

Table of Contents

- [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](#)
- [How to Convince Investors](#)
- [Investor Herd Dynamics](#)
- [How to Raise Money](#)
- [Before the Startup](#)
- [Mean People Fail](#)
- [The Fatal Pinch](#)
- [How You Know](#)
- [How to Be an Expert in a Changing World](#)
- [Let the Other 95% of Great Programmers In](#)
- [Don't Talk to Corp Dev](#)
- [What Doesn't Seem Like Work?](#)
- [The Ronco Principle](#)
- [What Microsoft Is this the Altair Basic of?](#)
- [Change Your Name](#)
- [Why It's Safe for Founders to Be Nice](#)
- [Default Alive or Default Dead?](#)
- [Write Like You Talk](#)
- [A Way to Detect Bias](#)
- [Jessica Livingston](#)
- [The Refragmentation](#)
- [Economic Inequality](#)
- [Life is Short](#)
- [How to Make Pittsburgh a Startup Hub](#)
- [The Risk of Discovery](#)
- [Charisma / Power](#)

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.

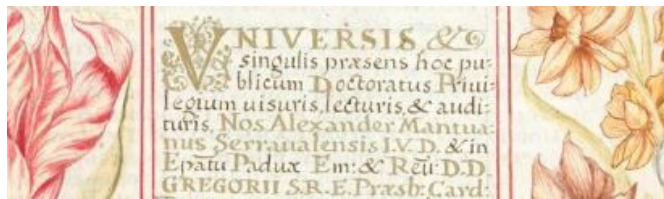
VCs think they're playing a zero sum game. In fact, it's not even that. If you lose a deal to Benchmark, you lose that deal, but VC as an industry still wins. If you lose a deal to None, all VCs lose.

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?

Notes

[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. [2] 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.

- [Russian Translation](#)

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 response:

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.

- [Russian Translation](#)

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 adieu 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 downturn 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. [2]

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 !\[\]\(467d80e979964f7f8c752fb22248b5b7_img.jpg\) Software Eats the World !\[\]\(b71552d33dbf62adf5e5199a70ee02bf_img.jpg\)?](#)

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. [\[15\]](#)

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♦ 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 scaleable 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 untargated 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](#)

How to Convince Investors

August 2013

When people hurt themselves lifting heavy things, it's usually because they try to lift with their back. The right way to lift heavy things is to let your legs do the work. Inexperienced founders make the same mistake when trying to convince investors. They try to convince with their pitch. Most would be better off if they let their startup do the work — if they started by understanding why their startup is worth investing in, then simply explained this well to investors.

Investors are looking for startups that will be very successful. But that test is not as simple as it sounds. In startups, as in a lot of other domains, the distribution of outcomes follows a power law, but in startups the curve is startlingly steep. The big successes are so big they dwarf the rest. And since there are only a handful each year (the conventional wisdom is 15), investors treat "big success" as if it were binary. Most are interested in you if you seem like you have a chance, however small, of being one of the 15 big successes, and otherwise not. [1]

(There are a handful of angels who'd be interested in a company with a high probability of being moderately successful. But angel investors like big successes too.)

How do you seem like you'll be one of the big successes? You need three things: formidable founders, a promising market, and (usually) some evidence of success so far.

Formidable

The most important ingredient is formidable founders. Most investors decide in the first few minutes whether you seem like a winner or a loser, and once their opinion is set it's hard to change. [2] Every startup has reasons both to invest and not to invest. If investors think you're a winner they focus on the former, and if not they focus on the latter. For example, it might be a rich market, but with a slow sales cycle. If investors are impressed with you as founders, they say they want to invest because it's a rich market, and if not, they say they can't invest because of the slow sales cycle.

They're not necessarily trying to mislead you. Most investors are genuinely unclear in their own minds why they like or dislike startups. If you seem like a winner, they'll like your idea more. But don't be too smug about this weakness of theirs, because you have it too; almost everyone does.

There is a role for ideas of course. They're fuel for the fire that starts with liking the founders. Once investors like you, you'll see them reaching for ideas: they'll be saying "yes, and you could also do x." (Whereas when they don't like you, they'll be saying "but

what about y?"

But the foundation of convincing investors is to seem formidable, and since this isn't a word most people use in conversation much, I should explain what it means. A formidable person is one who seems like they'll get what they want, regardless of whatever obstacles are in the way. Formidable is close to confident, except that someone could be confident and mistaken. Formidable is roughly justifiably confident.

There are a handful of people who are really good at seeming formidable — some because they actually are very formidable and just let it show, and others because they are more or less con artists. [3] But most founders, including many who will go on to start very successful companies, are not that good at seeming formidable the first time they try fundraising. What should they do? [4]

What they should not do is try to imitate the swagger of more experienced founders. Investors are not always that good at judging technology, but they're good at judging confidence. If you try to act like something you're not, you'll just end up in an uncanny valley. You'll depart from sincere, but never arrive at convincing.

Truth

The way to seem most formidable as an inexperienced founder is to stick to the truth. How formidable you seem isn't a constant. It varies depending on what you're saying. Most people can seem confident when they're saying "one plus one is two," because they know it's true. The most diffident person would be puzzled and even slightly contemptuous if they told a VC "one plus one is two" and the VC reacted with skepticism. The magic ability of people who are good at seeming formidable is that they can do this with the sentence "we're going to make a billion dollars a year." But you can do the same, if not with that sentence with some fairly impressive ones, so long as you convince yourself first.

That's the secret. Convince yourself that your startup is worth investing in, and then when you explain this to investors they'll believe you. And by convince yourself, I don't mean play mind games with yourself to boost your confidence. I mean truly evaluate whether your startup is worth investing in. If it isn't, don't try to raise money. [5] But if it is, you'll be telling the truth when you tell investors it's worth investing in, and they'll sense that. You don't have to be a smooth presenter if you understand something well and tell the truth about it.

To evaluate whether your startup is worth investing in, you have to be a domain expert. If you're not a domain expert, you can be as convinced as you like about your idea, and it will seem to investors no more than an instance of the Dunning-Kruger effect. Which in fact it will usually be. And investors can tell fairly quickly whether you're a domain expert by how well you answer their questions. Know everything about your market. [6]

Why do founders persist in trying to convince investors of things they're not convinced of themselves? Partly because we've all been trained to.

When my friends Robert Morris and Trevor Blackwell were in grad school, one of their fellow students was on the receiving end of a question from their faculty advisor that we still quote today. When the unfortunate fellow got to his last slide, the professor burst out:

Which one of these conclusions do you actually believe?

One of the artifacts of the way schools are organized is that we all get trained to talk even when we have nothing to say. If you have a ten page paper due, then ten pages you must write, even if you only have one page of ideas. Even if you have no ideas. You have to produce something. And all too many startups go into fundraising in the same spirit. When they think it's time to raise money, they try gamely to make the best case they can for their startup. Most never think of pausing beforehand to ask whether what they're saying is actually convincing, because they've all been trained to treat the need to present as a given — as an area of fixed size, over which however much truth they have must needs be spread, however thinly.

The time to raise money is not when you need it, or when you reach some artificial deadline like a Demo Day. It's when you can convince investors, and not before. [Z]

And unless you're a good con artist, you'll never convince investors if you're not convinced yourself. They're far better at detecting bullshit than you are at producing it, even if you're producing it unknowingly. If you try to convince investors before you've convinced yourself, you'll be wasting both your time.

But pausing first to convince yourself will do more than save you from wasting your time. It will force you to organize your thoughts. To convince yourself that your startup is worth investing in, you'll have to figure out why it's worth investing in. And if you can do that you'll end up with more than added confidence. You'll also have a provisional roadmap of how to succeed.

Market

Notice I've been careful to talk about whether a startup is worth investing in, rather than whether it's going to succeed. No one knows whether a startup is going to succeed. And it's a good thing for investors that this is so, because if you could know in advance whether a startup would succeed, the stock price would already be the future price, and there would be no room for investors to make money. Startup investors know that every investment is a bet, and against pretty long odds.

So to prove you're worth investing in, you don't have to prove you're going to succeed, just that you're a sufficiently good bet. What makes a startup a sufficiently good bet? In addition to formidable founders, you need a plausible path to owning a big piece of a big market. Founders think of startups as ideas, but investors think of them as markets. If there are x number of customers who'd pay an average of $\$y$ per year for what you're making, then the total addressable market, or TAM, of your company is $\$xy$. Investors don't expect you to collect all that money, but it's an upper bound on how big you can get.

Your target market has to be big, and it also has to be capturable by you. But the market doesn't have to be big yet, nor do you necessarily have to be in it yet. Indeed, it's often better to start in a [small](#) market that will either turn into a big one or from which you can move into a big one. There just has to be some plausible sequence of hops that leads to dominating a big market a few years down the line.

The standard of plausibility varies dramatically depending on the age of the startup. A three month old company at Demo Day only needs to be a promising experiment that's

worth funding to see how it turns out. Whereas a two year old company raising a series A round needs to be able to show the experiment worked. [8]

But every company that gets really big is "lucky" in the sense that their growth is due mostly to some external wave they're riding, so to make a convincing case for becoming huge, you have to identify some specific trend you'll benefit from. Usually you can find this by asking "why now?" If this is such a great idea, why hasn't someone else already done it? Ideally the answer is that it only recently became a good idea, because something changed, and no one else has noticed yet.

Microsoft for example was not going to grow huge selling Basic interpreters. But by starting there they were perfectly poised to expand up the stack of microcomputer software as microcomputers grew powerful enough to support one. And microcomputers turned out to be a really huge wave, bigger than even the most optimistic observers would have predicted in 1975.

But while Microsoft did really well and there is thus a temptation to think they would have seemed a great bet a few months in, they probably didn't. Good, but not great. No company, however successful, ever looks more than a pretty good bet a few months in. Microcomputers turned out to be a big deal, and Microsoft both executed well and got lucky. But it was by no means obvious that this was how things would play out. Plenty of companies seem as good a bet a few months in. I don't know about startups in general, but at least half the startups we fund could make as good a case as Microsoft could have for being on a path to dominating a large market. And who can reasonably expect more of a startup than that?

Rejection

If you can make as good a case as Microsoft could have, will you convince investors? Not always. A lot of VCs would have rejected Microsoft. [9] Certainly some rejected Google. And getting rejected will put you in a slightly awkward position, because as you'll see when you start fundraising, the most common question you'll get from investors will be "who else is investing?" What do you say if you've been fundraising for a while and no one has committed yet? [10]

The people who are really good at acting formidable often solve this problem by giving investors the impression that while no investors have committed yet, several are about to. This is arguably a permissible tactic. It's slightly dickish of investors to care more about who else is investing than any other aspect of your startup, and misleading them about how far along you are with other investors seems the complementary countermove. It's arguably an instance of scamming a scammer. But I don't recommend this approach to most founders, because most founders wouldn't be able to carry it off. This is the single most common lie told to investors, and you have to be really good at lying to tell members of some profession the most common lie they're told.

If you're not a master of negotiation (and perhaps even if you are) the best solution is to tackle the problem head-on, and to explain why investors have turned you down and why they're mistaken. If you know you're on the right track, then you also know why investors were wrong to reject you. Experienced investors are well aware that the best ideas are also the scariest. They all know about the VCs who rejected Google. If instead of seeming evasive and ashamed about having been turned down (and thereby

implicitly agreeing with the verdict) you talk candidly about what scared investors about you, you'll seem more confident, which they like, and you'll probably also do a better job of presenting that aspect of your startup. At the very least, that worry will now be out in the open instead of being a gotcha left to be discovered by the investors you're currently talking to, who will be proud of and thus attached to their discovery. [\[11\]](#)

This strategy will work best with the best investors, who are both hard to bluff and who already believe most other investors are conventional-minded drones doomed always to miss the big outliers. Raising money is not like applying to college, where you can assume that if you can get into MIT, you can also get into Foobar State. Because the best investors are much smarter than the rest, and the best startup ideas look initially like [bad ideas](#), it's not uncommon for a startup to be rejected by all the VCs except the best ones. That's what happened to Dropbox. Y Combinator started in Boston, and for the first 3 years we ran alternating batches in Boston and Silicon Valley. Because Boston investors were so few and so timid, we used to ship Boston batches out for a second Demo Day in Silicon Valley. Dropbox was part of a Boston batch, which means all those Boston investors got the first look at Dropbox, and none of them closed the deal. Yet another backup and syncing thing, they all thought. A couple weeks later, Dropbox raised a series A round from Sequoia. [\[12\]](#)

Different

Not understanding that investors view investments as bets combines with the ten page paper mentality to prevent founders from even considering the possibility of being certain of what they're saying. They think they're trying to convince investors of something very uncertain — that their startup will be huge — and convincing anyone of something like that must obviously entail some wild feat of salesmanship. But in fact when you raise money you're trying to convince investors of something so much less speculative — whether the company has all the elements of a good bet — that you can approach the problem in a qualitatively different way. You can convince yourself, then convince them.

And when you convince them, use the same matter-of-fact language you used to convince yourself. You wouldn't use vague, grandiose marketing-speak among yourselves. Don't use it with investors either. It not only doesn't work on them, but seems a mark of incompetence. Just be concise. Many investors explicitly use that as a test, reasoning (correctly) that if you can't explain your plans concisely, you don't really understand them. But even investors who don't have a rule about this will be bored and frustrated by unclear explanations. [\[13\]](#)

So here's the recipe for impressing investors when you're not already good at seeming formidable:

1. Make something worth investing in.
2. Understand why it's worth investing in.
3. Explain that clearly to investors.

If you're saying something you know is true, you'll seem confident when you're saying it. Conversely, never let pitching draw you into bullshitting. As long as you stay on the territory of truth, you're strong. Make the truth good, then just tell it.

Notes

[1] There's no reason to believe this number is a constant. In fact it's our explicit goal at Y Combinator to increase it, by encouraging people to start startups who otherwise wouldn't have.

[2] Or more precisely, investors decide whether you're a loser or possibly a winner. If you seem like a winner, they may then, depending on how much you're raising, have several more meetings with you to test whether that initial impression holds up.

But if you seem like a loser they're done, at least for the next year or so. And when they decide you're a loser they usually decide in way less than the 50 minutes they may have allotted for the first meeting. Which explains the astonished stories one always hears about VC inattentiveness. How could these people make investment decisions well when they're checking their messages during startups' presentations? The solution to that mystery is that they've already made the decision.

[3] The two are not mutually exclusive. There are people who are both genuinely formidable, and also really good at acting that way.

[4] How can people who will go on to create giant companies not seem formidable early on? I think the main reason is that their experience so far has trained them to keep their wings folded, as it were. Family, school, and jobs encourage cooperation, not conquest. And it's just as well they do, because even being Genghis Khan is probably 99% cooperation. But the result is that most people emerge from the tube of their upbringing in their early twenties compressed into the shape of the tube. Some find they have wings and start to spread them. But this takes a few years. In the beginning even they don't know yet what they're capable of.

[5] In fact, change what you're doing. You're investing your own time in your startup. If you're not convinced that what you're working on is a sufficiently good bet, why are you even working on that?

[6] When investors ask you a question you don't know the answer to, the best response is neither to bluff nor give up, but instead to explain how you'd figure out the answer. If you can work out a preliminary answer on the spot, so much the better, but explain that's what you're doing.

[7] At YC we try to ensure startups are ready to raise money on Demo Day by encouraging them to ignore investors and instead focus on their companies till about a week before. That way most reach the stage where they're sufficiently convincing well before Demo Day. But not all do, so we also give any startup that wants to the option of deferring to a later Demo Day.

[8] Founders are often surprised by how much harder it is to raise the next round. There is a qualitative difference in investors' attitudes. It's like the difference between being judged as a kid and as an adult. The next time you raise money, it's not enough to be promising. You have to be delivering results.

So although it works well to show growth graphs at either stage, investors treat them differently. At three months, a growth graph is mostly evidence that the founders are effective. At two years, it has to be evidence of a promising market and a company tuned to exploit it.

[9] By this I mean that if the present day equivalent of the 3 month old Microsoft presented at a Demo Day, there would be investors who turned them down. Microsoft itself didn't raise outside money, and indeed the venture business barely existed when they got started in 1975.

[10] The best investors rarely care who else is investing, but mediocre investors almost all do. So you can use this question as a test of investor quality.

[11] To use this technique, you'll have to find out why investors who rejected you did so, or at least what they claim was the reason. That may require asking, because investors don't always volunteer a lot of detail. Make it clear when you ask that you're not trying to dispute their decision — just that if there is some weakness in your plans, you need to know about it. You won't always get a real reason out of them, but you should at least try.

[12] Dropbox wasn't rejected by all the East Coast VCs. There was one firm that wanted to invest but tried to lowball them.

[13] Alfred Lin points out that it's doubly important for the explanation of a startup to be clear and concise, because it has to convince at one remove: it has to work not just on the partner you talk to, but when that partner re-tells it to colleagues.

We consciously optimize for this at YC. When we work with founders create a Demo Day pitch, the last step is to imagine how an investor would sell it to colleagues.

Thanks to Marc Andreessen, Sam Altman, Patrick Collison, Ron Conway, Chris Dixon, Alfred Lin, Ben Horowitz, Steve Huffman, Jessica Livingston, Greg Mcadoo, Andrew Mason, Geoff Ralston, Yuri Sagalov, Emmett Shear, Rajat Suri, Garry Tan, Albert Wenger, Fred Wilson, and Qasar Younis for reading drafts of this.

Investor Herd Dynamics

August 2013

The biggest component in most investors' opinion of you is the opinion of other investors. Which is of course a recipe for exponential growth. When one investor wants to invest in you, that makes other investors want to, which makes others want to, and so on.

Sometimes inexperienced founders mistakenly conclude that manipulating these forces is the essence of fundraising. They hear stories about stampedes to invest in successful startups, and think it's therefore the mark of a successful startup to have this happen. But actually the two are not that highly correlated. Lots of startups that cause stampedes end up flaming out (in extreme cases, partly as a result of the stampede), and lots of very successful startups were only moderately popular with investors the first time they raised money.

So the point of this essay is not to explain how to create a stampede, but merely to explain the forces that generate them. These forces are always at work to some degree in fundraising, and they can cause surprising situations. If you understand them, you can at least avoid being surprised.

One reason investors like you more when other investors like you is that you actually become a better investment. Raising money decreases the risk of failure. Indeed, although investors hate it, you are for this reason justified in raising your valuation for later investors. The investors who invested when you had no money were taking more risk, and are entitled to higher returns. Plus a company that has raised money is literally more valuable. After you raise the first million dollars, the company is at least a million dollars more valuable, because it's the same company as before, plus it has a million dollars in the bank. [1]

Beware, though, because later investors so hate to have the price raised on them that they resist even this self-evident reasoning. Only raise the price on an investor you're comfortable with losing, because some will angrily refuse. [2]

The second reason investors like you more when you've had some success at fundraising is that it makes you more confident, and an investors' opinion of [you](#) is the foundation of their opinion of your company. Founders are often surprised how quickly investors seem to know when they start to succeed at raising money. And while there are in fact lots of ways for such information to spread among investors, the main vector is probably the founders themselves. Though they're often clueless about technology, most investors are pretty good at reading people. When fundraising is going well, investors are quick to sense it in your increased confidence. (This is one case where the average

founder's inability to remain poker-faced works to your advantage.)

But frankly the most important reason investors like you more when you've started to raise money is that they're bad at judging startups. Judging startups is hard even for the best investors. The mediocre ones might as well be flipping coins. So when mediocre investors see that lots of other people want to invest in you, they assume there must be a reason. This leads to the phenomenon known in the Valley as the "hot deal," where you have more interest from investors than you can handle.

The best investors aren't influenced much by the opinion of other investors. It would only dilute their own judgment to average it together with other people's. But they are indirectly influenced in the practical sense that interest from other investors imposes a deadline. This is the fourth way in which offers beget offers. If you start to get far along the track toward an offer with one firm, it will sometimes provoke other firms, even good ones, to make up their minds, lest they lose the deal.

Unless you're a wizard at negotiation (and if you're not sure, you're not) be very careful about exaggerating this to push a good investor to decide. Founders try this sort of thing all the time, and investors are very sensitive to it. If anything oversensitive. But you're safe so long as you're telling the truth. If you're getting far along with investor B, but you'd rather raise money from investor A, you can tell investor A that this is happening. There's no manipulation in that. You're genuinely in a bind, because you really would rather raise money from A, but you can't safely reject an offer from B when it's still uncertain what A will decide.

Do not, however, tell A who B is. VCs will sometimes ask which other VCs you're talking to, but you should never tell them. Angels you can sometimes tell about other angels, because angels cooperate more with one another. But if VCs ask, just point out that they wouldn't want you telling other firms about your conversations, and you feel obliged to do the same for any firm you talk to. If they push you, point out that you're inexperienced at fundraising — which is always a safe card to play — and you feel you have to be extra cautious. [3]

While few startups will experience a stampede of interest, almost all will at least initially experience the other side of this phenomenon, where the herd remains clumped together at a distance. The fact that investors are so much influenced by other investors' opinions means you always start out in something of a hole. So don't be demoralized by how hard it is to get the first commitment, because much of the difficulty comes from this external force. The second will be easier.

Notes

[1] An accountant might say that a company that has raised a million dollars is no richer if it's convertible debt, but in practice money raised as convertible debt is little different from money raised in an equity round.

[2] Founders are often surprised by this, but investors can get very emotional. Or rather indignant; that's the main emotion I've observed; but it is very common, to the point where it sometimes causes investors to act against their own interests. I know of one investor who invested in a startup at a \$15 million valuation cap. Earlier he'd had an opportunity to invest at a \$5 million cap, but he refused because a friend who invested earlier had been able to invest at a \$3 million cap.

[3] If an investor pushes you hard to tell them about your conversations with other investors, is this someone you want as an investor?

Thanks to Paul Buchheit, Jessica Livingston, Geoff Ralston, and Garry Tan for reading drafts of this.

- [Russian Translation](#)

How to Raise Money

September 2013

Most startups that raise money do it more than once. A typical trajectory might be (1) to get started with a few tens of thousands from something like Y Combinator or individual angels, then (2) raise a few hundred thousand to a few million to build the company, and then (3) once the company is clearly succeeding, raise one or more later rounds to accelerate growth.

Reality can be messier. Some companies raise money twice in phase 2. Others skip phase 1 and go straight to phase 2. And at Y Combinator we get an increasing number of companies that have already raised amounts in the hundreds of thousands. But the three phase path is at least the one about which individual startups' paths oscillate.

This essay focuses on phase 2 fundraising. That's the type the startups we fund are doing on Demo Day, and this essay is the advice we give them.

Forces

Fundraising is hard in both senses: hard like lifting a heavy weight, and hard like solving a puzzle. It's hard like lifting a weight because it's intrinsically hard to convince people to part with large sums of money. That problem is irreducible; it should be hard. But much of the other kind of difficulty can be eliminated. Fundraising only seems a puzzle because it's an alien world to most founders, and I hope to fix that by supplying a map through it.

To founders, the behavior of investors is often opaque — partly because their motivations are obscure, but partly because they deliberately mislead you. And the misleading ways of investors combine horribly with the wishful thinking of inexperienced founders. At YC we're always warning founders about this danger, and investors are probably more circumspect with YC startups than with other companies they talk to, and even so we witness a constant series of explosions as these two volatile components combine. [1]

If you're an inexperienced founder, the only way to survive is by imposing external constraints on yourself. You can't trust your intuitions. I'm going to give you a set of rules here that will get you through this process if anything will. At certain moments you'll be tempted to ignore them. So rule number zero is: these rules exist for a reason. You wouldn't need a rule to keep you going in one direction if there weren't powerful forces pushing you in another.

The ultimate source of the forces acting on you are the forces acting on investors.

Investors are pinched between two kinds of fear: fear of investing in startups that fizzle, and fear of missing out on startups that take off. The cause of all this fear is the very thing that makes startups such attractive investments: the successful ones grow very fast. But that fast growth means investors can't wait around. If you wait till a startup is obviously a success, it's too late. To get the really high returns, you have to invest in startups when it's still unclear how they'll do. But that in turn makes investors nervous they're about to invest in a flop. As indeed they often are.

What investors would like to do, if they could, is wait. When a startup is only a few months old, every week that passes gives you significantly more information about them. But if you wait too long, other investors might take the deal away from you. And of course the other investors are all subject to the same forces. So what tends to happen is that they all wait as long as they can, then when some act the rest have to.

Don't raise money unless you want it and it wants you.

Such a high proportion of successful startups raise money that it might seem fundraising is one of the defining qualities of a startup. Actually it isn't. [Rapid growth](#) is what makes a company a startup. Most companies in a position to grow rapidly find that (a) taking outside money helps them grow faster, and (b) their growth potential makes it easy to attract such money. It's so common for both (a) and (b) to be true of a successful startup that practically all do raise outside money. But there may be cases where a startup either wouldn't want to grow faster, or outside money wouldn't help them to, and if you're one of them, don't raise money.

The other time not to raise money is when you won't be able to. If you try to raise money before you can [convince](#) investors, you'll not only waste your time, but also burn your reputation with those investors.

Be in fundraising mode or not.

One of the things that surprises founders most about fundraising is how distracting it is. When you start fundraising, everything else grinds to a halt. The problem is not the time fundraising consumes but that it becomes the [top idea in your mind](#). A startup can't endure that level of distraction for long. An early stage startup grows mostly because the founders [make](#) it grow, and if the founders look away, growth usually drops sharply.

Because fundraising is so distracting, a startup should either be in fundraising mode or not. And when you do decide to raise money, you should focus your whole attention on it so you can get it done quickly and get back to work. ^[2]

You can take money from investors when you're not in fundraising mode. You just can't expend any attention on it. There are two things that take attention: convincing investors, and negotiating with them. So when you're not in fundraising mode, you should take money from investors only if they require no convincing, and are willing to invest on terms you'll take without negotiation. For example, if a reputable investor is willing to invest on a convertible note, using standard paperwork, that is either uncapped or capped at a good valuation, you can take that without having to think. ^[3] The terms will be whatever they turn out to be in your next equity round. And "no convincing" means just that: zero time spent meeting with investors or preparing materials for them. If an investor says they're ready to invest, but they need you to come

in for one meeting to meet some of the partners, tell them no, if you're not in fundraising mode, because that's fundraising. [4] Tell them politely; tell them you're focusing on the company right now, and that you'll get back to them when you're fundraising; but do not get sucked down the slippery slope.

