On Paul Graham

Blas Moros

Intro

The hope is that this "teacher's reference guide" helps highlight the most impactful points from Paul Graham's essays. Paul Graham is best known for starting and selling a company called Viaweb and later starting a new model for funding early stage startups called Y Combinator. His unique background of art, history, philosophy, and writing combined with his experience starting, growing, leading, selling and investing in startups gives him a fascinating and often surprising outlook on broad spectrum of topics. He is one of the few people to have deep fluency in nearly every aspect pertinent to startups and has perhaps more experience and better pattern recognition than anybody else.

Some key filters I consider in order to figure out what topics to focus on and dive into are universality and timelessness. It is worth spending the most time reading, analyzing, understanding and acting upon the things which don't change, or at least change relatively slowly. This effort tends to be well spent as it can help expose and develop "invariant strategies." These strategies are so powerful because they are optimal, timeless and universal. By doing this deep work upfront, you have "pre-paid" in a sense and this can give you the ability and confidence to act and invest in the future when others are retreating. I believe Paul Graham's essays fall into this category as they often address key universal questions, experiences and difficulties and it was an absolute pleasure digging into and digesting them.

One of my favorite essays was *How to Do Philosophy* (pg. 71) and he writes, "The test of utility I propose is whether we cause people who read what we've written to do anything differently afterward." Based on this metric, he achieved spectacularly as he has changed how I approach reading, writing, thinking about startups and more.

Some of the most important and powerful things we can do to learn, grow and improve tend to have three distinctive qualities: nobody can do it for you, nobody can stop you and it costs nothing. I believe these essays are well worth the effort and time and if this summary resonates with you, I highly recommend reading the essays in full and drawing your own conclusions. There can be no replacement for putting in the time to read first-source material (paulgraham.com/articles.html), to mark it up, write it down and absorb it yourself.

Please note that this guide is something I plan to update, iterate, improve and expand upon over time as new essays come out and as I come to deeper understand the already released essays.

1993

Programming Bottom-Up

- If some component grows beyond the stage where it's readily comprehensible, it becomes a mass of complexity which conceals errors as easily as a big city conceals fugitives. Such software will be hard to read, hard to test and hard to debug. In accordance with this principle, a large program must be divided into pieces, and the larger the program, the more it must be divided
- This is just what the new model of programming does assume. Instead of hoping that people won't make mistakes, it tries to make the cost of mistakes very low. The cost of a mistake is the time required to correct it. Programming style can then depend less on planning and more on exploration. Planning is a necessary evil. It is a response to risk: the more dangerous an undertaking, the more important it is to plan ahead. The most useful source of information available to utilize is the experience of implementing it and more powerful tools allow for risk and planning to be reduced.
    - *The danger of planning is becoming rigid and "sticking to the plan" even if the situation has changed. The organic model – going to where the opportunities are – allows you to be dynamic and opportunistic depending on the context and goals. Although a tree doesn't know in which direction it will grow, its growth path will be optimal given the environment and circumstances as it continuously surveys, adapts and takes advantage of any space, nutrients, etc. that it needs to grow.*

Chapter 1 of ANSI Common Lisp, by Paul Graham

- Even the most ambitious people shrink from big undertakings. It's easier to start something if one can convince oneself (however speciously) that it won't be too much work. That's why so many big things have begun as small things. Rapid prototyping lets us start small
    - *What does the easiest, simplest version of what you want to accomplish look like? Start with a baby step in that direction and you may just eventually end up with something grander and more impressive than you ever could have dreamed of. Many of the world's great entrepreneurs had no idea how large their company would become. Bill Gates thought the couple, small HQ buildings in Seattle would be capable of handling their growth forever…*

2001

Java's Cover

- When you choose technology, you have to ignore what other people are doing and consider only what will work the best. Big companies can elect to do what all other big companies are doing and generally be alright. Startups, however, can't follow this same path. If they follow what every other startup is doing, they will go out of business as the survival rate is way less than 50%. So, if you're running a startup, you had better be doing something odd. If not, you're in trouble
    - *When you do as everyone else does, do not be surprised when you get what everyone else gets*

