry and elephants and hippos aren't. You can't make a mouse by scaling down an elephant.

YC represents a new, smaller kind of animal—so much smaller that all the rules are different.

Before us, most companies in the startup funding business were venture capital funds. VCs generally fund later stage companies than we do. And they supply so much money that, even though the other things they do may be very valuable, it's not that inaccurate to regard VCs as sources of money. Good VCs are "smart money," but they're still money.

All good investors supply a combination of money and help. But these scale differently, just as volume and surface area do. Late stage investors supply huge amounts of money and comparatively little help: when a company about to go public gets a mezzanine round of \$50 million, the deal tends to be almost entirely about money. As you move earlier in the venture funding process, the ratio of help to money increases, because earlier stage companies have different needs. Early stage companies need less money because they're smaller and cheaper to run, but they need more help because life is so precarious for them. So when VCs do a series A round for, say, \$2 million, they generally expect to offer a significant amount of help along with the money.

Y Combinator occupies the earliest end of the spectrum. We're at least one and generally two steps before VC funding. (Though some startups go straight from YC to VC, the most common trajectory is to do an angel round first.) And what happens at Y Combinator is as different from what happens in a series A round as a series A round is from a mezzanine financing.

At our end, money is almost a negligible factor. The startup usually consists of just the founders. Their living expenses are the company's main expense, and since most founders are under 30, their living expenses are low. But at this early stage companies need a lot of help. Practically every question is still unanswered. Some companies we've funded have been working on their software for a year or more, but others haven't decided what to work on, or even who the founders should be.

When PR people and journalists recount the histories of startups after they've become big, they always underestimate how uncertain things were at first. They're not being deliberately misleading. When you look at a company like Google, it's hard to imagine they could once have been small and helpless. Sure, at one point they were a just a couple guys in a garage—but even then their greatness was assured, and all they had to do was roll forward along the railroad tracks of destiny.

Far from it. A lot of startups with just as promising beginnings end up failing. Google has such momentum now that it would be hard for anyone to stop them. But all it would have taken in the beginning would have been for two Google employees to focus on the wrong things for six months, and the company could have died.

We know, because we've been there, just how vulnerable startups are in the earliest phases. Curiously enough, that's why founders tend to get so rich from them. Reward is always proportionate to risk, and very early stage startups are insanely risky.

What we really do at Y Combinator is get startups launched straight. One of many metaphors you could use for YC is a steam catapult on an aircraft carrier. We get startups airborne. Barely airborne, but enough that they can accelerate fast.

When you're launching planes they have to be set up properly or you're just launching projectiles. They have to be pointed straight down the deck; the wings have to be trimmed properly; the engines have to be at full power; the pilot has to be ready. These are the kind of problems we deal with. After we fund startups we work closely with them for three months—so closely in fact that we insist they move to where we are. And what we do in those three months is make sure everything is set up for launch. If there are tensions between cofounders we help sort them out. We get all the paperwork set up properly so there are no nasty surprises later. If the founders aren't sure what to focus on first, we try to figure that out. If there is some obstacle right in front of them, we either try to remove it, or shift the startup sideways. The goal is to get every distraction out of the way so the founders can use that time to build (or finish building) something impressive. And then near the end of the three months we push the button on the steam catapult in the form of Demo Day, where the current group of startups present to pretty much every investor in Silicon Valley.

Launching companies isn't identical with launching products. Though we do spend a lot of time on launch strategies for products, there are some things that take too long to build for a startup to launch them before raising their next round of funding. Several of the most promising startups we've funded haven't launched their products yet, but are definitely launched as companies.

In the earliest stage, startups not only have more questions to answer, but they tend to be different kinds of questions. In later stage startups the questions are about deals, or hiring, or organization. In the earliest phase they tend to be about technology and design. What do you make? That's the first problem to solve. That's why our motto is "Make something people want." This is always a good thing for companies to do, but it's even more important early on, because it sets the bounds for every other question. Who you hire, how much money you raise, how you market yourself—they all depend on what you're making.

Because the early problems are so much about technology and design, you probably need to be hackers to do what we do. While some VCs have technical backgrounds, I don't know any who still write code. Their expertise is mostly in business—as it should be, because that's the kind of expertise you need in the phase between series A and (if you're lucky) IPO.

We're so different from VCs that we're really a different kind of animal. Can we claim founders are better off as a result of this new type of venture firm? I'm pretty sure the answer is yes, because YC is an improved version of what happened to our startup, and our case was not atypical. We started Viaweb with \$10,000 in seed money from our friend Julian. He was a lawyer and arranged all our paperwork, so we could just code. We spent three months building a version 1, which we then presented to investors to raise more money. Sounds familiar, doesn't it? But YC improves on that significantly. Julian knew a lot about law and business, but his advice ended there; he was not a startup guy. So we made some basic mistakes early on. And when we presented to investors, we presented to only 2, because that was all we knew. If we'd had our later selves to encourage and

advise us, and Demo Day to present at, we would have been in much better shape. We probably could have raised money at 3 to 5 times the valuation we did.

If we take 7% of a company we fund, the founders only have to do [7.5%](#) better in their next round of funding to end up net ahead. We certainly manage that.

So who is our 7% coming out of? If the founders end up net ahead it's not coming out of them. So is it coming out of later stage investors? Well, they do end up paying more. But I think they pay more because the company is actually more valuable. And later stage investors have no problem with that. The returns of a VC fund depend on the quality of the companies they invest in, not how cheaply they can buy stock in them.

If what we do is useful, why wasn't anyone doing it before? There are two answers to that. One is that people were doing it before, just haphazardly on a smaller scale. Before us, seed funding came primarily from individual angel investors. Larry and Sergey, for example, got their seed funding from Andy Bechtolsheim, one of the founders of Sun. And because he was a startup guy he probably gave them useful advice. But raising money from angel investors is a hit or miss thing. It's a sideline for most of them, so they only do a handful of deals a year and they don't spend a lot of time on the startups they invest in. And they're hard to reach, because they don't want random startups pestering them with business plans. The Google guys were lucky because they knew someone who knew Bechtolsheim. It generally takes a personal introduction with angels.

The other reason no one was doing quite what we do is that till recently it was a lot more expensive to start a startup. You'll notice we haven't funded any biotech startups. That's still expensive. But advancing technology has made web startups so cheap that you really can get a company airborne for \$15,000. If you understand how to operate a steam catapult, at least.

So in effect what's happened is that a new ecological niche has opened up, and Y Combinator is the new kind of animal that has moved into it. We're not a replacement for venture capital funds. We occupy a new, adjacent niche. And conditions in our niche are really quite different. It's not just that the problems we face are different; the whole structure of the business is different. VCs are playing a zero-sum game. They're all competing for a slice of a fixed amount of "deal flow," and that explains a lot of their behavior. Whereas our m.o. is to create new deal flow, by encouraging hackers who would have gotten jobs to start their own startups instead. We compete more with employers than VCs.

It's not surprising something like this would happen. Most fields become more specialized—more articulated—as they develop, and startups are certainly an area in which there has been a lot of development over the past couple decades. The venture business in its present form is only about forty years old. It stands to reason it would evolve.

And it's natural that the new niche would at first be described, even by its

inhabitants, in terms of the old one. But really Y Combinator is not in the startup funding business. Really we're more of a small, furry steam catapult.

**Thanks** to Trevor Blackwell, Jessica Livingston, and Robert Morris for reading drafts of this.



[Comment](#) on this essay.



# You Weren't Meant to Have a Boss



March 2008, rev. June 2008

Technology tends to separate normal from natural. Our bodies weren't designed to eat the foods that people in rich countries eat, or to get so little exercise. There may be a similar problem with the way we work: a normal job may be as bad for us intellectually as white flour or sugar is for us physically.

I began to suspect this after spending several years working with startup founders. I've now worked with over 200 of them, and I've noticed a definite difference between programmers working on their own startups and those working for large organizations. I wouldn't say founders seem happier, necessarily; starting a startup can be very stressful. Maybe the best way to put it is to say that they're happier in the sense that your body is happier during a long run than sitting on a sofa eating doughnuts.

Though they're statistically abnormal, startup founders seem to be working in a way that's more natural for humans.

I was in Africa last year and saw a lot of animals in the wild that I'd only seen in zoos before. It was remarkable how different they seemed. Particularly lions. Lions in the wild seem about ten times more alive. They're like different animals. I suspect that working for oneself feels better to humans in much the same way that living in the wild must feel better to a wide-ranging predator like a lion. Life in a zoo is easier, but it isn't the life they were designed for.

## Trees

What's so unnatural about working for a big company? The root of the problem is that humans weren't meant to work in such large groups.

Another thing you notice when you see animals in the wild is that each species

thrives in groups of a certain size. A herd of impalas might have 100 adults; baboons maybe 20; lions rarely 10. Humans also seem designed to work in groups, and what I've read about hunter-gatherers accords with research on organizations and my own experience to suggest roughly what the ideal size is: groups of 8 work well; by 20 they're getting hard to manage; and a group of 50 is really unwieldy. [\[1\]](#)

Whatever the upper limit is, we are clearly not meant to work in groups of several hundred. And yet—for reasons having more to do with technology than human nature—a great many people work for companies with hundreds or thousands of employees.

Companies know groups that large wouldn't work, so they divide themselves into units small enough to work together. But to coordinate these they have to introduce something new: bosses.

These smaller groups are always arranged in a tree structure. Your boss is the point where your group attaches to the tree. But when you use this trick for dividing a large group into smaller ones, something strange happens that I've never heard anyone mention explicitly. In the group one level up from yours, your boss represents your entire group. A group of 10 managers is not merely a group of 10 people working together in the usual way. It's really a group of groups. Which means for a group of 10 managers to work together as if they were simply a group of 10 individuals, the group working for each manager would have to work as if they were a single person—the workers and manager would each share only one person's worth of freedom between them.

In practice a group of people are never able to act as if they were one person. But in a large organization divided into groups in this way, the pressure is always in that direction. Each group tries its best to work as if it were the small group of individuals that humans were designed to work in. That was the point of creating it. And when you propagate that constraint, the result is that each person gets freedom of action in inverse proportion to the size of the entire tree. [\[2\]](#)

Anyone who's worked for a large organization has felt this. You can feel the difference between working for a company with 100 employees and one with 10,000, even if your group has only 10 people.

## **Corn Syrup**

A group of 10 people within a large organization is a kind of fake tribe. The number of people you interact with is about right. But something is missing: individual initiative. Tribes of hunter-gatherers have much more freedom. The leaders have a little more power than other members of the tribe, but they don't generally tell them what to do and when the way a boss can.

It's not your boss's fault. The real problem is that in the group above you in the hierarchy, your entire group is one virtual person. Your boss is just the way that

constraint is imparted to you.

So working in a group of 10 people within a large organization feels both right and wrong at the same time. On the surface it feels like the kind of group you're meant to work in, but something major is missing. A job at a big company is like high fructose corn syrup: it has some of the qualities of things you're meant to like, but is disastrously lacking in others.

Indeed, food is an excellent metaphor to explain what's wrong with the usual sort of job.

For example, working for a big company is the default thing to do, at least for programmers. How bad could it be? Well, food shows that pretty clearly. If you were dropped at a random point in America today, nearly all the food around you would be bad for you. Humans were not designed to eat white flour, refined sugar, high fructose corn syrup, and hydrogenated vegetable oil. And yet if you analyzed the contents of the average grocery store you'd probably find these four ingredients accounted for most of the calories. "Normal" food is terribly bad for you. The only people who eat what humans were actually designed to eat are a few Birkenstock-wearing weirdos in Berkeley.

If "normal" food is so bad for us, why is it so common? There are two main reasons. One is that it has more immediate appeal. You may feel lousy an hour after eating that pizza, but eating the first couple bites feels great. The other is economies of scale. Producing junk food scales; producing fresh vegetables doesn't. Which means (a) junk food can be very cheap, and (b) it's worth spending a lot to market it.

If people have to choose between something that's cheap, heavily marketed, and appealing in the short term, and something that's expensive, obscure, and appealing in the long term, which do you think most will choose?

It's the same with work. The average MIT graduate wants to work at Google or Microsoft, because it's a recognized brand, it's safe, and they'll get paid a good salary right away. It's the job equivalent of the pizza they had for lunch. The drawbacks will only become apparent later, and then only in a vague sense of malaise.

And founders and early employees of startups, meanwhile, are like the Birkenstock-wearing weirdos of Berkeley: though a tiny minority of the population, they're the ones living as humans are meant to. In an artificial world, only extremists live naturally.

## **Programmers**

The restrictiveness of big company jobs is particularly hard on programmers, because the essence of programming is to build new things. Sales people make much the same pitches every day; support people answer much the same

questions; but once you've written a piece of code you don't need to write it again. So a programmer working as programmers are meant to is always making new things. And when you're part of an organization whose structure gives each person freedom in inverse proportion to the size of the tree, you're going to face resistance when you do something new.

This seems an inevitable consequence of bigness. It's true even in the smartest companies. I was talking recently to a founder who considered starting a startup right out of college, but went to work for Google instead because he thought he'd learn more there. He didn't learn as much as he expected. Programmers learn by doing, and most of the things he wanted to do, he couldn't—sometimes because the company wouldn't let him, but often because the company's code wouldn't let him. Between the drag of legacy code, the overhead of doing development in such a large organization, and the restrictions imposed by interfaces owned by other groups, he could only try a fraction of the things he would have liked to. He said he has learned much more in his own startup, despite the fact that he has to do all the company's errands as well as programming, because at least when he's programming he can do whatever he wants.

An obstacle downstream propagates upstream. If you're not allowed to implement new ideas, you stop having them. And vice versa: when you can do whatever you want, you have more ideas about what to do. So working for yourself makes your brain more powerful in the same way a low-restriction exhaust system makes an engine more powerful.

Working for yourself doesn't have to mean starting a startup, of course. But a programmer deciding between a regular job at a big company and their own startup is probably going to learn more doing the startup.

You can adjust the amount of freedom you get by scaling the size of company you work for. If you start the company, you'll have the most freedom. If you become one of the first 10 employees you'll have almost as much freedom as the founders. Even a company with 100 people will feel different from one with 1000.

Working for a small company doesn't ensure freedom. The tree structure of large organizations sets an upper bound on freedom, not a lower bound. The head of a small company may still choose to be a tyrant. The point is that a large organization is compelled by its structure to be one.

## **Consequences**

That has real consequences for both organizations and individuals. One is that companies will inevitably slow down as they grow larger, no matter how hard they try to keep their startup mojo. It's a consequence of the tree structure that every large organization is forced to adopt.

Or rather, a large organization could only avoid slowing down if they avoided tree structure. And since human nature limits the size of group that can work together,

the only way I can imagine for larger groups to avoid tree structure would be to have no structure: to have each group actually be independent, and to work together the way components of a market economy do.

That might be worth exploring. I suspect there are already some highly partitionable businesses that lean this way. But I don't know any technology companies that have done it.

There is one thing companies can do short of structuring themselves as sponges: they can stay small. If I'm right, then it really pays to keep a company as small as it can be at every stage. Particularly a technology company. Which means it's doubly important to hire the best people. Mediocre hires hurt you twice: they get less done, but they also make you big, because you need more of them to solve a given problem.

For individuals the upshot is the same: aim small. It will always suck to work for large organizations, and the larger the organization, the more it will suck.

In an [essay](#) I wrote a couple years ago I advised graduating seniors to work for a couple years for another company before starting their own. I'd modify that now. Work for another company if you want to, but only for a small one, and if you want to start your own startup, go ahead.

The reason I suggested college graduates not start startups immediately was that I felt most would fail. And they will. But ambitious programmers are better off doing their own thing and failing than going to work at a big company. Certainly they'll learn more. They might even be better off financially. A lot of people in their early twenties get into debt, because their expenses grow even faster than the salary that seemed so high when they left school. At least if you start a startup and fail your net worth will be zero rather than negative. [3]

We've now funded so many different types of founders that we have enough data to see patterns, and there seems to be no benefit from working for a big company. The people who've worked for a few years do seem better than the ones straight out of college, but only because they're that much older.

The people who come to us from big companies often seem kind of conservative. It's hard to say how much is because big companies made them that way, and how much is the natural conservatism that made them work for the big companies in the first place. But certainly a large part of it is learned. I know because I've seen it burn off.

Having seen that happen so many times is one of the things that convinces me that working for oneself, or at least for a small group, is the natural way for programmers to live. Founders arriving at Y Combinator often have the downtrodden air of refugees. Three months later they're transformed: they have so much more [confidence](#) that they seem as if they've grown several inches taller. [4] Strange as this sounds, they seem both more worried and happier at the same

time. Which is exactly how I'd describe the way lions seem in the wild.

Watching employees get transformed into founders makes it clear that the difference between the two is due mostly to environment—and in particular that the environment in big companies is toxic to programmers. In the first couple weeks of working on their own startup they seem to come to life, because finally they're working the way people are meant to.

## Notes

[1] When I talk about humans being meant or designed to live a certain way, I mean by evolution.

[2] It's not only the leaves who suffer. The constraint propagates up as well as down. So managers are constrained too; instead of just doing things, they have to act through subordinates.

[3] Do not finance your startup with credit cards. Financing a startup with debt is usually a stupid move, and credit card debt stupidest of all. Credit card debt is a bad idea, period. It is a trap set by evil companies for the desperate and the foolish.

[4] The founders we fund used to be younger (initially we encouraged undergrads to apply), and the first couple times I saw this I used to wonder if they were actually getting physically taller.

**Thanks** to Trevor Blackwell, Ross Boucher, Aaron Iba, Abby Kirigin, Ivan Kirigin, Jessica Livingston, and Robert Morris for reading drafts of this.

[French Translation](#)

[Russian Translation](#)

# How to Disagree

March 2008

The web is turning writing into a conversation. Twenty years ago, writers wrote and readers read. The web lets readers respond, and increasingly they do—in comment threads, on forums, and in their own blog posts.

Many who respond to something disagree with it. That's to be expected. Agreeing tends to motivate people less than disagreeing. And when you agree there's less to say. You could expand on something the author said, but he has probably already explored the most interesting implications. When you disagree you're entering territory he may not have explored.

The result is there's a lot more disagreeing going on, especially measured by the word. That doesn't mean people are getting angrier. The structural change in the way we communicate is enough to account for it. But though it's not anger that's driving the increase in disagreement, there's a danger that the increase in disagreement will make people angrier. Particularly online, where it's easy to say things you'd never say face to face.

If we're all going to be disagreeing more, we should be careful to do it well. What does it mean to disagree well? Most readers can tell the difference between mere name-calling and a carefully reasoned refutation, but I think it would help to put names on the intermediate stages. So here's an attempt at a disagreement hierarchy:

## **DH0. Name-calling.**

This is the lowest form of disagreement, and probably also the most common. We've all seen comments like this:

u r a fag!!!!!!!!!!

But it's important to realize that more articulate name-calling has just as little weight. A comment like

The author is a self-important dilettante.

is really nothing more than a pretentious version of "u r a fag."

## **DH1. Ad Hominem.**

An ad hominem attack is not quite as weak as mere name-calling. It might actually carry some weight. For example, if a senator wrote an article saying senators' salaries should be increased, one could respond:

Of course he would say that. He's a senator.

This wouldn't refute the author's argument, but it may at least be relevant to the case. It's still a very weak form of disagreement, though. If there's something wrong with the senator's argument, you should say what it is; and if there isn't, what difference does it make that he's a senator?

Saying that an author lacks the authority to write about a topic is a variant of ad hominem—and a particularly useless sort, because good ideas often come from outsiders. The question is whether the author is correct or not. If his lack of authority caused him to make mistakes, point those out. And if it didn't, it's not a problem.

## **DH2. Responding to Tone.**

The next level up we start to see responses to the writing, rather than the writer. The lowest form of these is to disagree with the author's tone. E.g.

I can't believe the author dismisses intelligent design in such a cavalier fashion.

Though better than attacking the author, this is still a weak form of disagreement. It matters much more whether the author is wrong or right than what his tone is. Especially since tone is so hard to judge. Someone who has a chip on their shoulder about some topic might be offended by a tone that to other readers seemed neutral.

So if the worst thing you can say about something is to criticize its tone, you're not saying much. Is the author flippant, but correct? Better that than grave and wrong. And if the author is incorrect somewhere, say where.

## **DH3. Contradiction.**

In this stage we finally get responses to what was said, rather than how or by whom. The lowest form of response to an argument is simply to state the opposing case, with little or no supporting evidence.

This is often combined with DH2 statements, as in:

I can't believe the author dismisses intelligent design in such a cavalier fashion. Intelligent design is a legitimate scientific theory.

Contradiction can sometimes have some weight. Sometimes merely seeing the opposing case stated explicitly is enough to see that it's right. But usually evidence



will help.

#### **DH4. Counterargument.**

At level 4 we reach the first form of convincing disagreement: counterargument. Forms up to this point can usually be ignored as proving nothing. Counterargument might prove something. The problem is, it's hard to say exactly what.

Counterargument is contradiction plus reasoning and/or evidence. When aimed squarely at the original argument, it can be convincing. But unfortunately it's common for counterarguments to be aimed at something slightly different. More often than not, two people arguing passionately about something are actually arguing about two different things. Sometimes they even agree with one another, but are so caught up in their squabble they don't realize it.

There could be a legitimate reason for arguing against something slightly different from what the original author said: when you feel they missed the heart of the matter. But when you do that, you should say explicitly you're doing it.

#### **DH5. Refutation.**

The most convincing form of disagreement is refutation. It's also the rarest, because it's the most work. Indeed, the disagreement hierarchy forms a kind of pyramid, in the sense that the higher you go the fewer instances you find.

To refute someone you probably have to quote them. You have to find a "smoking gun," a passage in whatever you disagree with that you feel is mistaken, and then explain why it's mistaken. If you can't find an actual quote to disagree with, you may be arguing with a straw man.

While refutation generally entails quoting, quoting doesn't necessarily imply refutation. Some writers quote parts of things they disagree with to give the appearance of legitimate refutation, then follow with a response as low as DH3 or even DH0.

#### **DH6. Refuting the Central Point.**

The force of a refutation depends on what you refute. The most powerful form of disagreement is to refute someone's central point.

Even as high as DH5 we still sometimes see deliberate dishonesty, as when someone picks out minor points of an argument and refutes those. Sometimes the spirit in which this is done makes it more of a sophisticated form of ad hominem than actual refutation. For example, correcting someone's grammar, or harping on minor mistakes in names or numbers. Unless the opposing argument actually depends on such things, the only purpose of correcting them is to discredit one's opponent.

Truly refuting something requires one to refute its central point, or at least one of them. And that means one has to commit explicitly to what the central point is. So a truly effective refutation would look like:

The author's main point seems to be x. As he says:

<quotation>

But this is wrong for the following reasons...

The quotation you point out as mistaken need not be the actual statement of the author's main point. It's enough to refute something it depends upon.

## **What It Means**

Now we have a way of classifying forms of disagreement. What good is it? One thing the disagreement hierarchy *doesn't* give us is a way of picking a winner. DH levels merely describe the form of a statement, not whether it's correct. A DH6 response could still be completely mistaken.

But while DH levels don't set a lower bound on the convincingness of a reply, they do set an upper bound. A DH6 response might be unconvincing, but a DH2 or lower response is always unconvincing.

The most obvious advantage of classifying the forms of disagreement is that it will help people to evaluate what they read. In particular, it will help them to see through intellectually dishonest arguments. An eloquent speaker or writer can give the impression of vanquishing an opponent merely by using forceful words. In fact that is probably the defining quality of a demagogue. By giving names to the different forms of disagreement, we give critical readers a pin for popping such balloons.

Such labels may help writers too. Most intellectual dishonesty is unintentional. Someone arguing against the tone of something he disagrees with may believe he's really saying something. Zooming out and seeing his current position on the disagreement hierarchy may inspire him to try moving up to counterargument or refutation.

But the greatest benefit of disagreeing well is not just that it will make conversations better, but that it will make the people who have them happier. If you study conversations, you find there is a lot more meanness down in DH1 than up in DH6. You don't have to be mean when you have a real point to make. In fact, you don't want to. If you have something real to say, being mean just gets in the way.

If moving up the disagreement hierarchy makes people less mean, that will make most of them happier. Most people don't really enjoy being mean; they do it because they can't help it.

**Thanks** to Trevor Blackwell and Jessica Livingston for reading drafts of this.

**Related:**

[What You Can't Say](#)

[The Age of the Essay](#)

[Italian Translation](#)

[Russian Translation](#)

[Swedish Translation](#)

[Spanish Translation](#)

[German Translation](#)

[French Translation](#)

[Arabic Translation](#)

[Finnish Translation](#)

[Italian Translation](#)

[Turkish Translation](#)

# Some Heroes

April 2008

There are some topics I save up because they'll be so much fun to write about. This is one of them: a list of my heroes.

I'm not claiming this is a list of the  $n$  most admirable people. Who could make such a list, even if they wanted to?

Einstein isn't on the list, for example, even though he probably deserves to be on any shortlist of admirable people. I once asked a physicist friend if Einstein was really as smart as his fame implies, and she said that yes, he was. So why isn't he on the list? Because I had to ask. This is a list of people who've influenced me, not people who would have if I understood their work.

My test was to think of someone and ask "is this person my hero?" It often returned surprising answers. For example, it returned false for Montaigne, who was arguably the inventor of the essay. Why? When I thought about what it meant to call someone a hero, it meant I'd decide what to do by asking what they'd do in the same situation. That's a stricter standard than admiration.

After I made the list, I looked to see if there was a pattern, and there was, a very clear one. Everyone on the list had two qualities: they cared almost excessively about their work, and they were absolutely honest. By honest I don't mean trustworthy so much as that they never pander: they never say or do something because that's what the audience wants. They are all fundamentally subversive for this reason, though they conceal it to varying degrees.

## Jack Lambert

I grew up in Pittsburgh in the 1970s. Unless you were there it's hard to imagine how that town felt about the Steelers. Locally, all the news was bad. The steel industry was dying. But the Steelers were the best team in football — and moreover, in a way that seemed to reflect the personality of the city. They didn't do anything fancy. They just got the job done.

Other players were more famous: Terry Bradshaw, Franco Harris, Lynn Swann. But they played offense, and you always get more attention for that. It seemed to me as a twelve year old football expert that the best of them all was [Jack Lambert](#).

And what made him so good was that he was utterly relentless. He didn't just care about playing well; he cared almost too much. He seemed to regard it as a personal insult when someone from the other team had possession of the ball on his side of the line of scrimmage.

The suburbs of Pittsburgh in the 1970s were a pretty dull place. School was boring. All the adults around were bored with their jobs working for big companies. Everything that came to us through the mass media was (a) blandly uniform and (b) produced elsewhere. Jack Lambert was the exception. He was like nothing else I'd seen.

## **Kenneth Clark**

Kenneth Clark is the best nonfiction writer I know of, on any subject. Most people who write about art history don't really like art; you can tell from a thousand little signs. But Clark did, and not just intellectually, but the way one anticipates a delicious dinner.

What really makes him stand out, though, is the quality of his ideas. His style is deceptively casual, but there is more in his books than in a library of art monographs. Reading [The Nude](#) is like a ride in a Ferrari. Just as you're getting settled, you're slammed back in your seat by the acceleration. Before you can adjust, you're thrown sideways as the car screeches into the first turn. His brain throws off ideas almost too fast to grasp them. Finally at the end of the chapter you come to a halt, with your eyes wide and a big smile on your face.

Kenneth Clark was a star in his day, thanks to the documentary series [Civilisation](#). And if you read only one book about art history, [Civilisation](#) is the one I'd recommend. It's much better than the drab Sears Catalogs of art that undergraduates are forced to buy for Art History 101.

## **Larry Mihalko**

A lot of people have a great teacher at some point in their childhood. Larry Mihalko was mine. When I look back it's like there's a line drawn between third and fourth grade. After Mr. Mihalko, everything was different.

Why? First of all, he was intellectually curious. I had a few other teachers who were smart, but I wouldn't describe them as intellectually curious. In retrospect, he was out of place as an elementary school teacher, and I think he knew it. That must have been hard for him, but it was wonderful for us, his students. His class was a constant adventure. I used to like going to school every day.

The other thing that made him different was that he liked us. Kids are good at telling that. The other teachers were at best benevolently indifferent. But Mr. Mihalko seemed like he actually wanted to be our friend. On the last day of fourth grade, he got out one of the heavy school record players and played James Taylor's "You've Got a Friend" to us. Just call out my name, and you know wherever I am,

I'll come running. He died at 59 of lung cancer. I've never cried like I cried at his funeral.

## **Leonardo**

One of the things I've learned about making things that I didn't realize when I was a kid is that much of the best stuff isn't made for audiences, but for oneself. You see paintings and drawings in museums and imagine they were made for you to look at. Actually a lot of the best ones were made as a way of exploring the world, not as a way to please other people. The best of these explorations are sometimes more pleasing than stuff made explicitly to please.

Leonardo did a lot of things. One of his most admirable qualities was that he did so many different things that were admirable. What people know of him now is his paintings and his more flamboyant inventions, like flying machines. That makes him seem like some kind of dreamer who sketched artists' conceptions of rocket ships on the side. In fact he made a large number of far more practical technical discoveries. He was as good an engineer as a painter.

His most impressive work, to me, is his [drawings](#). They're clearly made more as a way of studying the world than producing something beautiful. And yet they can hold their own with any work of art ever made. No one else, before or since, was that good when no one was looking.

## **Robert Morris**

Robert Morris has a very unusual quality: he's never wrong. It might seem this would require you to be omniscient, but actually it's surprisingly easy. Don't say anything unless you're fairly sure of it. If you're not omniscient, you just don't end up saying much.

More precisely, the trick is to pay careful attention to how you qualify what you say. By using this trick, Robert has, as far as I know, managed to be mistaken only once, and that was when he was an undergrad. When the Mac came out, he said that little desktop computers would never be suitable for real hacking.

It's wrong to call it a trick in his case, though. If it were a conscious trick, he would have slipped in a moment of excitement. With Robert this quality is wired-in. He has an almost superhuman integrity. He's not just generally correct, but also correct about how correct he is.

You'd think it would be such a great thing never to be wrong that everyone would do this. It doesn't seem like that much extra work to pay as much attention to the error on an idea as to the idea itself. And yet practically no one does. I know how hard it is, because since meeting Robert I've tried to do in software what he seems to do in hardware.

## **P. G. Wodehouse**

People are finally starting to admit that Wodehouse was a great writer. If you want to be thought a great novelist in your own time, you have to sound intellectual. If what you write is popular, or entertaining, or funny, you're ipso facto suspect. That makes Wodehouse doubly impressive, because it meant that to write as he wanted to, he had to commit to being despised in his own lifetime.

Evelyn Waugh called him a great writer, but to most people at the time that would have read as a chivalrous or deliberately perverse gesture. At the time any random autobiographical novel by a recent college grad could count on more respectful treatment from the literary establishment.

Wodehouse may have begun with simple atoms, but the way he composed them into molecules was near faultless. His rhythm in particular. It makes me self-conscious to write about it. I can think of only two other writers who came near him for style: Evelyn Waugh and Nancy Mitford. Those three used the English language like they owned it.

But Wodehouse has something neither of them did. He's at ease. Evelyn Waugh and Nancy Mitford cared what other people thought of them: he wanted to seem aristocratic; she was afraid she wasn't smart enough. But Wodehouse didn't give a damn what anyone thought of him. He wrote exactly what he wanted.

## **Alexander Calder**

Calder's on this list because he makes me happy. Can his work stand up to Leonardo's? Probably not. There might not be anything from the 20th Century that can. But what was good about Modernism, Calder had, and had in a way that he made seem effortless.

What was good about Modernism was its freshness. Art became stuffy in the nineteenth century. The paintings that were popular at the time were mostly the art equivalent of McMansions—big, pretentious, and fake. Modernism meant starting over, making things with the same earnest motives that children might. The artists who benefited most from this were the ones who had preserved a child's confidence, like Klee and Calder.

Klee was impressive because he could work in so many different styles. But between the two I like Calder better, because his work seemed happier. Ultimately the point of art is to engage the viewer. It's hard to predict what will; often something that seems interesting at first will bore you after a month. Calder's [sculptures](#) never get boring. They just sit there quietly radiating optimism, like a battery that never runs out. As far as I can tell from books and photographs, the happiness of Calder's work is his own happiness showing through.

## **Jane Austen**

Everyone admires Jane Austen. Add my name to the list. To me she seems the

best novelist of all time.

I'm interested in how things work. When I read most novels, I pay as much attention to the author's choices as to the story. But in her novels I can't see the gears at work. Though I'd really like to know how she does what she does, I can't figure it out, because she's so good that her stories don't seem made up. I feel like I'm reading a description of something that actually happened.

I used to read a lot of novels when I was younger. I can't read most anymore, because they don't have enough information in them. Novels seem so impoverished compared to history and biography. But reading Austen is like reading nonfiction. She writes so well you don't even notice her.

## **John McCarthy**

John McCarthy invented Lisp, the field of (or at least the term) artificial intelligence, and was an early member of both of the top two computer science departments, MIT and Stanford. No one would dispute that he's one of the greats, but he's an especial hero to me because of [Lisp](#).

It's hard for us now to understand what a conceptual leap that was at the time. Paradoxically, one of the reasons his achievement is hard to appreciate is that it was so successful. Practically every programming language invented in the last 20 years includes ideas from Lisp, and each year the median language gets more Lisplike.

In 1958 these ideas were anything but obvious. In 1958 there seem to have been two ways of thinking about programming. Some people thought of it as math, and proved things about Turing Machines. Others thought of it as a way to get things done, and designed languages all too influenced by the technology of the day. McCarthy alone bridged the gap. He designed a language that was math. But designed is not really the word; discovered is more like it.

## **The Spitfire**

As I was making this list I found myself thinking of people like [Douglas Bader](#) and [R.J. Mitchell](#) and [Jeffrey Quill](#) and I realized that though all of them had done many things in their lives, there was one factor above all that connected them: the Spitfire.

This is supposed to be a list of heroes. How can a machine be on it? Because that machine was not just a machine. It was a lens of heroes. Extraordinary devotion went into it, and extraordinary courage came out.

It's a cliché to call World War II a contest between good and evil, but between fighter designs, it really was. The Spitfire's original nemesis, the ME 109, was a brutally practical plane. It was a killing machine. The Spitfire was optimism embodied. And not just in its beautiful lines: it was at the edge of what could be



manufactured. But taking the high road worked. In the air, beauty had the edge, just.

## **Steve Jobs**

People alive when Kennedy was killed usually remember exactly where they were when they heard about it. I remember exactly where I was when a friend asked if I'd heard Steve Jobs had cancer. It was like the floor dropped out. A few seconds later she told me that it was a rare operable type, and that he'd be ok. But those seconds seemed long.

I wasn't sure whether to include Jobs on this list. A lot of people at Apple seem to be afraid of him, which is a bad sign. But he compels admiration.

There's no name for what Steve Jobs is, because there hasn't been anyone quite like him before. He doesn't design Apple's products himself. Historically the closest analogy to what he does are the great Renaissance patrons of the arts. As the CEO of a company, that makes him unique.

Most CEOs delegate [taste](#) to a subordinate. The [design paradox](#) means they're choosing more or less at random. But Steve Jobs actually has taste himself — such good taste that he's shown the world how much more important taste is than they realized.

## **Isaac Newton**

Newton has a strange role in my pantheon of heroes: he's the one I reproach myself with. He worked on big things, at least for part of his life. It's so easy to get distracted working on small stuff. The questions you're answering are pleasantly familiar. You get immediate rewards — in fact, you get bigger rewards in your time if you work on matters of passing importance. But I'm uncomfortably aware that this is the route to well-deserved obscurity.

To do really great things, you have to seek out questions people didn't even realize were questions. There have probably been other people who did this as well as Newton, for their time, but Newton is my model of this kind of thought. I can just begin to understand what it must have felt like for him.

You only get one life. Why not do something huge? The phrase "paradigm shift" is overused now, but Kuhn was onto something. And you know more are out there, separated from us by what will later seem a surprisingly thin wall of laziness and stupidity. If we work like Newton.

**Thanks** to Trevor Blackwell, Jessica Livingston, and Jackie McDonough for reading drafts of this.

[Japanese Translation](#)

# Why There Aren't More Googles

April 2008

Umair Haque [wrote](#) recently that the reason there aren't more Googles is that most startups get bought before they can change the world.

Google, despite serious interest from Microsoft and Yahoo—what must have seemed like lucrative interest at the time—didn't sell out. Google might simply have been nothing but Yahoo's or MSN's search box.

Why isn't it? Because Google had a deeply felt sense of purpose: a conviction to change the world for the better.

This has a nice sound to it, but it isn't true. Google's founders were willing to sell early on. They just wanted more than acquirers were willing to pay.

It was the same with Facebook. They would have sold, but Yahoo blew it by offering too little.

Tip for acquirers: when a startup turns you down, consider raising your offer, because there's a good chance the outrageous price they want will later seem a bargain. [[1](#)]

From the evidence I've seen so far, startups that turn down acquisition offers usually end up doing better. Not always, but usually there's a bigger offer coming, or perhaps even an IPO.

Of course, the reason startups do better when they turn down acquisition offers is not necessarily that all such offers undervalue startups. More likely the reason is that the kind of founders who have the balls to turn down a big offer also tend to be very successful. That spirit is exactly what you want in a startup.

While I'm sure Larry and Sergey do want to change the world, at least now, the reason Google survived to become a big, independent company is the same reason Facebook has so far remained independent: acquirers underestimated them.

Corporate M&A is a strange business in that respect. They consistently lose the best deals, because turning down reasonable offers is the most reliable test you could invent for whether a startup will make it big.

## **VCs**

So what's the real reason there aren't more Googles? Curiously enough, it's the same reason Google and Facebook have remained independent: money guys undervalue the most innovative startups.

The reason there aren't more Googles is not that investors encourage innovative startups to sell out, but that they won't even fund them. I've learned a lot about VCs during the 3 years we've been doing Y Combinator, because we often have to work quite closely with them. The most surprising thing I've learned is how conservative they are. VC firms present an image of boldly encouraging innovation. Only a handful actually do, and even they are more conservative in reality than you'd guess from reading their sites.

I used to think of VCs as piratical: bold but unscrupulous. On closer acquaintance they turn out to be more like bureaucrats. They're more upstanding than I used to think (the good ones, at least), but less bold. Maybe the VC industry has changed. Maybe they used to be bolder. But I suspect it's the startup world that has changed, not them. The low cost of starting a startup means the average good bet is a riskier one, but most existing VC firms still operate as if they were investing in hardware startups in 1985.

Howard Aiken said "Don't worry about people stealing your ideas. If your ideas are any good, you'll have to ram them down people's throats." I have a similar feeling when I'm trying to convince VCs to invest in startups Y Combinator has funded. They're terrified of really novel ideas, unless the founders are good enough salesmen to compensate.

But it's the bold ideas that generate the biggest returns. Any really good new idea will seem bad to most people; otherwise someone would already be doing it. And yet most VCs are driven by consensus, not just within their firms, but within the VC community. The biggest factor determining how a VC will feel about your startup is how other VCs feel about it. I doubt they realize it, but this algorithm guarantees they'll miss all the very best ideas. The more people who have to like a new idea, the more outliers you lose.

Whoever the next Google is, they're probably being told right now by VCs to come back when they have more "traction."

Why are VCs so conservative? It's probably a combination of factors. The large size of their investments makes them conservative. Plus they're investing other people's money, which makes them worry they'll get in trouble if they do something risky and it fails. Plus most of them are money guys rather than technical guys, so they don't understand what the startups they're investing in do.

## **What's Next**

The exciting thing about market economies is that stupidity equals opportunity.

And so it is in this case. There is a huge, unexploited opportunity in startup investing. Y Combinator funds startups at the very beginning. VCs will fund them once they're already starting to succeed. But between the two there is a substantial gap.

There are companies that will give \$20k to a startup that has nothing more than the founders, and there are companies that will give \$2 million to a startup that's already taking off, but there aren't enough investors who will give \$200k to a startup that seems very promising but still has some things to figure out. This territory is occupied mostly by individual angel investors—people like Andy Bechtolsheim, who gave Google \$100k when they seemed promising but still had some things to figure out. I like angels, but there just aren't enough of them, and investing is for most of them a part time job.

And yet as it gets cheaper to start startups, this sparsely occupied territory is becoming more and more valuable. Nowadays a lot of startups don't want to raise multi-million dollar series A rounds. They don't need that much money, and they don't want the hassles that come with it. The median startup coming out of Y Combinator wants to raise \$250-500k. When they go to VC firms they have to ask for more because they know VCs aren't interested in such small deals.

VCs are money managers. They're looking for ways to put large sums to work. But the startup world is evolving away from their current model.

Startups have gotten cheaper. That means they want less money, but also that there are more of them. So you can still get large returns on large amounts of money; you just have to spread it more broadly.

I've tried to explain this to VC firms. Instead of making one \$2 million investment, make five \$400k investments. Would that mean sitting on too many boards? Don't sit on their boards. Would that mean too much due diligence? Do less. If you're investing at a tenth the valuation, you only have to be a tenth as sure.

It seems obvious. But I've proposed to several VC firms that they set aside some money and designate one partner to make more, smaller bets, and they react as if I'd proposed the partners all get nose rings. It's remarkable how wedded they are to their standard m.o.

But there is a big opportunity here, and one way or the other it's going to get filled. Either VCs will evolve down into this gap or, more likely, new investors will appear to fill it. That will be a good thing when it happens, because these new investors will be compelled by the structure of the investments they make to be ten times bolder than present day VCs. And that will get us a lot more Googles. At least, as long as acquirers remain stupid.

## Notes

[1] Another tip: If you want to get all that value, don't destroy the startup after you buy it. Give the founders enough autonomy that they can grow the acquisition into what it would have become.

**Thanks** to Sam Altman, Paul Buchheit, David Hornik, Jessica Livingston, Robert Morris, and Fred Wilson for reading drafts of this.

[Russian Translation](#)

# Be Good



April 2008

*(This essay is derived from a talk at the 2008 Startup School.)*

About a month after we started Y Combinator we came up with the phrase that became our motto: Make something people want. We've learned a lot since then, but if I were choosing now that's still the one I'd pick.

Another thing we tell founders is not to worry too much about the business model, at least at first. Not because making money is unimportant, but because it's so much easier than building something great.

A couple weeks ago I realized that if you put those two ideas together, you get something surprising. Make something people want. Don't worry too much about making money. What you've got is a description of a charity.

When you get an unexpected result like this, it could either be a bug or a new discovery. Either businesses aren't supposed to be like charities, and we've proven by *reductio ad absurdum* that one or both of the principles we began with is false. Or we have a new idea.

I suspect it's the latter, because as soon as this thought occurred to me, a whole bunch of other things fell into place.

## Examples

For example, Craigslist. It's not a charity, but they run it like one. And they're astoundingly successful. When you scan down the list of most popular web sites, the number of employees at Craigslist looks like a misprint. Their revenues aren't

as high as they could be, but most startups would be happy to trade places with them.

In Patrick O'Brian's novels, his captains always try to get upwind of their opponents. If you're upwind, you decide when and if to engage the other ship. Craigslist is effectively upwind of enormous revenues. They'd face some challenges if they wanted to make more, but not the sort you face when you're tacking upwind, trying to force a crappy product on ambivalent users by spending ten times as much on sales as on development. [\[1\]](#)

I'm not saying startups should aim to end up like Craigslist. They're a product of unusual circumstances. But they're a good model for the early phases.

Google looked a lot like a charity in the beginning. They didn't have ads for over a year. At year 1, Google was indistinguishable from a nonprofit. If a nonprofit or government organization had started a project to index the web, Google at year 1 is the limit of what they'd have produced.

Back when I was working on spam filters I thought it would be a good idea to have a web-based email service with good spam filtering. I wasn't thinking of it as a company. I just wanted to keep people from getting spammed. But as I thought more about this project, I realized it would probably have to be a company. It would cost something to run, and it would be a pain to fund with grants and donations.

That was a surprising realization. Companies often claim to be benevolent, but it was surprising to realize there were purely benevolent projects that had to be embodied as companies to work.

I didn't want to start another company, so I didn't do it. But if someone had, they'd probably be quite rich now. There was a window of about two years when spam was increasing rapidly but all the big email services had terrible filters. If someone had launched a new, spam-free mail service, users would have flocked to it.

Notice the pattern here? From either direction we get to the same spot. If you start from successful startups, you find they often behaved like nonprofits. And if you start from ideas for nonprofits, you find they'd often make good startups.

## **Power**

How wide is this territory? Would all good nonprofits be good companies? Possibly not. What makes Google so valuable is that their users have money. If you make people with money love you, you can probably get some of it. But could you also base a successful startup on behaving like a nonprofit to people who don't have money? Could you, for example, grow a successful startup out of curing an unfashionable but deadly disease like malaria?



I'm not sure, but I suspect that if you pushed this idea, you'd be surprised how far it would go. For example, people who apply to Y Combinator don't generally have much money, and yet we can profit by helping them, because with our help they could make money. Maybe the situation is similar with malaria. Maybe an organization that helped lift its weight off a country could benefit from the resulting growth.

I'm not proposing this is a serious idea. I don't know anything about malaria. But I've been kicking ideas around long enough to know when I come across a powerful one.

One way to guess how far an idea extends is to ask yourself at what point you'd bet against it. The thought of betting against benevolence is alarming in the same way as saying that something is technically impossible. You're just asking to be made a fool of, because these are such powerful forces. [2]

For example, initially I thought maybe this principle only applied to Internet startups. Obviously it worked for Google, but what about Microsoft? Surely Microsoft isn't benevolent? But when I think back to the beginning, they were. Compared to IBM they were like Robin Hood. When IBM introduced the PC, they thought they were going to make money selling hardware at high prices. But by gaining control of the PC standard, Microsoft opened up the market to any manufacturer. Hardware prices plummeted, and lots of people got to have computers who couldn't otherwise have afforded them. It's the sort of thing you'd expect Google to do.

Microsoft isn't so benevolent now. Now when one thinks of what Microsoft does to users, all the verbs that come to mind begin with F. [3] And yet it doesn't seem to pay. Their stock price has been flat for years. Back when they were Robin Hood, their stock price rose like Google's. Could there be a connection?

You can see how there would be. When you're small, you can't bully customers, so you have to charm them. Whereas when you're big you can maltreat them at will, and you tend to, because it's easier than satisfying them. You grow big by being nice, but you can stay big by being mean.

You get away with it till the underlying conditions change, and then all your victims escape. So "Don't be evil" may be the most valuable thing Paul Buchheit made for Google, because it may turn out to be an elixir of corporate youth. I'm sure they find it constraining, but think how valuable it will be if it saves them from lapsing into the fatal laziness that afflicted Microsoft and IBM.

The curious thing is, this elixir is freely available to any other company. Anyone can adopt "Don't be evil." The catch is that people will hold you to it. So I don't think you're going to see record labels or tobacco companies using this discovery.

## **Morale**

There's a lot of external evidence that benevolence works. But how does it work? One advantage of investing in a large number of startups is that you get a lot of data about how they work. From what we've seen, being good seems to help startups in three ways: it improves their morale, it makes other people want to help them, and above all, it helps them be decisive.

Morale is tremendously important to a startup—so important that morale alone is almost enough to determine success. Startups are often described as emotional roller-coasters. One minute you're going to take over the world, and the next you're doomed. The problem with feeling you're doomed is not just that it makes you unhappy, but that it makes you *stop working*. So the downhills of the roller-coaster are more of a self fulfilling prophecy than the uphill. If feeling you're going to succeed makes you work harder, that probably improves your chances of succeeding, but if feeling you're going to fail makes you stop working, that practically guarantees you'll fail.

Here's where benevolence comes in. If you feel you're really helping people, you'll keep working even when it seems like your startup is doomed. Most of us have some amount of natural benevolence. The mere fact that someone needs you makes you want to help them. So if you start the kind of startup where users come back each day, you've basically built yourself a giant tamagotchi. You've made something you need to take care of.

Blogger is a famous example of a startup that went through really low lows and survived. At one point they ran out of money and everyone left. Evan Williams came in to work the next day, and there was no one but him. What kept him going? Partly that users needed him. He was hosting thousands of people's blogs. He couldn't just let the site die.

There are many advantages of launching quickly, but the most important may be that once you have users, the tamagotchi effect kicks in. Once you have users to take care of, you're forced to figure out what will make them happy, and that's actually very valuable information.

The added confidence that comes from trying to help people can also help you with investors. One of the founders of [Chatterous](#) told me recently that he and his cofounder had decided that this service was something the world needed, so they were going to keep working on it no matter what, even if they had to move back to Canada and live in their parents' basements.

Once they realized this, they stopped caring so much what investors thought about them. They still met with them, but they weren't going to die if they didn't get their money. And you know what? The investors got a lot more interested. They could sense that the Chatterouses were going to do this startup with or without them.

If you're really committed and your startup is cheap to run, you become very hard to kill. And practically all startups, even the most successful, come close to death

at some point. So if doing good for people gives you a sense of mission that makes you harder to kill, that alone more than compensates for whatever you lose by not choosing a more selfish project.

## Help

Another advantage of being good is that it makes other people want to help you. This too seems to be an inborn trait in humans.

One of the startups we've funded, [Octopart](#), is currently locked in a classic battle of good versus evil. They're a search site for industrial components. A lot of people need to search for components, and before Octopart there was no good way to do it. That, it turned out, was no coincidence.

Octopart built the right way to search for components. Users like it and they've been growing rapidly. And yet for most of Octopart's life, the biggest distributor, Digi-Key, has been trying to force them take their prices off the site. Octopart is sending them customers for free, and yet Digi-Key is trying to make that traffic stop. Why? Because their current business model depends on overcharging people who have incomplete information about prices. They don't want search to work.

The Octoparts are the nicest guys in the world. They dropped out of the PhD program in physics at Berkeley to do this. They just wanted to fix a problem they encountered in their research. Imagine how much time you could save the world's engineers if they could do searches online. So when I hear that a big, evil company is trying to stop them in order to keep search broken, it makes me really want to help them. It makes me spend more time on the Octoparts than I do with most of the other startups we've funded. It just made me spend several minutes telling you how great they are. Why? Because they're good guys and they're trying to help the world.

If you're benevolent, people will rally around you: investors, customers, other companies, and potential employees. In the long term the most important may be the potential employees. I think everyone knows now that [good hackers](#) are much better than mediocre ones. If you can attract the best hackers to work for you, as Google has, you have a big advantage. And the very best hackers tend to be idealistic. They're not desperate for a job. They can work wherever they want. So most want to work on things that will make the world better.

## Compass

But the most important advantage of being good is that it acts as a compass. One of the hardest parts of doing a startup is that you have so many choices. There are just two or three of you, and a thousand things you could do. How do you decide?

Here's the answer: Do whatever's best for your users. You can hold onto this like a rope in a hurricane, and it will save you if anything can. Follow it and it will take you through everything you need to do.

It's even the answer to questions that seem unrelated, like how to convince investors to give you money. If you're a good salesman, you could try to just talk them into it. But the more reliable route is to convince them through your users: if you make something users love enough to tell their friends, you grow exponentially, and that will convince any investor.

Being good is a particularly useful strategy for making decisions in complex situations because it's stateless. It's like telling the truth. The trouble with lying is that you have to remember everything you've said in the past to make sure you don't contradict yourself. If you tell the truth you don't have to remember anything, and that's a really useful property in domains where things happen fast.

For example, Y Combinator has now invested in 80 startups, 57 of which are still alive. (The rest have died or merged or been acquired.) When you're trying to advise 57 startups, it turns out you have to have a stateless algorithm. You can't have ulterior motives when you have 57 things going on at once, because you can't remember them. So our rule is just to do whatever's best for the founders. Not because we're particularly benevolent, but because it's the only algorithm that works on that scale.

When you write something telling people to be good, you seem to be claiming to be good yourself. So I want to say explicitly that I am not a particularly good person. When I was a kid I was firmly in the camp of bad. The way adults used the word good, it seemed to be synonymous with quiet, so I grew up very suspicious of it.

You know how there are some people whose names come up in conversation and everyone says "He's *such* a great guy?" People never say that about me. The best I get is "he means well." I am not claiming to be good. At best I speak good as a second language.

So I'm not suggesting you be good in the usual sanctimonious way. I'm suggesting it because it works. It will work not just as a statement of "values," but as a guide to strategy, and even a design spec for software. Don't just not be evil. Be good.

## Notes

[1] Fifty years ago it would have seemed shocking for a public company not to pay dividends. Now many tech companies don't. The markets seem to have figured out how to value potential dividends. Maybe that isn't the last step in this evolution. Maybe markets will eventually get comfortable with potential earnings. (VCs already are, and at least some of them consistently make money.)

I realize this sounds like the stuff one used to hear about the "new economy" during the Bubble. Believe me, I was not drinking that kool-aid at the time. But I'm convinced there were some [good ideas](#) buried in Bubble thinking. For example, it's ok to focus on growth instead of profits—but only if the growth is genuine. You can't be buying users; that's a pyramid scheme. But a company with rapid, genuine growth is valuable, and eventually markets learn how to value valuable things.

[2] The idea of starting a company with benevolent aims is currently undervalued, because the kind of people who currently make that their explicit goal don't usually do a very good job.

It's one of the standard career paths of trustafarians to start some vaguely benevolent business. The problem with most of them is that they either have a bogus political agenda or are feebly executed. The trustafarians' ancestors didn't get rich by preserving their traditional culture; maybe people in Bolivia don't want to either. And starting an organic farm, though it's at least straightforwardly benevolent, doesn't help people on the scale that Google does.

Most explicitly benevolent projects don't hold themselves sufficiently accountable. They act as if having good intentions were enough to guarantee good effects.

[3] Users dislike their new operating system so much that they're starting petitions to save the old one. And the old one was nothing special. The hackers within Microsoft must know in their hearts that if the company really cared about users they'd just advise them to switch to OSX.

**Thanks** to Trevor Blackwell, Paul Buchheit, Jessica Livingston, and Robert Morris for reading drafts of this.

[Russian Translation](#)

[German Translation](#)

# Lies We Tell Kids

May 2008

Adults lie constantly to kids. I'm not saying we should stop, but I think we should at least examine which lies we tell and why.

There may also be a benefit to us. We were all lied to as kids, and some of the lies we were told still affect us. So by studying the ways adults lie to kids, we may be able to clear our heads of lies we were told.

I'm using the word "lie" in a very general sense: not just overt falsehoods, but also all the more subtle ways we mislead kids. Though "lie" has negative connotations, I don't mean to suggest we should never do this—just that we should pay attention when we do. [\[1\]](#)

One of the most remarkable things about the way we lie to kids is how broad the conspiracy is. All adults know what their culture lies to kids about: they're the questions you answer "Ask your parents." If a kid asked who won the World Series in 1982 or what the atomic weight of carbon was, you could just tell him. But if a kid asks you "Is there a God?" or "What's a prostitute?" you'll probably say "Ask your parents."

Since we all agree, kids see few cracks in the view of the world presented to them. The biggest disagreements are between parents and schools, but even those are small. Schools are careful what they say about controversial topics, and if they do contradict what parents want their kids to believe, parents either pressure the school into keeping [quiet](#) or move their kids to a new school.

The conspiracy is so thorough that most kids who discover it do so only by discovering internal contradictions in what they're told. It can be traumatic for the ones who wake up during the operation. Here's what happened to Einstein:

Through the reading of popular scientific books I soon reached the conviction that much in the stories of the Bible could not be true. The consequence was a positively fanatic freethinking coupled with the impression that youth is intentionally being deceived by the state through lies: it was a crushing impression. [\[2\]](#)

I remember that feeling. By 15 I was convinced the world was corrupt from end to end. That's why movies like *The Matrix* have such resonance. Every kid grows up in

a fake world. In a way it would be easier if the forces behind it were as clearly differentiated as a bunch of evil machines, and one could make a clean break just by taking a pill.

## **Protection**

If you ask adults why they lie to kids, the most common reason they give is to protect them. And kids do need protecting. The environment you want to create for a newborn child will be quite unlike the streets of a big city.

That seems so obvious it seems wrong to call it a lie. It's certainly not a bad lie to tell, to give a baby the impression the world is quiet and warm and safe. But this harmless type of lie can turn sour if left unexamined.

Imagine if you tried to keep someone in as protected an environment as a newborn till age 18. To mislead someone so grossly about the world would seem not protection but abuse. That's an extreme example, of course; when parents do that sort of thing it becomes national news. But you see the same problem on a smaller scale in the malaise teenagers feel in suburbia.

The main purpose of suburbia is to provide a protected environment for children to grow up in. And it seems great for 10 year olds. I liked living in suburbia when I was 10. I didn't notice how sterile it was. My whole world was no bigger than a few friends' houses I bicycled to and some woods I ran around in. On a log scale I was midway between crib and globe. A suburban street was just the right size. But as I grew older, suburbia started to feel suffocatingly fake.

Life can be pretty good at 10 or 20, but it's often frustrating at 15. This is too big a problem to solve here, but certainly one reason life sucks at 15 is that kids are trapped in a world designed for 10 year olds.

What do parents hope to protect their children from by raising them in suburbia? A friend who moved out of Manhattan said merely that her 3 year old daughter "saw too much." Off the top of my head, that might include: people who are high or drunk, poverty, madness, gruesome medical conditions, sexual behavior of various degrees of oddness, and violent anger.

I think it's the anger that would worry me most if I had a 3 year old. I was 29 when I moved to New York and I was surprised even then. I wouldn't want a 3 year old to see some of the disputes I saw. It would be too frightening. A lot of the things adults conceal from smaller children, they conceal because they'd be frightening, not because they want to conceal the existence of such things. Misleading the child is just a byproduct.

This seems one of the most justifiable types of lying adults do to kids. But because the lies are indirect we don't keep a very strict accounting of them. Parents know they've concealed the facts about sex, and many at some point sit their kids down and explain more. But few tell their kids about the differences between the real

world and the cocoon they grew up in. Combine this with the confidence parents try to instill in their kids, and every year you get a new crop of 18 year olds who think they know how to run the world.

Don't all 18 year olds think they know how to run the world? Actually this seems to be a recent innovation, no more than about 100 years old. In preindustrial times teenage kids were junior members of the adult world and comparatively well aware of their shortcomings. They could see they weren't as strong or skillful as the village smith. In past times people lied to kids about some things more than we do now, but the lies implicit in an artificial, protected environment are a recent invention. Like a lot of new inventions, the rich got this first. Children of kings and great magnates were the first to grow up out of touch with the world. Suburbia means half the population can live like kings in that respect.

## **Sex (and Drugs)**

I'd have different worries about raising teenage kids in New York. I'd worry less about what they'd see, and more about what they'd do. I went to college with a lot of kids who grew up in Manhattan, and as a rule they seemed pretty jaded. They seemed to have lost their virginity at an average of about 14 and by college had tried more drugs than I'd even heard of.

The reasons parents don't want their teenage kids having sex are complex. There are some obvious dangers: pregnancy and sexually transmitted diseases. But those aren't the only reasons parents don't want their kids having sex. The average parents of a 14 year old girl would hate the idea of her having sex even if there were zero risk of pregnancy or sexually transmitted diseases.

Kids can probably sense they aren't being told the whole story. After all, pregnancy and sexually transmitted diseases are just as much a problem for adults, and they have sex.

What really bothers parents about their teenage kids having sex? Their dislike of the idea is so visceral it's probably inborn. But if it's inborn it should be universal, and there are plenty of societies where parents don't mind if their teenage kids have sex—indeed, where it's normal for 14 year olds to become mothers. So what's going on? There does seem to be a universal taboo against sex with prepubescent children. One can imagine evolutionary reasons for that. And I think this is the main reason parents in industrialized societies dislike teenage kids having sex. They still think of them as children, even though biologically they're not, so the taboo against child sex still has force.

One thing adults conceal about sex they also conceal about drugs: that it can cause great pleasure. That's what makes sex and drugs so dangerous. The desire for them can cloud one's judgement—which is especially frightening when the judgement being clouded is the already wretched judgement of a teenage kid.

Here parents' desires conflict. Older societies told kids they had bad judgement,



but modern parents want their children to be confident. This may well be a better plan than the old one of putting them in their place, but it has the side effect that after having implicitly lied to kids about how good their judgement is, we then have to lie again about all the things they might get into trouble with if they believed us.

If parents told their kids the truth about sex and drugs, it would be: the reason you should avoid these things is that you have lousy judgement. People with twice your experience still get burned by them. But this may be one of those cases where the truth wouldn't be convincing, because one of the symptoms of bad judgement is believing you have good judgement. When you're too weak to lift something, you can tell, but when you're making a decision impetuously, you're all the more sure of it.

## **Innocence**

Another reason parents don't want their kids having sex is that they want to keep them innocent. Adults have a certain model of how kids are supposed to behave, and it's different from what they expect of other adults.

One of the most obvious differences is the words kids are allowed to use. Most parents use words when talking to other adults that they wouldn't want their kids using. They try to hide even the existence of these words for as long as they can. And this is another of those conspiracies everyone participates in: everyone knows you're not supposed to swear in front of kids.

I've never heard more different explanations for anything parents tell kids than why they shouldn't swear. Every parent I know forbids their children to swear, and yet no two of them have the same justification. It's clear most start with not wanting kids to swear, then make up the reason afterward.

So my theory about what's going on is that the *function* of swearwords is to mark the speaker as an adult. There's no difference in the meaning of "shit" and "poopoo." So why should one be ok for kids to say and one forbidden? The only explanation is: by definition. [\[3\]](#)

Why does it bother adults so much when kids do things reserved for adults? The idea of a foul-mouthed, cynical 10 year old leaning against a lamppost with a cigarette hanging out of the corner of his mouth is very disconcerting. But why?

One reason we want kids to be innocent is that we're programmed to like certain kinds of helplessness. I've several times heard mothers say they deliberately refrained from correcting their young children's mispronunciations because they were so cute. And if you think about it, cuteness is helplessness. Toys and cartoon characters meant to be cute always have clueless expressions and stubby, ineffectual limbs.

It's not surprising we'd have an inborn desire to love and protect helpless

creatures, considering human offspring are so helpless for so long. Without the helplessness that makes kids cute, they'd be very annoying. They'd merely seem like incompetent adults. But there's more to it than that. The reason our hypothetical jaded 10 year old bothers me so much is not just that he'd be annoying, but that he'd have cut off his prospects for growth so early. To be jaded you have to think you know how the world works, and any theory a 10 year old had about that would probably be a pretty narrow one.

Innocence is also open-mindedness. We want kids to be innocent so they can continue to learn. Paradoxical as it sounds, there are some kinds of knowledge that get in the way of other kinds of knowledge. If you're going to learn that the world is a brutal place full of people trying to take advantage of one another, you're better off learning it last. Otherwise you won't bother learning much more.

Very smart adults often seem unusually innocent, and I don't think this is a coincidence. I think they've deliberately avoided learning about certain things. Certainly I do. I used to think I wanted to know everything. Now I know I don't.

## **Death**

After sex, death is the topic adults lie most conspicuously about to kids. Sex I believe they conceal because of deep taboos. But why do we conceal death from kids? Probably because small children are particularly horrified by it. They want to feel safe, and death is the ultimate threat.

One of the most spectacular lies our parents told us was about the death of our first cat. Over the years, as we asked for more details, they were compelled to invent more, so the story grew quite elaborate. The cat had died at the vet's office. Of what? Of the anaesthesia itself. Why was the cat at the vet's office? To be fixed. And why had such a routine operation killed it? It wasn't the vet's fault; the cat had a congenitally weak heart; the anaesthesia was too much for it; but there was no way anyone could have known this in advance. It was not till we were in our twenties that the truth came out: my sister, then about three, had accidentally stepped on the cat and broken its back.

They didn't feel the need to tell us the cat was now happily in cat heaven. My parents never claimed that people or animals who died had "gone to a better place," or that we'd meet them again. It didn't seem to harm us.

My grandmother told us an edited version of the death of my grandfather. She said they'd been sitting reading one day, and when she said something to him, he didn't answer. He seemed to be asleep, but when she tried to rouse him, she couldn't. "He was gone." Having a heart attack sounded like falling asleep. Later I learned it hadn't been so neat, and the heart attack had taken most of a day to kill him.

Along with such outright lies, there must have been a lot of changing the subject when death came up. I can't remember that, of course, but I can infer it from the fact that I didn't really grasp I was going to die till I was about 19. How could I

have missed something so obvious for so long? Now that I've seen parents managing the subject, I can see how: questions about death are gently but firmly turned aside.

On this topic, especially, they're met half-way by kids. Kids often want to be lied to. They want to believe they're living in a comfortable, safe world as much as their parents want them to believe it. [\[4\]](#)

## **Identity**

Some parents feel a strong adherence to an ethnic or religious group and want their kids to feel it too. This usually requires two different kinds of lying: the first is to tell the child that he or she is an X, and the second is whatever specific lies Xes differentiate themselves by believing. [\[5\]](#)

Telling a child they have a particular ethnic or religious identity is one of the stickiest things you can tell them. Almost anything else you tell a kid, they can change their mind about later when they start to think for themselves. But if you tell a kid they're a member of a certain group, that seems nearly impossible to shake.