Investors will try to lure you into fundraising when you're not. It's great for them if they can, because they can thereby get a shot at you before everyone else. They'll send you emails saying they want to meet to learn more about you. If you get cold-emailed by an associate at a VC firm, you shouldn't meet even if you are in fundraising mode. Deals don't happen that way. [5] But even if you get an email from a partner you should try to delay meeting till you're in fundraising mode. They may say they just want to meet and chat, but investors never just want to meet and chat. What if they like you? What if they start to talk about giving you money? Will you be able to resist having that conversation? Unless you're experienced enough at fundraising to have a casual conversation with investors that stays casual, it's safer to tell them that you'd be happy to later, when you're fundraising, but that right now you need to focus on the company. [6]

Companies that are successful at raising money in phase 2 sometimes tack on a few investors after leaving fundraising mode. This is fine; if fundraising went well, you'll be able to do it without spending time convincing them or negotiating about terms.

Get introductions to investors.

Before you can talk to investors, you have to be introduced to them. If you're presenting at a Demo Day, you'll be introduced to a whole bunch simultaneously. But even if you are, you should supplement these with intros you collect yourself.

Do you have to be introduced? In phase 2, yes. Some investors will let you email them a business plan, but you can tell from the way their sites are organized that they don't really want startups to approach them directly.

Intros vary greatly in effectiveness. The best type of intro is from a well-known investor who has just invested in you. So when you get an investor to commit, ask them to introduce you to other investors they respect. [7] The next best type of intro is from a founder of a company they've funded. You can also get intros from other people in the startup community, like lawyers and reporters.

There are now sites like AngelList, FundersClub, and WeFunder that can introduce you to investors. We recommend startups treat them as auxiliary sources of money. Raise money first from leads you get yourself. Those will on average be better investors. Plus you'll have an easier time raising money on these sites once you can say you've already raised some from well-known investors.

Hear no till you hear yes.

Treat investors as saying no till they unequivocally say yes, in the form of a definite offer with no contingencies.

I mentioned earlier that investors prefer to wait if they can. What's particularly dangerous for founders is the way they wait. Essentially, they lead you on. They seem like they're about to invest right up till the moment they say no. If they even say no. Some of the worse ones never actually do say no; they just stop replying to your emails.

They hope that way to get a free option on investing. If they decide later that they want to invest — usually because they've heard you're a hot deal — they can pretend they just got distracted and then restart the conversation as if they'd been about to. [8]

That's not the worst thing investors will do. Some will use language that makes it sound as if they're committing, but which doesn't actually commit them. And wishful thinking founders are happy to meet them half way. [9]

Fortunately, the next rule is a tactic for neutralizing this behavior. But to work it depends on you not being tricked by the no that sounds like yes. It's so common for founders to be misled/mistaken about this that we designed a [protocol](#) to fix the problem. If you believe an investor has committed, get them to confirm it. If you and they have different views of reality, whether the source of the discrepancy is their sketchiness or your wishful thinking, the prospect of confirming a commitment in writing will flush it out. And till they confirm, regard them as saying no.

Do breadth-first search weighted by expected value.

When you talk to investors your m.o. should be breadth-first search, weighted by expected value. You should always talk to investors in parallel rather than serially. You can't afford the time it takes to talk to investors serially, plus if you only talk to one investor at a time, they don't have the pressure of other investors to make them act. But you shouldn't pay the same attention to every investor, because some are more promising prospects than others. The optimal solution is to talk to all potential investors in parallel, but give higher priority to the more promising ones. [10]

Expected value = how likely an investor is to say yes, multiplied by how good it would be if they did. So for example, an eminent investor who would invest a lot, but will be hard to convince, might have the same expected value as an obscure angel who won't invest much, but will be easy to convince. Whereas an obscure angel who will only invest a small amount, and yet needs to meet multiple times before making up his mind, has very low expected value. Meet such investors last, if at all. [11]

Doing breadth-first search weighted by expected value will save you from investors who never explicitly say no but merely drift away, because you'll drift away from them at the same rate. It protects you from investors who flake in much the same way that a distributed algorithm protects you from processors that fail. If some investor isn't returning your emails, or wants to have lots of meetings but isn't progressing toward making you an offer, you automatically focus less on them. But you have to be disciplined about assigning probabilities. You can't let how much you want an investor influence your estimate of how much they want you.

Know where you stand.

How do you judge how well you're doing with an investor, when investors habitually seem more positive than they are? By looking at their actions rather than their words. Every investor has some track they need to move along from the first conversation to wiring the money, and you should always know what that track consists of, where you are on it, and how fast you're moving forward.

Never leave a meeting with an investor without asking what happens next. What more do they need in order to decide? Do they need another meeting with you? To talk about

what? And how soon? Do they need to do something internally, like talk to their partners, or investigate some issue? How long do they expect it to take? Don't be too pushy, but know where you stand. If investors are vague or resist answering such questions, assume the worst; investors who are seriously interested in you will usually be happy to talk about what has to happen between now and wiring the money, because they're already running through that in their heads. [12]

If you're experienced at negotiations, you already know how to ask such questions. [13] If you're not, there's a trick you can use in this situation. Investors know you're inexperienced at raising money. Inexperience there doesn't make you unattractive. Being a noob at technology would, if you're starting a technology startup, but not being a noob at fundraising. Larry and Sergey were noobs at fundraising. So you can just confess that you're inexperienced at this and ask how their process works and where you are in it. [14]

Get the first commitment.

The biggest factor in most investors' opinions of you is the opinion of [other investors](#). Once you start getting investors to commit, it becomes increasingly easy to get more to. But the other side of this coin is that it's often hard to get the first commitment.

Getting the first substantial offer can be half the total difficulty of fundraising. What counts as a substantial offer depends on who it's from and how much it is. Money from friends and family doesn't usually count, no matter how much. But if you get \$50k from a well known VC firm or angel investor, that will usually be enough to set things rolling. [15]

Close committed money.

It's not a deal till the money's in the bank. I often hear inexperienced founders say things like "We've raised \$800,000," only to discover that zero of it is in the bank so far. Remember the twin fears that torment investors? The fear of missing out that makes them jump early, and the fear of jumping onto a turd that results? This is a market where people are exceptionally prone to buyer's remorse. And it's also one that furnishes them plenty of excuses to gratify it. The public markets snap startup investing around like a whip. If the Chinese economy blows up tomorrow, all bets are off. But there are lots of surprises for individual startups too, and they tend to be concentrated around fundraising. Tomorrow a big competitor could appear, or you could get C&Ded, or your cofounder could quit. [16]

Even a day's delay can bring news that causes an investor to change their mind. So when someone commits, get the money. Knowing where you stand doesn't end when they say they'll invest. After they say yes, know what the timetable is for getting the money, and then babysit that process till it happens. Institutional investors have people in charge of wiring money, but you may have to hunt angels down in person to collect a check.

Inexperienced investors are the ones most likely to get buyer's remorse. Established ones have learned to treat saying yes as like diving off a diving board, and they also have more brand to preserve. But I've heard of cases of even top-tier VC firms welching on deals.

Avoid investors who don't "lead."

Since getting the first offer is most of the difficulty of fundraising, that should be part of your calculation of expected value when you start. You have to estimate not just the probability that an investor will say yes, but the probability that they'd be the *first* to say yes, and the latter is not simply a constant fraction of the former. Some investors are known for deciding quickly, and those are extra valuable early on.

Conversely, an investor who will only invest once other investors have is worthless initially. And while most investors are influenced by how interested other investors are in you, there are some who have an explicit policy of only investing after other investors have. You can recognize this contemptible subspecies of investor because they often talk about "leads." They say that they don't lead, or that they'll invest once you have a lead. Sometimes they even claim to be willing to lead themselves, by which they mean they won't invest till you get \$x from other investors. (It's great if by "lead" they mean they'll invest unilaterally, and in addition will help you raise more. What's lame is when they use the term to mean they won't invest unless you can raise more elsewhere.) [\[17\]](#)

Where does this term "lead" come from? Up till a few years ago, startups raising money in phase 2 would usually raise equity rounds in which several investors invested at the same time using the same paperwork. You'd negotiate the terms with one "lead" investor, and then all the others would sign the same documents and all the money change hands at the closing.

Series A rounds still work that way, but things now work differently for most fundraising prior to the series A. Now there are rarely actual rounds before the A round, or leads for them. Now startups simply raise money from investors one at a time till they feel they have enough.

Since there are no longer leads, why do investors use that term? Because it's a more legitimate-sounding way of saying what they really mean. All they really mean is that their interest in you is a function of other investors' interest in you. I.e. the spectral signature of all mediocre investors. But when phrased in terms of leads, it sounds like there is something structural and therefore legitimate about their behavior.

When an investor tells you "I want to invest in you, but I don't lead," translate that in your mind to "No, except yes if you turn out to be a hot deal." And since that's the default opinion of any investor about any startup, they've essentially just told you nothing.

When you first start fundraising, the expected value of an investor who won't "lead" is zero, so talk to such investors last if at all.

Have multiple plans.

Many investors will ask how much you're planning to raise. This question makes founders feel they should be planning to raise a specific amount. But in fact you shouldn't. It's a mistake to have fixed plans in an undertaking as unpredictable as fundraising.

So why do investors ask how much you plan to raise? For much the same reasons a salesperson in a store will ask "How much were you planning to spend?" if you walk in

looking for a gift for a friend. You probably didn't have a precise amount in mind; you just want to find something good, and if it's inexpensive, so much the better. The salesperson asks you this not because you're supposed to have a plan to spend a specific amount, but so they can show you only things that cost the most you'll pay.

Similarly, when investors ask how much you plan to raise, it's not because you're supposed to have a plan. It's to see whether you'd be a suitable recipient for the size of investment they like to make, and also to judge your ambition, reasonableness, and how far you are along with fundraising.

If you're a wizard at fundraising, you can say "We plan to raise a \$7 million series A round, and we'll be accepting termsheets next tuesday." I've known a handful of founders who could pull that off without having VCs laugh in their faces. But if you're in the inexperienced but earnest majority, the solution is analogous to the solution I recommend for [pitching](#) your startup: do the right thing and then just tell investors what you're doing.

And the right strategy, in fundraising, is to have multiple plans depending on how much you can raise. Ideally you should be able to tell investors something like: we can make it to profitability without raising any more money, but if we raise a few hundred thousand we can hire one or two smart friends, and if we raise a couple million, we can hire a whole engineering team, etc.

Different plans match different investors. If you're talking to a VC firm that only does series A rounds (though there are few of those left), it would be a waste of time talking about any but your most expensive plan. Whereas if you're talking to an angel who invests \$20k at a time and you haven't raised any money yet, you probably want to focus on your least expensive plan.

If you're so fortunate as to have to think about the upper limit on what you should raise, a good rule of thumb is to multiply the number of people you want to hire times \$15k times 18 months. In most startups, nearly all the costs are a function of the number of people, and \$15k per month is the conventional total cost (including benefits and even office space) per person. \$15k per month is high, so don't actually spend that much. But it's ok to use a high estimate when fundraising to add a margin for error. If you have additional expenses, like manufacturing, add in those at the end. Assuming you have none and you think you might hire 20 people, the most you'd want to raise is $20 \times \$15k \times 18 = \5.4 million . [\[18\]](#)

Underestimate how much you want.

Though you can focus on different plans when talking to different types of investors, you should on the whole err on the side of underestimating the amount you hope to raise.

For example, if you'd like to raise \$500k, it's better to say initially that you're trying to raise \$250k. Then when you reach \$150k you're more than half done. That sends two useful signals to investors: that you're doing well, and that they have to decide quickly because you're running out of room. Whereas if you'd said you were raising \$500k, you'd be less than a third done at \$150k. If fundraising stalled there for an appreciable time, you'd start to read as a failure.

Saying initially that you're raising \$250k doesn't limit you to raising that much. When you reach your initial target and you still have investor interest, you can just decide to raise more. Startups do that all the time. In fact, most startups that are very successful at fundraising end up raising more than they originally intended.

I'm not saying you should lie, but that you should lower your expectations initially. There is almost no downside in starting with a low number. It not only won't cap the amount you raise, but will on the whole tend to increase it.

A good metaphor here is angle of attack. If you try to fly at too steep an angle of attack, you just stall. If you say right out of the gate that you want to raise a \$5 million series A round, unless you're in a very strong position, you not only won't get that but won't get anything. Better to start at a low angle of attack, build up speed, and then gradually increase the angle if you want.

Be profitable if you can.

You will be in a much stronger position if your collection of plans includes one for raising zero dollars — i.e. if you can make it to profitability without raising any additional money. Ideally you want to be able to say to investors "We'll succeed no matter what, but raising money will help us do it faster."

There are many analogies between fundraising and dating, and this is one of the strongest. No one wants you if you seem desperate. And the best way not to seem desperate is not to *be* desperate. That's one reason we urge startups during YC to keep expenses low and to try to make it to [ramen profitability](#) before Demo Day. Though it sounds slightly paradoxical, if you want to raise money, the best thing you can do is get yourself to the point where you don't need to.

There are almost two distinct modes of fundraising: one in which founders who need money knock on doors seeking it, knowing that otherwise the company will die or at the very least people will have to be fired, and one in which founders who don't need money take some to grow faster than they could merely on their own revenues. To emphasize the distinction I'm going to name them: type A fundraising is when you don't need money, and type B fundraising is when you do.

Inexperienced founders read about famous startups doing what was type A fundraising, and decide they should raise money too, since that seems to be how startups work. Except when they raise money they don't have a clear path to profitability and are thus doing type B fundraising. And they are then surprised how difficult and unpleasant it is.

Of course not all startups can make it to ramen profitability in a few months. And some that don't still manage to have the upper hand over investors, if they have some other advantage like extraordinary growth numbers or exceptionally formidable founders. But as time passes it gets increasingly difficult to fundraise from a position of strength without being profitable. [\[19\]](#)

Don't optimize for valuation.

When you raise money, what should your valuation be? The most important thing to understand about valuation is that it's not that important.

Founders who raise money at high valuations tend to be unduly proud of it. Founders are often competitive people, and since valuation is usually the only visible number attached to a startup, they end up competing to raise money at the highest valuation. This is stupid, because fundraising is not the test that matters. The real test is revenue. Fundraising is just a means to that end. Being proud of how well you did at fundraising is like being proud of your college grades.

Not only is fundraising not the test that matters, valuation is not even the thing to optimize about fundraising. The number one thing you want from phase 2 fundraising is to get the money you need, so you can get back to focusing on the real test, the success of your company. Number two is good investors. Valuation is at best third.

The empirical evidence shows just how unimportant it is. Dropbox and Airbnb are the most successful companies we've funded so far, and they raised money after Y Combinator at premoney valuations of \$4 million and \$2.6 million respectively. Prices are so much higher now that if you can raise money at all you'll probably raise it at higher valuations than Dropbox and Airbnb. So let that satisfy your competitiveness. You're doing better than Dropbox and Airbnb! At a test that doesn't matter.

When you start fundraising, your initial valuation (or valuation cap) will be set by the deal you make with the first investor who commits. You can increase the price for later investors, if you get a lot of interest, but by default the valuation you got from the first investor becomes your asking price.

So if you're raising money from multiple investors, as most companies do in phase 2, you have to be careful to avoid raising the first from an over-eager investor at a price you won't be able to sustain. You can of course lower your price if you need to (in which case you should give the same terms to investors who invested earlier at a higher price), but you may lose a bunch of leads in the process of realizing you need to do this.

What you can do if you have eager first investors is raise money from them on an uncapped convertible note with an MFN clause. This is essentially a way of saying that the valuation cap of the note will be determined by the next investors you raise money from.

It will be easier to raise money at a lower valuation. It shouldn't be, but it is. Since phase 2 prices vary at most 10x and the big successes generate returns of at least 100x, investors should pick startups entirely based on their estimate of the probability that the company will be a big success and hardly at all on price. But although it's a mistake for investors to care about price, a significant number do. A startup that investors seem to like but won't invest in at a cap of \$x will have an easier time at \$x/2. [\[20\]](#)

Yes/no before valuation.

Some investors want to know what your valuation is before they even talk to you about investing. If your valuation has already been set by a prior investment at a specific valuation or cap, you can tell them that number. But if it isn't set because you haven't closed anyone yet, and they try to push you to name a price, resist doing so. If this would be the first investor you've closed, then this could be the tipping point of fundraising. That means closing this investor is the first priority, and you need to get the conversation onto that instead of being dragged sideways into a discussion of price.

Fortunately there is a way to avoid naming a price in this situation. And it is not just a negotiating trick; it's how you (both) should be operating. Tell them that valuation is not the most important thing to you and that you haven't thought much about it, that you are looking for investors you want to partner with and who want to partner with you, and that you should talk first about whether they want to invest at all. Then if they decide they do want to invest, you can figure out a price. But first things first.

Since valuation isn't that important and getting fundraising rolling is, we usually tell founders to give the first investor who commits as low a price as they need to. This is a safe technique so long as you combine it with the next one. [\[21\]](#)

Beware "valuation sensitive" investors.

Occasionally you'll encounter investors who describe themselves as "valuation sensitive." What this means in practice is that they are compulsive negotiators who will suck up a lot of your time trying to push your price down. You should therefore never approach such investors first. While you shouldn't chase high valuations, you also don't want your valuation to be set artificially low because the first investor who committed happened to be a compulsive negotiator. Some such investors have value, but the time to approach them is near the end of fundraising, when you're in a position to say "this is the price everyone else has paid; take it or leave it" and not mind if they leave it. This way, you'll not only get market price, but it will also take less time.

Ideally you know which investors have a reputation for being "valuation sensitive" and can postpone dealing with them till last, but occasionally one you didn't know about will pop up early on. The rule of doing breadth first search weighted by expected value already tells you what to do in this case: slow down your interactions with them.

There are a handful of investors who will try to invest at a lower valuation even when your price has already been set. Lowering your price is a backup plan you resort to when you discover you've let the price get set too high to close all the money you need. So you'd only want to talk to this sort of investor if you were about to do that anyway. But since investor meetings have to be arranged at least a few days in advance and you can't predict when you'll need to resort to lowering your price, this means in practice that you should approach this type of investor last if at all.

If you're surprised by a lowball offer, treat it as a backup offer and delay responding to it. When someone makes an offer in good faith, you have a moral obligation to respond in a reasonable time. But lowballing you is a dick move that should be met with the corresponding countermove.

Accept offers greedily.

I'm a little leery of using the term "greedily" when writing about fundraising lest non-programmers misunderstand me, but a greedy algorithm is simply one that doesn't try to look into the future. A greedy algorithm takes the best of the options in front of it right now. And that is how startups should approach fundraising in phases 2 and later. Don't try to look into the future because (a) the future is unpredictable, and indeed in this business you're often being deliberately misled about it and (b) your first priority in fundraising should be to get it finished and get back to work anyway.

If someone makes you an acceptable offer, take it. If you have multiple incompatible

offers, take the best. Don't reject an acceptable offer in the hope of getting a better one in the future.

These simple rules cover a wide variety of cases. If you're raising money from many investors, roll them up as they say yes. As you start to feel you've raised enough, the threshold for acceptable will start to get higher.

In practice offers exist for stretches of time, not points. So when you get an acceptable offer that would be incompatible with others (e.g. an offer to invest most of the money you need), you can tell the other investors you're talking to that you have an offer good enough to accept, and give them a few days to make their own. This could lose you some that might have made an offer if they had more time. But by definition you don't care; the initial offer was acceptable.

Some investors will try to prevent others from having time to decide by giving you an "exploding" offer, meaning one that's only valid for a few days. Offers from the very best investors explode less frequently and less rapidly — Fred Wilson never gives exploding offers, for example — because they're confident you'll pick them. But lower-tier investors sometimes give offers with very short fuses, because they believe no one who had other options would choose them. A deadline of three working days is acceptable. You shouldn't need more than that if you've been talking to investors in parallel. But a deadline any shorter is a sign you're dealing with a sketchy investor. You can usually call their bluff, and you may need to. [\[22\]](#)

It might seem that instead of accepting offers greedily, your goal should be to get the best investors as partners. That is certainly a good goal, but in phase 2 "get the best investors" only rarely conflicts with "accept offers greedily," because the best investors don't usually take any longer to decide than the others. The only case where the two strategies give conflicting advice is when you have to forgo an offer from an acceptable investor to see if you'll get an offer from a better one. If you talk to investors in parallel and push back on exploding offers with excessively short deadlines, that will almost never happen. But if it does, "get the best investors" is in the average case bad advice. The best investors are also the most selective, because they get their pick of all the startups. They reject nearly everyone they talk to, which means in the average case it's a bad trade to exchange a definite offer from an acceptable investor for a potential offer from a better one.

(The situation is different in phase 1. You can't apply to all the incubators in parallel, because some offset their schedules to prevent this. In phase 1, "accept offers greedily" and "get the best investors" do conflict, so if you want to apply to multiple incubators, you should do it in such a way that the ones you want most decide first.)

Sometimes when you're raising money from multiple investors, a series A will emerge out of those conversations, and these rules even cover what to do in that case. When an investor starts to talk to you about a series A, keep taking smaller investments till they actually give you a termsheet. There's no practical difficulty. If the smaller investments are on convertible notes, they'll just convert into the series A round. The series A investor won't like having all these other random investors as bedfellows, but if it bothers them so much they should get on with giving you a termsheet. Till they do, you don't know for sure they will, and the greedy algorithm tells you what to do. [\[23\]](#)

Don't sell more than 25% in phase 2.

If you do well, you will probably raise a series A round eventually. I say probably because things are changing with series A rounds. Startups may start to skip them. But only one company we've funded has so far, so tentatively assume the path to huge passes through an A round. [\[24\]](#)

Which means you should avoid doing things in earlier rounds that will mess up raising an A round. For example, if you've sold more than about 40% of your company total, it starts to get harder to raise an A round, because VCs worry there will not be enough stock left to keep the founders motivated.

Our rule of thumb is not to sell more than 25% in phase 2, on top of whatever you sold in phase 1, which should be less than 15%. If you're raising money on uncapped notes, you'll have to guess what the eventual equity round valuation might be. Guess conservatively.

(Since the goal of this rule is to avoid messing up the series A, there's obviously an exception if you end up raising a series A in phase 2, as a handful of startups do.)

Have one person handle fundraising.

If you have multiple founders, pick one to handle fundraising so the other(s) can keep working on the company. And since the danger of fundraising is not the time taken up by the actual meetings but that it becomes the top idea in your mind, the founder who handles fundraising should make a conscious effort to insulate the other founder(s) from the details of the process. [\[25\]](#)

(If the founders mistrust one another, this could cause some friction. But if the founders mistrust one another, you have worse problems to worry about than how to organize fundraising.)

The founder who handles fundraising should be the CEO, who should in turn be the most formidable of the founders. Even if the CEO is a programmer and another founder is a salesperson? Yes. If you happen to be that type of founding team, you're effectively a single founder when it comes to fundraising.

It's ok to bring all the founders to meet an investor who will invest a lot, and who needs this meeting as the final step before deciding. But wait till that point. Introducing an investor to your cofounder(s) should be like introducing a girl/boyfriend to your parents — something you do only when things reach a certain stage of seriousness.

Even if there are still one or more founders focusing on the company during fundraising, growth will slow. But try to get as much growth as you can, because fundraising is a segment of time, not a point, and what happens to the company during that time affects the outcome. If your numbers grow significantly between two investor meetings, investors will be hot to close, and if your numbers are flat or down they'll start to get cold feet.

You'll need an executive summary and (maybe) a deck.

Traditionally phase 2 fundraising consists of presenting a slide deck in person to investors. Sequoia describes what such a deck should [contain](#), and since they're the

customer you can take their word for it.

I say "traditionally" because I'm ambivalent about decks, and (though perhaps this is wishful thinking) they seem to be on the way out. A lot of the most successful startups we fund never make decks in phase 2. They just talk to investors and explain what they plan to do. Fundraising usually takes off fast for the startups that are most successful at it, and they're thus able to excuse themselves by saying that they haven't had time to make a deck.

You'll also want an executive summary, which should be no more than a page long and describe in the most matter of fact language what you plan to do, why it's a good idea, and what progress you've made so far. The point of the summary is to remind the investor (who may have met many startups that day) what you talked about.

Assume that if you give someone a copy of your deck or executive summary, it will be passed on to whoever you'd least like to have it. But don't refuse on that account to give copies to investors you meet. You just have to treat such leaks as a cost of doing business. In practice it's not that high a cost. Though founders are rightly indignant when their plans get leaked to competitors, I can't think of a startup whose outcome has been affected by it.

Sometimes an investor will ask you to send them your deck and/or executive summary before they decide whether to meet with you. I wouldn't do that. It's a sign they're not really interested.

Stop fundraising when it stops working.

When do you stop fundraising? Ideally when you've raised enough. But what if you haven't raised as much as you'd like? When do you give up?

It's hard to give general advice about this, because there have been cases of startups that kept trying to raise money even when it seemed hopeless, and miraculously succeeded. But what I usually tell founders is to stop fundraising when you start to get a lot of air in the straw. When you're drinking through a straw, you can tell when you get to the end of the liquid because you start to get a lot of air in the straw. When your fundraising options run out, they usually run out in the same way. Don't keep sucking on the straw if you're just getting air. It's not going to get better.

Don't get addicted to fundraising.

Fundraising is a chore for most founders, but some find it more interesting than working on their startup. The work at an early stage startup often consists of unglamorous [schleps](#). Whereas fundraising, when it's going well, can be quite the opposite. Instead of sitting in your grubby apartment listening to users complain about bugs in your software, you're being offered millions of dollars by famous investors over lunch at a nice restaurant. [\[26\]](#)

The danger of fundraising is particularly acute for people who are good at it. It's always fun to work on something you're good at. If you're one of these people, beware. Fundraising is not what will make your company successful. Listening to users complain about bugs in your software is what will make you successful. And the big danger of getting addicted to fundraising is not merely that you'll spend too long on it or raise too

much money. It's that you'll start to think of yourself as being already successful, and lose your taste for the schleps you need to undertake to actually be successful. Startups can be destroyed by this.

When I see a startup with young founders that is fabulously successful at fundraising, I mentally decrease my estimate of the probability that they'll succeed. The press may be writing about them as if they'd been anointed as the next Google, but I'm thinking "this is going to end badly."

Don't raise too much.