Beating the Averages

- In business, there is nothing more valuable than a technical advantage your competitors don't understand. In business, as in war, surprise is worth as much as force.
    - *Graham chose to use Lisp in his startup, Viaweb, because he thought that with Lisp he could get features done faster than competitors, could do things in their software that the competitors couldn't do, it would cost less so they could provide a better product for less money and still make a profit*
    - *The power of iteration, velocity, nimbleness, agility. Even if competitors introduced features Viaweb didn't currently have, they could quickly implement them*
- A startup should give its competitors as little information as possible. If they didn't know what language our software was written in, or didn't care, I wanted to keep it that way.
- Programming languages are not merely technologies, but habits of mind as well, and nothing changes slower. The only programmers in a position to see all the differences in power between the various languages are those who understand the most powerful one. You can't trust the opinions of others, because of the "Blub paradox" – they're satisfied with whatever language they happen to use, because it dictates the way they think about programs. Can use technology that competitors don't understand like aikido – using their unwillingness to change and adapt against them
    - *Programming languages are different, they are not just technology. They are half technology and half religion*
    - *This applies to all domains as well and shows the power of multi-disciplinary thinking. If you only have one mental tool, you will be like the man with a hammer. But, if you can draw from multiple fields and have* true *deep fluency in them, you will be able to pick and choose the most powerful and appropriate tools for the situation at hand*
- A suspicious programmer might begin to wonder if there was some correlation here. A big chunk of our code was doing things that are very hard to do in other languages besides Lisp. The resulting software did things our competitors' software couldn't do. Maybe there was some kind of connection. I encourage you to follow that thread. There may be more to that old man hobbling along on his crutches than meets the eye. Lisp's power is multiplied by the fact that your competitors don't get it
    - *Where there is mystery there is margin*

- A good tool to evaluate what competitors are doing is to view their job listings. The rest can be made pretty but they have to be real and authentic about their job listings or else they'll get the wrong candidates.
    - *Graham used this technique over many years to see which competitors he had to worry about and which not to*
    - *Graham's partner, Robert Morris, said later that he didn't have to be so secretive about Lisp because even if the competitors had known they were using Lisp, they wouldn't have understood why. "If they were that smart, they'd already be programming in Lisp."*
- The old adage, "you can't tell a book by its cover," originated in the times when books were sold in plain cardboard covers, to be bound by each purchaser according to his taste. In those days, you couldn't tell a book by its cover. But publishing has advanced since then – present day publishers work hard to make the cover something you can tell a book by.
    - *Graham has spent enough time around technology that he can look at the "cover" of technology and know which to avoid. His deep fluency in this space gives him superior pattern recognition*
- Reasons to dislike Java - really good products don't need to be promoted or over hyped; languages designed for other people to use have been bad and the good languages have been those that were designed for their own creators; ulterior motives are bad; no one loves it; people are forced to use it; it has too many cooks; it's bureaucratic; it's pseudo-hip; it's designed for large organizations; the wrong people like it; Sun's business model is being undermined; the DoD likes it

Being Popular

- Good programming languages have to feature very powerful abstractions
- Programming languages become popular if hackers use them because this tiny minority designs all the good software and their influence is such that the rest of the programmers will tend to use whatever language they use
    - *The power of knowing the right customers and designing something they love. Better to have a small yet passionate user base than a massive one which is only temperate about your product*
    - *Thinks that languages have to be popular to be good and that getting the first twenty users may be harder than going from twenty to a thousand. Best way to convince new users is through a Trojan horse – give people an application they want which happens to be written in a new language*
- A new language must have free implementation, a book, something to hack, must be brief, must be able to do what they want (can never guess all the ways a language will be used), dirty and clean – cleanly designed but let's hackers have their way with it, ability to write quick/throwaway programs but these tend to stick around and be the foundation for great programs (big things tend to be too scary to start but starting small and quick is easy and benefits from evolution), interactive, available, start up quickly, large libraries for manipulating strings, hackers will wait until they're more sure that the language will be around for some time and simple repetition often solves the problem (it's not when people

notice you're there that they pay attention; it's when they notice you're still there), early adopters are demanding and help flush out problems quickly, the most important design is the ability to redesign, no language designed by committee as they yield bad design and interfere with redesign,