This despite the fact that it can be one of the most premeditated lies parents tell. When parents are of different religions, they'll often agree between themselves that their children will be "raised as Xes." And it works. The kids obligingly grow up considering themselves as Xes, despite the fact that if their parents had chosen the other way, they'd have grown up considering themselves as Ys.

One reason this works so well is the second kind of lie involved. The truth is common property. You can't distinguish your group by doing things that are rational, and believing things that are true. If you want to set yourself apart from other people, you have to do things that are arbitrary, and believe things that are false. And after having spent their whole lives doing things that are arbitrary and believing things that are false, and being regarded as odd by "outsiders" on that account, the cognitive dissonance pushing children to regard themselves as Xes must be enormous. If they aren't an X, why are they attached to all these arbitrary beliefs and customs? If they aren't an X, why do all the non-Xes call them one?

This form of lie is not without its uses. You can use it to carry a payload of beneficial beliefs, and they will also become part of the child's identity. You can tell the child that in addition to never wearing the color yellow, believing the world was created by a giant rabbit, and always snapping their fingers before eating fish, Xes are also particularly honest and industrious. Then X children will grow up feeling it's part of their identity to be honest and industrious.

This probably accounts for a lot of the spread of modern religions, and explains why their doctrines are a combination of the useful and the bizarre. The bizarre half is what makes the religion stick, and the useful half is the payload. [\[6\]](#)

## Authority

One of the least excusable reasons adults lie to kids is to maintain power over them. Sometimes these lies are truly sinister, like a child molester telling his victims they'll get in trouble if they tell anyone what happened to them. Others seem more innocent; it depends how badly adults lie to maintain their power, and what they use it for.

Most adults make some effort to conceal their flaws from children. Usually their motives are mixed. For example, a father who has an affair generally conceals it from his children. His motive is partly that it would worry them, partly that this would introduce the topic of sex, and partly (a larger part than he would admit) that he doesn't want to tarnish himself in their eyes.

If you want to learn what lies are told to kids, read almost any book written to teach them about "issues." [7] Peter Mayle wrote one called *Why Are We Getting a Divorce?* It begins with the three most important things to remember about divorce, one of which is:

You shouldn't put the blame on one parent, because divorce is never only one person's fault. [8]

Really? When a man runs off with his secretary, is it always partly his wife's fault? But I can see why Mayle might have said this. Maybe it's more important for kids to respect their parents than to know the truth about them.

But because adults conceal their flaws, and at the same time insist on high standards of behavior for kids, a lot of kids grow up feeling they fall hopelessly short. They walk around feeling horribly evil for having used a swearword, while in fact most of the adults around them are doing much worse things.

This happens in intellectual as well as moral questions. The more confident people are, the more willing they seem to be to answer a question "I don't know." Less confident people feel they have to have an answer or they'll look bad. My parents were pretty good about admitting when they didn't know things, but I must have been told a lot of lies of this type by teachers, because I rarely heard a teacher say "I don't know" till I got to college. I remember because it was so surprising to hear someone say that in front of a class.

The first hint I had that teachers weren't omniscient came in sixth grade, after my father contradicted something I'd learned in school. When I protested that the teacher had said the opposite, my father replied that the guy had no idea what he was talking about—that he was just an elementary school teacher, after all.

*Just* a teacher? The phrase seemed almost grammatically ill-formed. Didn't teachers know everything about the subjects they taught? And if not, why were they the ones teaching us?

The sad fact is, US public school teachers don't generally understand the stuff they're teaching very well. There are some sterling exceptions, but as a rule people planning to go into teaching rank academically near the bottom of the college population. So the fact that I still thought at age 11 that teachers were infallible shows what a job the system must have done on my brain.

## **School**

What kids get taught in school is a complex mix of lies. The most excusable are those told to simplify ideas to make them easy to learn. The problem is, a lot of propaganda gets slipped into the curriculum in the name of simplification.

Public school textbooks represent a compromise between what various powerful groups want kids to be told. The lies are rarely overt. Usually they consist either of omissions or of over-emphasizing certain topics at the expense of others. The view of history we got in elementary school was a crude hagiography, with at least one representative of each powerful group.

The famous scientists I remember were Einstein, Marie Curie, and George Washington Carver. Einstein was a big deal because his work led to the atom bomb. Marie Curie was involved with X-rays. But I was mystified about Carver. He seemed to have done stuff with peanuts.

It's obvious now that he was on the list because he was black (and for that matter that Marie Curie was on it because she was a woman), but as a kid I was confused for years about him. I wonder if it wouldn't have been better just to tell us the truth: that there weren't any famous black scientists. Ranking George Washington Carver with Einstein misled us not only about science, but about the obstacles blacks faced in his time.

As subjects got softer, the lies got more frequent. By the time you got to politics and recent history, what we were taught was pretty much pure propaganda. For example, we were taught to regard political leaders as saints—especially the recently martyred Kennedy and King. It was astonishing to learn later that they'd both been serial womanizers, and that Kennedy was a speed freak to boot. (By the time King's plagiarism emerged, I'd lost the ability to be surprised by the misdeeds of famous people.)

I doubt you could teach kids recent history without teaching them lies, because practically everyone who has anything to say about it has some kind of spin to put on it. Much recent history *consists* of spin. It would probably be better just to teach them metafacts like that.

Probably the biggest lie told in schools, though, is that the way to succeed is through following "the rules." In fact most such rules are just hacks to manage large groups efficiently.

## **Peace**

Of all the reasons we lie to kids, the most powerful is probably the same mundane reason they lie to us.

Often when we lie to people it's not part of any conscious strategy, but because they'd react violently to the truth. Kids, almost by definition, lack self-control. They react violently to things—and so they get lied to a lot. [9]

A few Thanksgivings ago, a friend of mine found himself in a situation that perfectly illustrates the complex motives we have when we lie to kids. As the roast turkey appeared on the table, his alarmingly perceptive 5 year old son suddenly asked if the turkey had wanted to die. Foreseeing disaster, my friend and his wife rapidly improvised: yes, the turkey had wanted to die, and in fact had lived its whole life with the aim of being their Thanksgiving dinner. And that (phew) was the end of that.

Whenever we lie to kids to protect them, we're usually also lying to keep the peace.

One consequence of this sort of calming lie is that we grow up thinking horrible things are normal. It's hard for us to feel a sense of urgency as adults over something we've literally been trained not to worry about. When I was about 10 I saw a documentary on pollution that put me into a panic. It seemed the planet was being irretrievably ruined. I went to my mother afterward to ask if this was so. I don't remember what she said, but she made me feel better, so I stopped worrying about it.

That was probably the best way to handle a frightened 10 year old. But we should understand the price. This sort of lie is one of the main reasons bad things persist: we're all trained to ignore them.

## **Detox**

A sprinter in a race almost immediately enters a state called "oxygen debt." His body switches to an emergency source of energy that's faster than regular aerobic respiration. But this process builds up waste products that ultimately require extra oxygen to break down, so at the end of the race he has to stop and pant for a while to recover.

We arrive at adulthood with a kind of truth debt. We were told a lot of lies to get us (and our parents) through our childhood. Some may have been necessary. Some probably weren't. But we all arrive at adulthood with heads full of lies.

There's never a point where the adults sit you down and explain all the lies they told you. They've forgotten most of them. So if you're going to clear these lies out of your head, you're going to have to do it yourself.

Few do. Most people go through life with bits of packing material adhering to their

minds and never know it. You probably never can completely undo the effects of lies you were told as a kid, but it's worth trying. I've found that whenever I've been able to undo a lie I was told, a lot of other things fell into place.

Fortunately, once you arrive at adulthood you get a valuable new resource you can use to figure out what lies you were told. You're now one of the liars. You get to watch behind the scenes as adults spin the world for the next generation of kids.

The first step in clearing your head is to realize how far you are from a neutral observer. When I left high school I was, I thought, a complete skeptic. I'd realized high school was crap. I thought I was ready to question everything I knew. But among the many other things I was ignorant of was how much debris there already was in my head. It's not enough to consider your mind a blank slate. You have to consciously erase it.

## Notes

[1] One reason I stuck with such a brutally simple word is that the lies we tell kids are probably not quite as harmless as we think. If you look at what adults told children in the past, it's shocking how much they lied to them. Like us, they did it with the best intentions. So if we think we're as open as one could reasonably be with children, we're probably fooling ourselves. Odds are people in 100 years will be as shocked at some of the lies we tell as we are at some of the lies people told 100 years ago.

I can't predict which these will be, and I don't want to write an essay that will seem dumb in 100 years. So instead of using special euphemisms for lies that seem excusable according to present fashions, I'm just going to call all our lies lies.

(I have omitted one type: lies told to play games with kids' credulity. These range from "make-believe," which is not really a lie because it's told with a wink, to the frightening lies told by older siblings. There's not much to say about these: I wouldn't want the first type to go away, and wouldn't expect the second type to.)

[2] Calaprice, Alice (ed.), *The Quotable Einstein*, Princeton University Press, 1996.

[3] If you ask parents why kids shouldn't swear, the less educated ones usually reply with some question-begging answer like "it's inappropriate," while the more educated ones come up with elaborate rationalizations. In fact the less educated parents seem closer to the truth.

[4] As a friend with small children pointed out, it's easy for small children to consider themselves immortal, because time seems to pass so slowly for them. To

a 3 year old, a day feels like a month might to an adult. So 80 years sounds to him like 2400 years would to us.

[5] I realize I'm going to get endless grief for classifying religion as a type of lie. Usually people skirt that issue with some equivocation implying that lies believed for a sufficiently long time by sufficiently large numbers of people are immune to the usual standards for truth. But because I can't predict which lies future generations will consider inexcusable, I can't safely omit any type we tell. Yes, it seems unlikely that religion will be out of fashion in 100 years, but no more unlikely than it would have seemed to someone in 1880 that schoolchildren in 1980 would be taught that masturbation was perfectly normal and not to feel guilty about it.

[6] Unfortunately the payload can consist of bad customs as well as good ones. For example, there are certain qualities that some groups in America consider "acting white." In fact most of them could as accurately be called "acting Japanese." There's nothing specifically white about such customs. They're common to all cultures with long traditions of living in cities. So it is probably a losing bet for a group to consider behaving the opposite way as part of its identity.

[7] In this context, "issues" basically means "things we're going to lie to them about." That's why there's a special name for these topics.

[8] Mayle, Peter, *Why Are We Getting a Divorce?*, Harmony, 1988.

[9] The ironic thing is, this is also the main reason kids lie to adults. If you freak out when people tell you alarming things, they won't tell you them. Teenagers don't tell their parents what happened that night they were supposed to be staying at a friend's house for the same reason parents don't tell 5 year olds the truth about the Thanksgiving turkey. They'd freak if they knew.

**Thanks** to Sam Altman, Marc Andreessen, Trevor Blackwell, Patrick Collison, Jessica Livingston, Jackie McDonough, Robert Morris, and David Sloo for reading drafts of this. And since there are some controversial ideas here, I should add that none of them agreed with everything in it.

[German Translation](#)

[French Translation](#)

[Russian Translation](#)



# Disconnecting Distraction

*Note: The strategy described at the end of this essay didn't work. It would work for a while, and then I'd gradually find myself using the Internet on my work computer. I'm trying other strategies now, but I think this time I'll wait till I'm sure they work before writing about them.*

May 2008

Procrastination feeds on distractions. Most people find it uncomfortable just to sit and do nothing; you avoid work by doing something else.

So one way to beat procrastination is to starve it of distractions. But that's not as straightforward as it sounds, because there are people working hard to distract you. Distraction is not a static obstacle that you avoid like you might avoid a rock in the road. Distraction seeks you out.

Chesterfield described dirt as matter out of place. Distracting is, similarly, desirable at the wrong time. And technology is continually being refined to produce more and more desirable things. Which means that as we learn to avoid one class of distractions, new ones constantly appear, like drug-resistant bacteria.

Television, for example, has after 50 years of refinement reached the point where it's like visual crack. I realized when I was 13 that TV was addictive, so I stopped watching it. But I read recently that the average American watches [4 hours](#) of TV a day. A quarter of their life.

TV is in decline now, but only because people have found even more addictive ways of wasting time. And what's especially dangerous is that many happen at your computer. This is no accident. An ever larger percentage of office workers sit in front of computers connected to the Internet, and distractions always evolve toward the procrastinators.

I remember when computers were, for me at least, exclusively for work. I might occasionally dial up a server to get mail or ftp files, but most of the time I was offline. All I could do was write and program. Now I feel as if someone snuck a television onto my desk. Terribly addictive things are just a click away. Run into an obstacle in what you're working on? Hmm, I wonder what's new online. Better check.

After years of carefully avoiding classic time sinks like TV, games, and Usenet, I still managed to fall prey to distraction, because I didn't realize that it evolves. Something that used to be safe, using the Internet, gradually became more and more dangerous. Some days I'd wake up, get a cup of tea and check the news, then check email, then check the news again, then answer a few emails, then suddenly notice it was almost lunchtime and I hadn't gotten any real work done. And this started to happen more and more often.

It took me surprisingly long to realize how distracting the Internet had become, because the problem was intermittent. I ignored it the way you let yourself ignore a bug that only appears intermittently. When I was in the middle of a project, distractions weren't really a problem. It was when I'd finished one project and was deciding what to do next that they always bit me.

Another reason it was hard to notice the danger of this new type of distraction was that social customs hadn't yet caught up with it. If I'd spent a whole morning sitting on a sofa watching TV, I'd have noticed very quickly. That's a known danger sign, like drinking alone. But using the Internet still looked and felt a lot like work.

Eventually, though, it became clear that the Internet had become so much more distracting that I had to start treating it differently. Basically, I had to add a new application to my list of known time sinks: Firefox.

\* \* \*

The problem is a hard one to solve because most people still need the Internet for some things. If you drink too much, you can solve that problem by stopping entirely. But you can't solve the problem of overeating by stopping eating. I couldn't simply avoid the Internet entirely, as I'd done with previous time sinks.

At first I tried rules. For example, I'd tell myself I was only going to use the Internet twice a day. But these schemes never worked for long. Eventually something would come up that required me to use it more than that. And then I'd gradually slip back into my old ways.

Addictive things have to be treated as if they were sentient adversaries—as if there were a little man in your head always cooking up the most plausible arguments for doing whatever you're trying to stop doing. If you leave a path to it, he'll find it.

The key seems to be visibility. The biggest ingredient in most bad habits is denial. So you have to make it so that you can't merely *slip* into doing the thing you're trying to avoid. It has to set off alarms.

Maybe in the long term the right answer for dealing with Internet distractions will be [software](#) that watches and controls them. But in the meantime I've found a more drastic solution that definitely works: to set up a separate computer for using the Internet.

I now leave wifi turned off on my main computer except when I need to transfer a file or edit a web page, and I have a separate laptop on the other side of the room that I use to check mail or browse the web. (Irony of ironies, it's the computer Steve Huffman wrote Reddit on. When Steve and Alexis auctioned off their old laptops for charity, I bought them for the Y Combinator museum.)

My rule is that I can spend as much time online as I want, as long as I do it on that computer. And this turns out to be enough. When I have to sit on the other side of the room to check email or browse the web, I become much more aware of it. Sufficiently aware, in my case at least, that it's hard to spend more than about an hour a day online.

And my main computer is now freed for work. If you try this trick, you'll probably be struck by how different it feels when your computer is disconnected from the Internet. It was alarming to me how foreign it felt to sit in front of a computer that could only be used for work, because that showed how much time I must have been wasting.

*Wow. All I can do at this computer is work. Ok, I better work then.*

That's the good part. Your old bad habits now help you to work. You're used to sitting in front of that computer for hours at a time. But you can't browse the web or check email now. What are you going to do? You can't just sit there. So you start working.

[Good and Bad Procrastination](#)

[Spanish Translation](#)

[Arabic Translation](#)

[Catalan Translation](#)

[Russian Translation](#)

[Spanish Translation](#)

# Cities and Ambition

May 2008

Great cities attract ambitious people. You can sense it when you walk around one. In a hundred subtle ways, the city sends you a message: you could do more; you should try harder.

The surprising thing is how different these messages can be. New York tells you, above all: you should make more money. There are other messages too, of course. You should be hipper. You should be better looking. But the clearest message is that you should be richer.

What I like about Boston (or rather Cambridge) is that the message there is: you should be smarter. You really should get around to reading all those books you've been meaning to.

When you ask what message a city sends, you sometimes get surprising answers. As much as they respect brains in Silicon Valley, the message the Valley sends is: you should be more powerful.

That's not quite the same message New York sends. Power matters in New York too of course, but New York is pretty impressed by a billion dollars even if you merely inherited it. In Silicon Valley no one would care except a few real estate agents. What matters in Silicon Valley is how much effect you have on the world. The reason people there care about Larry and Sergey is not their wealth but the fact that they control Google, which affects practically everyone.

---

How much does it matter what message a city sends? Empirically, the answer seems to be: a lot. You might think that if you had enough strength of mind to do great things, you'd be able to transcend your environment. Where you live should make at most a couple percent difference. But if you look at the historical evidence, it seems to matter more than that. Most people who did great things were clumped together in a few places where that sort of thing was done at the time.

You can see how powerful cities are from something I wrote about [earlier](#): the case

of the Milanese Leonardo. Practically every fifteenth century Italian painter you've heard of was from Florence, even though Milan was just as big. People in Florence weren't genetically different, so you have to assume there was someone born in Milan with as much natural ability as Leonardo. What happened to him?

If even someone with the same natural ability as Leonardo couldn't beat the force of environment, do you suppose you can?

I don't. I'm fairly stubborn, but I wouldn't try to fight this force. I'd rather use it. So I've thought a lot about where to live.

I'd always imagined Berkeley would be the ideal place — that it would basically be Cambridge with good weather. But when I finally tried living there a couple years ago, it turned out not to be. The message Berkeley sends is: you should live better. Life in Berkeley is very civilized. It's probably the place in America where someone from Northern Europe would feel most at home. But it's not humming with ambition.

In retrospect it shouldn't have been surprising that a place so pleasant would attract people interested above all in quality of life. Cambridge with good weather, it turns out, is not Cambridge. The people you find in Cambridge are not there by accident. You have to make sacrifices to live there. It's expensive and somewhat grubby, and the weather's often bad. So the kind of people you find in Cambridge are the kind of people who want to live where the smartest people are, even if that means living in an expensive, grubby place with bad weather.

As of this writing, Cambridge seems to be the intellectual capital of the world. I realize that seems a preposterous claim. What makes it true is that it's more preposterous to claim about anywhere else. American universities currently seem to be the best, judging from the flow of ambitious students. And what US city has a stronger claim? New York? A fair number of smart people, but diluted by a much larger number of neanderthals in suits. The Bay Area has a lot of smart people too, but again, diluted; there are two great universities, but they're far apart. Harvard and MIT are practically adjacent by West Coast standards, and they're surrounded by about 20 other colleges and universities. [\[1\]](#)

Cambridge as a result feels like a town whose main industry is ideas, while New York's is finance and Silicon Valley's is startups.

---

When you talk about cities in the sense we are, what you're really talking about is collections of people. For a long time cities were the only large collections of people, so you could use the two ideas interchangeably. But we can see how much things are changing from the examples I've mentioned. New York is a classic great city. But Cambridge is just part of a city, and Silicon Valley is not even that. (San Jose is not, as it sometimes claims, the capital of Silicon Valley. It's just 178

square miles at one end of it.)

Maybe the Internet will change things further. Maybe one day the most important community you belong to will be a virtual one, and it won't matter where you live physically. But I wouldn't bet on it. The physical world is very high bandwidth, and some of the ways cities send you messages are quite subtle.

One of the exhilarating things about coming back to Cambridge every spring is walking through the streets at dusk, when you can see into the houses. When you walk through Palo Alto in the evening, you see nothing but the blue glow of TVs. In Cambridge you see shelves full of promising-looking books. Palo Alto was probably much like Cambridge in 1960, but you'd never guess now that there was a university nearby. Now it's just one of the richer neighborhoods in Silicon Valley.

[2]

A city speaks to you mostly by accident — in things you see through windows, in conversations you overhear. It's not something you have to seek out, but something you can't turn off. One of the occupational hazards of living in Cambridge is overhearing the conversations of people who use interrogative intonation in declarative sentences. But on average I'll take Cambridge conversations over New York or Silicon Valley ones.

A friend who moved to Silicon Valley in the late 90s said the worst thing about living there was the low quality of the eavesdropping. At the time I thought she was being deliberately eccentric. Sure, it can be interesting to eavesdrop on people, but is good quality eavesdropping so important that it would affect where you chose to live? Now I understand what she meant. The conversations you overhear tell you what sort of people you're among.

---

No matter how determined you are, it's hard not to be influenced by the people around you. It's not so much that you do whatever a city expects of you, but that you get discouraged when no one around you cares about the same things you do.

There's an imbalance between encouragement and discouragement like that between gaining and losing money. Most people overvalue negative amounts of money: they'll work much harder to avoid losing a dollar than to gain one. Similarly, although there are plenty of people strong enough to resist doing something just because that's what one is supposed to do where they happen to be, there are few strong enough to keep working on something no one around them cares about.

Because ambitions are to some extent incompatible and admiration is a zero-sum game, each city tends to focus on one type of ambition. The reason Cambridge is the intellectual capital is not just that there's a concentration of smart people there, but that there's nothing *e/se* people there care about more. Professors in

New York and the Bay area are second class citizens — till they start hedge funds or startups respectively.

This suggests an answer to a question people in New York have wondered about since the Bubble: whether New York could grow into a startup hub to rival Silicon Valley. One reason that's unlikely is that someone starting a startup in New York would feel like a second class citizen. [3] There's already something else people in New York admire more.

In the long term, that could be a bad thing for New York. The power of an important new technology does eventually convert to money. So by caring more about money and less about power than Silicon Valley, New York is recognizing the same thing, but slower. [4] And in fact it has been losing to Silicon Valley at its own game: the ratio of New York to California residents in the Forbes 400 has decreased from 1.45 (81:56) when the list was first published in 1982 to .83 (73:88) in 2007.

---

Not all cities send a message. Only those that are centers for some type of ambition do. And it can be hard to tell exactly what message a city sends without living there. I understand the messages of New York, Cambridge, and Silicon Valley because I've lived for several years in each of them. DC and LA seem to send messages too, but I haven't spent long enough in either to say for sure what they are.

The big thing in LA seems to be fame. There's an A List of people who are most in demand right now, and what's most admired is to be on it, or friends with those who are. Beneath that, the message is much like New York's, though perhaps with more emphasis on physical attractiveness.

In DC the message seems to be that the most important thing is who you know. You want to be an insider. In practice this seems to work much as in LA. There's an A List and you want to be on it or close to those who are. The only difference is how the A List is selected. And even that is not that different.

At the moment, San Francisco's message seems to be the same as Berkeley's: you should live better. But this will change if enough startups choose SF over the Valley. During the Bubble that was a predictor of failure — a self-indulgent choice, like buying expensive office furniture. Even now I'm suspicious when startups choose SF. But if enough good ones do, it stops being a self-indulgent choice, because the center of gravity of Silicon Valley will shift there.

I haven't found anything like Cambridge for intellectual ambition. Oxford and Cambridge (England) feel like Ithaca or Hanover: the message is there, but not as strong.

Paris was once a great intellectual center. If you went there in 1300, it might have sent the message Cambridge does now. But I tried living there for a bit last year, and the ambitions of the inhabitants are not intellectual ones. The message Paris sends now is: do things with style. I liked that, actually. Paris is the only city I've lived in where people genuinely cared about art. In America only a few rich people buy original art, and even the more sophisticated ones rarely get past judging it by the brand name of the artist. But looking through windows at dusk in Paris you can see that people there actually care what paintings look like. Visually, Paris has the best eavesdropping I know. [5]

There's one more message I've heard from cities: in London you can still (barely) hear the message that one should be more aristocratic. If you listen for it you can also hear it in Paris, New York, and Boston. But this message is everywhere very faint. It would have been strong 100 years ago, but now I probably wouldn't have picked it up at all if I hadn't deliberately tuned in to that wavelength to see if there was any signal left.

---

So far the complete list of messages I've picked up from cities is: wealth, style, hipness, physical attractiveness, fame, political power, economic power, intelligence, social class, and quality of life.

My immediate reaction to this list is that it makes me slightly queasy. I'd always considered ambition a good thing, but I realize now that was because I'd always implicitly understood it to mean ambition in the areas I cared about. When you list everything ambitious people are ambitious about, it's not so pretty.

On closer examination I see a couple things on the list that are surprising in the light of history. For example, physical attractiveness wouldn't have been there 100 years ago (though it might have been 2400 years ago). It has always mattered for women, but in the late twentieth century it seems to have started to matter for men as well. I'm not sure why — probably some combination of the increasing power of women, the increasing influence of actors as models, and the fact that so many people work in offices now: you can't show off by wearing clothes too fancy to wear in a factory, so you have to show off with your body instead.

Hipness is another thing you wouldn't have seen on the list 100 years ago. Or wouldn't you? What it means is to know what's what. So maybe it has simply replaced the component of social class that consisted of being "au fait." That could explain why hipness seems particularly admired in London: it's version 2 of the traditional English delight in obscure codes that only insiders understand.

Economic power would have been on the list 100 years ago, but what we mean by it is changing. It used to mean the control of vast human and material resources. But increasingly it means the ability to direct the course of technology, and some of the people in a position to do that are not even rich — leaders of important open



source projects, for example. The Captains of Industry of times past had laboratories full of clever people cooking up new technologies for them. The new breed are themselves those people.

As this force gets more attention, another is dropping off the list: social class. I think the two changes are related. Economic power, wealth, and social class are just names for the same thing at different stages in its life: economic power converts to wealth, and wealth to social class. So the focus of admiration is simply shifting upstream.

---

Does anyone who wants to do great work have to live in a great city? No; all great cities inspire some sort of ambition, but they aren't the only places that do. For some kinds of work, all you need is a handful of talented colleagues.

What cities provide is an audience, and a funnel for peers. These aren't so critical in something like math or physics, where no audience matters except your peers, and judging ability is sufficiently straightforward that hiring and admissions committees can do it reliably. In a field like math or physics all you need is a department with the right colleagues in it. It could be anywhere — in Los Alamos, New Mexico, for example.

It's in fields like the arts or writing or technology that the larger environment matters. In these the best practitioners aren't conveniently collected in a few top university departments and research labs — partly because talent is harder to judge, and partly because people pay for these things, so one doesn't need to rely on teaching or research funding to support oneself. It's in these more chaotic fields that it helps most to be in a great city: you need the encouragement of feeling that people around you care about the kind of work you do, and since you have to find peers for yourself, you need the much larger intake mechanism of a great city.

You don't have to live in a great city your whole life to benefit from it. The critical years seem to be the early and middle ones of your career. Clearly you don't have to grow up in a great city. Nor does it seem to matter if you go to college in one. To most college students a world of a few thousand people seems big enough. Plus in college you don't yet have to face the hardest kind of work — discovering new problems to solve.

It's when you move on to the next and much harder step that it helps most to be in a place where you can find peers and encouragement. You seem to be able to leave, if you want, once you've found both. The Impressionists show the typical pattern: they were born all over France (Pissarro was born in the Caribbean) and died all over France, but what defined them were the years they spent together in Paris.

---

Unless you're sure what you want to do and where the leading center for it is, your best bet is probably to try living in several places when you're young. You can never tell what message a city sends till you live there, or even whether it still sends one. Often your information will be wrong: I tried living in Florence when I was 25, thinking it would be an art center, but it turned out I was 450 years too late.

Even when a city is still a live center of ambition, you won't know for sure whether its message will resonate with you till you hear it. When I moved to New York, I was very excited at first. It's an exciting place. So it took me quite a while to realize I just wasn't like the people there. I kept searching for the Cambridge of New York. It turned out it was way, way uptown: an hour uptown by air.

Some people know at 16 what sort of work they're going to do, but in most ambitious kids, ambition seems to precede anything specific to be ambitious about. They know they want to do something great. They just haven't decided yet whether they're going to be a rock star or a brain surgeon. There's nothing wrong with that. But it means if you have this most common type of ambition, you'll probably have to figure out where to live by trial and error. You'll probably have to find the city where you feel at home to know what sort of ambition you have.

## Notes

[1] This is one of the advantages of not having the universities in your country controlled by the government. When governments decide how to allocate resources, political deal-making causes things to be spread out geographically. No central government would put its two best universities in the same town, unless it was the capital (which would cause other problems). But scholars seem to like to cluster together as much as people in any other field, and when given the freedom to they derive the same advantages from it.

[2] There are still a few old professors in Palo Alto, but one by one they die and their houses are transformed by developers into McMansions and sold to VPs of Bus Dev.

[3] How many times have you read about startup founders who continued to live inexpensively as their companies took off? Who continued to dress in jeans and t-shirts, to drive the old car they had in grad school, and so on? If you did that in New York, people would treat you like shit. If you walk into a fancy restaurant in San Francisco wearing a jeans and a t-shirt, they're nice to you; who knows who

you might be? Not in New York.

One sign of a city's potential as a technology center is the number of restaurants that still require jackets for men. According to Zagat's there are none in San Francisco, LA, Boston, or Seattle, 4 in DC, 6 in Chicago, 8 in London, 13 in New York, and 20 in Paris.

(Zagat's lists the Ritz Carlton Dining Room in SF as requiring jackets but I couldn't believe it, so I called to check and in fact they don't. Apparently there's only one restaurant left on the entire West Coast that still requires jackets: The French Laundry in Napa Valley.)

[4] Ideas are one step upstream from economic power, so it's conceivable that intellectual centers like Cambridge will one day have an edge over Silicon Valley like the one the Valley has over New York.

This seems unlikely at the moment; if anything Boston is falling further and further behind. The only reason I even mention the possibility is that the path from ideas to startups has recently been getting smoother. It's a lot easier now for a couple of hackers with no business experience to start a startup than it was 10 years ago. If you extrapolate another 20 years, maybe the balance of power will start to shift back. I wouldn't bet on it, but I wouldn't bet against it either.

[5] If Paris is where people care most about art, why is New York the center of gravity of the art business? Because in the twentieth century, art as brand split apart from art as stuff. New York is where the richest buyers are, but all they demand from art is brand, and since you can base brand on anything with a sufficiently identifiable style, you may as well use the local stuff.

**Thanks** to Trevor Blackwell, Sarah Harlin, Jessica Livingston, Jackie McDonough, Robert Morris, and David Sloo for reading drafts of this.

[Italian Translation](#)

[Portuguese Translation](#)

[Chinese Translation](#)

[Korean Translation](#)

# The Pooled-Risk Company Management Company

July 2008

At this year's startup school, David Heinemeier Hansson gave a [talk](#) in which he suggested that startup founders should do things the old fashioned way. Instead of hoping to get rich by building a valuable company and then selling stock in a "liquidity event," founders should start companies that make money and live off the revenues.

Sounds like a good plan. Let's think about the optimal way to do this.

One disadvantage of living off the revenues of your company is that you have to keep running it. And as anyone who runs their own business can tell you, that requires your complete attention. You can't just start a business and check out once things are going well, or they stop going well surprisingly fast.

The main economic motives of startup founders seem to be freedom and security. They want enough money that (a) they don't have to worry about running out of money and (b) they can spend their time how they want. Running your own business offers neither. You certainly don't have freedom: no boss is so demanding. Nor do you have security, because if you stop paying attention to the company, its revenues go away, and with them your income.

The best case, for most people, would be if you could hire someone to manage the company for you once you'd grown it to a certain size. Suppose you could find a really good manager. Then you would have both freedom and security. You could pay as little attention to the business as you wanted, knowing that your manager would keep things running smoothly. And that being so, revenues would continue to flow in, so you'd have security as well.

There will of course be some founders who wouldn't like that idea: the ones who like running their company so much that there's nothing else they'd rather do. But this group must be small. The way you succeed in most businesses is to be fanatically attentive to customers' needs. What are the odds that your own desires would coincide exactly with the demands of this powerful, external force?

Sure, running your own company can be fairly interesting. Viaweb was more

interesting than any job I'd had before. And since I made much more money from it, it offered the highest ratio of income to boringness of anything I'd done, by orders of magnitude. But was it *the* most interesting work I could imagine doing? No.

Whether the number of founders in the same position is asymptotic or merely large, there are certainly a lot of them. For them the right approach would be to hand the company over to a professional manager eventually, if they could find one who was good enough.

---

So far so good. But what if your manager was hit by a bus? What you really want is a management company to run your company for you. Then you don't depend on any one person.

If you own rental property, there are companies you can hire to manage it for you. Some will do everything, from finding tenants to fixing leaks. Of course, running companies is a lot more complicated than managing rental property, but let's suppose there were management companies that could do it for you. They'd charge a lot, but wouldn't it be worth it? I'd sacrifice a large percentage of the income for the extra peace of mind.

I realize what I'm describing already sounds too good to be true, but I can think of a way to make it even more attractive. If company management companies existed, there would be an additional service they could offer clients: they could let them insure their returns by pooling their risk. After all, even a perfect manager can't save a company when, as sometimes happens, its whole market dies, just as property managers can't save you from the building burning down. But a company that managed a large enough number of companies could say to all its clients: we'll combine the revenues from all your companies, and pay you your proportionate share.

If such management companies existed, they'd offer the maximum of freedom and security. Someone would run your company for you, and you'd be protected even if it happened to die.

Let's think about how such a management company might be organized. The simplest way would be to have a new kind of stock representing the total pool of companies they were managing. When you signed up, you'd trade your company's stock for shares of this pool, in proportion to an estimate of your company's value that you'd both agreed upon. Then you'd automatically get your share of the returns of the whole pool.

The catch is that because this kind of trade would be hard to undo, you couldn't switch management companies. But there's a way they could fix that: suppose all the company management companies got together and agreed to allow their

clients to exchange shares in all their pools. Then you could, in effect, simultaneously choose all the management companies to run yours for you, in whatever proportion you wanted, and change your mind later as often as you wanted.

If such pooled-risk company management companies existed, signing up with one would seem the ideal plan for most people following the route David advocated.

Good news: they do exist. What I've just described is an acquisition by a public company.

---

Unfortunately, though public acquirers are structurally identical to pooled-risk company management companies, they don't think of themselves that way. With a property management company, you can just walk in whenever you want and say "manage my rental property for me" and they'll do it. Whereas acquirers are, as of this writing, extremely fickle. Sometimes they're in a buying mood and they'll overpay enormously; other times they're not interested. They're like property management companies run by madmen. Or more precisely, by Benjamin Graham's Mr. Market.

So while on average public acquirers behave like pooled-risk company managers, you need a window of several years to get average case performance. If you wait long enough (five years, say) you're likely to hit an up cycle where some acquirer is hot to buy you. But you can't choose when it happens.

You can't assume investors will carry you for as long as you might have to wait. Your company has to make money. Opinions are divided about how early to focus on that. [Joe Kraus](#) says you should try charging customers right away. And yet some of the most successful startups, including Google, ignored revenue at first and concentrated exclusively on development. The answer probably depends on the type of company you're starting. I can imagine some where trying to make sales would be a good heuristic for product design, and others where it would just be a distraction. The test is probably whether it helps you to understand your users.

You can choose whichever revenue strategy you think is best for the type of company you're starting, so long as you're profitable. Being profitable ensures you'll get at least the average of the acquisition market—in which public companies do behave as pooled-risk company management companies.

David isn't mistaken in saying you should start a company to live off its revenues. The mistake is thinking this is somehow opposed to starting a company and selling it. In fact, for most people the latter is merely the optimal case of the former.

**Thanks** to Trevor Blackwell, Jessica Livingston, Michael Mandel, Robert Morris, and Fred Wilson for reading drafts of this.

[Russian Translation](#)

# A Fundraising Survival Guide

August 2008

Raising money is the second hardest part of starting a startup. The hardest part is making something people want: most startups that die, die because they didn't do that. But the second biggest cause of death is probably the difficulty of raising money. Fundraising is brutal.

One reason it's so brutal is simply the brutality of markets. People who've spent most of their lives in schools or big companies may not have been exposed to that. Professors and bosses usually feel some sense of responsibility toward you; if you make a valiant effort and fail, they'll cut you a break. Markets are less forgiving. Customers don't care how hard you worked, only whether you solved their problems.

Investors evaluate startups the way customers evaluate products, not the way bosses evaluate employees. If you're making a valiant effort and failing, maybe they'll invest in your next startup, but not this one.

But raising money from investors is harder than selling to customers, because there are so few of them. There's nothing like an efficient market. You're unlikely to have more than 10 who are interested; it's difficult to talk to more. So the randomness of any one investor's behavior can really affect you.

Problem number 3: investors are very random. All investors, including us, are by ordinary standards incompetent. We constantly have to make decisions about things we don't understand, and more often than not we're wrong.

And yet a lot is at stake. The amounts invested by different types of investors vary from five thousand dollars to fifty million, but the amount usually seems large for whatever type of investor it is. Investment decisions are big decisions.

That combination—making big decisions about things they don't understand—tends to make investors very skittish. VCs are notorious for leading founders on. Some of the more unscrupulous do it deliberately. But even the most well-intentioned investors can behave in a way that would seem crazy in everyday life. One day they're full of enthusiasm and seem ready to write you a check on the spot; the next they won't return your phone calls. They're not playing games with you. They just can't make up their minds. [\[1\]](#)

If that weren't bad enough, these wildly fluctuating nodes are all linked together. Startup investors all know one another, and (though they hate to admit it) the biggest factor in their opinion of you is the opinion of other investors. [\[2\]](#) Talk



about a recipe for an unstable system. You get the opposite of the damping that the fear/greed balance usually produces in markets. No one is interested in a startup that's a "bargain" because everyone else hates it.

So the inefficient market you get because there are so few players is exacerbated by the fact that they act less than independently. The result is a system like some kind of primitive, multi-celled sea creature, where you irritate one extremity and the whole thing contracts violently.

Y Combinator is working to fix this. We're trying to increase the number of investors just as we're increasing the number of startups. We hope that as the number of both increases we'll get something more like an efficient market. As  $t$  approaches infinity, Demo Day approaches an auction.

Unfortunately,  $t$  is still very far from infinity. What does a startup do now, in the imperfect world we currently inhabit? The most important thing is not to let fundraising get you down. Startups live or die on morale. If you let the difficulty of raising money destroy your morale, it will become a self-fulfilling prophecy.

### **Bootstrapping (= Consulting)**

Some would-be founders may by now be thinking, why deal with investors at all? If raising money is so painful, why do it?

One answer to that is obvious: because you need money to live on. It's a fine idea in principle to finance your startup with its own revenues, but you can't create instant customers. Whatever you make, you have to sell a certain amount to break even. It will take time to grow your sales to that point, and it's hard to predict, till you try, how long it will take.

We could not have bootstrapped Viaweb, for example. We charged quite a lot for our software—about \$140 per user per month—but it was at least a year before our revenues would have covered even our paltry costs. We didn't have enough saved to live on for a year.

If you factor out the "bootstrapped" companies that were actually funded by their founders through savings or a day job, the remainder either (a) got really lucky, which is hard to do on demand, or (b) began life as consulting companies and gradually transformed themselves into product companies.

Consulting is the only option you can count on. But consulting is far from free money. It's not as painful as raising money from investors, perhaps, but the pain is spread over a longer period. Years, probably. And for many types of startup, that delay could be fatal. If you're working on something so unusual that no one else is likely to think of it, you can take your time. Joshua Schachter gradually built Delicious on the side while working on Wall Street. He got away with it because no one else realized it was a good idea. But if you were building something as obviously necessary as online store software at about the same time as Viaweb, and you were working on it on the side while spending most of your time on client work, you were not in a good position.

Bootstrapping sounds great in principle, but this apparently verdant territory is one from which few startups emerge alive. The mere fact that bootstrapped startups tend to be famous on that account should set off alarm bells. If it worked so well, it

would be the norm. [3]

Bootstrapping may get easier, because starting a company is getting cheaper. But I don't think we'll ever reach the point where most startups can do without outside funding. Technology tends to get dramatically cheaper, but living expenses don't.

The upshot is, you can choose your pain: either the short, sharp pain of raising money, or the chronic ache of consulting. For a given total amount of pain, raising money is the better choice, because new technology is usually more valuable now than later.

But although for most startups raising money will be the lesser evil, it's still a pretty big evil—so big that it can easily kill you. Not merely in the obvious sense that if you fail to raise money you might have to shut the company down, but because the *process* of raising money itself can kill you.

To survive it you need a set of techniques mostly orthogonal to the ones used in convincing investors, just as mountain climbers need to know survival techniques that are mostly orthogonal to those used in physically getting up and down mountains.

## **1. Have low expectations.**

The reason raising money destroys so many startups' morale is not simply that it's hard, but that it's so much harder than they expected. What kills you is the disappointment. And the lower your expectations, the harder it is to be disappointed.

Startup founders tend to be optimistic. This can work well in technology, at least some of the time, but it's the wrong way to approach raising money. Better to assume investors will always let you down. Acquirers too, while we're at it. At YC one of our secondary mantras is "Deals fall through." No matter what deal you have going on, assume it will fall through. The predictive power of this simple rule is amazing.

There will be a tendency, as a deal progresses, to start to believe it will happen, and then to depend on it happening. You must resist this. Tie yourself to the mast. This is what kills you. Deals do not have a trajectory like most other human interactions, where shared plans solidify linearly over time. Deals often fall through at the last moment. Often the other party doesn't really think about what they want till the last moment. So you can't use your everyday intuitions about shared plans as a guide. When it comes to deals, you have to consciously turn them off and become pathologically cynical.

This is harder to do than it sounds. It's very flattering when eminent investors seem interested in funding you. It's easy to start to believe that raising money will be quick and straightforward. But it hardly ever is.

## **2. Keep working on your startup.**

It sounds obvious to say that you should keep working on your startup while raising money. Actually this is hard to do. Most startups don't manage to.

Raising money has a mysterious capacity to suck up all your attention. Even if you

only have one meeting a day with investors, somehow that one meeting will burn up your whole day. It costs not just the time of the actual meeting, but the time getting there and back, and the time preparing for it beforehand and thinking about it afterward.

The best way to survive the distraction of meeting with investors is probably to partition the company: to pick one founder to deal with investors while the others keep the company going. This works better when a startup has 3 founders than 2, and better when the leader of the company is not also the lead developer. In the best case, the company keeps moving forward at about half speed.

That's the best case, though. More often than not the company comes to a standstill while raising money. And that is dangerous for so many reasons. Raising money always takes longer than you expect. What seems like it's going to be a 2 week interruption turns into a 4 month interruption. That can be very demoralizing. And worse still, it can make you less attractive to investors. They want to invest in companies that are dynamic. A company that hasn't done anything new in 4 months doesn't seem dynamic, so they start to lose interest. Investors rarely grasp this, but much of what they're responding to when they lose interest in a startup is the damage done by their own indecision.

The solution: put the startup first. Fit meetings with investors into the spare moments in your development schedule, rather than doing development in the spare moments between meetings with investors. If you keep the company moving forward—releasing new features, increasing traffic, doing deals, getting written about—those investor meetings are more likely to be productive. Not just because your startup will seem more alive, but also because it will be better for your own morale, which is one of the main ways investors judge you.

### **3. Be conservative.**

As conditions get worse, the optimal strategy becomes more conservative. When things go well you can take risks; when things are bad you want to play it safe.

I advise approaching fundraising as if it were always going badly. The reason is that between your ability to delude yourself and the wildly unstable nature of the system you're dealing with, things probably either already are or could easily become much worse than they seem.

What I tell most startups we fund is that if someone reputable offers you funding on reasonable terms, take it. There have been startups that ignored this advice and got away with it—startups that ignored a good offer in the hope of getting a better one, and actually did. But in the same position I'd give the same advice again. Who knows how many bullets were in the gun they were playing Russian roulette with?

Corollary: if an investor seems interested, don't just let them sit. You can't assume someone interested in investing will stay interested. In fact, you can't even tell (*they* can't even tell) if they're really interested till you try to convert that interest into money. So if you have hot prospect, either close them now or write them off. And unless you already have enough funding, that reduces to: close them now.

Startups don't win by getting great funding rounds, but by making great products. So finish raising money and get back to work.

## 4. Be flexible.

There are two questions VCs ask that you shouldn't answer: "Who else are you talking to?" and "How much are you trying to raise?"

VCs don't expect you to answer the first question. They ask it just in case. [4] They do seem to expect an answer to the second. But I don't think you should just tell them a number. Not as a way to play games with them, but because you shouldn't *have* a fixed amount you need to raise.

The custom of a startup needing a fixed amount of funding is an obsolete one left over from the days when startups were more expensive. A company that needed to build a factory or hire 50 people obviously needed to raise a certain minimum amount. But few technology startups are in that position today.

We advise startups to tell investors there are several different routes they could take depending on how much they raised. As little as \$50k could pay for food and rent for the founders for a year. A couple hundred thousand would let them get office space and hire some smart people they know from school. A couple million would let them really blow this thing out. The message (and not just the message, but the fact) should be: we're going to succeed no matter what. Raising more money just lets us do it faster.

If you're raising an angel round, the size of the round can even change on the fly. In fact, it's just as well to make the round small initially, then expand as needed, rather than trying to raise a large round and risk losing the investors you already have if you can't raise the full amount. You may even want to do a "rolling close," where the round has no predetermined size, but instead you sell stock to investors one at a time as they say yes. That helps break deadlocks, because you can start as soon as the first one is ready to buy. [5]

## 5. Be independent.

A startup with a couple founders in their early twenties can have expenses so low that they could be profitable on as little as \$2000 per month. That's negligible as corporate revenues go, but the effect on your morale and your bargaining position is anything but. At YC we use the phrase "ramen profitable" to describe the situation where you're making just enough to pay your living expenses. Once you cross into ramen profitable, everything changes. You may still need investment to make it big, but you don't need it this month.

You can't plan when you start a startup how long it will take to become profitable. But if you find yourself in a position where a little more effort expended on sales would carry you over the threshold of ramen profitable, do it.

Investors like it when you're ramen profitable. It shows you've thought about making money, instead of just working on amusing technical problems; it shows you have the discipline to keep your expenses low; but above all, it means you don't need them.

There is nothing investors like more than a startup that seems like it's going to succeed even without them. Investors like it when they can help a startup, but they don't like startups that would die without that help.

At YC we spend a lot of time trying to predict how the startups we've funded will do, because we're trying to learn how to pick winners. We've now watched the trajectories of so many startups that we're getting better at predicting them. And when we're talking about startups we think are likely to succeed, what we find ourselves saying is things like "Oh, those guys can take care of themselves. They'll be fine." Not "those guys are really smart" or "those guys are working on a great idea." [6] When we predict good outcomes for startups, the qualities that come up in the supporting arguments are toughness, adaptability, determination. Which means to the extent we're correct, those are the qualities you need to win.

Investors know this, at least unconsciously. The reason they like it when you don't need them is not simply that they like what they can't have, but because that quality is what makes founders succeed.

[Sam Altman](#) has it. You could parachute him into an island full of cannibals and come back in 5 years and he'd be the king. If you're Sam Altman, you don't have to be profitable to convey to investors that you'll succeed with or without them. (He wasn't, and he did.) Not everyone has Sam's deal-making ability. I myself don't. But if you don't, you can let the numbers speak for you.

## **6. Don't take rejection personally.**

Getting rejected by investors can make you start to doubt yourself. After all, they're more experienced than you. If they think your startup is lame, aren't they probably right?

Maybe, maybe not. The way to handle rejection is with precision. You shouldn't simply ignore rejection. It might mean something. But you shouldn't automatically get demoralized either.

To understand what rejection means, you have to understand first of all how common it is. Statistically, the average VC is a rejection machine. David Hornik, a partner at August, told me:

The numbers for me ended up being something like 500 to 800 plans received and read, somewhere between 50 and 100 initial 1 hour meetings held, about 20 companies that I got interested in, about 5 that I got serious about and did a bunch of work, 1 to 2 deals done in a year. So the odds are against you. You may be a great entrepreneur, working on interesting stuff, etc. but it is still incredibly unlikely that you get funded.

This is less true with angels, but VCs reject practically everyone. The structure of their business means a partner does at most 2 new investments a year, no matter how many good startups approach him.

In addition to the odds being terrible, the average investor is, as I mentioned, a pretty bad judge of startups. It's harder to judge startups than most other things, because great startup ideas tend to seem wrong. A good startup idea has to be not just good but novel. And to be both good and novel, an idea probably has to seem bad to most people, or someone would already be doing it and it wouldn't be novel.

That makes judging startups harder than most other things one judges. You have to be an intellectual contrarian to be a good startup investor. That's a problem for VCs, most of whom are not particularly imaginative. VCs are mostly money guys, not people who make things. [7] Angels are better at appreciating novel ideas, because most were founders themselves.

So when you get a rejection, use the data that's in it, and not what's not. If an investor gives you specific reasons for not investing, look at your startup and ask if they're right. If they're real problems, fix them. But don't just take their word for it. You're supposed to be the domain expert; you have to decide.

Though a rejection doesn't necessarily tell you anything about your startup, it does suggest your pitch could be improved. Figure out what's not working and change it. Don't just think "investors are stupid." Often they are, but figure out precisely where you lose them.

Don't let rejections pile up as a depressing, undifferentiated heap. Sort them and analyze them, and then instead of thinking "no one likes us," you'll know precisely how big a problem you have, and what to do about it.

## **7. Be able to downshift into consulting (if appropriate).**

Consulting, as I mentioned, is a dangerous way to finance a startup. But it's better than dying. It's a bit like anaerobic respiration: not the optimum solution for the long term, but it can save you from an immediate threat. If you're having trouble raising money from investors at all, it could save you to be able to shift toward consulting.

This works better for some startups than others. It wouldn't have been a natural fit for, say, Google, but if your company was making software for building web sites, you could degrade fairly gracefully into consulting by building sites for clients with it.

So long as you were careful not to get sucked permanently into consulting, this could even have advantages. You'd understand your users well if you were using the software for them. Plus as a consulting company you might be able to get big-name users using your software that you wouldn't have gotten as a product company.

At Viaweb we were forced to operate like a consulting company initially, because we were so desperate for users that we'd offer to build merchants' sites for them if they'd sign up. But we never charged for such work, because we didn't want them to start treating us like actual consultants, and calling us every time they wanted something changed on their site. We knew we had to stay a product company, because only that scales.

## **8. Avoid inexperienced investors.**

Though novice investors seem unthreatening they can be the most dangerous sort, because they're so nervous. Especially in proportion to the amount they invest. Raising \$20,000 from a first-time angel investor can be as much work as raising \$2 million from a VC fund.

Their lawyers are generally inexperienced too. But while the investors can admit they don't know what they're doing, their lawyers can't. One YC startup negotiated terms for a tiny round with an angel, only to receive a 70-page agreement from his lawyer. And since the lawyer could never admit, in front of his client, that he'd screwed up, he instead had to insist on retaining all the draconian terms in it, so the deal fell through.

Of course, someone has to take money from novice investors, or there would never be any experienced ones. But if you do, either (a) drive the process yourself, including supplying the [paperwork](#), or (b) use them only to fill up a larger round led by someone else.

## **9. Know where you stand.**

The most dangerous thing about investors is their indecisiveness. The worst case scenario is the long no, the no that comes after months of meetings. Rejections from investors are like design flaws: inevitable, but much less costly if you discover them early.

So while you're talking to investors, constantly look for signs of where you stand. How likely are they to offer you a term sheet? What do they have to be convinced of first? You shouldn't necessarily always be asking these questions outright—that could get annoying—but you should always be collecting data about them.

Investors tend to resist committing except to the extent you push them to. It's in their interest to collect the maximum amount of information while making the minimum number of decisions. The best way to force them to act is, of course, competing investors. But you can also apply some force by focusing the discussion: by asking what specific questions they need answered to make up their minds, and then answering them. If you get through several obstacles and they keep raising new ones, assume that ultimately they're going to flake.

You have to be disciplined when collecting data about investors' intentions. Otherwise their desire to lead you on will combine with your own desire to be led on to produce completely inaccurate impressions.

Use the data to weight your strategy. You'll probably be talking to several investors. Focus on the ones that are most likely to say yes. The value of a potential investor is a combination of how good it would be if they said yes, and how likely they are to say it. Put the most weight on the second factor. Partly because the most important quality in an investor is simply investing. But also because, as I mentioned, the biggest factor in investors' opinion of you is other

investors' opinion of you. If you're talking to several investors and you manage to get one over the threshold of saying yes, it will make the others much more interested. So you're not sacrificing the lukewarm investors if you focus on the hot ones; convincing the hot investors is the best way to convince the lukewarm ones.

## **Future**

I'm hopeful things won't always be so awkward. I hope that as startups get cheaper and the number of investors increases, raising money will become, if not easy, at least straightforward.

In the meantime, the brokenness of the funding process offers a big opportunity. Most investors have no idea how dangerous they are. They'd be surprised to hear that raising money from them is something that has to be treated as a threat to a company's survival. They just think they need a little more information to make up their minds. They don't get that there are 10 other investors who also want a little more information, and that the process of talking to them all can bring a startup to a standstill for months.

Because investors don't understand the cost of dealing with them, they don't realize how much room there is for a potential competitor to undercut them. I know from my own experience how much faster investors could decide, because we've brought our own time down to 20 minutes (5 minutes of reading an application plus a 10 minute interview plus 5 minutes of discussion). If you were investing more money you'd want to take longer, of course. But if we can decide in 20 minutes, should it take anyone longer than a couple days?

Opportunities like this don't sit unexploited forever, even in an industry as conservative as venture capital. So either existing investors will start to make up their minds faster, or new investors will emerge who do.

In the meantime founders have to treat raising money as a dangerous process. Fortunately, I can fix the biggest danger right here. The biggest danger is surprise. It's that startups will underestimate the difficulty of raising money—that they'll cruise through all the initial steps, but when they turn to raising money they'll find it surprisingly hard, get demoralized, and give up. So I'm telling you in advance: raising money is hard.

## **Notes**

[1] When investors can't make up their minds, they sometimes describe it as if it were a property of the startup. "You're too early for us," they sometimes say. But which of them, if they were taken back in a time machine to the hour Google was founded, wouldn't offer to invest at any valuation the founders chose? An hour old



is not too early if it's the right startup. What "you're too early" really means is "we can't figure out yet whether you'll succeed."

[2] Investors influence one another both directly and indirectly. They influence one another directly through the "buzz" that surrounds a hot startup. But they also influence one another indirectly *through the founders*. When a lot of investors are interested in you, it increases your confidence in a way that makes you much more attractive to investors.

No VC will admit they're influenced by buzz. Some genuinely aren't. But there are few who can say they're not influenced by confidence.

[3] One VC who read this essay wrote:

"We try to avoid companies that got bootstrapped with consulting. It creates very bad behaviors/instincts that are hard to erase from a company's culture."

[4] The optimal way to answer the first question is to say that it would be improper to name names, while simultaneously implying that you're talking to a bunch of other VCs who are all about to give you term sheets. If you're the sort of person who understands how to do that, go ahead. If not, don't even try. Nothing annoys VCs more than clumsy efforts to manipulate them.

[5] The disadvantage of expanding a round on the fly is that the valuation is fixed at the start, so if you get a sudden rush of interest, you may have to decide between turning some investors away and selling more of the company than you meant to. That's a good problem to have, however.

[6] I wouldn't say that intelligence doesn't matter in startups. We're only comparing YC startups, who've already made it over a certain threshold.

[7] But not all are. Though most VCs are suits at heart, the most successful ones tend not to be. Oddly enough, the best VCs tend to be the least VC-like.

**Thanks** to Trevor Blackwell, David Hornik, Jessica Livingston, Robert Morris, and Fred Wilson for reading drafts of this.

[Russian Translation](#)

# Why to Start a Startup in a Bad Economy

October 2008

The economic situation is apparently so grim that some experts fear we may be in for a stretch as bad as the mid seventies.

When Microsoft and Apple were founded.

As those examples suggest, a recession may not be such a bad time to start a startup. I'm not claiming it's a particularly good time either. The truth is more boring: the state of the economy doesn't matter much either way.

If we've learned one thing from funding so many startups, it's that they succeed or fail based on the qualities of the founders. The economy has some effect, certainly, but as a predictor of success it's rounding error compared to the founders.

Which means that what matters is who you are, not when you do it. If you're the right sort of person, you'll win even in a bad economy. And if you're not, a good economy won't save you. Someone who thinks "I better not start a startup now, because the economy is so bad" is making the same mistake as the people who thought during the Bubble "all I have to do is start a startup, and I'll be rich."

So if you want to improve your chances, you should think far more about who you can recruit as a cofounder than the state of the economy. And if you're worried about threats to the survival of your company, don't look for them in the news. Look in the mirror.

But for any given team of founders, would it not pay to wait till the economy is better before taking the leap? If you're starting a restaurant, maybe, but not if you're working on technology. Technology progresses more or less independently of the stock market. So for any given idea, the payoff for acting fast in a bad economy will be higher than for waiting. Microsoft's first product was a Basic interpreter for the Altair. That was exactly what the world needed in 1975, but if Gates and Allen had decided to wait a few years, it would have been too late.

Of course, the idea you have now won't be the last you have. There are always

new ideas. But if you have a specific idea you want to act on, act now.

That doesn't mean you can ignore the economy. Both customers and investors will be feeling pinched. It's not necessarily a problem if customers feel pinched: you may even be able to benefit from it, by making things that [save money](#). Startups often make things cheaper, so in that respect they're better positioned to prosper in a recession than big companies.

Investors are more of a problem. Startups generally need to raise some amount of external funding, and investors tend to be less willing to invest in bad times. They shouldn't be. Everyone knows you're supposed to buy when times are bad and sell when times are good. But of course what makes investing so counterintuitive is that in equity markets, good times are defined as everyone thinking it's time to buy. You have to be a contrarian to be correct, and by definition only a minority of investors can be.

So just as investors in 1999 were tripping over one another trying to buy into lousy startups, investors in 2009 will presumably be reluctant to invest even in good ones.

You'll have to adapt to this. But that's nothing new: startups always have to adapt to the whims of investors. Ask any founder in any economy if they'd describe investors as fickle, and watch the face they make. Last year you had to be prepared to explain how your startup was viral. Next year you'll have to explain how it's recession-proof.

(Those are both good things to be. The mistake investors make is not the criteria they use but that they always tend to focus on one to the exclusion of the rest.)

Fortunately the way to make a startup recession-proof is to do exactly what you should do anyway: run it as cheaply as possible. For years I've been telling founders that the surest route to success is to be the cockroaches of the corporate world. The immediate cause of death in a startup is always running out of money. So the cheaper your company is to operate, the harder it is to kill. And fortunately it has gotten very cheap to run a startup. A recession will if anything make it cheaper still.

If nuclear winter really is here, it may be safer to be a cockroach even than to keep your job. Customers may drop off individually if they can no longer afford you, but you're not going to lose them all at once; markets don't "reduce headcount."

What if you quit your job to start a startup that fails, and you can't find another? That could be a problem if you work in sales or marketing. In those fields it can take months to find a new job in a bad economy. But hackers seem to be more liquid. Good hackers can always get some kind of job. It might not be your dream job, but you're not going to starve.

Another advantage of bad times is that there's less competition. Technology trains leave the station at regular intervals. If everyone else is cowering in a corner, you may have a whole car to yourself.

You're an investor too. As a founder, you're buying stock with work: the reason Larry and Sergey are so rich is not so much that they've done work worth tens of billions of dollars, but that they were the first investors in Google. And like any investor you should buy when times are bad.

Were you nodding in agreement, thinking "stupid investors" a few paragraphs ago when I was talking about how investors are reluctant to put money into startups in bad markets, even though that's the time they should rationally be most willing to buy? Well, founders aren't much better. When times get bad, hackers go to grad school. And no doubt that will happen this time too. In fact, what makes the preceding paragraph true is that most readers won't believe it—at least to the extent of acting on it.

So maybe a recession is a good time to start a startup. It's hard to say whether advantages like lack of competition outweigh disadvantages like reluctant investors. But it doesn't matter much either way. It's the people that matter. And for a given set of people working on a given technology, the time to act is always now.

[Russian Translation](#)

[Chinese Translation](#)

[Japanese Translation](#)

# The Other Half of "Artists Ship"

November 2008

One of the differences between big companies and startups is that big companies tend to have developed procedures to protect themselves against mistakes. A startup walks like a toddler, bashing into things and falling over all the time. A big company is more deliberate.

The gradual accumulation of checks in an organization is a kind of learning, based on disasters that have happened to it or others like it. After giving a contract to a supplier who goes bankrupt and fails to deliver, for example, a company might require all suppliers to prove they're solvent before submitting bids.

As companies grow they invariably get more such checks, either in response to disasters they've suffered, or (probably more often) by hiring people from bigger companies who bring with them customs for protecting against new types of disasters.

It's natural for organizations to learn from mistakes. The problem is, people who propose new checks almost never consider that the check itself has a cost.

*Every check has a cost.* For example, consider the case of making suppliers verify their solvency. Surely that's mere prudence? But in fact it could have substantial costs. There's obviously the direct cost in time of the people on both sides who supply and check proofs of the supplier's solvency. But the real costs are the ones you never hear about: the company that would be the best supplier, but doesn't bid because they can't spare the effort to get verified. Or the company that would be the best supplier, but falls just short of the threshold for solvency—which will of course have been set on the high side, since there is no apparent cost of increasing it.

Whenever someone in an organization proposes to add a new check, they should have to explain not just the benefit but the cost. No matter how bad a job they did of analyzing it, this meta-check would at least remind everyone there had to *be* a cost, and send them looking for it.

If companies started doing that, they'd find some surprises. Joel Spolsky recently spoke at Y Combinator about selling software to corporate customers. He said that in most companies software costing up to about \$1000 could be bought by

individual managers without any additional approvals. Above that threshold, software purchases generally had to be approved by a committee. But babysitting this process was so expensive for software vendors that it didn't make sense to charge less than \$50,000. Which means if you're making something you might otherwise have charged \$5000 for, you have to sell it for \$50,000 instead.

The purpose of the committee is presumably to ensure that the company doesn't waste money. And yet the result is that the company pays 10 times as much.

Checks on purchases will always be expensive, because the harder it is to sell something to you, the more it has to cost. And not merely linearly, either. If you're hard enough to sell to, the people who are best at making things don't want to bother. The only people who will sell to you are companies that specialize in selling to you. Then you've sunk to a whole new level of inefficiency. Market mechanisms no longer protect you, because the good suppliers are no longer in the market.

Such things happen constantly to the biggest organizations of all, governments. But checks instituted by governments can cause much worse problems than merely overpaying. Checks instituted by governments can cripple a country's whole economy. Up till about 1400, China was richer and more technologically advanced than Europe. One reason Europe pulled ahead was that the Chinese government restricted long trading voyages. So it was left to the Europeans to explore and eventually to dominate the rest of the world, including China.

In more recent times, Sarbanes-Oxley has practically destroyed the US IPO market. That wasn't the intention of the legislators who wrote it. They just wanted to add a few more checks on public companies. But they forgot to consider the cost. They forgot that companies about to go public are usually rather stretched, and that the weight of a few extra checks that might be easy for General Electric to bear are enough to prevent younger companies from being public at all.

Once you start to think about the cost of checks, you can start to ask other interesting questions. Is the cost increasing or decreasing? Is it higher in some areas than others? Where does it increase discontinuously? If large organizations started to ask questions like that, they'd learn some frightening things.

I think the cost of checks may actually be increasing. The reason is that software plays an increasingly important role in companies, and the people who write software are particularly harmed by checks.

Programmers are unlike many types of workers in that the best ones actually prefer to work hard. This doesn't seem to be the case in most types of work. When I worked in fast food, we didn't prefer the busy times. And when I used to mow lawns, I definitely didn't prefer it when the grass was long after a week of rain.

Programmers, though, like it better when they write more code. Or more precisely, when they release more code. Programmers like to make a difference. Good ones, anyway.

For good programmers, one of the best things about working for a startup is that there are few checks on releases. In true startups, there are no external checks at all. If you have an idea for a new feature in the morning, you can write it and push it to the production servers before lunch. And when you can do that, you have more ideas.

At big companies, software has to go through various approvals before it can be launched. And the cost of doing this can be enormous—in fact, discontinuous. I was talking recently to a group of three programmers whose startup had been acquired a few years before by a big company. When they'd been independent, they could release changes instantly. Now, they said, the absolute fastest they could get code released on the production servers was two weeks.

This didn't merely make them less productive. It made them hate working for the acquirer.

Here's a sign of how much programmers like to be able to work hard: these guys would have *paid* to be able to release code immediately, the way they used to. I asked them if they'd trade 10% of the acquisition price for the ability to release code immediately, and all three instantly said yes. Then I asked what was the maximum percentage of the acquisition price they'd trade for it. They said they didn't want to think about it, because they didn't want to know how high they'd go, but I got the impression it might be as much as half.

They'd have sacrificed hundreds of thousands of dollars, perhaps millions, just to be able to deliver more software to users. And you know what? It would have been perfectly safe to let them. In fact, the acquirer would have been better off; not only wouldn't these guys have broken anything, they'd have gotten a lot more done. So the acquirer is in fact getting worse performance at greater cost. Just like the committee approving software purchases.