Though only a handful of startups have to worry about this, it is possible to raise too much. The dangers of raising too much are subtle but insidious. One is that it will set impossibly high expectations. If you raise an excessive amount of money, it will be at a high valuation, and the danger of raising money at too high a valuation is that you won't be able to increase it sufficiently the next time you raise money.

A company's valuation is expected to rise each time it raises money. If not it's a sign of a company in trouble, which makes you unattractive to investors. So if you raise money in phase 2 at a post-money valuation of \$30 million, the pre-money valuation of your next round, if you want to raise one, is going to have to be at least \$50 million. And you have to be doing really, really well to raise money at \$50 million.

It's very dangerous to let the competitiveness of your current round set the performance threshold you have to meet to raise your next one, because the two are only loosely coupled.

But the money itself may be more dangerous than the valuation. The more you raise, the more you spend, and spending a lot of money can be disastrous for an early stage startup. Spending a lot makes it harder to become profitable, and perhaps even worse, it makes you more rigid, because the main way to spend money is people, and the more people you have, the harder it is to change directions. So if you do raise a huge amount of money, don't spend it. (You will find that advice almost impossible to follow, so hot will be the money burning a hole in your pocket, but I feel obliged at least to try.)

Be nice.

Startups raising money occasionally alienate investors by seeming arrogant. Sometimes because they are arrogant, and sometimes because they're noobs clumsily attempting to mimic the toughness they've observed in experienced founders.

It's a mistake to behave arrogantly to investors. While there are certain situations in which certain investors like certain kinds of arrogance, investors vary greatly in this respect, and a flick of the whip that will bring one to heel will make another roar with indignation. The only safe strategy is never to seem arrogant at all.

That will require some diplomacy if you follow the advice I've given here, because the advice I've given is essentially how to play hardball back. When you refuse to meet an investor because you're not in fundraising mode, or slow down your interactions with an investor who moves too slow, or treat a contingent offer as the no it actually is and then, by accepting offers greedily, end up leaving that investor out, you're going to be doing things investors don't like. So you must cushion the blow with soft words. At YC we tell

startups they can blame us. And now that I've written this, everyone else can blame me if they want. That plus the inexperience card should work in most situations: sorry, we think you're great, but PG said startups shouldn't ____, and since we're new to fundraising, we feel like we have to play it safe.

The danger of behaving arrogantly is greatest when you're doing well. When everyone wants you, it's hard not to let it go to your head. Especially if till recently no one wanted you. But restrain yourself. The startup world is a small place, and startups have lots of ups and downs. This is a domain where it's more true than usual that pride goeth before a fall. [\[27\]](#)

Be nice when investors reject you as well. The best investors are not wedded to their initial opinion of you. If they reject you in phase 2 and you end up doing well, they'll often invest in phase 3. In fact investors who reject you are some of your warmest leads for future fundraising. Any investor who spent significant time deciding probably came close to saying yes. Often you have some internal champion who only needs a little more evidence to convince the skeptics. So it's wise not merely to be nice to investors who reject you, but (unless they behaved badly) to treat it as the beginning of a relationship.

The bar will be higher next time.

Assume the money you raise in phase 2 will be the last you ever raise. You must make it to profitability on this money if you can.

Over the past several years, the investment community has evolved from a strategy of anointing a small number of winners early and then supporting them for years to a strategy of spraying money at early stage startups and then ruthlessly culling them at the next stage. This is probably the optimal strategy for investors. It's too hard to pick winners early on. Better to let the market do it for you. But it often comes as a surprise to startups how much harder it is to raise money in phase 3.

When your company is only a couple months old, all it has to be is a promising experiment that's worth funding to see how it turns out. The next time you raise money, the experiment has to have worked. You have to be on a trajectory that leads to going public. And while there are some ideas where the proof that the experiment worked might consist of e.g. query response times, usually the proof is profitability. Usually phase 3 fundraising has to be type A fundraising.

In practice there are two ways startups hose themselves between phases 2 and 3. Some are just too slow to become profitable. They raise enough money to last for two years. There doesn't seem any particular urgency to be profitable. So they don't make any effort to make money for a year. But by that time, not making money has become habitual. When they finally decide to try, they find they can't.

The other way companies hose themselves is by letting their expenses grow too fast. Which almost always means hiring too many people. You usually shouldn't go out and hire 8 people as soon as you raise money at phase 2. Usually you want to wait till you have growth (and thus usually revenues) to justify them. A lot of VCs will encourage you to hire aggressively. VCs generally tell you to spend too much, partly because as money people they err on the side of solving problems by spending money, and partly because they want you to sell them more of your company in subsequent rounds. Don't

listen to them.

Don't make things complicated.

I realize it may seem odd to sum up this huge treatise by saying that my overall advice is not to make fundraising too complicated, but if you go back and look at this list you'll see it's basically a simple recipe with a lot of implications and edge cases. Avoid investors till you decide to raise money, and then when you do, talk to them all in parallel, prioritized by expected value, and accept offers greedily. That's fundraising in one sentence. Don't introduce complicated optimizations, and don't let investors introduce complications either.

Fundraising is not what will make you successful. It's just a means to an end. Your primary goal should be to get it over with and get back to what will make you successful — making things and talking to users — and the path I've described will for most startups be the surest way to that destination.

Be good, take care of yourselves, and *don't leave the path*.

Notes

[1] The worst explosions happen when unpromising-seeming startups encounter mediocre investors. Good investors don't lead startups on; their reputations are too valuable. And startups that seem promising can usually get enough money from good investors that they don't have to talk to mediocre ones. It is the unpromising-seeming startups that have to resort to raising money from mediocre investors. And it's particularly damaging when these investors flake, because unpromising-seeming startups are usually more desperate for money.

(Not all unpromising-seeming startups do badly. Some are merely ugly ducklings in the sense that they violate current startup fashions.)

[2] One YC founder told me:

I think in general we've done ok at fundraising, but I managed to screw up twice at the exact same thing — trying to focus on building the company and fundraising at the same time.

[3] There is one subtle danger you have to watch out for here, which I warn about later: beware of getting too high a valuation from an eager investor, lest that set an impossibly high target when raising additional money.

[4] If they really need a meeting, then they're not ready to invest, regardless of what they say. They're still deciding, which means you're being asked to come in and convince them. Which is fundraising.

[5] Associates at VC firms regularly cold email startups. Naive founders think "Wow, a VC is interested in us!" But an associate is not a VC. They have no decision-making

power. And while they may introduce startups they like to partners at their firm, the partners discriminate against deals that come to them this way. I don't know of a single VC investment that began with an associate cold-emailing a startup. If you want to approach a specific firm, get an intro to a partner from someone they respect.

It's ok to talk to an associate if you get an intro to a VC firm or they see you at a Demo Day and they begin by having an associate vet you. That's not a promising lead and should therefore get low priority, but it's not as completely worthless as a cold email.

Because the title "associate" has gotten a bad reputation, a few VC firms have started to give their associates the title "partner," which can make things very confusing. If you're a YC startup you can ask us who's who; otherwise you may have to do some research online. There may be a special title for actual partners. If someone speaks for the firm in the press or a blog on the firm's site, they're probably a real partner. If they're on boards of directors they're probably a real partner.

There are titles between "associate" and "partner," including "principal" and "venture partner." The meanings of these titles vary too much to generalize.

[6] For similar reasons, avoid casual conversations with potential acquirers. They can lead to distractions even more dangerous than fundraising. Don't even take a meeting with a potential acquirer unless you want to sell your company right now.

[7] Joshua Reeves specifically suggests asking each investor to intro you to two more investors.

Don't ask investors who say no for introductions to other investors. That will in many cases be an anti-recommendation.

[8] This is not always as deliberate as it sounds. A lot of the delays and disconnects between founders and investors are induced by the customs of the venture business, which have evolved the way they have because they suit investors' interests.

[9] One YC founder who read a draft of this essay wrote:

This is the most important section. I think it might bear stating even more clearly. "Investors will deliberately affect more interest than they have to preserve optionality. If an investor seems very interested in you, they still probably won't invest. The solution for this is to assume the worst — that an investor is just feigning interest — until you get a definite commitment."

[10] Though you should probably pack investor meetings as closely as you can, Jeff Byun mentions one reason not to: if you pack investor meetings too closely, you'll have less time for your pitch to evolve.

Some founders deliberately schedule a handful of lame investors first, to get the bugs out of their pitch.

[11] There is not an efficient market in this respect. Some of the most useless investors are also the highest maintenance.

[12] Incidentally, this paragraph is sales 101. If you want to see it in action, go talk to a

car dealer.

[13] I know one very smooth founder who used to end investor meetings with "So, can I count you in?" delivered as if it were "Can you pass the salt?" Unless you're very smooth (if you're not sure...), do not do this yourself. There is nothing more unconvincing, for an investor, than a nerdy founder trying to deliver the lines meant for a smooth one.

Investors are fine with funding nerds. So if you're a nerd, just try to be a good nerd, rather than doing a bad imitation of a smooth salesman.

[14] Ian Hogarth suggests a good way to tell how serious potential investors are: the resources they expend on you after the first meeting. An investor who's seriously interested will already be working to help you even before they've committed.

[15] In principle you might have to think about so-called "signalling risk." If a prestigious VC makes a small seed investment in you, what if they don't want to invest the next time you raise money? Other investors might assume that the VC knows you well, since they're an existing investor, and if they don't want to invest in your next round, that must mean you suck. The reason I say "in principle" is that in practice signalling hasn't been much of a problem so far. It rarely arises, and in the few cases where it does, the startup in question usually is doing badly and is doomed anyway.

If you have the luxury of choosing among seed investors, you can play it safe by excluding VC firms. But it isn't critical to.

[16] Sometimes a competitor will deliberately threaten you with a lawsuit just as you start fundraising, because they know you'll have to disclose the threat to potential investors and they hope this will make it harder for you to raise money. If this happens it will probably frighten you more than investors. Experienced investors know about this trick, and know the actual lawsuits rarely happen. So if you're attacked in this way, be forthright with investors. They'll be more alarmed if you seem evasive than if you tell them everything.

[17] A related trick is to claim that they'll only invest contingently on other investors doing so because otherwise you'd be "undercapitalized." This is almost always bullshit. They can't estimate your minimum capital needs that precisely.

[18] You won't hire all those 20 people at once, and you'll probably have some revenues before 18 months are out. But those too are acceptable or at least accepted additions to the margin for error.

[19] Type A fundraising is so much better that it might even be worth doing something different if it gets you there sooner. One YC founder told me that if he were a first-time founder again he'd "leave ideas that are up-front capital intensive to founders with established reputations."

[20] I don't know whether this happens because they're innumerate, or because they believe they have zero ability to predict startup outcomes (in which case this behavior at least wouldn't be irrational). In either case the implications are similar.

[21] If you're a YC startup and you have an investor who for some reason insists that

you decide the price, any YC partner can estimate a market price for you.

[22] You should respond in kind when investors behave upstandingly too. When an investor makes you a clean offer with no deadline, you have a moral obligation to respond promptly.

[23] Tell the investors talking to you about an A round about the smaller investments you raise as you raise them. You owe them such updates on your cap table, and this is also a good way to pressure them to act. They won't like you raising other money and may pressure you to stop, but they can't legitimately ask you to commit to them till they also commit to you. If they want you to stop raising money, the way to do it is to give you a series A termsheet with a no-shop clause.

You can relent a little if the potential series A investor has a great reputation and they're clearly working fast to get you a termsheet, particularly if a third party like YC is involved to ensure there are no misunderstandings. But be careful.

[24] The company is Weebly, which made it to profitability on a seed investment of \$650k. They did try to raise a series A in the fall of 2008 but (no doubt partly because it was the fall of 2008) the terms they were offered were so bad that they decided to skip raising an A round.

[25] Another advantage of having one founder take fundraising meetings is that you never have to negotiate in real time, which is something inexperienced founders should avoid. One YC founder told me:

Investors are professional negotiators and can negotiate on the spot very easily. If only one founder is in the room, you can say "I need to circle back with my co-founder" before making any commitments. I used to do this all the time.

[26] You'll be lucky if fundraising feels pleasant enough to become addictive. More often you have to worry about the other extreme — becoming demoralized when investors reject you. As one (very successful) YC founder wrote after reading a draft of this:

It's hard to mentally deal with the sheer scale of rejection in fundraising and if you are not in the right mindset you will fail. Users may love you but these supposedly smart investors may not understand you at all. At this point for me, rejection still rankles but I've come to accept that investors are just not super thoughtful for the most part and you need to play the game according to certain somewhat depressing rules (many of which you are listing) in order to win.

[27] The actual sentence in the King James Bible is "Pride goeth before destruction, and an haughty spirit before a fall."

Thanks to Slava Akhmechet, Sam Altman, Nate Blecharczyk, Adora Cheung, Bill Clerico, John Collison, Patrick Collison, Parker Conrad, Ron Conway, Travis Deyle, Jason Freedman, Joe Gebbia, Mattan Griffel, Kevin Hale, Jacob Heller, Ian Hogarth, Justin Kan, Professor Moriarty, Nikhil Nirmel, David Petersen, Geoff Ralston, Joshua Reeves, Yuri Sagalov, Emmett Shear, Rajat Suri, Garry Tan, and Nick Tomarelli for

reading drafts of this.

- [Russian Translation](#)

Before the Startup

October 2014

(This essay is derived from a guest lecture in Sam Altman's [startup class](#) at Stanford. It's intended for college students, but much of it is applicable to potential founders at other ages.)

One of the advantages of having kids is that when you have to give advice, you can ask yourself "what would I tell my own kids?" My kids are little, but I can imagine what I'd tell them about startups if they were in college, and that's what I'm going to tell you.

Startups are very counterintuitive. I'm not sure why. Maybe it's just because knowledge about them hasn't permeated our culture yet. But whatever the reason, starting a startup is a task where you can't always trust your instincts.

It's like skiing in that way. When you first try skiing and you want to slow down, your instinct is to lean back. But if you lean back on skis you fly down the hill out of control. So part of learning to ski is learning to suppress that impulse. Eventually you get new habits, but at first it takes a conscious effort. At first there's a list of things you're trying to remember as you start down the hill.

Startups are as unnatural as skiing, so there's a similar list for startups. Here I'm going to give you the first part of it — the things to remember if you want to prepare yourself to start a startup.

Counterintuitive

The first item on it is the fact I already mentioned: that startups are so weird that if you trust your instincts, you'll make a lot of mistakes. If you know nothing more than this, you may at least pause before making them.

When I was running Y Combinator I used to joke that our function was to tell founders things they would ignore. It's really true. Batch after batch, the YC partners warn founders about mistakes they're about to make, and the founders ignore them, and then come back a year later and say "I wish we'd listened."

Why do the founders ignore the partners' advice? Well, that's the thing about counterintuitive ideas: they contradict your intuitions. They seem wrong. So of course your first impulse is to disregard them. And in fact my joking description is not merely the curse of Y Combinator but part of its *raison d'être*. If founders' instincts already gave them the right answers, they wouldn't need us. You only need other people to give you advice that surprises you. That's why there are a lot of ski instructors and not many running instructors. [\[1\]](#)

You can, however, trust your instincts about people. And in fact one of the most common mistakes young founders make is not to do that enough. They get involved with people who seem impressive, but about whom they feel some misgivings personally. Later when things blow up they say "I knew there was something off about him, but I ignored it because he seemed so impressive."

If you're thinking about getting involved with someone — as a cofounder, an employee, an investor, or an acquirer — and you have misgivings about them, trust your gut. If someone seems slippery, or bogus, or a jerk, don't ignore it.

This is one case where it pays to be self-indulgent. Work with people you genuinely like, and you've known long enough to be sure.

Expertise

The second counterintuitive point is that it's not that important to know a lot about startups. The way to succeed in a startup is not to be an expert on startups, but to be an expert on your users and the problem you're solving for them. Mark Zuckerberg didn't succeed because he was an expert on startups. He succeeded despite being a complete noob at startups, because he understood his users really well.

If you don't know anything about, say, how to raise an angel round, don't feel bad on that account. That sort of thing you can learn when you need to, and forget after you've done it.

In fact, I worry it's not merely unnecessary to learn in great detail about the mechanics of startups, but possibly somewhat dangerous. If I met an undergrad who knew all about convertible notes and employee agreements and (God forbid) class FF stock, I wouldn't think "here is someone who is way ahead of their peers." It would set off alarms. Because another of the characteristic mistakes of young founders is to go through the motions of starting a startup. They make up some plausible-sounding idea, raise money at a good valuation, rent a cool office, hire a bunch of people. From the outside that seems like what startups do. But the next step after rent a cool office and hire a bunch of people is: gradually realize how completely fucked they are, because while imitating all the outward forms of a startup they have neglected the one thing that's actually essential: making something people want.

Game

We saw this happen so often that we made up a name for it: playing house. Eventually I realized why it was happening. The reason young founders go through the motions of starting a startup is because that's what they've been trained to do for their whole lives up to that point. Think about what you have to do to get into college, for example. Extracurricular activities, check. Even in college classes most of the work is as artificial as running laps.

I'm not attacking the educational system for being this way. There will always be a certain amount of fakeness in the work you do when you're being taught something, and if you measure their performance it's inevitable that people will exploit the difference to the point where much of what you're measuring is artifacts of the fakeness.

I confess I did it myself in college. I found that in a lot of classes there might only be 20 or 30 ideas that were the right shape to make good exam questions. The way I studied for exams in these classes was not (except incidentally) to master the material taught in the class, but to make a list of potential exam questions and work out the answers in advance. When I walked into the final, the main thing I'd be feeling was curiosity about which of my questions would turn up on the exam. It was like a game.

It's not surprising that after being trained for their whole lives to play such games, young founders' first impulse on starting a startup is to try to figure out the tricks for winning at this new game. Since fundraising appears to be the measure of success for startups (another classic noob mistake), they always want to know what the tricks are for convincing investors. We tell them the best way to [convince investors](#) is to make a startup that's actually doing well, meaning [growing fast](#), and then simply tell investors so. Then they want to know what the tricks are for growing fast. And we have to tell them the best way to do that is simply to make something people want.

So many of the conversations YC partners have with young founders begin with the founder asking "How do we..." and the partner replying "Just..."

Why do the founders always make things so complicated? The reason, I realized, is that they're looking for the trick.

So this is the third counterintuitive thing to remember about startups: starting a startup is where gaming the system stops working. Gaming the system may continue to work if you go to work for a big company. Depending on how broken the company is, you can succeed by sucking up to the right people, giving the impression of productivity, and so on. [2] But that doesn't work with startups. There is no boss to trick, only users, and all users care about is whether your product does what they want. Startups are as impersonal as physics. You have to make something people want, and you prosper only to the extent you do.

The dangerous thing is, faking does work to some degree on investors. If you're super good at sounding like you know what you're talking about, you can fool investors for at least one and perhaps even two rounds of funding. But it's not in your interest to. The company is ultimately doomed. All you're doing is wasting your own time riding it down.

So stop looking for the trick. There are tricks in startups, as there are in any domain, but they are an order of magnitude less important than solving the real problem. A founder who knows nothing about fundraising but has made something users love will have an easier time raising money than one who knows every trick in the book but has a flat usage graph. And more importantly, the founder who has made something users love is the one who will go on to succeed after raising the money.

Though in a sense it's bad news in that you're deprived of one of your most powerful weapons, I think it's exciting that gaming the system stops working when you start a startup. It's exciting that there even exist parts of the world where you win by doing good work. Imagine how depressing the world would be if it were all like school and big companies, where you either have to spend a lot of time on bullshit things or lose to people who do. [3] I would have been delighted if I'd realized in college that there were parts of the real world where gaming the system mattered less than others, and a few where it hardly mattered at all. But there are, and this variation is one of the most

important things to consider when you're thinking about your future. How do you win in each type of work, and what would you like to win by doing? [4]

All-Consuming

That brings us to our fourth counterintuitive point: startups are all-consuming. If you start a startup, it will take over your life to a degree you cannot imagine. And if your startup succeeds, it will take over your life for a long time: for several years at the very least, maybe for a decade, maybe for the rest of your working life. So there is a real opportunity cost here.

Larry Page may seem to have an enviable life, but there are aspects of it that are unenviable. Basically at 25 he started running as fast as he could and it must seem to him that he hasn't stopped to catch his breath since. Every day new shit happens in the Google empire that only the CEO can deal with, and he, as CEO, has to deal with it. If he goes on vacation for even a week, a whole week's backlog of shit accumulates. And he has to bear this uncomplainingly, partly because as the company's daddy he can never show fear or weakness, and partly because billionaires get less than zero sympathy if they talk about having difficult lives. Which has the strange side effect that the difficulty of being a successful startup founder is concealed from almost everyone except those who've done it.

Y Combinator has now funded several companies that can be called big successes, and in every single case the founders say the same thing. It never gets any easier. The nature of the problems change. You're worrying about construction delays at your London office instead of the broken air conditioner in your studio apartment. But the total volume of worry never decreases; if anything it increases.

Starting a successful startup is similar to having kids in that it's like a button you push that changes your life irrevocably. And while it's truly wonderful having kids, there are a lot of things that are easier to do before you have them than after. Many of which will make you a better parent when you do have kids. And since you can delay pushing the button for a while, most people in rich countries do.

Yet when it comes to startups, a lot of people seem to think they're supposed to start them while they're still in college. Are you crazy? And what are the universities thinking? They go out of their way to ensure their students are well supplied with contraceptives, and yet they're setting up entrepreneurship programs and startup incubators left and right.

To be fair, the universities have their hand forced here. A lot of incoming students are interested in startups. Universities are, at least de facto, expected to prepare them for their careers. So students who want to start startups hope universities can teach them about startups. And whether universities can do this or not, there's some pressure to claim they can, lest they lose applicants to other universities that do.

Can universities teach students about startups? Yes and no. They can teach students about startups, but as I explained before, this is not what you need to know. What you need to learn about are the needs of your own users, and you can't do that until you actually start the company. [5] So starting a startup is intrinsically something you can only really learn by doing it. And it's impossible to do that in college, for the reason I just explained: startups take over your life. You can't start a startup for real as a student,

because if you start a startup for real you're not a student anymore. You may be nominally a student for a bit, but you won't even be that for long. [6]

Given this dichotomy, which of the two paths should you take? Be a real student and not start a startup, or start a real startup and not be a student? I can answer that one for you. Do not start a startup in college. How to start a startup is just a subset of a bigger problem you're trying to solve: how to have a good life. And though starting a startup can be part of a good life for a lot of ambitious people, age 20 is not the optimal time to do it. Starting a startup is like a brutally fast depth-first search. Most people should still be searching breadth-first at 20.

You can do things in your early 20s that you can't do as well before or after, like plunge deeply into projects on a whim and travel super cheaply with no sense of a deadline. For unambitious people, this sort of thing is the dreaded "failure to launch," but for the ambitious ones it can be an incomparably valuable sort of exploration. If you start a startup at 20 and you're sufficiently successful, you'll never get to do it. [7]

Mark Zuckerberg will never get to bum around a foreign country. He can do other things most people can't, like charter jets to fly him to foreign countries. But success has taken a lot of the serendipity out of his life. Facebook is running him as much as he's running Facebook. And while it can be very cool to be in the grip of a project you consider your life's work, there are advantages to serendipity too, especially early in life. Among other things it gives you more options to choose your life's work from.

There's not even a tradeoff here. You're not sacrificing anything if you forgo starting a startup at 20, because you're more likely to succeed if you wait. In the unlikely case that you're 20 and one of your side projects takes off like Facebook did, you'll face a choice of running with it or not, and it may be reasonable to run with it. But the usual way startups take off is for the founders to [make them](#) take off, and it's gratuitously stupid to do that at 20.

Try

Should you do it at any age? I realize I've made startups sound pretty hard. If I haven't, let me try again: starting a startup is really hard. What if it's too hard? How can you tell if you're up to this challenge?

The answer is the fifth counterintuitive point: you can't tell. Your life so far may have given you some idea what your prospects might be if you tried to become a mathematician, or a professional football player. But unless you've had a very strange life you haven't done much that was [like](#) being a startup founder. Starting a startup will change you a lot. So what you're trying to estimate is not just what you are, but what you could grow into, and who can do that?

For the past 9 years it was my job to predict whether people would have what it took to start successful startups. It was easy to tell how smart they were, and most people reading this will be over that threshold. The hard part was predicting how tough and ambitious they would become. There may be no one who has more experience at trying to predict that, so I can tell you how much an expert can know about it, and the answer is: not much. I learned to keep a completely open mind about which of the startups in each batch would turn out to be the stars.

The founders sometimes think they know. Some arrive feeling sure they will ace Y Combinator just as they've aced every one of the (few, artificial, easy) tests they've faced in life so far. Others arrive wondering how they got in, and hoping YC doesn't discover whatever mistake caused it to accept them. But there is little correlation between founders' initial attitudes and how well their companies do.

I've read that the same is true in the military — that the swaggering recruits are no more likely to turn out to be really tough than the quiet ones. And probably for the same reason: that the tests involved are so different from the ones in their previous lives.

If you're absolutely terrified of starting a startup, you probably shouldn't do it. But if you're merely unsure whether you're up to it, the only way to find out is to try. Just not now.

Ideas

So if you want to start a startup one day, what should you do in college? There are only two things you need initially: an idea and cofounders. And the m.o. for getting both is the same. Which leads to our sixth and last counterintuitive point: that the way to get startup ideas is not to try to think of startup ideas.

I've written a whole [essay](#) on this, so I won't repeat it all here. But the short version is that if you make a conscious effort to think of startup ideas, the ideas you come up with will not merely be bad, but bad and plausible-sounding, meaning you'll waste a lot of time on them before realizing they're bad.

The way to come up with good startup ideas is to take a step back. Instead of making a conscious effort to think of startup ideas, turn your mind into the type that startup ideas form in without any conscious effort. In fact, so unconsciously that you don't even realize at first that they're startup ideas.

This is not only possible, it's how Apple, Yahoo, Google, and Facebook all got started. None of these companies were even meant to be companies at first. They were all just side projects. The best startups almost have to start as side projects, because great ideas tend to be such outliers that your conscious mind would reject them as ideas for companies.

Ok, so how do you turn your mind into the type that startup ideas form in unconsciously? (1) Learn a lot about things that matter, then (2) work on problems that interest you (3) with people you like and respect. The third part, incidentally, is how you get cofounders at the same time as the idea.