- A friend of mine rarely does anything the first time someone asks him. He knows that people sometimes ask for things that they turn out not to want. To avoid wasting his time, he waits till the third or fourth time he's asked to do something; by then, whoever's asking him may be fairly annoyed, but at least they probably really do want whatever they're asking for.
- There are typically two ways new technology gets introduced – the organic growth method (two guys in a garage) and the big-bang (VC-backed). Most of the garage boys are envious of the big-bang buys but more often than not it is the organic growth technology which yields better technology and richer founders
- To write good software you must simultaneously keep two opposing ideas in your head. You need the young hacker's naïve faith in his abilities, and at the same time the veteran's skepticism. You have to be able to think how hard can it be? With one half of your brain while thinking it will never work with the other. The trick is to realize there's no real contradiction here. You want to be optimistic and skeptical about two different things. You have to be optimistic about the possibility of solving the problem, but skeptical about the value of whatever solution you have so far. People who do good work often think that whatever they're working on is no good. Others see what they've done and are full of wonder, but the creator is full of worry. This pattern is no coincidence: it is the worry that made the work good. If you can keep hope and worry balanced, they will drive a project forward the same way your two legs drive a bicycle forward.
    - *A good definition of intelligence is the ability to hold two opposing thoughts in your head at the same time while still being able to function*
    - *This constant worry and self-doubt seems ubiquitous in people trying to do great things. Love the connection that it is in fact this worrying which makes the work great, not in spite of it*
    - *Life is full of false dualities. People only present choice A and choice B to us when in reality there are also choices C-Z…*
- Users are a double-edged sword. They can help you improve your language, but they can also deter you from improving it. So choose your users carefully, and be slow to grow their number. Having users is like optimization: the wise course is to delay it. Also, as a general rule, you can at any given time get away with changing more than you think. Introducing a change is like pulling off a bandage: the pain is a memory almost as soon as you feel it.

Five Questions About Language Design

- Designing software and algorithms is all about dealing with human weaknesses and most of us hate to acknowledge this
- You shouldn't design languages for bad programmers though. In fact, I think you ought to design for the best programmers, but even the best programmers have limitations
- Syntax and semantics may not be totally separate. What you want in your language may be related to how you express it

The Roots of Lisp

- McCarthy's key idea was to use a simple data structure called a list for both code and data
- The unusual thing about Lisp – in fact, the defining quality of Lisp – is that it can be written in itself.