And just as the greatest danger of being hard to sell to is not that you overpay but that the best suppliers won't even sell to you, the greatest danger of applying too many checks to your programmers is not that you'll make them unproductive, but that good programmers won't even want to work for you.

Steve Jobs's famous maxim "artists ship" works both ways. Artists aren't merely capable of shipping. They insist on it. So if you don't let people ship, you won't have any artists.

# The High-Res Society

December 2008

For nearly all of history the success of a society was proportionate to its ability to assemble large and disciplined organizations. Those who bet on economies of scale generally won, which meant the largest organizations were the most successful ones.

Things have already changed so much that this is hard for us to believe, but till just a few decades ago the largest organizations tended to be the most progressive. An ambitious kid graduating from college in 1960 wanted to work in the huge, gleaming offices of Ford, or General Electric, or NASA. Small meant small-time. Small in 1960 didn't mean a cool little startup. It meant uncle Sid's shoe store.

When I grew up in the 1970s, the idea of the "corporate ladder" was still very much alive. The standard plan was to try to get into a good college, from which one would be drafted into some organization and then rise to positions of gradually increasing responsibility. The more ambitious merely hoped to climb the same ladder faster. [\[1\]](#)

But in the late twentieth century something changed. It turned out that economies of scale were not the only force at work. Particularly in technology, the increase in speed one could get from smaller groups started to trump the advantages of size.

The future turned out to be different from the one we were expecting in 1970. The domed cities and flying cars we expected have failed to materialize. But fortunately so have the jumpsuits with badges indicating our specialty and rank. Instead of being dominated by a few, giant tree-structured organizations, it's now looking like the economy of the future will be a fluid network of smaller, independent units.

It's not so much that large organizations stopped working. There's no evidence that famously successful organizations like the Roman army or the British East India Company were any less afflicted by protocol and politics than organizations of the same size today. But they were competing against opponents who couldn't change the rules on the fly by discovering new technology. Now it turns out the rule "large and disciplined organizations win" needs to have a qualification appended: "at games that change slowly." No one knew till change reached a sufficient speed.



Large organizations *will* start to do worse now, though, because for the first time in history they're no longer getting the best people. An ambitious kid graduating from college now doesn't want to work for a big company. They want to work for the hot startup that's rapidly growing into one. If they're really ambitious, they want to start it. [2]

This doesn't mean big companies will disappear. To say that startups will succeed implies that big companies will exist, because startups that succeed either become big companies or are acquired by them. [3] But large organizations will probably never again play the leading role they did up till the last quarter of the twentieth century.

It's kind of surprising that a trend that lasted so long would ever run out. How often does it happen that a rule works for thousands of years, then switches polarity?

The millennia-long run of bigger-is-better left us with a lot of [traditions](#) that are now obsolete, but extremely deeply rooted. Which means the ambitious can now do arbitrage on them. It will be very valuable to understand precisely which ideas to keep and which can now be discarded.

The place to look is where the spread of smallness began: in the world of startups.

There have always been occasional cases, particularly in the US, of ambitious people who grew the ladder under them instead of climbing it. But till recently this was an anomalous route that tended to be followed only by outsiders. It was no coincidence that the great industrialists of the nineteenth century had so little formal education. As huge as their companies eventually became, they were all essentially mechanics and shopkeepers at first. That was a social step no one with a college education would take if they could avoid it. Till the rise of technology startups, and in particular, Internet startups, it was very unusual for educated people to start their own businesses.

The eight men who left Shockley Semiconductor to found Fairchild Semiconductor, the original Silicon Valley startup, weren't even trying to start a company at first. They were just looking for a company willing to hire them as a group. Then one of their parents introduced them to a small investment bank that offered to find funding for them to start their own, so they did. But starting a company was an alien idea to them; it was something they backed into. [4]

Now I would guess that practically every Stanford or Berkeley undergrad who knows how to program has at least considered the idea of starting a startup. East Coast universities are not far behind, and British universities only a little behind them. This pattern suggests that attitudes at Stanford and Berkeley are not an anomaly, but a leading indicator. This is the way the world is going.

Of course, Internet startups are still only a fraction of the world's economy. Could a

trend based on them be that powerful?

I think so. There's no reason to suppose there's any limit to the amount of work that could be done in this area. Like science, wealth seems to expand fractally. Steam power was a sliver of the British economy when Watt started working on it. But his work led to more work till that sliver had expanded into something bigger than the whole economy of which it had initially been a part.

The same thing could happen with the Internet. If Internet startups offer the best opportunity for ambitious people, then a lot of ambitious people will start them, and this bit of the economy will balloon in the usual fractal way.

Even if Internet-related applications only become a tenth of the world's economy, this component will set the tone for the rest. The most dynamic part of the economy always does, in everything from salaries to standards of dress. Not just because of its prestige, but because the principles underlying the most dynamic part of the economy tend to be ones that work.

For the future, the trend to bet on seems to be networks of small, autonomous groups whose performance is measured individually. And the societies that win will be the ones with the least impedance.

As with the original industrial revolution, some societies are going to be better at this than others. Within a generation of its birth in England, the Industrial Revolution had spread to continental Europe and North America. But it didn't spread everywhere. This new way of doing things could only take root in places that were prepared for it. It could only spread to places that already had a vigorous middle class.

There is a similar social component to the transformation that began in Silicon Valley in the 1960s. Two new kinds of techniques were developed there: techniques for building integrated circuits, and techniques for building a new type of company designed to grow fast by creating new technology. The techniques for building integrated circuits spread rapidly to other countries. But the techniques for building startups didn't. Fifty years later, startups are ubiquitous in Silicon Valley and common in a handful of other US cities, but they're still an anomaly in most of the world.

Part of the reason—possibly the main reason—that startups have not spread as broadly as the Industrial Revolution did is their social disruptiveness. Though it brought many social changes, the Industrial Revolution was not fighting the principle that bigger is better. Quite the opposite: the two dovetailed beautifully. The new industrial companies adapted the customs of existing large organizations like the military and the civil service, and the resulting hybrid worked well. "Captains of industry" issued orders to "armies of workers," and everyone knew what they were supposed to do.

Startups seem to go more against the grain, socially. It's hard for them to flourish

in societies that value hierarchy and stability, just as it was hard for industrialization to flourish in societies ruled by people who stole at will from the merchant class. But there were already a handful of countries past that stage when the Industrial Revolution happened. There do not seem to be that many ready this time.

## Notes

[1] One of the bizarre consequences of this model was that the usual way to make more money was to become a manager. This is one of the things startups fix.

[2] There are a lot of reasons American car companies have been doing so much worse than Japanese car companies, but at least one of them is a cause for optimism: American graduates have more options.

[3] It's possible that companies will one day be able to grow big in revenues without growing big in people, but we are not very far along that trend yet.

[4] Lecuyer, Christophe, *Making Silicon Valley*, MIT Press, 2006.

**Thanks** to Trevor Blackwell, Paul Buchheit, Jessica Livingston, and Robert Morris for reading drafts of this.

# Could VC be a Casualty of the Recession?

December 2008

*(I originally wrote this at the request of a company producing a report about entrepreneurship. Unfortunately after reading it they decided it was too controversial to include.)*

VC funding will probably dry up somewhat during the present recession, like it usually does in bad times. But this time the result may be different. This time the number of new startups may not decrease. And that could be dangerous for VCs.

When VC funding dried up after the Internet Bubble, startups dried up too. There were not a lot of new startups being founded in 2003. But startups aren't tied to VC the way they were 10 years ago. It's now possible for VCs and startups to diverge. And if they do, they may not reconverge once the economy gets better.

The reason startups no longer depend so much on VCs is one that everyone in the startup business knows by now: it has gotten much cheaper to start a startup. There are four main reasons: Moore's law has made hardware cheap; open source has made software free; the web has made marketing and distribution free; and more powerful programming languages mean development teams can be smaller. These changes have pushed the cost of starting a startup down into the noise. In a lot of startups—probably most startups funded by Y Combinator—the biggest expense is simply the founders' living expenses. We've had startups that were profitable on revenues of \$3000 a month.

\$3000 is insignificant as revenues go. Why should anyone care about a startup making \$3000 a month? Because, although insignificant as *revenue*, this amount of money can change a startup's *funding* situation completely.

Someone running a startup is always calculating in the back of their mind how much "runway" they have—how long they have till the money in the bank runs out and they either have to be profitable, raise more money, or go out of business. Once you cross the threshold of profitability, however low, your runway becomes infinite. It's a qualitative change, like the stars turning into lines and disappearing when the Enterprise accelerates to warp speed. Once you're profitable you don't need investors' money. And because Internet startups have become so cheap to

run, the threshold of profitability can be trivially low. Which means many Internet startups don't need VC-scale investments anymore. For many startups, VC funding has, in the language of VCs, gone from a must-have to a nice-to-have.

This change happened while no one was looking, and its effects have been largely masked so far. It was during the trough after the Internet Bubble that it became trivially cheap to start a startup, but few realized it because startups were so out of fashion. When startups came back into fashion, around 2005, investors were starting to write checks again. And while founders may not have needed VC money the way they used to, they were willing to take it if offered—partly because there was a tradition of startups taking VC money, and partly because startups, like dogs, tend to eat when given the opportunity. As long as VCs were writing checks, founders were never forced to explore the limits of how little they needed them. There were a few startups who hit these limits accidentally because of their unusual circumstances—most famously 37signals, which hit the limit because they crossed into startup land from the other direction: they started as a consulting firm, so they had revenue before they had a product.

Vcs and founders are like two components that used to be bolted together. Around 2000 the bolt was removed. Because the components have so far been subjected to the same forces, they still seem to be joined together, but really one is just resting on the other. A sharp impact would make them fly apart. And the present recession could be that impact.

Because of Y Combinator's position at the extreme end of the spectrum, we'd be the first to see signs of a separation between founders and investors, and we are in fact seeing it. For example, though the stock market crash does seem to have made investors more cautious, it doesn't seem to have had any effect on the number of people who want to start startups. We take applications for funding every 6 months. Applications for the current funding cycle closed on October 17, well after the markets tanked, and even so we got a record number, up 40% from the same cycle a year before.

Maybe things will be different a year from now, if the economy continues to get worse, but so far there is zero slackening of interest among potential founders. That's different from the way things felt in 2001. Then there was a widespread feeling among potential founders that startups were over, and that one should just go to grad school. That isn't happening this time, and part of the reason is that even in a bad economy it's not that hard to build something that makes \$3000 a month. If investors stop writing checks, who cares?

We also see signs of a divergence between founders and investors in the attitudes of existing startups we've funded. I was talking to one recently that had a round fall through at the last minute over the sort of trifle that breaks deals when investors feel they have the upper hand—over an uncertainty about whether the founders had correctly filed their 83(b) forms, if you can believe that. And yet this startup is obviously going to succeed: their traffic and revenue graphs look like a jet taking off. So I asked them if they wanted me to introduce them to more

investors. To my surprise, they said no—that they'd just spent four months dealing with investors, and they were actually a lot happier now that they didn't have to. There was a friend they wanted to hire with the investor money, and now they'd have to postpone that. But otherwise they felt they had enough in the bank to make it to profitability. To make sure, they were moving to a cheaper apartment. And in this economy I bet they got a good deal on it.

I've detected this "investors aren't worth the trouble" vibe from several YC founders I've talked to recently. At least one startup from the most recent (summer) cycle may not even raise angel money, let alone VC. [Ticketstumbler](#) made it to profitability on Y Combinator's \$15,000 investment and they hope not to need more. This surprised even us. Although YC is based on the idea of it being cheap to start a startup, we never anticipated that founders would grow successful startups on nothing more than YC funding.

If founders decide VCs aren't worth the trouble, that could be bad for VCs. When the economy bounces back in a few years and they're ready to write checks again, they may find that founders have moved on.

There is a founder community just as there's a VC community. They all know one another, and techniques spread rapidly between them. If one tries a new programming language or a new hosting provider and gets good results, 6 months later half of them are using it. And the same is true for funding. The current generation of founders want to raise money from VCs, and Sequoia specifically, because Larry and Sergey took money from VCs, and Sequoia specifically. Imagine what it would do to the VC business if the next hot company didn't take VC at all.

VCs think they're playing a zero sum game. In fact, it's not even that. If you lose a deal to Benchmark, you lose that deal, but VC as an industry still wins. If you lose a deal to None, all VCs lose.

This recession may be different from the one after the Internet Bubble. This time founders may keep starting startups. And if they do, VCs will have to keep writing checks, or they could become irrelevant.

**Thanks** to Sam Altman, Trevor Blackwell, David Hornik, Jessica Livingston, Robert Morris, and Fred Wilson for reading drafts of this.

[Russian Translation](#)



# After Credentials



December 2008

A few months ago I read a *New York Times* article on South Korean cram schools that said

Admission to the right university can make or break an ambitious young South Korean.

A parent added:

"In our country, college entrance exams determine 70 to 80 percent of a person's future."

It was striking how old fashioned this sounded. And yet when I was in high school it wouldn't have seemed too far off as a description of the US. Which means things must have been changing here.

The course of people's lives in the US now seems to be determined less by credentials and more by performance than it was 25 years ago. Where you go to college still matters, but not like it used to.

What happened?

---

Judging people by their academic credentials was in its time an advance. The practice seems to have begun in China, where starting in 587 candidates for the imperial civil service had to take an exam on classical literature. [\[1\]](#) It was also a test of wealth, because the knowledge it tested was so specialized that passing required years of expensive training. But though wealth was a necessary condition for passing, it was not a sufficient one. By the standards of the rest of the world in



587, the Chinese system was very enlightened. Europeans didn't introduce formal civil service exams till the nineteenth century, and even then they seem to have been influenced by the Chinese example.

Before credentials, government positions were obtained mainly by family influence, if not outright bribery. It was a great step forward to judge people by their performance on a test. But by no means a perfect solution. When you judge people that way, you tend to get cram schools—which they did in Ming China and nineteenth century England just as much as in present day South Korea.

What cram schools are, in effect, is leaks in a seal. The use of credentials was an attempt to seal off the direct transmission of power between generations, and cram schools represent that power finding holes in the seal. Cram schools turn wealth in one generation into credentials in the next.

It's hard to beat this phenomenon, because the schools adjust to suit whatever the tests measure. When the tests are narrow and predictable, you get cram schools on the classic model, like those that prepared candidates for Sandhurst (the British West Point) or the classes American students take now to improve their SAT scores. But as the tests get broader, the schools do too. Preparing a candidate for the Chinese imperial civil service exams took years, as prep school does today. But the *raison d'être* of all these institutions has been the same: to beat the system.

[2]

---

History suggests that, all other things being equal, a society prospers in proportion to its ability to prevent parents from influencing their children's success directly. It's a fine thing for parents to help their children indirectly—for example, by helping them to become smarter or more disciplined, which then makes them more successful. The problem comes when parents use direct methods: when they are able to use their own wealth or power as a substitute for their children's qualities.

Parents will tend to do this when they can. Parents will die for their kids, so it's not surprising to find they'll also push their scruples to the limits for them. Especially if other parents are doing it.

Sealing off this force has a double advantage. Not only does a society get "the best man for the job," but parents' ambitions are diverted from direct methods to indirect ones—to actually trying to raise their kids well.

But we should expect it to be very hard to contain parents' efforts to obtain an unfair advantage for their kids. We're dealing with one of the most powerful forces in human nature. We shouldn't expect naive solutions to work, any more than we'd expect naive solutions for keeping heroin out of a prison to work.

---

The obvious way to solve the problem is to make credentials better. If the tests a society uses are currently hackable, we can study the way people beat them and try to plug the holes. You can use the cram schools to show you where most of the holes are. They also tell you when you're succeeding in fixing them: when cram schools become less popular.

A more general solution would be to push for increased transparency, especially at critical social bottlenecks like college admissions. In the US this process still shows many outward signs of corruption. For example, legacy admissions. The official story is that legacy status doesn't carry much weight, because all it does is break ties: applicants are bucketed by ability, and legacy status is only used to decide between the applicants in the bucket that straddles the cutoff. But what this means is that a university can make legacy status have as much or as little weight as they want, by adjusting the size of the bucket that straddles the cutoff.

By gradually chipping away at the abuse of credentials, you could probably make them more airtight. But what a long fight it would be. Especially when the institutions administering the tests don't really want them to be airtight.

---

Fortunately there's a better way to prevent the direct transmission of power between generations. Instead of trying to make credentials harder to hack, we can also make them matter less.

Let's think about what credentials are for. What they are, functionally, is a way of predicting performance. If you could measure actual performance, you wouldn't need them.

So why did they even evolve? Why haven't we just been measuring actual performance? Think about where credentialism first appeared: in selecting candidates for large organizations. Individual performance is hard to measure in large organizations, and the harder performance is to measure, the more important it is to predict it. If an organization could immediately and cheaply measure the performance of recruits, they wouldn't need to examine their credentials. They could take everyone and keep just the good ones.

Large organizations can't do this. But a bunch of small organizations in a market can come close. A market takes every organization and keeps just the good ones. As organizations get smaller, this approaches taking every person and keeping just the good ones. So all other things being equal, a society consisting of more, smaller organizations will care less about credentials.

---

That's what's been happening in the US. That's why those quotes from Korea sound so old fashioned. They're talking about an economy like America's a few decades ago, dominated by a few big companies. The route for the ambitious in that sort of environment is to join one and climb to the top. Credentials matter a lot then. In the culture of a large organization, an elite pedigree becomes a self-fulfilling prophecy.

This doesn't work in small companies. Even if your colleagues were impressed by your credentials, they'd soon be parted from you if your performance didn't match, because the company would go out of business and the people would be dispersed.

In a world of small companies, performance is all anyone cares about. People hiring for a startup don't care whether you've even graduated from college, let alone which one. All they care about is what you can do. Which is in fact all that should matter, even in a large organization. The reason credentials have such prestige is that for so long the large organizations in a society tended to be the most powerful. But in the US at least they don't have the monopoly on power they once did, precisely because they can't measure (and thus reward) individual performance. Why spend twenty years climbing the corporate ladder when you can get rewarded directly by the market?

I realize I see a more exaggerated version of the change than most other people. As a partner at an early stage venture funding firm, I'm like a jumpmaster shoving people out of the old world of credentials and into the new one of performance. I'm an agent of the change I'm seeing. But I don't think I'm imagining it. It was not so easy 25 years ago for an ambitious person to choose to be judged directly by the market. You had to go through bosses, and they were influenced by where you'd been to college.

---

What made it possible for small organizations to succeed in America? I'm still not entirely sure. Startups are certainly a large part of it. Small organizations can develop new ideas faster than large ones, and new ideas are increasingly valuable.

But I don't think startups account for all the shift from credentials to measurement. My friend Julian Weber told me that when he went to work for a New York law firm in the 1950s they paid associates far less than firms do today. Law firms then made no pretense of paying people according to the value of the work they'd done. Pay was based on seniority. The younger employees were paying their dues. They'd be rewarded later.

The same principle prevailed at industrial companies. When my father was working

at Westinghouse in the 1970s, he had people working for him who made more than he did, because they'd been there longer.

Now companies increasingly have to pay employees market price for the work they do. One reason is that employees no longer trust companies to deliver [deferred rewards](#): why work to accumulate deferred rewards at a company that might go bankrupt, or be taken over and have all its implicit obligations wiped out? The other is that some companies broke ranks and started to pay young employees large amounts. This was particularly true in consulting, law, and finance, where it led to the phenomenon of yuppies. The word is rarely used today because it's no longer surprising to see a 25 year old with money, but in 1985 the sight of a 25 year old *professional* able to afford a new BMW was so novel that it called forth a new word.

The classic yuppie worked for a small organization. He didn't work for General Widget, but for the law firm that handled General Widget's acquisitions or the investment bank that floated their bond issues.

Startups and yuppies entered the American conceptual vocabulary roughly simultaneously in the late 1970s and early 1980s. I don't think there was a causal connection. Startups happened because technology started to change so fast that big companies could no longer keep a lid on the smaller ones. I don't think the rise of yuppies was inspired by it; it seems more as if there was a change in the social conventions (and perhaps the laws) governing the way big companies worked. But the two phenomena rapidly fused to produce a principle that now seems obvious: paying energetic young people market rates, and getting correspondingly high performance from them.

At about the same time the US economy rocketed out of the doldrums that had afflicted it for most of the 1970s. Was there a connection? I don't know enough to say, but it felt like it at the time. There was a lot of energy released.

---

Countries worried about their competitiveness are right to be concerned about the number of startups started within them. But they would do even better to examine the underlying principle. Do they let energetic young people get paid market rate for the work they do? The young are the test, because when people aren't rewarded according to performance, they're invariably rewarded according to seniority instead.

All it takes is a few beachheads in your economy that pay for performance. Measurement spreads like heat. If one part of a society is better at measurement than others, it tends to push the others to do better. If people who are young but smart and driven can make more by starting their own companies than by working for existing ones, the existing companies are forced to pay more to keep them. So market rates gradually permeate every organization, even the government. [3]

The measurement of performance will tend to push even the organizations issuing credentials into line. When we were kids I used to annoy my sister by ordering her to do things I knew she was about to do anyway. As credentials are superseded by performance, a similar role is the best former gatekeepers can hope for. Once credential granting institutions are no longer in the self-fulfilling prophecy business, they'll have to work harder to predict the future.

---

Credentials are a step beyond bribery and influence. But they're not the final step. There's an even better way to block the transmission of power between generations: to encourage the trend toward an economy made of more, smaller units. Then you can measure what credentials merely predict.

No one likes the transmission of power between generations—not the left or the right. But the market forces favored by the right turn out to be a better way of preventing it than the credentials the left are forced to fall back on.

The era of credentials began to end when the power of large organizations [peaked](#) in the late twentieth century. Now we seem to be entering a new era based on measurement. The reason the new model has advanced so rapidly is that it works so much better. It shows no sign of slowing.

## Notes

[1] Miyazaki, Ichisada (Conrad Schirokauer trans.), *China's Examination Hell: The Civil Service Examinations of Imperial China*, Yale University Press, 1981.

Scribes in ancient Egypt took exams, but they were more the type of proficiency test any apprentice might have to pass.

[2] When I say the *raison d'être* of prep schools is to get kids into better colleges, I mean this in the narrowest sense. I'm not saying that's all prep schools do, just that if they had zero effect on college admissions there would be far less demand for them.

[3] Progressive tax rates will tend to damp this effect, however, by decreasing the difference between good and bad measurers.

**Thanks** to Trevor Blackwell, Sarah Harlin, Jessica Livingston, and David Sloo for reading drafts of this.

# Keep Your Identity Small

February 2009

I finally realized today why politics and religion yield such uniquely useless discussions.

As a rule, any mention of religion on an online forum degenerates into a religious argument. Why? Why does this happen with religion and not with Javascript or baking or other topics people talk about on forums?

What's different about religion is that people don't feel they need to have any particular expertise to have opinions about it. All they need is strongly held beliefs, and anyone can have those. No thread about Javascript will grow as fast as one about religion, because people feel they have to be over some threshold of expertise to post comments about that. But on religion everyone's an expert.

Then it struck me: this is the problem with politics too. Politics, like religion, is a topic where there's no threshold of expertise for expressing an opinion. All you need is strong convictions.

Do religion and politics have something in common that explains this similarity? One possible explanation is that they deal with questions that have no definite answers, so there's no back pressure on people's opinions. Since no one can be proven wrong, every opinion is equally valid, and sensing this, everyone lets fly with theirs.

But this isn't true. There are certainly some political questions that have definite answers, like how much a new government policy will cost. But the more precise political questions suffer the same fate as the vaguer ones.

I think what religion and politics have in common is that they become part of people's identity, and people can never have a fruitful argument about something that's part of their identity. By definition they're partisan.

Which topics engage people's identity depends on the people, not the topic. For example, a discussion about a battle that included citizens of one or more of the countries involved would probably degenerate into a political argument. But a discussion today about a battle that took place in the Bronze Age probably wouldn't. No one would know what side to be on. So it's not politics that's the

source of the trouble, but identity. When people say a discussion has degenerated into a religious war, what they really mean is that it has started to be driven mostly by people's identities. [1]

Because the point at which this happens depends on the people rather than the topic, it's a mistake to conclude that because a question tends to provoke religious wars, it must have no answer. For example, the question of the relative merits of programming languages often degenerates into a religious war, because so many programmers identify as X programmers or Y programmers. This sometimes leads people to conclude the question must be unanswerable—that all languages are equally good. Obviously that's false: anything else people make can be well or badly designed; why should this be uniquely impossible for programming languages? And indeed, you can have a fruitful discussion about the relative merits of programming languages, so long as you exclude people who respond from identity.

More generally, you can have a fruitful discussion about a topic only if it doesn't engage the identities of any of the participants. What makes politics and religion such minefields is that they engage so many people's identities. But you could in principle have a useful conversation about them with some people. And there are other topics that might seem harmless, like the relative merits of Ford and Chevy pickup trucks, that you couldn't safely talk about with [others](#).

The most intriguing thing about this theory, if it's right, is that it explains not merely which kinds of discussions to avoid, but how to have better ideas. If people can't think clearly about anything that has become part of their identity, then all other things being equal, the best plan is to let as few things into your identity as possible. [2]

Most people reading this will already be fairly tolerant. But there is a step beyond thinking of yourself as x but tolerating y: not even to consider yourself an x. The more labels you have for yourself, the dumber they make you.

## Notes

[1] When that happens, it tends to happen fast, like a core going critical. The threshold for participating goes down to zero, which brings in more people. And they tend to say incendiary things, which draw more and angrier counterarguments.

[2] There may be some things it's a net win to include in your identity. For example, being a scientist. But arguably that is more of a placeholder than an



actual label—like putting NMI on a form that asks for your middle initial—because it doesn't commit you to believing anything in particular. A scientist isn't committed to believing in natural selection in the same way a biblical literalist is committed to rejecting it. All he's committed to is following the evidence wherever it leads.

Considering yourself a scientist is equivalent to putting a sign in a cupboard saying "this cupboard must be kept empty." Yes, strictly speaking, you're putting something in the cupboard, but not in the ordinary sense.

**Thanks** to Sam Altman, Trevor Blackwell, Paul Buchheit, and Robert Morris for reading drafts of this.

[Russian Translation](#)

[Portuguese Translation](#)

[Romanian Translation](#)

# Startups in 13 Sentences

Watch how this essay was [written](#).

February 2009

One of the things I always tell startups is a principle I learned from Paul Buchheit: it's better to make a few people really happy than to make a lot of people semi-happy. I was saying recently to a reporter that if I could only tell startups 10 things, this would be one of them. Then I thought: what would the other 9 be?

When I made the list there turned out to be 13:

## **1. Pick good cofounders.**

Cofounders are for a startup what location is for real estate. You can change anything about a house except where it is. In a startup you can change your idea easily, but changing your cofounders is hard. [\[1\]](#) And the success of a startup is almost always a function of its founders.

## **2. Launch fast.**

The reason to launch fast is not so much that it's critical to get your product to market early, but that you haven't really started working on it till you've launched. Launching teaches you what you should have been building. Till you know that you're wasting your time. So the main value of whatever you launch with is as a pretext for engaging users.

## **3. Let your idea evolve.**

This is the second half of launching fast. Launch fast and iterate. It's a big mistake to treat a startup as if it were merely a matter of implementing some brilliant initial idea. As in an essay, most of the ideas appear in the implementing.

## **4. Understand your users.**

You can envision the wealth created by a startup as a rectangle, where one side is the number of users and the other is how much you improve their lives. [\[2\]](#) The second dimension is the one you have most control over. And indeed, the growth in the first will be driven by how well you do in the second. As in science, the hard part is not answering questions but asking them: the hard part is seeing something new that users lack. The better you understand them the better the odds of doing that. That's why so many successful startups make something the

founders needed.

## **5. Better to make a few users love you than a lot ambivalent.**

Ideally you want to make large numbers of users love you, but you can't expect to hit that right away. Initially you have to choose between satisfying all the needs of a subset of potential users, or satisfying a subset of the needs of all potential users. Take the first. It's easier to expand userwise than satisfactionwise. And perhaps more importantly, it's harder to lie to yourself. If you think you're 85% of the way to a great product, how do you know it's not 70%? Or 10%? Whereas it's easy to know how many users you have.

## **6. Offer surprisingly good customer service.**

Customers are used to being maltreated. Most of the companies they deal with are quasi-monopolies that get away with atrocious customer service. Your own ideas about what's possible have been unconsciously lowered by such experiences. Try making your customer service not merely good, but [surprisingly good](#). Go out of your way to make people happy. They'll be overwhelmed; you'll see. In the earliest stages of a startup, it pays to offer customer service on a level that wouldn't scale, because it's a way of learning about your users.

## **7. You make what you measure.**

I learned this one from Joe Kraus. [3] Merely measuring something has an uncanny tendency to improve it. If you want to make your user numbers go up, put a big piece of paper on your wall and every day plot the number of users. You'll be delighted when it goes up and disappointed when it goes down. Pretty soon you'll start noticing what makes the number go up, and you'll start to do more of that. Corollary: be careful what you measure.

## **8. Spend little.**

I can't emphasize enough how important it is for a startup to be cheap. Most startups fail before they make something people want, and the most common form of failure is running out of money. So being cheap is (almost) interchangeable with iterating rapidly. [4] But it's more than that. A culture of cheapness keeps companies young in something like the way exercise keeps people young.

## **9. Get ramen profitable.**

"Ramen profitable" means a startup makes just enough to pay the founders' living expenses. It's not rapid prototyping for business models (though it can be), but more a way of hacking the investment process. Once you cross over into ramen profitable, it completely changes your relationship with investors. It's also great for morale.

## **10. Avoid distractions.**

Nothing kills startups like distractions. The worst type are those that pay money: day jobs, consulting, profitable side-projects. The startup may have more long-term potential, but you'll always interrupt working on it to answer calls from people paying you now. Paradoxically, [fundraising](#) is this type of distraction, so try to minimize that too.

## **11. Don't get demoralized.**

Though the immediate cause of death in a startup tends to be running out of money, the underlying cause is usually lack of focus. Either the company is run by stupid people (which can't be fixed with advice) or the people are smart but got demoralized. Starting a startup is a huge moral weight. Understand this and make a conscious effort not to be ground down by it, just as you'd be careful to bend at the knees when picking up a heavy box.

## **12. Don't give up.**

Even if you get demoralized, [don't give up](#). You can get surprisingly far by just not giving up. This isn't true in all fields. There are a lot of people who couldn't become good mathematicians no matter how long they persisted. But startups aren't like that. Sheer effort is usually enough, so long as you keep morphing your idea.

## **13. Deals fall through.**

One of the most useful skills we learned from Viaweb was not getting our hopes up. We probably had 20 deals of various types fall through. After the first 10 or so we learned to treat deals as background processes that we should ignore till they terminated. It's very dangerous to morale to start to depend on deals closing, not just because they so often don't, but because it makes them less likely to.

Having gotten it down to 13 sentences, I asked myself which I'd choose if I could only keep one.

Understand your users. That's the key. The essential task in a startup is to create wealth; the dimension of wealth you have most control over is how much you improve users' lives; and the hardest part of that is knowing what to make for them. Once you know what to make, it's mere effort to make it, and most decent hackers are capable of that.

Understanding your users is part of half the principles in this list. That's the reason to launch early, to understand your users. Evolving your idea is the embodiment of understanding your users. Understanding your users well will tend to push you toward making something that makes a few people deeply happy. The most important reason for having surprisingly good customer service is that it helps you understand your users. And understanding your users will even ensure your morale, because when everything else is collapsing around you, having just ten users who love you will keep you going.

## **Notes**

[1] Strictly speaking it's impossible without a time machine.

[2] In practice it's more like a ragged comb.

[3] Joe thinks one of the founders of Hewlett Packard said it first, but he doesn't remember which.

[4] They'd be interchangeable if markets stood still. Since they don't, working twice as fast is better than having twice as much time.

[Turkish Translation](#)

[Spanish Translation](#)

[Bulgarian Translation](#)

[Japanese Translation](#)

[Persian Translation](#)

# What I've Learned from Hacker News

February 2009

Hacker News was two years old last week. Initially it was supposed to be a side project—an application to sharpen Arc on, and a place for current and future Y Combinator founders to exchange news. It's grown bigger and taken up more time than I expected, but I don't regret that because I've learned so much from working on it.

## Growth

When we launched in February 2007, weekday traffic was around 1600 daily uniques. It's since [grown](#) to around 22,000. This growth rate is a bit higher than I'd like. I'd like the site to grow, since a site that isn't growing at least slowly is probably dead. But I wouldn't want it to grow as large as Digg or Reddit—mainly because that would dilute the character of the site, but also because I don't want to spend all my time dealing with scaling.

I already have problems enough with that. Remember, the original motivation for HN was to test a new programming language, and moreover one that's focused on experimenting with language design, not performance. Every time the site gets slow, I fortify myself by recalling McIlroy and Bentley's famous quote

The key to performance is elegance, not battalions of special cases.

and look for the bottleneck I can remove with least code. So far I've been able to keep up, in the sense that performance has remained consistently mediocre despite 14x growth. I don't know what I'll do next, but I'll probably think of something.

This is my attitude to the site generally. Hacker News is an experiment, and an experiment in a very young field. Sites of this type are only a few years old. Internet conversation generally is only a few decades old. So we've probably only discovered a fraction of what we eventually will.

That's why I'm so optimistic about HN. When a technology is this young, the existing solutions are usually terrible; which means it must be possible to do much better; which means many problems that seem insoluble aren't. Including, I hope, the problem that has afflicted so many previous communities: being ruined by growth.

## Dilution

Users have worried about that since the site was a few months old. So far these alarms have been false, but they may not always be. Dilution is a hard problem. But probably soluble; it doesn't mean much that open conversations have "always" been destroyed by growth when "always" equals 20 instances.

But it's important to remember we're trying to solve a new problem, because that means we're going to have to try new things, most of which probably won't work. A couple weeks ago I tried displaying the names of users with the highest average comment scores in orange. [1] That was a mistake. Suddenly a culture that had been more or less united was divided into haves and have-nots. I didn't realize how united the culture had been till I saw it divided. It was painful to watch. [2]

So orange usernames won't be back. (Sorry about that.) But there will be other equally broken-seeming ideas in the future, and the ones that turn out to work will probably seem just as broken as those that don't.

Probably the most important thing I've learned about dilution is that it's measured more in behavior than users. It's bad behavior you want to keep out more than bad people. User behavior turns out to be surprisingly malleable. If people are [expected](#) to behave well, they tend to; and vice versa.

Though of course forbidding bad behavior does tend to keep away bad people, because they feel uncomfortably constrained in a place where they have to behave well. But this way of keeping them out is gentler and probably also more effective than overt barriers.

It's pretty clear now that the broken windows theory applies to community sites as well. The theory is that minor forms of bad behavior encourage worse ones: that a neighborhood with lots of graffiti and broken windows becomes one where robberies occur. I was living in New York when Giuliani introduced the reforms that made the broken windows theory famous, and the transformation was miraculous. And I was a Reddit user when the opposite happened there, and the transformation was equally dramatic.

I'm not criticizing Steve and Alexis. What happened to Reddit didn't happen out of neglect. From the start they had a policy of censoring nothing except spam. Plus Reddit had different goals from Hacker News. Reddit was a startup, not a side project; its goal was to grow as fast as possible. Combine rapid growth and zero censorship, and the result is a free for all. But I don't think they'd do much differently if they were doing it again. Measured by traffic, Reddit is much more successful than Hacker News.

But what happened to Reddit won't inevitably happen to HN. There are several local maxima. There can be places that are free for alls and places that are more thoughtful, just as there are in the real world; and people will behave differently

depending on which they're in, just as they do in the real world.

I've observed this in the wild. I've seen people cross-posting on Reddit and Hacker News who actually took the trouble to write two versions, a flame for Reddit and a more subdued version for HN.

## **Submissions**

There are two major types of problems a site like Hacker News needs to avoid: bad stories and bad comments. So far the danger of bad stories seems smaller. The stories on the frontpage now are still roughly the ones that would have been there when HN started.

I once thought I'd have to weight votes to keep crap off the frontpage, but I haven't had to yet. I wouldn't have predicted the frontpage would hold up so well, and I'm not sure why it has. Perhaps only the more thoughtful users care enough to submit and upvote links, so the marginal cost of one random new user approaches zero. Or perhaps the frontpage protects itself, by advertising what type of submission is expected.

The most dangerous thing for the frontpage is stuff that's too easy to upvote. If someone proves a new theorem, it takes some work by the reader to decide whether or not to upvote it. An amusing cartoon takes less. A rant with a rallying cry as the title takes zero, because people vote it up without even reading it.

Hence what I call the Fluff Principle: on a user-voted news site, the links that are easiest to judge will take over unless you take specific measures to prevent it.

Hacker News has two kinds of protections against fluff. The most common types of fluff links are banned as off-topic. Pictures of kittens, political diatribes, and so on are explicitly banned. This keeps out most fluff, but not all of it. Some links are both fluff, in the sense of being very short, and also on topic.

There's no single solution to that. If a link is just an empty rant, editors will sometimes kill it even if it's on topic in the sense of being about hacking, because it's not on topic by the real standard, which is to engage one's intellectual curiosity. If the posts on a site are characteristically of this type I sometimes ban it, which means new stuff at that url is auto-killed. If a post has a linkbait title, editors sometimes rephrase it to be more matter-of-fact. This is especially necessary with links whose titles are rallying cries, because otherwise they become implicit "vote up if you believe such-and-such" posts, which are the most extreme form of fluff.

The techniques for dealing with links have to evolve, because the links do. The existence of aggregators has already affected what they aggregate. Writers now deliberately write things to draw traffic from aggregators—sometimes even specific ones. (No, the irony of this statement is not lost on me.) Then there are the more sinister mutations, like linkjacking—posting a paraphrase of someone else's article and submitting that instead of the original. These can get a lot of upvotes, because



a lot of what's good in an article often survives; indeed, the closer the paraphrase is to plagiarism, the more survives. [3]

I think it's important that a site that kills submissions provide a way for users to see what got killed if they want to. That keeps editors honest, and just as importantly, makes users confident they'd know if the editors stopped being honest. HN users can do this by flipping a switch called showdead in their profile. [4]

## Comments

Bad comments seem to be a harder problem than bad submissions. While the quality of links on the frontpage of HN hasn't changed much, the quality of the median comment may have decreased somewhat.

There are two main kinds of badness in comments: meanness and stupidity. There is a lot of overlap between the two—mean comments are disproportionately likely also to be dumb—but the strategies for dealing with them are different. Meanness is easier to control. You can have rules saying one shouldn't be mean, and if you enforce them it seems possible to keep a lid on meanness.

Keeping a lid on stupidity is harder, perhaps because stupidity is not so easily distinguishable. Mean people are more likely to know they're being mean than stupid people are to know they're being stupid.

The most dangerous form of stupid comment is not the long but mistaken argument, but the dumb joke. Long but mistaken arguments are actually quite rare. There is a strong correlation between comment quality and length; if you wanted to compare the quality of comments on community sites, average length would be a good predictor. Probably the cause is human nature rather than anything specific to comment threads. Probably it's simply that stupidity more often takes the form of having few ideas than wrong ones.

Whatever the cause, stupid comments tend to be short. And since it's hard to write a short comment that's distinguished for the amount of information it conveys, people try to distinguish them instead by being funny. The most tempting format for stupid comments is the supposedly witty put-down, probably because put-downs are the easiest form of humor. [5] So one advantage of forbidding meanness is that it also cuts down on these.

Bad comments are like kudzu: they take over rapidly. Comments have much more effect on new comments than submissions have on new submissions. If someone submits a lame article, the other submissions don't all become lame. But if someone posts a stupid comment on a thread, that sets the tone for the region around it. People reply to dumb jokes with dumb jokes.

Maybe the solution is to add a delay before people can respond to a comment, and make the length of the delay inversely proportional to some prediction of its

quality. Then dumb threads would grow slower. [6]

## People

I notice most of the techniques I've described are conservative: they're aimed at preserving the character of the site rather than enhancing it. I don't think that's a bias of mine. It's due to the shape of the problem. Hacker News had the good fortune to start out good, so in this case it's literally a matter of preservation. But I think this principle would also apply to sites with different origins.

The good things in a community site come from people more than technology; it's mainly in the prevention of bad things that technology comes into play. Technology certainly can enhance discussion. Nested comments do, for example. But I'd rather use a site with primitive features and smart, nice users than a more advanced one whose users were idiots or [trolls](#).

So the most important thing a community site can do is attract the kind of people it wants. A site trying to be as big as possible wants to attract everyone. But a site aiming at a particular subset of users has to attract just those—and just as importantly, repel everyone else. I've made a conscious effort to do this on HN. The graphic design is as plain as possible, and the site rules discourage dramatic link titles. The goal is that the only thing to interest someone arriving at HN for the first time should be the ideas expressed there.

The downside of tuning a site to attract certain people is that, to those people, it can be too attractive. I'm all too aware how addictive Hacker News can be. For me, as for many users, it's a kind of virtual town square. When I want to take a break from working, I walk into the square, just as I might into Harvard Square or University Ave in the physical world. [7] But an online square is more dangerous than a physical one. If I spent half the day loitering on University Ave, I'd notice. I have to walk a mile to get there, and sitting in a cafe feels different from working. But visiting an online forum takes just a click, and feels superficially very much like working. You may be wasting your time, but you're not idle. Someone is [wrong](#) on the Internet, and you're fixing the problem.

Hacker News is definitely useful. I've learned a lot from things I've read on HN. I've written several essays that began as comments there. So I wouldn't want the site to go away. But I would like to be sure it's not a net drag on productivity. What a disaster that would be, to attract thousands of smart people to a site that caused them to waste lots of time. I wish I could be 100% sure that's not a description of HN.

I feel like the addictiveness of games and social applications is still a mostly unsolved problem. The situation now is like it was with crack in the 1980s: we've invented terribly addictive new things, and we haven't yet evolved ways to protect ourselves from them. We will eventually, and that's one of the problems I hope to focus on next.

## Notes

[1] I tried ranking users by both average and median comment score, and average (with the high score thrown out) seemed the more accurate predictor of high quality. Median may be the more accurate predictor of low quality though.

[2] Another thing I learned from this experiment is that if you're going to distinguish between people, you better be sure you do it right. This is one problem where rapid prototyping doesn't work.

Indeed, that's the intellectually honest argument for not discriminating between various types of people. The reason not to do it is not that everyone's the same, but that it's bad to do wrong and hard to do right.

[3] When I catch egregiously linkjacked posts I replace the url with that of whatever they copied. Sites that habitually linkjack get banned.

[4] Digg is notorious for its lack of transparency. The root of the problem is not that the guys running Digg are especially sneaky, but that they use the wrong algorithm for generating their frontpage. Instead of bubbling up from the bottom as they get more votes, as on Reddit, stories start at the top and get pushed down by new arrivals.

The reason for the difference is that Digg is derived from Slashdot, while Reddit is derived from Delicious/popular. Digg is Slashdot with voting instead of editors, and Reddit is Delicious/popular with voting instead of bookmarking. (You can still see fossils of their origins in their graphic design.)

Digg's algorithm is very vulnerable to gaming, because any story that makes it onto the frontpage is the new top story. Which in turn forces Digg to respond with extreme countermeasures. A lot of startups have some kind of secret about the subterfuges they had to resort to in the early days, and I suspect Digg's is the extent to which the top stories were de facto chosen by human editors.

[5] The dialog on Beavis and Butthead was composed largely of these, and when I read comments on really bad sites I can hear them in their voices.

[6] I suspect most of the techniques for discouraging stupid comments have yet to be discovered. Xkcd implemented a particularly clever one in its IRC channel: don't allow the same thing twice. Once someone has said "fail," no one can ever say it again. This would penalize short comments especially, because they have less room to avoid collisions in.

Another promising idea is the [stupid filter](#), which is just like a probabilistic spam

filter, but trained on corpora of stupid and non-stupid comments instead.

You may not have to kill bad comments to solve the problem. Comments at the bottom of a long thread are rarely seen, so it may be enough to incorporate a prediction of quality in the comment sorting algorithm.

[7] What makes most suburbs so demoralizing is that there's no center to walk to.

**Thanks** to Justin Kan, Jessica Livingston, Robert Morris, Alexis Ohanian, Emmet Shear, and Fred Wilson for reading drafts of this.



[Comment](#) on this essay.

# Can You Buy a Silicon Valley? Maybe.

February 2009

A lot of cities look at Silicon Valley and ask "How could we make something like that happen here?" The [organic](#) way to do it is to establish a first-rate university in a place where rich people want to live. That's how Silicon Valley happened. But could you shortcut the process by funding startups?

Possibly. Let's consider what it would take.

The first thing to understand is that encouraging startups is a different problem from encouraging startups in a particular city. The latter is much more expensive.

People sometimes think they could improve the startup scene in their town by starting something like [Y Combinator](#) there, but in fact it will have near zero effect. I know because Y Combinator itself had near zero effect on Boston when we were based there half the year. The people we funded came from all over the country (indeed, the world) and afterward they went wherever they could get more funding—which generally meant Silicon Valley.

The seed funding business is not a regional business, because at that stage startups are mobile. They're just a couple founders with laptops. [\[1\]](#)

If you want to encourage startups in a particular city, you have to fund startups that won't leave. There are two ways to do that: have rules preventing them from leaving, or fund them at the point in their life when they naturally take root. The first approach is a mistake, because it becomes a filter for selecting bad startups. If your terms force startups to do things they don't want to, only the desperate ones will take your money.

Good startups will move to another city as a condition of funding. What they won't do is agree not to move the next time they need funding. So the only way to get them to stay is to give them enough that they never need to leave.

---

How much would that take? If you want to keep startups from leaving your town, you have to give them enough that they're not tempted by an offer from Silicon

Valley VCs that requires them to move. A startup would be able to refuse such an offer if they had grown to the point where they were (a) rooted in your town and/or (b) so successful that VCs would fund them even if they didn't move.

How much would it cost to grow a startup to that point? A minimum of several hundred thousand dollars. [Wufoo](#) seem to have rooted themselves in Tampa on \$118k, but they're an extreme case. On average it would take at least half a million.

So if it seems too good to be true to think you could grow a local silicon valley by giving startups \$15-20k each like Y Combinator, that's because it is. To make them stick around you'd have to give them at least 20 times that much.

However, even that is an interesting prospect. Suppose to be on the safe side it would cost a million dollars per startup. If you could get startups to stick to your town for a million apiece, then for a billion dollars you could bring in a thousand startups. That probably wouldn't push you past Silicon Valley itself, but it might get you second place.

For the price of a football stadium, any town that was decent to live in could make itself one of the biggest startup hubs in the world.

What's more, it wouldn't take very long. You could probably do it in five years. During the term of one mayor. And it would get easier over time, because the more startups you had in town, the less it would take to get new ones to move there. By the time you had a thousand startups in town, the VCs wouldn't be trying so hard to get them to move to Silicon Valley; instead they'd be opening local offices. Then you'd really be in good shape. You'd have started a self-sustaining chain reaction like the one that drives the Valley.

---

But now comes the hard part. You have to pick the startups. How do you do that? Picking startups is a rare and valuable skill, and the handful of people who have it are not readily hireable. And this skill is so hard to measure that if a government did try to hire people with it, they'd almost certainly get the wrong ones.

For example, a city could give money to a VC fund to establish a local branch, and let them make the choices. But only a bad VC fund would take that deal. They wouldn't *seem* bad to the city officials. They'd seem very impressive. But they'd be bad at picking startups. That's the characteristic failure mode of VCs. All VCs look impressive to limited partners. The difference between the good ones and the bad ones only becomes visible in the other half of their jobs: choosing and advising startups. [\[2\]](#)

What you really want is a pool of local angel investors—people investing money they made from their own startups. But unfortunately you run into a chicken and

egg problem here. If your city isn't already a startup hub, there won't be people there who got rich from startups. And there is no way I can think of that a city could attract angels from outside. By definition they're rich. There's no incentive that would make them move. [3]

However, a city could select startups by piggybacking on the expertise of investors who weren't local. It would be pretty straightforward to make a list of the most eminent Silicon Valley angels and from that to generate a list of all the startups they'd invested in. If a city offered these companies a million dollars each to move, a lot of the earlier stage ones would probably take it.

Preposterous as this plan sounds, it's probably the most efficient way a city could select good startups.

It would hurt the startups somewhat to be separated from their original investors. On the other hand, the extra million dollars would give them a lot more runway.

---

Would the transplanted startups survive? Quite possibly. The only way to find out would be to try it. It would be a pretty cheap experiment, as civil expenditures go. Pick 30 startups that eminent angels have recently invested in, give them each a million dollars if they'll relocate to your city, and see what happens after a year. If they seem to be thriving, you can try importing startups on a larger scale.

Don't be too legalistic about the conditions under which they're allowed to leave. Just have a gentlemen's agreement.

Don't try to do it on the cheap and pick only 10 for the initial experiment. If you do this on too small a scale you'll just guarantee failure. Startups need to be around other startups. 30 would be enough to feel like a community.

Don't try to make them all work in some renovated warehouse you've made into an "incubator." Real startups prefer to work in their own spaces.

In fact, don't impose any restrictions on the startups at all. Startup founders are mostly [hackers](#), and hackers are much more constrained by gentlemen's agreements than regulations. If they shake your hand on a promise, they'll keep it. But show them a lock and their first thought is how to pick it.

Interestingly, the 30-startup experiment could be done by any sufficiently rich private citizen. And what pressure it would put on the city if it worked. [4]

---

Should the city take stock in return for the money? In principle they're entitled to,

but how would they choose valuations for the startups? You couldn't just give them all the same valuation: that would be too low for some (who'd turn you down) and too high for others (because it might make their next round a "down round"). And since we're assuming we're doing this without being able to pick startups, we also have to assume we can't value them, since that's practically the same thing.

Another reason not to take stock in the startups is that startups are often involved in disreputable things. So are established companies, but they don't get blamed for it. If someone gets murdered by someone they met on Facebook, the press will treat the story as if it were about Facebook. If someone gets murdered by someone they met at a supermarket, the press will just treat it as a story about a murder. So understand that if you invest in startups, they might build things that get used for pornography, or file-sharing, or the expression of unfashionable opinions. You should probably sponsor this project jointly with your political opponents, so they can't use whatever the startups do as a club to beat you with.

It would be too much of a political liability just to give the startups the money, though. So the best plan would be to make it convertible debt, but which didn't convert except in a really big round, like \$20 million.

---

How well this scheme worked would depend on the [city](#). There are some towns, like Portland, that would be easy to turn into startup hubs, and others, like Detroit, where it would really be an uphill battle. So be honest with yourself about the sort of town you have before you try this.

It will be easier in proportion to how much your town resembles San Francisco. Do you have good weather? Do people live downtown, or have they abandoned the center for the suburbs? Would the city be described as "hip" and "tolerant," or as reflecting "traditional values?" Are there good universities nearby? Are there walkable neighborhoods? Would nerds feel at home? If you answered yes to all these questions, you might be able not only to pull off this scheme, but to do it for less than a million per startup.

I realize the chance of any city having the political will to carry out this plan is microscopically small. I just wanted to explore what it would take if one did. How hard would it be to jumpstart a silicon valley? It's fascinating to think this prize might be within the reach of so many cities. So even though they'll all still spend the money on the stadium, at least now someone can ask them: why did you choose to do that instead of becoming a serious rival to Silicon Valley?



[1] What people who start these supposedly local seed firms always find is that (a) their applicants come from all over, not just the local area, and (b) the local startups also apply to the other seed firms. So what ends up happening is that the applicant pool gets partitioned by quality rather than geography.

[2] Interestingly, the bad VCs fail by choosing startups run by people like them—people who are good presenters, but have no real substance. It's a case of the fake leading the fake. And since everyone involved is so plausible, the LPs who invest in these funds have no idea what's happening till they measure their returns.

[3] Not even being a tax haven, I suspect. That makes some rich people move, but not the type who would make good angel investors in startups.

[4] Thanks to Michael Keenan for pointing this out.

**Thanks** to Trevor Blackwell, Jessica Livingston, Robert Morris, and Fred Wilson for reading drafts of this.

# Why TV Lost

March 2009

About twenty years ago people noticed computers and TV were on a collision course and started to speculate about what they'd produce when they converged. We now know the answer: computers. It's clear now that even by using the word "convergence" we were giving TV too much credit. This won't be convergence so much as replacement. People may still watch things they call "TV shows," but they'll watch them mostly on computers.

What decided the contest for computers? Four forces, three of which one could have predicted, and one that would have been harder to.

One predictable cause of victory is that the Internet is an open platform. Anyone can build whatever they want on it, and the market picks the winners. So innovation happens at hacker speeds instead of big company speeds.

The second is Moore's Law, which has worked its usual magic on Internet bandwidth. [\[1\]](#)

The third reason computers won is piracy. Users prefer it not just because it's free, but because it's more convenient. Bittorrent and YouTube have already trained a new generation of viewers that the place to watch shows is on a computer screen. [\[2\]](#)

The somewhat more surprising force was one specific type of innovation: social applications. The average teenage kid has a pretty much infinite capacity for talking to their friends. But they can't physically be with them all the time. When I was in high school the solution was the telephone. Now it's social networks, multiplayer games, and various messaging applications. The way you reach them all is through a computer. [\[3\]](#) Which means every teenage kid (a) wants a computer with an Internet connection, (b) has an incentive to figure out how to use it, and (c) spends countless hours in front of it.

This was the most powerful force of all. This was what made everyone want computers. Nerds got computers because they liked them. Then gamers got them to play games on. But it was connecting to other people that got everyone else: that's what made even grandmas and 14 year old girls want computers.

After decades of running an IV drip right into their audience, people in the entertainment business had understandably come to think of them as rather passive. They thought they'd be able to dictate the way shows reached audiences. But they underestimated the force of their desire to connect with one another.

Facebook killed TV. That is wildly oversimplified, of course, but probably as close to the truth as you can get in three words.

---

The TV networks already seem, grudgingly, to see where things are going, and have responded by putting their stuff, grudgingly, online. But they're still dragging their heels. They still seem to wish people would watch shows on TV instead, just as newspapers that put their stories online still seem to wish people would wait till the next morning and read them printed on paper. They should both just face the fact that the Internet is the primary medium.

They'd be in a better position if they'd done that earlier. When a new medium arises that's powerful enough to make incumbents nervous, then it's probably powerful enough to win, and the best thing they can do is jump in immediately.

Whether they like it or not, big changes are coming, because the Internet dissolves the two cornerstones of broadcast media: synchronicity and locality. On the Internet, you don't have to send everyone the same signal, and you don't have to send it to them from a local source. People will watch what they want when they want it, and group themselves according to whatever shared interest they feel most strongly. Maybe their strongest shared interest will be their physical location, but I'm guessing not. Which means local TV is probably dead. It was an artifact of limitations imposed by old technology. If someone were creating an Internet-based TV company from scratch now, they might have some plan for shows aimed at specific regions, but it wouldn't be a top priority.

Synchronicity and locality are tied together. TV network affiliates care what's on at 10 because that delivers viewers for local news at 11. This connection adds more brittleness than strength, however: people don't watch what's on at 10 because they want to watch the news afterward.

TV networks will fight these trends, because they don't have sufficient flexibility to adapt to them. They're hemmed in by local affiliates in much the same way car companies are hemmed in by dealers and unions. Inevitably, the people running the networks will take the easy route and try to keep the old model running for a couple more years, just as the record labels have done.

A recent article in the *Wall Street Journal* described how TV networks were trying to add more live shows, partly as a way to make viewers watch TV synchronously instead of watching recorded shows when it suited them. Instead of delivering what viewers want, they're trying to force them to change their habits to suit the

networks' obsolete business model. That never works unless you have a monopoly or cartel to enforce it, and even then it only works temporarily.

The other reason networks like live shows is that they're cheaper to produce. There they have the right idea, but they haven't followed it to its conclusion. Live content can be way cheaper than networks realize, and the way to take advantage of dramatic decreases in cost is to [increase volume](#). The networks are prevented from seeing this whole line of reasoning because they still think of themselves as being in the broadcast business—as sending one signal to everyone. [4]

---

[Now](#) would be a good time to start any company that competes with TV networks. That's what a lot of Internet startups are, though they may not have had this as an explicit goal. People only have so many leisure hours a day, and TV is premised on such long sessions (unlike Google, which prides itself on sending users on their way quickly) that anything that takes up their time is competing with it. But in addition to such indirect competitors, I think TV companies will increasingly face direct ones.

Even in cable TV, the long tail was lopped off prematurely by the threshold you had to get over to start a new channel. It will be longer on the Internet, and there will be more mobility within it. In this new world, the existing players will only have the advantages any big company has in its market.

That will change the balance of power between the networks and the people who produce shows. The networks used to be gatekeepers. They distributed your work, and sold advertising on it. Now the people who produce a show can distribute it themselves. The main value networks supply now is ad sales. Which will tend to put them in the position of service providers rather than publishers.

Shows will change even more. On the Internet there's no reason to keep their current format, or even the fact that they have a single format. Indeed, the more interesting sort of convergence that's coming is between shows and games. But on the question of what sort of entertainment gets distributed on the Internet in 20 years, I wouldn't dare to make any predictions, except that things will change a lot. We'll get whatever the most imaginative people can cook up. That's why the Internet won.

## Notes

[1] Thanks to Trevor Blackwell for this point. He adds: "I remember the eyes of phone companies gleaming in the early 90s when they talked about convergence.

They thought most programming would be on demand, and they would implement it and make a lot of money. It didn't work out. They assumed that their local network infrastructure would be critical to do video on-demand, because you couldn't possibly stream it from a few data centers over the internet. At the time (1992) the entire cross-country Internet bandwidth wasn't enough for one video stream. But wide-area bandwidth increased more than they expected and they were beaten by iTunes and Hulu."

[2] Copyright owners tend to focus on the aspect they see of piracy, which is the lost revenue. They therefore think what drives users to do it is the desire to get something for free. But iTunes shows that people will pay for stuff online, if you make it easy. A significant component of piracy is simply that it offers a better user experience.

[3] Or a phone that is actually a computer. I'm not making any predictions about the size of the device that will replace TV, just that it will have a browser and get data via the Internet.

[4] Emmett Shear writes: "I'd argue the long tail for sports may be even larger than the long tail for other kinds of content. Anyone can broadcast a high school football game that will be interesting to 10,000 people or so, even if the quality of production is not so good."

**Thanks** to Sam Altman, Trevor Blackwell, Nancy Cook, Michael Seibel, Emmett Shear, and Fred Wilson for reading drafts of this.

[Japanese Translation](#)

# How to Be an Angel Investor

March 2009

*(This essay is derived from a talk at [AngelConf.](#))*

When we sold our startup in 1998 I thought one day I'd do some angel investing. Seven years later I still hadn't started. I put it off because it seemed mysterious and complicated. It turns out to be easier than I expected, and also more interesting.

The part I thought was hard, the mechanics of investing, really isn't. You give a startup money and they give you stock. You'll probably get either preferred stock, which means stock with extra rights like getting your money back first in a sale, or convertible debt, which means (on paper) you're lending the company money, and the debt converts to stock at the next sufficiently big funding round. [[1](#)]

There are sometimes minor tactical advantages to using one or the other. The paperwork for convertible debt is simpler. But really it doesn't matter much which you use. Don't spend much time worrying about the details of deal terms, especially when you first start angel investing. That's not how you win at this game. When you hear people talking about a successful angel investor, they're not saying "He got a 4x liquidation preference." They're saying "He invested in Google."

That's how you win: by investing in the right startups. That is so much more important than anything else that I worry I'm misleading you by even talking about other things.

## **Mechanics**

Angel investors often syndicate deals, which means they join together to invest on the same terms. In a syndicate there is usually a "lead" investor who negotiates the terms with the startup. But not always: sometimes the startup cobbles together a syndicate of investors who approach them independently, and the startup's lawyer supplies the paperwork.

The easiest way to get started in angel investing is to find a friend who already does it, and try to get included in his syndicates. Then all you have to do is write checks.

Don't feel like you have to join a syndicate, though. It's not that hard to do it yourself. You can just use the standard [series AA](#) documents Wilson Sonsini and Y Combinator published online. You should of course have your lawyer review everything. Both you and the startup should have lawyers. But the lawyers don't have to create the agreement from scratch. [2]

When you negotiate terms with a startup, there are two numbers you care about: how much money you're putting in, and the valuation of the company. The valuation determines how much stock you get. If you put \$50,000 into a company at a pre-money valuation of \$1 million, then the post-money valuation is \$1.05 million, and you get .05/1.05, or 4.76% of the company's stock.

If the company raises more money later, the new investor will take a chunk of the company away from all the existing shareholders just as you did. If in the next round they sell 10% of the company to a new investor, your 4.76% will be reduced to 4.28%.

That's ok. Dilution is normal. What saves you from being mistreated in future rounds, usually, is that you're in the same boat as the founders. They can't dilute you without diluting themselves just as much. And they won't dilute themselves unless they end up [net ahead](#). So in theory, each further round of investment leaves you with a smaller share of an even more valuable company, till after several more rounds you end up with .5% of the company at the point where it IPOs, and you are very happy because your \$50,000 has become \$5 million. [3]

The agreement by which you invest should have provisions that let you contribute to future rounds to maintain your percentage. So it's your choice whether you get diluted. [4] If the company does really well, you eventually will, because eventually the valuations will get so high it's not worth it for you.

How much does an angel invest? That varies enormously, from \$10,000 to hundreds of thousands or in rare cases even millions. The upper bound is obviously the total amount the founders want to raise. The lower bound is 5-10% of the total or \$10,000, whichever is greater. A typical angel round these days might be \$150,000 raised from 5 people.

Valuations don't vary as much. For angel rounds it's rare to see a valuation lower than half a million or higher than 4 or 5 million. 4 million is starting to be VC territory.

How do you decide what valuation to offer? If you're part of a round led by someone else, that problem is solved for you. But what if you're investing by yourself? There's no real answer. There is no rational way to value an early stage startup. The valuation reflects nothing more than the strength of the company's bargaining position. If they really want you, either because they desperately need money, or you're someone who can help them a lot, they'll let you invest at a low valuation. If they don't need you, it will be higher. So guess. The startup may not

have any more idea what the number should be than you do. [5]

Ultimately it doesn't matter much. When angels make a lot of money from a deal, it's not because they invested at a valuation of \$1.5 million instead of \$3 million. It's because the company was really successful.

I can't emphasize that too much. Don't get hung up on mechanics or deal terms. What you should spend your time thinking about is whether the company is good.

(Similarly, founders also should not get hung up on deal terms, but should spend their time thinking about how to make the company good.)

There's a second less obvious component of an angel investment: how much you're expected to help the startup. Like the amount you invest, this can vary a lot. You don't have to do anything if you don't want to; you could simply be a source of money. Or you can become a de facto employee of the company. Just make sure that you and the startup agree in advance about roughly how much you'll do for them.

Really hot companies sometimes have high standards for angels. The ones everyone wants to invest in practically audition investors, and only take money from people who are famous and/or will work hard for them. But don't feel like you have to put in a lot of time or you won't get to invest in any good startups. There is a surprising lack of correlation between how hot a deal a startup is and how well it ends up doing. Lots of hot startups will end up failing, and lots of startups no one likes will end up succeeding. And the latter are so desperate for money that they'll take it from anyone at a low valuation. [6]

## **Picking Winners**

It would be nice to be able to pick those out, wouldn't it? The part of angel investing that has most effect on your returns, picking the right companies, is also the hardest. So you should practically ignore (or more precisely, archive, in the Gmail sense) everything I've told you so far. You may need to refer to it at some point, but it is not the central issue.

The central issue is picking the right startups. What "Make something people want" is for startups, "Pick the right startups" is for investors. Combined they yield "Pick the startups that will make something people want."

How do you do that? It's not as simple as picking startups that are already making something wildly popular. By then it's too late for angels. VCs will already be onto them. As an angel, you have to pick startups before they've got a hit—either because they've made something great but users don't realize it yet, like Google early on, or because they're still an iteration or two away from the big hit, like Paypal when they were making software for transferring money between PDAs.

To be a good angel investor, you have to be a good judge of potential. That's what



it comes down to. VCs can be fast followers. Most of them don't try to predict what will win. They just try to notice quickly when something already is winning. But angels have to be able to predict. [7]

One interesting consequence of this fact is that there are a lot of people out there who have never even made an angel investment and yet are already better angel investors than they realize. Someone who doesn't know the first thing about the mechanics of venture funding but knows what a successful startup founder looks like is actually far ahead of someone who knows termsheets inside out, but thinks "["hacker"](#)" means someone who breaks into computers. If you can recognize good startup founders by empathizing with them—if you both resonate at the same frequency—then you may already be a better startup picker than the median professional VC. [8]

Paul Buchheit, for example, started angel investing about a year after me, and he was pretty much immediately as good as me at picking startups. My extra year of experience was rounding error compared to our ability to empathize with founders.

What makes a good founder? If there were a word that meant the opposite of hapless, that would be the one. Bad founders seem hapless. They may be smart, or not, but somehow events overwhelm them and they get discouraged and give up. Good founders make things happen the way they want. Which is not to say they force things to happen in a predefined way. Good founders have a healthy respect for reality. But they are relentlessly resourceful. That's the closest I can get to the opposite of hapless. You want to fund people who are relentlessly resourceful.