The first time I wrote that paragraph, instead of "learn a lot about things that matter," I wrote "become good at some technology." But that prescription, though sufficient, is too narrow. What was special about Brian Chesky and Joe Gebbia was not that they were experts in technology. They were good at design, and perhaps even more importantly, they were good at organizing groups and making projects happen. So you don't have to work on technology per se, so long as you work on problems demanding enough to stretch you.

What kind of problems are those? That is very hard to answer in the general case. History is full of examples of young people who were working on important problems

that [no one else](#) at the time thought were important, and in particular that their parents didn't think were important. On the other hand, history is even fuller of examples of parents who thought their kids were wasting their time and who were right. So how do you know when you're working on real stuff? [\[8\]](#)

I know how *I* know. Real problems are interesting, and I am self-indulgent in the sense that I always want to work on interesting things, even if no one else cares about them (in fact, especially if no one else cares about them), and find it very hard to make myself work on boring things, even if they're supposed to be important.

My life is full of case after case where I worked on something just because it seemed interesting, and it turned out later to be useful in some worldly way. [Y Combinator itself](#) was something I only did because it seemed interesting. So I seem to have some sort of internal compass that helps me out. But I don't know what other people have in their heads. Maybe if I think more about this I can come up with heuristics for recognizing genuinely interesting problems, but for the moment the best I can offer is the hopelessly question-begging advice that if you have a taste for genuinely interesting problems, indulging it energetically is the best way to prepare yourself for a startup. And indeed, probably also the best way to live. [\[9\]](#)

But although I can't explain in the general case what counts as an interesting problem, I can tell you about a large subset of them. If you think of technology as something that's spreading like a sort of fractal stain, every moving point on the edge represents an interesting problem. So one guaranteed way to turn your mind into the type that has good startup ideas is to get yourself to the leading edge of some technology — to cause yourself, as Paul Buchheit put it, to "live in the future." When you reach that point, ideas that will seem to other people uncannily prescient will seem obvious to you. You may not realize they're startup ideas, but you'll know they're something that ought to exist.

For example, back at Harvard in the mid 90s a fellow grad student of my friends Robert and Trevor wrote his own voice over IP software. He didn't mean it to be a startup, and he never tried to turn it into one. He just wanted to talk to his girlfriend in Taiwan without paying for long distance calls, and since he was an expert on networks it seemed obvious to him that the way to do it was turn the sound into packets and ship it over the Internet. He never did any more with his software than talk to his girlfriend, but this is exactly the way the best startups get started.

So strangely enough the optimal thing to do in college if you want to be a successful startup founder is not some sort of new, vocational version of college focused on "entrepreneurship." It's the classic version of college as education for its own sake. If you want to start a startup after college, what you should do in college is learn powerful things. And if you have genuine intellectual curiosity, that's what you'll naturally tend to do if you just follow your own inclinations. [\[10\]](#)

The component of entrepreneurship that really matters is domain expertise. The way to become Larry Page was to become an expert on search. And the way to become an expert on search was to be driven by genuine curiosity, not some ulterior motive.

At its best, starting a startup is merely an ulterior motive for curiosity. And you'll do it best if you introduce the ulterior motive toward the end of the process.

So here is the ultimate advice for young would-be startup founders, boiled down to two words: just learn.

Notes

[1] Some founders listen more than others, and this tends to be a [predictor of success](#). One of the things I remember about the Airbnbs during YC is how intently they listened.

[2] In fact, this is one of the reasons startups are possible. If big companies weren't plagued by internal inefficiencies, they'd be proportionately more effective, leaving less room for startups.

[3] In a startup you have to spend a lot of time on [schleps](#), but this sort of work is merely unglamorous, not bogus.

[4] What should you do if your true calling is gaming the system? Management consulting.

[5] The company may not be incorporated, but if you start to get significant numbers of users, you've started it, whether you realize it yet or not.

[6] It shouldn't be that surprising that colleges can't teach students how to be good startup founders, because they can't teach them how to be good employees either.

The way universities "teach" students how to be employees is to hand off the task to companies via internship programs. But you couldn't do the equivalent thing for startups, because by definition if the students did well they would never come back.

[7] Charles Darwin was 22 when he received an invitation to travel aboard the HMS Beagle as a naturalist. It was only because he was otherwise unoccupied, to a degree that alarmed his family, that he could accept it. And yet if he hadn't we probably would not know his name.

[8] Parents can sometimes be especially conservative in this department. There are some whose definition of important problems includes only those on the critical path to med school.

[9] I did manage to think of a heuristic for detecting whether you have a taste for interesting ideas: whether you find known boring ideas intolerable. Could you endure studying literary theory, or working in middle management at a large company?

[10] In fact, if your goal is to start a startup, you can stick even more closely to the ideal of a liberal education than past generations have. Back when students focused mainly on getting a job after college, they thought at least a little about how the courses they took might look to an employer. And perhaps even worse, they might shy away from

taking a difficult class lest they get a low grade, which would harm their all-important GPA. Good news: users [don't care](#) what your GPA was. And I've never heard of investors caring either. Y Combinator certainly never asks what classes you took in college or what grades you got in them.

Thanks to Sam Altman, Paul Buchheit, John Collison, Patrick Collison, Jessica Livingston, Robert Morris, Geoff Ralston, and Fred Wilson for reading drafts of this.

- [Arabic Translation](#)

Mean People Fail

November 2014

It struck me recently how few of the most successful people I know are mean. There are exceptions, but remarkably few.

Meanness isn't rare. In fact, one of the things the internet has shown us is how mean people can be. A few decades ago, only famous people and professional writers got to publish their opinions. Now everyone can, and we can all see the long tail of meanness that had previously been hidden.

And yet while there are clearly a lot of mean people out there, there are next to none among the most successful people I know. What's going on here? Are meanness and success inversely correlated?

Part of what's going on, of course, is selection bias. I only know people who work in certain fields: startup founders, programmers, professors. I'm willing to believe that successful people in other fields are mean. Maybe successful hedge fund managers are mean; I don't know enough to say. It seems quite likely that most successful drug lords are mean. But there are at least big chunks of the world that mean people don't rule, and that territory seems to be growing.

My wife and Y Combinator cofounder Jessica is one of those rare people who have x-ray vision for character. Being married to her is like standing next to an airport baggage scanner. She came to the startup world from investment banking, and she has always been struck both by how consistently successful startup founders turn out to be good people, and how consistently bad people fail as startup founders.

Why? I think there are several reasons. One is that being mean makes you stupid. That's why I hate fights. You never do your best work in a fight, because fights are not sufficiently general. Winning is always a function of the situation and the people involved. You don't win fights by thinking of big ideas but by thinking of tricks that work in one particular case. And yet fighting is just as much work as thinking about real problems. Which is particularly painful to someone who cares how their brain is used: your brain goes fast but you get nowhere, like a car spinning its wheels.

Startups don't win by attacking. They win by transcending. There are exceptions of course, but usually the way to win is to race ahead, not to stop and fight.

Another reason mean founders lose is that they can't get the best people to work for them. They can hire people who will put up with them because they need a job. But the best people have other options. A mean person can't convince the best people to work for him unless he is super convincing. And while having the best people helps any

organization, it's critical for startups.

There is also a complementary force at work: if you want to build great things, it helps to be driven by a spirit of benevolence. The startup founders who end up richest are not the ones driven by money. The ones driven by money take the big acquisition offer that nearly every successful startup gets en route. [1] The ones who keep going are driven by something else. They may not say so explicitly, but they're usually trying to improve the world. Which means people with a desire to improve the world have a natural advantage. [2]

The exciting thing is that startups are not just one random type of work in which meanness and success are inversely correlated. This kind of work is the future.

For most of history success meant control of scarce resources. One got that by fighting, whether literally in the case of pastoral nomads driving hunter-gatherers into marginal lands, or metaphorically in the case of Gilded Age financiers contending with one another to assemble railroad monopolies. For most of history, success meant success at zero-sum games. And in most of them meanness was not a handicap but probably an advantage.

That is changing. Increasingly the games that matter are not zero-sum. Increasingly you win not by fighting to get control of a scarce resource, but by having new ideas and building new things. [3]

There have long been games where you won by having new ideas. In the third century BC, Archimedes won by doing that. At least until an invading Roman army killed him. Which illustrates why this change is happening: for new ideas to matter, you need a certain degree of civil order. And not just not being at war. You also need to prevent the sort of economic violence that nineteenth century magnates practiced against one another and communist countries practiced against their citizens. People need to feel that what they create can't be stolen. [4]

That has always been the case for thinkers, which is why this trend began with them. When you think of successful people from history who weren't ruthless, you get mathematicians and writers and artists. The exciting thing is that their m.o. seems to be spreading. The games played by intellectuals are leaking into the real world, and this is reversing the historical polarity of the relationship between meanness and success.

So I'm really glad I stopped to think about this. Jessica and I have always worked hard to teach our kids not to be mean. We tolerate noise and mess and junk food, but not meanness. And now I have both an additional reason to crack down on it, and an additional argument to use when I do: that being mean makes you fail.

Notes

[1] I'm not saying all founders who take big acquisition offers are driven only by money,

but rather that those who don't aren't. Plus one can have benevolent motives for being driven by money — for example, to take care of one's family, or to be free to work on projects that improve the world.

[2] It's unlikely that every successful startup improves the world. But their founders, like parents, truly believe they do. Successful founders are in love with their companies. And while this sort of love is as blind as the love people have for one another, it is genuine.

[3] [Peter Thiel](#) would point out that successful founders still get rich from controlling monopolies, just monopolies they create rather than ones they capture. And while this is largely true, it means a big change in the sort of person who wins.

[4] To be fair, the Romans didn't mean to kill Archimedes. The Roman commander specifically ordered that he be spared. But he got killed in the chaos anyway.

In sufficiently disordered times, even thinking requires control of scarce resources, because living at all is a scarce resource.

Thanks to Sam Altman, Ron Conway, Daniel Gackle, Jessica Livingston, Robert Morris, Geoff Ralston, and Fred Wilson for reading drafts of this.

- [Portuguese Translation](#)
- [Japanese Translation](#)
- [Arabic Translation](#)

The Fatal Pinch

December 2014

Many startups go through a point a few months before they die where although they have a significant amount of money in the bank, they're also losing a lot each month, and revenue growth is either nonexistent or mediocre. The company has, say, 6 months of runway. Or to put it more brutally, 6 months before they're out of business. They expect to avoid that by raising more from investors. [\[1\]](#)

That last sentence is the fatal one.

There may be nothing founders are so prone to delude themselves about as how interested investors will be in giving them additional funding. It's hard to convince investors the first time too, but founders expect that. What bites them the second time is a confluence of three forces:

1. The company is spending more now than it did the first time it raised money.
2. Investors have much higher standards for companies that have already raised money.
3. The company is now starting to read as a failure. The first time it raised money, it was neither a success nor a failure; it was too early to ask. Now it's possible to ask that question, and the default answer is failure, because at this point that is the default outcome.

I'm going to call the situation I described in the first paragraph "the fatal pinch." I try to resist coining phrases, but making up a name for this situation may snap founders into realizing when they're in it.

One of the things that makes the fatal pinch so dangerous is that it's self-reinforcing. Founders overestimate their chances of raising more money, and so are slack about reaching profitability, which further decreases their chances of raising money.

Now that you know about the fatal pinch, how do you avoid it? Y Combinator tells founders who raise money to act as if it's the last they'll ever get. Because the self-reinforcing nature of this situation works the other way too: the less you need further investment, the easier it is to get.

What do you do if you're already in the fatal pinch? The first step is to re-evaluate the probability of raising more money. I will now, by an amazing feat of clairvoyance, do this for you: the probability is zero. [\[2\]](#)

Three options remain: you can shut down the company, you can increase how much you make, and you can decrease how much you spend.

You should shut down the company if you're certain it will fail no matter what you do. Then at least you can give back the money you have left, and save yourself however many months you would have spent riding it down.

Companies rarely *have* to fail though. What I'm really doing here is giving you the option of admitting you've already given up.

If you don't want to shut down the company, that leaves increasing revenues and decreasing expenses. In most startups, expenses = people, and decreasing expenses = firing people. [3] Deciding to fire people is usually hard, but there's one case in which it shouldn't be: when there are people you already know you should fire but you're in denial about it. If so, now's the time.

If that makes you profitable, or will enable you to make it to profitability on the money you have left, you've avoided the immediate danger.

Otherwise you have three options: you either have to fire good people, get some or all of the employees to take less salary for a while, or increase revenues.

Getting people to take less salary is a weak solution that will only work when the problem isn't too bad. If your current trajectory won't quite get you to profitability but you can get over the threshold by cutting salaries a little, you might be able to make the case to everyone for doing it. Otherwise you're probably just postponing the problem, and that will be obvious to the people whose salaries you're proposing to cut. [4]

Which leaves two options, firing good people and making more money. While trying to balance them, keep in mind the eventual goal: to be a successful product company in the sense of having a single thing lots of people use.

You should lean more toward firing people if the source of your trouble is overhiring. If you went out and hired 15 people before you even knew what you were building, you've created a broken company. You need to figure out what you're building, and it will probably be easier to do that with a handful of people than 15. Plus those 15 people might not even be the ones you need for whatever you end up building. So the solution may be to shrink and then figure out what direction to grow in. After all, you're not doing those 15 people any favors if you fly the company into ground with them aboard. They'll all lose their jobs eventually, along with all the time they expended on this doomed company.

Whereas if you only have a handful of people, it may be better to focus on trying to make more money. It may seem facile to suggest a startup make more money, as if that could be done for the asking. Usually a startup is already trying as hard as it can to sell whatever it sells. What I'm suggesting here is not so much to try harder to make money but to try to make money in a different way. For example, if you have only one person selling while the rest are writing code, consider having everyone work on selling. What good will more code do you when you're out of business? If you have to write code to close a certain deal, go ahead; that follows from everyone working on selling. But only work on whatever will get you the most revenue the soonest.

Another way to make money differently is to sell different things, and in particular to do more consultingish work. I say consultingish because there is a long slippery slope from making products to pure consulting, and you don't have to go far down it before you start to offer something really attractive to customers. Although your product may not be very appealing yet, if you're a startup your programmers will often be way better than the ones your customers have. Or you may have expertise in some new field they don't understand. So if you change your sales conversations just a little from "do you want to buy our product?" to "what do you need that you'd pay a lot for?" you may find it's suddenly a lot easier to extract money from customers.

Be ruthlessly mercenary when you start doing this, though. You're trying to save your company from death here, so make customers pay a lot, quickly. And to the extent you can, try to avoid the worst pitfalls of consulting. The ideal thing might be if you built a precisely defined derivative version of your product for the customer, and it was otherwise a straight product sale. You keep the IP and no billing by the hour.

In the best case, this consultingish work may not be just something you do to survive, but may turn out to be the [thing-that-doesn't-scale](#) that defines your company. Don't expect it to be, but as you dive into individual users' needs, keep your eyes open for narrow openings that have wide vistas beyond.

There is usually so much demand for custom work that unless you're really incompetent there has to be some point down the slope of consulting at which you can survive. But I didn't use the term slippery slope by accident; customers' insatiable demand for custom work will always be pushing you toward the bottom. So while you'll probably survive, the problem now becomes to survive with the least damage and distraction.

The good news is, plenty of successful startups have passed through near-death experiences and gone on to flourish. You just have to realize in time that you're near death. And if you're in the fatal pinch, you are.

Notes

[1] There are a handful of companies that can't reasonably expect to make money for the first year or two, because what they're building takes so long. For these companies substitute "progress" for "revenue growth." You're not one of these companies unless your initial investors agreed in advance that you were. And frankly even these companies wish they weren't, because the illiquidity of "progress" puts them at the mercy of investors.

[2] There's a variant of the fatal pinch where your existing investors help you along by promising to invest more. Or rather, where you read them as promising to invest more, while they think they're just mentioning the possibility. The way to solve this problem, if you have 8 months of runway or less, is to try to get the money right now. Then you'll either get the money, in which case (immediate) problem solved, or at least prevent your investors from helping you to remain in denial about your fundraising prospects.

[3] Obviously, if you have significant expenses other than salaries that you can

eliminate, do it now.

[4] Unless of course the source of the problem is that you're paying yourselves high salaries. If by cutting the founders' salaries to the minimum you need, you can make it to profitability, you should. But it's a bad sign if you needed to read this to realize that.

Thanks to Sam Altman, Paul Buchheit, Jessica Livingston, and Geoff Ralston for reading drafts of this.

- [Arabic Translation](#)

How You Know

December 2014

I've read Villehardouin's chronicle of the Fourth Crusade at least two times, maybe three. And yet if I had to write down everything I remember from it, I doubt it would amount to much more than a page. Multiply this times several hundred, and I get an uneasy feeling when I look at my bookshelves. What use is it to read all these books if I remember so little from them?

A few months ago, as I was reading Constance Reid's excellent biography of Hilbert, I figured out if not the answer to this question, at least something that made me feel better about it. She writes:

Hilbert had no patience with mathematical lectures which filled the students with facts but did not teach them how to frame a problem and solve it. He often used to tell them that "a perfect formulation of a problem is already half its solution."

That has always seemed to me an important point, and I was even more convinced of it after hearing it confirmed by Hilbert.

But how had I come to believe in this idea in the first place? A combination of my own experience and other things I'd read. None of which I could at that moment remember! And eventually I'd forget that Hilbert had confirmed it too. But my increased belief in the importance of this idea would remain something I'd learned from this book, even after I'd forgotten I'd learned it.

Reading and experience train your model of the world. And even if you forget the experience or what you read, its effect on your model of the world persists. Your mind is like a compiled program you've lost the source of. It works, but you don't know why.

The place to look for what I learned from Villehardouin's chronicle is not what I remember from it, but my mental models of the crusades, Venice, medieval culture, siege warfare, and so on. Which doesn't mean I couldn't have read more attentively, but at least the harvest of reading is not so miserably small as it might seem.

This is one of those things that seem obvious in retrospect. But it was a surprise to me and presumably would be to anyone else who felt uneasy about (apparently) forgetting so much they'd read.

Realizing it does more than make you feel a little better about forgetting, though. There are specific implications.

For example, reading and experience are usually "compiled" at the time they happen, using the state of your brain at that time. The same book would get compiled differently at different points in your life. Which means it is very much worth reading important books multiple times. I always used to feel some misgivings about rereading books. I unconsciously lumped reading together with work like carpentry, where having to do something again is a sign you did it wrong the first time. Whereas now the phrase "already read" seems almost ill-formed.

Intriguingly, this implication isn't limited to books. Technology will increasingly make it possible to relive our experiences. When people do that today it's usually to enjoy them again (e.g. when looking at pictures of a trip) or to find the origin of some bug in their compiled code (e.g. when Stephen Fry succeeded in remembering the childhood trauma that prevented him from singing). But as technologies for recording and playing back your life improve, it may become common for people to relive experiences without any goal in mind, simply to learn from them again as one might when rereading a book.

Eventually we may be able not just to play back experiences but also to index and even edit them. So although not knowing how you know things may seem part of being human, it may not be.

Thanks to Sam Altman, Jessica Livingston, and Robert Morris for reading drafts of this.

▪ [Japanese Translation](#)

How to Be an Expert in a Changing World

December 2014

If the world were static, we could have monotonically increasing confidence in our beliefs. The more (and more varied) experience a belief survived, the less likely it would be false. Most people implicitly believe something like this about their opinions. And they're justified in doing so with opinions about things that don't change much, like human nature. But you can't trust your opinions in the same way about things that change, which could include practically everything else.

When experts are wrong, it's often because they're experts on an earlier version of the world.

Is it possible to avoid that? Can you protect yourself against obsolete beliefs? To some extent, yes. I spent almost a decade investing in early stage startups, and curiously enough protecting yourself against obsolete beliefs is exactly what you have to do to succeed as a startup investor. Most really good startup ideas look like bad ideas at first, and many of those look bad specifically because some change in the world just switched them from bad to good. I spent a lot of time learning to recognize such ideas, and the techniques I used may be applicable to ideas in general.

The first step is to have an explicit belief in change. People who fall victim to a monotonically increasing confidence in their opinions are implicitly concluding the world is static. If you consciously remind yourself it isn't, you start to look for change.

Where should one look for it? Beyond the moderately useful generalization that human nature doesn't change much, the unfortunate fact is that change is hard to predict. This is largely a tautology but worth remembering all the same: change that matters usually comes from an unforeseen quarter.

So I don't even try to predict it. When I get asked in interviews to predict the future, I always have to struggle to come up with something plausible-sounding on the fly, like a student who hasn't prepared for an exam. [\[1\]](#) But it's not out of laziness that I haven't prepared. It seems to me that beliefs about the future are so rarely correct that they usually aren't worth the extra rigidity they impose, and that the best strategy is simply to be aggressively open-minded. Instead of trying to point yourself in the right direction, admit you have no idea what the right direction is, and try instead to be super sensitive to the winds of change.

It's ok to have working hypotheses, even though they may constrain you a bit, because they also motivate you. It's exciting to chase things and exciting to try to guess answers.

But you have to be disciplined about not letting your hypotheses harden into anything more. [2]

I believe this passive m.o. works not just for evaluating new ideas but also for having them. The way to come up with new ideas is not to try explicitly to, but to try to solve problems and simply not discount weird hunches you have in the process.

The winds of change originate in the unconscious minds of domain experts. If you're sufficiently expert in a field, any weird idea or apparently irrelevant question that occurs to you is ipso facto worth exploring. [3] Within Y Combinator, when an idea is described as crazy, it's a compliment—in fact, on average probably a higher compliment than when an idea is described as good.

Startup investors have extraordinary incentives for correcting obsolete beliefs. If they can realize before other investors that some apparently unpromising startup isn't, they can make a huge amount of money. But the incentives are more than just financial. Investors' opinions are explicitly tested: startups come to them and they have to say yes or no, and then, fairly quickly, they learn whether they guessed right. The investors who say no to a Google (and there were several) will remember it for the rest of their lives.

Anyone who must in some sense bet on ideas rather than merely commenting on them has similar incentives. Which means anyone who wants such incentives can have them, by turning their comments into bets: if you write about a topic in some fairly durable and public form, you'll find you worry much more about getting things right than most people would in a casual conversation. [4]

Another trick I've found to protect myself against obsolete beliefs is to focus initially on people rather than ideas. Though the nature of future discoveries is hard to predict, I've found I can predict quite well what sort of people will make them. Good new ideas come from earnest, energetic, independent-minded people.

Betting on people over ideas saved me countless times as an investor. We thought Airbnb was a bad idea, for example. But we could tell the founders were earnest, energetic, and independent-minded. (Indeed, almost pathologically so.) So we suspended disbelief and funded them.

This too seems a technique that should be generally applicable. Surround yourself with the sort of people new ideas come from. If you want to notice quickly when your beliefs become obsolete, you can't do better than to be friends with the people whose discoveries will make them so.

It's hard enough already not to become the prisoner of your own expertise, but it will only get harder, because change is accelerating. That's not a recent trend; change has been accelerating since the paleolithic era. Ideas beget ideas. I don't expect that to change. But I could be wrong.

Notes

[1] My usual trick is to talk about aspects of the present that most people haven't noticed yet.

[2] Especially if they become well enough known that people start to identify them with you. You have to be extra skeptical about things you want to believe, and once a hypothesis starts to be identified with you, it will almost certainly start to be in that category.

[3] In practice "sufficiently expert" doesn't require one to be recognized as an expert—which is a trailing indicator in any case. In many fields a year of focused work plus caring a lot would be enough.

[4] Though they are public and persist indefinitely, comments on e.g. forums and places like Twitter seem empirically to work like casual conversation. The threshold may be whether what you write has a title.

Thanks to Sam Altman, Patrick Collison, and Robert Morris for reading drafts of this.

- [Spanish Translation](#)

- [Arabic Translation](#)

Let the Other 95% of Great Programmers In

December 2014

American technology companies want the government to make immigration easier because they say they can't find enough programmers in the US. Anti-immigration people say that instead of letting foreigners take these jobs, we should train more Americans to be programmers. Who's right?

The technology companies are right. What the anti-immigration people don't understand is that there is a huge variation in ability between competent programmers and exceptional ones, and while you can train people to be competent, you can't train them to be exceptional. Exceptional programmers have an aptitude for and [interest in](#) programming that is not merely the product of training. [1]

The US has less than 5% of the world's population. Which means if the qualities that make someone a great programmer are evenly distributed, 95% of great programmers are born outside the US.

The anti-immigration people have to invent some explanation to account for all the effort technology companies have expended trying to make immigration easier. So they claim it's because they want to drive down salaries. But if you talk to startups, you find practically every one over a certain size has gone through legal contortions to get programmers into the US, where they then paid them the same as they'd have paid an American. Why would they go to extra trouble to get programmers for the same price? The only explanation is that they're telling the truth: there are just not enough great programmers to go around. [2]

I asked the CEO of a startup with about 70 programmers how many more he'd hire if he could get all the great programmers he wanted. He said "We'd hire 30 tomorrow morning." And this is one of the hot startups that always win recruiting battles. It's the same all over Silicon Valley. Startups are that constrained for talent.

It would be great if more Americans were trained as programmers, but no amount of training can flip a ratio as overwhelming as 95 to 5. Especially since programmers are being trained in other countries too. Barring some cataclysm, it will always be true that most great programmers are born outside the US. It will always be true that most people who are great at anything are born outside the US. [3]

Exceptional performance implies immigration. A country with only a few percent of the world's population will be exceptional in some field only if there are a lot of immigrants working in it.

But this whole discussion has taken something for granted: that if we let more great programmers into the US, they'll want to come. That's true now, and we don't realize how lucky we are that it is. If we want to keep this option open, the best way to do it is to take advantage of it: the more of the world's great programmers are here, the more the rest will want to come here.

And if we don't, the US could be seriously fucked. I realize that's strong language, but the people dithering about this don't seem to realize the power of the forces at work here. Technology gives the best programmers huge leverage. The world market in programmers seems to be becoming dramatically more liquid. And since good people like good colleagues, that means the best programmers could collect in just a few hubs. Maybe mostly in one hub.

What if most of the great programmers collected in one hub, and it wasn't here? That scenario may seem unlikely now, but it won't be if things change as much in the next 50 years as they did in the last 50.

We have the potential to ensure that the US remains a technology superpower just by letting in a few thousand great programmers a year. What a colossal mistake it would be to let that opportunity slip. It could easily be the defining mistake this generation of American politicians later become famous for. And unlike other potential mistakes on that scale, it costs nothing to fix.

So please, get on with it.