The Other Road Ahead

- Most people, most of the time, will take whatever choice requires least work – death before inconvenience
    - *When you understand this, you can take advantage of it – whether you're a programmer or the operator of a business. Often competitors will not be willing to put in the work required and although it is by definition difficult, it will be successful.*
    - *Successful design sees through the customer's eyes and makes whatever choice they are being asked to make as simple as possible. Little nudges go a long way…*
- When you have competitors, "you can" means "you must" because if you don't take advantage of this possibility, your competitors will
    - *Turned negatives into positives and win/wins – ability to release fixes continuously lead to developers being genuinely interested in customer support because they would often learn about bugs and customers would feel important and even triumphant in finding these bugs*
    - *Focusing on "bottom half of the roster" – made customer support and developer relationships win/win as customer supports' honor was on the line when bringing in new bugs and they always knew which users features wanted more. This kind of relationship is so much more fruitful than a contentious, hierarchical one. Me and you against the problem rather than me against you because of the problem*
- Working to implement one idea gives you more ideas. Shelving ideas probably even inhibits new ideas. At Viaweb, often didn't have future plans because plans are just another word for ideas on the shelf which they didn't allow to happen. When they thought of good ideas, they implemented them.
- Paying attention is more important to reliability than moving slowly. Because he pays close attention, a Navy pilot can land a 40,000 lb. aircraft at 140 miles per hour on a pitching carrier deck, at night, more safely than the average teenager can cut a bagel.
- Complexity and bureaucracy scales exponentially with the size of the group. "We never had more to say at any one time [with 13 programmers] than we could say as we were walking to lunch."
- Software should do what users think it will. But you can't have any idea what users will be thinking, believe me, until you watch them. By watching users, you can often tell when they're in trouble and since the customer is always right, that's a sign of something you need to fix.
    - *Success comes from seeing through your counterparties' eyes*
- Software is ideally suited for price discrimination as the marginal cost is close to zero

- You'll sell more of something when it's easy to buy. Make it difficult to buy and give people a chance to second guess themselves and they will buy much less.
- There is always a tendency for rich customers to buy expensive solutions, even when cheap solutions are better, because the people offering expensive solutions can spend more to sell them. There is nothing you can do about this conundrum, so the best plan is to go for the smaller customers first. The rest will come in time
- If a company wants to make a platform that startups will build on, they have to make it something that hackers themselves will want to use. That means it has to be inexpensive and well-designed
- If not willing to disrupt or cannibalize yourself – "A cash cow can be a damned heavy monkey on your back"
- "Just good enough" is often a powerful stepping stone to outweigh any cons or awkwardness in usage
- Most hackers don't start their startup because they think they don't know anything about business and are afraid of competition. Neither of these fences have any "current" in them. There are only two things you have to know about business: build something users love and make more than you spend. If you can get these two right, you'll be ahead of most startups. You can figure out the rest as you go…Start by making something clean and simple that you would want to use yourself. The customer is always right but about different things; the least sophisticated users show you what you need to clarify and simplify and the most sophisticated tell you what features you need to add.
    - *When building a highway, you must build it taking the lowest common denominator driver into consideration – making the signage, rules, etc. as straightforward and understandable as possible. This "design for dummies" approach should be considered everywhere…*
- The standard to compare your software to is what it could be, not what your current competitors happen to have.
- Companies often wonder what to outsource and what not to. One possible answer: outsource any job that's not directly exposed to competitive pressure, because outsourcing it will thereby expose it to competitive pressure
- There is always room for something new if the current options suck enough. Make sure it works on all the free OSes first – new things start with their users.

Taste for Makers

- For those of us who design things, these are not just theoretical questions, if there is such a thing as beauty, we need to be able to recognize it. We need good taste to make good things
- Once you start to examine the question, it's surprising how much different fields' ideas of beauty have in common. The same principles of good design crop up again and again. Good design is simple. Everything above simplicity is evasion. When you can't deliver ornament, you have to deliver substance. Good design is timeless. Good design solves the right problem. Good design is suggestive. Good design is often slightly funny. I think it is because humor is related to strength. To have a sense of humor is to be strong: to keep one's sense of humor is to shrug off misfortunes, and to lose one's sense of humor is to be