Notice we started out talking about things, and now we're talking about people. There is an ongoing debate between investors which is more important, the people, or the idea—or more precisely, the market. Some, like Ron Conway, say it's the people—that the idea will change, but the people are the foundation of the company. Whereas Marc Andreessen says he'd back ok founders in a hot market over great founders in a bad one. [9]

These two positions are not so far apart as they seem, because good people find good markets. Bill Gates would probably have ended up pretty rich even if IBM hadn't happened to drop the PC standard in his lap.

I've thought a lot about the disagreement between the investors who prefer to bet on people and those who prefer to bet on markets. It's kind of surprising that it even exists. You'd expect opinions to have converged more.

But I think I've figured out what's going on. The three most prominent people I know who favor markets are Marc, Jawed Karim, and Joe Kraus. And all three of them, in their own startups, basically flew into a thermal: they hit a market growing so fast that it was all they could do to keep up with it. That kind of experience is hard to ignore. Plus I think they underestimate themselves: they think back to how easy it felt to ride that huge thermal upward, and they think

"anyone could have done it." But that isn't true; they are not ordinary people.

So as an angel investor I think you want to go with Ron Conway and bet on people. Thermals happen, yes, but no one can predict them—not even the founders, and certainly not you as an investor. And only good people can ride the thermals if they hit them anyway.

## **Deal Flow**

Of course the question of how to choose startups presumes you have startups to choose between. How do you find them? This is yet another problem that gets solved for you by syndicates. If you tag along on a friend's investments, you don't have to find startups.

The problem is not finding startups, exactly, but finding a stream of reasonably high quality ones. The traditional way to do this is through contacts. If you're friends with a lot of investors and founders, they'll send deals your way. The Valley basically runs on referrals. And once you start to become known as reliable, useful investor, people will refer lots of deals to you. I certainly will.

There's also a newer way to find startups, which is to come to events like Y Combinator's Demo Day, where a batch of newly created startups presents to investors all at once. We have two Demo Days a year, one in March and one in August. These are basically mass referrals.

But events like Demo Day only account for a fraction of matches between startups and investors. The personal referral is still the most common route. So if you want to hear about new startups, the best way to do it is to get lots of referrals.

The best way to get lots of referrals is to invest in startups. No matter how smart and nice you seem, insiders will be reluctant to send you referrals until you've proven yourself by doing a couple investments. Some smart, nice guys turn out to be flaky, high-maintenance investors. But once you prove yourself as a good investor, the deal flow, as they call it, will increase rapidly in both quality and quantity. At the extreme, for someone like Ron Conway, it is basically identical with the deal flow of the whole Valley.

So if you want to invest seriously, the way to get started is to bootstrap yourself off your existing connections, be a good investor in the startups you meet that way, and eventually you'll start a chain reaction. Good investors are rare, even in Silicon Valley. There probably aren't more than a couple hundred serious angels in the whole Valley, and yet they're probably the single most important ingredient in making the Valley what it is. Angels are the limiting reagent in startup formation.

If there are only a couple hundred serious angels in the Valley, then by deciding to become one you could single-handedly make the pipeline for startups in Silicon Valley significantly wider. That is kind of mind-blowing.

## Being Good

How do you be a good angel investor? The first thing you need is to be decisive. When we talk to founders about good and bad investors, one of the ways we describe the good ones is to say "he writes checks." That doesn't mean the investor says yes to everyone. Far from it. It means he makes up his mind quickly, and follows through. You may be thinking, how hard could that be? You'll see when you try it. It follows from the nature of angel investing that the decisions are hard. You have to guess early, at the stage when the most promising ideas still seem counterintuitive, because if they were obviously good, VCs would already have funded them.

Suppose it's 1998. You come across a startup founded by a couple grad students. They say they're going to work on Internet search. There are already a bunch of big public companies doing search. How can these grad students possibly compete with them? And does search even matter anyway? All the search engines are trying to get people to start calling them "portals" instead. Why would you want to invest in a startup run by a couple of nobodies who are trying to compete with large, aggressive companies in an area they themselves have declared passe? And yet the grad students seem pretty smart. What do you do?

There's a hack for being decisive when you're inexperienced: ratchet down the size of your investment till it's an amount you wouldn't care too much about losing. For every rich person (you probably shouldn't try angel investing unless you think of yourself as rich) there's some amount that would be painless, though annoying, to lose. Till you feel comfortable investing, don't invest more than that per startup.

For example, if you have \$5 million in investable assets, it would probably be painless (though annoying) to lose \$15,000. That's less than .3% of your net worth. So start by making 3 or 4 \$15,000 investments. Nothing will teach you about angel investing like experience. Treat the first few as an educational expense. \$60,000 is less than a lot of graduate programs. Plus you get equity.

What's really uncool is to be strategically indecisive: to string founders along while trying to gather more information about the startup's trajectory. [\[10\]](#) There's always a temptation to do that, because you just have so little to go on, but you have to consciously resist it. In the long term it's to your advantage to be good.

The other component of being a good angel investor is simply to be a good person. Angel investing is not a business where you make money by screwing people over. Startups create wealth, and creating wealth is not a zero sum game. No one has to lose for you to win. In fact, if you mistreat the founders you invest in, they'll just get demoralized and the company will do worse. Plus your referrals will dry up. So I recommend being good.

The most successful angel investors I know are all basically good people. Once they invest in a company, all they want to do is help it. And they'll help people they haven't invested in too. When they do favors they don't seem to keep track of

them. It's too much overhead. They just try to help everyone, and assume good things will flow back to them somehow. Empirically that seems to work.

## Notes

[1] Convertible debt can be either capped at a particular valuation, or can be done at a discount to whatever the valuation turns out to be when it converts. E.g. convertible debt at a discount of 30% means when it converts you get stock as if you'd invested at a 30% lower valuation. That can be useful in cases where you can't or don't want to figure out what the valuation should be. You leave it to the next investor. On the other hand, a lot of investors want to know exactly what they're getting, so they will only do convertible debt with a cap.

[2] The expensive part of creating an agreement from scratch is not writing the agreement, but bickering at several hundred dollars an hour over the details. That's why the series AA paperwork aims at a middle ground. You can just start from the compromise you'd have reached after lots of back and forth.

When you fund a startup, both your lawyers should be specialists in startups. Do not use ordinary corporate lawyers for this. Their inexperience makes them overbuild: they'll create huge, overcomplicated agreements, and spend hours arguing over irrelevant things.

In the Valley, the top startup law firms are Wilson Sonsini, Orrick, Fenwick & West, Gunderson Dettmer, and Cooley Godward. In Boston the best are Goodwin Procter, Wilmer Hale, and Foley Hoag.

[3] Your mileage may vary.

[4] These anti-dilution provisions also protect you against tricks like a later investor trying to steal the company by doing another round that values the company at \$1. If you have a competent startup lawyer handle the deal for you, you should be protected against such tricks initially. But it could become a problem later. If a big VC firm wants to invest in the startup after you, they may try to make you take out your anti-dilution protections. And if they do the startup will be pressuring you to agree. They'll tell you that if you don't, you're going to kill their deal with the VC. I recommend you solve this problem by having a gentlemen's agreement with the founders: agree with them in advance that you're not going to give up your anti-dilution protections. Then it's up to them to tell VCs early on.

The reason you don't want to give them up is the following scenario. The VCs recapitalize the company, meaning they give it additional funding at a pre-money

valuation of zero. This wipes out the existing shareholders, including both you and the founders. They then grant the founders lots of options, because they need them to stay around, but you get nothing.

Obviously this is not a nice thing to do. It doesn't happen often. Brand-name VCs wouldn't recapitalize a company just to steal a few percent from an angel. But there's a continuum here. A less upstanding, lower-tier VC might be tempted to do it to steal a big chunk of stock.

I'm not saying you should always absolutely refuse to give up your anti-dilution protections. Everything is a negotiation. If you're part of a powerful syndicate, you might be able to give up legal protections and rely on social ones. If you invest in a deal led by a big angel like Ron Conway, for example, you're pretty well protected against being mistreated, because any VC would think twice before crossing him. This kind of protection is one of the reasons angels like to invest in syndicates.

[5] Don't invest so much, or at such a low valuation, that you end up with an excessively large share of a startup, unless you're sure your money will be the last they ever need. Later stage investors won't invest in a company if the founders don't have enough equity left to motivate them. I talked to a VC recently who said he'd met with a company he really liked, but he turned them down because investors already owned more than half of it. Those investors probably thought they'd been pretty clever by getting such a large chunk of this desirable company, but in fact they were shooting themselves in the foot.

[6] At any given time I know of at least 3 or 4 YC alumni who I believe will be big successes but who are running on vapor, financially, because investors don't yet get what they're doing. (And no, unfortunately, I can't tell you who they are. I can't refer a startup to an investor I don't know.)

[7] There are some VCs who can predict instead of reacting. Not surprisingly, these are the most successful ones.

[8] It's somewhat sneaky of me to put it this way, because the median VC loses money. That's one of the most surprising things I've learned about VC while working on Y Combinator. Only a fraction of VCs even have positive returns. The rest exist to satisfy demand among fund managers for venture capital as an asset class. Learning this explained a lot about some of the VCs I encountered when we were working on Viaweb.

[9] VCs also generally say they prefer great markets to great people. But what they're really saying is they want both. They're so selective that they only even consider great people. So when they say they care above all about big markets, they mean that's how they choose between great people.

[10] Founders rightly dislike the sort of investor who says he's interested in investing but doesn't want to lead. There are circumstances where this is an acceptable excuse, but more often than not what it means is "No, but if you turn

out to be a hot deal, I want to be able to claim retroactively I said yes."

If you like a startup enough to invest in it, then invest in it. Just use the standard [series AA](#) terms and write them a check.

**Thanks** to Sam Altman, Paul Buchheit, Jessica Livingston, Robert Morris, and Fred Wilson for reading drafts of this.



[Comment](#) on this essay.

# Relentlessly Resourceful

March 2009

A couple days ago I finally got being a good startup founder down to two words: relentlessly resourceful.

Till then the best I'd managed was to get the opposite quality down to one: hapless. Most dictionaries say hapless means unlucky. But the dictionaries are not doing a very good job. A team that outplays its opponents but loses because of a bad decision by the referee could be called unlucky, but not hapless. Hapless implies passivity. To be hapless is to be battered by circumstances — to let the world have its way with you, instead of having your way with the world. [[1](#)]

Unfortunately there's no antonym of hapless, which makes it difficult to tell founders what to aim for. "Don't be hapless" is not much of a rallying cry.

It's not hard to express the quality we're looking for in metaphors. The best is probably a running back. A good running back is not merely determined, but flexible as well. They want to get downfield, but they adapt their plans on the fly.

Unfortunately this is just a metaphor, and not a useful one to most people outside the US. "Be like a running back" is no better than "Don't be hapless."

But finally I've figured out how to express this quality directly. I was writing a talk for [investors](#), and I had to explain what to look for in founders. What would someone who was the opposite of hapless be like? They'd be relentlessly resourceful. Not merely relentless. That's not enough to make things go your way except in a few mostly uninteresting domains. In any interesting domain, the difficulties will be novel. Which means you can't simply plow through them, because you don't know initially how hard they are; you don't know whether you're about to plow through a block of foam or granite. So you have to be resourceful. You have to keep trying new things.

Be relentlessly resourceful.

That sounds right, but is it simply a description of how to be successful in general? I don't think so. This isn't the recipe for success in writing or painting, for example. In that kind of work the recipe is more to be actively curious. Resourceful implies

the obstacles are external, which they generally are in startups. But in writing and painting they're mostly internal; the obstacle is your own obtuseness. [2]

There probably are other fields where "relentlessly resourceful" is the recipe for success. But though other fields may share it, I think this is the best short description we'll find of what makes a good startup founder. I doubt it could be made more precise.

Now that we know what we're looking for, that leads to other questions. For example, can this quality be taught? After four years of trying to teach it to people, I'd say that yes, surprisingly often it can. Not to everyone, but to many people. [3] Some people are just constitutionally passive, but others have a latent ability to be relentlessly resourceful that only needs to be brought out.

This is particularly true of young people who have till now always been under the thumb of some kind of authority. Being relentlessly resourceful is definitely not the recipe for success in big companies, or in most schools. I don't even want to think what the recipe is in big companies, but it is certainly longer and messier, involving some combination of resourcefulness, obedience, and building alliances.

Identifying this quality also brings us closer to answering a question people often wonder about: how many startups there could be. There is not, as some people seem to think, any economic upper bound on this number. There's no reason to believe there is any limit on the amount of newly created wealth consumers can absorb, any more than there is a limit on the number of theorems that can be proven. So probably the limiting factor on the number of startups is the pool of potential founders. Some people would make good founders, and others wouldn't. And now that we can say what makes a good founder, we know how to put an upper bound on the size of the pool.

This test is also useful to individuals. If you want to know whether you're the right sort of person to start a startup, ask yourself whether you're relentlessly resourceful. And if you want to know whether to recruit someone as a cofounder, ask if they are.

You can even use it tactically. If I were running a startup, this would be the phrase I'd tape to the mirror. "Make something people want" is the destination, but "Be relentlessly resourceful" is how you get there.

## Notes

[1] I think the reason the dictionaries are wrong is that the meaning of the word



has shifted. No one writing a dictionary from scratch today would say that hapless meant unlucky. But a couple hundred years ago they might have. People were more at the mercy of circumstances in the past, and as a result a lot of the words we use for good and bad outcomes have origins in words about luck.

When I was living in Italy, I was once trying to tell someone that I hadn't had much success in doing something, but I couldn't think of the Italian word for success. I spent some time trying to describe the word I meant. Finally she said "Ah! Fortuna!"

[2] There are aspects of startups where the recipe is to be actively curious. There can be times when what you're doing is almost pure discovery. Unfortunately these times are a small proportion of the whole. On the other hand, they are in research too.

[3] I'd almost say to most people, but I realize (a) I have no idea what most people are like, and (b) I'm pathologically optimistic about people's ability to change.

**Thanks** to Trevor Blackwell and Jessica Livingston for reading drafts of this.

# Five Founders

April 2009

*Inc* recently asked me who I thought were the 5 most interesting startup founders of the last 30 years. How do you decide who's the most interesting? The best test seemed to be influence: who are the 5 who've influenced me most? Who do I use as examples when I'm talking to companies we fund? Who do I find myself quoting?

## 1. Steve Jobs

I'd guess Steve is the most influential founder not just for me but for most people you could ask. A lot of startup culture is Apple culture. He was the original young founder. And while the concept of "insanely great" already existed in the arts, it was a novel idea to introduce into a company in the 1980s.

More remarkable still, he's stayed interesting for 30 years. People await new Apple products the way they'd await new books by a popular novelist. Steve may not literally design them, but they wouldn't happen if he weren't CEO.

Steve is clever and driven, but so are a lot of people in the Valley. What makes him unique is his [sense of design](#). Before him, most companies treated design as a frivolous extra. Apple's competitors now know better.

## 2. TJ Rodgers

TJ Rodgers isn't as famous as Steve Jobs, but he may be the best writer among Silicon Valley CEOs. I've probably learned more from him about the startup way of thinking than from anyone else. Not so much from specific things he's written as by reconstructing the mind that produced them: brutally candid; aggressively garbage-collecting outdated ideas; and yet driven by pragmatism rather than ideology.

The first essay of his that I read was so electrifying that I remember exactly where I was at the time. It was [High Technology Innovation: Free Markets or Government Subsidies?](#) and I was downstairs in the Harvard Square T Station. It felt as if someone had flipped on a light switch inside my head.

## 3. Larry & Sergey

I'm sorry to treat Larry and Sergey as one person. I've always thought that was unfair to them. But it does seem as if Google was a collaboration.

Before Google, companies in Silicon Valley already knew it was important to have the best hackers. So they claimed, at least. But Google pushed this idea further than anyone had before. Their hypothesis seems to have been that, in the initial stages at least, *all* you need is good hackers: if you hire all the smartest people and put them to work on a problem where their success can be measured, you win. All the other stuff—which includes all the stuff that business schools think business consists of—you can figure out along the way. The results won't be perfect, but they'll be optimal. If this was their hypothesis, it's now been verified experimentally.

#### **4. Paul Buchheit**

Few know this, but one person, Paul Buchheit, is responsible for three of the best things Google has done. He was the original author of GMail, which is the most impressive thing Google has after search. He also wrote the first prototype of AdSense, and was the author of Google's mantra "Don't be evil."

PB made a point in a talk once that I now mention to every startup we fund: that it's better, initially, to make a small number of users really love you than a large number kind of like you. If I could tell startups only [ten sentences](#), this would be one of them.

Now he's cofounder of a startup called Friendfeed. It's only a year old, but already everyone in the Valley is watching them. Someone responsible for three of the biggest ideas at Google is going to come up with more.

#### **5. Sam Altman**

I was told I shouldn't mention founders of YC-funded companies in this list. But Sam Altman can't be stopped by such flimsy rules. If he wants to be on this list, he's going to be.

Honestly, Sam is, along with Steve Jobs, the founder I refer to most when I'm advising startups. On questions of design, I ask "What would Steve do?" but on questions of strategy or ambition I ask "What would Sama do?"

What I learned from meeting Sama is that the doctrine of the elect applies to startups. It applies way less than most people think: startup investing does not consist of trying to pick winners the way you might in a horse race. But there are a few people with such force of will that they're going to get whatever they want.

# The Founder Visa

April 2009

I usually avoid politics, but since we now seem to have an administration that's open to suggestions, I'm going to risk making one. The single biggest thing the government could do to increase the number of startups in this country is a policy that would cost nothing: establish a new class of visa for startup founders.

The biggest constraint on the number of new startups that get created in the US is not tax policy or employment law or even Sarbanes-Oxley. It's that we won't let the people who want to start them into the country.

Letting just 10,000 startup founders into the country each year could have a visible effect on the economy. If we assume 4 people per startup, which is probably an overestimate, that's 2500 new companies. *Each year*. They wouldn't all grow as big as Google, but out of 2500 some would come close.

By definition these 10,000 founders wouldn't be taking jobs from Americans: it could be part of the terms of the visa that they couldn't work for existing companies, only new ones they'd founded. In fact they'd cause there to be more jobs for Americans, because the companies they started would hire more employees as they grew.

The tricky part might seem to be how one defined a startup. But that could be solved quite easily: let the market decide. Startup investors work hard to find the best startups. The government could not do better than to piggyback on their expertise, and use investment by recognized startup investors as the test of whether a company was a real startup.

How would the government decide who's a startup investor? The same way they decide what counts as a university for student visas. We'll establish our own accreditation procedure. We know who one another are.

10,000 people is a drop in the bucket by immigration standards, but would represent a huge increase in the pool of startup founders. I think this would have such a visible effect on the economy that it would make the legislator who introduced the bill famous. The only way to know for sure would be to try it, and that would cost practically nothing.

**Thanks** to Trevor Blackwell, Paul Buchheit, Jeff Clavier, David Hornik, Jessica Livingston, Greg Mcadoo, Aydin Senkut, and Fred Wilson for reading drafts of this.

**Related:**

[The United States of Entrepreneurs](#)

[About Half of VC-Backed Company Founders are Immigrants](#)

# Why Twitter is a Big Deal

April 2009

[Om Malik](#) is the most recent of many people to ask why Twitter is such a big deal.

The reason is that it's a new messaging protocol, where you don't specify the recipients. New protocols are rare. Or more precisely, new protocols that take off are. There are only a handful of commonly used ones: TCP/IP (the Internet), SMTP (email), HTTP (the web), and so on. So any new protocol is a big deal. But Twitter is a protocol owned by a private company. That's even rarer.

Curiously, the fact that the founders of Twitter have been slow to monetize it may in the long run prove to be an advantage. Because they haven't tried to control it too much, Twitter feels to everyone like previous protocols. One forgets it's owned by a private company. That must have made it easier for Twitter to spread.

# A Local Revolution?

April 2009

Recently I realized I'd been holding two ideas in my head that would explode if combined.

The first is that startups may represent a [new economic phase](#), on the scale of the Industrial Revolution. I'm not sure of this, but there seems a decent chance it's true. People are dramatically more productive as founders or early employees of startups—imagine how much less Larry and Sergey would have achieved if they'd gone to work for a big company—and that scale of improvement can change social customs.

The second idea is that startups are a type of business that flourishes in certain places that [specialize](#) in it—that Silicon Valley specializes in startups in the same way Los Angeles specializes in movies, or New York in finance. [[1](#)]

What if both are true? What if startups are both a new economic phase and also a type of business that only flourishes in certain centers?

If so, this revolution is going to be particularly revolutionary. All previous revolutions have spread. Agriculture, cities, and industrialization all spread widely. If startups end up being like the movie business, with just a handful of centers and one dominant one, that's going to have novel consequences.

There are already signs that startups may not spread particularly well. The spread of startups seems to be proceeding slower than the spread of the Industrial Revolution, despite the fact that communication is so much faster now.

Within a few decades of the founding of Boulton & Watt there were steam engines scattered over northern Europe and North America. Industrialization didn't spread much beyond those regions for a while. It only spread to places where there was a strong middle class—countries where a private citizen could make a fortune without having it confiscated. Otherwise it wasn't worth investing in factories. But in a country with a strong middle class it was easy for industrial techniques to take root. An individual mine or factory owner could decide to install a steam engine, and within a few years he could probably find someone local to make him one. So steam engines spread fast. And they spread widely, because the locations of mines and factories were determined by features like rivers, harbors, and sources of raw

materials. [2]

Startups don't seem to spread so well, partly because they're more a social than a technical phenomenon, and partly because they're not tied to geography. An individual European manufacturer could import industrial techniques and they'd work fine. This doesn't seem to work so well with startups: you need a community of expertise, as you do in the movie business. [3] Plus there aren't the same forces driving startups to spread. Once railroads or electric power grids were invented, every region had to have them. An area without railroads or power was a rich potential market. But this isn't true with startups. There's no need for a Microsoft of France or Google of Germany.

Governments may decide they want to encourage startups locally, but government policy can't call them into being the way a genuine need could.

How will this all play out? If I had to predict now, I'd say that startups will spread, but very slowly, because their spread will be driven not by government policies (which won't work) or by market need (which doesn't exist) but, to the extent that it happens at all, by the same random factors that have caused startup culture to spread thus far. And such random factors will increasingly be outweighed by the pull of existing startup hubs.

Silicon Valley is where it is because William Shockley wanted to move back to Palo Alto, where he grew up, and the experts he lured west to work with him liked it so much they stayed. Seattle owes much of its position as a tech center to the same cause: Gates and Allen wanted to move home. Otherwise Albuquerque might have Seattle's place in the rankings. Boston is a tech center because it's the intellectual capital of the US and probably the world. And if Battery Ventures hadn't turned down Facebook, Boston would be significantly bigger now on the startup radar screen.

But of course it's not a coincidence that Facebook got funded in the Valley and not Boston. There are more and bolder investors in Silicon Valley than in Boston, and even undergrads know it.

Boston's case illustrates the difficulty you'd have establishing a new startup hub this late in the game. If you wanted to create a startup hub by reproducing the way existing ones happened, the [way to do it](#) would be to establish a first-rate research university in a place so nice that rich people wanted to live there. Then the town would be hospitable to both groups you need: both founders and investors. That's the combination that yielded Silicon Valley. But Silicon Valley didn't have Silicon Valley to compete with. If you tried now to create a startup hub by planting a great university in a nice place, it would have a harder time getting started, because many of the best startups it produced would be sucked away to existing startup hubs.

Recently I suggested a potential shortcut: [pay startups to move](#). Once you had enough good startups in one place, it would create a self-sustaining chain reaction.



Founders would start to move there without being paid, because that was where their peers were, and investors would appear too, because that was where the deals were.

In practice I doubt any government would have the balls to try this, or the brains to do it right. I didn't mean it as a practical suggestion, but more as an exploration of the lower bound of what it would take to create a startup hub deliberately.

The most likely scenario is (1) that no government will successfully establish a startup hub, and (2) that the spread of startup culture will thus be driven by the random factors that have driven it so far, but (3) that these factors will be increasingly outweighed by the pull of existing startup hubs. Result: this revolution, if it is one, will be unusually localized.

## Notes

[1] There are two very different types of startup: one kind that evolves naturally, and one kind that's called into being to "commercialize" a scientific discovery. Most computer/software startups are now the first type, and most pharmaceutical startups the second. When I talk about startups in this essay, I mean type I startups. There is no difficulty making type II startups spread: all you have to do is fund medical research labs; commercializing whatever new discoveries the boffins throw off is as straightforward as building a new airport. Type II startups neither require nor produce startup culture. But that means having type II startups won't get you type I startups. Philadelphia is a case in point: lots of type II startups, but hardly any type I.

Incidentally, Google may appear to be an instance of a type II startup, but it wasn't. Google is not pagerank commercialized. They could have used another algorithm and everything would have turned out the same. What made Google Google is that they cared about doing search well at a critical point in the evolution of the web.

[2] Watt didn't invent the steam engine. His critical invention was a refinement that made steam engines dramatically more efficient: the separate condenser. But that oversimplifies his role. He had such a different attitude to the problem and approached it with such energy that he transformed the field. Perhaps the most accurate way to put it would be to say that Watt reinvented the steam engine.

[3] The biggest counterexample here is Skype. If you're doing something that would get shut down in the US, it becomes an advantage to be located elsewhere. That's why Kazaa took the place of Napster. And the expertise and connections the founders gained from running Kazaa helped ensure the success of Skype.

**Thanks** to Patrick Collison, Jessica Livingston, and Fred Wilson for reading drafts of this.

# Maker's Schedule, Manager's Schedule

"...the mere consciousness of an engagement will sometimes worry a whole day."

◆ Charles Dickens

July 2009

One reason programmers dislike meetings so much is that they're on a different type of schedule from other people. Meetings cost them more.

There are two types of schedule, which I'll call the manager's schedule and the maker's schedule. The manager's schedule is for bosses. It's embodied in the traditional appointment book, with each day cut into one hour intervals. You can block off several hours for a single task if you need to, but by default you change what you're doing every hour.

When you use time that way, it's merely a practical problem to meet with someone. Find an open slot in your schedule, book them, and you're done.

Most powerful people are on the manager's schedule. It's the schedule of command. But there's another way of using time that's common among people who make things, like programmers and writers. They generally prefer to use time in units of half a day at least. You can't write or program well in units of an hour. That's barely enough time to get started.

When you're operating on the maker's schedule, meetings are a disaster. A single meeting can blow a whole afternoon, by breaking it into two pieces each too small to do anything hard in. Plus you have to remember to go to the meeting. That's no problem for someone on the manager's schedule. There's always something coming on the next hour; the only question is what. But when someone on the maker's schedule has a meeting, they have to think about it.

For someone on the maker's schedule, having a meeting is like throwing an exception. It doesn't merely cause you to switch from one task to another; it changes the mode in which you work.

I find one meeting can sometimes affect a whole day. A meeting commonly blows at least half a day, by breaking up a morning or afternoon. But in addition there's sometimes a cascading effect. If I know the afternoon is going to be broken up, I'm slightly less likely to start something ambitious in the morning. I know this

may sound oversensitive, but if you're a maker, think of your own case. Don't your spirits rise at the thought of having an entire day free to work, with no appointments at all? Well, that means your spirits are correspondingly depressed when you don't. And ambitious projects are by definition close to the limits of your capacity. A small decrease in morale is enough to kill them off.

Each type of schedule works fine by itself. Problems arise when they meet. Since most powerful people operate on the manager's schedule, they're in a position to make everyone resonate at their frequency if they want to. But the smarter ones restrain themselves, if they know that some of the people working for them need long chunks of time to work in.

Our case is an unusual one. Nearly all investors, including all VCs I know, operate on the manager's schedule. But [Y Combinator](#) runs on the maker's schedule. Rtm and Trevor and I do because we always have, and Jessica does too, mostly, because she's gotten into sync with us.

I wouldn't be surprised if there start to be more companies like us. I suspect founders may increasingly be able to resist, or at least postpone, turning into managers, just as a few decades ago they started to be able to resist switching from jeans to suits.

How do we manage to advise so many startups on the maker's schedule? By using the classic device for simulating the manager's schedule within the maker's: office hours. Several times a week I set aside a chunk of time to meet founders we've funded. These chunks of time are at the end of my working day, and I wrote a signup program that ensures all the appointments within a given set of office hours are clustered at the end. Because they come at the end of my day these meetings are never an interruption. (Unless their working day ends at the same time as mine, the meeting presumably interrupts theirs, but since they made the appointment it must be worth it to them.) During busy periods, office hours sometimes get long enough that they compress the day, but they never interrupt it.

When we were working on [our own startup](#), back in the 90s, I evolved another trick for partitioning the day. I used to program from dinner till about 3 am every day, because at night no one could interrupt me. Then I'd sleep till about 11 am, and come in and work until dinner on what I called "business stuff." I never thought of it in these terms, but in effect I had two workdays each day, one on the manager's schedule and one on the maker's.

When you're operating on the manager's schedule you can do something you'd never want to do on the maker's: you can have speculative meetings. You can meet someone just to get to know one another. If you have an empty slot in your schedule, why not? Maybe it will turn out you can help one another in some way.

Business people in Silicon Valley (and the whole world, for that matter) have speculative meetings all the time. They're effectively free if you're on the

manager's schedule. They're so common that there's distinctive language for proposing them: saying that you want to "grab coffee," for example.

Speculative meetings are terribly costly if you're on the maker's schedule, though. Which puts us in something of a bind. Everyone assumes that, like other investors, we run on the manager's schedule. So they introduce us to someone they think we ought to meet, or send us an email proposing we grab coffee. At this point we have two options, neither of them good: we can meet with them, and lose half a day's work; or we can try to avoid meeting them, and probably offend them.

Till recently we weren't clear in our own minds about the source of the problem. We just took it for granted that we had to either blow our schedules or offend people. But now that I've realized what's going on, perhaps there's a third option: to write something explaining the two types of schedule. Maybe eventually, if the conflict between the manager's schedule and the maker's schedule starts to be more widely understood, it will become less of a problem.

Those of us on the maker's schedule are willing to compromise. We know we have to have some number of meetings. All we ask from those on the manager's schedule is that they understand the cost.

**Thanks** to Sam Altman, Trevor Blackwell, Paul Buchheit, Jessica Livingston, and Robert Morris for reading drafts of this.

## **Related:**

[How to Do What You Love](#)

[Good and Bad Procrastination](#)

[Turkish Translation](#)

[French Translation](#)

[Korean Translation](#)

[German Translation](#)



# Ramen Profitable

July 2009

Now that the term "ramen profitable" has become widespread, I ought to explain precisely what the idea entails.

Ramen profitable means a startup makes just enough to pay the founders' living expenses. This is a different form of profitability than startups have traditionally aimed for. Traditional profitability means a big bet is finally paying off, whereas the main importance of ramen profitability is that it buys you time. [\[1\]](#)

In the past, a startup would usually become profitable only after raising and spending quite a lot of money. A company making computer hardware might not become profitable for 5 years, during which they spent \$50 million. But when they did they might have revenues of \$50 million a year. This kind of profitability means the startup has succeeded.

Ramen profitability is the other extreme: a startup that becomes profitable after 2 months, even though its revenues are only \$3000 a month, because the only employees are a couple 25 year old founders who can live on practically nothing. Revenues of \$3000 a month do not mean the company has succeeded. But it does share something with the one that's profitable in the traditional way: they don't need to raise money to survive.

Ramen profitability is an unfamiliar idea to most people because it only recently became feasible. It's still not feasible for a lot of startups; it would not be for most biotech startups, for example; but it is for many software startups because they're now so cheap. For many, the only real cost is the founders' living expenses.

The main significance of this type of profitability is that you're no longer at the mercy of investors. If you're still losing money, then eventually you'll either have to raise more or shut down. Once you're ramen profitable this painful choice goes away. You can still raise money, but you don't have to do it now.

\* \* \*

The most obvious advantage of not needing money is that you can get better terms. If investors know you need money, they'll sometimes take advantage of you. Some may even deliberately stall, because they know that as you run out of money you'll become increasingly pliable.

But there are also three less obvious advantages of ramen profitability. One is that it makes you more attractive to investors. If you're already profitable, on however small a scale, it shows that (a) you can get at least someone to pay you, (b) you're serious about building things people want, and (c) you're disciplined enough to keep expenses low.

This is reassuring to investors, because you've addressed three of their biggest worries. It's common for them to fund companies that have smart founders and a big market, and yet still fail. When these companies fail, it's usually because (a) people wouldn't pay for what they made, e.g. because it was too hard to sell to them, or the market wasn't ready yet, (b) the founders solved the wrong problem, instead of paying attention to what users needed, or (c) the company spent too much and burned through their funding before they started to make money. If you're ramen profitable, you're already avoiding these mistakes.

Another advantage of ramen profitability is that it's good for morale. A company tends to feel rather theoretical when you first start it. It's legally a company, but you feel like you're lying when you call it one. When people start to pay you significant amounts, the company starts to feel real. And your own living expenses are the milestone you feel most, because at that point the future flips state. Now survival is the default, instead of dying.

A morale boost on that scale is very valuable in a startup, because the moral weight of running a startup is what makes it hard. Startups are still very rare. Why don't more people do it? The financial risk? Plenty of 25 year olds save nothing anyway. The long hours? Plenty of people work just as long hours in regular jobs. What keeps people from starting startups is the fear of having so much responsibility. And this is not an irrational fear: it really is hard to bear. Anything that takes some of that weight off you will greatly increase your chances of surviving.

A startup that reaches ramen profitability may be more likely to succeed than not. Which is pretty exciting, considering the bimodal distribution of outcomes in startups: you either fail or make a lot of money.

The fourth advantage of ramen profitability is the least obvious but may be the most important. If you don't need to raise money, you don't have to interrupt working on the company to do it.

[Raising money](#) is terribly distracting. You're lucky if your productivity is a third of what it was before. And it can last for months.

I didn't understand (or rather, remember) precisely why raising money was so distracting till earlier this year. I'd noticed that startups we funded would usually grind to a halt when they switched to raising money, but I didn't remember exactly why till YC raised money itself. We had a comparatively easy time of it; the first people I asked said yes; but it took months to work out the details, and during



that time I got hardly any real work done. Why? Because I thought about it all the time.

At any given time there tends to be one problem that's the most urgent for a startup. This is what you think about as you fall asleep at night and when you take a shower in the morning. And when you start raising money, that becomes the problem you think about. You only take one shower in the morning, and if you're thinking about investors during it, then you're not thinking about the product.

Whereas if you can choose when you raise money, you can pick a time when you're not in the middle of something else, and you can probably also insist that the round close fast. You may even be able to avoid having the round occupy your thoughts, if you don't care whether it closes.

\* \* \*

Ramen profitable means no more than the definition implies. It does not, for example, imply that you're "bootstrapping" the startup—that you're never going to take money from investors. Empirically that doesn't seem to work very well. Few startups succeed without taking investment. Maybe as startups get cheaper it will become more common. On the other hand, the money is there, waiting to be invested. If startups need it less, they'll be able to get it on better terms, which will make them more inclined to take it. That will tend to produce an equilibrium. [2]

Another thing ramen profitability doesn't imply is Joe Kraus's idea that you should put your [business model](#) in beta when you put your product in beta. He believes you should get people to pay you from the beginning. I think that's too constraining. Facebook didn't, and they've done better than most startups. Making money right away was not only unnecessary for them, but probably would have been harmful. I do think Joe's rule could be useful for many startups, though. When founders seem unfocused, I sometimes suggest they try to get customers to pay them for something, in the hope that this constraint will prod them into action.

The difference between Joe's idea and ramen profitability is that a ramen profitable company doesn't have to be making money the way it ultimately will. It just has to be making money. The most famous example is Google, which initially made money by licensing search to sites like Yahoo.

Is there a downside to ramen profitability? Probably the biggest danger is that it might turn you into a consulting firm. Startups have to be product companies, in the sense of making a single thing that everyone uses. The defining quality of startups is that they grow fast, and consulting just can't scale the way a product can. [3] But it's pretty easy to make \$3000 a month consulting; in fact, that would be a low rate for contract programming. So there could be a temptation to slide into consulting, and telling yourselves you're a ramen profitable startup, when in fact you're not a startup at all.

It's ok to do a little consulting-type work at first. Startups usually have to do something weird at first. But remember that ramen profitability is not the destination. A startup's destination is to grow really big; ramen profitability is a trick for [not dying](#) en route.

## Notes

[1] The "ramen" in "ramen profitable" refers to instant ramen, which is just about the cheapest food available.

Please do not take the term literally. Living on instant ramen would be very unhealthy. Rice and beans are a better source of food. Start by investing in a rice cooker, if you don't have one.

### Rice and Beans for 2n

- olive oil or butter
- n yellow onions
- other fresh vegetables; experiment
- 3n cloves garlic
- n 12-oz cans white, kidney, or black beans
- n cubes Knorr beef or vegetable bouillon
- n teaspoons freshly ground black pepper
- 3n teaspoons ground cumin
- n cups dry rice, preferably brown

Put rice in rice cooker. Add water as specified on rice package. (Default: 2 cups water per cup of rice.) Turn on rice cooker and forget about it.

Chop onions and other vegetables and fry in oil, over fairly low heat, till onions are glassy. Put in chopped garlic, pepper, cumin, and a little more fat, and stir. Keep heat low. Cook another 2 or 3 minutes, then add beans (don't drain the beans), and stir. Throw in the bouillon cube(s), cover, and cook on lowish heat for at least 10 minutes more. Stir vigilantly to avoid sticking.

If you want to save money, buy beans in giant cans from discount stores. Spices are also much cheaper when bought in bulk. If there's an Indian grocery store near you, they'll have big bags of cumin for the same price as the little jars in supermarkets.

[2] There's a good chance that a shift in power from investors to founders would actually increase the size of the venture business. I think investors currently err too far on the side of being harsh to founders. If they were forced to stop, the whole venture business would work better, and you might see something like the increase in trade you always see when restrictive laws are removed.

Investors are one of the biggest sources of pain for founders; if they stopped causing so much pain, it would be better to be a founder; and if it were better to be a founder, more people would do it.

[3] It's conceivable that a startup could grow big by transforming consulting into a form that would scale. But if they did that they'd really be a product company.

**Thanks** to Jessica Livingston for reading drafts of this.

[Japanese Translation](#)

# The Trouble with the Segway

July 2009

The Segway hasn't delivered on its initial promise, to put it mildly. There are several reasons why, but one is that people don't want to be seen riding them. Someone riding a Segway looks like a dork.

My friend Trevor Blackwell built [his own Segway](#), which we called the Segwell. He also built a one-wheeled version, [the Eunicycle](#), which looks exactly like a regular unicycle till you realize the rider isn't pedaling. He has ridden them both to downtown Mountain View to get coffee. When he rides the Eunicycle, people smile at him. But when he rides the Segwell, they shout abuse from their cars: "Too lazy to walk, ya fuckin homo?"

Why do Segways provoke this reaction? The reason you look like a dork riding a Segway is that you look *smug*. You don't seem to be working hard enough.

Someone riding a motorcycle isn't working any harder. But because he's sitting astride it, he seems to be making an effort. When you're riding a Segway you're just standing there. And someone who's being whisked along while seeming to do no work — someone in a sedan chair, for example — can't help but look smug.

Try this thought experiment and it becomes clear: imagine something that worked like the Segway, but that you rode with one foot in front of the other, like a skateboard. That wouldn't seem nearly as uncool.

So there may be a way to capture more of the market Segway hoped to reach: make a version that doesn't look so easy for the rider. It would also be helpful if the styling was in the tradition of skateboards or bicycles rather than medical devices.

Curiously enough, what got Segway into this problem was that the company was itself a kind of Segway. It was too easy for them; they were too successful raising money. If they'd had to grow the company gradually, by iterating through several versions they sold to real users, they'd have learned pretty quickly that people looked stupid riding them. Instead they had enough to work in secret. They had focus groups aplenty, I'm sure, but they didn't have the people yelling insults out of cars. So they never realized they were zooming confidently down a blind alley.



# What Kate Saw in Silicon Valley

August 2009

Kate Courteau is the architect who designed Y Combinator's office. Recently we managed to recruit her to help us run YC when she's not busy with architectural projects. Though she'd heard a lot about YC since the beginning, the last 9 months have been a total immersion.

I've been around the startup world for so long that it seems normal to me, so I was curious to hear what had surprised her most about it. This was her list:

## **1. How many startups fail.**

Kate knew in principle that startups were very risky, but she was surprised to see how constant the threat of failure was — not just for the minnows, but even for the famous startups whose founders came to speak at YC dinners.

## **2. How much startups' ideas change.**

As usual, by Demo Day about half the startups were doing something significantly different than they started with. We encourage that. Starting a startup is like science in that you have to follow the truth wherever it leads. In the rest of the world, people don't start things till they're sure what they want to do, and once started they tend to continue on their initial path even if it's mistaken.

## **3. How little money it can take to start a startup.**

In Kate's world, everything is still physical and expensive. You can barely renovate a bathroom for the cost of starting a startup.

## **4. How scrappy founders are.**

That was her actual word. I agree with her, but till she mentioned this it never occurred to me how little this quality is appreciated in most of the rest of the world. It wouldn't be a compliment in most organizations to call someone scrappy.

What does it mean, exactly? It's basically the diminutive form of belligerent.

Someone who's scrappy manages to be both threatening and undignified at the same time. Which seems to me exactly what one would want to be, in any kind of work. If you're not threatening, you're probably not doing anything new, and dignity is merely a sort of plaque.

## **5. How tech-saturated Silicon Valley is.**

"It seems like everybody here is in the industry." That isn't literally true, but there is a qualitative difference between Silicon Valley and other places. You tend to keep your voice down, because there's a good chance the person at the next table would know some of the people you're talking about. I never felt that in Boston. The good news is, there's also a good chance the person at the next table could help you in some way.

## **6. That the speakers at YC were so consistent in their advice.**

Actually, I've noticed this too. I always worry the speakers will put us in an embarrassing position by contradicting what we tell the startups, but it happens surprisingly rarely.

When I asked her what specific things she remembered speakers always saying, she mentioned: that the way to succeed was to launch something fast, listen to users, and then iterate; that startups required resilience because they were always an emotional rollercoaster; and that most VCs were sheep.

I've been impressed by how consistently the speakers advocate launching fast and iterating. That was contrarian advice 10 years ago, but it's clearly now the established practice.

## **7. How casual successful startup founders are.**

Most of the famous founders in Silicon Valley are people you'd overlook on the street. It's not merely that they don't dress up. They don't project any kind of aura of power either. "They're not trying to impress anyone."

Interestingly, while Kate said that she could never pick out successful founders, she could recognize VCs, both by the way they dressed and the way they carried themselves.

## **8. How important it is for founders to have people to ask for advice.**

(I swear I didn't prompt this one.) Without advice "they'd just be sort of lost." Fortunately, there are a lot of people to help them. There's a strong tradition within YC of helping other YC-funded startups. But we didn't invent that idea: it's just a slightly more concentrated form of existing Valley culture.

## **9. What a solitary task startups are.**

Architects are constantly interacting face to face with other people, whereas doing a technology startup, at least, tends to require long stretches of uninterrupted time to work. "You could do it in a box."

By inverting this list, we can get a portrait of the "normal" world. It's populated by people who talk a lot with one another as they work slowly but harmoniously on conservative, expensive projects whose destinations are decided in advance, and who carefully adjust their manner to reflect their position in the hierarchy.

That's also a fairly accurate description of the past. So startup culture may not merely be different in the way you'd expect any subculture to be, but a leading indicator.

[Japanese Translation](#)



# The Anatomy of Determination

September 2009

Like all investors, we spend a lot of time trying to learn how to predict which startups will succeed. We probably spend more time thinking about it than most, because we invest the earliest. Prediction is usually all we have to rely on.

We learned quickly that the most important predictor of success is determination. At first we thought it might be intelligence. Everyone likes to believe that's what makes startups succeed. It makes a better story that a company won because its founders were so smart. The PR people and reporters who spread such stories probably believe them themselves. But while it certainly helps to be smart, it's not the deciding factor. There are plenty of people as smart as Bill Gates who achieve nothing.

In most domains, talent is overrated compared to determination—partly because it makes a better story, partly because it gives onlookers an excuse for being lazy, and partly because after a while determination starts to look like talent.

I can't think of any field in which determination is overrated, but the relative importance of determination and talent probably do vary somewhat. Talent probably matters more in types of work that are purer, in the sense that one is solving mostly a single type of problem instead of many different types. I suspect determination would not take you as far in math as it would in, say, organized crime.

I don't mean to suggest by this comparison that types of work that depend more on talent are always more admirable. Most people would agree it's more admirable to be good at math than memorizing long strings of digits, even though the latter depends more on natural ability.

Perhaps one reason people believe startup founders win by being smarter is that intelligence does matter more in technology startups than it used to in earlier types of companies. You probably do need to be a bit smarter to dominate Internet search than you had to be to dominate railroads or hotels or newspapers. And that's probably an ongoing trend. But even in the highest of high tech industries, success still depends more on determination than brains.

If determination is so important, can we isolate its components? Are some more important than others? Are there some you can cultivate?

The simplest form of determination is sheer willfulness. When you want something, you must have it, no matter what.

A good deal of willfulness must be inborn, because it's common to see families where one sibling has much more of it than another. Circumstances can alter it, but at the high end of the scale, nature seems to be more important than nurture. Bad circumstances can break the spirit of a strong-willed person, but I don't think there's much you can do to make a weak-willed person stronger-willed.

Being strong-willed is not enough, however. You also have to be hard on yourself. Someone who was strong-willed but self-indulgent would not be called determined. Determination implies your willfulness is balanced by discipline.

That word balance is a significant one. The more willful you are, the more disciplined you have to be. The stronger your will, the less anyone will be able to argue with you except yourself. And someone has to argue with you, because everyone has base impulses, and if you have more will than discipline you'll just give into them and end up on a local maximum like drug addiction.

We can imagine will and discipline as two fingers squeezing a slippery melon seed. The harder they squeeze, the further the seed flies, but they must both squeeze equally or the seed spins off sideways.

If this is true it has interesting implications, because discipline can be cultivated, and in fact does tend to vary quite a lot in the course of an individual's life. If determination is effectively the product of will and discipline, then you can become more determined by being more disciplined. [\[1\]](#)

Another consequence of the melon seed model is that the more willful you are, the more dangerous it is to be undisciplined. There seem to be plenty of examples to confirm that. In some very energetic people's lives you see something like wing flutter, where they alternate between doing great work and doing absolutely nothing. Externally this would look a lot like bipolar disorder.

The melon seed model is inaccurate in at least one respect, however: it's static. In fact the dangers of indiscipline increase with temptation. Which means, interestingly, that determination tends to erode itself. If you're sufficiently determined to achieve great things, this will probably increase the number of temptations around you. Unless you become proportionally more disciplined, willfulness will then get the upper hand, and your achievement will revert to the mean.

That's why Shakespeare's Caesar thought thin men so dangerous. They weren't tempted by the minor perquisites of power.

The melon seed model implies it's possible to be too disciplined. Is it? I think there probably are people whose willfulness is crushed down by excessive discipline, and who would achieve more if they weren't so hard on themselves. One reason the young sometimes succeed where the old fail is that they don't realize how incompetent they are. This lets them do a kind of deficit spending. When they first start working on something, they overrate their achievements. But that gives them confidence to keep working, and their performance improves. Whereas someone clearer-eyed would see their initial incompetence for what it was, and perhaps be discouraged from continuing.

There's one other major component of determination: ambition. If willfulness and discipline are what get you to your destination, ambition is how you choose it.

I don't know if it's exactly right to say that ambition is a component of determination, but they're not entirely orthogonal. It would seem a misnomer if someone said they were very determined to do something trivially easy.

And fortunately ambition seems to be quite malleable; there's a lot you can do to increase it. Most people don't know how ambitious to be, especially when they're young. They don't know what's hard, or what they're capable of. And this problem is exacerbated by having few peers. Ambitious people are rare, so if everyone is mixed together randomly, as they tend to be early in people's lives, then the ambitious ones won't have many ambitious peers. When you take people like this and put them together with other ambitious people, they bloom like dying plants given water. Probably most ambitious people are starved for the sort of encouragement they'd get from ambitious peers, whatever their age. [2]

Achievements also tend to increase your ambition. With each step you gain confidence to stretch further next time.

So here in sum is how determination seems to work: it consists of willfulness balanced with discipline, aimed by ambition. And fortunately at least two of these three qualities can be cultivated. You may be able to increase your strength of will somewhat; you can definitely learn self-discipline; and almost everyone is practically malnourished when it comes to ambition.

I feel like I understand determination a bit better now. But only a bit: willfulness, discipline, and ambition are all concepts almost as complicated as determination. [3]

Note too that determination and talent are not the whole story. There's a third factor in achievement: how much you like the work. If you really [love](#) working on something, you don't need determination to drive you; it's what you'd do anyway. But most types of work have aspects one doesn't like, because most types of work consist of doing things for other people, and it's very unlikely that the tasks imposed by their needs will happen to align exactly with what you want to do.

Indeed, if you want to create the most [wealth](#), the way to do it is to focus more on

their needs than your interests, and make up the difference with determination.

## Notes

[1] Loosely speaking. What I'm claiming with the melon seed model is more like determination is proportionate to  $w d^m - k|w - d|^n$ , where  $w$  is will and  $d$  discipline.

[2] Which means one of the best ways to help a society generally is to create [events](#) and [institutions](#) that bring ambitious people together. It's like pulling the control rods out of a reactor: the energy they emit encourages other ambitious people, instead of being absorbed by the normal people they're usually surrounded with.

Conversely, it's probably a mistake to do as some European countries have done and try to ensure none of your universities is significantly better than the others.

[3] For example, willfulness clearly has two subcomponents, stubbornness and energy. The first alone yields someone who's stubbornly inert. The second alone yields someone flighty. As willful people get older or otherwise lose their energy, they tend to become merely stubborn.

**Thanks** to Sam Altman, Jessica Livingston, and Robert Morris for reading drafts of this.

[Italian Translation](#)

[Portuguese Translation](#)

[Russian Translation](#)

# The List of N Things

September 2009

I bet you the current issue of *Cosmopolitan* has an article whose title begins with a number. "7 Things He Won't Tell You about Sex," or something like that. Some popular magazines feature articles of this type on the cover of every issue. That can't be happening by accident. Editors must know they attract readers.

Why do readers like the list of n things so much? Mainly because it's easier to read than a regular article. [1] Structurally, the list of n things is a degenerate case of essay. An essay can go anywhere the writer wants. In a list of n things the writer agrees to constrain himself to a collection of points of roughly equal importance, and he tells the reader explicitly what they are.

Some of the work of reading an article is understanding its structure—figuring out what in high school we'd have called its "outline." Not explicitly, of course, but someone who really understands an article probably has something in his brain afterward that corresponds to such an outline. In a list of n things, this work is done for you. Its structure is an exoskeleton.

As well as being explicit, the structure is guaranteed to be of the simplest possible type: a few main points with few to no subordinate ones, and no particular connection between them.

Because the main points are unconnected, the list of n things is random access. There's no thread of reasoning you have to follow. You could read the list in any order. And because the points are independent of one another, they work like watertight compartments in an unsinkable ship. If you get bored with, or can't understand, or don't agree with one point, you don't have to give up on the article. You can just abandon that one and skip to the next. A list of n things is parallel and therefore fault tolerant.

There are times when this format is what a writer wants. One, obviously, is when what you have to say actually is a list of n things. I once wrote an essay about the [mistakes that kill startups](#), and a few people made fun of me for writing something whose title began with a number. But in that case I really was trying to make a complete catalog of a number of independent things. In fact, one of the questions I was trying to answer was how many there were.

There are other less legitimate reasons for using this format. For example, I use it when I get close to a deadline. If I have to give a talk and I haven't started it a few days beforehand, I'll sometimes play it safe and make the talk a list of  $n$  things.

The list of  $n$  things is easier for writers as well as readers. When you're writing a real essay, there's always a chance you'll hit a dead end. A real essay is a train of thought, and some trains of thought just peter out. That's an alarming possibility when you have to give a talk in a few days. What if you run out of ideas? The compartmentalized structure of the list of  $n$  things protects the writer from his own stupidity in much the same way it protects the reader. If you run out of ideas on one point, no problem: it won't kill the essay. You can take out the whole point if you need to, and the essay will still survive.

Writing a list of  $n$  things is so relaxing. You think of  $n/2$  of them in the first 5 minutes. So bang, there's the structure, and you just have to fill it in. As you think of more points, you just add them to the end. Maybe you take out or rearrange or combine a few, but at every stage you have a valid (though initially low-res) list of  $n$  things. It's like the sort of programming where you write a version 1 very quickly and then gradually modify it, but at every point have working code—or the style of painting where you begin with a complete but very blurry sketch done in an hour, then spend a week cranking up the resolution.

Because the list of  $n$  things is easier for writers too, it's not always a damning sign when readers prefer it. It's not necessarily evidence readers are lazy; it could also mean they don't have much confidence in the writer. The list of  $n$  things is in that respect the cheeseburger of essay forms. If you're eating at a restaurant you suspect is bad, your best bet is to order the cheeseburger. Even a bad cook can make a decent cheeseburger. And there are pretty strict conventions about what a cheeseburger should look like. You can assume the cook isn't going to try something weird and artistic. The list of  $n$  things similarly limits the damage that can be done by a bad writer. You know it's going to be about whatever the title says, and the format prevents the writer from indulging in any flights of fancy.

Because the list of  $n$  things is the easiest essay form, it should be a good one for beginning writers. And in fact it is what most beginning writers are taught. The classic 5 paragraph essay is really a list of  $n$  things for  $n = 3$ . But the students writing them don't realize they're using the same structure as the articles they read in *Cosmopolitan*. They're not allowed to include the numbers, and they're expected to spackle over the gaps with gratuitous transitions ("Furthermore...") and cap the thing at either end with introductory and concluding paragraphs so it will look superficially like a real essay. [2]

It seems a fine plan to start students off with the list of  $n$  things. It's the easiest form. But if we're going to do that, why not do it openly? Let them write lists of  $n$  things like the pros, with numbers and no transitions or "conclusion."

There is one case where the list of  $n$  things is a dishonest format: when you use it to attract attention by falsely claiming the list is an exhaustive one. I.e. if you write

an article that purports to be about *the* 7 secrets of success. That kind of title is the same sort of reflexive challenge as a whodunit. You have to at least look at the article to check whether they're the same 7 you'd list. Are you overlooking one of the secrets of success? Better check.

It's fine to put "The" before the number if you really believe you've made an exhaustive list. But evidence suggests most things with titles like this are linkbait.

The greatest weakness of the list of  $n$  things is that there's so little room for new thought. The main point of essay writing, when done right, is the new ideas you have while doing it. A real essay, as the name implies, is [dynamic](#): you don't know what you're going to write when you start. It will be about whatever you discover in the course of writing it.

This can only happen in a very limited way in a list of  $n$  things. You make the title first, and that's what it's going to be about. You can't have more new ideas in the writing than will fit in the watertight compartments you set up initially. And your brain seems to know this: because you don't have room for new ideas, you don't have them.

Another advantage of admitting to beginning writers that the 5 paragraph essay is really a list of  $n$  things is that we can warn them about this. It only lets you experience the defining characteristic of essay writing on a small scale: in thoughts of a sentence or two. And it's particularly dangerous that the 5 paragraph essay buries the list of  $n$  things within something that looks like a more sophisticated type of essay. If you don't know you're using this form, you don't know you need to escape it.

## Notes

[1] Articles of this type are also startlingly popular on Delicious, but I think that's because [delicious/popular](#) is driven by bookmarking, not because Delicious users are stupid. Delicious users are collectors, and a list of  $n$  things seems particularly collectible because it's a collection itself.

[2] Most "word problems" in school math textbooks are similarly misleading. They look superficially like the application of math to real problems, but they're not. So if anything they reinforce the impression that math is merely a complicated but pointless collection of stuff to be memorized.





# Post-Medium Publishing

September 2009

Publishers of all types, from news to music, are unhappy that consumers won't pay for content anymore. At least, that's how they see it.

In fact consumers never really were paying for content, and publishers weren't really selling it either. If the content was what they were selling, why has the price of books or music or movies always depended mostly on the format? Why didn't better content cost more? [\[1\]](#)

A copy of *Time* costs \$5 for 58 pages, or 8.6 cents a page. *The Economist* costs \$7 for 86 pages, or 8.1 cents a page. Better journalism is actually slightly cheaper.

Almost every form of publishing has been organized as if the medium was what they were selling, and the content was irrelevant. Book publishers, for example, set prices based on the cost of producing and distributing books. They treat the words printed in the book the same way a textile manufacturer treats the patterns printed on its fabrics.

Economically, the print media are in the business of marking up paper. We can all imagine an old-style editor getting a scoop and saying "this will sell a lot of papers!" Cross out that final S and you're describing their business model. The reason they make less money now is that people don't need as much paper.

A few months ago I ran into a friend in a cafe. I had a copy of the *New York Times*, which I still occasionally buy on weekends. As I was leaving I offered it to him, as I've done countless times before in the same situation. But this time something new happened. I felt that sheepish feeling you get when you offer someone something worthless. "Do you, er, want a printout of yesterday's news?" I asked. (He didn't.)

Now that the medium is evaporating, publishers have nothing left to sell. Some seem to think they're going to sell content—that they were always in the content business, really. But they weren't, and it's unclear whether anyone could be.

## Selling

There have always been people in the business of selling information, but that has

historically been a distinct business from publishing. And the business of selling information to consumers has always been a marginal one. When I was a kid there were people who used to sell newsletters containing stock tips, printed on colored paper that made them hard for the copiers of the day to reproduce. That is a different world, both culturally and economically, from the one publishers currently inhabit.

People will pay for information they think they can make money from. That's why they paid for those stock tip newsletters, and why companies pay now for Bloomberg terminals and Economist Intelligence Unit reports. But will people pay for information otherwise? History offers little encouragement.

If audiences were willing to pay more for better content, why wasn't anyone already selling it to them? There was no reason you couldn't have done that in the era of physical media. So were the print media and the music labels simply overlooking this opportunity? Or is it, rather, nonexistent?

What about iTunes? Doesn't that show people will pay for content? Well, not really. iTunes is more of a tollbooth than a store. Apple controls the default path onto the iPod. They offer a convenient list of songs, and whenever you choose one they ding your credit card for a small amount, just below the threshold of attention. Basically, iTunes makes money by taxing people, not selling them stuff. You can only do that if you own the channel, and even then you don't make much from it, because a toll has to be ignorable to work. Once a toll becomes painful, people start to find ways around it, and that's pretty easy with digital content.

The situation is much the same with digital books. Whoever controls the device sets the terms. It's in their interest for content to be as cheap as possible, and since they own the channel, there's a lot they can do to drive prices down. Prices will fall even further once writers realize they don't need publishers. Getting a book printed and distributed is a daunting prospect for a writer, but most can upload a file.

Is software a counterexample? People pay a lot for desktop software, and that's just information. True, but I don't think publishers can learn much from software. Software companies can charge a lot because (a) many of the customers are businesses, who get in [trouble](#) if they use pirated versions, and (b) though in form merely information, software is treated by both maker and purchaser as a different type of thing from a song or an article. A Photoshop user needs Photoshop in a way that no one needs a particular song or article.

That's why there's a separate word, "content," for information that's not software. Software is a different business. Software and content blur together in some of the most lightweight software, like casual games. But those are usually free. To make money the way software companies do, publishers would have to become software companies, and being publishers gives them no particular head start in that domain. [2]

The most promising countertrend is the premium cable channel. People still pay for those. But broadcasting isn't publishing: you're not selling a copy of something. That's one reason the movie business hasn't seen their revenues decline the way the news and music businesses have. They only have one foot in publishing.

To the extent the movie business can avoid becoming publishers, they may avoid publishing's problems. But there are limits to how well they'll be able to do that. Once publishing—giving people copies—becomes the most natural way of distributing your content, it probably doesn't work to stick to old forms of distribution just because you make more that way. If free copies of your content are available online, then you're competing with publishing's form of distribution, and that's just as bad as being a publisher.

Apparently some people in the music business hope to retroactively convert it away from publishing, by getting listeners to pay for subscriptions. It seems unlikely that will work if they're just streaming the same files you can get as mp3s.

## Next

What happens to publishing if you can't sell content? You have two choices: give it away and make money from it indirectly, or find ways to embody it in things people will pay for.

The first is probably the future of most current media. [Give music away](#) and make money from concerts and t-shirts. Publish articles for free and make money from one of a dozen permutations of advertising. Both publishers and investors are down on advertising at the moment, but it has more potential than they realize.

I'm not claiming that potential will be realized by the existing players. The [optimal](#) ways to make money from the written word probably require different words written by different people.

It's harder to say what will happen to movies. They could evolve into ads. Or they could return to their roots and make going to the theater a treat. If they made the experience good enough, audiences might start to prefer it to watching pirated movies at home. [3] Or maybe the movie business will dry up, and the people working in it will go to work for game developers.

I don't know how big embodying information in physical form will be. It may be surprisingly large; people overvalue [physical stuff](#). There should remain some market for printed books, at least.

I can see the evolution of book publishing in the books on my shelves. Clearly at some point in the 1960s the big publishing houses started to ask: how cheaply can we make books before people refuse to buy them? The answer turned out to be one step short of phonebooks. As long as it isn't floppy, consumers still perceive it as a book.

That worked as long as buying printed books was the only way to read them. If printed books are optional, publishers will have to work harder to entice people to buy them. There should be some market, but it's hard to foresee how big, because its size will depend not on macro trends like the amount people read, but on the ingenuity of individual publishers. [4]

Some magazines may thrive by focusing on the magazine as a physical object. Fashion magazines could be made lush in a way that would be hard to match digitally, at least for a while. But this is probably not an option for most magazines.

I don't know exactly what the future will look like, but I'm not too worried about it. This sort of change tends to create as many good things as it kills. Indeed, the really interesting question is not what will happen to existing forms, but what new forms will appear.

The reason I've been writing about existing forms is that I don't *know* what new forms will appear. But though I can't predict specific winners, I can offer a recipe for recognizing them. When you see something that's taking advantage of new technology to give people something they want that they couldn't have before, you're probably looking at a winner. And when you see something that's merely reacting to new technology in an attempt to preserve some existing source of revenue, you're probably looking at a loser.

## Notes

[1] I don't like the word "content" and tried for a while to avoid using it, but I have to admit there's no other word that means the right thing. "Information" is too general.

Ironically, the main reason I don't like "content" is the thesis of this essay. The word suggests an undifferentiated slurry, but economically that's how both publishers and audiences treat it. Content is information you don't need.

[2] Some types of publishers would be at a disadvantage trying to enter the software business. Record labels, for example, would probably find it more natural to expand into casinos than software, because the kind of people who run them would be more at home at the mafia end of the business spectrum than the don't-be-evil end.

[3] I never watch movies in theaters anymore. The tipping point for me was the ads they show first.

[4] Unfortunately, making physically nice books will only be a niche within a niche. Publishers are more likely to resort to expedients like selling autographed copies,

or editions with the buyer's picture on the cover.

**Thanks** to Michael Arrington, Trevor Blackwell, Steven Levy, Robert Morris, and Geoff Ralston for reading drafts of this.

# Persuade xor Discover

September 2009

When meeting people you don't know very well, the convention is to seem extra friendly. You smile and say "pleased to meet you," whether you are or not. There's nothing dishonest about this. Everyone knows that these little social lies aren't meant to be taken literally, just as everyone knows that "Can you pass the salt?" is only grammatically a question.

I'm perfectly willing to smile and say "pleased to meet you" when meeting new people. But there is another set of customs for being ingratiating in print that are not so harmless.

The reason there's a convention of being ingratiating in print is that most essays are written to persuade. And as any politician could tell you, the way to persuade people is not just to baldly state the facts. You have to add a spoonful of sugar to make the medicine go down.

For example, a politician announcing the cancellation of a government program will not merely say "The program is canceled." That would seem offensively curt. Instead he'll spend most of his time talking about the noble effort made by the people who worked on it.

The reason these conventions are more dangerous is that they interact with the ideas. Saying "pleased to meet you" is just something you prepend to a conversation, but the sort of spin added by politicians is woven through it. We're starting to move from social lies to real lies.

Here's an example of a paragraph from an essay I wrote about [labor unions](#). As written, it tends to offend people who like unions.

People who think the labor movement was the creation of heroic union organizers have a problem to explain: why are unions shrinking now? The best they can do is fall back on the default explanation of people living in fallen civilizations. Our ancestors were giants. The workers of the early twentieth century must have had a moral courage that's lacking today.

Now here's the same paragraph rewritten to please instead of offending them:

Early union organizers made heroic sacrifices to improve conditions for workers. But though labor unions are shrinking now, it's not because present union leaders are any less courageous. An employer couldn't get away with hiring thugs to beat up union leaders today, but if they did, I see no reason to believe today's union leaders would shrink from the challenge. So I think it would be a mistake to attribute the decline of unions to some kind of decline in the people who run them. Early union leaders were heroic, certainly, but we should not suppose that if unions have declined, it's because present union leaders are somehow inferior. The cause must be external. [1]

It makes the same point: that it can't have been the personal qualities of early union organizers that made unions successful, but must have been some external factor, or otherwise present-day union leaders would have to be inferior people. But written this way it seems like a defense of present-day union organizers rather than an attack on early ones. That makes it more persuasive to people who like unions, because it seems sympathetic to their cause.