Notes

[1] How much better is a great programmer than an ordinary one? So much better that you can't even measure the difference directly. A great programmer doesn't merely do the same work faster. A great programmer will invent things an ordinary programmer would never even think of. This doesn't mean a great programmer is infinitely more valuable, because any invention has a finite market value. But it's easy to imagine cases where a great programmer might invent things worth 100x or even 1000x an average programmer's salary.

[2] There are a handful of consulting firms that rent out big pools of foreign programmers they bring in on H1-B visas. By all means crack down on these. It should be easy to write legislation that distinguishes them, because they are so different from technology companies. But it is dishonest of the anti-immigration people to claim that companies like Google and Facebook are driven by the same motives. An influx of inexpensive but mediocre programmers is the last thing they'd want; it would destroy them.

[3] Though this essay talks about programmers, the group of people we need to import is broader, ranging from designers to programmers to electrical engineers. The best one

could do as a general term might be "digital talent." It seemed better to make the argument a little too narrow than to confuse everyone with a neologism.

Thanks to Sam Altman, John Collison, Patrick Collison, Jessica Livingston, Geoff Ralston, Fred Wilson, and Qasar Younis for reading drafts of this.

- [Spanish Translation](#)

Don't Talk to Corp Dev

January 2015

Corporate Development, aka corp dev, is the group within companies that buys other companies. If you're talking to someone from corp dev, that's why, whether you realize it yet or not.

It's usually a mistake to talk to corp dev unless (a) you want to sell your company right now and (b) you're sufficiently likely to get an offer at an acceptable price. In practice that means startups should only talk to corp dev when they're either doing really well or really badly. If you're doing really badly, meaning the company is about to die, you may as well talk to them, because you have nothing to lose. And if you're doing really well, you can safely talk to them, because you both know the price will have to be high, and if they show the slightest sign of wasting your time, you'll be confident enough to tell them to get lost.

The danger is to companies in the middle. Particularly to young companies that are growing fast, but haven't been doing it for long enough to have grown big yet. It's usually a mistake for a promising company less than a year old even to talk to corp dev.

But it's a mistake founders constantly make. When someone from corp dev wants to meet, the founders tell themselves they should at least find out what they want. Besides, they don't want to offend Big Company by refusing to meet.

Well, I'll tell you what they want. They want to talk about buying you. That's what the title "corp dev" means. So before agreeing to meet with someone from corp dev, ask yourselves, "Do we want to sell the company right now?" And if the answer is no, tell them "Sorry, but we're focusing on growing the company." They won't be offended. And certainly the founders of Big Company won't be offended. If anything they'll think more highly of you. You'll remind them of themselves. They didn't sell either; that's why they're in a position now to buy other companies. [\[1\]](#)

Most founders who get contacted by corp dev already know what it means. And yet even when they know what corp dev does and know they don't want to sell, they take the meeting. Why do they do it? The same mix of denial and wishful thinking that underlies most mistakes founders make. It's flattering to talk to someone who wants to buy you. And who knows, maybe their offer will be surprisingly high. You should at least see what it is, right?

No. If they were going to send you an offer immediately by email, sure, you might as well open it. But that is not how conversations with corp dev work. If you get an offer at all, it will be at the end of a long and unbelievably distracting process. And if the offer is surprising, it will be surprisingly low.

Distractions are the thing you can least afford in a startup. And conversations with corp dev are the worst sort of distraction, because as well as consuming your [attention](#) they undermine your morale. One of the tricks to surviving a grueling process is not to stop and think how tired you are. Instead you get into a sort of flow. [2] Imagine what it would do to you if at mile 20 of a marathon, someone ran up beside you and said "You must feel really tired. Would you like to stop and take a rest?" Conversations with corp dev are like that but worse, because the suggestion of stopping gets combined in your mind with the imaginary high price you think they'll offer.

And then you're really in trouble. If they can, corp dev people like to turn the tables on you. They like to get you to the point where you're trying to convince them to buy instead of them trying to convince you to sell. And surprisingly often they succeed.

This is a very slippery slope, greased with some of the most powerful forces that can work on founders' minds, and attended by an experienced professional whose full time job is to push you down it.

Their tactics in pushing you down that slope are usually fairly brutal. Corp dev people's whole job is to buy companies, and they don't even get to choose which. The only way their performance is measured is by how cheaply they can buy you, and the more ambitious ones will stop at nothing to achieve that. For example, they'll almost always start with a lowball offer, just to see if you'll take it. Even if you don't, a low initial offer will demoralize you and make you easier to manipulate.

And that is the most innocent of their tactics. Just wait till you've agreed on a price and think you have a done deal, and then they come back and say their boss has vetoed the deal and won't do it for more than half the agreed upon price. Happens all the time. If you think investors can behave badly, it's nothing compared to what corp dev people can do. Even corp dev people at companies that are otherwise benevolent.

I remember once complaining to a friend at Google about some nasty trick their corp dev people had pulled on a YC startup.

"What happened to Don't be Evil?" I asked.

"I don't think corp dev got the memo," he replied.

The tactics you encounter in M&A conversations can be like nothing you've experienced in the otherwise comparatively [upstanding](#) world of Silicon Valley. It's as if a chunk of genetic material from the old-fashioned robber baron business world got incorporated into the startup world. [3]

The simplest way to protect yourself is to use the trick that John D. Rockefeller, whose grandfather was an alcoholic, used to protect himself from becoming one. He once told a Sunday school class

Boys, do you know why I never became a drunkard? Because I never took the first drink.

Do you want to sell your company right now? Not eventually, right now. If not, just don't take the first meeting. They won't be offended. And you in turn will be guaranteed

to be spared one of the worst experiences that can happen to a startup.

If you do want to sell, there's another set of [techniques](#) for doing that. But the biggest mistake founders make in dealing with corp dev is not doing a bad job of talking to them when they're ready to, but talking to them before they are. So if you remember only the title of this essay, you already know most of what you need to know about M&A in the first year.

Notes

[1] I'm not saying you should never sell. I'm saying you should be clear in your own mind about whether you want to sell or not, and not be led by manipulation or wishful thinking into trying to sell earlier than you otherwise would have.

[2] In a startup, as in most competitive sports, the task at hand almost does this for you; you're too busy to feel tired. But when you lose that protection, e.g. at the final whistle, the fatigue hits you like a wave. To talk to corp dev is to let yourself feel it mid-game.

[3] To be fair, the apparent misdeeds of corp dev people are magnified by the fact that they function as the face of a large organization that often doesn't know its own mind. Acquirers can be surprisingly indecisive about acquisitions, and their flakiness is indistinguishable from dishonesty by the time it filters down to you.

Thanks to Marc Andreessen, Jessica Livingston, Geoff Ralston, and Qasar Younis for reading drafts of this.

What Doesn't Seem Like Work?

January 2015

My father is a mathematician. For most of my childhood he worked for Westinghouse, modelling nuclear reactors.

He was one of those lucky people who know early on what they want to do. When you talk to him about his childhood, there's a clear watershed at about age 12, when he "got interested in maths."

He grew up in the small Welsh seacoast town of [Pwllheli](#). As we retraced his walk to school on Google Street View, he said that it had been nice growing up in the country.

"Didn't it get boring when you got to be about 15?" I asked.

"No," he said, "by then I was interested in maths."

In another conversation he told me that what he really liked was solving problems. To me the exercises at the end of each chapter in a math textbook represent work, or at best a way to reinforce what you learned in that chapter. To him the problems were the reward. The text of each chapter was just some advice about solving them. He said that as soon as he got a new textbook he'd immediately work out all the problems — to the slight annoyance of his teacher, since the class was supposed to work through the book gradually.

Few people know so early or so certainly what they want to work on. But talking to my father reminded me of a heuristic the rest of us can use. If something that seems like work to other people doesn't seem like work to you, that's something you're well suited for. For example, a lot of programmers I know, including me, actually like debugging. It's not something people tend to volunteer; one likes it the way one likes popping zits. But you may have to like debugging to like programming, considering the degree to which programming consists of it.

The stranger your tastes seem to other people, the stronger evidence they probably are of what you should do. When I was in college I used to write papers for my friends. It was quite interesting to write a paper for a class I wasn't taking. Plus they were always so relieved.

It seemed curious that the same task could be painful to one person and pleasant to another, but I didn't realize at the time what this imbalance implied, because I wasn't looking for it. I didn't realize how hard it can be to decide what you should work on, and that you sometimes have to [figure it out](#) from subtle clues, like a detective solving a case in a mystery novel. So I bet it would help a lot of people to ask themselves about

this explicitly. What seems like work to other people that doesn't seem like work to you?

Thanks to Sam Altman, Trevor Blackwell, Jessica Livingston, Robert Morris, and my father for reading drafts of this.

- [Robert Morris: All About Programming](#)
- [French Translation](#)

The Ronco Principle

January 2015

No one, VC or angel, has invested in more of the top startups than Ron Conway. He knows what happened in every deal in the Valley, half the time because he arranged it.

And yet he's a super nice guy. In fact, nice is not the word. Ronco is good. I know of zero instances in which he has behaved badly. It's hard even to imagine.

When I first came to Silicon Valley I thought "How lucky that someone so powerful is so benevolent." But gradually I realized it wasn't luck. It was by being benevolent that Ronco became so powerful. All the deals he gets to invest in come to him through referrals. Google did. Facebook did. Twitter was a referral from Evan Williams himself. And the reason so many people refer deals to him is that he's proven himself to be a good guy.

Good does not mean being a pushover. I would not want to face an angry Ronco. But if Ron's angry at you, it's because you did something wrong. Ron is so old school he's Old Testament. He will smite you in his just wrath, but there's no malice in it.

In almost every domain there are advantages to seeming good. It makes people trust you. But actually being good is an expensive way to seem good. To an amoral person it might seem to be overkill.

In some fields it might be, but apparently not in the startup world. Though plenty of investors are jerks, there is a clear trend among them: the most successful investors are also the most upstanding. [1]

It was not always this way. I would not feel confident saying that about investors twenty years ago.

What changed? The startup world became more transparent and more unpredictable. Both make it harder to seem good without actually being good.

It's obvious why transparency has that effect. When an investor maltreats a founder now, it gets out. Maybe not all the way to the press, but other founders hear about it, and that investor starts to lose deals. [2]

The effect of unpredictability is more subtle. It increases the work of being inconsistent. If you're going to be two-faced, you have to know who you should be nice to and who you can get away with being nasty to. In the startup world, things change so rapidly that you can't tell. The random college kid you talk to today might in a couple years be the CEO of the hottest startup in the Valley. If you can't tell who to be nice to, you have to

be nice to everyone. And probably the only people who can manage that are the people who are genuinely good.

In a sufficiently connected and unpredictable world, you can't seem good without being good.

As often happens, Ron discovered how to be the investor of the future by accident. He didn't foresee the future of startup investing, realize it would pay to be upstanding, and force himself to behave that way. It would feel unnatural to him to behave any other way. He was already [living in the future](#).

Fortunately that future is not limited to the startup world. The startup world is more transparent and unpredictable than most, but almost everywhere the trend is in that direction.

Notes

[1] I'm not saying that if you sort investors by benevolence you've also sorted them by returns, but rather that if you do a scatterplot with benevolence on the x axis and returns on the y, you'd see a clear upward trend.

[2] Y Combinator in particular, because it aggregates data from so many startups, has a pretty comprehensive view of investor behavior.

Thanks to Sam Altman and Jessica Livingston for reading drafts of this.

▪ [Japanese Translation](#)

What Microsoft Is this the Altair Basic of?

February 2015

One of the most valuable exercises you can try if you want to understand startups is to look at the most successful companies and explain why they were not as lame as they seemed when they first launched. Because they practically all seemed lame at first. Not just small, lame. Not just the first step up a big mountain. More like the first step into a swamp.

A Basic interpreter for the Altair? How could that ever grow into a giant company? People sleeping on airbeds in strangers' apartments? A web site for college students to stalk one another? A wimpy little single-board computer for hobbyists that used a TV as a monitor? A new search engine, when there were already about 10, and they were all trying to de-emphasize search? These ideas didn't just seem small. They seemed wrong. They were the kind of ideas you could not merely ignore, but ridicule.

Often the founders themselves didn't know why their ideas were promising. They were attracted to these ideas by instinct, because they were [living in the future](#) and they sensed that something was missing. But they could not have put into words exactly how their ugly ducklings were going to grow into big, beautiful swans.

Most people's first impulse when they hear about a lame-sounding new startup idea is to make fun of it. Even a lot of people who should know better.

When I encounter a startup with a lame-sounding idea, I ask "What Microsoft is this the Altair Basic of?" Now it's a puzzle, and the burden is on me to solve it. Sometimes I can't think of an answer, especially when the idea is a made-up one. But it's remarkable how often there does turn out to be an answer. Often it's one the founders themselves hadn't seen yet.

Intriguingly, there are sometimes multiple answers. I talked to a startup a few days ago that could grow into 3 distinct Microsofts. They'd probably vary in size by orders of magnitude. But you can never predict how big a Microsoft is going to be, so in cases like that I encourage founders to follow whichever path is most immediately exciting to them. Their instincts got them this far. Why stop now?

Change Your Name

August 2015

If you have a US startup called X and you don't have x.com, you should probably change your name.

The reason is not just that people can't find you. For companies with mobile apps, especially, having the right domain name is not as critical as it used to be for getting users. The problem with not having the .com of your name is that it signals weakness. Unless you're so big that your reputation precedes you, a marginal domain suggests you're a marginal company. Whereas (as Stripe shows) having x.com signals strength even if it has no relation to what you do.

Even good founders can be in denial about this. Their denial derives from two very powerful forces: identity, and lack of imagination.

X is what we *are*, founders think. There's no other name as good. Both of which are false.

You can fix the first by stepping back from the problem. Imagine you'd called your company something else. If you had, surely you'd be just as attached to that name as you are to your current one. The idea of switching to your current name would seem repellent. [1]

There's nothing intrinsically great about your current name. Nearly all your attachment to it comes from it being attached to you. [2]

The way to neutralize the second source of denial, your inability to think of other potential names, is to acknowledge that you're bad at naming. Naming is a completely separate skill from those you need to be a good founder. You can be a great startup founder but hopeless at thinking of names for your company.

Once you acknowledge that, you stop believing there is nothing else you could be called. There are lots of other potential names that are as good or better; you just can't think of them.

How do you find them? One answer is the default way to solve problems you're bad at: find someone else who can think of names. But with company names there is another possible approach. It turns out almost any word or word pair that is not an obviously bad name is a sufficiently good one, and the number of such domains is so large that you can find plenty that are cheap or even untaken. So make a list and try to buy some. That's what [Stripe](#) did. (Their search also turned up parse.com, which their friends at Parse took.)

The reason I know that naming companies is a distinct skill orthogonal to the others you need in a startup is that I happen to have it. Back when I was running YC and did more office hours with startups, I would often help them find new names. 80% of the time we could find at least one good name in a 20 minute office hour slot.

Now when I do office hours I have to focus on more important questions, like what the company is doing. I tell them when they need to change their name. But I know the power of the forces that have them in their grip, so I know most won't listen. [3]

There are of course examples of startups that have succeeded without having the .com of their name. There are startups that have succeeded despite any number of different mistakes. But this mistake is less excusable than most. It's something that can be fixed in a couple days if you have sufficient discipline to acknowledge the problem.

100% of the top 20 YC companies by valuation have the .com of their name. 94% of the top 50 do. But only 66% of companies in the current batch have the .com of their name. Which suggests there are lessons ahead for most of the rest, one way or another.

Notes

[1] Incidentally, this thought experiment works for [nationality and religion](#) too.

[2] The liking you have for a name that has become part of your identity manifests itself not directly, which would be easy to discount, but as a collection of specious beliefs about its intrinsic qualities. (This too is true of nationality and religion as well.)

[3] Sometimes founders know it's a problem that they don't have the .com of their name, but delusion strikes a step later in the belief that they'll be able to buy it despite having no evidence it's for sale. Don't believe a domain is for sale unless the owner has already told you an asking price.

Thanks to Sam Altman, Jessica Livingston, and Geoff Ralston for reading drafts of this.

Why It's Safe for Founders to Be Nice

August 2015

I recently got an email from a founder that helped me understand something important: why it's safe for startup founders to be nice people.

I grew up with a cartoon idea of a very successful businessman (in the cartoon it was always a man): a rapacious, cigar-smoking, table-thumping guy in his fifties who wins by exercising power, and isn't too fussy about how. As I've written before, one of the things that has surprised me most about startups is [how few](#) of the most successful founders are like that. Maybe successful people in other industries are; I don't know; but not startup founders. [1]

I knew this empirically, but I never saw the math of why till I got this founder's email. In it he said he worried that he was fundamentally soft-hearted and tended to give away too much for free. He thought perhaps he needed "a little dose of sociopath-ness."

I told him not to worry about it, because so long as he built something good enough to spread by word of mouth, he'd have a superlinear growth curve. If he was bad at extracting money from people, at worst this curve would be some constant multiple less than 1 of what it might have been. But a constant multiple of any curve is exactly the same shape. The numbers on the Y axis are smaller, but the curve is just as steep, and when anything grows at the rate of a successful startup, the Y axis will take care of itself.

Some examples will make this clear. Suppose your company is making \$1000 a month now, and you've made something so great that it's growing at 5% a week. Two years from now, you'll be making about \$160k a month.

Now suppose you're so un-rapacious that you only extract half as much from your users as you could. That means two years later you'll be making \$80k a month instead of \$160k. How far behind are you? How long will it take to catch up with where you'd have been if you were extracting every penny? A mere 15 weeks. After two years, the un-rapacious founder is only 3.5 months behind the rapacious one. [2]

If you're going to optimize a number, the one to choose is your [growth rate](#). Suppose as before that you only extract half as much from users as you could, but that you're able to grow 6% a week instead of 5%. Now how are you doing compared to the rapacious founder after two years? You're already ahead—\$214k a month versus \$160k—and pulling away fast. In another year you'll be making \$4.4 million a month to the rapacious founder's \$2 million.

Obviously one case where it would help to be rapacious is when growth depends on

that. What makes startups different is that usually it doesn't. Startups usually win by making something so great that people recommend it to their friends. And being rapacious not only doesn't help you do that, but probably hurts. [3]

The reason startup founders can safely be nice is that making great things is compounded, and rapacity isn't.

So if you're a founder, here's a deal you can make with yourself that will both make you happy and make your company successful. Tell yourself you can be as nice as you want, so long as you work hard on your growth rate to compensate. Most successful startups make that tradeoff unconsciously. Maybe if you do it consciously you'll do it even better.

Notes

[1] Many think successful startup founders are driven by money. In fact the secret weapon of the most successful founders is that they aren't. If they were, they'd have taken one of the acquisition offers that every fast-growing startup gets on the way up. What drives the most successful founders is the same thing that drives most people who make things: the company is their project.

[2] In fact since $2 \approx 1.05^{15}$, the un-rapacious founder is always 15 weeks behind the rapacious one.

[3] The other reason it might help to be good at squeezing money out of customers is that startups usually lose money at first, and making more per customer makes it easier to get to profitability before your initial funding runs out. But while it is very common for startups to [die](#) from running through their initial funding and then being unable to raise more, the underlying cause is usually slow growth or excessive spending rather than insufficient effort to extract money from existing customers.

Thanks to Sam Altman, Harj Taggar, Jessica Livingston, and Geoff Ralston for reading drafts of this, and to Randall Bennett for being such a nice guy.

Default Alive or Default Dead?

October 2015

When I talk to a startup that's been operating for more than 8 or 9 months, the first thing I want to know is almost always the same. Assuming their expenses remain constant and their revenue growth is what it has been over the last several months, do they make it to profitability on the money they have left? Or to put it more dramatically, by default do they live or die?

The startling thing is how often the founders themselves don't know. Half the founders I talk to don't know whether they're default alive or default dead.

If you're among that number, Trevor Blackwell has made a handy [calculator](#) you can use to find out.

The reason I want to know first whether a startup is default alive or default dead is that the rest of the conversation depends on the answer. If the company is default alive, we can talk about ambitious new things they could do. If it's default dead, we probably need to talk about how to save it. We know the current trajectory ends badly. How can they get off that trajectory?

Why do so few founders know whether they're default alive or default dead? Mainly, I think, because they're not used to asking that. It's not a question that makes sense to ask early on, any more than it makes sense to ask a 3 year old how he plans to support himself. But as the company grows older, the question switches from meaningless to critical. That kind of switch often takes people by surprise.

I propose the following solution: instead of starting to ask too late whether you're default alive or default dead, start asking too early. It's hard to say precisely when the question switches polarity. But it's probably not that dangerous to start worrying too early that you're default dead, whereas it's very dangerous to start worrying too late.

The reason is a phenomenon I wrote about earlier: the [fatal pinch](#). The fatal pinch is default dead + slow growth + not enough time to fix it. And the way founders end up in it is by not realizing that's where they're headed.

There is another reason founders don't ask themselves whether they're default alive or default dead: they assume it will be easy to raise more money. But that assumption is often false, and worse still, the more you depend on it, the falser it becomes.

Maybe it will help to separate facts from hopes. Instead of thinking of the future with vague optimism, explicitly separate the components. Say "We're default dead, but we're counting on investors to save us." Maybe as you say that, it will set off the same alarms

in your head that it does in mine. And if you set off the alarms sufficiently early, you may be able to avoid the fatal pinch.

It would be safe to be default dead if you could count on investors saving you. As a rule their interest is a function of growth. If you have steep revenue growth, say over 5x a year, you can start to count on investors being interested even if you're not profitable. [1] But investors are so fickle that you can never do more than start to count on them. Sometimes something about your business will spook investors even if your growth is great. So no matter how good your growth is, you can never safely treat fundraising as more than a plan A. You should always have a plan B as well: you should know (as in write down) precisely what you'll need to do to survive if you can't raise more money, and precisely when you'll have to switch to plan B if plan A isn't working.

In any case, growing fast versus operating cheaply is far from the sharp dichotomy many founders assume it to be. In practice there is surprisingly little connection between how much a startup spends and how fast it grows. When a startup grows fast, it's usually because the product hits a nerve, in the sense of hitting some big need straight on. When a startup spends a lot, it's usually because the product is expensive to develop or sell, or simply because they're wasteful.

If you're paying attention, you'll be asking at this point not just how to avoid the fatal pinch, but how to avoid being default dead. That one is easy: don't hire too fast. Hiring too fast is by far the biggest killer of startups that raise money. [2]

Founders tell themselves they need to hire in order to grow. But most err on the side of overestimating this need rather than underestimating it. Why? Partly because there's so much work to do. Naïve founders think that if they can just hire enough people, it will all get done. Partly because successful startups have lots of employees, so it seems like that's what one does in order to be successful. In fact the large staffs of successful startups are probably more the effect of growth than the cause. And partly because when founders have slow growth they don't want to face what is usually the real reason: the product is not appealing enough.

Plus founders who've just raised money are often encouraged to overhire by the VCs who funded them. Kill-or-cure strategies are optimal for VCs because they're protected by the portfolio effect. VCs want to blow you up, in one sense of the phrase or the other. But as a founder your incentives are different. You want above all to survive. [3]

Here's a common way startups die. They make something moderately appealing and have decent initial growth. They raise their first round fairly easily, because the founders seem smart and the idea sounds plausible. But because the product is only moderately appealing, growth is ok but not great. The founders convince themselves that hiring a bunch of people is the way to boost growth. Their investors agree. But (because the product is only moderately appealing) the growth never comes. Now they're rapidly running out of runway. They hope further investment will save them. But because they have high expenses and slow growth, they're now unappealing to investors. They're unable to raise more, and the company dies.

What the company should have done is address the fundamental problem: that the product is only moderately appealing. Hiring people is rarely the way to fix that. More often than not it makes it harder. At this early stage, the product needs to evolve more than to be "built out," and that's usually easier with fewer people. [4]

Asking whether you're default alive or default dead may save you from this. Maybe the alarm bells it sets off will counteract the forces that push you to overhire. Instead you'll be compelled to seek growth in other ways. For example, by [doing things that don't scale](#), or by redesigning the product in the way only founders can. And for many if not most startups, these paths to growth will be the ones that actually work.

Airbnb waited 4 months after raising money at the end of Y Combinator before they hired their first employee. In the meantime the founders were terribly overworked. But they were overworked evolving Airbnb into the astonishingly successful organism it is now.

Notes

[1] Steep usage growth will also interest investors. Revenue will ultimately be a constant multiple of usage, so $x\%$ usage growth predicts $x\%$ revenue growth. But in practice investors discount merely predicted revenue, so if you're measuring usage you need a higher growth rate to impress investors.

[2] Startups that don't raise money are saved from hiring too fast because they can't afford to. But that doesn't mean you should avoid raising money in order to avoid this problem, any more than that total abstinence is the only way to avoid becoming an alcoholic.

[3] I would not be surprised if VCs' tendency to push founders to overhire is not even in their own interest. They don't know how many of the companies that get killed by overspending might have done well if they'd survived. My guess is a significant number.

[4] After reading a draft, Sam Altman wrote:

"I think you should make the hiring point more strongly. I think it's roughly correct to say that YC's most successful companies have never been the fastest to hire, and one of the marks of a great founder is being able to resist this urge."

Paul Buchheit adds:

"A related problem that I see a lot is premature scaling—founders take a small business that isn't really working (bad unit economics, typically) and then scale it up because they want impressive growth numbers. This is similar to over-hiring in that it makes the business much harder to fix once it's big, plus they are bleeding cash really fast."

Thanks to Sam Altman, Paul Buchheit, Joe Gebbia, Jessica Livingston, and Geoff Ralston for reading drafts of this.

Write Like You Talk

October 2015

Here's a simple trick for getting more people to read what you write: write in spoken language.

Something comes over most people when they start writing. They write in a different language than they'd use if they were talking to a friend. The sentence structure and even the words are different. No one uses "pen" as a verb in spoken English. You'd feel like an idiot using "pen" instead of "write" in a conversation with a friend.

The last straw for me was a sentence I read a couple days ago:

The mercurial Spaniard himself declared: "After Altamira, all is decadence."

It's from Neil Oliver's *A History of Ancient Britain*. I feel bad making an example of this book, because it's no worse than lots of others. But just imagine calling Picasso "the mercurial Spaniard" when talking to a friend. Even one sentence of this would raise eyebrows in conversation. And yet people write whole books of it.

Ok, so written and spoken language are different. Does that make written language worse?