wounded by them. Good design is hard. If function is hard enough, form is forced to follow it, because there is no effort to spare for error. Wild animals are beautiful because they have hard lives. Good design looks easy. Like great athletes, great designers make it look easy. Mostly this is an illusion. In science and engineering, some of the greatest discoveries seem so simple that you say to yourself, I could have thought of that! In most fields the appearance of ease seems to come with practice. Perhaps what practice does is train your unconscious mind to handle tasks that used to require conscious thought. In some cases, you literally train your body. An expert pianist can play notes faster than the brain can send signals to his hand. Likewise, an artist, after a while, can make visual perception flow in through his eye and out through his hand as automatically as someone tapping his foot to a beat. When people talk about being in "the zone," I think what they mean is that the spinal cord has the situation under control. Your spinal cord is less hesitant, and it frees conscious thought for the hard problems. Good design uses symmetry (repetition and recursion). Nature uses them a lot, which is a good sign. The danger of symmetry, and repetition especially, is that it can be used as a substitute for thought. Good design resembles nature. It's not so much that resembling nature is intrinsically good as that nature has had a long time to work on the problem. It's not cheating to copy. Good design is redesign. It's rare to get things right the first time. Experts expect to throw away some early work. Mistakes are natural. Instead of treating them as disasters, make them easy to acknowledge and easy to fix. Good design can copy. It is more important to be right than original. Unknowing imitation is almost a recipe for bad design. The ambitious are not content to imitate. The second phase in the growth of taste is a conscious attempt at originality but are selfless and confident enough to take from anyone without feeling that their own vision will be lost in the process. Good design is often strange. The only style worth having is the one you can't help. And this is especially true for strangeness. Good design happens in chunks. Nothing is more powerful than a community of talented people working on related problems. Genes count for little by comparison: being a genetic Leonardo was not enough to compensate for having been born near Milan instead of Florence. Good design is often daring, teetering on the border of ostracism. This problem afflicts not just every era, but in some degree every field. Today's experimental error is tomorrow's new theory. If you want to discover great new things, then instead of turning a blind eye to the places where conventional wisdom and truth don't quite meet, you should pay particular attention to them.

  - *Wow*

- Problems can be improved as well as solutions. In software, an intractable problem can usually be replaced by an equivalent one that's easy to solve. Physics progressed faster as the problem became predicting observable behavior, instead of reconciling it with scripture
- As a practical matter, I think it's easier to see ugliness than to imagine beauty. Most of the people who've made beautiful things seem to have done it by fixing something that they thought ugly. Intolerance for ugliness is not in itself enough. You have to understand a field well before you develop a good nose for what needs fixing. You have to do your homework. But as you become an expert in a field, you'll start to hear little voices saying, what a hack! There must be a better way. Don't ignore those voices. Cultivate them. The recipe for great work is: very exacting taste, plus the ability to gratify it.

2002

Succinctness is Power

- The main purpose of a language is to be good to think in (rather than just to tell a computer what to do once you've thought of it). The true test of a language is how well you can discover and solve new problems, not how well you can use it to solve a problem someone else has already formulated
- I think the best way to find (or design) the best language is to become hypersensitive to how well a language lets you think, then choose/design the language that feels best
- Restrictiveness is mostly lack of succinctness

Revenge of the Nerds

- The pointy-haired boss miraculously combines two qualities that are common by themselves, but rarely seen together: (a) he knows nothing whatsoever about technology, and (b) he has very strong opinions about it. He sticks to "industry best practice" to avoid responsibility and blame in case the company loses. However, it never gets you the best, merely the average
  - *Must have credibility – can't be in an ivory tower preaching downwards when those "below" you know that you have no true domain expertise*
  - *A great way to develop credibility in your opinions is to know the other side's arguments better than they do. Darwin took 20 years before publishing his* On the Origin of Species *because he was thinking through every counterpoint to his theory and answering them better than the opposition ever could*
- The main reason many ideas and practices are widespread is because they are comfortable
- Lisp allows one to express the language in its own data structures which turns out to be a very powerful feature
  - *Much like DNA which holds its own information and its own instructions*
- One tends to magnify risks one doesn't truly understand
- Often, what differentiates you or your company and gives you a competitive advantage seems like an anomaly to outsiders but is in fact the cause and effect of your moat