I believe everything I wrote in the second version. Early union leaders did make heroic sacrifices. And present union leaders probably would rise to the occasion if necessary. People tend to; I'm skeptical about the idea of "the greatest generation." [2]

If I believe everything I said in the second version, why didn't I write it that way? Why offend people needlessly?

Because I'd rather offend people than pander to them, and if you write about controversial topics you have to choose one or the other. The degree of courage of past or present union leaders is beside the point; all that matters for the argument is that they're the same. But if you want to please people who are mistaken, you can't simply tell the truth. You're always going to have to add some sort of padding to protect their misconceptions from bumping against reality.

Most writers do. Most writers write to persuade, if only out of habit or politeness. But I don't write to persuade; I write to figure out. I write to persuade a hypothetical perfectly unbiased reader.

Since the custom is to write to persuade the actual reader, someone who doesn't will seem arrogant. In fact, worse than arrogant: since readers are used to essays that try to please someone, an essay that displeases one side in a dispute reads as an attempt to pander to the other. To a lot of pro-union readers, the first paragraph sounds like the sort of thing a right-wing radio talk show host would say to stir up his followers. But it's not. Something that curtly contradicts one's beliefs can be hard to distinguish from a partisan attack on them, but though they can end up in the same place they come from different sources.

Would it be so bad to add a few extra words, to make people feel better? Maybe

not. Maybe I'm excessively attached to conciseness. I write [code](#) the same way I write essays, making pass after pass looking for anything I can cut. But I have a legitimate reason for doing this. You don't know what the ideas are until you get them down to the fewest words. [3]

The danger of the second paragraph is not merely that it's longer. It's that you start to lie to yourself. The ideas start to get mixed together with the spin you've added to get them past the readers' misconceptions.

I think the goal of an essay should be to discover [surprising](#) things. That's my goal, at least. And most surprising means most different from what people currently believe. So writing to persuade and writing to discover are diametrically opposed. The more your conclusions disagree with readers' present beliefs, the more effort you'll have to expend on selling your ideas rather than having them. As you accelerate, this drag increases, till eventually you reach a point where 100% of your energy is devoted to overcoming it and you can't go any faster.

It's hard enough to overcome one's own misconceptions without having to think about how to get the resulting ideas past other people's. I worry that if I wrote to persuade, I'd start to shy away unconsciously from ideas I knew would be hard to sell. When I notice something surprising, it's usually very faint at first. There's nothing more than a slight stirring of discomfort. I don't want anything to get in the way of noticing it consciously.

## Notes

[1] I had a strange feeling of being back in high school writing this. To get a good grade you had to both write the sort of pious crap you were expected to, but also seem to be writing with conviction. The solution was a kind of method acting. It was revoltingly familiar to slip back into it.

[2] Exercise for the reader: rephrase that thought to please the same people the first version would offend.

[3] Come to think of it, there is one way in which I deliberately pander to readers, because it doesn't change the number of words: I switch person. This flattering distinction seems so natural to the average reader that they probably don't notice even when I switch in mid-sentence, though you tend to notice when it's done as conspicuously as this.

**Thanks** to Jessica Livingston and Robert Morris for reading drafts of this.



**Note:** An earlier version of this essay began by talking about why people dislike Michael Arrington. I now believe that was mistaken, and that most people don't dislike him for the same reason I did when I first met him, but simply because he writes about controversial things.

# What Startups Are Really Like

October 2009

*(This essay is derived from a talk at the 2009 Startup School.)*

I wasn't sure what to talk about at Startup School, so I decided to ask the founders of the startups we'd funded. What hadn't I written about yet?

I'm in the unusual position of being able to test the essays I write about startups. I hope the ones on other topics are right, but I have no way to test them. The ones on startups get tested by about 70 people every 6 months.

So I sent all the founders an email asking what surprised them about starting a startup. This amounts to asking what I got wrong, because if I'd explained things well enough, nothing should have surprised them.

I'm proud to report I got one response saying:

What surprised me the most is that everything was actually fairly predictable!

The bad news is that I got over 100 other responses listing the surprises they encountered.

There were very clear patterns in the responses; it was remarkable how often several people had been surprised by exactly the same thing. These were the biggest:

## **1. Be Careful with Cofounders**

This was the surprise mentioned by the most founders. There were two types of responses: that you have to be careful who you pick as a cofounder, and that you have to work hard to maintain your relationship.

What people wished they'd paid more attention to when choosing cofounders was character and commitment, not ability. This was particularly true with startups that failed. The lesson: don't pick cofounders who will flake.

Here's a typical reponse:

You haven't seen someone's true colors unless you've worked with them on a startup.

The reason character is so important is that it's tested more severely than in most other situations. One founder said explicitly that the relationship between founders was more important than ability:

I would rather cofound a startup with a friend than a stranger with higher output. Startups are so hard and emotional that the bonds and emotional and social support that come with friendship outweigh the extra output lost.

We learned this lesson a long time ago. If you look at the YC application, there are more questions about the commitment and relationship of the founders than their ability.

Founders of successful startups talked less about choosing cofounders and more about how hard they worked to maintain their relationship.

One thing that surprised me is how the relationship of startup founders goes from a friendship to a marriage. My relationship with my cofounder went from just being friends to seeing each other all the time, fretting over the finances and cleaning up shit. And the startup was our baby. I summed it up once like this: "It's like we're married, but we're not fucking."

Several people used that word "married." It's a far more intense relationship than you usually see between coworkers—partly because the stresses are so much greater, and partly because at first the founders are the whole company. So this relationship has to be built of top quality materials and carefully maintained. It's the basis of everything.

## **2. Startups Take Over Your Life**

Just as the relationship between cofounders is more intense than it usually is between coworkers, so is the relationship between the founders and the company. Running a startup is not like having a job or being a student, because it never stops. This is so foreign to most people's experience that they don't get it till it happens. [\[1\]](#)

I didn't realize I would spend almost every waking moment either working or thinking about our startup. You enter a whole different way of life when it's your company vs. working for someone else's company.

It's exacerbated by the fast pace of startups, which makes it seem like time slows down:

I think the thing that's been most surprising to me is how one's perspective on time shifts. Working on our startup, I remember time

seeming to stretch out, so that a month was a huge interval. In the best case, total immersion can be exciting:

It's surprising how much you become consumed by your startup, in that you think about it day and night, but never once does it feel like "work."

Though I have to say, that quote is from someone we funded this summer. In a couple years he may not sound so chipper.

### **3. It's an Emotional Roller-coaster**

This was another one lots of people were surprised about. The ups and downs were more extreme than they were prepared for.

In a startup, things seem great one moment and hopeless the next. And by next, I mean a couple hours later.

The emotional ups and downs were the biggest surprise for me. One day, we'd think of ourselves as the next Google and dream of buying islands; the next, we'd be pondering how to let our loved ones know of our utter failure; and on and on.

The hard part, obviously, is the lows. For a lot of founders that was the big surprise:

How hard it is to keep everyone motivated during rough days or weeks, i.e. how low the lows can be.

After a while, if you don't have significant success to cheer you up, it wears you out:

Your most basic advice to founders is "just don't die," but the energy to keep a company going in lieu of unburdening success isn't free; it is siphoned from the founders themselves.

There's a limit to how much you can take. If you get to the point where you can't keep working anymore, it's not the end of the world. Plenty of famous founders have had some failures along the way.

### **4. It Can Be Fun**

The good news is, the highs are also very high. Several founders said what surprised them most about doing a startup was how fun it was:

I think you've left out just how fun it is to do a startup. I am more fulfilled in my work than pretty much any of my friends who did not start companies.

What they like most is the freedom:

I'm surprised by how much better it feels to be working on something that is challenging and creative, something I believe in, as opposed to

the hired-gun stuff I was doing before. I knew it would feel better; what's surprising is how much better.

Frankly, though, if I've misled people here, I'm not eager to fix that. I'd rather have everyone think starting a startup is grim and hard than have founders go into it expecting it to be fun, and a few months later saying "This is supposed to be *fun*? Are you kidding?"

The truth is, it wouldn't be fun for most people. A lot of what we try to do in the application process is to weed out the people who wouldn't like it, both for our sake and theirs.

The best way to put it might be that starting a startup is fun the way a survivalist training course would be fun, if you're into that sort of thing. Which is to say, not at all, if you're not.

## **5. Persistence Is the Key**

A lot of founders were surprised how important persistence was in startups. It was both a negative and a positive surprise: they were surprised both by the degree of persistence required

Everyone said how determined and resilient you must be, but going through it made me realize that the determination required was still understated.

and also by the degree to which persistence alone was able to dissolve obstacles:

If you are persistent, even problems that seem out of your control (i.e. immigration) seem to work themselves out.

Several founders mentioned specifically how much more important persistence was than intelligence.

I've been surprised again and again by just how much more important persistence is than raw intelligence.

This applies not just to intelligence but to ability in general, and that's why so many people said character was more important in choosing cofounders.

## **6. Think Long-Term**

You need persistence because everything takes longer than you expect. A lot of people were surprised by that.

I'm continually surprised by how long everything can take. Assuming your product doesn't experience the explosive growth that very few products do, everything from development to dealmaking (especially dealmaking) seems to take 2-3x longer than I always imagine.

One reason founders are surprised is that because they work fast, they expect everyone else to. There's a shocking amount of shear stress at every point where a

startup touches a more bureaucratic organization, like a big company or a VC fund. That's why fundraising and the enterprise market kill and maim so many startups.

[2]

But I think the reason most founders are surprised by how long it takes is that they're overconfident. They think they're going to be an instant success, like YouTube or Facebook. You tell them only 1 out of 100 successful startups has a trajectory like that, and they all think "we're going to be that 1."

Maybe they'll listen to one of the more successful founders:

The top thing I didn't understand before going into it is that persistence is the name of the game. For the vast majority of startups that become successful, it's going to be a *really* long journey, at least 3 years and probably 5+.

There is a positive side to thinking longer-term. It's not just that you have to resign yourself to everything taking longer than it should. If you work patiently it's less stressful, and you can do better work:

Because we're relaxed, it's so much easier to have fun doing what we do. Gone is the awkward nervous energy fueled by the desperate need to not fail guiding our actions. We can concentrate on doing what's best for our company, product, employees and customers.

That's why things get so much better when you hit ramen profitability. You can shift into a different mode of working.

## **7. Lots of Little Things**

We often emphasize how rarely startups win simply because they hit on some magic idea. I think founders have now gotten that into their heads. But a lot were surprised to find this also applies within startups. You have to do lots of different things:

It's much more of a grind than glamorous. A timeslice selected at random would more likely find me tracking down a weird DLL loading bug on Swedish Windows, or tracking down a bug in the financial model Excel spreadsheet the night before a board meeting, rather than having brilliant flashes of strategic insight.

Most hacker-founders would like to spend all their time programming. You won't get to, unless you fail. Which can be transformed into: If you spend all your time programming, you will fail.

The principle extends even into programming. There is rarely a single brilliant hack that ensures success:

I learnt never to bet on any one feature or deal or anything to bring you success. It is never a single thing. Everything is just incremental

and you just have to keep doing lots of those things until you strike something.

Even in the rare cases where a clever hack makes your fortune, you probably won't know till later:

There is no such thing as a killer feature. Or at least you won't know what it is.

So the best strategy is to try lots of different things. The reason not to put all your eggs in one basket is not the usual one, which applies even when you know which basket is best. In a startup you don't even know that.

## **8. Start with Something Minimal**

Lots of founders mentioned how important it was to launch with the simplest possible thing. By this point everyone knows you should release fast and iterate. It's practically a mantra at YC. But even so a lot of people seem to have been burned by not doing it:

Build the absolute smallest thing that can be considered a complete application and ship it.

Why do people take too long on the first version? Pride, mostly. They hate to release something that could be better. They worry what people will say about them. But you have to overcome this:

Doing something "simple" at first glance does not mean you aren't doing something meaningful, defensible, or valuable.

Don't worry what people will say. If your first version is so impressive that trolls don't make fun of it, you waited too long to launch. [\[3\]](#)

One founder said this should be your approach to all programming, not just startups, and I tend to agree.

Now, when coding, I try to think "How can I write this such that if people saw my code, they'd be amazed at how little there is and how little it does?"

Over-engineering is poison. It's not like doing extra work for extra credit. It's more like telling a lie that you then have to remember so you don't contradict it.

## **9. Engage Users**

Product development is a conversation with the user that doesn't really start till you launch. Before you launch, you're like a police artist before he's shown the first version of his sketch to the witness.

It's so important to launch fast that it may be better to think of your initial version

not as a product, but as a trick for getting users to start talking to you.

I learned to think about the initial stages of a startup as a giant experiment. All products should be considered experiments, and those that have a market show promising results extremely quickly.

Once you start talking to users, I guarantee you'll be surprised by what they tell you.

When you let customers tell you what they're after, they will often reveal amazing details about what they find valuable as well what they're willing to pay for.

The surprise is generally positive as well as negative. They won't like what you've built, but there will be other things they would like that would be trivially easy to implement. It's not till you start the conversation by launching the wrong thing that they can express (or perhaps even realize) what they're looking for.

## **10. Change Your Idea**

To benefit from engaging with users you have to be willing to change your idea. We've always encouraged founders to see a startup idea as a hypothesis rather than a blueprint. And yet they're still surprised how well it works to change the idea.

Normally if you complain about something being hard, the general advice is to work harder. With a startup, I think you should find a problem that's easy for you to solve. Optimizing in solution-space is familiar and straightforward, but you can make enormous gains playing around in problem-space.

Whereas mere determination, without flexibility, is a greedy algorithm that may get you nothing more than a mediocre local maximum:

When someone is determined, there's still a danger that they'll follow a long, hard path that ultimately leads nowhere.

You want to push forward, but at the same time twist and turn to find the most promising path. One founder put it very succinctly:

Fast iteration is the key to success.

One reason this advice is so hard to follow is that people don't realize how hard it is to judge startup ideas, particularly their own. Experienced founders learn to keep an open mind:

Now I don't laugh at ideas anymore, because I realized how terrible I was at knowing if they were good or not.

You can never tell what will work. You just have to do whatever seems best at each point. We do this with YC itself. We still don't know if it will work, but it seems like a decent hypothesis.



## 11. Don't Worry about Competitors

When you think you've got a great idea, it's sort of like having a guilty conscience about something. All someone has to do is look at you funny, and you think "Oh my God, *they know*."

These alarms are almost always false:

Companies that seemed like competitors and threats at first glance usually never were when you really looked at it. Even if they were operating in the same area, they had a different goal.

One reason people overreact to competitors is that they overvalue ideas. If ideas really were the key, a competitor with the same idea would be a real threat. But it's usually execution that matters:

All the scares induced by seeing a new competitor pop up are forgotten weeks later. It always comes down to your own product and approach to the market.

This is generally true even if competitors get lots of attention.

Competitors riding on lots of good blogger perception aren't really the winners and can disappear from the map quickly. You need consumers after all.

Hype doesn't make satisfied users, at least not for something as complicated as technology.

## 12. It's Hard to Get Users

A lot of founders complained about how hard it was to get users, though.

I had no idea how much time and effort needed to go into attaining users.

This is a complicated topic. When you can't get users, it's hard to say whether the problem is lack of exposure, or whether the product's simply bad. Even good products can be blocked by switching or integration costs:

Getting people to use a new service is incredibly difficult. This is especially true for a service that other companies can use, because it requires their developers to do work. If you're small, they don't think it is urgent. [\[4\]](#)

The sharpest criticism of YC came from a founder who said we didn't focus enough on customer acquisition:

YC preaches "make something people want" as an engineering task, a never ending stream of feature after feature until enough people are happy and the application takes off. There's very little focus on the cost of customer acquisition.

This may be true; this may be something we need to fix, especially for applications like games. If you make something where the challenges are mostly technical, you can rely on word of mouth, like Google did. One founder was surprised by how well that worked for him:

There is an irrational fear that no one will buy your product. But if you work hard and incrementally make it better, there is no need to worry.

But with other types of startups you may win less by features and more by deals and marketing.

### **13. Expect the Worst with Deals**

Deals fall through. That's a constant of the startup world. Startups are powerless, and good startup ideas generally seem wrong. So everyone is nervous about closing deals with you, and you have no way to make them.

This is particularly true with investors:

In retrospect, it would have been much better if we had operated under the assumption that we would never get any additional outside investment. That would have focused us on finding revenue streams early.

My advice is generally pessimistic. Assume you won't get money, and if someone does offer you any, assume you'll never get any more.

If someone offers you money, take it. You say it a lot, but I think it needs even more emphasizing. We had the opportunity to raise a lot more money than we did last year and I wish we had.

Why do founders ignore me? Mostly because they're optimistic by nature. The mistake is to be optimistic about things you can't control. By all means be optimistic about your ability to make something great. But you're asking for trouble if you're optimistic about big companies or investors.

### **14. Investors Are Clueless**

A lot of founders mentioned how surprised they were by the cluelessness of investors:

They don't even know about the stuff they've invested in. I met some investors that had invested in a hardware device and when I asked them to demo the device they had difficulty switching it on.

Angels are a bit better than VCs, because they usually have startup experience themselves:

VC investors don't know half the time what they are talking about and are years behind in their thinking. A few were great, but 95% of the investors we dealt with were unprofessional, didn't seem to be very

good at business or have any kind of creative vision. Angels were generally much better to talk to.

Why are founders surprised that VCs are clueless? I think it's because they seem so formidable.

The reason VCs seem formidable is that it's their profession to. You get to be a VC by convincing asset managers to trust you with hundreds of millions of dollars. How do you do that? You have to seem confident, and you have to seem like you understand technology. [\[5\]](#)

## 15. You May Have to Play Games

Because investors are so bad at judging you, you have to work harder than you should at selling yourself. One founder said the thing that surprised him most was

The degree to which feigning certitude impressed investors.

This is the thing that has surprised *me* most about YC founders' experiences. This summer we invited some of the alumni to talk to the new startups about fundraising, and pretty much 100% of their advice was about investor psychology. I thought I was cynical about VCs, but the founders were much more cynical.

A lot of what startup founders do is just posturing. It works.

VCs themselves have no idea of the extent to which the startups they like are the ones that are best at selling themselves to VCs. [\[6\]](#) It's exactly the same phenomenon we saw a step earlier. VCs get money by seeming confident to LPs, and founders get money by seeming confident to VCs.

## 16. Luck Is a Big Factor

With two such random linkages in the path between startups and money, it shouldn't be surprising that luck is a big factor in deals. And yet a lot of founders are surprised by it.

I didn't realize how much of a role luck plays and how much is outside of our control.

If you think about famous startups, it's pretty clear how big a role luck plays. Where would Microsoft be if IBM insisted on an exclusive license for DOS?

Why are founders fooled by this? Business guys probably aren't, but hackers are used to a world where skill is paramount, and you get what you deserve.

When we started our startup, I had bought the hype of the startup founder dream: that this is a game of skill. It is, in some ways. Having skill is valuable. So is being determined as all hell. But being lucky is the critical ingredient.

Actually the best model would be to say that the outcome is the *product* of skill, determination, and luck. No matter how much skill and determination you have, if

you roll a zero for luck, the outcome is zero.

These quotes about luck are not from founders whose startups failed. Founders who fail quickly tend to blame themselves. Founders who succeed quickly don't usually realize how lucky they were. It's the ones in the middle who see how important luck is.

## **17. The Value of Community**

A surprising number of founders said what surprised them most about starting a startup was the value of community. Some meant the micro-community of YC founders:

The immense value of the peer group of YC companies, and facing similar obstacles at similar times.

which shouldn't be that surprising, because that's why it's structured that way. Others were surprised at the value of the startup community in the larger sense:

How advantageous it is to live in Silicon Valley, where you can't help but hear all the cutting-edge tech and startup news, and run into useful people constantly.

The specific thing that surprised them most was the general spirit of benevolence:

One of the most surprising things I saw was the willingness of people to help us. Even people who had nothing to gain went out of their way to help our startup succeed.

and particularly how it extended all the way to the top:

The surprise for me was how accessible important and interesting people are. It's amazing how easily you can reach out to people and get immediate feedback.

This is one of the reasons I like being part of this world. Creating wealth is not a zero-sum game, so you don't have to stab people in the back to win.

## **18. You Get No Respect**

There was one surprise founders mentioned that I'd forgotten about: that outside the startup world, startup founders get no respect.

In social settings, I found that I got a lot more respect when I said, "I worked on Microsoft Office" instead of "I work at a small startup you've never heard of called x."

Partly this is because the rest of the world just doesn't get startups, and partly it's yet another consequence of the fact that most good startup ideas seem bad:

If you pitch your idea to a random person, 95% of the time you'll find the person instinctively thinks the idea will be a flop and you're

wasting your time (although they probably won't say this directly). Unfortunately this extends even to dating:

It surprised me that being a startup founder does not get you more admiration from women.

I did know about that, but I'd forgotten.

## **19. Things Change as You Grow**

The last big surprise founders mentioned is how much things changed as they grew. The biggest change was that you got to program even less:

Your job description as technical founder/CEO is completely rewritten every 6-12 months. Less coding, more managing/planning/company building, hiring, cleaning up messes, and generally getting things in place for what needs to happen a few months from now.

In particular, you now have to deal with employees, who often have different motivations:

I knew the founder equation and had been focused on it since I knew I wanted to start a startup as a 19 year old. The employee equation is quite different so it took me a while to get it down.

Fortunately, it can become a lot less stressful once you reach cruising altitude:

I'd say 75% of the stress is gone now from when we first started. Running a business is so much more enjoyable now. We're more confident. We're more patient. We fight less. We sleep more.

I wish I could say it was this way for every startup that succeeded, but 75% is probably on the high side.

## **The Super-Pattern**

There were a few other patterns, but these were the biggest. One's first thought when looking at them all is to ask if there's a super-pattern, a pattern to the patterns.

I saw it immediately, and so did a YC founder I read the list to. These are supposed to be the surprises, the things I didn't tell people. What do they all have in common? They're all things I tell people. If I wrote a new essay with the same outline as this that wasn't summarizing the founders' responses, everyone would say I'd run out of ideas and was just repeating myself.

What is going on here?

When I look at the responses, the common theme is that starting a startup was like I said, but way more so. People just don't seem to get how different it is till they do it. Why? The key to that mystery is to ask, how different *from what*? Once

you phrase it that way, the answer is obvious: from a job. Everyone's model of work is a job. It's completely pervasive. Even if you've never had a job, your parents probably did, along with practically every other adult you've met.

Unconsciously, everyone expects a startup to be like a job, and that explains most of the surprises. It explains why people are surprised how carefully you have to choose cofounders and how hard you have to work to maintain your relationship. You don't have to do that with coworkers. It explains why the ups and downs are surprisingly extreme. In a job there is much more damping. But it also explains why the good times are surprisingly good: most people can't imagine such freedom. As you go down the list, almost all the surprises are surprising in how much a startup differs from a job.

You probably can't overcome anything so pervasive as the model of work you grew up with. So the best solution is to be consciously aware of that. As you go into a startup, you'll be thinking "everyone says it's really extreme." Your next thought will probably be "but I can't believe it will be that bad." If you want to avoid being surprised, the next thought after that should be: "and the reason I can't believe it will be that bad is that my model of work is a job."

## Notes

[1] Graduate students might understand it. In grad school you always feel you should be working on your thesis. It doesn't end every semester like classes do.

[2] The best way for a startup to engage with slow-moving organizations is to fork off separate processes to deal with them. It's when they're on the critical path that they kill you—when you depend on closing a deal to move forward. It's worth taking extreme measures to avoid that.

[3] This is a variant of Reid Hoffman's principle that if you aren't embarrassed by what you launch with, you waited too long to launch.

[4] The question to ask about what you've built is not whether it's good, but whether it's good enough to supply the activation energy required.

[5] Some VCs seem to understand technology because they actually do, but that's overkill; the defining test is whether you can talk about it well enough to convince limited partners.

[6] This is the same phenomenon you see with defense contractors or fashion brands. The dumber the customers, the more effort you expend on the process of

selling things to them rather than making the things you sell.

**Thanks:** to Jessica Livingston for reading drafts of this, and to all the founders who responded to my email.

## **Related:**

[Startups in 13 Sentences](#)

[The Hardest Lessons for Startups to Learn](#)

[How Not to Die](#)

[The 18 Mistakes That Kill Startups](#)

[A Fundraising Survival Guide](#)

[Russian Translation](#)

[Korean Translation](#)

[Hebrew Translation](#)

# Apple's Mistake

November 2009

I don't think Apple realizes how badly the App Store approval process is broken. Or rather, I don't think they realize how much it matters that it's broken.

The way Apple runs the App Store has harmed their reputation with programmers more than anything else they've ever done. Their reputation with programmers used to be great. It used to be the most common complaint you heard about Apple was that their fans admired them too uncritically. The App Store has changed that. Now a lot of programmers have started to see Apple as evil.

How much of the goodwill Apple once had with programmers have they lost over the App Store? A third? Half? And that's just so far. The App Store is an ongoing karma leak.

\* \* \*

How did Apple get into this mess? Their fundamental problem is that they don't understand software.

They treat iPhone apps the way they treat the music they sell through iTunes. Apple is the channel; they own the user; if you want to reach users, you do it on their terms. The record labels agreed, reluctantly. But this model doesn't work for software. It doesn't work for an intermediary to own the user. The software business learned that in the early 1980s, when companies like VisiCorp showed that although the words "software" and "publisher" fit together, the underlying concepts don't. Software isn't like music or books. It's too complicated for a third party to act as an intermediary between developer and user. And yet that's what Apple is trying to be with the App Store: a software publisher. And a particularly overreaching one at that, with fussy tastes and a rigidly enforced house style.

If software publishing didn't work in 1980, it works even less now that software development has evolved from a small number of big releases to a constant stream of small ones. But Apple doesn't understand that either. Their model of product development derives from hardware. They work on something till they think it's finished, then they release it. You have to do that with hardware, but because software is so easy to change, its design can benefit from evolution. The



standard way to develop applications now is to launch fast and iterate. Which means it's a disaster to have long, random delays each time you release a new version.

Apparently Apple's attitude is that developers should be more careful when they submit a new version to the App Store. They would say that. But powerful as they are, they're not powerful enough to turn back the evolution of technology. Programmers don't use launch-fast-and-iterate out of laziness. They use it because it yields the best results. By obstructing that process, Apple is making them do bad work, and programmers hate that as much as Apple would.

How would Apple like it if when they discovered a serious bug in OS X, instead of releasing a software update immediately, they had to submit their code to an intermediary who sat on it for a month and then rejected it because it contained an icon they didn't like?

By breaking software development, Apple gets the opposite of what they intended: the version of an app currently available in the App Store tends to be an old and buggy one. One developer told me:

As a result of their process, the App Store is full of half-baked applications. I make a new version almost every day that I release to beta users. The version on the App Store feels old and crappy. I'm sure that a lot of developers feel this way: One emotion is "I'm not really proud about what's in the App Store", and it's combined with the emotion "Really, it's Apple's fault."

Another wrote:

I believe that they think their approval process helps users by ensuring quality. In reality, bugs like ours get through all the time and then it can take 4-8 weeks to get that bug fix approved, leaving users to think that iPhone apps sometimes just don't work. Worse for Apple, these apps work just fine on other platforms that have immediate approval processes.

Actually I suppose Apple has a third misconception: that all the complaints about App Store approvals are not a serious problem. They must hear developers complaining. But partners and suppliers are always complaining. It would be a bad sign if they weren't; it would mean you were being too easy on them. Meanwhile the iPhone is selling better than ever. So why do they need to fix anything?

They get away with maltreating developers, in the short term, because they make such great hardware. I just bought a new 27" iMac a couple days ago. It's fabulous. The screen's too shiny, and the disk is surprisingly loud, but it's so beautiful that you can't make yourself care.

So I bought it, but I bought it, for the first time, with misgivings. I felt the way I'd feel buying something made in a country with a bad human rights record. That was new. In the past when I bought things from Apple it was an unalloyed pleasure. Oh

boy! They make such great stuff. This time it felt like a Faustian bargain. They make such great stuff, but they're such assholes. Do I really want to support this company?

\* \* \*

Should Apple care what people like me think? What difference does it make if they alienate a small minority of their users?

There are a couple reasons they should care. One is that these users are the people they want as employees. If your company seems evil, the best programmers won't work for you. That hurt Microsoft a lot starting in the 90s. Programmers started to feel sheepish about working there. It seemed like selling out. When people from Microsoft were talking to other programmers and they mentioned where they worked, there were a lot of self-deprecating jokes about having gone over to the dark side. But the real problem for Microsoft wasn't the embarrassment of the people they hired. It was the people they never got. And you know who got them? Google and Apple. If Microsoft was the Empire, they were the Rebel Alliance. And it's largely because they got more of the best people that Google and Apple are doing so much better than Microsoft today.

Why are programmers so fussy about their employers' morals? Partly because they can afford to be. The best programmers can work wherever they want. They don't have to work for a company they have qualms about.

But the other reason programmers are fussy, I think, is that evil begets stupidity. An organization that wins by exercising power starts to lose the ability to win by doing better work. And it's not fun for a smart person to work in a place where the best ideas aren't the ones that win. I think the reason Google embraced "Don't be evil" so eagerly was not so much to impress the outside world as to inoculate themselves against arrogance. [\[1\]](#)

That has worked for Google so far. They've become more bureaucratic, but otherwise they seem to have held true to their original principles. With Apple that seems less the case. When you look at the famous [1984 ad](#) now, it's easier to imagine Apple as the dictator on the screen than the woman with the hammer. [\[2\]](#) In fact, if you read the dictator's speech it sounds uncannily like a prophecy of the App Store.

We have triumphed over the unprincipled dissemination of facts.

We have created, for the first time in all history, a garden of pure ideology, where each worker may bloom secure from the pests of contradictory and confusing truths.

The other reason Apple should care what programmers think of them is that when you sell a platform, developers make or break you. If anyone should know this, Apple should. VisiCalc made the Apple II.

And programmers build applications for the platforms they use. Most applications—most startups, probably—grow out of personal projects. Apple itself did. Apple made microcomputers because that's what Steve Wozniak wanted for himself. He couldn't have afforded a minicomputer. [3] Microsoft likewise started out making interpreters for little microcomputers because Bill Gates and Paul Allen were interested in using them. It's a rare startup that doesn't build something the founders use.

The main reason there are so many iPhone apps is that so many programmers have iPhones. They may know, because they read it in an article, that Blackberry has such and such market share. But in practice it's as if RIM didn't exist. If they're going to build something, they want to be able to use it themselves, and that means building an iPhone app.

So programmers continue to develop iPhone apps, even though Apple continues to maltreat them. They're like someone stuck in an abusive relationship. They're so attracted to the iPhone that they can't leave. But they're looking for a way out. One wrote:

While I did enjoy developing for the iPhone, the control they place on the App Store does not give me the drive to develop applications as I would like. In fact I don't intend to make any more iPhone applications unless absolutely necessary. [4]

Can anything break this cycle? No device I've seen so far could. Palm and RIM haven't a hope. The only credible contender is Android. But Android is an orphan; Google doesn't really care about it, not the way Apple cares about the iPhone. Apple cares about the iPhone the way Google cares about search.

\* \* \*

Is the future of handheld devices one locked down by Apple? It's a worrying prospect. It would be a bummer to have another grim monoculture like we had in the 1990s. In 1995, writing software for end users was effectively identical with writing Windows applications. Our horror at that prospect was the single biggest thing that drove us to start building [web apps](#).

At least we know now what it would take to break Apple's lock. You'd have to get iPhones out of programmers' hands. If programmers used some other device for mobile web access, they'd start to develop apps for that instead.

How could you make a device programmers liked better than the iPhone? It's unlikely you could make something better designed. Apple leaves no room there. So this alternative device probably couldn't win on general appeal. It would have to win by virtue of some appeal it had to programmers specifically.

One way to appeal to programmers is with software. If you could think of an

application programmers had to have, but that would be impossible in the circumscribed world of the iPhone, you could presumably get them to switch.

That would definitely happen if programmers started to use handhelds as development machines—if handhelds displaced laptops the way laptops displaced desktops. You need more control of a development machine than Apple will let you have over an iPhone.

Could anyone make a device that you'd carry around in your pocket like a phone, and yet would also work as a development machine? It's hard to imagine what it would look like. But I've learned never to say never about technology. A phone-sized device that would work as a development machine is no more miraculous by present standards than the iPhone itself would have seemed by the standards of 1995.

My current development machine is a MacBook Air, which I use with an external monitor and keyboard in my office, and by itself when traveling. If there was a version half the size I'd prefer it. That still wouldn't be small enough to carry around everywhere like a phone, but we're within a factor of 4 or so. Surely that gap is bridgeable. In fact, let's make it an [RFS](#). Wanted: Woman with hammer.

## Notes

[1] When Google adopted "Don't be evil," they were still so small that no one would have expected them to be, yet.

[2] The dictator in the 1984 ad isn't Microsoft, incidentally; it's IBM. IBM seemed a lot more frightening in those days, but they were friendlier to developers than Apple is now.

[3] He couldn't even afford a *monitor*. That's why the Apple I used a TV as a monitor.

[4] Several people I talked to mentioned how much they liked the iPhone SDK. The problem is not Apple's products but their policies. Fortunately policies are software; Apple can change them instantly if they want to. Handy that, isn't it?

**Thanks** to Sam Altman, Trevor Blackwell, Ross Boucher, James Bracy, Gabor Cselle, Patrick Collison, Jason Freedman, John Gruber, Joe Hewitt, Jessica Livingston, Robert Morris, Teng Siong Ong, Nikhil Pandit, Savraj Singh, and Jared Tame for reading drafts of this.



# Organic Startup Ideas

April 2010

The best way to come up with startup ideas is to ask yourself the question: what do you wish someone would make for you?

There are two types of startup ideas: those that grow organically out of your own life, and those that you decide, from afar, are going to be necessary to some class of users other than you. Apple was the first type. Apple happened because Steve Wozniak wanted a computer. Unlike most people who wanted computers, he could design one, so he did. And since lots of other people wanted the same thing, Apple was able to sell enough of them to get the company rolling. They still rely on this principle today, incidentally. The iPhone is the phone Steve Jobs wants. [\[1\]](#)

Our own startup, Viaweb, was of the second type. We made software for building online stores. We didn't need this software ourselves. We weren't direct marketers. We didn't even know when we started that our users were called "direct marketers." But we were comparatively old when we started the company (I was 30 and Robert Morris was 29), so we'd seen enough to know users would need this type of software. [\[2\]](#)

There is no sharp line between the two types of ideas, but the most successful startups seem to be closer to the Apple type than the Viaweb type. When he was writing that first Basic interpreter for the Altair, Bill Gates was writing something he would use, as were Larry and Sergey when they wrote the first versions of Google.

Organic ideas are generally preferable to the made up kind, but particularly so when the founders are young. It takes experience to predict what other people will want. The worst ideas we see at Y Combinator are from young founders making things they think other people will want.

So if you want to start a startup and don't know yet what you're going to do, I'd encourage you to focus initially on organic ideas. What's missing or broken in your daily life? Sometimes if you just ask that question you'll get immediate answers. It must have seemed obviously broken to Bill Gates that you could only program the Altair in machine language.

You may need to stand outside yourself a bit to see brokenness, because you tend

to get used to it and take it for granted. You can be sure it's there, though. There are always great ideas sitting right under our noses. In 2004 it was ridiculous that Harvard undergrads were still using a Facebook printed on paper. Surely that sort of thing should have been online.

There are ideas that obvious lying around now. The reason you're overlooking them is the same reason you'd have overlooked the idea of building Facebook in 2004: organic startup ideas usually don't seem like startup ideas at first. We know now that Facebook was very successful, but put yourself back in 2004. Putting undergraduates' profiles online wouldn't have seemed like much of a startup idea. And in fact, it wasn't initially a startup idea. When Mark spoke at a YC dinner this winter he said he wasn't trying to start a company when he wrote the first version of Facebook. It was just a project. So was the Apple I when Woz first started working on it. He didn't think he was starting a company. If these guys had thought they were starting companies, they might have been tempted to do something more "serious," and that would have been a mistake.

So if you want to come up with organic startup ideas, I'd encourage you to focus more on the idea part and less on the startup part. Just fix things that seem broken, regardless of whether it seems like the problem is important enough to build a company on. If you keep pursuing such threads it would be hard not to end up making something of value to a lot of people, and when you do, surprise, you've got a company. [3]

Don't be discouraged if what you produce initially is something other people dismiss as a toy. In fact, that's a good sign. That's probably why everyone else has been overlooking the idea. The first microcomputers were dismissed as toys. And the first planes, and the first cars. At this point, when someone comes to us with something that users like but that we could envision forum trolls dismissing as a toy, it makes us especially likely to invest.

While young founders are at a disadvantage when coming up with made-up ideas, they're the best source of organic ones, because they're at the forefront of technology. They use the latest stuff. They only just decided what to use, so why wouldn't they? And because they use the latest stuff, they're in a position to discover valuable types of fixable brokenness first.

There's nothing more valuable than an unmet need that is just becoming fixable. If you find something broken that you can fix for a lot of people, you've found a gold mine. As with an actual gold mine, you still have to work hard to get the gold out of it. But at least you know where the seam is, and that's the hard part.

## Notes

[1] This suggests a way to predict areas where Apple will be weak: things Steve Jobs doesn't use. E.g. I doubt he is much into gaming.

[2] In retrospect, we should have *become* direct marketers. If I were doing Viaweb again, I'd open our own online store. If we had, we'd have understood users a lot better. I'd encourage anyone starting a startup to become one of its users, however unnatural it seems.

[3] Possible exception: It's hard to compete directly with open source software. You can build things for programmers, but there has to be some part you can charge for.

**Thanks** to Sam Altman, Trevor Blackwell, and Jessica Livingston for reading drafts of this.



# How to Lose Time and Money

July 2010

When we sold our startup in 1998 I suddenly got a lot of money. I now had to think about something I hadn't had to think about before: how not to lose it. I knew it was possible to go from rich to poor, just as it was possible to go from poor to rich. But while I'd spent a lot of the past several years studying the paths from [poor to rich](#), I knew practically nothing about the paths from rich to poor. Now, in order to avoid them, I had to learn where they were.

So I started to pay attention to how fortunes are lost. If you'd asked me as a kid how rich people became poor, I'd have said by spending all their money. That's how it happens in books and movies, because that's the colorful way to do it. But in fact the way most fortunes are lost is not through excessive expenditure, but through bad investments.

It's hard to spend a fortune without noticing. Someone with ordinary tastes would find it hard to blow through more than a few tens of thousands of dollars without thinking "wow, I'm spending a lot of money." Whereas if you start trading derivatives, you can lose a million dollars (as much as you want, really) in the blink of an eye.

In most people's minds, spending money on luxuries sets off alarms that making investments doesn't. Luxuries seem self-indulgent. And unless you got the money by inheriting it or winning a lottery, you've already been thoroughly trained that self-indulgence leads to trouble. Investing bypasses those alarms. You're not spending the money; you're just moving it from one asset to another. Which is why people trying to sell you expensive things say "it's an investment."

The solution is to develop new alarms. This can be a tricky business, because while the alarms that prevent you from overspending are so basic that they may even be in our DNA, the ones that prevent you from making bad investments have to be learned, and are sometimes fairly counterintuitive.

A few days ago I realized something surprising: the situation with time is much the same as with money. The most dangerous way to lose time is not to spend it having fun, but to spend it doing fake work. When you spend time having fun, you know you're being self-indulgent. Alarms start to go off fairly quickly. If I woke up one morning and sat down on the sofa and watched TV all day, I'd feel like

something was terribly wrong. Just thinking about it makes me wince. I'd start to feel uncomfortable after sitting on a sofa watching TV for 2 hours, let alone a whole day.

And yet I've definitely had days when I might as well have sat in front of a TV all day — days at the end of which, if I asked myself what I got done that day, the answer would have been: basically, nothing. I feel bad after these days too, but nothing like as bad as I'd feel if I spent the whole day on the sofa watching TV. If I spent a whole day watching TV I'd feel like I was descending into perdition. But the same alarms don't go off on the days when I get nothing done, because I'm doing stuff that seems, superficially, like real work. Dealing with email, for example. You do it sitting at a desk. It's not fun. So it must be work.

With time, as with money, avoiding pleasure is no longer enough to protect you. It probably was enough to protect hunter-gatherers, and perhaps all pre-industrial societies. So nature and nurture combine to make us avoid self-indulgence. But the world has gotten more complicated: the most dangerous traps now are new behaviors that bypass our alarms about self-indulgence by mimicking more virtuous types. And the worst thing is, they're not even fun.

**Thanks** to Sam Altman, Trevor Blackwell, Patrick Collison, Jessica Livingston, and Robert Morris for reading drafts of this.

# The Top Idea in Your Mind

July 2010

I realized recently that what one thinks about in the shower in the morning is more important than I'd thought. I knew it was a good time to have ideas. Now I'd go further: now I'd say it's hard to do a really good job on anything you don't think about in the shower.

Everyone who's worked on difficult problems is probably familiar with the phenomenon of working hard to figure something out, failing, and then suddenly seeing the answer a bit later while doing something else. There's a kind of thinking you do without trying to. I'm increasingly convinced this type of thinking is not merely helpful in solving hard problems, but necessary. The tricky part is, you can only control it indirectly. [\[1\]](#)

I think most people have one top idea in their mind at any given time. That's the idea their thoughts will drift toward when they're allowed to drift freely. And this idea will thus tend to get all the benefit of that type of thinking, while others are starved of it. Which means it's a disaster to let the wrong idea become the top one in your mind.

What made this clear to me was having an idea I didn't want as the top one in my mind for two long stretches.

I'd noticed startups got way less done when they started raising money, but it was not till we ourselves raised money that I understood why. The problem is not the actual time it takes to meet with investors. The problem is that once you start raising money, raising money becomes the top idea in your mind. That becomes what you think about when you take a shower in the morning. And that means other questions aren't.

I'd hated raising money when I was running Viaweb, but I'd forgotten why I hated it so much. When we raised money for Y Combinator, I remembered. Money matters are particularly likely to become the top idea in your mind. The reason is that they have to be. It's hard to get money. It's not the sort of thing that happens by default. It's not going to happen unless you let it become the thing you think about in the shower. And then you'll make little progress on anything else you'd rather be working on. [\[2\]](#)

(I hear similar complaints from friends who are professors. Professors nowadays seem to have become professional fundraisers who do a little research on the side. It may be time to fix that.)

The reason this struck me so forcibly is that for most of the preceding 10 years I'd been able to think about what I wanted. So the contrast when I couldn't was sharp. But I don't think this problem is unique to me, because just about every startup I've seen grinds to a halt when they start raising money ♦ or [talking to acquirers](#).

You can't directly control where your thoughts drift. If you're controlling them, they're not drifting. But you can control them indirectly, by controlling what situations you let yourself get into. That has been the lesson for me: be careful what you let become critical to you. Try to get yourself into situations where the most urgent problems are ones you want to think about.

You don't have complete control, of course. An emergency could push other thoughts out of your head. But barring emergencies you have a good deal of indirect control over what becomes the top idea in your mind.

I've found there are two types of thoughts especially worth avoiding ♦ thoughts like the Nile Perch in the way they push out more interesting ideas. One I've already mentioned: thoughts about money. Getting money is almost by definition an attention sink. The other is disputes. These too are engaging in the wrong way: they have the same velcro-like shape as genuinely interesting ideas, but without the substance. So avoid disputes if you want to get real work done. [3]

Even Newton fell into this trap. After publishing his theory of colors in 1672 he found himself distracted by disputes for years, finally concluding that the only solution was to stop publishing:

I see I have made myself a slave to Philosophy, but if I get free of Mr Linus's business I will resolutely bid adew to it eternally, excepting what I do for my privat satisfaction or leave to come out after me. For I see a man must either resolve to put out nothing new or become a slave to defend it. [4]

Linus and his students at Liege were among the more tenacious critics. Newton's biographer Westfall seems to feel he was overreacting:

Recall that at the time he wrote, Newton's "slavery" consisted of five replies to Liege, totalling fourteen printed pages, over the course of a year.

I'm more sympathetic to Newton. The problem was not the 14 pages, but the pain of having this stupid controversy constantly reintroduced as the top idea in a mind that wanted so eagerly to think about other things.

Turning the other cheek turns out to have selfish advantages. Someone who does you an injury hurts you twice: first by the injury itself, and second by taking up your time afterward thinking about it. If you learn to ignore injuries you can at least avoid the second half. I've found I can to some extent avoid thinking about nasty things people have done to me by telling myself: this doesn't deserve space in my head. I'm always delighted to find I've forgotten the details of disputes, because that means I hadn't been thinking about them. My wife thinks I'm more forgiving than she is, but my motives are purely selfish.

I suspect a lot of people aren't sure what's the top idea in their mind at any given time. I'm often mistaken about it. I tend to think it's the idea I'd want to be the top one, rather than the one that is. But it's easy to figure this out: just take a shower. What topic do your thoughts keep returning to? If it's not what you want to be thinking about, you may want to change something.

## Notes

[1] No doubt there are already names for this type of thinking, but I call it "ambient thought."

[2] This was made particularly clear in our case, because neither of the funds we raised was difficult, and yet in both cases the process dragged on for months. Moving large amounts of money around is never something people treat casually. The attention required increases with the amount—maybe not linearly, but definitely monotonically.

[3] Corollary: Avoid becoming an administrator, or your job will consist of dealing with money and disputes.

[4] Letter to Oldenburg, quoted in Westfall, Richard, *Life of Isaac Newton*, p. 107.

**Thanks** to Sam Altman, Patrick Collison, Jessica Livingston, and Robert Morris for reading drafts of this.

# The Acceleration of Addictiveness

July 2010

What hard liquor, cigarettes, heroin, and crack have in common is that they're all more concentrated forms of less addictive predecessors. Most if not all the things we describe as addictive are. And the scary thing is, the process that created them is accelerating.

We wouldn't want to stop it. It's the same process that cures diseases: technological progress. Technological progress means making things do more of what we want. When the thing we want is something we want to want, we consider technological progress good. If some new technique makes solar cells x% more efficient, that seems strictly better. When progress concentrates something we don't want to want — when it transforms opium into heroin — it seems bad. But it's the same process at work. [\[1\]](#)

No one doubts this process is accelerating, which means increasing numbers of things we like will be transformed into things we like too much. [\[2\]](#)

As far as I know there's no word for something we like too much. The closest is the colloquial sense of "addictive." That usage has become increasingly common during my lifetime. And it's clear why: there are an increasing number of things we need it for. At the extreme end of the spectrum are crack and meth. Food has been transformed by a combination of factory farming and innovations in food processing into something with way more immediate bang for the buck, and you can see the results in any town in America. Checkers and solitaire have been replaced by World of Warcraft and FarmVille. TV has become much more engaging, and even so it [can't compete](#) with Facebook.

The world is more addictive than it was 40 years ago. And unless the forms of technological progress that produced these things are subject to different laws than technological progress in general, the world will get more addictive in the next 40 years than it did in the last 40.

The next 40 years will bring us some wonderful things. I don't mean to imply they're all to be avoided. Alcohol is a dangerous drug, but I'd rather live in a world with wine than one without. Most people can coexist with alcohol; but you have to be careful. More things we like will mean more things we have to be careful about.

Most people won't, unfortunately. Which means that as the world becomes more addictive, the two senses in which one can live a normal life will be driven ever further apart. One sense of "normal" is statistically normal: what everyone else does. The other is the sense we mean when we talk about the normal operating range of a piece of machinery: what works best.

These two senses are already quite far apart. Already someone trying to live well would seem eccentrically abstemious in most of the US. That phenomenon is only going to become more pronounced. You can probably take it as a rule of thumb from now on that if people don't think you're weird, you're living badly.

Societies eventually develop antibodies to addictive new things. I've seen that happen with cigarettes. When cigarettes first appeared, they spread the way an infectious disease spreads through a previously isolated population. Smoking rapidly became a (statistically) normal thing. There were ashtrays everywhere. We had ashtrays in our house when I was a kid, even though neither of my parents smoked. You had to for guests.

As knowledge spread about the dangers of smoking, customs changed. In the last 20 years, smoking has been transformed from something that seemed totally normal into a rather seedy habit: from something movie stars did in publicity shots to something small huddles of addicts do outside the doors of office buildings. A lot of the change was due to legislation, of course, but the legislation couldn't have happened if customs hadn't already changed.

It took a while though—on the order of 100 years. And unless the rate at which social antibodies evolve can increase to match the accelerating rate at which technological progress throws off new addictions, we'll be increasingly unable to rely on customs to protect us. [3] Unless we want to be canaries in the coal mine of each new addiction—the people whose sad example becomes a lesson to future generations—we'll have to figure out for ourselves what to avoid and how. It will actually become a reasonable strategy (or a more reasonable strategy) to suspect [everything new](#).

In fact, even that won't be enough. We'll have to worry not just about new things, but also about existing things becoming more addictive. That's what bit me. I've avoided most addictions, but the Internet got me because it became addictive while I was using it. [4]

Most people I know have problems with Internet addiction. We're all trying to figure out our own customs for getting free of it. That's why I don't have an iPhone, for example; the last thing I want is for the Internet to follow me out into the world. [5] My latest trick is taking long hikes. I used to think running was a better form of exercise than hiking because it took less time. Now the slowness of hiking seems an advantage, because the longer I spend on the trail, the longer I have to think without interruption.

Sounds pretty eccentric, doesn't it? It always will when you're trying to solve

problems where there are no customs yet to guide you. Maybe I can't plead Occam's razor; maybe I'm simply eccentric. But if I'm right about the acceleration of addictiveness, then this kind of lonely squirming to avoid it will increasingly be the fate of anyone who wants to get things done. We'll increasingly be defined by what we say no to.

## Notes

[1] Could you restrict technological progress to areas where you wanted it? Only in a limited way, without becoming a police state. And even then your restrictions would have undesirable side effects. "Good" and "bad" technological progress aren't sharply differentiated, so you'd find you couldn't slow the latter without also slowing the former. And in any case, as Prohibition and the "war on drugs" show, bans often do more harm than good.

[2] Technology has always been accelerating. By Paleolithic standards, technology evolved at a blistering pace in the Neolithic period.

[3] Unless we mass produce social customs. I suspect the recent resurgence of evangelical Christianity in the US is partly a reaction to drugs. In desperation people reach for the sledgehammer; if their kids won't listen to them, maybe they'll listen to God. But that solution has broader consequences than just getting kids to say no to drugs. You end up saying no to [science](#) as well.

I worry we may be heading for a future in which only a few people plot their own itinerary through no-land, while everyone else books a package tour. Or worse still, has one booked for them by the government.

[4] People commonly use the word "procrastination" to describe what they do on the Internet. It seems to me too mild to describe what's happening as merely not-doing-work. We don't call it procrastination when someone gets drunk instead of working.

[5] Several people have told me they like the iPad because it lets them bring the Internet into situations where a laptop would be too conspicuous. In other words, it's a hip flask. (This is true of the iPhone too, of course, but this advantage isn't as obvious because it reads as a phone, and everyone's used to those.)

**Thanks** to Sam Altman, Patrick Collison, Jessica Livingston, and Robert Morris for reading drafts of this.





# The Future of Startup Funding

August 2010

Two years ago I [wrote](#) about what I called "a huge, unexploited opportunity in startup funding:" the growing disconnect between VCs, whose current business model requires them to invest large amounts, and a large class of startups that need less than they used to. Increasingly, startups want a couple hundred thousand dollars, not a couple million. [\[1\]](#)

The opportunity is a lot less unexploited now. Investors have poured into this territory from both directions. VCs are much more likely to make angel-sized investments than they were a year ago. And meanwhile the past year has seen a dramatic increase in a new type of investor: the super-angel, who operates like an angel, but using other people's money, like a VC.

Though a lot of investors are entering this territory, there is still room for more. The distribution of investors should mirror the distribution of startups, which has the usual power law dropoff. So there should be a lot more people investing tens or hundreds of thousands than millions. [\[2\]](#)

In fact, it may be good for angels that there are more people doing angel-sized deals, because if angel rounds become more legitimate, then startups may start to opt for angel rounds even when they could, if they wanted, raise series A rounds from VCs. One reason startups prefer series A rounds is that they're more prestigious. But if angel investors become more active and better known, they'll increasingly be able to compete with VCs in brand.

Of course, prestige isn't the main reason to prefer a series A round. A startup will probably get more attention from investors in a series A round than an angel round. So if a startup is choosing between an angel round and an A round from a good VC fund, I usually advise them to take the A round. [\[3\]](#)

But while series A rounds aren't going away, I think VCs should be more worried about super-angels than vice versa. Despite their name, the super-angels are really mini VC funds, and they clearly have existing VCs in their sights.

They would seem to have history on their side. The pattern here seems the same one we see when startups and established companies enter a new market. Online video becomes possible, and YouTube plunges right in, while existing media companies embrace it only half-willingly, driven more by fear than hope, and aiming more to protect their turf than to do great things for users. Ditto for PayPal. This pattern is repeated over and over, and it's usually the invaders who win. In this case the super-angels are the invaders. Angel rounds are their whole business,

as online video was for YouTube. Whereas VCs who make angel investments mostly do it as a way to generate deal flow for series A rounds. [4]

On the other hand, startup investing is a very strange business. Nearly all the returns are concentrated in a few big winners. If the super-angels merely fail to invest in (and to some extent produce) the big winners, they'll be out of business, even if they invest in all the others.

## **VCs**

Why don't VCs start doing smaller series A rounds? The sticking point is board seats. In a traditional series A round, the partner whose deal it is takes a seat on the startup's board. If we assume the average startup runs for 6 years and a partner can bear to be on 12 boards at once, then a VC fund can do 2 series A deals per partner per year.

It has always seemed to me the solution is to take fewer board seats. You don't have to be on the board to help a startup. Maybe VCs feel they need the power that comes with board membership to ensure their money isn't wasted. But have they tested that theory? Unless they've tried not taking board seats and found their returns are lower, they're not bracketing the problem.

I'm not saying VCs don't help startups. The good ones help them a lot. What I'm saying is that the kind of help that matters, you may not have to be a board member to give. [5]

How will this all play out? Some VCs will probably adapt, by doing more, smaller deals. I wouldn't be surprised if by streamlining their selection process and taking fewer board seats, VC funds could do 2 to 3 times as many series A rounds with no loss of quality.

But other VCs will make no more than superficial changes. VCs are conservative, and the threat to them isn't mortal. The VC funds that don't adapt won't be violently displaced. They'll edge gradually into a different business without realizing it. They'll still do what they will call series A rounds, but these will increasingly be de facto series B rounds. [6]

In such rounds they won't get the 25 to 40% of the company they do now. You don't give up as much of the company in later rounds unless something is seriously wrong. Since the VCs who don't adapt will be investing later, their returns from winners may be smaller. But investing later should also mean they have fewer losers. So their ratio of risk to return may be the same or even better. They'll just have become a different, more conservative, type of investment.

## **Angels**

In the big angel rounds that increasingly compete with series A rounds, the investors won't take as much equity as VCs do now. And VCs who try to compete with angels by doing more, smaller deals will probably find they have to take less equity to do it. Which is good news for founders: they'll get to keep more of the company.

The deal terms of angel rounds will become less restrictive too—not just less restrictive than series A terms, but less restrictive than angel terms have

traditionally been.

In the future, angel rounds will less often be for specific amounts or have a lead investor. In the old days, the standard m.o. for startups was to find one angel to act as the lead investor. You'd negotiate a round size and valuation with the lead, who'd supply some but not all of the money. Then the startup and the lead would cooperate to find the rest.

The future of angel rounds looks more like this: instead of a fixed round size, startups will do a rolling close, where they take money from investors one at a time till they feel they have enough. [7] And though there's going to be one investor who gives them the first check, and his or her help in recruiting other investors will certainly be welcome, this initial investor will no longer be the lead in the old sense of managing the round. The startup will now do that themselves.

There will continue to be lead investors in the sense of investors who take the lead in *advising* a startup. They may also make the biggest investment. But they won't always have to be the one terms are negotiated with, or be the first money in, as they have in the past. Standardized paperwork will do away with the need to negotiate anything except the valuation, and that will get easier too.

If multiple investors have to share a valuation, it will be whatever the startup can get from the first one to write a check, limited by their guess at whether this will make later investors balk. But there may not have to be just one valuation. Startups are increasingly raising money on convertible notes, and convertible notes have not valuations but at most valuation *caps*: caps on what the effective valuation will be when the debt converts to equity (in a later round, or upon acquisition if that happens first). That's an important difference because it means a startup could do multiple notes at once with different caps. This is now starting to happen, and I predict it will become more common.

## Sheep

The reason things are moving this way is that the old way sucked for startups. Leads could (and did) use a fixed size round as a legitimate-seeming way of saying what all founders hate to hear: I'll invest if other people will. Most investors, unable to judge startups for themselves, rely instead on the opinions of other investors. If everyone wants in, they want in too; if not, not. Founders hate this because it's a recipe for deadlock, and delay is the thing a startup can least afford. Most investors know this m.o. is lame, and few say openly that they're doing it. But the craftier ones achieve the same result by offering to lead rounds of fixed size and supplying only part of the money. If the startup can't raise the rest, the lead is out too. How could they go ahead with the deal? The startup would be underfunded!

In the future, investors will increasingly be unable to offer investment subject to contingencies like other people investing. Or rather, investors who do that will get last place in line. Startups will go to them only to fill up rounds that are mostly subscribed. And since hot startups tend to have rounds that are oversubscribed, being last in line means they'll probably miss the hot deals. Hot deals and successful startups are not identical, but there is a significant correlation. [8] So investors who won't invest unilaterally will have lower returns.

Investors will probably find they do better when deprived of this crutch anyway.

Chasing hot deals doesn't make investors choose better; it just makes them feel better about their choices. I've seen feeding frenzies both form and fall apart many times, and as far as I can tell they're mostly random. [\[9\]](#) If investors can no longer rely on their herd instincts, they'll have to think more about each startup before investing. They may be surprised how well this works.

Deadlock wasn't the only disadvantage of letting a lead investor manage an angel round. The investors would not infrequently collude to push down the valuation. And rounds took too long to close, because however motivated the lead was to get the round closed, he was not a tenth as motivated as the startup.

Increasingly, startups are taking charge of their own angel rounds. Only a few do so far, but I think we can already declare the old way dead, because those few are the best startups. They're the ones in a position to tell investors how the round is going to work. And if the startups you want to invest in do things a certain way, what difference does it make what the others do?

## **Traction**

In fact, it may be slightly misleading to say that angel rounds will increasingly take the place of series A rounds. What's really happening is that startup-controlled rounds are taking the place of investor-controlled rounds.

This is an instance of a very important meta-trend, one that Y Combinator itself has been based on from the beginning: founders are becoming increasingly powerful relative to investors. So if you want to predict what the future of venture funding will be like, just ask: how would founders like it to be? One by one, all the things founders dislike about raising money are going to get eliminated. [\[10\]](#)

Using that heuristic, I'll predict a couple more things. One is that investors will increasingly be unable to wait for startups to have "traction" before they put in significant money. It's hard to predict in advance which startups will succeed. So most investors prefer, if they can, to wait till the startup is already succeeding, then jump in quickly with an offer. Startups hate this as well, partly because it tends to create deadlock, and partly because it seems kind of slimy. If you're a promising startup but don't yet have significant growth, all the investors are your friends in words, but few are in actions. They all say they love you, but they all wait to invest. Then when you start to see growth, they claim they were your friend all along, and are aghast at the thought you'd be so disloyal as to leave them out of your round. If founders become more powerful, they'll be able to make investors give them more money upfront.

(The worst variant of this behavior is the tranching deal, where the investor makes a small initial investment, with more to follow if the startup does well. In effect, this structure gives the investor a free option on the next round, which they'll only take if it's worse for the startup than they could get in the open market. Tranching deals are an abuse. They're increasingly rare, and they're going to get rarer.) [\[11\]](#)

Investors don't like trying to predict which startups will succeed, but increasingly they'll have to. Though the way that happens won't necessarily be that the behavior of existing investors will change; it may instead be that they'll be replaced by other investors with different behavior—that investors who understand startups well enough to take on the hard problem of predicting their trajectory will tend to displace suits whose skills lie more in raising money from LPs.

## Speed

The other thing founders hate most about fundraising is how long it takes. So as founders become more powerful, rounds should start to close faster.

Fundraising is still terribly distracting for startups. If you're a founder in the middle of raising a round, the round is the [top idea in your mind](#), which means working on the company isn't. If a round takes 2 months to close, which is reasonably fast by present standards, that means 2 months during which the company is basically treading water. That's the worst thing a startup could do.

So if investors want to get the best deals, the way to do it will be to close faster. Investors don't need weeks to make up their minds anyway. We decide based on about 10 minutes of reading an application plus 10 minutes of in person interview, and we only regret about 10% of our decisions. If we can decide in 20 minutes, surely the next round of investors can decide in a couple days. [\[12\]](#)

There are a lot of institutionalized delays in startup funding: the multi-week mating dance with investors; the distinction between termsheets and deals; the fact that each series A has enormously elaborate, custom paperwork. Both founders and investors tend to take these for granted. It's the way things have always been. But ultimately the reason these delays exist is that they're to the advantage of investors. More time gives investors more information about a startup's trajectory, and it also tends to make startups more pliable in negotiations, since they're usually short of money.

These conventions weren't designed to drag out the funding process, but that's why they're allowed to persist. Slowness is to the advantage of investors, who have in the past been the ones with the most power. But there is no need for rounds to take months or even weeks to close, and once founders realize that, it's going to stop. Not just in angel rounds, but in series A rounds too. The future is simple deals with standard terms, done quickly.

One minor abuse that will get corrected in the process is option pools. In a traditional series A round, before the VCs invest they make the company set aside a block of stock for future hires—usually between 10 and 30% of the company. The point is to ensure this dilution is borne by the existing shareholders. The practice isn't dishonest; founders know what's going on. But it makes deals unnecessarily complicated. In effect the valuation is 2 numbers. There's no need to keep doing this. [\[13\]](#)

The final thing founders want is to be able to sell some of their own stock in later rounds. This won't be a change, because the practice is now quite common. A lot of investors hated the idea, but the world hasn't exploded as a result, so it will happen more, and more openly.

## Surprise

I've talked here about a bunch of changes that will be forced on investors as founders become more powerful. Now the good news: investors may actually make more money as a result.

A couple days ago an interviewer [asked me](#) if founders having more power would

be better or worse for the world. I was surprised, because I'd never considered that question. Better or worse, it's happening. But after a second's reflection, the answer seemed obvious. Founders understand their companies better than investors, and it has to be better if the people with more knowledge have more power.

One of the mistakes novice pilots make is overcontrolling the aircraft: applying corrections too vigorously, so the aircraft oscillates about the desired configuration instead of approaching it asymptotically. It seems probable that investors have till now on average been overcontrolling their portfolio companies. In a lot of startups, the biggest source of stress for the founders is not competitors but investors. Certainly it was for us at Viaweb. And this is not a new phenomenon: investors were James Watt's biggest problem too. If having less power prevents investors from overcontrolling startups, it should be better not just for founders but for investors too.

Investors may end up with less stock per startup, but startups will probably do better with founders more in control, and there will almost certainly be more of them. Investors all compete with one another for deals, but they aren't one another's main competitor. Our main competitor is employers. And so far that competitor is crushing us. Only a tiny fraction of people who could start a startup do. Nearly all customers choose the competing product, a job. Why? Well, let's look at the product we're offering. An unbiased review would go something like this:

Starting a startup gives you more freedom and the opportunity to make a lot more money than a job, but it's also hard work and at times very stressful.

Much of the stress comes from dealing with investors. If reforming the investment process removed that stress, we'd make our product much more attractive. The kind of people who make good startup founders don't mind dealing with technical problems—they enjoy technical problems—but they hate the type of problems investors cause.

Investors have no idea that when they maltreat one startup, they're preventing 10 others from happening, but they are. Indirectly, but they are. So when investors stop trying to squeeze a little more out of their existing deals, they'll find they're net ahead, because so many more new deals appear.

One of our axioms at Y Combinator is not to think of deal flow as a zero-sum game. Our main focus is to encourage more startups to happen, not to win a larger share of the existing stream. We've found this principle very useful, and we think as it spreads outward it will help later stage investors as well.

"Make something people want" applies to us too.

## Notes

[1] In this essay I'm talking mainly about software startups. These points don't apply to types of startups that are still expensive to start, e.g. in energy or biotech.

Even the cheap kinds of startups will generally raise large amounts at some point, when they want to hire a lot of people. What has changed is how much they can get done before that.

[2] It's not the distribution of good startups that has a power law dropoff, but the distribution of potentially good startups, which is to say, good deals. There are lots of potential winners, from which a few actual winners emerge with superlinear certainty.