If you want people to read and understand what you write, yes. Written language is more complex, which makes it more work to read. It's also more formal and distant, which gives the reader's attention permission to drift. But perhaps worst of all, the complex sentences and fancy words give you, the writer, the false impression that you're saying more than you actually are.

You don't need complex sentences to express complex ideas. When specialists in some abstruse topic talk to one another about ideas in their field, they don't use sentences any more complex than they do when talking about what to have for lunch. They use different words, certainly. But even those they use no more than necessary. And in my experience, the harder the subject, the more informally experts speak. Partly, I think, because they have less to prove, and partly because the harder the ideas you're talking about, the less you can afford to let language get in the way.

Informal language is the athletic clothing of ideas.

I'm not saying spoken language always works best. Poetry is as much music as text, so you can say things you wouldn't say in conversation. And there are a handful of writers who can get away with using fancy language in prose. And then of course there are

cases where writers don't want to make it easy to understand what they're saying—in corporate announcements of bad news, for example, or at the more [bogus](#) end of the humanities. But for nearly everyone else, spoken language is better.

It seems to be hard for most people to write in spoken language. So perhaps the best solution is to write your first draft the way you usually would, then afterward look at each sentence and ask "Is this the way I'd say this if I were talking to a friend?" If it isn't, imagine what you would say, and use that instead. After a while this filter will start to operate as you write. When you write something you wouldn't say, you'll hear the clank as it hits the page.

Before I publish a new essay, I read it out loud and fix everything that doesn't sound like conversation. I even fix bits that are phonetically awkward; I don't know if that's necessary, but it doesn't cost much.

This trick may not always be enough. I've seen writing so far removed from spoken language that it couldn't be fixed sentence by sentence. For cases like that there's a more drastic solution. After writing the first draft, try explaining to a friend what you just wrote. Then replace the draft with what you said to your friend.

People often tell me how much my essays sound like me talking. The fact that this seems worthy of comment shows how rarely people manage to write in spoken language. Otherwise everyone's writing would sound like them talking.

If you simply manage to write in spoken language, you'll be ahead of 95% of writers. And it's so easy to do: just don't let a sentence through unless it's the way you'd say it to a friend.

Thanks to Patrick Collison and Jessica Livingston for reading drafts of this.

- [Japanese Translation](#)
- [Arabic Translation](#)

A Way to Detect Bias

October 2015

This will come as a surprise to a lot of people, but in some cases it's possible to detect bias in a selection process without knowing anything about the applicant pool. Which is exciting because among other things it means third parties can use this technique to detect bias whether those doing the selecting want them to or not.

You can use this technique whenever (a) you have at least a random sample of the applicants that were selected, (b) their subsequent performance is measured, and (c) the groups of applicants you're comparing have roughly equal distribution of ability.

How does it work? Think about what it means to be biased. What it means for a selection process to be biased against applicants of type x is that it's harder for them to make it through. Which means applicants of type x have to be better to get selected than applicants not of type x . [\[1\]](#) Which means applicants of type x who do make it through the selection process will outperform other successful applicants. And if the performance of all the successful applicants is measured, you'll know if they do.

Of course, the test you use to measure performance must be a valid one. And in particular it must not be invalidated by the bias you're trying to measure. But there are some domains where performance can be measured, and in those detecting bias is straightforward. Want to know if the selection process was biased against some type of applicant? Check whether they outperform the others. This is not just a heuristic for detecting bias. It's what bias means.

For example, many suspect that venture capital firms are biased against female founders. This would be easy to detect: among their portfolio companies, do startups with female founders outperform those without? A couple months ago, one VC firm (almost certainly unintentionally) published a study showing bias of this type. First Round Capital found that among its portfolio companies, startups with female founders [outperformed](#) those without by 63%. [\[2\]](#)

The reason I began by saying that this technique would come as a surprise to many people is that we so rarely see analyses of this type. I'm sure it will come as a surprise to First Round that they performed one. I doubt anyone there realized that by limiting their sample to their own portfolio, they were producing a study not of startup trends but of their own biases when selecting companies.

I predict we'll see this technique used more in the future. The information needed to conduct such studies is increasingly available. Data about who applies for things is usually closely guarded by the organizations selecting them, but nowadays data about who gets selected is often publicly available to anyone who takes the trouble to

aggregate it.

Notes

[1] This technique wouldn't work if the selection process looked for different things from different types of applicants—for example, if an employer hired men based on their ability but women based on their appearance.

[2] As Paul Buchheit points out, First Round excluded their most successful investment, Uber, from the study. And while it makes sense to exclude outliers from some types of studies, studies of returns from startup investing, which is all about hitting outliers, are not one of them.

Thanks to Sam Altman, Jessica Livingston, and Geoff Ralston for reading drafts of this.

- [Arabic Translation](#)
- [Swedish Translation](#)

Jessica Livingston

November 2015

A few months ago an article about Y Combinator said that early on it had been a "one-man show." It's sadly common to read that sort of thing. But the problem with that description is not just that it's unfair. It's also misleading. Much of what's most novel about YC is due to Jessica Livingston. If you don't understand her, you don't understand YC. So let me tell you a little about Jessica.

YC had 4 founders. Jessica and I decided one night to start it, and the next day we recruited my friends Robert Morris and Trevor Blackwell. Jessica and I ran YC day to day, and Robert and Trevor read applications and did interviews with us.

Jessica and I were already dating when we started YC. At first we tried to act "professional" about this, meaning we tried to conceal it. In retrospect that seems ridiculous, and we soon dropped the pretense. And the fact that Jessica and I were a couple is a big part of what made YC what it was. YC felt like a family. The founders early on were mostly young. We all had dinner together once a week, cooked for the first couple years by me. Our first building had been a private home. The overall atmosphere was shockingly different from a VC's office on Sand Hill Road, in a way that was entirely for the better. There was an authenticity that everyone who walked in could sense. And that didn't just mean that people trusted us. It was the perfect quality to instill in startups. Authenticity is one of the most important things YC looks for in founders, not just because fakers and opportunists are annoying, but because authenticity is one of the main things that separates the most successful startups from the rest.

Early YC was a family, and Jessica was its mom. And the culture she defined was one of YC's most important innovations. Culture is important in any organization, but at YC culture wasn't just how we behaved when we built the product. At YC, the culture was the product.

Jessica was also the mom in another sense: she had the last word. Everything we did as an organization went through her first — who to fund, what to say to the public, how to deal with other companies, who to hire, everything.

Before we had kids, YC was more or less our life. There was no real distinction between working hours and not. We talked about YC all the time. And while there might be some businesses that it would be tedious to let infect your private life, we liked it. We'd started YC because it was something we were interested in. And some of the problems we were trying to solve were endlessly difficult. How do you recognize good founders? You could talk about that for years, and we did; we still do.

I'm better at some things than Jessica, and she's better at some things than me. One of the things she's best at is judging people. She's one of those rare individuals with x-ray vision for character. She can see through any kind of faker almost immediately. Her nickname within YC was the Social Radar, and this special power of hers was critical in making YC what it is. The earlier you pick startups, the more you're picking the founders. Later stage investors get to try products and look at growth numbers. At the stage where YC invests, there is often neither a product nor any numbers.

Others thought YC had some special insight about the future of technology. Mostly we had the same sort of insight Socrates claimed: we at least knew we knew nothing. What made YC successful was being able to pick good founders. We thought Airbnb was a bad idea. We funded it because we liked the founders.

During interviews, Robert and Trevor and I would pepper the applicants with technical questions. Jessica would mostly watch. A lot of the applicants probably read her as some kind of secretary, especially early on, because she was the one who'd go out and get each new group and she didn't ask many questions. She was ok with that. It was easier for her to watch people if they didn't notice her. But after the interview, the three of us would turn to Jessica and ask "What does the Social Radar say?" [1]

Having the Social Radar at interviews wasn't just how we picked founders who'd be successful. It was also how we picked founders who were good people. At first we did this because we couldn't help it. Imagine what it would feel like to have x-ray vision for character. Being around bad people would be intolerable. So we'd refuse to fund founders whose characters we had doubts about even if we thought they'd be successful.

Though we initially did this out of self-indulgence, it turned out to be very valuable to YC. We didn't realize it in the beginning, but the people we were picking would become the YC alumni network. And once we picked them, unless they did something really egregious, they were going to be part of it for life. Some now think YC's alumni network is its most valuable feature. I personally think YC's advice is pretty good too, but the alumni network is certainly among the most valuable features. The level of trust and helpfulness is remarkable for a group of such size. And Jessica is the main reason why.

(As we later learned, it probably cost us little to reject people whose characters we had doubts about, because how good founders are and how well they do are [not orthogonal](#). If bad founders succeed at all, they tend to sell early. The most successful founders are almost all good.)

If Jessica was so important to YC, why don't more people realize it? Partly because I'm a writer, and writers always get disproportionate attention. YC's brand was initially my brand, and our applicants were people who'd read my essays. But there is another reason: Jessica hates attention. Talking to reporters makes her nervous. The thought of giving a talk paralyzes her. She was even uncomfortable at our wedding, because the bride is always the center of attention. [2]

It's not just because she's shy that she hates attention, but because it throws off the Social Radar. She can't be herself. You can't watch people when everyone is watching you.

Another reason attention worries her is that she hates bragging. In anything she does that's publicly visible, her biggest fear (after the obvious fear that it will be bad) is that it

will seem ostentatious. She says being too modest is a common problem for women. But in her case it goes beyond that. She has a horror of ostentation so visceral it's almost a phobia.

She also hates fighting. She can't do it; she just shuts down. And unfortunately there is a good deal of fighting in being the public face of an organization.

So although Jessica more than anyone made YC unique, the very qualities that enabled her to do it mean she tends to get written out of YC's history. Everyone buys this story that PG started YC and his wife just kind of helped. Even YC's haters buy it. A couple years ago when people were attacking us for not funding more female founders (than exist), they all treated YC as identical with PG. It would have spoiled the narrative to acknowledge Jessica's central role at YC.

Jessica was boiling mad that people were accusing *her* company of sexism. I've never seen her angrier about anything. But she did not contradict them. Not publicly. In private there was a great deal of profanity. And she wrote three separate essays about the question of female founders. But she could never bring herself to publish any of them. She'd seen the level of vitriol in this debate, and she shrank from engaging. [3]

It wasn't just because she disliked fighting. She's so sensitive to character that it repels her even to fight with dishonest people. The idea of mixing it up with linkbait journalists or Twitter trolls would seem to her not merely frightening, but disgusting.

But Jessica knew her example as a successful female founder would encourage more women to start companies, so last year she did something YC had never done before and hired a PR firm to get her some interviews. At one of the first she did, the reporter brushed aside her insights about startups and turned it into a sensationalistic story about how some guy had tried to chat her up as she was waiting outside the bar where they had arranged to meet. Jessica was mortified, partly because the guy had done nothing wrong, but more because the story treated her as a victim significant only for being a woman, rather than one of the most knowledgeable investors in the Valley.

After that she told the PR firm to stop.

You're not going to be hearing in the press about what Jessica has achieved. So let me tell you what Jessica has achieved. Y Combinator is fundamentally a nexus of people, like a university. It doesn't make a product. What defines it is the people. Jessica more than anyone curated and nurtured that collection of people. In that sense she literally made YC.

Jessica knows more about the qualities of startup founders than anyone else ever has. Her immense data set and x-ray vision are the perfect storm in that respect. The qualities of the founders are the best predictor of how a startup will do. And startups are in turn the most important source of growth in mature economies.

The person who knows the most about the most important factor in the growth of mature economies — that is who Jessica Livingston is. Doesn't that sound like someone who should be better known?

Notes

[1] Harj Taggar reminded me that while Jessica didn't ask many questions, they tended to be important ones:

"She was always good at sniffing out any red flags about the team or their determination and disarmingly asking the right question, which usually revealed more than the founders realized."

[2] Or more precisely, while she likes getting attention in the sense of getting credit for what she has done, she doesn't like getting attention in the sense of being watched in real time. Unfortunately, not just for her but for a lot of people, how much you get of the former depends a lot on how much you get of the latter.

Incidentally, if you saw Jessica at a public event, you would never guess she hates attention, because (a) she is very polite and (b) when she's nervous, she expresses it by smiling more.

[3] The existence of people like Jessica is not just something the mainstream media needs to learn to acknowledge, but something feminists need to learn to acknowledge as well. There are successful women who don't like to fight. Which means if the public conversation about women consists of fighting, their voices will be silenced.

There's a sort of Gresham's Law of conversations. If a conversation reaches a certain level of incivility, the more thoughtful people start to leave. No one understands female founders better than Jessica. But it's unlikely anyone will ever hear her speak candidly about the topic. She ventured a toe in that water a while ago, and the reaction was so violent that she decided "never again."

Thanks to Sam Altman, Paul Buchheit, Patrick Collison, Daniel Gackle, Carolynn Levy, Jon Levy, Kirsty Nathoo, Robert Morris, Geoff Ralston, and Harj Taggar for reading drafts of this. And yes, Jessica Livingston, who made me cut surprisingly little.

The Refragmentation

January 2016

One advantage of being old is that you can see change happen in your lifetime. A lot of the change I've seen is fragmentation. US politics is much more polarized than it used to be. Culturally we have ever less common ground. The creative class flocks to a handful of happy cities, abandoning the rest. And increasing economic inequality means the spread between rich and poor is growing too. I'd like to propose a hypothesis: that all these trends are instances of the same phenomenon. And moreover, that the cause is not some force that's pulling us apart, but rather the erosion of forces that had been pushing us together.

Worse still, for those who worry about these trends, the forces that were pushing us together were an anomaly, a one-time combination of circumstances that's unlikely to be repeated — and indeed, that we would not want to repeat.

The two forces were war (above all World War II), and the rise of large corporations.

The effects of World War II were both economic and social. Economically, it decreased variation in income. Like all modern armed forces, America's were socialist economically. From each according to his ability, to each according to his need. More or less. Higher ranking members of the military got more (as higher ranking members of socialist societies always do), but what they got was fixed according to their rank. And the flattening effect wasn't limited to those under arms, because the US economy was conscripted too. Between 1942 and 1945 all wages were set by the National War Labor Board. Like the military, they defaulted to flatness. And this national standardization of wages was so pervasive that its effects could still be seen years after the war ended. [\[1\]](#)

Business owners weren't supposed to be making money either. FDR said "not a single war millionaire" would be permitted. To ensure that, any increase in a company's profits over prewar levels was taxed at 85%. And when what was left after corporate taxes reached individuals, it was taxed again at a marginal rate of 93%. [\[2\]](#)

Socially too the war tended to decrease variation. Over 16 million men and women from all sorts of different backgrounds were brought together in a way of life that was literally uniform. Service rates for men born in the early 1920s approached 80%. And working toward a common goal, often under stress, brought them still closer together.

Though strictly speaking World War II lasted less than 4 years for the US, its effects lasted longer. Wars make central governments more powerful, and World War II was an extreme case of this. In the US, as in all the other Allied countries, the federal government was slow to give up the new powers it had acquired. Indeed, in some respects the war didn't end in 1945; the enemy just switched to the Soviet Union. In tax

rates, federal power, defense spending, conscription, and nationalism, the decades after the war looked more like wartime than prewar peacetime. [3] And the social effects lasted too. The kid pulled into the army from behind a mule team in West Virginia didn't simply go back to the farm afterward. Something else was waiting for him, something that looked a lot like the army.

If total war was the big political story of the 20th century, the big economic story was the rise of a new kind of company. And this too tended to produce both social and economic cohesion. [4]

The 20th century was the century of the big, national corporation. General Electric, General Foods, General Motors. Developments in finance, communications, transportation, and manufacturing enabled a new type of company whose goal was above all scale. Version 1 of this world was low-res: a Duplo world of a few giant companies dominating each big market. [5]

The late 19th and early 20th centuries had been a time of consolidation, led especially by J. P. Morgan. Thousands of companies run by their founders were merged into a couple hundred giant ones run by professional managers. Economies of scale ruled the day. It seemed to people at the time that this was the final state of things. John D. Rockefeller said in 1880

The day of combination is here to stay. Individualism has gone, never to return.

He turned out to be mistaken, but he seemed right for the next hundred years.

The consolidation that began in the late 19th century continued for most of the 20th. By the end of World War II, as Michael Lind writes, "the major sectors of the economy were either organized as government-backed cartels or dominated by a few oligopolistic corporations."

For consumers this new world meant the same choices everywhere, but only a few of them. When I grew up there were only 2 or 3 of most things, and since they were all aiming at the middle of the market there wasn't much to differentiate them.

One of the most important instances of this phenomenon was in TV. Here there were 3 choices: NBC, CBS, and ABC. Plus public TV for eggheads and communists. The programs that the 3 networks offered were indistinguishable. In fact, here there was a triple pressure toward the center. If one show did try something daring, local affiliates in conservative markets would make them stop. Plus since TVs were expensive, whole families watched the same shows together, so they had to be suitable for everyone.

And not only did everyone get the same thing, they got it at the same time. It's difficult to imagine now, but every night tens of millions of families would sit down together in front of their TV set watching the same show, at the same time, as their next door neighbors. What happens now with the Super Bowl used to happen every night. We were literally in sync. [6]

In a way mid-century TV culture was good. The view it gave of the world was like you'd find in a children's book, and it probably had something of the effect that (parents hope) children's books have in making people behave better. But, like children's books,

TV was also misleading. Dangerously misleading, for adults. In his autobiography, Robert MacNeil talks of seeing gruesome images that had just come in from Vietnam and thinking, we can't show these to families while they're having dinner.

I know how pervasive the common culture was, because I tried to opt out of it, and it was practically impossible to find alternatives. When I was 13 I realized, more from internal evidence than any outside source, that the ideas we were being fed on TV were crap, and I stopped watching it. [7] But it wasn't just TV. It seemed like everything around me was crap. The politicians all saying the same things, the consumer brands making almost identical products with different labels stuck on to indicate how prestigious they were meant to be, the balloon-frame houses with fake "colonial" skins, the cars with several feet of gratuitous metal on each end that started to fall apart after a couple years, the "red delicious" apples that were red but only nominally apples. And in retrospect, it *was* crap. [8]

But when I went looking for alternatives to fill this void, I found practically nothing. There was no Internet then. The only place to look was in the chain bookstore in our local shopping mall. [9] There I found a copy of *The Atlantic*. I wish I could say it became a gateway into a wider world, but in fact I found it boring and incomprehensible. Like a kid tasting whisky for the first time and pretending to like it, I preserved that magazine as carefully as if it had been a book. I'm sure I still have it somewhere. But though it was evidence that there was, somewhere, a world that wasn't red delicious, I didn't find it till college.

It wasn't just as consumers that the big companies made us similar. They did as employers too. Within companies there were powerful forces pushing people toward a single model of how to look and act. IBM was particularly notorious for this, but they were only a little more extreme than other big companies. And the models of how to look and act varied little between companies. Meaning everyone within this world was expected to seem more or less the same. And not just those in the corporate world, but also everyone who aspired to it — which in the middle of the 20th century meant most people who weren't already in it. For most of the 20th century, working-class people tried hard to look middle class. You can see it in old photos. Few adults aspired to look dangerous in 1950.

But the rise of national corporations didn't just compress us culturally. It compressed us economically too, and on both ends.

Along with giant national corporations, we got giant national labor unions. And in the mid 20th century the corporations cut deals with the unions where they paid over market price for labor. Partly because the unions were monopolies. [10] Partly because, as components of oligopolies themselves, the corporations knew they could safely pass the cost on to their customers, because their competitors would have to as well. And partly because in mid-century most of the giant companies were still focused on finding new ways to milk economies of scale. Just as startups rightly pay AWS a premium over the cost of running their own servers so they can focus on growth, many of the big national corporations were willing to pay a premium for labor. [11]

As well as pushing incomes up from the bottom, by overpaying unions, the big companies of the 20th century also pushed incomes down at the top, by underpaying their top management. Economist J. K. Galbraith wrote in 1967 that "There are few corporations in which it would be suggested that executive salaries are at a maximum."

[12]

To some extent this was an illusion. Much of the de facto pay of executives never showed up on their income tax returns, because it took the form of perks. The higher the rate of income tax, the more pressure there was to pay employees upstream of it. (In the UK, where taxes were even higher than in the US, companies would even pay their kids' private school tuitions.) One of the most valuable things the big companies of the mid 20th century gave their employees was job security, and this too didn't show up in tax returns or income statistics. So the nature of employment in these organizations tended to yield falsely low numbers about economic inequality. But even accounting for that, the big companies paid their best people less than market price. There was no market; the expectation was that you'd work for the same company for decades if not your whole career. [13]

Your work was so illiquid there was little chance of getting market price. But that same illiquidity also encouraged you not to seek it. If the company promised to employ you till you retired and give you a pension afterward, you didn't want to extract as much from it this year as you could. You needed to take care of the company so it could take care of you. Especially when you'd been working with the same group of people for decades. If you tried to squeeze the company for more money, you were squeezing the organization that was going to take care of *them*. Plus if you didn't put the company first you wouldn't be promoted, and if you couldn't switch ladders, promotion on this one was the only way up. [14]

To someone who'd spent several formative years in the armed forces, this situation didn't seem as strange as it does to us now. From their point of view, as big company executives, they were high-ranking officers. They got paid a lot more than privates. They got to have expense account lunches at the best restaurants and fly around on the company's Gulfstreams. It probably didn't occur to most of them to ask if they were being paid market price.

The ultimate way to get market price is to work for yourself, by starting your own company. That seems obvious to any ambitious person now. But in the mid 20th century it was an alien concept. Not because starting one's own company seemed too ambitious, but because it didn't seem ambitious enough. Even as late as the 1970s, when I grew up, the ambitious plan was to get lots of education at prestigious institutions, and then join some other prestigious institution and work one's way up the hierarchy. Your prestige was the prestige of the institution you belonged to. People did start their own businesses of course, but educated people rarely did, because in those days there was practically zero concept of starting what we now call a [startup](#): a business that starts small and grows big. That was much harder to do in the mid 20th century. Starting one's own business meant starting a business that would start small and stay small. Which in those days of big companies often meant scurrying around trying to avoid being trampled by elephants. It was more prestigious to be one of the executive class riding the elephant.

By the 1970s, no one stopped to wonder where the big prestigious companies had come from in the first place. It seemed like they'd always been there, like the chemical elements. And indeed, there was a double wall between ambitious kids in the 20th century and the origins of the big companies. Many of the big companies were roll-ups that didn't have clear founders. And when they did, the founders didn't seem like us. Nearly all of them had been uneducated, in the sense of not having been to college.

They were what Shakespeare called rude mechanicals. College trained one to be a member of the professional classes. Its graduates didn't expect to do the sort of grubby menial work that Andrew Carnegie or Henry Ford started out doing. [\[15\]](#)

And in the 20th century there were more and more college graduates. They increased from about 2% of the population in 1900 to about 25% in 2000. In the middle of the century our two big forces intersect, in the form of the GI Bill, which sent 2.2 million World War II veterans to college. Few thought of it in these terms, but the result of making college the canonical path for the ambitious was a world in which it was socially acceptable to work for Henry Ford, but not to be Henry Ford. [\[16\]](#)

I remember this world well. I came of age just as it was starting to break up. In my childhood it was still dominant. Not quite so dominant as it had been. We could see from old TV shows and yearbooks and the way adults acted that people in the 1950s and 60s had been even more conformist than us. The mid-century model was already starting to get old. But that was not how we saw it at the time. We would at most have said that one could be a bit more daring in 1975 than 1965. And indeed, things hadn't changed much yet.

But change was coming soon. And when the Duplo economy started to disintegrate, it disintegrated in several different ways at once. Vertically integrated companies literally dis-integrated because it was more efficient to. Incumbents faced new competitors as (a) markets went global and (b) technical innovation started to trump economies of scale, turning size from an asset into a liability. Smaller companies were increasingly able to survive as formerly narrow channels to consumers broadened. Markets themselves started to change faster, as whole new categories of products appeared. And last but not least, the federal government, which had previously smiled upon J. P. Morgan's world as the natural state of things, began to realize it wasn't the last word after all.

What J. P. Morgan was to the horizontal axis, Henry Ford was to the vertical. He wanted to do everything himself. The giant plant he built at River Rouge between 1917 and 1928 literally took in iron ore at one end and sent cars out the other. 100,000 people worked there. At the time it seemed the future. But that is not how car companies operate today. Now much of the design and manufacturing happens in a long supply chain, whose products the car companies ultimately assemble and sell. The reason car companies operate this way is that it works better. Each company in the supply chain focuses on what they know best. And they each have to do it well or they can be swapped out for another supplier.

Why didn't Henry Ford realize that networks of cooperating companies work better than a single big company? One reason is that supplier networks take a while to evolve. In 1917, doing everything himself seemed to Ford the only way to get the scale he needed. And the second reason is that if you want to solve a problem using a network of cooperating companies, you have to be able to coordinate their efforts, and you can do that much better with computers. Computers reduce the transaction costs that Coase argued are the *raison d'être* of corporations. That is a fundamental change.

In the early 20th century, big companies were synonymous with efficiency. In the late 20th century they were synonymous with inefficiency. To some extent this was because the companies themselves had become sclerotic. But it was also because our standards were higher.

It wasn't just within existing industries that change occurred. The industries themselves changed. It became possible to make lots of new things, and sometimes the existing companies weren't the ones who did it best.

Microcomputers are a classic example. The market was pioneered by upstarts like Apple. When it got big enough, IBM decided it was worth paying attention to. At the time IBM completely dominated the computer industry. They assumed that all they had to do, now that this market was ripe, was to reach out and pick it. Most people at the time would have agreed with them. But what happened next illustrated how much more complicated the world had become. IBM did launch a microcomputer. Though quite successful, it did not crush Apple. But even more importantly, IBM itself ended up being supplanted by a supplier coming in from the side — from software, which didn't even seem to be the same business. IBM's big mistake was to accept a non-exclusive license for DOS. It must have seemed a safe move at the time. No other computer manufacturer had ever been able to outsell them. What difference did it make if other manufacturers could offer DOS too? The result of that miscalculation was an explosion of inexpensive PC clones. Microsoft now owned the PC standard, and the customer. And the microcomputer business ended up being Apple vs Microsoft.

Basically, Apple bumped IBM and then Microsoft stole its wallet. That sort of thing did not happen to big companies in mid-century. But it was going to happen increasingly often in the future.