2003

Design and Research

- The difference between design and research seems to be a question of new versus good. Design doesn't have to be new, but it has to be good. Research doesn't have to be good, but it has to be new
- You have to design for the user, but you have to design what the user needs, not simply what he says he wants
- Must choose your group of users and almost always this group must include the designer himself
    - *This is why most great startups are started by founders who are solving a problem they personally face*
- There are two broad strategies: Worse is Better and the Hail Mary strategy. The Worse is Better gets a prototype in front of users as quickly as possible and slowly refines along the way. A prototype doesn't have to be just a mode; you can refine it into the finished product. I think you should always do this when you can. It lets you take advantage of new insights you have along the way. But perhaps even more important, it's good for morale and morale is key in design because it keeps you engaged. The Hail Mary tries to create the complete, finished, product in one long touchdown pass. As far as I know, this is a recipe for disaster
    - *Taking lessons from and mimicking nature is often a route to success. The Worse is Better course of action allows for iteration and evolution of your product which is of course observed in nature*
- Notice all this time I've been talking about "the designer." Design usually has to be under the control of a single person to be any good. And yet it seems to be possible for several people to collaborate on a research project. This seems to me one of the most interesting differences between research and design. Design by committee is a synonym for bad design. Good design requires a dictator. One reason is that good design has to be all of a piece. Design is not just for humans, but for individual humans. If a design represents an idea that fits in one person's head, then the idea will fit in the user's head too.
- There's nothing more valuable than the advice of someone whose judgment you trust
    - *Few delights can equal the mere presence of one whom we trust utterly. – George MacDonald*

Better Bayesian Filtering

- Don't ignore data
    - *Large data sets are a statistician's best friend (with the caveat that the data is relevant, timely, reliable…)*

Why Nerds Are Unpopular

- So, if intelligence itself is not a factor in popularity, why are smart kids so consistently unpopular? The answer, I think, is that they don't really want to be popular. Not enough, anyway. There was something else they wanted more: to be smart, to make great things. The main reasons nerds are unpopular is that they have other things to think about.
- Another reason kids persecute nerds is to make themselves feel better. When you tread water, you lift yourself up by pushing water down. Likewise, in any social hierarchy, people unsure of their own position will try to emphasize it by maltreating those they think rank below. I've read that is why poor whites in the United States are the group most hostile to blacks
    - *A fantastic definition of mental health is the extent to which you help others*
    - *A great lesson I learned from my mom has to do with "filling your cup." Those with self-love, compassion, confidence, mental health, etc. have a "full cup." They don't need to extract these things from others as they have enough. Instead, they can spend their time thinking of others and filling their cups by giving praise, showing empathy, etc. These people with full cups tend to be the most admired people as they are able to give, in copious amounts, these much sought after qualities in greater amounts than anybody else. The irony is that these people also tend to receive as much, if not more, than they give away. "There is a wonderful, almost mystical law of nature that says three of the things we want most – happiness, freedom and peace of mind – are always attained when we give them to others." – John Wooden. There is much wisdom in paradoxes*
- I think the important thing about the real world is not that it's populated by adults, but that it's very large, and the things you do have real effects. That's what school, prison, and ladies-who-lunch all lack. The inhabitants of all those worlds are trapped in little bubbles where nothing they do can have more than a local effect. Naturally these societies degenerate into savagery. They have no function for their form to follow. When the things you do have real effects, it's no longer enough just to be pleasing. It starts to be important to have the right answers, and that's where nerds show the advantage. The other thing that's different about the real world is that it's much larger. In a large enough pool, even the smallest minorities can achieve a critical mass if they clump together
- What bothers me is not that the kids are kept in "prisons" but that (a) they aren't told about it, and (b) the prisons are mostly run by the inmates. Life in this twisted world is stressful for the kids. And not just for the nerds. Like any war, it's damaging even to the winners.
- As far as I can tell, the concept of the hormone-crazed teenager is coeval with suburbia. I don't think this is a coincidence. I think teenagers are driven crazy by the life they're made to lead. Teenage apprentices in the Renaissance were working dogs. Teenagers now are neurotic lapdogs. Their craziness is the craziness of the idle everywhere.
- Teenagers seem to have respected adults more in the past because the adults were the visible experts in the skills they were apprenticing in. Now most kids have little idea what their parents do in their distant offices, and see no connection (indeed, there is precious little) between schoolwork and the work they'll do as adults. What happened? We're up against a hard one here. The cause of this problem is the same as the cause of so many present ills: specialization. As jobs become more specialized, we have to train longer for

them and teenagers are now useless, except as cheap labor in industries like fast food, which evolved explicitly to exploit this fact