[3] As I was writing this, I asked some founders who'd taken series A rounds from top VC funds whether it was worth it, and they unanimously said yes.

The quality of investor is more important than the type of round, though. I'd take an angel round from good angels over a series A from a mediocre VC.

[4] Founders also worry that taking an angel investment from a VC means they'll look bad if the VC declines to participate in the next round. The trend of VC angel investing is so new that it's hard to say how justified this worry is.

Another danger, pointed out by Mitch Kapor, is that if VCs are only doing angel deals to generate series A deal flow, then their incentives aren't aligned with the founders'. The founders want the valuation of the next round to be high, and the VCs want it to be low. Again, hard to say yet how much of a problem this will be.

[5] Josh Kopelman pointed out that another way to be on fewer boards at once is to take board seats for shorter periods.

[6] Google was in this respect as so many others the pattern for the future. It would be great for VCs if the similarity extended to returns. That's probably too much to hope for, but the returns may be somewhat higher, as I explain later.

[7] Doing a rolling close doesn't mean the company is always raising money. That would be a distraction. The point of a rolling close is to make fundraising take less time, not more. With a classic fixed sized round, you don't get any money till all the investors agree, and that often creates a situation where they all sit waiting for the others to act. A rolling close usually prevents this.

[8] There are two (non-exclusive) causes of hot deals: the quality of the company, and domino effects among investors. The former is obviously a better predictor of success.

[9] Some of the randomness is concealed by the fact that investment is a self fulfilling prophecy.



[10] The shift in power to founders is exaggerated now because it's a seller's market. On the next downtick it will seem like I overstated the case. But on the next uptick after that, founders will seem more powerful than ever.

[11] More generally, it will become less common for the same investor to invest in successive rounds, except when exercising an option to maintain their percentage. When the same investor invests in successive rounds, it often means the startup isn't getting market price. They may not care; they may prefer to work with an investor they already know; but as the investment market becomes more efficient, it will become increasingly easy to get market price if they want it. Which in turn means the investment community will tend to become more stratified.

[12] The two 10 minuteses have 3 weeks between them so founders can get cheap plane tickets, but except for that they could be adjacent.

[13] I'm not saying option pools themselves will go away. They're an administrative convenience. What will go away is investors requiring them.

**Thanks** to Sam Altman, John Bautista, Trevor Blackwell, Paul Buchheit, Jeff Clavier, Patrick Collison, Ron Conway, Matt Cohler, Chris Dixon, Mitch Kapor, Josh Kopelman, Pete Koomen,Carolynn Levy, Jessica Livingston, Ariel Poler, Geoff Ralston, Naval Ravikant, Dan Siroker, Harj Taggar, and Fred Wilson for reading drafts of this.

# What Happened to Yahoo

August 2010

When I went to work for Yahoo after they bought our startup in 1998, it felt like the center of the world. It was supposed to be the next big thing. It was supposed to be what Google turned out to be.

What went wrong? The problems that hosed Yahoo go back a long time, practically to the beginning of the company. They were already very visible when I got there in 1998. Yahoo had two problems Google didn't: easy money, and ambivalence about being a technology company.

## Money

The first time I met Jerry Yang, we thought we were meeting for different reasons. He thought we were meeting so he could check us out in person before buying us. I thought we were meeting so we could show him our new technology, Revenue Loop. It was a way of sorting shopping search results. Merchants bid a percentage of sales for traffic, but the results were sorted not by the bid but by the bid times the average amount a user would buy. It was like the algorithm Google uses now to sort ads, but this was in the spring of 1998, before Google was founded.

Revenue Loop was the optimal sort for shopping search, in the sense that it sorted in order of how much money Yahoo would make from each link. But it wasn't just optimal in that sense. Ranking search results by user behavior also makes search better. Users train the search: you can start out finding matches based on mere textual similarity, and as users buy more stuff the search results get better and better.

Jerry didn't seem to care. I was confused. I was showing him technology that extracted the maximum value from search traffic, and he didn't care? I couldn't tell whether I was explaining it badly, or he was just very poker faced.

I didn't realize the answer till later, after I went to work at Yahoo. It was neither of my guesses. The reason Yahoo didn't care about a technique that extracted the full value of traffic was that advertisers were already overpaying for it. If Yahoo merely extracted the actual value, they'd have made less.

Hard as it is to believe now, the big money then was in banner ads. Advertisers were willing to pay ridiculous amounts for banner ads. So Yahoo's sales force had evolved to exploit this source of revenue. Led by a large and terrifyingly formidable man called Anil Singh, Yahoo's sales guys would fly out to Procter & Gamble and come back with million dollar orders for banner ad impressions.

The prices seemed cheap compared to print, which was what advertisers, for lack of any other reference, compared them to. But they were expensive compared to what they were worth. So these big, dumb companies were a dangerous source of revenue to depend on. But there was another source even more dangerous: other Internet startups.

By 1998, Yahoo was the beneficiary of a de facto Ponzi scheme. Investors were excited about the Internet. One reason they were excited was Yahoo's revenue growth. So they invested in new Internet startups. The startups then used the money to buy ads on Yahoo to get traffic. Which caused yet more revenue growth for Yahoo, and further convinced investors the Internet was worth investing in. When I realized this one day, sitting in my cubicle, I jumped up like Archimedes in his bathtub, except instead of "Eureka!" I was shouting "Sell!"

Both the Internet startups and the Procter & Gambles were doing brand advertising. They didn't care about targeting. They just wanted lots of people to see their ads. So traffic became the thing to get at Yahoo. It didn't matter what type. [\[1\]](#)

It wasn't just Yahoo. All the search engines were doing it. This was why they were trying to get people to start calling them "portals" instead of "search engines." Despite the actual meaning of the word portal, what they meant by it was a site where users would find what they wanted on the site itself, instead of just passing through on their way to other destinations, as they did at a search engine.

I remember telling David Filo in late 1998 or early 1999 that Yahoo should buy Google, because I and most of the other programmers in the company were using it instead of Yahoo for search. He told me that it wasn't worth worrying about. Search was only 6% of our traffic, and we were growing at 10% a month. It wasn't worth doing better.

I didn't say "But search traffic is worth more than other traffic!" I said "Oh, ok." Because I didn't realize either how much search traffic was worth. I'm not sure even Larry and Sergey did then. If they had, Google presumably wouldn't have expended any effort on enterprise search.

If circumstances had been different, the people running Yahoo might have realized sooner how important search was. But they had the most opaque obstacle in the world between them and the truth: money. As long as customers were writing big checks for banner ads, it was hard to take search seriously. Google didn't have that to distract them.

## Hackers

But Yahoo also had another problem that made it hard to change directions. They'd been thrown off balance from the start by their ambivalence about being a technology company.

One of the weirdest things about Yahoo when I went to work there was the way they insisted on calling themselves a "media company." If you walked around their offices, it seemed like a software company. The cubicles were full of programmers writing code, product managers thinking about feature lists and ship dates, support people (yes, there were actually support people) telling users to restart their browsers, and so on, just like a software company. So why did they call themselves a media company?

One reason was the way they made money: by selling ads. In 1995 it was hard to imagine a technology company making money that way. Technology companies made money by selling their software to users. Media companies sold ads. So they must be a media company.

Another big factor was the fear of Microsoft. If anyone at Yahoo considered the idea that they should be a technology company, the next thought would have been that Microsoft would crush them.

It's hard for anyone much younger than me to understand the fear Microsoft still inspired in 1995. Imagine a company with several times the power Google has now, but way meaner. It was perfectly reasonable to be afraid of them. Yahoo watched them crush the first hot Internet company, Netscape. It was reasonable to worry that if they tried to be the next Netscape, they'd suffer the same fate. How were they to know that Netscape would turn out to be Microsoft's last victim?

It would have been a clever move to pretend to be a media company to throw Microsoft off their scent. But unfortunately Yahoo actually tried to be one, sort of. Project managers at Yahoo were called "producers," for example, and the different parts of the company were called "properties." But what Yahoo really needed to be was a technology company, and by trying to be something else, they ended up being something that was neither here nor there. That's why Yahoo as a company has never had a sharply defined identity.

The worst consequence of trying to be a media company was that they didn't take programming seriously enough. Microsoft (back in the day), Google, and Facebook have all had hacker-centric cultures. But Yahoo treated programming as a commodity. At Yahoo, user-facing software was controlled by product managers and designers. The job of programmers was just to take the work of the product managers and designers the final step, by translating it into code.

One obvious result of this practice was that when Yahoo built things, they often weren't very good. But that wasn't the worst problem. The worst problem was that they hired bad programmers.

Microsoft (back in the day), Google, and Facebook have all been obsessed with hiring the best programmers. Yahoo wasn't. They preferred good programmers to bad ones, but they didn't have the kind of single-minded, almost obnoxiously elitist focus on hiring the smartest people that the big winners have had. And when you consider how much competition there was for programmers when they were hiring, during the Bubble, it's not surprising that the quality of their programmers was uneven.

In technology, once you have bad programmers, you're doomed. I can't think of an instance where a company has sunk into technical mediocrity and recovered. Good programmers want to work with other good programmers. So once the quality of programmers at your company starts to drop, you enter a death spiral from which there is no recovery. [2]

At Yahoo this death spiral started early. If there was ever a time when Yahoo was a Google-style talent magnet, it was over by the time I got there in 1998.

The company felt prematurely old. Most technology companies eventually get taken over by suits and middle managers. At Yahoo it felt as if they'd deliberately accelerated this process. They didn't want to be a bunch of hackers. They wanted to be suits. A media company should be run by suits.

The first time I visited Google, they had about 500 people, the same number Yahoo had when I went to work there. But boy did things seem different. It was still very much a hacker-centric culture. I remember talking to some programmers in the cafeteria about the problem of gaming search results (now known as SEO), and they asked "what should we do?" Programmers at Yahoo wouldn't have asked that. Theirs was not to reason why; theirs was to build what product managers spec'd. I remember coming away from Google thinking "Wow, it's still a startup."

There's not much we can learn from Yahoo's first fatal flaw. It's probably too much to hope any company could avoid being damaged by depending on a bogus source of revenue. But startups can learn an important lesson from the second one. In the software business, you can't afford not to have a hacker-centric culture.

Probably the most impressive commitment I've heard to having a hacker-centric culture came from Mark Zuckerberg, when he spoke at Startup School in 2007. He said that in the early days Facebook made a point of hiring programmers even for jobs that would not ordinarily consist of programming, like HR and marketing.

So which companies need to have a hacker-centric culture? Which companies are "in the software business" in this respect? As Yahoo discovered, the area covered by this rule is bigger than most people realize. The answer is: any company that needs to have good software.

Why would great programmers want to work for a company that didn't have a hacker-centric culture, as long as there were others that did? I can imagine two

reasons: if they were paid a huge amount, or if the domain was interesting and none of the companies in it were hacker-centric. Otherwise you can't attract good programmers to work in a suit-centric culture. And without good programmers you won't get good software, no matter how many people you put on a task, or how many procedures you establish to ensure "quality."

[Hacker culture](#) often seems kind of irresponsible. That's why people proposing to destroy it use phrases like "adult supervision." That was the phrase they used at Yahoo. But there are worse things than seeming irresponsible. Losing, for example.

## Notes

[1] The closest we got to targeting when I was there was when we created [pets.yahoo.com](#) in order to provoke a bidding war between 3 pet supply startups for the spot as top sponsor.

[2] In theory you could beat the death spiral by buying good programmers instead of hiring them. You can get programmers who would never have come to you as employees by buying their startups. But so far the only companies smart enough to do this are companies smart enough not to need to.

**Thanks** to Trevor Blackwell, Jessica Livingston, and Geoff Ralston for reading drafts of this.

# High Resolution Fundraising

September 2010

The reason startups have been using [more convertible notes](#) in angel rounds is that they make deals close faster. By making it easier for startups to give different prices to different investors, they help them break the sort of deadlock that happens when investors all wait to see who else is going to invest.

By far the biggest influence on investors' opinions of a startup is the opinion of other investors. There are very, very few who simply decide for themselves. Any startup founder can tell you the most common question they hear from investors is not about the founders or the product, but "who else is investing?"

That tends to produce deadlocks. Raising an old-fashioned fixed-size equity round can take weeks, because all the angels sit around waiting for the others to commit, like competitors in a bicycle sprint who deliberately ride slowly at the start so they can follow whoever breaks first.

Convertible notes let startups beat such deadlocks by rewarding investors willing to move first with lower (effective) valuations. Which they deserve because they're taking more risk. It's much safer to invest in a startup Ron Conway has already invested in; someone who comes after him should pay a higher price.

The reason convertible notes allow more flexibility in price is that valuation caps aren't actual valuations, and notes are cheap and easy to do. So you can do high-resolution fundraising: if you wanted you could have a separate note with a different cap for each investor.

That cap need not simply rise monotonically. A startup could also give better deals to investors they expected to help them most. The point is simply that different investors, whether because of the help they offer or their willingness to commit, have different values for startups, and their terms should reflect that.

Different terms for different investors is clearly the way of the future. Markets always evolve toward higher resolution. You may not need to use convertible notes to do it. With sufficiently lightweight standardized equity terms (and some changes in investors' and lawyers' expectations about equity rounds) you might be able to do the same thing with equity instead of debt. Either would be fine with startups,

so long as they can easily change their valuation.

Deadlocks weren't the only problem with fixed-size equity rounds. Another was that startups had to decide in advance how much to raise. I think it's a mistake for a startup to fix upon a specific number. If investors are easily convinced, the startup should raise more now, and if investors are skeptical, the startup should take a smaller amount and use that to get the company to the point where it's more convincing.

It's just not reasonable to expect startups to pick an optimal round size in advance, because that depends on the reactions of investors, and those are impossible to predict.

Fixed-size, multi-investor angel rounds are such a bad idea for startups that one wonders why things were ever done that way. One possibility is that this custom reflects the way investors like to collude when they can get away with it. But I think the actual explanation is less sinister. I think angels (and their lawyers) organized rounds this way in unthinking imitation of VC series A rounds. In a series A, a fixed-size equity round with a lead makes sense, because there is usually just one big investor, who is unequivocally the lead. Fixed-size series A rounds already are high res. But the more investors you have in a round, the less sense it makes for everyone to get the same price.

The most interesting question here may be what high res fundraising will do to the world of investors. Bolder investors will now get rewarded with lower prices. But more important, in a hits-driven business, is that they'll be able to get into the deals they want. Whereas the "who else is investing?" type of investors will not only pay higher prices, but may not be able to get into the best deals at all.

**Thanks** to Immad Akhund, Sam Altman, John Bautista, Pete Koomen, Jessica Livingston, Dan Siroker, Harj Taggar, and Fred Wilson for reading drafts of this.



# Where to See Silicon Valley

October 2010

Silicon Valley proper is mostly suburban sprawl. At first glance it doesn't seem there's anything to see. It's not the sort of place that has conspicuous monuments. But if you look, there are subtle signs you're in a place that's different from other places.

## 1. [Stanford University](#)

Stanford is a strange place. Structurally it is to an ordinary university what suburbia is to a city. It's enormously spread out, and feels surprisingly empty much of the time. But notice the weather. It's probably perfect. And notice the beautiful mountains to the west. And though you can't see it, cosmopolitan San Francisco is 40 minutes to the north. That combination is much of the reason Silicon Valley grew up around this university and not some other one.

## 2. [University Ave](#)

A surprising amount of the work of the Valley is done in the cafes on or just off University Ave in Palo Alto. If you visit on a weekday between 10 and 5, you'll often see founders pitching investors. In case you can't tell, the founders are the ones leaning forward eagerly, and the investors are the ones sitting back with slightly pained expressions.

## 3. [The Lucky Office](#)

The office at 165 University Ave was Google's first. Then it was Paypal's. (Now it's [Wepay](#)'s.) The interesting thing about it is the location. It's a smart move to put a startup in a place with restaurants and people walking around instead of in an office park, because then the people who work there want to stay there, instead of fleeing as soon as conventional working hours end. They go out for dinner together, talk about ideas, and then come back and implement them.

It's important to realize that Google's current location in an office park is not where they started; it's just where they were forced to move when they needed more space. Facebook was till recently across the street, till they too had to move because they needed more space.

#### 4. [Old Palo Alto](#)

Palo Alto was not originally a suburb. For the first 100 years or so of its existence, it was a college town out in the countryside. Then in the mid 1950s it was engulfed in a wave of suburbia that raced down the peninsula. But Palo Alto north of Oregon expressway still feels noticeably different from the area around it. It's one of the nicest places in the Valley. The buildings are old (though increasingly they are being torn down and replaced with generic McMansions) and the trees are tall. But houses are very expensive—around \$1000 per square foot. This is post-exit Silicon Valley.

#### 5. [Sand Hill Road](#)

It's interesting to see the VCs' offices on the north side of Sand Hill Road precisely because they're so boringly uniform. The buildings are all more or less the same, their exteriors express very little, and they are arranged in a confusing maze. (I've been visiting them for years and I still occasionally get lost.) It's not a coincidence. These buildings are a pretty accurate reflection of the VC business.

If you go on a weekday you may see groups of founders there to meet VCs. But mostly you won't see anyone; bustling is the last word you'd use to describe the atmos. Visiting Sand Hill Road reminds you that the opposite of "down and dirty" would be "up and clean."

#### 6. [Castro Street](#)

It's a tossup whether Castro Street or University Ave should be considered the heart of the Valley now. University Ave would have been 10 years ago. But Palo Alto is getting expensive. Increasingly startups are located in Mountain View, and Palo Alto is a place they come to meet investors. Palo Alto has a lot of different cafes, but there is one that clearly dominates in Mountain View: [Red Rock](#).

#### 7. [Google](#)

Google spread out from its first building in Mountain View to a lot of the surrounding ones. But because the buildings were built at different times by different people, the place doesn't have the sterile, walled-off feel that a typical large company's headquarters have. It definitely has a flavor of its own though. You sense there is something afoot. The general atmos is vaguely utopian; there are lots of Priuses, and people who look like they drive them.

You can't get into Google unless you know someone there. It's very much worth seeing inside if you can, though. Ditto for Facebook, at the end of California Ave in Palo Alto, though there is nothing to see outside.

#### 8. [Skyline Drive](#)

Skyline Drive runs along the crest of the Santa Cruz mountains. On one side is the Valley, and on the other is the sea—which because it's cold and foggy and has few harbors, plays surprisingly little role in the lives of people in the Valley, considering how close it is. Along some parts of Skyline the dominant trees are huge redwoods, and in others they're live oaks. Redwoods mean those are the parts where the fog off the coast comes in at night; redwoods condense rain out of fog. The MROSD manages a collection of [great walking trails](#) off Skyline.

## 9. [280](#)

Silicon Valley has two highways running the length of it: 101, which is pretty ugly, and 280, which is one of the more beautiful highways in the world. I always take 280 when I have a choice. Notice the long narrow lake to the west? That's the San Andreas Fault. It runs along the base of the hills, then heads uphill through Portola Valley. One of the MROSD trails runs [right along the fault](#). A string of rich neighborhoods runs along the foothills to the west of 280: Woodside, Portola Valley, Los Altos Hills, Saratoga, Los Gatos.

[SLAC](#) goes right under 280 a little bit south of Sand Hill Road. And a couple miles south of that is the Valley's equivalent of the "Welcome to Las Vegas" sign: [The Dish](#).

## Notes

I skipped the [Computer History Museum](#) because this is a list of where to see the Valley itself, not where to see artifacts from it. I also skipped San Jose. San Jose calls itself the capital of Silicon Valley, but when people in the Valley use the phrase "the city," they mean San Francisco. San Jose is a dotted line on a map.

**Thanks** to Sam Altman, Paul Buchheit, Patrick Collison, and Jessica Livingston for reading drafts of this.

# The New Funding Landscape

October 2010

After barely changing at all for decades, the startup funding business is now in what could, at least by comparison, be called turmoil. At Y Combinator we've seen dramatic changes in the funding environment for startups. Fortunately one of them is much higher valuations.

The trends we've been seeing are probably not YC-specific. I wish I could say they were, but the main cause is probably just that we see trends first—partly because the startups we fund are very plugged into the Valley and are quick to take advantage of anything new, and partly because we fund so many that we have enough data points to see patterns clearly.

What we're seeing now, everyone's probably going to be seeing in the next couple years. So I'm going to explain what we're seeing, and what that will mean for you if you try to raise money.

## Super-Angels

Let me start by describing what the world of startup funding used to look like. There used to be two sharply differentiated types of investors: angels and venture capitalists. Angels are individual rich people who invest small amounts of their own money, while VCs are employees of funds that invest large amounts of other people's.

For decades there were just those two types of investors, but now a third type has appeared halfway between them: the so-called super-angels. [\[1\]](#) And VCs have been provoked by their arrival into making a lot of angel-style investments themselves. So the previously sharp line between angels and VCs has become hopelessly blurred.

There used to be a no man's land between angels and VCs. Angels would invest \$20k to \$50k apiece, and VCs usually a million or more. So an angel round meant a collection of angel investments that combined to maybe \$200k, and a VC round meant a series A round in which a single VC fund (or occasionally two) invested \$1-5 million.

The no man's land between angels and VCs was a very inconvenient one for startups, because it coincided with the amount many wanted to raise. Most startups coming out of Demo Day wanted to raise around \$400k. But it was a pain to stitch together that much out of angel investments, and most VCs weren't interested in investments so small. That's the fundamental reason the super-angels have appeared. They're responding to the market.

The arrival of a new type of investor is big news for startups, because there used to be only two and they rarely competed with one another. Super-angels compete with both angels and VCs. That's going to change the rules about how to raise money. I don't know yet what the new rules will be, but it looks like most of the changes will be for the better.

A super-angel has some of the qualities of an angel, and some of the qualities of a VC. They're usually individuals, like angels. In fact many of the current super-angels were initially angels of the classic type. But like VCs, they invest other people's money. This allows them to invest larger amounts than angels: a typical super-angel investment is currently about \$100k. They make investment decisions quickly, like angels. And they make a lot more investments per partner than VCs—up to 10 times as many.

The fact that super-angels invest other people's money makes them doubly alarming to VCs. They don't just compete for startups; they also compete for investors. What super-angels really are is a new form of fast-moving, lightweight VC fund. And those of us in the technology world know what usually happens when something comes along that can be described in terms like that. Usually it's the replacement.

Will it be? As of now, few of the startups that take money from super-angels are ruling out taking VC money. They're just postponing it. But that's still a problem for VCs. Some of the startups that postpone raising VC money may do so well on the angel money they raise that they never bother to raise more. And those who do raise VC rounds will be able to get higher valuations when they do. If the best startups get 10x higher valuations when they raise series A rounds, that would cut VCs' returns from winners at least tenfold. [\[2\]](#)

So I think VC funds are seriously threatened by the super-angels. But one thing that may save them to some extent is the uneven distribution of startup outcomes: practically all the returns are concentrated in a few big successes. The expected value of a startup is the percentage chance it's Google. So to the extent that winning is a matter of absolute returns, the super-angels could win practically all the battles for individual startups and yet lose the war, if they merely failed to get those few big winners. And there's a chance that could happen, because the top VC funds have better brands, and can also do more for their portfolio companies. [\[3\]](#)

Because super-angels make more investments per partner, they have less partner per investment. They can't pay as much attention to you as a VC on your board

could. How much is that extra attention worth? It will vary enormously from one partner to another. There's no consensus yet in the general case. So for now this is something startups are deciding individually.

Till now, VCs' claims about how much value they added were sort of like the government's. Maybe they made you feel better, but you had no choice in the matter, if you needed money on the scale only VCs could supply. Now that VCs have competitors, that's going to put a market price on the help they offer. The interesting thing is, no one knows yet what it will be.

Do startups that want to get really big need the sort of advice and connections only the top VCs can supply? Or would super-angel money do just as well? The VCs will say you need them, and the super-angels will say you don't. But the truth is, no one knows yet, not even the VCs and super-angels themselves. All the super-angels know is that their new model seems promising enough to be worth trying, and all the VCs know is that it seems promising enough to worry about.

## **Rounds**

Whatever the outcome, the conflict between VCs and super-angels is good news for founders. And not just for the obvious reason that more competition for deals means better terms. The whole shape of deals is changing.

One of the biggest differences between angels and VCs is the amount of your company they want. VCs want a lot. In a series A round they want a third of your company, if they can get it. They don't care much how much they pay for it, but they want a lot because the number of series A investments they can do is so small. In a traditional series A investment, at least one partner from the VC fund takes a seat on your board. [4] Since board seats last about 5 years and each partner can't handle more than about 10 at once, that means a VC fund can only do about 2 series A deals per partner per year. And that means they need to get as much of the company as they can in each one. You'd have to be a very promising startup indeed to get a VC to use up one of his 10 board seats for only a few percent of you.

Since angels generally don't take board seats, they don't have this constraint. They're happy to buy only a few percent of you. And although the super-angels are in most respects mini VC funds, they've retained this critical property of angels. They don't take board seats, so they don't need a big percentage of your company.

Though that means you'll get correspondingly less attention from them, it's good news in other respects. Founders never really liked giving up as much equity as VCs wanted. It was a lot of the company to give up in one shot. Most founders doing series A deals would prefer to take half as much money for half as much stock, and then see what valuation they could get for the second half of the stock after using the first half of the money to increase its value. But VCs never offered that option.

Now startups have another alternative. Now it's easy to raise angel rounds about half the size of series A rounds. Many of the startups we fund are taking this route, and I predict that will be true of startups in general.

A typical big angel round might be \$600k on a convertible note with a valuation cap of \$4 million premoney. Meaning that when the note converts into stock (in a later round, or upon acquisition), the investors in that round will get .6 / 4.6, or 13% of the company. That's a lot less than the 30 to 40% of the company you usually give up in a series A round if you do it so early. [5]

But the advantage of these medium-sized rounds is not just that they cause less dilution. You also lose less control. After an angel round, the founders almost always still have control of the company, whereas after a series A round they often don't. The traditional board structure after a series A round is two founders, two VCs, and a (supposedly) neutral fifth person. Plus series A terms usually give the investors a veto over various kinds of important decisions, including selling the company. Founders usually have a lot of de facto control after a series A, as long as things are going well. But that's not the same as just being able to do what you want, like you could before.

A third and quite significant advantage of angel rounds is that they're less stressful to raise. Raising a traditional series A round has in the past taken weeks, if not months. When a VC firm can only do 2 deals per partner per year, they're careful about which they do. To get a traditional series A round you have to go through a series of meetings, culminating in a full partner meeting where the firm as a whole says yes or no. That's the really scary part for founders: not just that series A rounds take so long, but at the end of this long process the VCs might still say no. The chance of getting rejected after the full partner meeting averages about 25%. At some firms it's over 50%.

Fortunately for founders, VCs have been getting a lot faster. Nowadays Valley VCs are more likely to take 2 weeks than 2 months. But they're still not as fast as angels and super-angels, the most decisive of whom sometimes decide in hours.

Raising an angel round is not only quicker, but you get feedback as it progresses. An angel round is not an all or nothing thing like a series A. It's composed of multiple investors with varying degrees of seriousness, ranging from the upstanding ones who commit unequivocally to the jerks who give you lines like "come back to me to fill out the round." You usually start collecting money from the most committed investors and work your way out toward the ambivalent ones, whose interest increases as the round fills up.

But at each point you know how you're doing. If investors turn cold you may have to raise less, but when investors in an angel round turn cold the process at least degrades gracefully, instead of blowing up in your face and leaving you with nothing, as happens if you get rejected by a VC fund after a full partner meeting. Whereas if investors seem hot, you can not only close the round faster, but now that convertible notes are becoming the norm, actually [raise the price](#) to reflect

demand.

## Valuation

However, the VCs have a weapon they can use against the super-angels, and they have started to use it. VCs have started making angel-sized investments too. The term "angel round" doesn't mean that all the investors in it are angels; it just describes the structure of the round. Increasingly the participants include VCs making investments of a hundred thousand or two. And when VCs invest in angel rounds they can do things that super-angels don't like. VCs are quite valuation-insensitive in angel rounds—partly because they are in general, and partly because they don't care that much about the returns on angel rounds, which they still view mostly as a way to recruit startups for series A rounds later. So VCs who invest in angel rounds can blow up the valuations for angels and super-angels who invest in them. [6]

Some super-angels seem to care about valuations. Several turned down YC-funded startups after Demo Day because their valuations were too high. This was not a problem for the startups; by definition a high valuation means enough investors were willing to accept it. But it was mysterious to me that the super-angels would quibble about valuations. Did they not understand that the big returns come from a few big successes, and that it therefore mattered far more which startups you picked than how much you paid for them?

After thinking about it for a while and observing certain other signs, I have a theory that explains why the super-angels may be smarter than they seem. It would make sense for super-angels to want low valuations if they're hoping to invest in startups that get bought early. If you're hoping to hit the next Google, you shouldn't care if the valuation is 20 million. But if you're looking for companies that are going to get bought for 30 million, you care. If you invest at 20 and the company gets bought for 30, you only get 1.5x. You might as well buy Apple.

So if some of the super-angels were looking for companies that could get acquired quickly, that would explain why they'd care about valuations. But why would they be looking for those? Because depending on the meaning of "quickly," it could actually be very profitable. A company that gets acquired for 30 million is a failure to a VC, but it could be a 10x return for an angel, and moreover, a *quick* 10x return. Rate of return is what matters in investing—not the multiple you get, but the multiple per year. If a super-angel gets 10x in one year, that's a higher rate of return than a VC could ever hope to get from a company that took 6 years to go public. To get the same rate of return, the VC would have to get a multiple of  $10^6$ —one million x. Even Google didn't come close to that.

So I think at least some super-angels are looking for companies that will get bought. That's the only rational explanation for focusing on getting the right valuations, instead of the right companies. And if so they'll be different to deal with than VCs. They'll be tougher on valuations, but more accommodating if you want to sell early.



## Prognosis

Who will win, the super-angels or the VCs? I think the answer to that is, some of each. They'll each become more like one another. The super-angels will start to invest larger amounts, and the VCs will gradually figure out ways to make more, smaller investments faster. A decade from now the players will be hard to tell apart, and there will probably be survivors from each group.

What does that mean for founders? One thing it means is that the high valuations startups are presently getting may not last forever. To the extent that valuations are being driven up by price-insensitive VCs, they'll fall again if VCs become more like super-angels and start to become more miserly about valuations. Fortunately if this does happen it will take years.

The short term forecast is more competition between investors, which is good news for you. The super-angels will try to undermine the VCs by acting faster, and the VCs will try to undermine the super-angels by driving up valuations. Which for founders will result in the perfect combination: funding rounds that close fast, with high valuations.

But remember that to get that combination, your startup will have to appeal to both super-angels and VCs. If you don't seem like you have the potential to go public, you won't be able to use VCs to drive up the valuation of an angel round.

There is a danger of having VCs in an angel round: the so-called signalling risk. If VCs are only doing it in the hope of investing more later, what happens if they don't? That's a signal to everyone else that they think you're lame.

How much should you worry about that? The seriousness of signalling risk depends on how far along you are. If by the next time you need to raise money, you have graphs showing rising revenue or traffic month after month, you don't have to worry about any signals your existing investors are sending. Your results will speak for themselves. [7]

Whereas if the next time you need to raise money you won't yet have concrete results, you may need to think more about the message your investors might send if they don't invest more. I'm not sure yet how much you have to worry, because this whole phenomenon of VCs doing angel investments is so new. But my instincts tell me you don't have to worry much. Signalling risk smells like one of those things founders worry about that's not a real problem. As a rule, the only thing that can kill a good startup is the startup itself. Startups hurt themselves way more often than competitors hurt them, for example. I suspect signalling risk is in this category too.

One thing YC-funded startups have been doing to mitigate the risk of taking money from VCs in angel rounds is not to take too much from any one VC. Maybe that will help, if you have the luxury of turning down money.

Fortunately, more and more startups will. After decades of competition that could best be described as intramural, the startup funding business is finally getting some real competition. That should last several years at least, and maybe a lot longer. Unless there's some huge market crash, the next couple years are going to be a good time for startups to raise money. And that's exciting because it means lots more startups will happen.

## Notes

[1] I've also heard them called "Mini-VCs" and "Micro-VCs." I don't know which name will stick.

There were a couple predecessors. Ron Conway had angel funds starting in the 1990s, and in some ways First Round Capital is closer to a super-angel than a VC fund.

[2] It wouldn't cut their overall returns tenfold, because investing later would probably (a) cause them to lose less on investments that failed, and (b) not allow them to get as large a percentage of startups as they do now. So it's hard to predict precisely what would happen to their returns.

[3] The brand of an investor derives mostly from the success of their portfolio companies. The top VCs thus have a big brand advantage over the super-angels. They could make it self-perpetuating if they used it to get all the best new startups. But I don't think they'll be able to. To get all the best startups, you have to do more than make them want you. You also have to want them; you have to recognize them when you see them, and that's much harder. Super-angels will snap up stars that VCs miss. And that will cause the brand gap between the top VCs and the super-angels gradually to erode.

[4] Though in a traditional series A round VCs put two partners on your board, there are signs now that VCs may begin to conserve board seats by switching to what used to be considered an angel-round board, consisting of two founders and one VC. Which is also to the founders' advantage if it means they still control the company.

[5] In a series A round, you usually have to give up more than the actual amount of stock the VCs buy, because they insist you dilute yourselves to set aside an "option pool" as well. I predict this practice will gradually disappear though.

[6] The best thing for founders, if they can get it, is a convertible note with no

valuation cap at all. In that case the money invested in the angel round just converts into stock at the valuation of the next round, no matter how large. Angels and super-angels tend not to like uncapped notes. They have no idea how much of the company they're buying. If the company does well and the valuation of the next round is high, they may end up with only a sliver of it. So by agreeing to uncapped notes, VCs who don't care about valuations in angel rounds can make offers that super-angels hate to match.

[7] Obviously signalling risk is also not a problem if you'll never need to raise more money. But startups are often mistaken about that.

**Thanks** to Sam Altman, John Bautista, Patrick Collison, James Lindenbaum, Reid Hoffman, Jessica Livingston and Harj Taggar for reading drafts of this.

# What We Look for in Founders

October 2010

*(I wrote this for Forbes, who asked me to write something about the qualities we look for in founders. In print they had to cut the last item because they didn't have room.)*

## 1. Determination

This has turned out to be the most important quality in startup founders. We thought when we started Y Combinator that the most important quality would be intelligence. That's the myth in the Valley. And certainly you don't want founders to be stupid. But as long as you're over a certain threshold of intelligence, what matters most is determination. You're going to hit a lot of obstacles. You can't be the sort of person who gets [demoralized](#) easily.

Bill Clerico and Rich Aberman of [WePay](#) are a good example. They're doing a finance startup, which means endless negotiations with big, bureaucratic companies. When you're starting a startup that depends on deals with big companies to exist, it often feels like they're trying to ignore you out of existence. But when Bill Clerico starts calling you, you may as well do what he asks, because he is not going away.

## 2. Flexibility

You do not however want the sort of determination implied by phrases like "don't give up on your dreams." The world of startups is so unpredictable that you need to be able to modify your dreams on the fly. The best metaphor I've found for the combination of determination and flexibility you need is a [running back](#). He's determined to get downfield, but at any given moment he may need to go sideways or even backwards to get there.

The current record holder for flexibility may be Daniel Gross of [Greplin](#). He applied to YC with some bad ecommerce idea. We told him we'd fund him if he did something else. He thought for a second, and said ok. He then went through two more ideas before settling on Greplin. He'd only been working on it for a couple days when he presented to investors at Demo Day, but he got a lot of interest. He always seems to land on his feet.

### 3. Imagination

Intelligence does matter a lot of course. It seems like the type that matters most is imagination. It's not so important to be able to solve predefined problems quickly as to be able to come up with surprising new ideas. In the startup world, most good ideas [seem bad](#) initially. If they were obviously good, someone would already be doing them. So you need the kind of intelligence that produces ideas with just the right level of craziness.

[Airbnb](#) is that kind of idea. In fact, when we funded Airbnb, we thought it was too crazy. We couldn't believe large numbers of people would want to stay in other people's places. We funded them because we liked the founders so much. As soon as we heard they'd been supporting themselves by selling Obama and McCain branded breakfast cereal, they were in. And it turned out the idea was on the right side of crazy after all.

### 4. Naughtiness

Though the most successful founders are usually good people, they tend to have a piratical gleam in their eye. They're not Goody Two-Shoes type good. Morally, they care about getting the big questions right, but not about observing proprieties. That's why I'd use the word naughty rather than evil. They delight in [breaking rules](#), but not rules that matter. This quality may be redundant though; it may be implied by imagination.

Sam Altman of [Loopt](#) is one of the most successful alumni, so we asked him what question we could put on the Y Combinator application that would help us discover more people like him. He said to ask about a time when they'd hacked something to their advantage—hacked in the sense of beating the system, not breaking into computers. It has become one of the questions we pay most attention to when judging applications.

### 5. Friendship

Empirically it seems to be hard to start a startup with just [one founder](#). Most of the big successes have two or three. And the relationship between the founders has to be strong. They must genuinely like one another, and work well together. Startups do to the relationship between the founders what a dog does to a sock: if it can be pulled apart, it will be.

Emmett Shear and Justin Kan of [Justin.tv](#) are a good example of close friends who work well together. They've known each other since second grade. They can practically read one another's minds. I'm sure they argue, like all founders, but I have never once sensed any unresolved tension between them.

**Thanks** to Jessica Livingston and Chris Steiner for reading drafts of this.

# Tablets

December 2010

I was thinking recently how inconvenient it was not to have a general term for iPhones, iPads, and the corresponding things running Android. The closest to a general term seems to be "mobile devices," but that (a) applies to any mobile phone, and (b) doesn't really capture what's distinctive about the iPad.

After a few seconds it struck me that what we'll end up calling these things is tablets. The only reason we even consider calling them "mobile devices" is that the iPhone preceded the iPad. If the iPad had come first, we wouldn't think of the iPhone as a phone; we'd think of it as a tablet small enough to hold up to your ear.

The iPhone isn't so much a phone as a replacement for a phone. That's an important distinction, because it's an early instance of what will become a common pattern. Many if not most of the special-purpose objects around us are going to be replaced by apps running on tablets.

This is already clear in cases like GPSes, music players, and cameras. But I think it will surprise people how many things are going to get replaced. We funded one startup that's [replacing keys](#). The fact that you can change font sizes easily means the iPad effectively replaces reading glasses. I wouldn't be surprised if by playing some clever tricks with the accelerometer you could even replace the bathroom scale.

The advantages of doing things in software on a single device are so great that everything that can get turned into software will. So for the next couple years, a good [recipe for startups](#) will be to look around you for things that people haven't realized yet can be made unnecessary by a tablet app.

In 1938 Buckminster Fuller coined the term [ephemeralization](#) to describe the increasing tendency of physical machinery to be replaced by what we would now call software. The reason tablets are going to take over the world is not (just) that Steve Jobs and Co are industrial design wizards, but because they have this force behind them. The iPhone and the iPad have effectively drilled a hole that will allow ephemeralization to flow into a lot of new areas. No one who has studied the history of technology would want to underestimate the power of that force.

I worry about the power Apple could have with this force behind them. I don't want

to see another era of client monoculture like the Microsoft one in the 80s and 90s. But if ephemeralization is one of the main forces driving the spread of tablets, that suggests a way to compete with Apple: be a better platform for it.

It has turned out to be a great thing that Apple tablets have accelerometers in them. Developers have used the accelerometer in ways Apple could never have imagined. That's the nature of platforms. The more versatile the tool, the less you can predict how people will use it. So tablet makers should be thinking: what else can we put in there? Not merely hardware, but software too. What else can we give developers access to? Give hackers an inch and they'll take you a mile.

**Thanks** to Sam Altman, Paul Buchheit, Jessica Livingston, and Robert Morris for reading drafts of this.



# Founder Control

December 2010

Someone we funded is talking to VCs now, and asked me how common it was for a startup's founders to retain control of the board after a series A round. He said VCs told him this almost never happened.

Ten years ago that was true. In the past, founders rarely kept control of the board through a series A. The traditional series A board consisted of two founders, two VCs, and one independent member. More recently the recipe is often one founder, one VC, and one independent. In either case the founders lose their majority.

But not always. Mark Zuckerberg kept control of Facebook's board through the series A and still has it today. Mark Pincus has kept control of Zynga's too. But are these just outliers? How common is it for founders to keep control after an A round? I'd heard of several cases among the companies we've funded, but I wasn't sure how many there were, so I emailed the ycfounders list.

The replies surprised me. In a dozen companies we've funded, the founders still had a majority of the board seats after the series A round.

I feel like we're at a tipping point here. A lot of VCs still act as if founders retaining board control after a series A is unheard-of. A lot of them try to make you feel bad if you even ask — as if you're a noob or a control freak for wanting such a thing. But the founders I heard from aren't noobs or control freaks. Or if they are, they are, like Mark Zuckerberg, the kind of noobs and control freaks VCs should be trying to fund more of.

Founders retaining control after a series A is clearly heard-of. And barring financial catastrophe, I think in the coming year it will become the norm.

Control of a company is a more complicated matter than simply outvoting other parties in board meetings. Investors usually get vetos over certain big decisions, like selling the company, regardless of how many board seats they have. And board votes are rarely split. Matters are decided in the discussion preceding the vote, not in the vote itself, which is usually unanimous. But if opinion is divided in such discussions, the side that knows it would lose in a vote will tend to be less insistent. That's what board control means in practice. You don't simply get to do

whatever you want; the board still has to act in the interest of the shareholders; but if you have a majority of board seats, then your opinion about what's in the interest of the shareholders will tend to prevail.

So while board control is not total control, it's not imaginary either. There's inevitably a difference in how things feel within the company. Which means if it becomes the norm for founders to retain board control after a series A, that will change the way things feel in the whole startup world.

The switch to the new norm may be surprisingly fast, because the startups that can retain control tend to be the best ones. They're the ones that set the trends, both for other startups and for VCs.

A lot of the reason VCs are harsh when negotiating with startups is that they're embarrassed to go back to their partners looking like they got beaten. When they sign a termsheet, they want to be able to brag about the good terms they got. A lot of them don't care that much personally about whether founders keep board control. They just don't want to seem like they had to make concessions. Which means if letting the founders keep control stops being perceived as a concession, it will rapidly become much more common.

Like a lot of changes that have been forced on VCs, this change won't turn out to be as big a problem as they might think. VCs will still be able to convince; they just won't be able to compel. And the startups where they have to resort to compulsion are not the ones that matter anyway. VCs make most of their money from a few big hits, and those aren't them.

Knowing that founders will keep control of the board may even help VCs pick better. If they know they can't fire the founders, they'll have to choose founders they can trust. And that's who they should have been choosing all along.

**Thanks** to Sam Altman, John Bautista, Trevor Blackwell, Paul Buchheit, Brian Chesky, Bill Clerico, Patrick Collison, Adam Goldstein, James Lindenbaum, Jessica Livingston, and Fred Wilson for reading drafts of this.

# Subject: Airbnb

March 2011

Yesterday Fred Wilson published a remarkable [post](#) about missing [Airbnb](#). VCs miss good startups all the time, but it's extraordinarily rare for one to talk about it publicly till long afterward. So that post is further evidence what a rare bird Fred is. He's probably the nicest VC I know.

Reading Fred's post made me go back and look at the emails I exchanged with him at the time, trying to convince him to invest in Airbnb. It was quite interesting to read. You can see Fred's mind at work as he circles the deal.

Fred and the Airbnb founders have generously agreed to let me publish this email exchange (with one sentence redacted about something that's strategically important to Airbnb and not an important part of the conversation). It's an interesting illustration of an element of the startup ecosystem that few except the participants ever see: investors trying to convince one another to invest in their portfolio companies. Hundreds if not thousands of conversations of this type are happening now, but if one has ever been published, I haven't seen it. The Airbnbs themselves never even saw these emails at the time.

We do a lot of this behind the scenes stuff at YC, because we invest in such a large number of companies, and we invest so early that investors sometimes need a lot of convincing to see their merits. I don't always try as hard as this though. Fred must have found me quite annoying.

from: Paul Graham  
to: Fred Wilson, AirBedAndBreakfast Founders  
date: Fri, Jan 23, 2009 at 11:42 AM  
subject: meet the airbeds

One of the startups from the batch that just started, AirbedAndBreakfast, is in NYC right now meeting their users. (NYC is their biggest market.) I'd recommend meeting them if your schedule allows.

I'd been thinking to myself that though these guys were going to do really well, I should introduce them to angels, because VCs would never go for it. But then I thought maybe I should give you more credit. You'll certainly like meeting them. Be sure to ask about how they funded themselves with breakfast cereal.

There's no reason this couldn't be as big as Ebay. And this team is the right one to do it.

--pg

from: Brian Chesky  
to: Paul Graham  
cc: Nathan Blecharczyk, Joe Gebbia  
date: Fri, Jan 23, 2009 at 11:40 AM  
subject: Re: meet the airbeds

PG,

Thanks for the intro!

Brian

from: Paul Graham  
to: Brian Chesky  
cc: Nathan Blecharczyk, Joe Gebbia  
date: Fri, Jan 23, 2009 at 12:38 PM  
subject: Re: meet the airbeds

It's a longshot, at this stage, but if there was any VC who'd get you guys, it would be Fred. He is the least suburban-golf-playing VC I know.

He likes to observe startups for a while before acting, so don't be bummed if he seems ambivalent.

--pg

from: Fred Wilson  
to: Paul Graham,  
date: Sun, Jan 25, 2009 at 5:28 PM  
subject: Re: meet the airbeds

Thanks Paul

We are having a bit of a debate inside our partnership about the airbed concept. We'll finish that debate tomorrow in our weekly meeting and get back to you with our thoughts

Thanks

Fred

from: Paul Graham  
to: Fred Wilson  
date: Sun, Jan 25, 2009 at 10:48 PM  
subject: Re: meet the airbeds

I'd recommend having the debate after meeting them instead of before. We had big doubts about this idea, but they vanished on meeting the guys.

from: Fred Wilson  
to: Paul Graham  
date: Mon, Jan 26, 2009 at 11:08 AM  
subject: RE: meet the airbeds

We are still very suspect of this idea but will take a meeting as you suggest

Thanks

fred

from: Fred Wilson  
to: Paul Graham, AirBedAndBreakfast Founders  
date: Mon, Jan 26, 2009 at 11:09 AM  
subject: RE: meet the airbeds

Airbed team -

Are you still in NYC?

We'd like to meet if you are

Thanks

fred

from: Paul Graham  
to: Fred Wilson  
date: Mon, Jan 26, 2009 at 1:42 PM  
subject: Re: meet the airbeds

Ideas can morph. Practically every really big startup could say, five years later, "believe it or not, we started out doing \_\_\_\_." It just seemed a very good sign to me that these guys were actually on the ground in NYC hunting down (and understanding) their users. On top of several previous good signs.

--pg

from: Fred Wilson  
to: Paul Graham  
date: Sun, Feb 1, 2009 at 7:15 AM  
subject: Re: meet the airbeds

It's interesting

Our two junior team members were enthusiastic

The three "old guys" didn't get it

from: Paul Graham  
to: Fred Wilson  
date: Mon, Feb 9, 2009 at 5:58 PM  
subject: airbnb

The Airbeds just won the first poll among all the YC startups in their batch by a landslide. In the past this has not been a 100% indicator of success (if only anything were) but much better than random.

--pg

from: Fred Wilson  
to: Paul Graham  
date: Fri, Feb 13, 2009 at 5:29 PM  
subject: Re: airbnb

I met them today

They have an interesting business

I'm just not sure how big it's going to be

fred

from: Paul Graham  
to: Fred Wilson  
date: Sat, Feb 14, 2009 at 9:50 AM  
subject: Re: airbnb

Did they explain the long-term goal of being the market in accommodation the way eBay is in stuff? That seems like it would be huge. Hotels now are like airlines in the 1970s before they figured out how to increase their load factors.

from: Fred Wilson  
to: Paul Graham  
date: Tue, Feb 17, 2009 at 2:05 PM  
subject: Re: airbnb

They did but I am not sure I buy that

ABNB reminds me of Etsy in that it facilitates real commerce in a marketplace model directly between two people

So I think it can scale all the way to the bed and breakfast market

But I am not sure they can take on the hotel market

I could be wrong

But even so, if you include short term room rental, second home rental, bed and breakfast, and other similar classes of accommodations, you get to a pretty big opportunity

fred

from: Paul Graham  
to: Fred Wilson  
date: Wed, Feb 18, 2009 at 12:21 AM  
subject: Re: airbnb

So invest in them! They're very capital efficient. They would make an investor's money go a long way.

It's also counter-cyclical. They just arrived back from NYC, and when I asked them what was the most significant thing they'd observed, it was how many of their users actually needed to do these rentals to pay their rents.

--pg

from: Fred Wilson  
to: Paul Graham  
date: Wed, Feb 18, 2009 at 2:21 AM  
subject: Re: airbnb

There's a lot to like

I've done a few things, like intro it to my friends at Foundry who were investors in Service Metrics and understand this model

I am also talking to my friend Mark Pincus who had an idea like this a few years ago.

So we are working on it

Thanks for the lead

Fred

from: Paul Graham  
to: Fred Wilson  
date: Fri, Feb 20, 2009 at 10:00 PM  
subject: airbnb already spreading to pros

I know you're skeptical they'll ever get hotels, but there's a continuum between private sofas and hotel rooms, and they just moved one step further along it.

[link to an airbnb user]

This is after only a few months. I bet you they will get hotels eventually. It will start with small ones. Just wait till all the 10-room pensiones in Rome discover this site. And once it spreads to hotels, where is the point (in size of chain) at which it stops?

Once something becomes a big marketplace, you ignore it at your peril.

--pg

from: Fred Wilson  
to: Paul Graham  
date: Sat, Feb 21, 2009 at 4:26 AM  
subject: Re: airbnb already spreading to pros

That's true. It's also true that there are quite a few marketplaces out there that serve this same market

If you look at many of the people who list at ABNB, they list elsewhere too

I am not negative on this one, I am interested, but we are still in the gathering data phase.

fred



# The Patent Pledge

August 2011

I realized recently that we may be able to solve part of the patent problem without waiting for the government.

I've never been 100% sure whether patents help or hinder technological progress. When I was a kid I thought they helped. I thought they protected inventors from having their ideas stolen by big companies. Maybe that was truer in the past, when more things were physical. But regardless of whether patents are in general a good thing, there do seem to be bad ways of using them. And since bad uses of patents seem to be increasing, there is an increasing call for patent reform.

The problem with patent reform is that it has to go through the government. That tends to be slow. But recently I realized we can also attack the problem downstream. As well as pinching off the stream of patents at the point where they're issued, we may in some cases be able to pinch it off at the point where they're used.

One way of using patents that clearly does not encourage innovation is when established companies with bad products use patents to suppress small competitors with good products. This is the type of abuse we may be able to decrease without having to go through the government.

The way to do it is to get the companies that are above pulling this sort of trick to pledge publicly not to. Then the ones that won't make such a pledge will be very conspicuous. Potential employees won't want to work for them. And investors, too, will be able to see that they're the sort of company that competes by litigation rather than by making good products.

Here's the pledge:

No first use of software patents against companies with less than 25 people.

I've deliberately traded precision for brevity. The patent pledge is not legally binding. It's like Google's "Don't be evil." They don't define what evil is, but by publicly saying that, they're saying they're willing to be held to a standard that, say, Altria is not. And though constraining, "Don't be evil" has been good for Google. Technology companies win by attracting the most productive people, and

the most productive people are attracted to employers who hold themselves to a higher standard than the law requires. [1]

The patent pledge is in effect a narrower but open source "Don't be evil." I encourage every technology company to adopt it. If you want to help fix patents, encourage your employer to.

Already most technology companies wouldn't sink to using patents on startups. You don't see Google or Facebook suing startups for patent infringement. They don't need to. So for the better technology companies, the patent pledge requires no change in behavior. They're just promising to do what they'd do anyway. And when all the companies that won't use patents on startups have said so, the holdouts will be very conspicuous.

The patent pledge doesn't fix every problem with patents. It won't stop patent trolls, for example; they're already pariahs. But the problem the patent pledge does fix may be more serious than the problem of patent trolls. Patent trolls are just parasites. A clumsy parasite may occasionally kill the host, but that's not its goal. Whereas companies that sue startups for patent infringement generally do it with explicit goal of keeping their product off the market.

Companies that use patents on startups are attacking innovation at the root. Now there's something any individual can do about this problem, without waiting for the government: ask companies where they stand.

[Patent Pledge Site](#)

## **Notes:**

[1] Because the pledge is deliberately vague, we're going to need common sense when interpreting it. And even more vice versa: the pledge is vague in order to make people use common sense when interpreting it.

So for example I've deliberately avoided saying whether the 25 people have to be employees, or whether contractors count too. If a company has to split hairs that fine about whether a suit would violate the patent pledge, it's probably still a dick move.

[The Investment That Didn't Happen](#)



# Why Startup Hubs Work

October 2011

If you look at a list of US cities sorted by population, the number of successful startups per capita varies by orders of magnitude. Somehow it's as if most places were sprayed with startupicide.

I wondered about this for years. I could see the average town was like a roach motel for startup ambitions: smart, ambitious people went in, but no startups came out. But I was never able to figure out exactly what happened inside the motel—exactly what was killing all the potential startups. [1]

A couple weeks ago I finally figured it out. I was framing the question wrong. The problem is not that most towns kill startups. It's that death is the [default](#) for startups, and most towns don't save them. Instead of thinking of most places as being sprayed with startupicide, it's more accurate to think of startups as all being poisoned, and a few places being sprayed with the antidote.

Startups in other places are just doing what startups naturally do: fail. The real question is, what's *saving* startups in places like Silicon Valley? [2]

## Environment

I think there are two components to the antidote: being in a place where startups are the cool thing to do, and chance meetings with people who can help you. And what drives them both is the number of startup people around you.

The first component is particularly helpful in the first stage of a startup's life, when you go from merely having an interest in starting a company to actually doing it. It's quite a leap to start a startup. It's an unusual thing to do. But in Silicon Valley it seems normal. [3]

In most places, if you start a startup, people treat you as if you're unemployed. People in the Valley aren't automatically impressed with you just because you're starting a company, but they pay attention. Anyone who's been here any amount of time knows not to default to skepticism, no matter how inexperienced you seem or how unpromising your idea sounds at first, because they've all seen inexperienced founders with unpromising sounding ideas who a few years later

were billionaires.

Having people around you care about what you're doing is an extraordinarily [powerful](#) force. Even the most willful people are susceptible to it. About a year after we started Y Combinator I said something to a partner at a well known VC firm that gave him the (mistaken) impression I was considering starting another startup. He responded so eagerly that for about half a second I found myself considering doing it.

In most other cities, the prospect of starting a startup just doesn't seem real. In the Valley it's not only real but fashionable. That no doubt causes a lot of people to start startups who shouldn't. But I think that's ok. Few people are suited to running a startup, and it's very hard to predict beforehand which are (as I know all too well from being in the business of trying to predict beforehand), so lots of people starting startups who shouldn't is probably the optimal state of affairs. As long as you're at a point in your life when you can bear the risk of failure, the best way to find out if you're suited to running a startup is to [try it](#).

## Chance

The second component of the antidote is chance meetings with people who can help you. This force works in both phases: both in the transition from the desire to start a startup to starting one, and the transition from starting a company to succeeding. The power of chance meetings is more variable than people around you caring about startups, which is like a sort of background radiation that affects everyone equally, but at its strongest it is far stronger.

Chance meetings produce miracles to compensate for the disasters that characteristically befall startups. In the Valley, terrible things happen to startups all the time, just like they do to startups everywhere. The reason startups are more likely to make it here is that great things happen to them too. In the Valley, lightning has a sign bit.

For example, you start a site for college students and you decide to move to the Valley for the summer to work on it. And then on a random suburban street in Palo Alto you happen to run into Sean Parker, who understands the domain really well because he started a similar startup himself, and also knows all the investors. And moreover has advanced views, for 2004, on founders retaining [control](#) of their companies.

You can't say precisely what the miracle will be, or even for sure that one will happen. The best one can say is: if you're in a startup hub, unexpected good things will probably happen to you, especially if you deserve them.

I bet this is true even for startups we fund. Even with us working to make things happen for them on purpose rather than by accident, the frequency of helpful chance meetings in the Valley is so high that it's still a significant increment on what we can deliver.

Chance meetings play a role like the role relaxation plays in having ideas. Most people have had the experience of working hard on some problem, not being able to solve it, giving up and going to bed, and then thinking of the answer in the shower in the morning. What makes the answer appear is letting your thoughts [drift](#) a bit—and thus drift off the wrong path you'd been pursuing last night and onto the right one adjacent to it.

Chance meetings let your acquaintance drift in the same way taking a shower lets your thoughts drift. The critical thing in both cases is that they drift just the right amount. The meeting between Larry Page and Sergey Brin was a good example. They let their acquaintance drift, but only a little; they were both meeting someone they had a lot in common with.

For Larry Page the most important component of the antidote was Sergey Brin, and vice versa. The antidote is [people](#). It's not the physical infrastructure of Silicon Valley that makes it work, or the weather, or anything like that. Those helped get it started, but now that the reaction is self-sustaining what drives it is the people.

Many observers have noticed that one of the most distinctive things about startup hubs is the degree to which people help one another out, with no expectation of getting anything in return. I'm not sure why this is so. Perhaps it's because startups are less of a zero sum game than most types of business; they are rarely killed by competitors. Or perhaps it's because so many startup founders have backgrounds in the sciences, where collaboration is encouraged.

A large part of YC's function is to accelerate that process. We're a sort of Valley within the Valley, where the density of people working on startups and their willingness to help one another are both artificially amplified.

## Numbers

Both components of the antidote—an environment that encourages startups, and chance meetings with people who help you—are driven by the same underlying cause: the number of startup people around you. To make a startup hub, you need a *lot* of people interested in startups.

There are three reasons. The first, obviously, is that if you don't have enough density, the chance meetings don't happen. [\[4\]](#) The second is that different startups need such different things, so you need a lot of people to supply each startup with what they need most. Sean Parker was exactly what Facebook needed in 2004. Another startup might have needed a database guy, or someone with connections in the movie business.

This is one of the reasons we fund such a large number of companies, incidentally. The bigger the community, the greater the chance it will contain the person who has that one thing you need most.

The third reason you need a lot of people to make a startup hub is that once you have enough people interested in the same problem, they start to set the social norms. And it is a particularly valuable thing when the atmosphere around you encourages you to do something that would otherwise seem too ambitious. In most places the atmosphere pulls you back toward the mean.

I flew into the Bay Area a few days ago. I notice this every time I fly over the Valley: somehow you can sense something is going on. Obviously you can sense prosperity in how well kept a place looks. But there are different kinds of prosperity. Silicon Valley doesn't look like Boston, or New York, or LA, or DC. I tried asking myself what word I'd use to describe the feeling the Valley radiated, and the word that came to mind was optimism.

## Notes

[1] I'm not saying it's impossible to succeed in a city with few other startups, just harder. If you're sufficiently good at generating your own morale, you can survive without external encouragement. Wufoo was based in Tampa and they succeeded. But the Wufoos are exceptionally disciplined.

[2] Incidentally, this phenomenon is not limited to startups. Most unusual ambitions fail, unless the person who has them manages to find the right sort of community.

[3] Starting a company is common, but starting a startup is rare. I've talked about the distinction between the two elsewhere, but essentially a startup is a new business designed for scale. Most new businesses are service businesses and except in rare cases those don't scale.

[4] As I was writing this, I had a demonstration of the density of startup people in the Valley. Jessica and I bicycled to University Ave in Palo Alto to have lunch at the fabulous Oren's Hummus. As we walked in, we met Charlie Cheever sitting near the door. Selina Tobaccowala stopped to say hello on her way out. Then Josh Wilson came in to pick up a take out order. After lunch we went to get frozen yogurt. On the way we met Rajat Suri. When we got to the yogurt place, we found Dave Shen there, and as we walked out we ran into Yuri Sagalov. We walked with him for a block or so and we ran into Muzzammil Zaveri, and then a block later we met Aydin Senkut. This is everyday life in Palo Alto. I wasn't trying to meet people; I was just having lunch. And I'm sure for every startup founder or investor I saw that I knew, there were 5 more I didn't. If Ron Conway had been with us he would have met 30 people he knew.

**Thanks** to Sam Altman, Paul Buchheit, Jessica Livingston, and Harj Taggar for reading drafts of this.



# Snapshot: Viaweb, June 1998

January 2012

A few hours before the Yahoo acquisition was announced in June 1998 I took a [snapshot of Viaweb's site](#). I thought it might be interesting to look at one day.

The first thing one notices is how tiny the pages are. Screens were a lot smaller in 1998. If I remember correctly, our frontpage used to just fit in the size window people typically used then.

Browsers then (IE 6 was still 3 years in the future) had few fonts and they weren't antialiased. If you wanted to make pages that looked good, you had to render display text as images.

You may notice a certain similarity between the Viaweb and [Y Combinator](#) logos. We did that as an inside joke when we started YC. Considering how basic a red circle is, it seemed surprising to me when we started Viaweb how few other companies used one as their logo. A bit later I realized [why](#).

On the [Company page](#) you'll notice a mysterious individual called John McArtyem. Robert Morris (aka Rtm) was so publicity averse after the [Worm](#) that he didn't want his name on the site. I managed to get him to agree to a compromise: we could use his bio but not his name. He has since [relaxed](#) a bit on that point.

Trevor graduated at about the same time the acquisition closed, so in the course of 4 days he went from impecunious grad student to millionaire PhD. The culmination of my career as a writer of press releases was one [celebrating his graduation](#), illustrated with a drawing I did of him during a meeting.

(Trevor also appears as [Trevino Bagwell](#) in our directory of web designers merchants could hire to build stores for them. We inserted him as a ringer in case some competitor tried to spam our web designers. We assumed his logo would deter any actual customers, but it did not.)

Back in the 90s, to get users you had to get mentioned in magazines and newspapers. There were not the same ways to get found online that there are today. So we used to pay a [PR firm](#) \$16,000 a month to get us mentioned in the press. Fortunately reporters [liked us](#).

In our [advice about getting traffic from search engines](#) (I don't think the term SEO had been coined yet), we say there are only 7 that matter: Yahoo, AltaVista, Excite, WebCrawler, InfoSeek, Lycos, and HotBot. Notice anything missing? Google was incorporated that September.

We supported online transactions via a company called [Cybercash](#), since if we lacked that feature we'd have gotten beaten up in product comparisons. But Cybercash was so bad and most stores' order volumes were so low that it was better if merchants processed orders like phone orders. We had a page in our site trying to [talk merchants out of doing real time authorizations](#).

The whole site was organized like a funnel, directing people to the [test drive](#). It was a novel thing to be able to try out software online. We put cgi-bin in our dynamic urls to fool competitors about how our software worked.

We had some [well known users](#). Needless to say, Frederick's of Hollywood got the most traffic. We charged a flat fee of \$300/month for big stores, so it was a little alarming to have users who got lots of traffic. I once calculated how much Frederick's was costing us in bandwidth, and it was about \$300/month.

Since we hosted all the stores, which together were getting just over 10 million page views per month in June 1998, we consumed what at the time seemed a lot of bandwidth. We had 2 T1s (3 Mb/sec) coming into our offices. In those days there was no AWS. Even colocating servers seemed too risky, considering how often things went wrong with them. So we had our servers in our offices. Or more precisely, in Trevor's office. In return for the unique privilege of sharing his office with no other humans, he had to share it with 6 shrieking tower servers. His office was nicknamed the Hot Tub on account of the heat they generated. Most days his stack of window air conditioners could keep up.

For describing pages, we had a template language called [RTML](#), which supposedly stood for something, but which in fact I named after Rtm. RTML was Common Lisp augmented by some macros and libraries, and concealed under a structure editor that made it look like it had syntax.

Since we did continuous releases, our software didn't actually have versions. But in those days the trade press expected versions, so we made them up. If we wanted to get lots of attention, we made the version number [an integer](#). That "version 4.0" icon was generated by our own button generator, incidentally. The whole Viaweb site was made with our software, even though it wasn't an online store, because we wanted to experience what our users did.

At the end of 1997, we released a general purpose shopping search engine called [Shopfind](#). It was pretty advanced for the time. It had a programmable crawler that could crawl most of the different stores online and pick out the products.

# Schlep Blindness

January 2012

There are great startup ideas lying around unexploited right under our noses. One reason we don't see them is a phenomenon I call *schlep blindness*. Schlep was originally a Yiddish word but has passed into general use in the US. It means a tedious, unpleasant task.

No one likes schleps, but hackers especially dislike them. Most hackers who start startups wish they could do it by just writing some clever software, putting it on a server somewhere, and watching the money roll in—without ever having to talk to users, or negotiate with other companies, or deal with other people's broken code. Maybe that's possible, but I haven't seen it.

One of the many things we do at Y Combinator is teach hackers about the inevitability of schleps. No, you can't start a startup by just writing code. I remember going through this realization myself. There was a point in 1995 when I was still trying to convince myself I could start a company by just writing code. But I soon learned from experience that schleps are not merely inevitable, but pretty much what business consists of. A company is defined by the schleps it will undertake. And schleps should be dealt with the same way you'd deal with a cold swimming pool: just jump in. Which is not to say you should seek out unpleasant work per se, but that you should never shrink from it if it's on the path to something great.