Change happened mostly by itself in the computer business. In other industries, legal obstacles had to be removed first. Many of the mid-century oligopolies had been anointed by the federal government with policies (and in wartime, large orders) that kept out competitors. This didn't seem as dubious to government officials at the time as it sounds to us. They felt a two-party system ensured sufficient competition in politics. It ought to work for business too.

Gradually the government realized that anti-competitive policies were doing more harm than good, and during the Carter administration it started to remove them. The word used for this process was misleadingly narrow: deregulation. What was really happening was de-oligopolization. It happened to one industry after another. Two of the most visible to consumers were air travel and long-distance phone service, which both became dramatically cheaper after deregulation.

Deregulation also contributed to the wave of hostile takeovers in the 1980s. In the old days the only limit on the inefficiency of companies, short of actual bankruptcy, was the inefficiency of their competitors. Now companies had to face absolute rather than relative standards. Any public company that didn't generate sufficient returns on its assets risked having its management replaced with one that would. Often the new managers did this by breaking companies up into components that were more valuable separately. [\[17\]](#)

Version 1 of the national economy consisted of a few big blocks whose relationships were negotiated in back rooms by a handful of executives, politicians, regulators, and labor leaders. Version 2 was higher resolution: there were more companies, of more different sizes, making more different things, and their relationships changed faster. In this world there were still plenty of back room negotiations, but more was left to market forces. Which further accelerated the fragmentation.

It's a little misleading to talk of versions when describing a gradual process, but not as misleading as it might seem. There was a lot of change in a few decades, and what we ended up with was qualitatively different. The companies in the S&P 500 in 1958 had been there an average of 61 years. By 2012 that number was 18 years. [18]

The breakup of the Duplo economy happened simultaneously with the spread of computing power. To what extent were computers a precondition? It would take a book to answer that. Obviously the spread of computing power was a precondition for the rise of startups. I suspect it was for most of what happened in finance too. But was it a precondition for globalization or the LBO wave? I don't know, but I wouldn't discount the possibility. It may be that the refragmentation was driven by computers in the way the industrial revolution was driven by steam engines. Whether or not computers were a precondition, they have certainly accelerated it.

The new fluidity of companies changed people's relationships with their employers. Why climb a corporate ladder that might be yanked out from under you? Ambitious people started to think of a career less as climbing a single ladder than as a series of jobs that might be at different companies. More movement (or even potential movement) between companies introduced more competition in salaries. Plus as companies became smaller it became easier to estimate how much an employee contributed to the company's revenue. Both changes drove salaries toward market price. And since people vary dramatically in productivity, paying market price meant salaries started to diverge.

By no coincidence it was in the early 1980s that the term "yuppie" was coined. That word is not much used now, because the phenomenon it describes is so taken for granted, but at the time it was a label for something novel. Yuppies were young professionals who made lots of money. To someone in their twenties today, this wouldn't seem worth naming. Why wouldn't young professionals make lots of money? But until the 1980s, being underpaid early in your career was part of what it meant to be a professional. Young professionals were paying their dues, working their way up the ladder. The rewards would come later. What was novel about yuppies was that they wanted market price for the work they were doing now.

The first yuppies did not work for startups. That was still in the future. Nor did they work for big companies. They were professionals working in fields like law, finance, and consulting. But their example rapidly inspired their peers. Once they saw that new BMW 325i, they wanted one too.

Underpaying people at the beginning of their career only works if everyone does it. Once some employer breaks ranks, everyone else has to, or they can't get good people. And once started this process spreads through the whole economy, because at the beginnings of people's careers they can easily switch not merely employers but industries.

But not all young professionals benefitted. You had to produce to get paid a lot. It was no coincidence that the first yuppies worked in fields where it was easy to measure that.

More generally, an idea was returning whose name sounds old-fashioned precisely because it was so rare for so long: that you could make your fortune. As in the past there were multiple ways to do it. Some made their fortunes by creating wealth, and others by playing zero-sum games. But once it became possible to make one's fortune, the

ambitious had to decide whether or not to. A physicist who chose physics over Wall Street in 1990 was making a sacrifice that a physicist in 1960 didn't have to think about.

The idea even flowed back into big companies. CEOs of big companies make more now than they used to, and I think much of the reason is prestige. In 1960, corporate CEOs had immense prestige. They were the winners of the only economic game in town. But if they made as little now as they did then, in real dollar terms, they'd seem like small fry compared to professional athletes and whiz kids making millions from startups and hedge funds. They don't like that idea, so now they try to get as much as they can, which is more than they had been getting. [\[19\]](#)

Meanwhile a similar fragmentation was happening at the other end of the economic scale. As big companies' oligopolies became less secure, they were less able to pass costs on to customers and thus less willing to overpay for labor. And as the Duplo world of a few big blocks fragmented into many companies of different sizes — some of them overseas — it became harder for unions to enforce their monopolies. As a result workers' wages also tended toward market price. Which (inevitably, if unions had been doing their job) tended to be lower. Perhaps dramatically so, if automation had decreased the need for some kind of work.

And just as the mid-century model induced social as well as economic cohesion, its breakup brought social as well as economic fragmentation. People started to dress and act differently. Those who would later be called the "creative class" became more mobile. People who didn't care much for religion felt less pressure to go to church for appearances' sake, while those who liked it a lot opted for increasingly colorful forms. Some switched from meat loaf to tofu, and others to Hot Pockets. Some switched from driving Ford sedans to driving small imported cars, and others to driving SUVs. Kids who went to private schools or wished they did started to dress "preppy," and kids who wanted to seem rebellious made a conscious effort to look disreputable. In a hundred ways people spread apart. [\[20\]](#)

Almost four decades later, fragmentation is still increasing. Has it been net good or bad? I don't know; the question may be unanswerable. Not entirely bad though. We take for granted the forms of fragmentation we like, and worry only about the ones we don't. But as someone who caught the tail end of mid-century [conformism](#), I can tell you it was no utopia. [\[21\]](#)

My goal here is not to say whether fragmentation has been good or bad, just to explain why it's happening. With the centripetal forces of total war and 20th century oligopoly mostly gone, what will happen next? And more specifically, is it possible to reverse some of the fragmentation we've seen?

If it is, it will have to happen piecemeal. You can't reproduce mid-century cohesion the way it was originally produced. It would be insane to go to war just to induce more national unity. And once you understand the degree to which the economic history of the 20th century was a low-res version 1, it's clear you can't reproduce that either.

20th century cohesion was something that happened at least in a sense naturally. The war was due mostly to external forces, and the Duplo economy was an evolutionary phase. If you want cohesion now, you'd have to induce it deliberately. And it's not obvious how. I suspect the best we'll be able to do is address the symptoms of fragmentation. But that may be enough.

The form of fragmentation people worry most about lately is [economic inequality](#), and if you want to eliminate that you're up against a truly formidable headwind that has been in operation since the stone age. Technology.

Technology is a lever. It magnifies work. And the lever not only grows increasingly long, but the rate at which it grows is itself increasing.

Which in turn means the variation in the amount of wealth people can create has not only been increasing, but accelerating. The unusual conditions that prevailed in the mid 20th century masked this underlying trend. The ambitious had little choice but to join large organizations that made them march in step with lots of other people — literally in the case of the armed forces, figuratively in the case of big corporations. Even if the big corporations had wanted to pay people proportionate to their value, they couldn't have figured out how. But that constraint has gone now. Ever since it started to erode in the 1970s, we've seen the underlying forces at work again. [\[22\]](#)

Not everyone who gets rich now does it by creating wealth, certainly. But a significant number do, and the Baumol Effect means all their peers get dragged along too. [\[23\]](#) And as long as it's possible to get rich by creating wealth, the default tendency will be for economic inequality to increase. Even if you eliminate all the other ways to get rich. You can mitigate this with subsidies at the bottom and taxes at the top, but unless taxes are high enough to discourage people from creating wealth, you're always going to be fighting a losing battle against increasing variation in productivity. [\[24\]](#)

That form of fragmentation, like the others, is here to stay. Or rather, back to stay. Nothing is forever, but the tendency toward fragmentation should be more forever than most things, precisely because it's not due to any particular cause. It's simply a reversion to the mean. When Rockefeller said individualism was gone, he was right for a hundred years. It's back now, and that's likely to be true for longer.

I worry that if we don't acknowledge this, we're headed for trouble. If we think 20th century cohesion disappeared because of few policy tweaks, we'll be deluded into thinking we can get it back (minus the bad parts, somehow) with a few countertweaks. And then we'll waste our time trying to eliminate fragmentation, when we'd be better off thinking about how to mitigate its consequences.

Notes

[1] Lester Thurow, writing in 1975, said the wage differentials prevailing at the end of World War II had become so embedded that they "were regarded as 'just' even after the egalitarian pressures of World War II had disappeared. Basically, the same differentials exist to this day, thirty years later." But Goldin and Margo think market forces in the postwar period also helped preserve the wartime compression of wages — specifically increased demand for unskilled workers, and oversupply of educated ones.

(Oddly enough, the American custom of having employers pay for health insurance derives from efforts by businesses to circumvent NWLB wage controls in order to attract workers.)

[2] As always, tax rates don't tell the whole story. There were lots of exemptions, especially for individuals. And in World War II the tax codes were so new that the government had little acquired immunity to tax avoidance. If the rich paid high taxes during the war it was more because they wanted to than because they had to.

After the war, federal tax receipts as a percentage of GDP were about the same as they are now. In fact, for the entire period since the war, tax receipts have stayed close to 18% of GDP, despite dramatic changes in tax rates. The lowest point occurred when marginal income tax rates were highest: 14.1% in 1950. Looking at the data, it's hard to avoid the conclusion that tax rates have had little effect on what people actually paid.

[3] Though in fact the decade preceding the war had been a time of unprecedented federal power, in response to the Depression. Which is not entirely a coincidence, because the Depression was one of the causes of the war. In many ways the New Deal was a sort of dress rehearsal for the measures the federal government took during wartime. The wartime versions were much more drastic and more pervasive though. As Anthony Badger wrote, "for many Americans the decisive change in their experiences came not with the New Deal but with World War II."

[4] I don't know enough about the origins of the world wars to say, but it's not inconceivable they were connected to the rise of big corporations. If that were the case, 20th century cohesion would have a single cause.

[5] More precisely, there was a bimodal economy consisting, in Galbraith's words, of "the world of the technically dynamic, massively capitalized and highly organized corporations on the one hand and the hundreds of thousands of small and traditional proprietors on the other." Money, prestige, and power were concentrated in the former, and there was near zero crossover.

[6] I wonder how much of the decline in families eating together was due to the decline in families watching TV together afterward.

[7] I know when this happened because it was the season Dallas premiered. Everyone else was talking about what was happening on Dallas, and I had no idea what they meant.

[8] I didn't realize it till I started doing research for this essay, but the meretriciousness of the products I grew up with is a well-known byproduct of oligopoly. When companies can't compete on price, they compete on tailfins.

[9] Monroeville Mall was at the time of its completion in 1969 the largest in the country. In the late 1970s the movie *Dawn of the Dead* was shot there. Apparently the mall was not just the location of the movie, but its inspiration; the crowds of shoppers drifting through this huge mall reminded George Romero of zombies. My first job was scooping ice cream in the Baskin-Robbins.

[10] Labor unions were exempted from antitrust laws by the Clayton Antitrust Act in 1914 on the grounds that a person's work is not "a commodity or article of commerce."

I wonder if that means service companies are also exempt.

[11] The relationships between unions and unionized companies can even be symbiotic, because unions will exert political pressure to protect their hosts. According to Michael Lind, when politicians tried to attack the A&P supermarket chain because it was putting local grocery stores out of business, "A&P successfully defended itself by allowing the unionization of its workforce in 1938, thereby gaining organized labor as a constituency." I've seen this phenomenon myself: hotel unions are responsible for more of the political pressure against Airbnb than hotel companies.

[12] Galbraith was clearly puzzled that corporate executives would work so hard to make money for other people (the shareholders) instead of themselves. He devoted much of *The New Industrial State* to trying to figure this out.

His theory was that professionalism had replaced money as a motive, and that modern corporate executives were, like (good) scientists, motivated less by financial rewards than by the desire to do good work and thereby earn the respect of their peers. There is something in this, though I think lack of movement between companies combined with self-interest explains much of observed behavior.

[13] Galbraith (p. 94) says a 1952 study of the 800 highest paid executives at 300 big corporations found that three quarters of them had been with their company for more than 20 years.

[14] It seems likely that in the first third of the 20th century executive salaries were low partly because companies then were more dependent on banks, who would have disapproved if executives got too much. This was certainly true in the beginning. The first big company CEOs were J. P. Morgan's hired hands.

Companies didn't start to finance themselves with retained earnings till the 1920s. Till then they had to pay out their earnings in dividends, and so depended on banks for capital for expansion. Bankers continued to sit on corporate boards till the Glass-Steagall act in 1933.

By mid-century big companies funded 3/4 of their growth from earnings. But the early years of bank dependence, reinforced by the financial controls of World War II, must have had a big effect on social conventions about executive salaries. So it may be that the lack of movement between companies was as much the effect of low salaries as the cause.

Incidentally, the switch in the 1920s to financing growth with retained earnings was one cause of the 1929 crash. The banks now had to find someone else to lend to, so they made more margin loans.

[15] Even now it's hard to get them to. One of the things I find hardest to get into the heads of would-be startup founders is how important it is to do certain kinds of menial work early in the life of a company. Doing [things that don't scale](#) is to how Henry Ford got started as a high-fiber diet is to the traditional peasant's diet: they had no choice but to do the right thing, while we have to make a conscious effort.

[16] Founders weren't celebrated in the press when I was a kid. "Our founder" meant a photograph of a severe-looking man with a walrus mustache and a wing collar who had

died decades ago. The thing to be when I was a kid was an *executive*. If you weren't around then it's hard to grasp the cachet that term had. The fancy version of everything was called the "executive" model.

[17] The wave of hostile takeovers in the 1980s was enabled by a combination of circumstances: court decisions striking down state anti-takeover laws, starting with the Supreme Court's 1982 decision in *Edgar v. MITE Corp.*; the Reagan administration's comparatively sympathetic attitude toward takeovers; the Depository Institutions Act of 1982, which allowed banks and savings and loans to buy corporate bonds; a new SEC rule issued in 1982 (rule 415) that made it possible to bring corporate bonds to market faster; the creation of the junk bond business by Michael Milken; a vogue for conglomerates in the preceding period that caused many companies to be combined that never should have been; a decade of inflation that left many public companies trading below the value of their assets; and not least, the increasing complacency of managements.

[18] Foster, Richard. "Creative Destruction Whips through Corporate America." *Innosight*, February 2012.

[19] CEOs of big companies may be overpaid. I don't know enough about big companies to say. But it is certainly not impossible for a CEO to make 200x as much difference to a company's revenues as the average employee. Look at what Steve Jobs did for Apple when he came back as CEO. It would have been a good deal for the board to give him 95% of the company. Apple's market cap the day Steve came back in July 1997 was 1.73 billion. 5% of Apple now (January 2016) would be worth about 30 billion. And it would not be if Steve hadn't come back; Apple probably wouldn't even exist anymore.

Merely including Steve in the sample might be enough to answer the question of whether public company CEOs in the aggregate are overpaid. And that is not as facile a trick as it might seem, because the broader your holdings, the more the aggregate is what you care about.

[20] The late 1960s were famous for social upheaval. But that was more rebellion (which can happen in any era if people are provoked sufficiently) than fragmentation. You're not seeing fragmentation unless you see people breaking off to both left and right.

[21] Globally the trend has been in the other direction. While the US is becoming more fragmented, the world as a whole is becoming less fragmented, and mostly in good ways.

[22] There were a handful of ways to make a fortune in the mid 20th century. The main one was drilling for oil, which was open to newcomers because it was not something big companies could dominate through economies of scale. How did individuals accumulate large fortunes in an era of such high taxes? Giant tax loopholes defended by two of the most powerful men in Congress, Sam Rayburn and Lyndon Johnson.

But becoming a Texas oilman was not in 1950 something one could aspire to the way starting a startup or going to work on Wall Street were in 2000, because (a) there was a strong local component and (b) success depended so much on luck.

[23] The Baumol Effect induced by startups is very visible in Silicon Valley. Google will pay people millions of dollars a year to keep them from leaving to start or join startups.

[24] I'm not claiming variation in productivity is the only cause of economic inequality in the US. But it's a significant cause, and it will become as big a cause as it needs to, in the sense that if you ban other ways to get rich, people who want to get rich will use this route instead.

Thanks to Sam Altman, Trevor Blackwell, Paul Buchheit, Patrick Collison, Ron Conway, Chris Dixon, Benedict Evans, Richard Florida, Ben Horowitz, Jessica Livingston, Robert Morris, Tim O'Reilly, Geoff Ralston, Max Roser, Alexia Tsotsis, and Qasar Younis for reading drafts of this. Max also told me about several valuable sources.

Bibliography

Allen, Frederick Lewis. *The Big Change*. Harper, 1952.

Averitt, Robert. *The Dual Economy*. Norton, 1968.

Badger, Anthony. *The New Deal*. Hill and Wang, 1989.

Bainbridge, John. *The Super-Americans*. Doubleday, 1961.

Beatty, Jack. *Colossus*. Broadway, 2001.

Brinkley, Douglas. *Wheels for the World*. Viking, 2003.

Brownlee, W. Elliot. *Federal Taxation in America*. Cambridge, 1996.

Chandler, Alfred. *The Visible Hand*. Harvard, 1977.

Chernow, Ron. *The House of Morgan*. Simon & Schuster, 1990.

Chernow, Ron. *Titan: The Life of John D. Rockefeller*. Random House, 1998.

Galbraith, John. *The New Industrial State*. Houghton Mifflin, 1967.

Goldin, Claudia and Robert A. Margo. "The Great Compression: The Wage Structure in the United States at Mid-Century." NBER Working Paper 3817, 1991.

Gordon, John. *An Empire of Wealth*. HarperCollins, 2004.

Klein, Maury. *The Genesis of Industrial America, 1870-1920*. Cambridge, 2007.

Lind, Michael. *Land of Promise*. HarperCollins, 2012.

Mickelthwaite, John, and Adrian Wooldridge. *The Company*. Modern Library, 2003.

Nasaw, David. *Andrew Carnegie*. Penguin, 2006.

Sobel, Robert. *The Age of Giant Corporations*. Praeger, 1993.

Thurow, Lester. *Generating Inequality: Mechanisms of Distribution*. Basic Books, 1975.

Witte, John. *The Politics and Development of the Federal Income Tax*. Wisconsin, 1985.

Related:

- [Too Many Elite American Men Are Obsessed With Work and Wealth](#)

Economic Inequality

January 2016

Since the 1970s, economic inequality in the US has increased dramatically. And in particular, the rich have gotten a lot richer. Nearly everyone who writes about the topic says that economic inequality should be decreased.

I'm interested in this question because I was one of the founders of a company called Y Combinator that helps people start startups. Almost by definition, if a startup succeeds, its founders become rich. Which means by helping startup founders I've been helping to increase economic inequality. If economic inequality should be decreased, I shouldn't be helping founders. No one should be.

But that doesn't sound right. What's going on here? What's going on is that while economic inequality is a single measure (or more precisely, two: variation in income, and variation in wealth), it has multiple causes. Many of these causes are bad, like tax loopholes and drug addiction. But some are good, like Larry Page and Sergey Brin starting the company you use to find things online.

If you want to understand economic inequality — and more importantly, if you actually want to fix the bad aspects of it — you have to tease apart the components. And yet the trend in nearly everything written about the subject is to do the opposite: to squash together all the aspects of economic inequality as if it were a single phenomenon.

Sometimes this is done for ideological reasons. Sometimes it's because the writer only has very high-level data and so draws conclusions from that, like the proverbial drunk who looks for his keys under the lamppost, instead of where he dropped them, because the light is better there. Sometimes it's because the writer doesn't understand critical aspects of inequality, like the role of technology in wealth creation. Much of the time, perhaps most of the time, writing about economic inequality combines all three.

The most common mistake people make about economic inequality is to treat it as a single phenomenon. The most naive version of which is the one based on the pie fallacy: that the rich get rich by taking money from the poor.

Usually this is an assumption people start from rather than a conclusion they arrive at by examining the evidence. Sometimes the pie fallacy is stated explicitly:

...those at the top are grabbing an increasing fraction of the nation's income — so much of a larger share that what's left over for the rest is diminished.... [1]

Other times it's more unconscious. But the unconscious form is very widespread. I think because we grow up in a world where the pie fallacy is actually true. To kids, wealth *is* a fixed pie that's shared out, and if one person gets more, it's at the expense of another. It takes a conscious effort to remind oneself that the real world doesn't work that way.

In the real world you can create wealth as well as taking it from others. A woodworker creates wealth. He makes a chair, and you willingly give him money in return for it. A high-frequency trader does not. He makes a dollar only when someone on the other end of a trade loses a dollar.

If the rich people in a society got that way by taking wealth from the poor, then you have the degenerate case of economic inequality, where the cause of poverty is the same as the cause of wealth. But instances of inequality don't have to be instances of the degenerate case. If one woodworker makes 5 chairs and another makes none, the second woodworker will have less money, but not because anyone took anything from him.

Even people sophisticated enough to know about the pie fallacy are led toward it by the custom of describing economic inequality as a ratio of one quantile's income or wealth to another's. It's so easy to slip from talking about income shifting from one quantile to another, as a figure of speech, into believing that is literally what's happening.

Except in the degenerate case, economic inequality can't be described by a ratio or even a curve. In the general case it consists of multiple ways people become poor, and multiple ways people become rich. Which means to understand economic inequality in a country, you have to go find individual people who are poor or rich and figure out why. [2]

If you want to understand *change* in economic inequality, you should ask what those people would have done when it was different. This is one way I know the rich aren't all getting richer simply from some new system for transferring wealth to them from everyone else. When you use the would-have method with startup founders, you find what most would have done [back in 1960](#), when economic inequality was lower, was to join big companies or become professors. Before Mark Zuckerberg started Facebook, his default expectation was that he'd end up working at Microsoft. The reason he and most other startup founders are richer than they would have been in the mid 20th century is not because of some right turn the country took during the Reagan administration, but because progress in technology has made it much easier to start a new company that [grows fast](#).

Traditional economists seem strangely averse to studying individual humans. It seems to be a rule with them that everything has to start with statistics. So they give you very precise numbers about variation in wealth and income, then follow it with the most naive speculation about the underlying causes.

But while there are a lot of people who get rich through rent-seeking of various forms, and a lot who get rich by playing zero-sum games, there are also a significant number who get rich by creating wealth. And creating wealth, as a source of economic

inequality, is different from taking it — not just morally, but also practically, in the sense that it is harder to eradicate. One reason is that variation in productivity is accelerating. The rate at which individuals can create wealth depends on the technology available to them, and that grows exponentially. The other reason creating wealth is such a tenacious source of inequality is that it can expand to accommodate a lot of people.

I'm all for shutting down the crooked ways to get rich. But that won't eliminate great variations in wealth, because as long as you leave open the option of getting rich by creating wealth, people who want to get rich will do that instead.

Most people who get rich tend to be fairly driven. Whatever their other flaws, laziness is usually not one of them. Suppose new policies make it hard to make a fortune in finance. Does it seem plausible that the people who currently go into finance to make their fortunes will continue to do so, but be content to work for ordinary salaries? The reason they go into finance is not because they love finance but because they want to get rich. If the only way left to get rich is to start startups, they'll start startups. They'll do well at it too, because determination is the main factor in the success of a startup. [3] And while it would probably be a good thing for the world if people who wanted to get rich switched from playing zero-sum games to creating wealth, that would not only not eliminate great variations in wealth, but might even exacerbate them. In a zero-sum game there is at least a limit to the upside. Plus a lot of the new startups would create new technology that further accelerated variation in productivity.

Variation in productivity is far from the only source of economic inequality, but it is the irreducible core of it, in the sense that you'll have that left when you eliminate all other sources. And if you do, that core will be big, because it will have expanded to include the efforts of all the refugees. Plus it will have a large Baumol penumbra around it: anyone who could get rich by creating wealth on their own account will have to be paid enough to prevent them from doing it.

You can't prevent great variations in wealth without preventing people from getting rich, and you can't do that without preventing them from starting startups.

So let's be clear about that. Eliminating great variations in wealth would mean eliminating startups. And that doesn't seem a wise move. Especially since it would only mean you eliminated startups in your own country. Ambitious people already move halfway around the world to further their careers, and startups can operate from anywhere nowadays. So if you made it impossible to get rich by creating wealth in your country, people who wanted to do that would just leave and do it somewhere else. Which would certainly get you a lower Gini coefficient, along with a lesson in being careful what you ask for. [4]

I think rising economic inequality is the inevitable fate of countries that don't choose something worse. We had a 40 year stretch in the middle of the 20th century that convinced some people otherwise. But as I explained in [The Refragmentation](#), that was an anomaly — a unique combination of circumstances that compressed American society not just economically but culturally too. [5]

And while some of the growth in economic inequality we've seen since then has been

due to bad behavior of various kinds, there has simultaneously been a huge increase in individuals' ability to create wealth. Startups are almost entirely a product of this period. And even within the startup world, there has been a qualitative change in the last 10 years. Technology has decreased the cost of starting a startup so much that founders now have the upper hand over investors. Founders get less diluted, and it is now common for them to retain [board control](#) as well. Both further increase economic inequality, the former because founders own more stock, and the latter because, as investors have learned, founders tend to be better at running their companies than investors.

While the surface manifestations change, the underlying forces are very, very old. The acceleration of productivity we see in Silicon Valley has been happening for thousands of years. If you look at the history of stone tools, technology was already accelerating in the Mesolithic. The acceleration would have been too slow to perceive in one lifetime. Such is the nature of the leftmost part of an exponential curve. But it was the same curve.

You do not want to design your society in a way that's incompatible with this curve. The evolution of technology is one of the most powerful forces in history.

Louis Brandeis said "We may have democracy, or we may have wealth concentrated in the hands of a few, but we can't have both." That sounds plausible. But if I have to choose between ignoring him and ignoring an exponential curve that has been operating for thousands of years, I'll bet on the curve. Ignoring any trend that has been operating for thousands of years is dangerous. But exponential growth, especially, tends to bite you.

If accelerating variation in productivity is always going to produce some baseline growth in economic inequality, it would be a good idea to spend some time thinking about that future. Can you have a healthy society with great variation in wealth? What would it look like?