The most dangerous thing about our dislike of schleps is that much of it is unconscious. Your unconscious won't even let you see ideas that involve painful schleps. That's schlep blindness.

The phenomenon isn't limited to startups. Most people don't consciously decide not to be in as good physical shape as Olympic athletes, for example. Their unconscious mind decides for them, shrinking from the work involved.

The most striking example I know of schlep blindness is [Stripe](#), or rather Stripe's idea. For over a decade, every hacker who'd ever had to process payments online knew how painful the experience was. Thousands of people must have known about this problem. And yet when they started startups, they decided to build recipe sites, or aggregators for local events. Why? Why work on problems few care

much about and no one will pay for, when you could fix one of the most important components of the world's infrastructure? Because schlep blindness prevented people from even considering the idea of fixing payments.

Probably no one who applied to Y Combinator to work on a recipe site began by asking "should we fix payments, or build a recipe site?" and chose the recipe site. Though the idea of fixing payments was right there in plain sight, they never saw it, because their unconscious mind shrank from the complications involved. You'd have to make deals with banks. How do you do that? Plus you're moving money, so you're going to have to deal with fraud, and people trying to break into your servers. Plus there are probably all sorts of regulations to comply with. It's a lot more intimidating to start a startup like this than a recipe site.

That scariness makes ambitious ideas doubly valuable. In addition to their intrinsic value, they're like undervalued stocks in the sense that there's less demand for them among founders. If you pick an ambitious idea, you'll have less competition, because everyone else will have been frightened off by the challenges involved. (This is also true of starting a startup generally.)

How do you overcome schlep blindness? Frankly, the most valuable antidote to schlep blindness is probably ignorance. Most successful founders would probably say that if they'd known when they were starting their company about the obstacles they'd have to overcome, they might never have started it. Maybe that's one reason the most successful startups of all so often have young founders.

In practice the founders grow with the problems. But no one seems able to foresee that, not even older, more experienced founders. So the reason younger founders have an advantage is that they make two mistakes that cancel each other out. They don't know how much they can grow, but they also don't know how much they'll need to. Older founders only make the first mistake.

Ignorance can't solve everything though. Some ideas so obviously entail alarming schleps that anyone can see them. How do you see ideas like that? The trick I recommend is to take yourself out of the picture. Instead of asking "what problem should I solve?" ask "what problem do I wish someone else would solve for me?" If someone who had to process payments before Stripe had tried asking that, Stripe would have been one of the first things they wished for.

It's too late now to be Stripe, but there's plenty still broken in the world, if you know how to see it.

**Thanks** to Sam Altman, Paul Buchheit, Patrick Collison, Aaron Iba, Jessica Livingston, Emmett Shear, and Harj Taggar for reading drafts of this.



# A Word to the Resourceful

January 2012

A year ago I noticed a pattern in the least successful startups we'd funded: they all seemed hard to talk to. It felt as if there was some kind of wall between us. I could never quite tell if they understood what I was saying.

This caught my attention because earlier we'd noticed a pattern among the most successful startups, and it seemed to hinge on a different quality. We found the startups that did best were the ones with the sort of founders about whom we'd say "they can take care of themselves." The startups that do best are fire-and-forget in the sense that all you have to do is give them a lead, and they'll close it, whatever type of lead it is. When they're raising money, for example, you can do the initial intros knowing that if you wanted to you could stop thinking about it at that point. You won't have to babysit the round to make sure it happens. That type of founder is going to come back with the money; the only question is how much on what terms.

It seemed odd that the outliers at the two ends of the spectrum could be detected by what appeared to be unrelated tests. You'd expect that if the founders at one end were distinguished by the presence of quality  $x$ , at the other end they'd be distinguished by lack of  $x$ . Was there some kind of inverse relation between [resourcefulness](#) and being hard to talk to?

It turns out there is, and the key to the mystery is the old adage "a word to the wise is sufficient." Because this phrase is not only overused, but overused in an indirect way (by prepending the subject to some advice), most people who've heard it don't know what it means. What it means is that if someone is wise, all you have to do is say one word to them, and they'll understand immediately. You don't have to explain in detail; they'll chase down all the implications.

In much the same way that all you have to do is give the right sort of founder a one line intro to a VC, and he'll chase down the money. That's the connection. Understanding all the implications — even the inconvenient implications — of what someone tells you is a subset of resourcefulness. It's conversational resourcefulness.

Like real world resourcefulness, conversational resourcefulness often means doing

things you don't want to. Chasing down all the implications of what's said to you can sometimes lead to uncomfortable conclusions. The best word to describe the failure to do so is probably "denial," though that seems a bit too narrow. A better way to describe the situation would be to say that the unsuccessful founders had the sort of conservatism that comes from weakness. They traversed idea space as gingerly as a very old person traverses the physical world. [1]

The unsuccessful founders weren't stupid. Intellectually they were as capable as the successful founders of following all the implications of what one said to them. They just weren't eager to.

So being hard to talk to was not what was killing the unsuccessful startups. It was a sign of an underlying lack of resourcefulness. That's what was killing them. As well as failing to chase down the implications of what was said to them, the unsuccessful founders would also fail to chase down funding, and users, and sources of new ideas. But the most immediate evidence I had that something was amiss was that I couldn't talk to them.

## Notes

[1] A YC partner wrote:

My feeling with the bad groups is that coming into office hours, they've already decided what they're going to do and everything I say is being put through an internal process in their heads, which either desperately tries to munge what I've said into something that conforms with their decision or just outright dismisses it and creates a rationalization for doing so. They may not even be conscious of this process but that's what I think is happening when you say something to bad groups and they have that glazed over look. I don't think it's confusion or lack of understanding per se, it's this internal process at work.

With the good groups, you can tell that everything you say is being looked at with fresh eyes and even if it's dismissed, it's because of some logical reason e.g. "we already tried that" or "from speaking to our users that isn't what they'd like," etc. Those groups never have that glazed over look.

**Thanks** to Sam Altman, Patrick Collison, Aaron Iba, Jessica Livingston, Robert Morris, Harj Taggar, and Garry Tan for reading drafts of this.

# Frighteningly Ambitious Startup Ideas

March 2012

One of the more surprising things I've noticed while working on Y Combinator is how frightening the most ambitious startup ideas are. In this essay I'm going to demonstrate this phenomenon by describing some. Any one of them could make you a billionaire. That might sound like an attractive prospect, and yet when I describe these ideas you may notice you find yourself shrinking away from them.

Don't worry, it's not a sign of weakness. Arguably it's a sign of sanity. The biggest startup ideas are terrifying. And not just because they'd be a lot of work. The biggest ideas seem to threaten your identity: you wonder if you'd have enough ambition to carry them through.

There's a scene in *Being John Malkovich* where the nerdy hero encounters a very attractive, sophisticated woman. She says to him:

Here's the thing: If you ever got me, you wouldn't have a clue what to do with me.

That's what these ideas say to us.

This phenomenon is one of the most important things you can understand about startups. [\[1\]](#) You'd expect big startup ideas to be attractive, but actually they tend to repel you. And that has a bunch of consequences. It means these ideas are invisible to most people who try to think of startup ideas, because their subconscious filters them out. Even the most ambitious people are probably best off approaching them obliquely.

## 1. A New Search Engine

The best ideas are just on the right side of impossible. I don't know if this one is possible, but there are signs it might be. Making a new search engine means competing with Google, and recently I've noticed some cracks in their fortress.

The point when it became clear to me that Microsoft had lost their way was when they decided to get into the search business. That was not a natural move for Microsoft. They did it because they were afraid of Google, and Google was in the search business. But this meant (a) Google was now setting Microsoft's agenda, and (b) Microsoft's agenda consisted of stuff they weren't good at.



Microsoft : Google :: Google : Facebook.

That does not by itself mean there's room for a new search engine, but lately when using Google search I've found myself nostalgic for the old days, when Google was true to its own slightly aspy self. Google used to give me a page of the right answers, fast, with no clutter. Now the results seem inspired by the Scientologist principle that what's true is what's true for you. And the pages don't have the clean, sparse feel they used to. Google search results used to look like the output of a Unix utility. Now if I accidentally put the cursor in the wrong place, anything might happen.

The way to win here is to build the search engine all the hackers use. A search engine whose users consisted of the top 10,000 hackers and no one else would be in a very powerful position despite its small size, just as Google was when it was that search engine. And for the first time in over a decade the idea of switching seems thinkable to me.

Since anyone capable of starting this company is one of those 10,000 hackers, the route is at least straightforward: make the search engine you yourself want. Feel free to make it excessively hackerish. Make it really good for code search, for example. Would you like search queries to be Turing complete? Anything that gets you those 10,000 users is ipso facto good.

Don't worry if something you want to do will constrain you in the long term, because if you don't get that initial core of users, there won't be a long term. If you can just build something that you and your friends genuinely prefer to Google, you're already about 10% of the way to an IPO, just as Facebook was (though they probably didn't realize it) when they got all the Harvard undergrads.

## **2. Replace Email**

Email was not designed to be used the way we use it now. Email is not a messaging protocol. It's a todo list. Or rather, my inbox is a todo list, and email is the way things get onto it. But it is a disastrously bad todo list.

I'm open to different types of solutions to this problem, but I suspect that tweaking the inbox is not enough, and that email has to be replaced with a new protocol. This new protocol should be a todo list protocol, not a messaging protocol, although there is a degenerate case where what someone wants you to do is: read the following text.

As a todo list protocol, the new protocol should give more power to the recipient than email does. I want there to be more restrictions on what someone can put on my todo list. And when someone can put something on my todo list, I want them to tell me more about what they want from me. Do they want me to do something beyond just reading some text? How important is it? (There obviously has to be some mechanism to prevent people from saying everything is important.) When does it have to be done?

This is one of those ideas that's like an irresistible force meeting an immovable object. On one hand, entrenched protocols are impossible to replace. On the other, it seems unlikely that people in 100 years will still be living in the same email hell we do now. And if email is going to get replaced eventually, why not now?

If you do it right, you may be able to avoid the usual chicken and egg problem new protocols face, because some of the most powerful people in the world will be among the first to switch to it. They're all at the mercy of email too.

Whatever you build, make it fast. GMail has become painfully slow. [2] If you made something no better than GMail, but fast, that alone would let you start to pull users away from GMail.

GMail is slow because Google can't afford to spend a lot on it. But people will pay for this. I'd have no problem paying \$50 a month. Considering how much time I spend in email, it's kind of scary to think how much I'd be justified in paying. At least \$1000 a month. If I spend several hours a day reading and writing email, that would be a cheap way to make my life better.

### **3. Replace Universities**

People are all over this idea lately, and I think they're onto something. I'm reluctant to suggest that an institution that's been around for a millennium is finished just because of some mistakes they made in the last few decades, but certainly in the last few decades US universities seem to have been headed down the wrong path. One could do a lot better for a lot less money.

I don't think universities will disappear. They won't be replaced wholesale. They'll just lose the de facto monopoly on certain types of learning that they once had. There will be many different ways to learn different things, and some may look quite different from universities. Y Combinator itself is arguably one of them.

Learning is such a big problem that changing the way people do it will have a wave of secondary effects. For example, the name of the university one went to is treated by a lot of people (correctly or not) as a credential in its own right. If learning breaks up into many little pieces, credentialling may separate from it. There may even need to be replacements for campus social life (and oddly enough, YC even has aspects of that).

You could replace high schools too, but there you face bureaucratic obstacles that would slow down a startup. Universities seem the place to start.

### **4. Internet Drama**

Hollywood has been slow to embrace the Internet. That was a mistake, because I think we can now call a winner in the race between delivery mechanisms, and it is the Internet, not cable.

A lot of the reason is the horribleness of cable clients, also known as TVs. Our family didn't wait for Apple TV. We hated our last TV so much that a few months ago we replaced it with an iMac bolted to the wall. It's a little inconvenient to control it with a wireless mouse, but the overall experience is much better than the nightmare UI we had to deal with before.

Some of the attention people currently devote to watching movies and TV can be stolen by things that seem completely unrelated, like social networking apps. More can be stolen by things that are a little more closely related, like games. But there will probably always remain some residual demand for conventional drama, where you sit passively and watch as a plot happens. So how do you deliver drama via the Internet? Whatever you make will have to be on a larger scale than Youtube clips. When people sit down to watch a show, they want to know what they're going to get: either part of a series with familiar characters, or a single longer "movie" whose basic premise they know in advance.

There are two ways delivery and payment could play out. Either some company like Netflix or Apple will be the app store for entertainment, and you'll reach audiences through them. Or the would-be app stores will be too overreaching, or too technically inflexible, and companies will arise to supply payment and streaming a la carte to the producers of drama. If that's the way things play out, there will also be a need for such infrastructure companies.

## **5. The Next Steve Jobs**

I was talking recently to someone who knew Apple well, and I asked him if the people now running the company would be able to keep creating new things the way Apple had under Steve Jobs. His answer was simply "no." I already feared that would be the answer. I asked more to see how he'd qualify it. But he didn't qualify it at all. No, there will be no more great new stuff beyond whatever's currently in the pipeline. Apple's revenues may continue to rise for a long time, but as Microsoft shows, revenue is a lagging indicator in the technology business.

So if Apple's not going to make the next iPad, who is? None of the existing players. None of them are run by product visionaries, and empirically you can't seem to get those by hiring them. Empirically the way you get a product visionary as CEO is for him to found the company and not get fired. So the company that creates the next wave of hardware is probably going to have to be a startup.

I realize it sounds preposterously ambitious for a startup to try to become as big as Apple. But no more ambitious than it was for Apple to become as big as Apple, and they did it. Plus a startup taking on this problem now has an advantage the original Apple didn't: the example of Apple. Steve Jobs has shown us what's possible. That helps would-be successors both directly, as Roger Bannister did, by showing how much better you can do than people did before, and indirectly, as Augustus did, by lodging the idea in users' minds that a single person could unroll the future for them. [3]

Now Steve is gone there's a vacuum we can all feel. If a new company led boldly into the future of hardware, users would follow. The CEO of that company, the "next Steve Jobs," might not measure up to Steve Jobs. But he wouldn't have to. He'd just have to do a better job than Samsung and HP and Nokia, and that seems pretty doable.

## 6. Bring Back Moore's Law

The last 10 years have reminded us what Moore's Law actually says. Till about 2002 you could safely misinterpret it as promising that clock speeds would double every 18 months. Actually what it says is that circuit densities will double every 18 months. It used to seem pedantic to point that out. Not any more. Intel can no longer give us faster CPUs, just more of them.

This Moore's Law is not as good as the old one. Moore's Law used to mean that if your software was slow, all you had to do was wait, and the inexorable progress of hardware would solve your problems. Now if your software is slow you have to rewrite it to do more things in parallel, which is a lot more work than waiting.

It would be great if a startup could give us something of the old Moore's Law back, by writing software that could make a large number of CPUs look to the developer like one very fast CPU. There are several ways to approach this problem. The most ambitious is to try to do it automatically: to write a compiler that will parallelize our code for us. There's a name for this compiler, *the sufficiently smart compiler*, and it is a byword for impossibility. But is it really impossible? Is there no configuration of the bits in memory of a present day computer that is this compiler? If you really think so, you should try to prove it, because that would be an interesting result. And if it's not impossible but simply very hard, it might be worth trying to write it. The expected value would be high even if the chance of succeeding was low.

The reason the expected value is so high is web services. If you could write software that gave programmers the convenience of the way things were in the old days, you could offer it to them as a web service. And that would in turn mean that you got practically all the users.

Imagine there was another processor manufacturer that could still translate increased circuit densities into increased clock speeds. They'd take most of Intel's business. And since web services mean that no one sees their processors anymore, by writing the sufficiently smart compiler you could create a situation indistinguishable from you being that manufacturer, at least for the server market.

The least ambitious way of approaching the problem is to start from the other end, and offer programmers more parallelizable Lego blocks to build programs out of, like Hadoop and MapReduce. Then the programmer still does much of the work of optimization.

There's an intriguing middle ground where you build a semi-automatic weapon—where there's a human in the loop. You make something that looks to the user like the sufficiently smart compiler, but inside has people, using highly developed optimization tools to find and eliminate bottlenecks in users' programs. These people might be your employees, or you might create a marketplace for optimization.

An optimization marketplace would be a way to generate the sufficiently smart compiler piecemeal, because participants would immediately start writing bots. It would be a curious state of affairs if you could get to the point where everything could be done by bots, because then you'd have made the sufficiently smart compiler, but no one person would have a complete copy of it.

I realize how crazy all this sounds. In fact, what I like about this idea is all the different ways in which it's wrong. The whole idea of focusing on optimization is counter to the general trend in software development for the last several decades. Trying to write the sufficiently smart compiler is by definition a mistake. And even if it weren't, compilers are the sort of software that's supposed to be created by open source projects, not companies. Plus if this works it will deprive all the programmers who take pleasure in making multithreaded apps of so much amusing complexity. The forum troll I have by now internalized doesn't even know where to begin in raising objections to this project. Now that's what I call a startup idea.

## **7. Ongoing Diagnosis**

But wait, here's another that could face even greater resistance: ongoing, automatic medical diagnosis.

One of my tricks for generating startup ideas is to imagine the ways in which we'll seem backward to future generations. And I'm pretty sure that to people 50 or 100 years in the future, it will seem barbaric that people in our era waited till they had symptoms to be diagnosed with conditions like heart disease and cancer.

For example, in 2004 Bill Clinton found he was feeling short of breath. Doctors discovered that several of his arteries were over 90% blocked and 3 days later he had a quadruple bypass. It seems reasonable to assume Bill Clinton has the best medical care available. And yet even he had to wait till his arteries were over 90% blocked to learn that the number was over 90%. Surely at some point in the future we'll know these numbers the way we now know something like our weight. Ditto for cancer. It will seem preposterous to future generations that we wait till patients have physical symptoms to be diagnosed with cancer. Cancer will show up on some sort of radar screen immediately.

(Of course, what shows up on the radar screen may be different from what we think of now as cancer. I wouldn't be surprised if at any given time we have ten or even hundreds of microcancers going at once, none of which normally amount to anything.)

A lot of the obstacles to ongoing diagnosis will come from the fact that it's going against the grain of the medical profession. The way medicine has always worked is that patients come to doctors with problems, and the doctors figure out what's wrong. A lot of doctors don't like the idea of going on the medical equivalent of what lawyers call a "fishing expedition," where you go looking for problems without knowing what you're looking for. They call the things that get discovered this way "incidentalomas," and they are something of a nuisance.

For example, a friend of mine once had her brain scanned as part of a study. She was horrified when the doctors running the study discovered what appeared to be a large tumor. After further testing, it turned out to be a harmless cyst. But it cost her a few days of terror. A lot of doctors worry that if you start scanning people with no symptoms, you'll get this on a giant scale: a huge number of false alarms that make patients panic and require expensive and perhaps even dangerous tests to resolve. But I think that's just an artifact of current limitations. If people were scanned all the time and we got better at deciding what was a real problem, my friend would have known about this cyst her whole life and known it was harmless, just as we do a birthmark.

There is room for a lot of startups here. In addition to the technical obstacles all startups face, and the bureaucratic obstacles all medical startups face, they'll be going against thousands of years of medical tradition. But it will happen, and it will be a great thing—so great that people in the future will feel as sorry for us as we do for the generations that lived before anaesthesia and antibiotics.

## **Tactics**

Let me conclude with some tactical advice. If you want to take on a problem as big as the ones I've discussed, don't make a direct frontal attack on it. Don't say, for example, that you're going to replace email. If you do that you raise too many expectations. Your employees and investors will constantly be asking "are we there yet?" and you'll have an army of haters waiting to see you fail. Just say you're building todo-list software. That sounds harmless. People can notice you've replaced email when it's a *fait accompli*. [\[4\]](#)

Empirically, the way to do really big things seems to be to start with deceptively small things. Want to dominate microcomputer software? Start by writing a Basic interpreter for a machine with a few thousand users. Want to make the universal web site? Start by building a site for Harvard undergrads to stalk one another.

Empirically, it's not just for other people that you need to start small. You need to for your own sake. Neither Bill Gates nor Mark Zuckerberg knew at first how big their companies were going to get. All they knew was that they were onto something. Maybe it's a bad idea to have really big ambitions initially, because the bigger your ambition, the longer it's going to take, and the further you project into the future, the more likely you'll get it wrong.

I think the way to use these big ideas is not to try to identify a precise point in the future and then ask yourself how to get from here to there, like the popular image of a visionary. You'll be better off if you operate like Columbus and just head in a general westerly direction. Don't try to construct the future like a building, because your current blueprint is almost certainly mistaken. Start with something you know works, and when you expand, expand westward.

The popular image of the visionary is someone with a clear view of the future, but empirically it may be better to have a blurry one.

## Notes

[1] It's also one of the most important things VCs fail to understand about startups. Most expect founders to walk in with a clear plan for the future, and judge them based on that. Few consciously realize that in the biggest successes there is the least correlation between the initial plan and what the startup eventually becomes.

[2] This sentence originally read "GMail is painfully slow." Thanks to Paul Buchheit for the correction.

[3] Roger Bannister is famous as the first person to run a mile in under 4 minutes. But his world record only lasted 46 days. Once he showed it could be done, lots of others followed. Ten years later Jim Ryun ran a 3:59 mile as a high school junior.

[4] If you want to be the next Apple, maybe you don't even want to start with consumer electronics. Maybe at first you make something hackers use. Or you make something popular but apparently unimportant, like a headset or router. All you need is a bridgehead.

**Thanks** to Sam Altman, Trevor Blackwell, Paul Buchheit, Patrick Collison, Aaron Iba, Jessica Livingston, Robert Morris, Harj Taggar and Garry Tan for reading drafts of this.

# Defining Property

March 2012

As a child I read a book of stories about a famous judge in eighteenth century Japan called Ooka Tadasuke. One of the cases he decided was brought by the owner of a food shop. A poor student who could afford only rice was eating his rice while enjoying the delicious cooking smells coming from the food shop. The owner wanted the student to pay for the smells he was enjoying.

The student was stealing his smells!

This story often comes to mind when I hear the RIAA and MPAA accusing people of stealing music and movies.

It sounds ridiculous to us to treat smells as property. But I can imagine scenarios in which one could charge for smells. Imagine we were living on a moon base where we had to buy air by the liter. I could imagine air suppliers adding scents at an extra charge.

The reason it seems ridiculous to us to treat smells as property is that it wouldn't work to. It would work on a moon base, though.

What counts as property depends on what works to treat as property. And that not only can change, but has changed. Humans may always (for some definition of human and always) have treated small items carried on one's person as property. But hunter gatherers didn't treat land, for example, as property in the way we do.

[\[1\]](#)

The reason so many people think of property as having a single unchanging definition is that its definition changes very slowly. [\[2\]](#) But we are in the midst of such a change now. The record labels and movie studios used to distribute what they made like air shipped through tubes on a moon base. But with the arrival of networks, it's as if we've moved to a planet with a breathable atmosphere. Data moves like smells now. And through a combination of wishful thinking and short-term greed, the labels and studios have put themselves in the position of the food shop owner, accusing us all of stealing their smells.

(The reason I say short-term greed is that the underlying problem with the labels and studios is that the people who run them are driven by bonuses rather than



equity. If they were driven by equity they'd be looking for ways to take advantage of technological change instead of fighting it. But building new things takes too long. Their bonuses depend on this year's revenues, and the best way to increase those is to extract more money from stuff they do already.)

So what does this mean? Should people not be able to charge for content? There's not a single yes or no answer to that question. People should be able to charge for content when it works to charge for content.

But by "works" I mean something more subtle than "when they can get away with it." I mean when people can charge for content without warping society in order to do it. After all, the companies selling smells on the moon base could continue to sell them on the Earth, if they lobbied successfully for laws requiring us all to continue to breathe through tubes down here too, even though we no longer needed to.

The crazy legal measures that the labels and studios have been taking have a lot of that flavor. Newspapers and magazines are just as screwed, but they are at least declining gracefully. The RIAA and MPAA would make us breathe through tubes if they could.

Ultimately it comes down to common sense. When you're abusing the legal system by trying to use mass lawsuits against randomly chosen people as a form of exemplary punishment, or lobbying for laws that would break the Internet if they passed, that's ipso facto evidence you're using a definition of property that doesn't work.

This is where it's helpful to have working democracies and multiple sovereign countries. If the world had a single, autocratic government, the labels and studios could buy laws making the definition of property be whatever they wanted. But fortunately there are still some countries that are not copyright colonies of the US, and even in the US, [politicians](#) still seem to be afraid of actual voters, in sufficient numbers. [3]

The people running the US may not like it when voters or other countries refuse to bend to their will, but ultimately it's in all our interest that there's not a single point of attack for people trying to warp the law to serve their own purposes. Private property is an extremely useful idea — arguably one of our greatest inventions. So far, each new definition of it has brought us increasing material wealth. [4] It seems reasonable to suppose the newest one will too. It would be a disaster if we all had to keep running an obsolete version just because a few powerful people were too lazy to upgrade.

## Notes

[1] If you want to learn more about hunter gatherers I strongly recommend Elizabeth Marshall Thomas's [\*The Harmless People\*](#) and [\*The Old Way\*](#).

[2] Change in the definition of property is driven mostly by technological progress, however, and since technological progress is accelerating, so presumably will the rate of change in the definition of property. Which means it's all the more important for societies to be able to respond gracefully to such changes, because they will come at an ever increasing rate.

[3] As far as I know, the term "copyright colony" was first used by [Myles Peterson](#).

[4] The state of technology isn't simply a function of the definition of property. They each constrain the other. But that being so, you can't mess with the definition of property without affecting (and probably harming) the state of technology. The history of the USSR offers a vivid illustration of that.

**Thanks** to Sam Altman and Geoff Ralston for reading drafts of this.

[Japanese Translation](#)

# How Y Combinator Started

March 2012

Y Combinator's 7th birthday was March 11. As usual we were so busy we didn't notice till a few days after. I don't think we've ever managed to remember our birthday on our birthday.

On March 11 2005, Jessica and I were walking home from dinner in Harvard Square. Jessica was working at an investment bank at the time, but she didn't like it much, so she had interviewed for a job as director of marketing at a Boston VC fund. The VC fund was doing what now seems a comically familiar thing for a VC fund to do: taking a long time to make up their mind. Meanwhile I had been telling Jessica all the things they should change about the VC business ♦ essentially the ideas now underlying Y Combinator: investors should be making more, smaller investments, they should be funding hackers instead of suits, they should be willing to fund younger founders, etc.

At the time I had been thinking about doing some angel investing. I had just given a talk to the undergraduate computer club at Harvard about [how to start a startup](#), and it hit me afterward that although I had always meant to do angel investing, 7 years had now passed since I got enough money to do it, and I still hadn't started. I had also been thinking about ways to work with Robert Morris and Trevor Blackwell again. A few hours before I had sent them an email trying to figure out what we could do together.

Between Harvard Square and my house the idea gelled. We'd start our own investment firm and Jessica could work for that instead. As we turned onto Walker Street we decided to do it. I agreed to put \$100k into the new fund and Jessica agreed to quit her job to work for it. Over the next couple days I recruited Robert and Trevor, who put in another \$50k each. So YC started with \$200k.

Jessica was so happy to be able to quit her job and start her own company that I took her [picture](#) when we got home.

The company wasn't called Y Combinator yet. At first we called it Cambridge Seed. But that name never saw the light of day, because by the time we announced it a few days later, we'd changed the name to Y Combinator. We realized early on that what we were doing could be national in scope and we didn't want a name that tied us to one place.

Initially we only had part of the idea. We were going to do seed funding with standardized terms. Before YC, seed funding was very haphazard. You'd get that first \$10k from your friend's rich uncle. The deal terms were often a disaster; often neither the investor nor the founders nor the lawyer knew what the documents

should look like. Facebook's early history as a Florida LLC shows how random things could be in those days. We were going to be something there had not been before: a standard source of seed funding.

We modelled YC on the seed funding we ourselves had taken when we started Viaweb. We started Viaweb with \$10k we got from our friend [Julian Weber](#), the husband of Idelle Weber, whose painting class I took as a grad student at Harvard. Julian knew about business, but you would not describe him as a suit. Among other things he'd been president of the *National Lampoon*. He was also a lawyer, and got all our paperwork set up properly. In return for \$10k, getting us set up as a company, teaching us what business was about, and remaining calm in times of crisis, Julian got 10% of Viaweb. I remember thinking once what a good deal Julian got. And then a second later I realized that without Julian, Viaweb would never have made it. So even though it was a good deal for him, it was a good deal for us too. That's why I knew there was room for something like Y Combinator.

Initially we didn't have what turned out to be the most important idea: funding startups synchronously, instead of asynchronously as it had always been done before. Or rather we had the idea, but we didn't realize its significance. We decided very early that the first thing we'd do would be to fund a bunch of startups over the coming summer. But we didn't realize initially that this would be the way we'd do all our investing. The reason we began by funding a bunch of startups at once was not that we thought it would be a better way to fund startups, but simply because we wanted to learn how to be angel investors, and a summer program for undergrads seemed the fastest way to do it. No one takes summer jobs that seriously. The opportunity cost for a bunch of undergrads to spend a summer working on startups was low enough that we wouldn't feel guilty encouraging them to do it.

We knew students would already be making plans for the summer, so we did what we're always telling startups to do: we launched fast. Here are the initial [announcement](#) and [description](#) of what was at the time called the Summer Founders Program.

We got lucky in that the length and structure of a summer program turns out to be perfect for what we do. The structure of the YC cycle is still almost identical to what it was that first summer.

We also got lucky in who the first batch of founders were. We never expected to make any money from that first batch. We thought of the money we were investing as a combination of an educational expense and a charitable donation. But the founders in the first batch turned out to be surprisingly good. And great people too. We're still friends with a lot of them today.

It's hard for people to realize now how inconsequential YC seemed at the time. I can't blame people who didn't take us seriously, because we ourselves didn't take that first summer program seriously in the very beginning. But as the summer progressed we were increasingly impressed by how well the startups were doing. Other people started to be impressed too. Jessica and I invented a term, "the Y Combinator effect," to describe the moment when the realization hit someone that YC was not totally lame. When people came to YC to speak at the dinners that first summer, they came in the spirit of someone coming to address a Boy Scout troop. By the time they left the building they were all saying some variant of "Wow, these companies might actually succeed."

Now YC is well enough known that people are no longer surprised when the companies we fund are legit, but it took a while for reputation to catch up with reality. That's one of the reasons we especially like funding ideas that might be dismissed as "toys" ♦ because YC itself was dismissed as one initially.

When we saw how well it worked to fund companies synchronously, we decided we'd keep doing that. We'd fund two batches of startups a year.

We funded the second batch in Silicon Valley. That was a last minute decision. In retrospect I think what pushed me over the edge was going to Foo Camp that fall. The density of startup people in the Bay Area was so much greater than in Boston, and the weather was so nice. I remembered that from living there in the 90s. Plus I didn't want someone else to copy us and describe it as the Y Combinator of Silicon Valley. I wanted YC to be the Y Combinator of Silicon Valley. So doing the winter batch in California seemed like one of those rare cases where the self-indulgent choice and the ambitious one were the same.

If we'd had enough time to do what we wanted, Y Combinator would have been in Berkeley. That was our favorite part of the Bay Area. But we didn't have time to get a building in Berkeley. We didn't have time to get our own building anywhere. The only way to get enough space in time was to convince Trevor to let us take over part of his (as it then seemed) giant building in Mountain View. Yet again we lucked out, because Mountain View turned out to be the ideal place to put something like YC. But even then we barely made it. The first dinner in California, we had to warn all the founders not to touch the walls, because the paint was still wet.

# Writing and Speaking

March 2012

I'm not a very good speaker. I say "um" a lot. Sometimes I have to pause when I lose my train of thought. I wish I were a better speaker. But I don't wish I were a better speaker like I wish I were a better writer. What I really want is to have good ideas, and that's a much bigger part of being a good writer than being a good speaker.

Having good ideas is most of writing well. If you know what you're talking about, you can say it in the plainest words and you'll be perceived as having a good style. With speaking it's the opposite: having good ideas is an alarmingly small component of being a good speaker.

I first noticed this at a conference several years ago. There was another speaker who was much better than me. He had all of us roaring with laughter. I seemed awkward and halting by comparison. Afterward I put my talk online like I usually do. As I was doing it I tried to imagine what a transcript of the other guy's talk would be like, and it was only then I realized he hadn't said very much.

Maybe this would have been obvious to someone who knew more about speaking, but it was a revelation to me how much less ideas mattered in speaking than writing. [\[1\]](#)

A few years later I heard a talk by someone who was not merely a better speaker than me, but a famous speaker. Boy was he good. So I decided I'd pay close attention to what he said, to learn how he did it. After about ten sentences I found myself thinking "I don't want to be a good speaker."

Being a really good speaker is not merely orthogonal to having good ideas, but in many ways pushes you in the opposite direction. For example, when I give a talk, I usually write it out beforehand. I know that's a mistake; I know delivering a prewritten talk makes it harder to engage with an audience. The way to get the attention of an audience is to give them *your* full attention, and when you're delivering a prewritten talk, your attention is always divided between the audience and the talk — even if you've memorized it. If you want to engage an audience, it's better to start with no more than an outline of what you want to say and ad lib the individual sentences. But if you do that, you might spend no more time thinking about each sentence than it takes to say it. [\[2\]](#) Occasionally the stimulation of

talking to a live audience makes you think of new things, but in general this is not going to generate ideas as well as writing does, where you can spend as long on each sentence as you want.

If you rehearse a prewritten speech enough, you can get asymptotically close to the sort of engagement you get when speaking ad lib. Actors do. But here again there's a tradeoff between smoothness and ideas. All the time you spend practicing a talk, you could instead spend making it better. Actors don't face that temptation, except in the rare cases where they've written the script, but any speaker does. Before I give a talk I can usually be found sitting in a corner somewhere with a copy printed out on paper, trying to rehearse it in my head. But I always end up spending most of the time rewriting it instead. Every talk I give ends up being given from a manuscript full of things crossed out and rewritten. Which of course makes me um even more, because I haven't had any time to practice the new bits.

[3]

Depending on your audience, there are even worse tradeoffs than these. Audiences like to be flattered; they like jokes; they like to be swept off their feet by a vigorous stream of words. As you decrease the intelligence of the audience, being a good speaker is increasingly a matter of being a good bullshitter. That's true in writing too of course, but the descent is steeper with talks. Any given person is dumber as a member of an audience than as a reader. Just as a speaker ad libbing can only spend as long thinking about each sentence as it takes to say it, a person hearing a talk can only spend as long thinking about each sentence as it takes to hear it. Plus people in an audience are always affected by the reactions of those around them, and the reactions that spread from person to person in an audience are disproportionately the more brutish sort, just as low notes travel through walls better than high ones. Every audience is an incipient mob, and a good speaker uses that. Part of the reason I laughed so much at the talk by the good speaker at that conference was that everyone else did. [4]

So are talks useless? They're certainly inferior to the written word as a source of ideas. But that's not all talks are good for. When I go to a talk, it's usually because I'm interested in the speaker. Listening to a talk is the closest most of us can get to having a conversation with someone like the president, who doesn't have time to meet individually with all the people who want to meet him.

Talks are also good at motivating me to do things. It's probably no coincidence that so many famous speakers are described as motivational speakers. That may be what public speaking is really for. It's probably what it was originally for. The emotional reactions you can elicit with a talk can be a powerful force. I wish I could say that this force was more often used for good than ill, but I'm not sure.

## Notes

[1] I'm not talking here about academic talks, which are a different type of thing. While the audience at an academic talk might appreciate a joke, they will (or at least should) make a conscious effort to see what new ideas you're presenting.

[2] That's the lower bound. In practice you can often do better, because talks are usually about things you've written or talked about before, and when you ad lib, you end up reproducing some of those sentences. Like early medieval architecture, impromptu talks are made of spolia. Which feels a bit dishonest, incidentally, because you have to deliver these sentences as if you'd just thought of them.

[3] Robert Morris points out that there is a way in which practicing talks makes them better: reading a talk out loud can expose awkward parts. I agree and in fact I read most things I write out loud at least once for that reason.

[4] For sufficiently small audiences, it may not be true that being part of an audience makes people dumber. The real decline seems to set in when the audience gets too big for the talk to feel like a conversation — maybe around 10 people.

**Thanks** to Sam Altman and Robert Morris for reading drafts of this.



# The Top of My Todo List

April 2012

A palliative care nurse called Bronnie Ware made a list of the biggest [regrets of the dying](#). Her list seems plausible. I could see myself — *can* see myself — making at least 4 of these 5 mistakes.

If you had to compress them into a single piece of advice, it might be: don't be a cog. The 5 regrets paint a portrait of post-industrial man, who shrinks himself into a shape that fits his circumstances, then turns dutifully till he stops.

The alarming thing is, the mistakes that produce these regrets are all errors of omission. You forget your dreams, ignore your family, suppress your feelings, neglect your friends, and forget to be happy. Errors of omission are a particularly dangerous type of mistake, because you make them by default.

I would like to avoid making these mistakes. But how do you avoid mistakes you make by default? Ideally you transform your life so it has other defaults. But it may not be possible to do that completely. As long as these mistakes happen by default, you probably have to be reminded not to make them. So I inverted the 5 regrets, yielding a list of 5 commands

Don't ignore your dreams; don't work too much; say what you think;  
cultivate friendships; be happy.

which I then put at the top of the file I use as a todo list.

[Japanese Translation](#)

# Black Swan Farming

September 2012

I've done several types of work over the years but I don't know another as counterintuitive as startup investing.

The two most important things to understand about startup investing, as a business, are (1) that effectively all the returns are concentrated in a few big winners, and (2) that the best ideas look initially like bad ideas.

The first rule I knew intellectually, but didn't really grasp till it happened to us. The total value of the companies we've funded is around 10 billion, give or take a few. But just two companies, Dropbox and Airbnb, account for about three quarters of it.

In startups, the big winners are big to a degree that violates our expectations about variation. I don't know whether these expectations are innate or learned, but whatever the cause, we are just not prepared for the 1000x variation in outcomes that one finds in startup investing.

That yields all sorts of strange consequences. For example, in purely financial terms, there is probably at most one company in each YC batch that will have a significant effect on our returns, and the rest are just a cost of doing business. [\[1\]](#) I haven't really assimilated that fact, partly because it's so counterintuitive, and partly because we're not doing this just for financial reasons; YC would be a pretty lonely place if we only had one company per batch. And yet it's true.

To succeed in a domain that violates your intuitions, you need to be able to turn them off the way a pilot does when flying through clouds. [\[2\]](#) You need to do what you know intellectually to be right, even though it feels wrong.

It's a constant battle for us. It's hard to make ourselves take enough risks. When you interview a startup and think "they seem likely to succeed," it's hard not to fund them. And yet, financially at least, there is only one kind of success: they're either going to be one of the really big winners or not, and if not it doesn't matter whether you fund them, because even if they succeed the effect on your returns will be insignificant. In the same day of interviews you might meet some smart 19 year olds who aren't even sure what they want to work on. Their chances of

succeeding seem small. But again, it's not their chances of succeeding that matter but their chances of succeeding really big. The probability that any group will succeed really big is microscopically small, but the probability that those 19 year olds will might be higher than that of the other, safer group.

The probability that a startup will make it big is not simply a constant fraction of the probability that they will succeed at all. If it were, you could fund everyone who seemed likely to succeed at all, and you'd get that fraction of big hits. Unfortunately picking winners is harder than that. You have to ignore the elephant in front of you, the likelihood they'll succeed, and focus instead on the separate and almost invisibly intangible question of whether they'll succeed really big.

## **Harder**

That's made harder by the fact that the best startup ideas seem at first like bad ideas. I've written about this before: if a good idea were obviously good, someone else would already have done it. So the most successful founders tend to work on ideas that few beside them realize are good. Which is not that far from a description of insanity, till you reach the point where you see results.

The first time Peter Thiel spoke at YC he drew a Venn diagram that illustrates the situation perfectly. He drew two intersecting circles, one labelled "seems like a bad idea" and the other "is a good idea." The intersection is the sweet spot for startups.

This concept is a simple one and yet seeing it as a Venn diagram is illuminating. It reminds you that there is an intersection—that there are good ideas that seem bad. It also reminds you that the vast majority of ideas that seem bad are bad.

The fact that the best ideas seem like bad ideas makes it even harder to recognize the big winners. It means the probability of a startup making it really big is not merely not a constant fraction of the probability that it will succeed, but that the startups with a high probability of the former will seem to have a disproportionately low probability of the latter.

History tends to get rewritten by big successes, so that in retrospect it seems obvious they were going to make it big. For that reason one of my most valuable memories is how lame Facebook sounded to me when I first heard about it. A site for college students to waste time? It seemed the perfect bad idea: a site (1) for a niche market (2) with no money (3) to do something that didn't matter.

One could have described Microsoft and Apple in exactly the same terms. [\[3\]](#)

## **Harder Still**

Wait, it gets worse. You not only have to solve this hard problem, but you have to do it with no indication of whether you're succeeding. When you pick a big winner, you won't know it for two years.

Meanwhile, the one thing you *can* measure is dangerously misleading. The one thing we can track precisely is how well the startups in each batch do at fundraising after Demo Day. But we know that's the wrong metric. There's no correlation between the percentage of startups that raise money and the metric that does matter financially, whether that batch of startups contains a big winner or not.

Except an inverse one. That's the scary thing: fundraising is not merely a useless metric, but positively misleading. We're in a business where we need to pick unpromising-looking outliers, and the huge scale of the successes means we can afford to spread our net very widely. The big winners could generate 10,000x returns. That means for each big winner we could pick a thousand companies that returned nothing and still end up 10x ahead.

If we ever got to the point where 100% of the startups we funded were able to raise money after Demo Day, it would almost certainly mean we were being too conservative. [4]

It takes a conscious effort not to do that too. After 15 cycles of preparing startups for investors and then watching how they do, I can now look at a group we're interviewing through Demo Day investors' eyes. But those are the wrong eyes to look through!

We can afford to take at least 10x as much risk as Demo Day investors. And since risk is usually proportionate to reward, if you can afford to take more risk you should. What would it mean to take 10x more risk than Demo Day investors? We'd have to be willing to fund 10x more startups than they would. Which means that even if we're generous to ourselves and assume that YC can on average triple a startup's expected value, we'd be taking the right amount of risk if only 30% of the startups were able to raise significant funding after Demo Day.

I don't know what fraction of them currently raise more after Demo Day. I deliberately avoid calculating that number, because if you start measuring something you start optimizing it, and I know it's the wrong thing to optimize. [5] But the percentage is certainly way over 30%. And frankly the thought of a 30% success rate at fundraising makes my stomach clench. A Demo Day where only 30% of the startups were fundable would be a shambles. Everyone would agree that YC had jumped the shark. We ourselves would feel that YC had jumped the shark. And yet we'd all be wrong.

For better or worse that's never going to be more than a thought experiment. We could never stand it. How about that for counterintuitive? I can lay out what I know to be the right thing to do, and still not do it. I can make up all sorts of plausible justifications. It would hurt YC's brand (at least among the innumerate) if we invested in huge numbers of risky startups that flamed out. It might dilute the value of the alumni network. Perhaps most convincingly, it would be demoralizing for us to be up to our chins in failure all the time. But I know the real reason we're

so conservative is that we just haven't assimilated the fact of 1000x variation in returns.

We'll probably never be able to bring ourselves to take risks proportionate to the returns in this business. The best we can hope for is that when we interview a group and find ourselves thinking "they seem like good founders, but what are investors going to think of this crazy idea?" we'll continue to be able to say "who cares what investors think?" That's what we thought about Airbnb, and if we want to fund more Airbnbs we have to stay good at thinking it.

## Notes

[1] I'm not saying that the big winners are all that matters, just that they're all that matters financially for investors. Since we're not doing YC mainly for financial reasons, the big winners aren't all that matters to us. We're delighted to have funded Reddit, for example. Even though we made comparatively little from it, Reddit has had a big effect on the world, and it introduced us to Steve Huffman and Alexis Ohanian, both of whom have become good friends.

Nor do we push founders to try to become one of the big winners if they don't want to. We didn't "swing for the fences" in our own startup (Viaweb, which was acquired for \$50 million), and it would feel pretty bogus to press founders to do something we didn't do. Our rule is that it's up to the founders. Some want to take over the world, and some just want that first few million. But we invest in so many companies that we don't have to sweat any one outcome. In fact, we don't have to sweat whether startups have exits at all. The biggest exits are the only ones that matter financially, and those are guaranteed in the sense that if a company becomes big enough, a market for its shares will inevitably arise. Since the remaining outcomes don't have a significant effect on returns, it's cool with us if the founders want to sell early for a small amount, or grow slowly and never sell (i.e. become a so-called lifestyle business), or even shut the company down. We're sometimes disappointed when a startup we had high hopes for doesn't do well, but this disappointment is mostly the ordinary variety that anyone feels when that happens.

[2] Without visual cues (e.g. the horizon) you can't distinguish between gravity and acceleration. Which means if you're flying through clouds you can't tell what the attitude of the aircraft is. You could feel like you're flying straight and level while in fact you're descending in a spiral. The solution is to ignore what your body is telling you and listen only to your instruments. But it turns out to be very hard to ignore what your body is telling you. Every pilot knows about this [problem](#) and yet it is still a leading cause of accidents.

[3] Not all big hits follow this pattern though. The reason Google seemed a bad

idea was that there were already lots of search engines and there didn't seem to be room for another.

[4] A startup's success at fundraising is a function of two things: what they're selling and how good they are at selling it. And while we can teach startups a lot about how to appeal to investors, even the most convincing pitch can't sell an idea that investors don't like. I was genuinely worried that Airbnb, for example, would not be able to raise money after Demo Day. I couldn't convince [Fred Wilson](#) to fund them. They might not have raised money at all but for the coincidence that Greg McAdoo, our contact at Sequoia, was one of a handful of VCs who understood the vacation rental business, having spent much of the previous two years investigating it.

[5] I calculated it once for the last batch before a consortium of investors started offering investment automatically to every startup we funded, summer 2010. At the time it was 94% (33 of 35 companies that tried to raise money succeeded, and one didn't try because they were already profitable). Presumably it's lower now because of that investment; in the old days it was raise after Demo Day or die.

**Thanks** to Sam Altman, Paul Buchheit, Patrick Collison, Jessica Livingston, Geoff Ralston, and Harj Taggar for reading drafts of this.

# Startup = Growth

September 2012

A startup is a company designed to grow fast. Being newly founded does not in itself make a company a startup. Nor is it necessary for a startup to work on technology, or take venture funding, or have some sort of "exit." The only essential thing is growth. Everything else we associate with startups follows from growth.

If you want to start one it's important to understand that. Startups are so hard that you can't be pointed off to the side and hope to succeed. You have to know that growth is what you're after. The good news is, if you get growth, everything else tends to fall into place. Which means you can use growth like a compass to make almost every decision you face.

## Redwoods

Let's start with a distinction that should be obvious but is often overlooked: not every newly founded company is a startup. Millions of companies are started every year in the US. Only a tiny fraction are startups. Most are service businesses — restaurants, barbershops, plumbers, and so on. These are not startups, except in a few unusual cases. A barbershop isn't designed to grow fast. Whereas a search engine, for example, is.

When I say startups are designed to grow fast, I mean it in two senses. Partly I mean designed in the sense of intended, because most startups fail. But I also mean startups are different by nature, in the same way a redwood seedling has a different destiny from a bean sprout.

That difference is why there's a distinct word, "startup," for companies designed to grow fast. If all companies were essentially similar, but some through luck or the efforts of their founders ended up growing very fast, we wouldn't need a separate word. We could just talk about super-successful companies and less successful ones. But in fact startups do have a different sort of DNA from other businesses. Google is not just a barbershop whose founders were unusually lucky and hard-working. Google was different from the beginning.

To grow rapidly, you need to make something you can sell to a big market. That's the difference between Google and a barbershop. A barbershop doesn't scale.

For a company to grow really big, it must (a) make something lots of people want, and (b) reach and serve all those people. Barbershops are doing fine in the (a) department. Almost everyone needs their hair cut. The problem for a barbershop, as for any retail establishment, is (b). A barbershop serves customers in person,

and few will travel far for a haircut. And even if they did, the barbershop couldn't accomodate them. [1]

Writing software is a great way to solve (b), but you can still end up constrained in (a). If you write software to teach Tibetan to Hungarian speakers, you'll be able to reach most of the people who want it, but there won't be many of them. If you make software to teach English to Chinese speakers, however, you're in startup territory.

Most businesses are tightly constrained in (a) or (b). The distinctive feature of successful startups is that they're not.

## **Ideas**

It might seem that it would always be better to start a startup than an ordinary business. If you're going to start a company, why not start the type with the most potential? The catch is that this is a (fairly) efficient market. If you write software to teach Tibetan to Hungarians, you won't have much competition. If you write software to teach English to Chinese speakers, you'll face ferocious competition, precisely because that's such a larger prize. [2]

The constraints that limit ordinary companies also protect them. That's the tradeoff. If you start a barbershop, you only have to compete with other local barbers. If you start a search engine you have to compete with the whole world.

The most important thing that the constraints on a normal business protect it from is not competition, however, but the difficulty of coming up with new ideas. If you open a bar in a particular neighborhood, as well as limiting your potential and protecting you from competitors, that geographic constraint also helps define your company. Bar + neighborhood is a sufficient idea for a small business. Similarly for companies constrained in (a). Your niche both protects and defines you.

Whereas if you want to start a startup, you're probably going to have to think of something fairly novel. A startup has to make something it can deliver to a large market, and ideas of that type are so valuable that all the obvious ones are already taken.

That space of ideas has been so thoroughly picked over that a startup generally has to work on something everyone else has overlooked. I was going to write that one has to make a conscious effort to find ideas everyone else has overlooked. But that's not how most startups get started. Usually successful startups happen because the founders are sufficiently different from other people that ideas few others can see seem obvious to them. Perhaps later they step back and notice they've found an idea in everyone else's blind spot, and from that point make a deliberate effort to stay there. [3] But at the moment when successful startups get started, much of the innovation is unconscious.

What's different about successful founders is that they can see different problems. It's a particularly good combination both to be good at technology and to face problems that can be solved by it, because technology changes so rapidly that formerly bad ideas often become good without anyone noticing. Steve Wozniak's problem was that he wanted his own computer. That was an unusual problem to have in 1975. But technological change was about to make it a much more common one. Because he not only wanted a computer but knew how to build



them, Wozniak was able to make himself one. And the problem he solved for himself became one that Apple solved for millions of people in the coming years. But by the time it was obvious to ordinary people that this was a big market, Apple was already established.

Google has similar origins. Larry Page and Sergey Brin wanted to search the web. But unlike most people they had the technical expertise both to notice that existing search engines were not as good as they could be, and to know how to improve them. Over the next few years their problem became everyone's problem, as the web grew to a size where you didn't have to be a picky search expert to notice the old algorithms weren't good enough. But as happened with Apple, by the time everyone else realized how important search was, Google was entrenched.

That's one connection between startup ideas and technology. Rapid change in one area uncovers big, soluble problems in other areas. Sometimes the changes are advances, and what they change is solubility. That was the kind of change that yielded Apple; advances in chip technology finally let Steve Wozniak design a computer he could afford. But in Google's case the most important change was the growth of the web. What changed there was not solubility but bigness.

The other connection between startups and technology is that startups create new ways of doing things, and new ways of doing things are, in the broader sense of the word, new technology. When a startup both begins with an idea exposed by technological change and makes a product consisting of technology in the narrower sense (what used to be called "high technology"), it's easy to conflate the two. But the two connections are distinct and in principle one could start a startup that was neither driven by technological change, nor whose product consisted of technology except in the broader sense. [\[4\]](#)

## Rate

How fast does a company have to grow to be considered a startup? There's no precise answer to that. "Startup" is a pole, not a threshold. Starting one is at first no more than a declaration of one's ambitions. You're committing not just to starting a company, but to starting a fast growing one, and you're thus committing to search for one of the rare ideas of that type. But at first you have no more than commitment. Starting a startup is like being an actor in that respect. "Actor" too is a pole rather than a threshold. At the beginning of his career, an actor is a waiter who goes to auditions. Getting work makes him a successful actor, but he doesn't only become an actor when he's successful.

So the real question is not what growth rate makes a company a startup, but what growth rate successful startups tend to have. For founders that's more than a theoretical question, because it's equivalent to asking if they're on the right path.

The growth of a successful startup usually has three phases:

1. There's an initial period of slow or no growth while the startup tries to figure out what it's doing.
2. As the startup figures out how to make something lots of people want and how to reach those people, there's a period of rapid growth.

3. Eventually a successful startup will grow into a big company. Growth will slow, partly due to internal limits and partly because the company is starting to bump up against the limits of the markets it serves. [5]

Together these three phases produce an S-curve. The phase whose growth defines the startup is the second one, the ascent. Its length and slope determine how big the company will be.

The slope is the company's growth rate. If there's one number every founder should always know, it's the company's growth rate. That's the measure of a startup. If you don't know that number, you don't even know if you're doing well or badly.

When I first meet founders and ask what their growth rate is, sometimes they tell me "we get about a hundred new customers a month." That's not a rate. What matters is not the absolute number of new customers, but the ratio of new customers to existing ones. If you're really getting a constant number of new customers every month, you're in trouble, because that means your growth rate is decreasing.

During Y Combinator we measure growth rate per week, partly because there is so little time before Demo Day, and partly because startups early on need frequent feedback from their users to tweak what they're doing. [6]

A good growth rate during YC is 5-7% a week. If you can hit 10% a week you're doing exceptionally well. If you can only manage 1%, it's a sign you haven't yet figured out what you're doing.

The best thing to measure the growth rate of is revenue. The next best, for startups that aren't charging initially, is active users. That's a reasonable proxy for revenue growth because whenever the startup does start trying to make money, their revenues will probably be a constant multiple of active users. [7]

## Compass

We usually advise startups to pick a growth rate they think they can hit, and then just try to hit it every week. The key word here is "just." If they decide to grow at 7% a week and they hit that number, they're successful for that week. There's nothing more they need to do. But if they don't hit it, they've failed in the only thing that mattered, and should be correspondingly alarmed.

Programmers will recognize what we're doing here. We're turning starting a startup into an optimization problem. And anyone who has tried optimizing code knows how wonderfully effective that sort of narrow focus can be. Optimizing code means taking an existing program and changing it to use less of something, usually time or memory. You don't have to think about what the program should do, just make it faster. For most programmers this is very satisfying work. The narrow focus makes it a sort of puzzle, and you're generally surprised how fast you can solve it.

Focusing on hitting a growth rate reduces the otherwise bewilderingly multifarious problem of starting a startup to a single problem. You can use that target growth rate to make all your decisions for you; anything that gets you the growth you need is ipso facto right. Should you spend two days at a conference? Should you hire another programmer? Should you focus more on marketing? Should you spend time courting some big customer? Should you add x feature? Whatever gets you your target growth rate. [8]

Judging yourself by weekly growth doesn't mean you can look no more than a week ahead. Once you experience the pain of missing your target one week (it was the only thing that mattered, and you failed at it), you become interested in anything that could spare you such pain in the future. So you'll be willing for example to hire another programmer, who won't contribute to this week's growth but perhaps in a month will have implemented some new feature that will get you more users. But only if (a) the distraction of hiring someone won't make you miss your numbers in the short term, and (b) you're sufficiently worried about whether you can keep hitting your numbers without hiring someone new.

It's not that you don't think about the future, just that you think about it no more than necessary.

In theory this sort of hill-climbing could get a startup into trouble. They could end up on a local maximum. But in practice that never happens. Having to hit a growth number every week forces founders to act, and acting versus not acting is the high bit of succeeding. Nine times out of ten, sitting around strategizing is just a form of procrastination. Whereas founders' intuitions about which hill to climb are usually better than they realize. Plus the maxima in the space of startup ideas are not spiky and isolated. Most fairly good ideas are adjacent to even better ones.

The fascinating thing about optimizing for growth is that it can actually discover startup ideas. You can use the need for growth as a form of evolutionary pressure. If you start out with some initial plan and modify it as necessary to keep hitting, say, 10% weekly growth, you may end up with a quite different company than you meant to start. But anything that grows consistently at 10% a week is almost certainly a better idea than you started with.

There's a parallel here to small businesses. Just as the constraint of being located in a particular neighborhood helps define a bar, the constraint of growing at a certain rate can help define a startup.

You'll generally do best to follow that constraint wherever it leads rather than being influenced by some initial vision, just as a scientist is better off following the truth wherever it leads rather than being influenced by what he wishes were the case. When Richard Feynman said that the imagination of nature was greater than the imagination of man, he meant that if you just keep following the truth you'll discover cooler things than you could ever have made up. For startups, growth is a constraint much like truth. Every successful startup is at least partly a product of the imagination of growth. [9]

**Value**

It's hard to find something that grows consistently at several percent a week, but if you do you may have found something surprisingly valuable. If we project forward we see why.

weekly
yearly
1%
1.7x
2%
2.8x
5%
12.6x
7%
33.7x
10%
142.0x

A company that grows at 1% a week will grow 1.7x a year, whereas a company that grows at 5% a week will grow 12.6x. A company making \$1000 a month (a typical number early in YC) and growing at 1% a week will 4 years later be making \$7900 a month, which is less than a good programmer makes in salary in Silicon Valley. A startup that grows at 5% a week will in 4 years be making \$25 million a month. [\[10\]](#)

Our ancestors must rarely have encountered cases of exponential growth, because our intuitions are no guide here. What happens to fast growing startups tends to surprise even the founders.

Small variations in growth rate produce qualitatively different outcomes. That's why there's a separate word for startups, and why startups do things that ordinary companies don't, like raising money and getting acquired. And, strangely enough, it's also why they fail so frequently.

Considering how valuable a successful startup can become, anyone familiar with the concept of expected value would be surprised if the failure rate weren't high. If a successful startup could make a founder \$100 million, then even if the chance of succeeding were only 1%, the expected value of starting one would be \$1 million. And the probability of a group of sufficiently smart and determined founders succeeding on that scale might be significantly over 1%. For the right people — e.g. the young Bill Gates — the probability might be 20% or even 50%. So it's not surprising that so many want to take a shot at it. In an efficient market, the number of failed startups should be proportionate to the size of the successes. And since the latter is huge the former should be too. [\[11\]](#)

What this means is that at any given time, the great majority of startups will be working on something that's never going to go anywhere, and yet glorifying their

doomed efforts with the grandiose title of "startup."

This doesn't bother me. It's the same with other high-beta vocations, like being an actor or a novelist. I've long since gotten used to it. But it seems to bother a lot of people, particularly those who've started ordinary businesses. Many are annoyed that these so-called startups get all the attention, when hardly any of them will amount to anything.

If they stepped back and looked at the whole picture they might be less indignant. The mistake they're making is that by basing their opinions on anecdotal evidence they're implicitly judging by the median rather than the average. If you judge by the median startup, the whole concept of a startup seems like a fraud. You have to invent a bubble to explain why founders want to start them or investors want to fund them. But it's a mistake to use the median in a domain with so much variation. If you look at the average outcome rather than the median, you can understand why investors like them, and why, if they aren't median people, it's a rational choice for founders to start them.

## **Deals**

Why do investors like startups so much? Why are they so hot to invest in photo-sharing apps, rather than solid money-making businesses? Not only for the obvious reason.

The test of any investment is the ratio of return to risk. Startups pass that test because although they're appallingly risky, the returns when they do succeed are so high. But that's not the only reason investors like startups. An ordinary slower-growing business might have just as good a ratio of return to risk, if both were lower. So why are VCs interested only in high-growth companies? The reason is that they get paid by getting their capital back, ideally after the startup IPOs, or failing that when it's acquired.

The other way to get returns from an investment is in the form of dividends. Why isn't there a parallel VC industry that invests in ordinary companies in return for a percentage of their profits? Because it's too easy for people who control a private company to funnel its revenues to themselves (e.g. by buying overpriced components from a supplier they control) while making it look like the company is making little profit. Anyone who invested in private companies in return for dividends would have to pay close attention to their books.

The reason VCs like to invest in startups is not simply the returns, but also because such investments are so easy to oversee. The founders can't enrich themselves without also enriching the investors. [\[12\]](#)

Why do founders want to take the VCs' money? Growth, again. The constraint between good ideas and growth operates in both directions. It's not merely that you need a scalable idea to grow. If you have such an idea and don't grow fast enough, competitors will. Growing too slowly is particularly dangerous in a

business with network effects, which the best startups usually have to some degree.

Almost every company needs some amount of funding to get started. But startups often raise money even when they are or could be profitable. It might seem foolish to sell stock in a profitable company for less than you think it will later be worth, but it's no more foolish than buying insurance. Fundamentally that's how the most successful startups view fundraising. They could grow the company on its own revenues, but the extra money and help supplied by VCs will let them grow even faster. Raising money lets you *choose* your growth rate.

Money to grow faster is always at the command of the most successful startups, because the VCs need them more than they need the VCs. A profitable startup could if it wanted just grow on its own revenues. Growing slower might be slightly dangerous, but chances are it wouldn't kill them. Whereas VCs need to invest in startups, and in particular the most successful startups, or they'll be out of business. Which means that any sufficiently promising startup will be offered money on terms they'd be crazy to refuse. And yet because of the scale of the successes in the startup business, VCs can still make money from such investments. You'd have to be crazy to believe your company was going to become as valuable as a high growth rate can make it, but some do.

Pretty much every successful startup will get acquisition offers too. Why? What is it about startups that makes other companies want to buy them? [\[13\]](#)

Fundamentally the same thing that makes everyone else want the stock of successful startups: a rapidly growing company is valuable. It's a good thing eBay bought Paypal, for example, because Paypal is now responsible for 43% of their sales and probably more of their growth.

But acquirers have an additional reason to want startups. A rapidly growing company is not merely valuable, but dangerous. If it keeps expanding, it might expand into the acquirer's own territory. Most product acquisitions have some component of fear. Even if an acquirer isn't threatened by the startup itself, they might be alarmed at the thought of what a competitor could do with it. And because startups are in this sense doubly valuable to acquirers, acquirers will often pay more than an ordinary investor would. [\[14\]](#)

## **Understand**

The combination of founders, investors, and acquirers forms a natural ecosystem. It works so well that those who don't understand it are driven to invent conspiracy theories to explain how neatly things sometimes turn out. Just as our ancestors did to explain the apparently too neat workings of the natural world. But there is no secret cabal making it all work.

If you start from the mistaken assumption that Instagram was worthless, you have to invent a secret boss to force Mark Zuckerberg to buy it. To anyone who knows

Mark Zuckerberg, that is the *reductio ad absurdum* of the initial assumption. The reason he bought Instagram was that it was valuable and dangerous, and what made it so was growth.

If you want to understand startups, understand growth. Growth drives everything in this world. Growth is why startups usually work on technology — because ideas for fast growing companies are so rare that the best way to find new ones is to discover those recently made viable by change, and technology is the best source of rapid change. Growth is why it's a rational choice economically for so many founders to try starting a startup: growth makes the successful companies so valuable that the expected value is high even though the risk is too. Growth is why VCs want to invest in startups: not just because the returns are high but also because generating returns from capital gains is easier to manage than generating returns from dividends. Growth explains why the most successful startups take VC money even if they don't need to: it lets them choose their growth rate. And growth explains why successful startups almost invariably get acquisition offers. To acquirers a fast-growing company is not merely valuable but dangerous too.

It's not just that if you want to succeed in some domain, you have to understand the forces driving it. Understanding growth is what starting a startup *consists* of. What you're really doing (and to the dismay of some observers, all you're really doing) when you start a startup is committing to solve a harder type of problem than ordinary businesses do. You're committing to search for one of the rare ideas that generates rapid growth. Because these ideas are so valuable, finding one is hard. The startup is the embodiment of your discoveries so far. Starting a startup is thus very much like deciding to be a research scientist: you're not committing to solve any specific problem; you don't know for sure which problems are soluble; but you're committing to try to discover something no one knew before. A startup founder is in effect an economic research scientist. Most don't discover anything that remarkable, but some discover relativity.

## Notes

[1] Strictly speaking it's not lots of customers you need but a big market, meaning a high product of number of customers times how much they'll pay. But it's dangerous to have too few customers even if they pay a lot, or the power that individual customers have over you could turn you into a *de facto* consulting firm. So whatever market you're in, you'll usually do best to err on the side of making the broadest type of product for it.

[2] One year at Startup School David Heinemeier Hansson encouraged programmers who wanted to start businesses to use a restaurant as a model. What he meant, I believe, is that it's fine to start software companies constrained in (a) in the same way a restaurant is constrained in (b). I agree. Most people should not try to start startups.

[3] That sort of stepping back is one of the things we focus on at Y Combinator. It's common for founders to have discovered something intuitively without understanding all its implications. That's probably true of the biggest discoveries in any field.

[4] I got it wrong in ["How to Make Wealth"](#) when I said that a startup was a small company that takes on a hard technical problem. That is the most common recipe but not the only one.