Notice how novel it feels to think about that. The public conversation so far has been exclusively about the need to decrease economic inequality. We've barely given a thought to how to live with it.

I'm hopeful we'll be able to. Brandeis was a product of the Gilded Age, and things have changed since then. It's harder to hide wrongdoing now. And to get rich now you don't have to buy politicians the way railroad or oil magnates did. [\[6\]](#) The great concentrations of wealth I see around me in Silicon Valley don't seem to be destroying democracy.

There are lots of things wrong with the US that have economic inequality as a symptom. We should fix those things. In the process we may decrease economic inequality. But we can't start from the symptom and hope to fix the underlying causes. [\[7\]](#)

The most obvious is poverty. I'm sure most of those who want to decrease economic inequality want to do it mainly to help the poor, not to hurt the rich. [\[8\]](#) Indeed, a good

number are merely being sloppy by speaking of decreasing economic inequality when what they mean is decreasing poverty. But this is a situation where it would be good to be precise about what we want. Poverty and economic inequality are not identical. When the city is turning off your [water](#) because you can't pay the bill, it doesn't make any difference what Larry Page's net worth is compared to yours. He might only be a few times richer than you, and it would still be just as much of a problem that your water was getting turned off.

Closely related to poverty is lack of social mobility. I've seen this myself: you don't have to grow up rich or even upper middle class to get rich as a startup founder, but few successful founders grew up desperately poor. But again, the problem here is not simply economic inequality. There is an enormous difference in wealth between the household Larry Page grew up in and that of a successful startup founder, but that didn't prevent him from joining their ranks. It's not economic inequality per se that's blocking social mobility, but some specific combination of things that go wrong when kids grow up sufficiently poor.

One of the most important principles in Silicon Valley is that "you make what you measure." It means that if you pick some number to focus on, it will tend to improve, but that you have to choose the right number, because only the one you choose will improve; another that seems conceptually adjacent might not. For example, if you're a university president and you decide to focus on graduation rates, then you'll improve graduation rates. But only graduation rates, not how much students learn. Students could learn less, if to improve graduation rates you made classes easier.

Economic inequality is sufficiently far from identical with the various problems that have it as a symptom that we'll probably only hit whichever of the two we aim at. If we aim at economic inequality, we won't fix these problems. So I say let's aim at the problems.

For example, let's attack poverty, and if necessary damage wealth in the process. That's much more likely to work than attacking wealth in the hope that you will thereby fix poverty. [\[9\]](#) And if there are people getting rich by tricking consumers or lobbying the government for anti-competitive regulations or tax loopholes, then let's stop them. Not because it's causing economic inequality, but because it's stealing. [\[10\]](#)

If all you have is statistics, it seems like that's what you need to fix. But behind a broad statistical measure like economic inequality there are some things that are good and some that are bad, some that are historical trends with immense momentum and others that are random accidents. If we want to fix the world behind the statistics, we have to understand it, and focus our efforts where they'll do the most good.

Notes

[1] Stiglitz, Joseph. *The Price of Inequality*. Norton, 2012. p. 32.

[2] Particularly since economic inequality is a matter of outliers, and outliers are disproportionately likely to have gotten where they are by ways that have little to do with the sort of things economists usually think about, like wages and productivity, but rather by, say, ending up on the wrong side of the "War on Drugs."

[3] Determination is the most important factor in deciding between success and failure, which in startups tend to be sharply differentiated. But it takes more than determination to create one of the hugely successful startups. Though most founders start out excited about the idea of getting rich, purely mercenary founders will usually take one of the big acquisition offers most successful startups get on the way up. The founders who go on to the next stage tend to be driven by a sense of mission. They have the same attachment to their companies that an artist or writer has to their work. But it is very hard to predict at the outset which founders will do that. It's not simply a function of their initial attitude. Starting a company changes people.

[4] After reading a draft of this essay, Richard Florida told me how he had once talked to a group of Europeans "who said they wanted to make Europe more entrepreneurial and more like Silicon Valley. I said by definition this will give you more inequality. They thought I was insane — they could not process it."

[5] Economic inequality has been decreasing globally. But this is mainly due to the erosion of the kleptocracies that formerly dominated all the poorer countries. Once the playing field is leveler politically, we'll see economic inequality start to rise again. The US is the bellwether. The situation we face here, the rest of the world will sooner or later.

[6] Some people still get rich by buying politicians. My point is that it's no longer a precondition.

[7] As well as problems that have economic inequality as a symptom, there are those that have it as a cause. But in most if not all, economic inequality is not the primary cause. There is usually some injustice that is allowing economic inequality to turn into other forms of inequality, and that injustice is what we need to fix. For example, the police in the US treat the poor worse than the rich. But the solution is not to make people richer. It's to make the police treat people more equitably. Otherwise they'll continue to maltreat people who are weak in other ways.

[8] Some who read this essay will say that I'm clueless or even being deliberately misleading by focusing so much on the richer end of economic inequality — that economic inequality is really about poverty. But that is exactly the point I'm making, though sloppier language than I'd use to make it. The real problem is poverty, not economic inequality. And if you conflate them you're aiming at the wrong target.

Others will say I'm clueless or being misleading by focusing on people who get rich by creating wealth — that startups aren't the problem, but corrupt practices in finance, healthcare, and so on. Once again, that is exactly my point. The problem is not economic inequality, but those specific abuses.

It's a strange task to write an essay about why something isn't the problem, but that's the situation you find yourself in when so many people mistakenly think it is.

[9] Particularly since many causes of poverty are only partially driven by people trying to make money from them. For example, America's abnormally high incarceration rate is a major cause of poverty. But although [for-profit prison companies](#) and [prison guard unions](#) both spend a lot lobbying for harsh sentencing laws, they are not the original source of them.

[10] Incidentally, tax loopholes are definitely not a product of some power shift due to recent increases in economic inequality. The golden age of economic equality in the mid 20th century was also the golden age of tax avoidance. Indeed, it was so widespread and so effective that I'm skeptical whether economic inequality was really so low then as we think. In a period when people are trying to hide wealth from the government, it will tend to be hidden from statistics too. One sign of the potential magnitude of the problem is the discrepancy between government receipts as a percentage of GDP, which have remained more or less constant during the entire period from the end of World War II to the present, and tax rates, which have varied dramatically.

Thanks to Sam Altman, Tiffani Ashley Bell, Patrick Collison, Ron Conway, Richard Florida, Ben Horowitz, Jessica Livingston, Robert Morris, Tim O'Reilly, Max Roser, and Alexia Tsotsis for reading drafts of this.

Note: This is a new version from which I removed a pair of metaphors that made a lot of people mad, essentially by macroexpanding them. If anyone wants to see the old version, I put it [here](#).

Related:

- [The Short Version](#)
- [A Reply to Ezra Klein](#)
- [A Reply to Russell Okung](#)
- [French Translation](#)

Life is Short

January 2016

Life is short, as everyone knows. When I was a kid I used to wonder about this. Is life actually short, or are we really complaining about its finiteness? Would we be just as likely to feel life was short if we lived 10 times as long?

Since there didn't seem any way to answer this question, I stopped wondering about it. Then I had kids. That gave me a way to answer the question, and the answer is that life actually is short.

Having kids showed me how to convert a continuous quantity, time, into discrete quantities. You only get 52 weekends with your 2 year old. If Christmas-as-magic lasts from say ages 3 to 10, you only get to watch your child experience it 8 times. And while it's impossible to say what is a lot or a little of a continuous quantity like time, 8 is not a lot of something. If you had a handful of 8 peanuts, or a shelf of 8 books to choose from, the quantity would definitely seem limited, no matter what your lifespan was.

Ok, so life actually is short. Does it make any difference to know that?

It has for me. It means arguments of the form "Life is too short for x" have great force. It's not just a figure of speech to say that life is too short for something. It's not just a synonym for annoying. If you find yourself thinking that life is too short for something, you should try to eliminate it if you can.

When I ask myself what I've found life is too short for, the word that pops into my head is "bullshit." I realize that answer is somewhat tautological. It's almost the definition of bullshit that it's the stuff that life is too short for. And yet bullshit does have a distinctive character. There's something fake about it. It's the junk food of experience. [\[1\]](#)


If you ask yourself what you spend your time on that's bullshit, you probably already know the answer. Unnecessary meetings, pointless disputes, bureaucracy, posturing, dealing with other people's mistakes, traffic jams, addictive but unrewarding pastimes.

There are two ways this kind of thing gets into your life: it's either forced on you, or it tricks you. To some extent you have to put up with the bullshit forced on you by circumstances. You need to make money, and making money consists mostly of errands. Indeed, the law of supply and demand ensures that: the more rewarding some kind of work is, the cheaper people will do it. It may be that less bullshit is forced on you than you think, though. There has always been a stream of people who opt out of the default grind and go live somewhere where opportunities are fewer in the conventional sense, but life feels more authentic. This could become more common.

You can do it on a smaller scale without moving. The amount of time you have to spend on bullshit varies between employers. Most large organizations (and many small ones) are steeped in it. But if you consciously prioritize bullshit avoidance over other factors like money and prestige, you can probably find employers that will waste less of your time.

If you're a freelancer or a small company, you can do this at the level of individual customers. If you fire or avoid toxic customers, you can decrease the amount of bullshit in your life by more than you decrease your income.

But while some amount of bullshit is inevitably forced on you, the bullshit that sneaks into your life by tricking you is no one's fault but your own. And yet the bullshit you choose may be harder to eliminate than the bullshit that's forced on you. Things that lure you into wasting your time have to be really good at tricking you. An example that will be familiar to a lot of people is arguing online. When someone contradicts you, they're in a sense attacking you. Sometimes pretty overtly. Your instinct when attacked is to defend yourself. But like a lot of instincts, this one wasn't designed for the world we now live in. Counterintuitive as it feels, it's better most of the time not to defend yourself. Otherwise these people are literally taking your life. [2]

Arguing online is only incidentally addictive. There are more dangerous things than that. As I've written before, one byproduct of technical progress is that things we like tend to become [more addictive](#). Which means we will increasingly have to make a conscious effort to avoid addictions  to stand outside ourselves and ask "is this how I want to be spending my time?"

As well as avoiding bullshit, one should actively seek out things that matter. But different things matter to different people, and most have to learn what matters to them. A few are lucky and realize early on that they love math or taking care of animals or writing, and then figure out a way to spend a lot of time doing it. But most people start out with a life that's a mix of things that matter and things that don't, and only gradually learn to distinguish between them.

For the young especially, much of this confusion is induced by the artificial situations they find themselves in. In middle school and high school, what the other kids think of you seems the most important thing in the world. But when you ask adults what they got wrong at that age, nearly all say they cared too much what other kids thought of them.

One heuristic for distinguishing stuff that matters is to ask yourself whether you'll care about it in the future. Fake stuff that matters usually has a sharp peak of seeming to matter. That's how it tricks you. The area under the curve is small, but its shape jabs into your consciousness like a pin.

The things that matter aren't necessarily the ones people would call "important." Having coffee with a friend matters. You won't feel later like that was a waste of time.

One great thing about having small children is that they make you spend time on things that matter: them. They grab your sleeve as you're staring at your phone and say "will you play with me?" And odds are that is in fact the bullshit-minimizing option.

If life is short, we should expect its shortness to take us by surprise. And that is just what

tends to happen. You take things for granted, and then they're gone. You think you can always write that book, or climb that mountain, or whatever, and then you realize the window has closed. The saddest windows close when other people die. Their lives are short too. After my mother died, I wished I'd spent more time with her. I lived as if she'd always be there. And in her typical quiet way she encouraged that illusion. But an illusion it was. I think a lot of people make the same mistake I did.

The usual way to avoid being taken by surprise by something is to be consciously aware of it. Back when life was more precarious, people used to be aware of death to a degree that would now seem a bit morbid. I'm not sure why, but it doesn't seem the right answer to be constantly reminding oneself of the grim reaper hovering at everyone's shoulder. Perhaps a better solution is to look at the problem from the other end. Cultivate a habit of impatience about the things you most want to do. Don't wait before climbing that mountain or writing that book or visiting your mother. You don't need to be constantly reminding yourself why you shouldn't wait. Just don't wait.

I can think of two more things one does when one doesn't have much of something: try to get more of it, and savor what one has. Both make sense here.

How you live affects how long you live. Most people could do better. Me among them.

But you can probably get even more effect by paying closer attention to the time you have. It's easy to let the days rush by. The "flow" that imaginative people love so much has a darker cousin that prevents you from pausing to savor life amid the daily slurry of errands and alarms. One of the most striking things I've read was not in a book, but the title of one: James Salter's *Burning the Days*.

It is possible to slow time somewhat. I've gotten better at it. Kids help. When you have small children, there are a lot of moments so perfect that you can't help noticing.

It does help too to feel that you've squeezed everything out of some experience. The reason I'm sad about my mother is not just that I miss her but that I think of all the things we could have done that we didn't. My oldest son will be 7 soon. And while I miss the 3 year old version of him, I at least don't have any regrets over what might have been. We had the best time a daddy and a 3 year old ever had.

Relentlessly prune bullshit, don't wait to do things that matter, and savor the time you have. That's what you do when life is short.

Notes

[1] At first I didn't like it that the word that came to mind was one that had other meanings. But then I realized the other meanings are fairly closely related. Bullshit in the sense of things you waste your time on is a lot like intellectual bullshit.

[2] I chose this example deliberately as a note to self. I get attacked a lot online. People

tell the craziest lies about me. And I have so far done a pretty mediocre job of suppressing the natural human inclination to say "Hey, that's not true!"

Thanks to Jessica Livingston and Geoff Ralston for reading drafts of this.

- [Korean Translation](#)
- [Japanese Translation](#)
- [Chinese Translation](#)

How to Make Pittsburgh a Startup Hub

April 2016

(This is a talk I gave at an event called Opt412 in Pittsburgh. Much of it will apply to other towns. But not all, because as I say in the talk, Pittsburgh has some important advantages over most would-be startup hubs.)

What would it take to make Pittsburgh into a startup hub, like Silicon Valley? I understand Pittsburgh pretty well, because I grew up here, in Monroeville. And I understand Silicon Valley pretty well because that's where I live now. Could you get that kind of startup ecosystem going here?

When I agreed to speak here, I didn't think I'd be able to give a very optimistic talk. I thought I'd be talking about what Pittsburgh could do to become a startup hub, very much in the subjunctive. Instead I'm going to talk about what Pittsburgh can do.

What changed my mind was an article I read in, of all places, the *New York Times* food section. The title was "[Pittsburgh's Youth-Driven Food Boom](#)." To most people that might not even sound interesting, let alone something related to startups. But it was electrifying to me to read that title. I don't think I could pick a more promising one if I tried. And when I read the article I got even more excited. It said "people ages 25 to 29 now make up 7.6 percent of all residents, up from 7 percent about a decade ago." Wow, I thought, Pittsburgh could be the next Portland. It could become the cool place all the people in their twenties want to go live.

When I got here a couple days ago, I could feel the difference. I lived here from 1968 to 1984. I didn't realize it at the time, but during that whole period the city was in free fall. On top of the flight to the suburbs that happened everywhere, the steel and nuclear businesses were both dying. Boy are things different now. It's not just that downtown seems a lot more prosperous. There is an energy here that was not here when I was a kid.

When I was a kid, this was a place young people left. Now it's a place that attracts them.

What does that have to do with startups? Startups are made of people, and the average age of the people in a typical startup is right in that 25 to 29 bracket.

I've seen how powerful it is for a city to have those people. Five years ago they shifted the center of gravity of Silicon Valley from the peninsula to San Francisco. Google and Facebook are on the peninsula, but the next generation of big winners are all in SF. The reason the center of gravity shifted was the talent war, for programmers especially. Most

25 to 29 year olds want to live in the city, not down in the boring suburbs. So whether they like it or not, founders know they have to be in the city. I know multiple founders who would have preferred to live down in the Valley proper, but who made themselves move to SF because they knew otherwise they'd lose the talent war.

So being a magnet for people in their twenties is a very promising thing to be. It's hard to imagine a place becoming a startup hub without also being that. When I read that statistic about the increasing percentage of 25 to 29 year olds, I had exactly the same feeling of excitement I get when I see a startup's graphs start to creep upward off the x axis.

Nationally the percentage of 25 to 29 year olds is 6.8%. That means you're .8% ahead. The population is 306,000, so we're talking about a surplus of about 2500 people. That's the population of a small town, and that's just the surplus. So you have a toehold. Now you just have to expand it.

And though "youth-driven food boom" may sound frivolous, it is anything but. Restaurants and cafes are a big part of the personality of a city. Imagine walking down a street in Paris. What are you walking past? Little restaurants and cafes. Imagine driving through some depressing random exurb. What are you driving past? Starbucks and McDonalds and Pizza Hut. As Gertrude Stein said, there is no there there. You could be anywhere.

These independent restaurants and cafes are not just feeding people. They're making there be a there here.

So here is my first concrete recommendation for turning Pittsburgh into the next Silicon Valley: do everything you can to encourage this youth-driven food boom. What could the city do? Treat the people starting these little restaurants and cafes as your users, and go ask them what they want. I can guess at least one thing they might want: a fast permit process. San Francisco has left you a huge amount of room to beat them in that department.

I know restaurants aren't the prime mover though. The prime mover, as the Times article said, is cheap housing. That's a big advantage. But that phrase "cheap housing" is a bit misleading. There are plenty of places that are cheaper. What's special about Pittsburgh is not that it's cheap, but that it's a cheap place you'd actually want to live.

Part of that is the buildings themselves. I realized a long time ago, back when I was a poor twenty-something myself, that the best deals were places that had once been rich, and then became poor. If a place has always been rich, it's nice but too expensive. If a place has always been poor, it's cheap but grim. But if a place was once rich and then got poor, you can find palaces for cheap. And that's what's bringing people here. When Pittsburgh was rich, a hundred years ago, the people who lived here built big solid buildings. Not always in the best taste, but definitely solid. So here is another piece of advice for becoming a startup hub: don't destroy the buildings that are bringing people here. When cities are on the way back up, like Pittsburgh is now, developers race to tear down the old buildings. Don't let that happen. Focus on historic preservation. Big real estate development projects are not what's bringing the twenty-somethings here. They're the opposite of the new restaurants and cafes; they subtract personality from the city.

The empirical evidence suggests you cannot be too strict about historic preservation. The tougher cities are about it, the better they seem to do.

But the appeal of Pittsburgh is not just the buildings themselves. It's the neighborhoods they're in. Like San Francisco and New York, Pittsburgh is fortunate in being a pre-car city. It's not too spread out. Because those 25 to 29 year olds do not like driving. They prefer walking, or bicycling, or taking public transport. If you've been to San Francisco recently you can't help noticing the huge number of bicyclists. And this is not just a fad that the twenty-somethings have adopted. In this respect they have discovered a better way to live. The beards will go, but not the bikes. Cities where you can get around without driving are just better period. So I would suggest you do everything you can to capitalize on this. As with historic preservation, it seems impossible to go too far.

Why not make Pittsburgh the most bicycle and pedestrian friendly city in the country? See if you can go so far that you make San Francisco seem backward by comparison. If you do, it's very unlikely you'll regret it. The city will seem like a paradise to the young people you want to attract. If they do leave to get jobs elsewhere, it will be with regret at leaving behind such a place. And what's the downside? Can you imagine a headline "City ruined by becoming too bicycle-friendly?" It just doesn't happen.

So suppose cool old neighborhoods and cool little restaurants make this the next Portland. Will that be enough? It will put you in a way better position than Portland itself, because Pittsburgh has something Portland lacks: a first-rate research university. CMU plus little cafes means you have more than hipsters drinking lattes. It means you have hipsters drinking lattes while talking about distributed systems. Now you're getting really close to San Francisco.

In fact you're better off than San Francisco in one way, because CMU is downtown, but Stanford and Berkeley are out in the suburbs.

What can CMU do to help Pittsburgh become a startup hub? Be an even better research university. CMU is one of the best universities in the world, but imagine what things would be like if it were the very best, and everyone knew it. There are a lot of ambitious people who must go to the best place, wherever it is. If CMU were it, they would all come here. There would be kids in Kazakhstan dreaming of one day living in Pittsburgh.

Being that kind of talent magnet is the most important contribution universities can make toward making their city a startup hub. In fact it is practically the only contribution they can make.

But wait, shouldn't universities be setting up programs with words like "innovation" and "entrepreneurship" in their names? No, they should not. These kind of things almost always turn out to be disappointments. They're pursuing the wrong targets. The way to get innovation is not to aim for innovation but to aim for something more specific, like better batteries or better 3D printing. And the way to learn about entrepreneurship is to do it, which you [can't in school](#).

I know it may disappoint some administrators to hear that the best thing a university can do to encourage startups is to be a great university. It's like telling people who want to lose weight that the way to do it is to eat less.

But if you want to know where startups come from, look at the empirical evidence. Look at the histories of the most successful startups, and you'll find they grow organically out of a couple of founders building something that starts as an interesting side project. Universities are great at bringing together founders, but beyond that the best thing they can do is get out of the way. For example, by not claiming ownership of "intellectual property" that students and faculty develop, and by having liberal rules about deferred admission and leaves of absence.

In fact, one of the most effective things a university could do to encourage startups is an elaborate form of getting out of the way invented by Harvard. Harvard used to have exams for the fall semester after Christmas. At the beginning of January they had something called "Reading Period" when you were supposed to be studying for exams. And Microsoft and Facebook have something in common that few people realize: they were both started during Reading Period. It's the perfect situation for producing the sort of side projects that turn into startups. The students are all on campus, but they don't have to do anything because they're supposed to be studying for exams.

Harvard may have closed this window, because a few years ago they moved exams before Christmas and shortened reading period from 11 days to 7. But if a university really wanted to help its students start startups, the empirical evidence, weighted by market cap, suggests the best thing they can do is literally nothing.

The culture of Pittsburgh is another of its strengths. It seems like a city has to be socially liberal to be a startup hub, and it's pretty clear why. A city has to tolerate strangeness to be a home for startups, because startups are so strange. And you can't choose to allow just the forms of strangeness that will turn into big startups, because they're all intermingled. You have to tolerate all strangeness.

That immediately rules out [big chunks of the US](#). I'm optimistic it doesn't rule out Pittsburgh. One of the things I remember from growing up here, though I didn't realize at the time that there was anything unusual about it, is how well people got along. I'm still not sure why. Maybe one reason was that everyone felt like an immigrant. When I was a kid in Monroeville, people didn't call themselves American. They called themselves Italian or Serbian or Ukranian. Just imagine what it must have been like here a hundred years ago, when people were pouring in from twenty different countries. Tolerance was the only option.

What I remember about the culture of Pittsburgh is that it was both tolerant and pragmatic. That's how I'd describe the culture of Silicon Valley too. And it's not a coincidence, because Pittsburgh was the Silicon Valley of its time. This was a city where people built new things. And while the things people build have changed, the spirit you need to do that kind of work is the same.

So although an influx of latte-swilling hipsters may be annoying in some ways, I would go out of my way to encourage them. And more generally to tolerate strangeness, even unto the degree wacko Californians do. For Pittsburgh that is a conservative choice: it's a return to the city's roots.

Unfortunately I saved the toughest part for last. There is one more thing you need to be a startup hub, and Pittsburgh hasn't got it: investors. Silicon Valley has a big investor community because it's had 50 years to grow one. New York has a big investor community because it's full of people who like money a lot and are quick to notice new

ways to get it. But Pittsburgh has neither of these. And the cheap housing that draws other people here has no effect on investors.

If an investor community grows up here, it will happen the same way it did in Silicon Valley: slowly and organically. So I would not bet on having a big investor community in the short term. But fortunately there are three trends that make that less necessary than it used to be. One is that startups are increasingly cheap to start, so you just don't need as much outside money as you used to. The second is that thanks to things like Kickstarter, a startup can get to revenue faster. You can put something on Kickstarter from anywhere. The third is programs like Y Combinator. A startup from anywhere in the world can go to YC for 3 months, pick up funding, and then return home if they want.

My advice is to make Pittsburgh a great place for startups, and gradually more of them will stick. Some of those will succeed; some of their founders will become investors; and still more startups will stick.

This is not a fast path to becoming a startup hub. But it is at least a path, which is something few other cities have. And it's not as if you have to make painful sacrifices in the meantime. Think about what I've suggested you should do. Encourage local restaurants, save old buildings, take advantage of density, make CMU the best, promote tolerance. These are the things that make Pittsburgh good to live in now. All I'm saying is that you should do even more of them.

And that's an encouraging thought. If Pittsburgh's path to becoming a startup hub is to be even more itself, then it has a good chance of succeeding. In fact it probably has the best chance of any city its size. It will take some effort, and a lot of time, but if any city can do it, Pittsburgh can.

Thanks to Charlie Cheever and Jessica Livingston for reading drafts of this, and to Meg Cheever for organizing Opt412 and inviting me to speak.

The Risk of Discovery

January 2017

Because biographies of famous scientists tend to edit out their mistakes, we underestimate the degree of risk they were willing to take. And because anything a famous scientist did that wasn't a mistake has probably now become the conventional wisdom, those choices don't seem risky either.

Biographies of Newton, for example, understandably focus more on physics than alchemy or theology. The impression we get is that his unerring judgment led him straight to truths no one else had noticed. How to explain all the time he spent on alchemy and theology? Well, smart people are often kind of crazy.

But maybe there is a simpler explanation. Maybe the smartness and the craziness were not as separate as we think. Physics seems to us a promising thing to work on, and alchemy and theology obvious wastes of time. But that's because we know how things turned out. In Newton's day the three problems seemed roughly equally promising. No one knew yet what the payoff would be for inventing what we now call physics; if they had, more people would have been working on it. And alchemy and theology were still then in the category Marc Andreessen would describe as "huge, if true."

Newton made three bets. One of them worked. But they were all risky.

▪ [Japanese Translation](#)

Charisma / Power

January 2017

People who are powerful but uncharismatic will tend to be disliked. Their power makes them a target for criticism that they don't have the charisma to disarm. That was Hillary Clinton's problem. It also tends to be a problem for any CEO who is more of a builder than a schmoozer. And yet the builder-type CEO is (like Hillary) probably the best person for the job.

I don't think there is any solution to this problem. It's human nature. The best we can do is to recognize that it's happening, and to understand that being a magnet for criticism is sometimes a sign not that someone is the wrong person for a job, but that they're the right one.

THE END