[5] In principle companies aren't limited by the size of the markets they serve, because they could just expand into new markets. But there seem to be limits on the ability of big companies to do that. Which means the slowdown that comes from bumping up against the limits of one's markets is ultimately just another way in which internal limits are expressed.

It may be that some of these limits could be overcome by changing the shape of the organization — specifically by sharding it.

[6] This is, obviously, only for startups that have already launched or can launch during YC. A startup building a new database will probably not do that. On the other hand, launching something small and then using growth rate as evolutionary pressure is such a valuable technique that any company that could start this way probably should.

[7] If the startup is taking the Facebook/Twitter route and building something they hope will be very popular but from which they don't yet have a definite plan to make money, the growth rate has to be higher, even though it's a proxy for revenue growth, because such companies need huge numbers of users to succeed at all.

Beware too of the edge case where something spreads rapidly but the churn is high as well, so that you have good net growth till you run through all the potential users, at which point it suddenly stops.

[8] Within YC when we say it's ipso facto right to do whatever gets you growth, it's implicit that this excludes trickery like buying users for more than their lifetime value, counting users as active when they're really not, bleeding out invites at a regularly increasing rate to manufacture a perfect growth curve, etc. Even if you were able to fool investors with such tricks, you'd ultimately be hurting yourself, because you're throwing off your own compass.

[9] Which is why it's such a dangerous mistake to believe that successful startups



are simply the embodiment of some brilliant initial idea. What you're looking for initially is not so much a great idea as an idea that could evolve into a great one. The danger is that promising ideas are not merely blurry versions of great ones. They're often different in kind, because the early adopters you evolve the idea upon have different needs from the rest of the market. For example, the idea that evolves into Facebook isn't merely a subset of Facebook; the idea that evolves into Facebook is a site for Harvard undergrads.

[10] What if a company grew at 1.7x a year for a really long time? Could it not grow just as big as any successful startup? In principle yes, of course. If our hypothetical company making \$1000 a month grew at 1% a week for 19 years, it would grow as big as a company growing at 5% a week for 4 years. But while such trajectories may be common in, say, real estate development, you don't see them much in the technology business. In technology, companies that grow slowly tend not to grow as big.

[11] Any expected value calculation varies from person to person depending on their utility function for money. I.e. the first million is worth more to most people than subsequent millions. How much more depends on the person. For founders who are younger or more ambitious the utility function is flatter. Which is probably part of the reason the founders of the most successful startups of all tend to be on the young side.

[12] More precisely, this is the case in the biggest winners, which is where all the returns come from. A startup founder could pull the same trick of enriching himself at the company's expense by selling them overpriced components. But it wouldn't be worth it for the founders of Google to do that. Only founders of failing startups would even be tempted, but those are writeoffs from the VCs' point of view anyway.

[13] Acquisitions fall into two categories: those where the acquirer wants the business, and those where the acquirer just wants the employees. The latter type is sometimes called an HR acquisition. Though nominally acquisitions and sometimes on a scale that has a significant effect on the expected value calculation for potential founders, HR acquisitions are viewed by acquirers as more akin to hiring bonuses.

[14] I once explained this to some founders who had recently arrived from Russia. They found it novel that if you threatened a company they'd pay a premium for you. "In Russia they just kill you," they said, and they were only partly joking. Economically, the fact that established companies can't simply eliminate new competitors may be one of the most valuable aspects of the rule of law. And so to the extent we see incumbents suppressing competitors via regulations or patent suits, we should worry, not because it's a departure from the rule of law per se but from what the rule of law is aiming at.

**Thanks** to Sam Altman, Marc Andreessen, Paul Buchheit, Patrick Collison, Jessica

Livingston, Geoff Ralston, and Harj Taggar for reading drafts of this.

[Arabic Translation](#)

[Estonian Translation](#)

[Portuguese Translation](#)

[Italian Translation](#)

# The Hardware Renaissance

October 2012

One advantage of Y Combinator's early, broad focus is that we see trends before most other people. And one of the most conspicuous trends in the last batch was the large number of hardware startups. Out of 84 companies, 7 were making hardware. On the whole they've done better than the companies that weren't.

They've faced resistance from investors of course. Investors have a deep-seated bias against hardware. But investors' opinions are a trailing indicator. The best founders are better at seeing the future than the best investors, because the best founders are making it.

There is no one single force driving this trend. Hardware [does well](#) on crowdfunding sites. The spread of [tablets](#) makes it possible to build new things [controlled by](#) and even [incorporating](#) them. [Electric motors](#) have improved. Wireless connectivity of various types can now be taken for granted. It's getting more straightforward to get things manufactured. Arduinos, 3D printing, laser cutters, and more accessible CNC milling are making hardware easier to prototype. Retailers are less of a bottleneck as customers increasingly buy online.

One question I can answer is why hardware is suddenly cool. It always was cool. Physical things are great. They just haven't been as great a way to start a [rapidly growing](#) business as software. But that rule may not be permanent. It's not even that old; it only dates from about 1990. Maybe the advantage of software will turn out to have been temporary. Hackers love to build hardware, and customers love to buy it. So if the ease of shipping hardware even approached the ease of shipping software, we'd see a lot more hardware startups.

It wouldn't be the first time something was a bad idea till it wasn't. And it wouldn't be the first time investors learned that lesson from founders.

So if you want to work on hardware, don't be deterred from doing it because you worry investors will discriminate against you. And in particular, don't be deterred from [applying](#) to Y Combinator with a hardware idea, because we're especially interested in hardware startups.

We know there's room for the [next Steve Jobs](#). But there's almost certainly also

room for the first <Your Name Here>.

**Thanks** to Sam Altman, Trevor Blackwell, David Cann, Sanjay Dastoor, Paul Gerhardt, Cameron Robertson, Harj Taggar, and Garry Tan for reading drafts of this.

[A Hardware Renaissance while !\[\]\(2bdfe261b986065ee0ac76460d6528c9\_img.jpg\)Software Eats the World!\[\]\(eebbd3dc1abeccf4c1e5751ec03fc559\_img.jpg\).](#)

# How to Get Startup Ideas

November 2012

The way to get startup ideas is not to try to think of startup ideas. It's to look for problems, preferably problems you have yourself.

The very best startup ideas tend to have three things in common: they're something the founders themselves want, that they themselves can build, and that few others realize are worth doing. Microsoft, Apple, Yahoo, Google, and Facebook all began this way.

## Problems

Why is it so important to work on a problem you have? Among other things, it ensures the problem really exists. It sounds obvious to say you should only work on problems that exist. And yet by far the most common mistake startups make is to solve problems no one has.

I made it myself. In 1995 I started a company to put art galleries online. But galleries didn't want to be online. It's not how the art business works. So why did I spend 6 months working on this stupid idea? Because I didn't pay attention to users. I invented a model of the world that didn't correspond to reality, and worked from that. I didn't notice my model was wrong until I tried to convince users to pay for what we'd built. Even then I took embarrassingly long to catch on. I was attached to my model of the world, and I'd spent a lot of time on the software. They had to want it!

Why do so many founders build things no one wants? Because they begin by trying to think of startup ideas. That m.o. is doubly dangerous: it doesn't merely yield few good ideas; it yields bad ideas that sound plausible enough to fool you into working on them.

At YC we call these "made-up" or "sitcom" startup ideas. Imagine one of the characters on a TV show was starting a startup. The writers would have to invent something for it to do. But coming up with good startup ideas is hard. It's not something you can do for the asking. So (unless they got amazingly lucky) the writers would come up with an idea that sounded plausible, but was actually bad.

For example, a social network for pet owners. It doesn't sound obviously mistaken. Millions of people have pets. Often they care a lot about their pets and spend a lot of money on them. Surely many of these people would like a site where they could talk to other pet owners. Not all of them perhaps, but if just 2 or 3 percent were regular visitors, you could have millions of users. You could serve them targeted

offers, and maybe charge for premium features. [1]

The danger of an idea like this is that when you run it by your friends with pets, they don't say "I would *never* use this." They say "Yeah, maybe I could see using something like that." Even when the startup launches, it will sound plausible to a lot of people. They don't want to use it themselves, at least not right now, but they could imagine other people wanting it. Sum that reaction across the entire population, and you have zero users. [2]

## Well

When a startup launches, there have to be at least some users who really need what they're making — not just people who could see themselves using it one day, but who want it urgently. Usually this initial group of users is small, for the simple reason that if there were something that large numbers of people urgently needed and that could be built with the amount of effort a startup usually puts into a version one, it would probably already exist. Which means you have to compromise on one dimension: you can either build something a large number of people want a small amount, or something a small number of people want a large amount. Choose the latter. Not all ideas of that type are good startup ideas, but nearly all good startup ideas are of that type.

Imagine a graph whose x axis represents all the people who might want what you're making and whose y axis represents how much they want it. If you invert the scale on the y axis, you can envision companies as holes. Google is an immense crater: hundreds of millions of people use it, and they need it a lot. A startup just starting out can't expect to excavate that much volume. So you have two choices about the shape of hole you start with. You can either dig a hole that's broad but shallow, or one that's narrow and deep, like a well.

Made-up startup ideas are usually of the first type. Lots of people are mildly interested in a social network for pet owners.

Nearly all good startup ideas are of the second type. Microsoft was a well when they made Altair Basic. There were only a couple thousand Altair owners, but without this software they were programming in machine language. Thirty years later Facebook had the same shape. Their first site was exclusively for Harvard students, of which there are only a few thousand, but those few thousand users wanted it a lot.

When you have an idea for a startup, ask yourself: who wants this right now? Who wants this so much that they'll use it even when it's a crappy version one made by a two-person startup they've never heard of? If you can't answer that, the idea is probably bad. [3]

You don't need the narrowness of the well per se. It's depth you need; you get narrowness as a byproduct of optimizing for depth (and speed). But you almost always do get it. In practice the link between depth and narrowness is so strong that it's a good sign when you know that an idea will appeal strongly to a specific group or type of user.

But while demand shaped like a well is almost a necessary condition for a good startup idea, it's not a sufficient one. If Mark Zuckerberg had built something that could only ever have appealed to Harvard students, it would not have been a good

startup idea. Facebook was a good idea because it started with a small market there was a fast path out of. Colleges are similar enough that if you build a facebook that works at Harvard, it will work at any college. So you spread rapidly through all the colleges. Once you have all the college students, you get everyone else simply by letting them in.

Similarly for Microsoft: Basic for the Altair; Basic for other machines; other languages besides Basic; operating systems; applications; IPO.

## Self

How do you tell whether there's a path out of an idea? How do you tell whether something is the germ of a giant company, or just a niche product? Often you can't. The founders of Airbnb didn't realize at first how big a market they were tapping. Initially they had a much narrower idea. They were going to let hosts rent out space on their floors during conventions. They didn't foresee the expansion of this idea; it forced itself upon them gradually. All they knew at first is that they were onto something. That's probably as much as Bill Gates or Mark Zuckerberg knew at first.

Occasionally it's obvious from the beginning when there's a path out of the initial niche. And sometimes I can see a path that's not immediately obvious; that's one of our specialties at YC. But there are limits to how well this can be done, no matter how much experience you have. The most important thing to understand about paths out of the initial idea is the meta-fact that these are hard to see.

So if you can't predict whether there's a path out of an idea, how do you choose between ideas? The truth is disappointing but interesting: if you're the right sort of person, you have the right sort of hunches. If you're at the leading edge of a field that's changing fast, when you have a hunch that something is worth doing, you're more likely to be right.

In *Zen and the Art of Motorcycle Maintenance*, Robert Pirsig says:

You want to know how to paint a perfect painting? It's easy. Make yourself perfect and then just paint naturally.

I've wondered about that passage since I read it in high school. I'm not sure how useful his advice is for painting specifically, but it fits this situation well. Empirically, the way to have good startup ideas is to become the sort of person who has them.

Being at the leading edge of a field doesn't mean you have to be one of the people pushing it forward. You can also be at the leading edge as a user. It was not so much because he was a programmer that Facebook seemed a good idea to Mark Zuckerberg as because he used computers so much. If you'd asked most 40 year olds in 2004 whether they'd like to publish their lives semi-publicly on the Internet, they'd have been horrified at the idea. But Mark already lived online; to him it seemed natural.

Paul Buchheit says that people at the leading edge of a rapidly changing field "live

in the future." Combine that with Pirsig and you get:

Live in the future, then build what's missing.

That describes the way many if not most of the biggest startups got started. Neither Apple nor Yahoo nor Google nor Facebook were even supposed to be companies at first. They grew out of things their founders built because there seemed a gap in the world.

If you look at the way successful founders have had their ideas, it's generally the result of some external stimulus hitting a prepared mind. Bill Gates and Paul Allen hear about the Altair and think "I bet we could write a Basic interpreter for it." Drew Houston realizes he's forgotten his USB stick and thinks "I really need to make my files live online." Lots of people heard about the Altair. Lots forgot USB sticks. The reason those stimuli caused those founders to start companies was that their experiences had prepared them to notice the opportunities they represented.

The verb you want to be using with respect to startup ideas is not "think up" but "notice." At YC we call ideas that grow naturally out of the founders' own experiences "organic" startup ideas. The most successful startups almost all begin this way.

That may not have been what you wanted to hear. You may have expected recipes for coming up with startup ideas, and instead I'm telling you that the key is to have a mind that's prepared in the right way. But disappointing though it may be, this is the truth. And it is a recipe of a sort, just one that in the worst case takes a year rather than a weekend.

If you're not at the leading edge of some rapidly changing field, you can get to one. For example, anyone reasonably smart can probably get to an edge of programming (e.g. building mobile apps) in a year. Since a successful startup will consume at least 3-5 years of your life, a year's preparation would be a reasonable investment. Especially if you're also looking for a cofounder. [\[4\]](#)

You don't have to learn programming to be at the leading edge of a domain that's changing fast. Other domains change fast. But while learning to hack is not necessary, it is for the foreseeable future sufficient. As Marc Andreessen put it, software is eating the world, and this trend has decades left to run.

Knowing how to hack also means that when you have ideas, you'll be able to implement them. That's not absolutely necessary (Jeff Bezos couldn't) but it's an advantage. It's a big advantage, when you're considering an idea like putting a college facebook online, if instead of merely thinking "That's an interesting idea," you can think instead "That's an interesting idea. I'll try building an initial version tonight." It's even better when you're both a programmer and the target user, because then the cycle of generating new versions and testing them on users can happen inside one head.



## Noticing

Once you're living in the future in some respect, the way to notice startup ideas is to look for things that seem to be missing. If you're really at the leading edge of a rapidly changing field, there will be things that are obviously missing. What won't be obvious is that they're startup ideas. So if you want to find startup ideas, don't merely turn on the filter "What's missing?" Also turn off every other filter, particularly "Could this be a big company?" There's plenty of time to apply that test later. But if you're thinking about that initially, it may not only filter out lots of good ideas, but also cause you to focus on bad ones.

Most things that are missing will take some time to see. You almost have to trick yourself into seeing the ideas around you.

But you *know* the ideas are out there. This is not one of those problems where there might not be an answer. It's impossibly unlikely that this is the exact moment when technological progress stops. You can be sure people are going to build things in the next few years that will make you think "What did I do before x?"

And when these problems get solved, they will probably seem flamingly obvious in retrospect. What you need to do is turn off the filters that usually prevent you from seeing them. The most powerful is simply taking the current state of the world for granted. Even the most radically open-minded of us mostly do that. You couldn't get from your bed to the front door if you stopped to question everything.

But if you're looking for startup ideas you can sacrifice some of the efficiency of taking the status quo for granted and start to question things. Why is your inbox overflowing? Because you get a lot of email, or because it's hard to get email out of your inbox? Why do you get so much email? What problems are people trying to solve by sending you email? Are there better ways to solve them? And why is it hard to get emails out of your inbox? Why do you keep emails around after you've read them? Is an inbox the optimal tool for that?

Pay particular attention to things that chafe you. The advantage of taking the status quo for granted is not just that it makes life (locally) more efficient, but also that it makes life more tolerable. If you knew about all the things we'll get in the next 50 years but don't have yet, you'd find present day life pretty constraining, just as someone from the present would if they were sent back 50 years in a time machine. When something annoys you, it could be because you're living in the future.

When you find the right sort of problem, you should probably be able to describe it as *obvious*, at least to you. When we started Viaweb, all the online stores were built by hand, by web designers making individual HTML pages. It was obvious to us as programmers that these sites would have to be generated by software. [5]

Which means, strangely enough, that coming up with startup ideas is a question of

seeing the obvious. That suggests how weird this process is: you're trying to see things that are obvious, and yet that you hadn't seen.

Since what you need to do here is loosen up your own mind, it may be best not to make too much of a direct frontal attack on the problem — i.e. to sit down and try to think of ideas. The best plan may be just to keep a background process running, looking for things that seem to be missing. Work on hard problems, driven mainly by curiosity, but have a second self watching over your shoulder, taking note of gaps and anomalies. [6]

Give yourself some time. You have a lot of control over the rate at which you turn yours into a prepared mind, but you have less control over the stimuli that spark ideas when they hit it. If Bill Gates and Paul Allen had constrained themselves to come up with a startup idea in one month, what if they'd chosen a month before the Altair appeared? They probably would have worked on a less promising idea. Drew Houston did work on a less promising idea before Dropbox: an SAT prep startup. But Dropbox was a much better idea, both in the absolute sense and also as a match for his skills. [7]

A good way to trick yourself into noticing ideas is to work on projects that seem like they'd be cool. If you do that, you'll naturally tend to build things that are missing. It wouldn't seem as interesting to build something that already existed.

Just as trying to think up startup ideas tends to produce bad ones, working on things that could be dismissed as "toys" often produces good ones. When something is described as a toy, that means it has everything an idea needs except being important. It's cool; users love it; it just doesn't matter. But if you're living in the future and you build something cool that users love, it may matter more than outsiders think. Microcomputers seemed like toys when Apple and Microsoft started working on them. I'm old enough to remember that era; the usual term for people with their own microcomputers was "hobbyists." BackRub seemed like an inconsequential science project. The Facebook was just a way for undergrads to stalk one another.

At YC we're excited when we meet startups working on things that we could imagine know-it-alls on forums dismissing as toys. To us that's positive evidence an idea is good.

If you can afford to take a long view (and arguably you can't afford not to), you can turn "Live in the future and build what's missing" into something even better:

Live in the future and build what seems interesting.

## **School**

That's what I'd advise college students to do, rather than trying to learn about

"entrepreneurship." "Entrepreneurship" is something you learn best by doing it. The examples of the most successful founders make that clear. What you should be spending your time on in college is ratcheting yourself into the future. College is an incomparable opportunity to do that. What a waste to sacrifice an opportunity to solve the hard part of starting a startup — becoming the sort of person who can have organic startup ideas — by spending time learning about the easy part. Especially since you won't even really learn about it, any more than you'd learn about sex in a class. All you'll learn is the words for things.

The clash of domains is a particularly fruitful source of ideas. If you know a lot about programming and you start learning about some other field, you'll probably see problems that software could solve. In fact, you're doubly likely to find good problems in another domain: (a) the inhabitants of that domain are not as likely as software people to have already solved their problems with software, and (b) since you come into the new domain totally ignorant, you don't even know what the status quo is to take it for granted.

So if you're a CS major and you want to start a startup, instead of taking a class on entrepreneurship you're better off taking a class on, say, genetics. Or better still, go work for a biotech company. CS majors normally get summer jobs at computer hardware or software companies. But if you want to find startup ideas, you might do better to get a summer job in some unrelated field. [8]

Or don't take any extra classes, and just build things. It's no coincidence that Microsoft and Facebook both got started in January. At Harvard that is (or was) Reading Period, when students have no classes to attend because they're supposed to be studying for finals. [9]

But don't feel like you have to build things that will become startups. That's premature optimization. Just build things. Preferably with other students. It's not just the classes that make a university such a good place to crank oneself into the future. You're also surrounded by other people trying to do the same thing. If you work together with them on projects, you'll end up producing not just organic ideas, but organic ideas with organic founding teams — and that, empirically, is the best combination.

Beware of research. If an undergrad writes something all his friends start using, it's quite likely to represent a good startup idea. Whereas a PhD dissertation is extremely unlikely to. For some reason, the more a project has to count as research, the less likely it is to be something that could be turned into a startup. [10] I think the reason is that the subset of ideas that count as research is so narrow that it's unlikely that a project that satisfied that constraint would also satisfy the orthogonal constraint of solving users' problems. Whereas when students (or professors) build something as a side-project, they automatically gravitate toward solving users' problems — perhaps even with an additional energy that comes from being freed from the constraints of research.

## Competition

Because a good idea should seem obvious, when you have one you'll tend to feel that you're late. Don't let that deter you. Worrying that you're late is one of the signs of a good idea. Ten minutes of searching the web will usually settle the question. Even if you find someone else working on the same thing, you're probably not too late. It's exceptionally rare for startups to be killed by competitors — so rare that you can almost discount the possibility. So unless you discover a competitor with the sort of lock-in that would prevent users from choosing you, don't discard the idea.

If you're uncertain, ask users. The question of whether you're too late is subsumed by the question of whether anyone urgently needs what you plan to make. If you have something that no competitor does and that some subset of users urgently need, you have a beachhead. [[11](#)]

The question then is whether that beachhead is big enough. Or more importantly, who's in it: if the beachhead consists of people doing something lots more people will be doing in the future, then it's probably big enough no matter how small it is. For example, if you're building something differentiated from competitors by the fact that it works on phones, but it only works on the newest phones, that's probably a big enough beachhead.

Err on the side of doing things where you'll face competitors. Inexperienced founders usually give competitors more credit than they deserve. Whether you succeed depends far more on you than on your competitors. So better a good idea with competitors than a bad one without.

You don't need to worry about entering a "crowded market" so long as you have a thesis about what everyone else in it is overlooking. In fact that's a very promising starting point. Google was that type of idea. Your thesis has to be more precise than "we're going to make an x that doesn't suck" though. You have to be able to phrase it in terms of something the incumbents are overlooking. Best of all is when you can say that they didn't have the courage of their convictions, and that your plan is what they'd have done if they'd followed through on their own insights. Google was that type of idea too. The search engines that preceded them shied away from the most radical implications of what they were doing — particularly that the better a job they did, the faster users would leave.

A crowded market is actually a good sign, because it means both that there's demand and that none of the existing solutions are good enough. A startup can't hope to enter a market that's obviously big and yet in which they have no competitors. So any startup that succeeds is either going to be entering a market with existing competitors, but armed with some secret weapon that will get them all the users (like Google), or entering a market that looks small but which will turn out to be big (like Microsoft). [[12](#)]

## **Filters**

There are two more filters you'll need to turn off if you want to notice startup ideas: the unsexy filter and the schlep filter.

Most programmers wish they could start a startup by just writing some brilliant code, pushing it to a server, and having users pay them lots of money. They'd prefer not to deal with tedious problems or get involved in messy ways with the real world. Which is a reasonable preference, because such things slow you down. But this preference is so widespread that the space of convenient startup ideas has been stripped pretty clean. If you let your mind wander a few blocks down the street to the messy, tedious ideas, you'll find valuable ones just sitting there waiting to be implemented.

The schlep filter is so dangerous that I wrote a separate essay about the condition it induces, which I called [schlep blindness](#). I gave Stripe as an example of a startup that benefited from turning off this filter, and a pretty striking example it is. Thousands of programmers were in a position to see this idea; thousands of programmers knew how painful it was to process payments before Stripe. But when they looked for startup ideas they didn't see this one, because unconsciously they shrank from having to deal with payments. And dealing with payments is a schlep for Stripe, but not an intolerable one. In fact they might have had net less pain; because the fear of dealing with payments kept most people away from this idea, Stripe has had comparatively smooth sailing in other areas that are sometimes painful, like user acquisition. They didn't have to try very hard to make themselves heard by users, because users were desperately waiting for what they were building.

The unsexy filter is similar to the schlep filter, except it keeps you from working on problems you despise rather than ones you fear. We overcame this one to work on Viaweb. There were interesting things about the architecture of our software, but we weren't interested in ecommerce per se. We could see the problem was one that needed to be solved though.

Turning off the schlep filter is more important than turning off the unsexy filter, because the schlep filter is more likely to be an illusion. And even to the degree it isn't, it's a worse form of self-indulgence. Starting a successful startup is going to be fairly laborious no matter what. Even if the product doesn't entail a lot of schleps, you'll still have plenty dealing with investors, hiring and firing people, and so on. So if there's some idea you think would be cool but you're kept away from by fear of the schleps involved, don't worry: any sufficiently good idea will have as many.

The unsexy filter, while still a source of error, is not as entirely useless as the schlep filter. If you're at the leading edge of a field that's changing rapidly, your ideas about what's sexy will be somewhat correlated with what's valuable in practice. Particularly as you get older and more experienced. Plus if you find an idea sexy, you'll work on it more enthusiastically. [\[13\]](#)

## Recipes

While the best way to discover startup ideas is to become the sort of person who has them and then build whatever interests you, sometimes you don't have that luxury. Sometimes you need an idea now. For example, if you're working on a startup and your initial idea turns out to be bad.

For the rest of this essay I'll talk about tricks for coming up with startup ideas on demand. Although empirically you're better off using the organic strategy, you could succeed this way. You just have to be more disciplined. When you use the organic method, you don't even notice an idea unless it's evidence that something is truly missing. But when you make a conscious effort to think of startup ideas, you have to replace this natural constraint with self-discipline. You'll see a lot more ideas, most of them bad, so you need to be able to filter them.

One of the biggest dangers of not using the organic method is the example of the organic method. Organic ideas feel like inspirations. There are a lot of stories about successful startups that began when the founders had what seemed a crazy idea but "just knew" it was promising. When you feel that about an idea you've had while trying to come up with startup ideas, you're probably mistaken.

When searching for ideas, look in areas where you have some expertise. If you're a database expert, don't build a chat app for teenagers (unless you're also a teenager). Maybe it's a good idea, but you can't trust your judgment about that, so ignore it. There have to be other ideas that involve databases, and whose quality you can judge. Do you find it hard to come up with good ideas involving databases? That's because your expertise raises your standards. Your ideas about chat apps are just as bad, but you're giving yourself a Dunning-Kruger pass in that domain.

The place to start looking for ideas is things you need. There *must* be things you need. [\[14\]](#)

One good trick is to ask yourself whether in your previous job you ever found yourself saying "Why doesn't someone make x? If someone made x we'd buy it in a second." If you can think of any x people said that about, you probably have an idea. You know there's demand, and people don't say that about things that are impossible to build.

More generally, try asking yourself whether there's something unusual about you that makes your needs different from most other people's. You're probably not the only one. It's especially good if you're different in a way people will increasingly be.

If you're changing ideas, one unusual thing about you is the idea you'd previously been working on. Did you discover any needs while working on it? Several well-known startups began this way. Hotmail began as something its founders wrote to talk about their previous startup idea while they were working at their day jobs.

A particularly promising way to be unusual is to be young. Some of the most valuable new ideas take root first among people in their teens and early twenties. And while young founders are at a disadvantage in some respects, they're the only ones who really understand their peers. It would have been very hard for someone who wasn't a college student to start Facebook. So if you're a young founder (under 23 say), are there things you and your friends would like to do that current technology won't let you?

The next best thing to an unmet need of your own is an unmet need of someone else. Try talking to everyone you can about the gaps they find in the world. What's missing? What would they like to do that they can't? What's tedious or annoying, particularly in their work? Let the conversation get general; don't be trying too hard to find startup ideas. You're just looking for something to spark a thought. Maybe you'll notice a problem they didn't consciously realize they had, because you know how to solve it.

When you find an unmet need that isn't your own, it may be somewhat blurry at first. The person who needs something may not know exactly what they need. In that case I often recommend that founders act like consultants — that they do what they'd do if they'd been retained to solve the problems of this one user. People's problems are similar enough that nearly all the code you write this way will be reusable, and whatever isn't will be a small price to start out certain that you've reached the bottom of the well. [\[16\]](#)

One way to ensure you do a good job solving other people's problems is to make them your own. When Rajat Suri of E la Carte decided to write software for restaurants, he got a job as a waiter to learn how restaurants worked. That may seem like taking things to extremes, but startups are extreme. We love it when founders do such things.

In fact, one strategy I recommend to people who need a new idea is not merely to turn off their schlep and unsexy filters, but to seek out ideas that are unsexy or involve schleps. Don't try to start Twitter. Those ideas are so rare that you can't find them by looking for them. Make something unsexy that people will pay you for.

A good trick for bypassing the schlep and to some extent the unsexy filter is to ask what you wish someone else would build, so that you could use it. What would you pay for right now?

Since startups often garbage-collect broken companies and industries, it can be a good trick to look for those that are dying, or deserve to, and try to imagine what kind of company would profit from their demise. For example, journalism is in free fall at the moment. But there may still be money to be made from something like journalism. What sort of company might cause people in the future to say "this replaced journalism" on some axis?

But imagine asking that in the future, not now. When one company or industry replaces another, it usually comes in from the side. So don't look for a replacement for x; look for something that people will later say turned out to be a replacement for x. And be imaginative about the axis along which the replacement occurs. Traditional journalism, for example, is a way for readers to get information and to kill time, a way for writers to make money and to get attention, and a vehicle for several different types of advertising. It could be replaced on any of these axes (it has already started to be on most).

When startups consume incumbents, they usually start by serving some small but important market that the big players ignore. It's particularly good if there's an admixture of disdain in the big players' attitude, because that often misleads them. For example, after Steve Wozniak built the computer that became the Apple I, he felt obliged to give his then-employer Hewlett-Packard the option to produce it. Fortunately for him, they turned it down, and one of the reasons they did was that it used a TV for a monitor, which seemed intolerably ~~d~~class~~e~~ to a high-end hardware company like HP was at the time. [17]

Are there groups of [scruffy](#) but sophisticated users like the early microcomputer "hobbyists" that are currently being ignored by the big players? A startup with its sights set on bigger things can often capture a small market easily by expending an effort that wouldn't be justified by that market alone.

Similarly, since the most successful startups generally ride some wave bigger than themselves, it could be a good trick to look for waves and ask how one could benefit from them. The prices of gene sequencing and 3D printing are both experiencing Moore's Law-like declines. What new things will we be able to do in the new world we'll have in a few years? What are we unconsciously ruling out as impossible that will soon be possible?

## Organic

But talking about looking explicitly for waves makes it clear that such recipes are plan B for getting startup ideas. Looking for waves is essentially a way to simulate the organic method. If you're at the leading edge of some rapidly changing field, you don't have to look for waves; you are the wave.

Finding startup ideas is a subtle business, and that's why most people who try fail so miserably. It doesn't work well simply to try to think of startup ideas. If you do that, you get bad ones that sound dangerously plausible. The best approach is more indirect: if you have the right sort of background, good startup ideas will seem obvious to you. But even then, not immediately. It takes time to come across situations where you notice something missing. And often these gaps won't seem to be ideas for companies, just things that would be interesting to build. Which is why it's good to have the time and the inclination to build things just because they're interesting.

Live in the future and build what seems interesting. Strange as it sounds, that's



the real recipe.

## Notes

[1] This form of bad idea has been around as long as the web. It was common in the 1990s, except then people who had it used to say they were going to create a portal for x instead of a social network for x. Structurally the idea is stone soup: you post a sign saying "this is the place for people interested in x," and all those people show up and you make money from them. What lures founders into this sort of idea are statistics about the millions of people who might be interested in each type of x. What they forget is that any given person might have 20 affinities by this standard, and no one is going to visit 20 different communities regularly.

[2] I'm not saying, incidentally, that I know for sure a social network for pet owners is a bad idea. I know it's a bad idea the way I know randomly generated DNA would not produce a viable organism. The set of plausible sounding startup ideas is many times larger than the set of good ones, and many of the good ones don't even sound that plausible. So if all you know about a startup idea is that it sounds plausible, you have to assume it's bad.

[3] More precisely, the users' need has to give them sufficient activation energy to start using whatever you make, which can vary a lot. For example, the activation energy for enterprise software sold through traditional channels is very high, so you'd have to be a *lot* better to get users to switch. Whereas the activation energy required to switch to a new search engine is low. Which in turn is why search engines are so much better than enterprise software.

[4] This gets harder as you get older. While the space of ideas doesn't have dangerous local maxima, the space of careers does. There are fairly high walls between most of the paths people take through life, and the older you get, the higher the walls become.

[5] It was also obvious to us that the web was going to be a big deal. Few non-programmers grasped that in 1995, but the programmers had seen what GUIs had done for desktop computers.

[6] Maybe it would work to have this second self keep a journal, and each night to make a brief entry listing the gaps and anomalies you'd noticed that day. Not startup ideas, just the raw gaps and anomalies.

[7] Sam Altman points out that taking time to come up with an idea is not merely a better strategy in an absolute sense, but also like an undervalued stock in that

so few founders do it.

There's comparatively little competition for the best ideas, because few founders are willing to put in the time required to notice them. Whereas there is a great deal of competition for mediocre ideas, because when people make up startup ideas, they tend to make up the same ones.

[8] For the computer hardware and software companies, summer jobs are the first phase of the recruiting funnel. But if you're good you can skip the first phase. If you're good you'll have no trouble getting hired by these companies when you graduate, regardless of how you spent your summers.

[9] The empirical evidence suggests that if colleges want to help their students start startups, the best thing they can do is leave them alone in the right way.

[10] I'm speaking here of IT startups; in biotech things are different.

[11] This is an instance of a more general rule: focus on users, not competitors. The most important information about competitors is what you learn via users anyway.

[12] In practice most successful startups have elements of both. And you can describe each strategy in terms of the other by adjusting the boundaries of what you call the market. But it's useful to consider these two ideas separately.

[13] I almost hesitate to raise that point though. Startups are businesses; the point of a business is to make money; and with that additional constraint, you can't expect you'll be able to spend all your time working on what interests you most.

[14] The need has to be a strong one. You can retroactively describe any made-up idea as something you need. But do you really need that recipe site or local event aggregator as much as Drew Houston needed Dropbox, or Brian Chesky and Joe Gebbia needed Airbnb?

Quite often at YC I find myself asking founders "Would you use this thing yourself, if you hadn't written it?" and you'd be surprised how often the answer is no.

[15] Paul Buchheit points out that trying to sell something bad can be a source of better ideas:

"The best technique I've found for dealing with YC companies that have bad ideas is to tell them to go sell the product ASAP (before wasting time building it). Not only do they learn that nobody wants what they are building, they very often come back with a real idea that they discovered in the process of trying to sell the bad idea."

[16] Here's a recipe that might produce the next Facebook, if you're college

students. If you have a connection to one of the more powerful sororities at your school, approach the queen bees thereof and offer to be their personal IT consultants, building anything they could imagine needing in their social lives that didn't already exist. Anything that got built this way would be very promising, because such users are not just the most demanding but also the perfect point to spread from.

I have no idea whether this would work.

[17] And the reason it used a TV for a monitor is that Steve Wozniak started out by solving his own problems. He, like most of his peers, couldn't afford a monitor.

**Thanks** to Sam Altman, Mike Arrington, Paul Buchheit, John Collison, Patrick Collison, Garry Tan, and Harj Taggar for reading drafts of this, and Marc Andreessen, Joe Gebbia, Reid Hoffman, Shel Kaphan, Mike Moritz and Kevin Systrom for answering my questions about startup history.

[Japanese Translation](#)

[Italian Translation](#)

[Spanish Translation](#)

# Startup Investing Trends

June 2013

*(This talk was written for an audience of investors.)*

Y Combinator has now funded 564 startups including the current batch, which has 53. The total valuation of the 287 that have valuations (either by raising an equity round, getting acquired, or dying) is about \$11.7 billion, and the 511 prior to the current batch have collectively raised about \$1.7 billion. [\[1\]](#)

As usual those numbers are dominated by a few big winners. The top 10 startups account for 8.6 of that 11.7 billion. But there is a peloton of younger startups behind them. There are about 40 more that have a shot at being really big.

Things got a little out of hand last summer when we had 84 companies in the batch, so we tightened up our filter to decrease the batch size. [\[2\]](#) Several journalists have tried to interpret that as evidence for some macro story they were telling, but the reason had nothing to do with any external trend. The reason was that we discovered we were using an  $n^2$  algorithm, and we needed to buy time to fix it. Fortunately we've come up with several techniques for sharding YC, and the problem now seems to be fixed. With a new more scalable model and only 53 companies, the current batch feels like a walk in the park. I'd guess we can grow another 2 or 3x before hitting the next bottleneck. [\[3\]](#)

One consequence of funding such a large number of startups is that we see trends early. And since fundraising is one of the main things we help startups with, we're in a good position to notice trends in investing.

I'm going to take a shot at describing where these trends are leading. Let's start with the most basic question: will the future be better or worse than the past? Will investors, in the aggregate, make more money or less?

I think more. There are multiple forces at work, some of which will decrease returns, and some of which will increase them. I can't predict for sure which forces will prevail, but I'll describe them and you can decide for yourself.

There are two big forces driving change in startup funding: it's becoming cheaper to start a startup, and startups are becoming a more normal thing to do.

When I graduated from college in 1986, there were essentially two options: get a job or go to grad school. Now there's a third: start your own company. That's a big change. In principle it was possible to start your own company in 1986 too, but it didn't seem like a real possibility. It seemed possible to start a consulting company, or a niche product company, but it didn't seem possible to start a company that would become big. [4]

That kind of change, from 2 paths to 3, is the sort of big social shift that only happens once every few generations. I think we're still at the beginning of this one. It's hard to predict how big a deal it will be. As big a deal as the Industrial Revolution? Maybe. Probably not. But it will be a big enough deal that it takes almost everyone by surprise, because those big social shifts always do.

One thing we can say for sure is that there will be a lot more startups. The monolithic, hierarchical companies of the mid 20th century are being [replaced](#) by networks of smaller companies. This process is not just something happening now in Silicon Valley. It started decades ago, and it's happening as far afield as the car industry. It has a long way to run. [5]

The other big driver of change is that startups are becoming cheaper to start. And in fact the two forces are related: the decreasing cost of starting a startup is one of the reasons startups are becoming a more normal thing to do.

The fact that startups need less money means founders will increasingly have the upper hand over investors. You still need just as much of their energy and imagination, but they don't need as much of your money. Because founders have the upper hand, they'll retain an increasingly large share of the stock in, and [control of](#), their companies. Which means investors will get less stock and less control.

Does that mean investors will make less money? Not necessarily, because there will be more good startups. The total amount of desirable startup stock available to investors will probably increase, because the number of desirable startups will probably grow faster than the percentage they sell to investors shrinks.

There's a rule of thumb in the VC business that there are about 15 companies a year that will be really successful. Although a lot of investors unconsciously treat this number as if it were some sort of cosmological constant, I'm certain it isn't. There are probably limits on the rate at which technology can develop, but that's not the limiting factor now. If it were, each successful startup would be founded the month it became possible, and that is not the case. Right now the limiting factor on the number of big hits is the number of sufficiently good founders starting companies, and that number can and will increase. There are still a lot of people who'd make great founders who never end up starting a company. You can see that from how randomly some of the most successful startups got started. So many of the biggest startups almost didn't happen that there must be a lot of equally good startups that actually didn't happen.

There might be 10x or even 50x more good founders out there. As more of them go ahead and start startups, those 15 big hits a year could easily become 50 or even 100. [6]

What about returns, though? Are we heading for a world in which returns will be pinched by increasingly high valuations? I think the top firms will actually make more money than they have in the past. High returns don't come from investing at low valuations. They come from investing in the companies that do really well. So if there are more of those to be had each year, the best pickers should have more hits.

This means there should be more variability in the VC business. The firms that can recognize and attract the best startups will do even better, because there will be more of them to recognize and attract. Whereas the bad firms will get the leftovers, as they do now, and yet pay a higher price for them.

Nor do I think it will be a problem that founders keep control of their companies for longer. The empirical evidence on that is already clear: investors make more money as founders' bitches than their bosses. Though somewhat humiliating, this is actually good news for investors, because it takes less time to serve founders than to micromanage them.

What about angels? I think there is a lot of opportunity there. It used to suck to be an angel investor. You couldn't get access to the best deals, unless you got lucky like Andy Bechtolsheim, and when you did invest in a startup, VCs might try to strip you of your stock when they arrived later. Now an angel can go to something like Demo Day or AngelList and have access to the same deals VCs do. And the days when VCs could wash angels out of the cap table are long gone.

I think one of the biggest unexploited opportunities in startup investing right now is angel-sized investments made quickly. Few investors understand the cost that raising money from them imposes on startups. When the company consists only of the founders, everything grinds to a halt during fundraising, which can easily take 6 weeks. The current high cost of fundraising means there is room for low-cost investors to undercut the rest. And in this context, low-cost means deciding quickly. If there were a reputable investor who invested \$100k on good terms and promised to decide yes or no within 24 hours, they'd get access to almost all the best deals, because every good startup would approach them first. It would be up to them to pick, because every bad startup would approach them first too, but at least they'd see everything. Whereas if an investor is notorious for taking a long time to make up their mind or negotiating a lot about valuation, founders will save them for last. And in the case of the most promising startups, which tend to have an easy time raising money, last can easily become never.

Will the number of big hits grow linearly with the total number of new startups? Probably not, for two reasons. One is that the scariness of starting a startup in the old days was a pretty effective filter. Now that the cost of failing is becoming lower, we should expect founders to do it more. That's not a bad thing. It's common in

technology for an innovation that decreases the cost of failure to increase the number of failures and yet leave you net ahead.

The other reason the number of big hits won't grow proportionately to the number of startups is that there will start to be an increasing number of idea clashes. Although the finiteness of the number of good ideas is not the reason there are only 15 big hits a year, the number has to be finite, and the more startups there are, the more we'll see multiple companies doing the same thing at the same time. It will be interesting, in a bad way, if idea clashes become a lot more common. [7]

Mostly because of the increasing number of early failures, the startup business of the future won't simply be the same shape, scaled up. What used to be an obelisk will become a pyramid. It will be a little wider at the top, but a lot wider at the bottom.

What does that mean for investors? One thing it means is that there will be more opportunities for investors at the earliest stage, because that's where the volume of our imaginary solid is growing fastest. Imagine the obelisk of investors that corresponds to the obelisk of startups. As it widens out into a pyramid to match the startup pyramid, all the contents are adhering to the top, leaving a vacuum at the bottom.

That opportunity for investors mostly means an opportunity for new investors, because the degree of risk an existing investor or firm is comfortable taking is one of the hardest things for them to change. Different types of investors are adapted to different degrees of risk, but each has its specific degree of risk deeply imprinted on it, not just in the procedures they follow but in the personalities of the people who work there.

I think the biggest danger for VCs, and also the biggest opportunity, is at the series A stage. Or rather, what used to be the series A stage before series As turned into de facto series B rounds.

Right now, VCs often knowingly invest too much money at the series A stage. They do it because they feel they need to get a big chunk of each series A company to compensate for the opportunity cost of the board seat it consumes. Which means when there is a lot of competition for a deal, the number that moves is the valuation (and thus amount invested) rather than the percentage of the company being sold. Which means, especially in the case of more promising startups, that series A investors often make companies take more money than they want.

Some VCs lie and claim the company really needs that much. Others are more candid, and admit their financial models require them to own a certain percentage of each company. But we all know the amounts being raised in series A rounds are not determined by asking what would be best for the companies. They're determined by VCs starting from the amount of the company they want to own, and the market setting the valuation and thus the amount invested.

Like a lot of bad things, this didn't happen intentionally. The VC business backed into it as their initial assumptions gradually became obsolete. The traditions and financial models of the VC business were established when founders needed investors more. In those days it was natural for founders to sell VCs a big chunk of their company in the series A round. Now founders would prefer to sell less, and VCs are digging in their heels because they're not sure if they can make money buying less than 20% of each series A company.

The reason I describe this as a danger is that series A investors are increasingly at odds with the startups they supposedly serve, and that tends to come back to bite you eventually. The reason I describe it as an opportunity is that there is now a lot of potential energy built up, as the market has moved away from VCs' traditional business model. Which means the first VC to break ranks and start to do series A rounds for as much equity as founders want to sell (and with no "option pool" that comes only from the founders' shares) stands to reap huge benefits.

What will happen to the VC business when that happens? Hell if I know. But I bet that particular firm will end up ahead. If one top-tier VC firm started to do series A rounds that started from the amount the company needed to raise and let the percentage acquired vary with the market, instead of the other way around, they'd instantly get almost all the best startups. And that's where the money is.

You can't fight market forces forever. Over the last decade we've seen the percentage of the company sold in series A rounds creep inexorably downward. 40% used to be common. Now VCs are fighting to hold the line at 20%. But I am daily waiting for the line to collapse. It's going to happen. You may as well anticipate it, and look bold.

Who knows, maybe VCs will make more money by doing the right thing. It wouldn't be the first time that happened. Venture capital is a business where occasional big successes generate hundredfold returns. How much confidence can you really have in financial models for something like that anyway? The big successes only have to get a tiny bit less occasional to compensate for a 2x decrease in the stock sold in series A rounds.

If you want to find new opportunities for investing, look for things founders complain about. Founders are your customers, and the things they complain about are unsatisfied demand. I've given two examples of things founders complain about most—investors who take too long to make up their minds, and excessive dilution in series A rounds—so those are good places to look now. But the more general recipe is: do something founders want.

## Notes



[1] I realize revenue and not fundraising is the proper test of success for a startup. The reason we quote statistics about fundraising is because those are the numbers we have. We couldn't talk meaningfully about revenues without including the numbers from the most successful startups, and we don't have those. We often discuss revenue growth with the earlier stage startups, because that's how we gauge their progress, but when companies reach a certain size it gets presumptuous for a seed investor to do that.

In any case, companies' market caps do eventually become a function of revenues, and post-money valuations of funding rounds are at least guesses by pros about where those market caps will end up.

The reason only 287 have valuations is that the rest have mostly raised money on convertible notes, and although convertible notes often have valuation caps, a valuation cap is merely an upper bound on a valuation.

[2] We didn't try to accept a particular number. We have no way of doing that even if we wanted to. We just tried to be significantly pickier.

[3] Though you never know with bottlenecks, I'm guessing the next one will be coordinating efforts among partners.

[4] I realize starting a company doesn't have to mean starting a [startup](#). There will be lots of people starting normal companies too. But that's not relevant to an audience of investors.

Geoff Ralston reports that in Silicon Valley it seemed thinkable to start a startup in the mid 1980s. It would have started there. But I know it didn't to undergraduates on the East Coast.

[5] This trend is one of the main causes of the increase in economic inequality in the US since the mid twentieth century. The person who would in 1950 have been the general manager of the x division of Megacorp is now the founder of the x company, and owns significant equity in it.

[6] If Congress passes the [founder visa](#) in a non-broken form, that alone could in principle get us up to 20x, since 95% of the world's population lives outside the US.

[7] If idea clashes got bad enough, it could change what it means to be a startup. We currently advise startups mostly to ignore competitors. We tell them startups are competitive like running, not like soccer; you don't have to go and steal the ball away from the other team. But if idea clashes became common enough, maybe you'd start to have to. That would be unfortunate.

**Thanks** to Sam Altman, Paul Buchheit, Dalton Caldwell, Patrick Collison, Jessica Livingston, Andrew Mason, Geoff Ralston, and Garry Tan for reading drafts of this.



# Do Things that Don't Scale

July 2013

One of the most common types of advice we give at Y Combinator is to do things that don't scale. A lot of would-be founders believe that startups either take off or don't. You build something, make it available, and if you've made a better mousetrap, people beat a path to your door as promised. Or they don't, in which case the market must not exist. [\[1\]](#)

Actually startups take off because the founders make them take off. There may be a handful that just grew by themselves, but usually it takes some sort of push to get them going. A good metaphor would be the cranks that car engines had before they got electric starters. Once the engine was going, it would keep going, but there was a separate and laborious process to get it going.

## Recruit

The most common unscalable thing founders have to do at the start is to recruit users manually. Nearly all startups have to. You can't wait for users to come to you. You have to go out and get them.

Stripe is one of the most successful startups we've funded, and the problem they solved was an urgent one. If anyone could have sat back and waited for users, it was Stripe. But in fact they're famous within YC for aggressive early user acquisition.

Startups building things for other startups have a big pool of potential users in the other companies we've funded, and none took better advantage of it than Stripe. At YC we use the term "Collison installation" for the technique they invented. More diffident founders ask "Will you try our beta?" and if the answer is yes, they say "Great, we'll send you a link." But the Collison brothers weren't going to wait. When anyone agreed to try Stripe they'd say "Right then, give me your laptop" and set them up on the spot.

There are two reasons founders resist going out and recruiting users individually. One is a combination of shyness and laziness. They'd rather sit at home writing code than go out and talk to a bunch of strangers and probably be rejected by most of them. But for a startup to succeed, at least one founder (usually the CEO)

will have to spend a lot of time on sales and marketing. [2]

The other reason founders ignore this path is that the absolute numbers seem so small at first. This can't be how the big, famous startups got started, they think. The mistake they make is to underestimate the power of compound growth. We encourage every startup to measure their progress by weekly [growth rate](#). If you have 100 users, you need to get 10 more next week to grow 10% a week. And while 110 may not seem much better than 100, if you keep growing at 10% a week you'll be surprised how big the numbers get. After a year you'll have 14,000 users, and after 2 years you'll have 2 million.

You'll be doing different things when you're acquiring users a thousand at a time, and growth has to slow down eventually. But if the market exists you can usually start by recruiting users manually and then gradually switch to less manual methods. [3]

Airbnb is a classic example of this technique. Marketplaces are so hard to get rolling that you should expect to take heroic measures at first. In Airbnb's case, these consisted of going door to door in New York, recruiting new users and helping existing ones improve their listings. When I remember the Airbnbs during YC, I picture them with rolly bags, because when they showed up for tuesday dinners they'd always just flown back from somewhere.

## **Fragile**

Airbnb now seems like an unstoppable juggernaut, but early on it was so fragile that about 30 days of going out and engaging in person with users made the difference between success and failure.

That initial fragility was not a unique feature of Airbnb. Almost all startups are fragile initially. And that's one of the biggest things inexperienced founders and investors (and reporters and know-it-alls on forums) get wrong about them. They unconsciously judge larval startups by the standards of established ones. They're like someone looking at a newborn baby and concluding "there's no way this tiny creature could ever accomplish anything."

It's harmless if reporters and know-it-alls dismiss your startup. They always get things wrong. It's even ok if investors dismiss your startup; they'll change their minds when they see growth. The big danger is that you'll dismiss your startup yourself. I've seen it happen. I often have to encourage founders who don't see the full potential of what they're building. Even Bill Gates made that mistake. He returned to Harvard for the fall semester after starting Microsoft. He didn't stay long, but he wouldn't have returned at all if he'd realized Microsoft was going to be even a fraction of the size it turned out to be. [4]

The question to ask about an early stage startup is not "is this company taking over the world?" but "how big could this company get if the founders did the right things?" And the right things often seem both laborious and inconsequential at the

time. Microsoft can't have seemed very impressive when it was just a couple guys in Albuquerque writing Basic interpreters for a market of a few thousand hobbyists (as they were then called), but in retrospect that was the optimal path to dominating microcomputer software. And I know Brian Chesky and Joe Gebbia didn't feel like they were en route to the big time as they were taking "professional" photos of their first hosts' apartments. They were just trying to survive. But in retrospect that too was the optimal path to dominating a big market.

How do you find users to recruit manually? If you build something to solve [your own problems](#), then you only have to find your peers, which is usually straightforward. Otherwise you'll have to make a more deliberate effort to locate the most promising vein of users. The usual way to do that is to get some initial set of users by doing a comparatively untargeted launch, and then to observe which kind seem most enthusiastic, and seek out more like them. For example, Ben Silbermann noticed that a lot of the earliest Pinterest users were interested in design, so he went to a conference of design bloggers to recruit users, and that worked well. [5]

## **Delight**

You should take extraordinary measures not just to acquire users, but also to make them happy. For as long as they could (which turned out to be surprisingly long), Wufoo sent each new user a hand-written thank you note. Your first users should feel that signing up with you was one of the best choices they ever made. And you in turn should be racking your brains to think of new ways to delight them.

Why do we have to teach startups this? Why is it counterintuitive for founders? Three reasons, I think.

One is that a lot of startup founders are trained as engineers, and customer service is not part of the training of engineers. You're supposed to build things that are robust and elegant, not be slavishly attentive to individual users like some kind of salesperson. Ironically, part of the reason engineering is traditionally averse to handholding is that its traditions date from a time when engineers were less powerful — when they were only in charge of their narrow domain of building things, rather than running the whole show. You can be ornery when you're Scotty, but not when you're Kirk.

Another reason founders don't focus enough on individual customers is that they worry it won't scale. But when founders of larval startups worry about this, I point out that in their current state they have nothing to lose. Maybe if they go out of their way to make existing users super happy, they'll one day have too many to do so much for. That would be a great problem to have. See if you can make it happen. And incidentally, when it does, you'll find that delighting customers scales better than you expected. Partly because you can usually find ways to make anything scale more than you would have predicted, and partly because delighting customers will by then have permeated your culture.

I have never once seen a startup lured down a blind alley by trying too hard to make their initial users happy.

But perhaps the biggest thing preventing founders from realizing how attentive they could be to their users is that they've never experienced such attention themselves. Their standards for customer service have been set by the companies they've been customers of, which are mostly big ones. Tim Cook doesn't send you a hand-written note after you buy a laptop. He can't. But you can. That's one advantage of being small: you can provide a level of service no big company can.

[6]

Once you realize that existing conventions are not the upper bound on user experience, it's interesting in a very pleasant way to think about how far you could go to delight your users.

## Experience

I was trying to think of a phrase to convey how extreme your attention to users should be, and I realized Steve Jobs had already done it: insanely great. Steve wasn't just using "insanely" as a synonym for "very." He meant it more literally — that one should focus on quality of execution to a degree that in everyday life would be considered pathological.

All the most successful startups we've funded have, and that probably doesn't surprise would-be founders. What novice founders don't get is what insanely great translates to in a larval startup. When Steve Jobs started using that phrase, Apple was already an established company. He meant the Mac (and its documentation and even packaging — such is the nature of obsession) should be insanely well designed and manufactured. That's not hard for engineers to grasp. It's just a more extreme version of designing a robust and elegant product.

What founders have a hard time grasping (and Steve himself might have had a hard time grasping) is what insanely great morphs into as you roll the time slider back to the first couple months of a startup's life. It's not the product that should be insanely great, but the experience of being your user. The product is just one component of that. For a big company it's necessarily the dominant one. But you can and should give users an insanely great experience with an early, incomplete, buggy product, if you make up the difference with attentiveness.

Can, perhaps, but should? Yes. Over-engaging with early users is not just a permissible technique for getting growth rolling. For most successful startups it's a necessary part of the feedback loop that makes the product good. Making a better mousetrap is not an atomic operation. Even if you start the way most successful startups have, by building something you yourself need, the first thing you build is never quite right. And except in domains with big penalties for making mistakes, it's often better not to aim for perfection initially. In software, especially, it usually works best to get something in front of users as soon as it has a quantum of utility,

and then see what they do with it. Perfectionism is often an excuse for procrastination, and in any case your initial model of users is always inaccurate, even if you're one of them. [7]

The feedback you get from engaging directly with your earliest users will be the best you ever get. When you're so big you have to resort to focus groups, you'll wish you could go over to your users' homes and offices and watch them use your stuff like you did when there were only a handful of them.

## Fire

Sometimes the right unscalable trick is to focus on a deliberately narrow market. It's like keeping a fire contained at first to get it really hot before adding more logs.

That's what Facebook did. At first it was just for Harvard students. In that form it only had a potential market of a few thousand people, but because they felt it was really for them, a critical mass of them signed up. After Facebook stopped being for Harvard students, it remained for students at specific colleges for quite a while. When I interviewed Mark Zuckerberg at Startup School, he said that while it was a lot of work creating course lists for each school, doing that made students feel the site was their natural home.

Any startup that could be described as a marketplace usually has to start in a subset of the market, but this can work for other startups as well. It's always worth asking if there's a subset of the market in which you can get a critical mass of users quickly. [8]

Most startups that use the contained fire strategy do it unconsciously. They build something for themselves and their friends, who happen to be the early adopters, and only realize later that they could offer it to a broader market. The strategy works just as well if you do it unconsciously. The biggest danger of not being consciously aware of this pattern is for those who naively discard part of it. E.g. if you don't build something for yourself and your friends, or even if you do, but you come from the corporate world and your friends are not early adopters, you'll no longer have a perfect initial market handed to you on a platter.

Among companies, the best early adopters are usually other startups. They're more open to new things both by nature and because, having just been started, they haven't made all their choices yet. Plus when they succeed they grow fast, and you with them. It was one of many unforeseen advantages of the YC model (and specifically of making YC big) that B2B startups now have an instant market of hundreds of other startups ready at hand.

## Meraki

For [hardware startups](#) there's a variant of doing things that don't scale that we call "pulling a Meraki." Although we didn't fund Meraki, the founders were Robert

Morris's grad students, so we know their history. They got started by doing something that really doesn't scale: assembling their routers themselves.

Hardware startups face an obstacle that software startups don't. The minimum order for a factory production run is usually several hundred thousand dollars. Which can put you in a catch-22: without a product you can't generate the growth you need to raise the money to manufacture your product. Back when hardware startups had to rely on investors for money, you had to be pretty convincing to overcome this. The arrival of crowdfunding (or more precisely, preorders) has helped a lot. But even so I'd advise startups to pull a Meraki initially if they can. That's what Pebble did. The Pebbles [assembled](#) the first several hundred watches themselves. If they hadn't gone through that phase, they probably wouldn't have sold \$10 million worth of watches when they did go on Kickstarter.

Like paying excessive attention to early customers, fabricating things yourself turns out to be valuable for hardware startups. You can tweak the design faster when you're the factory, and you learn things you'd never have known otherwise. Eric Migicovsky of Pebble said one of the things he learned was "how valuable it was to source good screws." Who knew?

## Consult

Sometimes we advise founders of B2B startups to take over-engagement to an extreme, and to pick a single user and act as if they were consultants building something just for that one user. The initial user serves as the form for your mold; keep tweaking till you fit their needs perfectly, and you'll usually find you've made something other users want too. Even if there aren't many of them, there are probably adjacent territories that have more. As long as you can find just one user who really needs something and can act on that need, you've got a toehold in making something people want, and that's as much as any startup needs initially.

[9]

Consulting is the canonical example of work that doesn't scale. But (like other ways of bestowing one's favors liberally) it's safe to do it so long as you're not being paid to. That's where companies cross the line. So long as you're a product company that's merely being extra attentive to a customer, they're very grateful even if you don't solve all their problems. But when they start paying you specifically for that attentiveness — when they start paying you by the hour — they expect you to do everything.

Another consulting-like technique for recruiting initially lukewarm users is to use your software yourselves on their behalf. We did that at Viaweb. When we approached merchants asking if they wanted to use our software to make online stores, some said no, but they'd let us make one for them. Since we would do anything to get users, we did. We felt pretty lame at the time. Instead of organizing big strategic e-commerce partnerships, we were trying to sell luggage and pens and men's shirts. But in retrospect it was exactly the right thing to do, because it taught us how it would feel to merchants to use our software.



Sometimes the feedback loop was near instantaneous: in the middle of building some merchant's site I'd find I needed a feature we didn't have, so I'd spend a couple hours implementing it and then resume building the site.

## **Manual**

There's a more extreme variant where you don't just use your software, but are your software. When you only have a small number of users, you can sometimes get away with doing by hand things that you plan to automate later. This lets you launch faster, and when you do finally automate yourself out of the loop, you'll know exactly what to build because you'll have muscle memory from doing it yourself.

When manual components look to the user like software, this technique starts to have aspects of a practical joke. For example, the way Stripe delivered "instant" merchant accounts to its first users was that the founders manually signed them up for traditional merchant accounts behind the scenes.

Some startups could be entirely manual at first. If you can find someone with a problem that needs solving and you can solve it manually, go ahead and do that for as long as you can, and then gradually automate the bottlenecks. It would be a little frightening to be solving users' problems in a way that wasn't yet automatic, but less frightening than the far more common case of having something automatic that doesn't yet solve anyone's problems.

## **Big**

I should mention one sort of initial tactic that usually doesn't work: the Big Launch. I occasionally meet founders who seem to believe startups are projectiles rather than powered aircraft, and that they'll make it big if and only if they're launched with sufficient initial velocity. They want to launch simultaneously in 8 different publications, with embargoes. And on a tuesday, of course, since they read somewhere that's the optimum day to launch something.

It's easy to see how little launches matter. Think of some successful startups. How many of their launches do you remember? All you need from a launch is some initial core of users. How well you're doing a few months later will depend more on how happy you made those users than how many there were of them. [\[10\]](#)

So why do founders think launches matter? A combination of solipsism and laziness. They think what they're building is so great that everyone who hears about it will immediately sign up. Plus it would be so much less work if you could get users merely by broadcasting your existence, rather than recruiting them one at a time. But even if what you're building really is great, getting users will always be a gradual process — partly because great things are usually also novel, but mainly because users have other things to think about.

Partnerships too usually don't work. They don't work for startups in general, but

they especially don't work as a way to get growth started. It's a common mistake among inexperienced founders to believe that a partnership with a big company will be their big break. Six months later they're all saying the same thing: that was way more work than we expected, and we ended up getting practically nothing out of it. [11]

It's not enough just to do something extraordinary initially. You have to make an extraordinary *effort* initially. Any strategy that omits the effort — whether it's expecting a big launch to get you users, or a big partner — is ipso facto suspect.

## Vector

The need to do something unscalably laborious to get started is so nearly universal that it might be a good idea to stop thinking of startup ideas as scalars. Instead we should try thinking of them as pairs of what you're going to build, plus the unscalable thing(s) you're going to do initially to get the company going.

It could be interesting to start viewing startup ideas this way, because now that there are two components you can try to be imaginative about the second as well as the first. But in most cases the second component will be what it usually is — recruit users manually and give them an overwhelmingly good experience — and the main benefit of treating startups as vectors will be to remind founders they need to work hard in two dimensions. [12]

In the best case, both components of the vector contribute to your company's DNA: the unscalable things you have to do to get started are not merely a necessary evil, but change the company permanently for the better. If you have to be aggressive about user acquisition when you're small, you'll probably still be aggressive when you're big. If you have to manufacture your own hardware, or use your software on users's behalf, you'll learn things you couldn't have learned otherwise. And most importantly, if you have to work hard to delight users when you only have a handful of them, you'll keep doing it when you have a lot.

## Notes

[1] Actually Emerson never mentioned mousetraps specifically. He wrote "If a man has good corn or wood, or boards, or pigs, to sell, or can make better chairs or knives, crucibles or church organs, than anybody else, you will find a broad hard-beaten road to his house, though it be in the woods."

[2] Thanks to Sam Altman for suggesting I make this explicit. And no, you can't avoid doing sales by hiring someone to do it for you. You have to do sales yourself

initially. Later you can hire a real salesperson to replace you.

[3] The reason this works is that as you get bigger, your size helps you grow. Patrick Collison wrote "At some point, there was a very noticeable change in how Stripe felt. It tipped from being this boulder we had to push to being a train car that in fact had its own momentum."

[4] One of the more subtle ways in which YC can help founders is by calibrating their ambitions, because we know exactly how a lot of successful startups looked when they were just getting started.

[5] If you're building something for which you can't easily get a small set of users to observe — e.g. enterprise software — and in a domain where you have no connections, you'll have to rely on cold calls and introductions. But should you even be working on such an idea?

[6] Garry Tan pointed out an interesting trap founders fall into in the beginning. They want so much to seem big that they imitate even the flaws of big companies, like indifference to individual users. This seems to them more "professional." Actually it's better to embrace the fact that you're small and use whatever advantages that brings.

[7] Your user model almost couldn't be perfectly accurate, because users' needs often change in response to what you build for them. Build them a microcomputer, and suddenly they need to run spreadsheets on it, because the arrival of your new microcomputer causes someone to invent the spreadsheet.

[8] If you have to choose between the subset that will sign up quickest and those that will pay the most, it's usually best to pick the former, because those are probably the early adopters. They'll have a better influence on your product, and they won't make you expend as much effort on sales. And though they have less money, you don't need that much to maintain your target growth rate early on.

[9] Yes, I can imagine cases where you could end up making something that was really only useful for one user. But those are usually obvious, even to inexperienced founders. So if it's not obvious you'd be making something for a market of one, don't worry about that danger.

[10] There may even be an inverse correlation between launch magnitude and success. The only launches I remember are famous flops like the Segway and Google Wave. Wave is a particularly alarming example, because I think it was actually a great idea that was killed partly by its overdone launch.

[11] Google grew big on the back of Yahoo, but that wasn't a partnership. Yahoo was their customer.

[12] It will also remind founders that an idea where the second component is empty — an idea where there is nothing you can do to get going, e.g. because you

have no way to find users to recruit manually — is probably a bad idea, at least for those founders.

**Thanks** to Sam Altman, Paul Buchheit, Patrick Collison, Kevin Hale, Steven Levy, Jessica Livingston, Geoff Ralston, and Garry Tan for reading drafts of this.

[Japanese Translation](#)

[Russian Translation](#)

[French Translation](#)

[Arabic Translation](#)

[Italian Translation](#)

[Korean Translation](#)

**THE END**