- Misrule breeds rebellion; this is not a new idea. And yet the authorities still for the most part act as if drugs were themselves the cause of the problem
- It's important for nerds to realize, too, that school is not life. School is a strange, artificial thing, half sterile and half feral. It's all-encompassing, like life, but it isn't the real thing. It's only temporary, and if you look, you can see beyond it even while you're still in it
    - *Galilean Relativity – it is impossible to truly grasp or understand a system of which you are part of. But, if you can find ways to mitigate this and "step above time," above your system, you will get a view from the "crow's nest" and gain perspective on the proper course of action. Getting advice from those not in the system (whether they are older or in a totally different field) is one great tactic for rising above. Learning from the successes and failures of others is another*

The Hundred-Year Language

- Staying close to the main evolutionary tree branches which will carry on is a useful heuristic for finding languages that will be good to program in now and to make better decisions about language design
    - *Spend the most time on ideas, decisions, languages, etc. that will have the longest shelf life*
- There's good waste, and bad waste. I'm interested in good waste – the kind where, by spending more, we can get simpler designs. How will we take advantage of the opportunities to waste cycles that we'll get from new, faster hardware? The desire for speed is so deeply engrained in us, with our puny computers, that it will take conscious effort to overcome it. In language design, we should be consciously seeking out situations where we can trade efficiency for even the smallest increase in convenience…Inefficient software isn't gross. What's gross is a language that makes programmers do needless work. Wasting programmer time is the true inefficiency, not wasting machine time. This will become ever more clear as computers get faster.
- When you're working on language design, I think it is good to have such a target and to keep it consciously in mind. When you learn to drive, one of the principles they teach you is to align the car not by lining up the hood with the stripes painted on the road, but by aiming at some point in the distance. Even if all you care about is what happens in the next ten feet, this is the right answer. I think we can and should do the same with programming languages.
    - *The importance of having a long time horizon and a "true north" regarding your morals, values, behavior, goals, "life vision," and more. If you have the big picture, end goal in mind, then making decisions, saying no, taking advantage of opportunities, etc. becomes that much easier. You are able to "stand above time" and make optimal decisions when you have this end of the maze vision and mindset.*

If Lisp is so Great

- In languages, as in so many things, there’s not much correlation between popularity and quality
- I was so ignorant that learning almost anything meant learning new things
- Popularity is always self-perpetuating, but it’s especially so in programming languages

Hackers and Painters

- Hacking and painting have a lot in common. In fact, of all the different types of people I’ve known, hackers and painters are among the most alike. What hackers and painters have in common is that they’re both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is make good things. They’re not doing research per se, though if in the course of trying to make good things they discover some new technique, so much the better.
- For hackers, computers are just a medium of expression, as concrete is for architects or paint for painters.
- There is a small but real chasm between architects and engineers: architects decide what to do, and engineers figure out how to do it. What and how should not be kept too separate. You’re asking for trouble if you try to decide what to do without understanding how to do it.
- The way to create something beautiful is often to make subtle tweaks to something that already exists, or to combine existing ideas in a slightly new way. This kind of work is hard to convey in a research paper
- There is nothing so tempting as an easy test that kind of works
- The only external test is time. Over time, beautiful things tend to thrive, and ugly things tend to get discarded. Unfortunately, the amounts of time involved can be longer than human lifetimes
- I’ve found that the best sources of ideas are not the other fields that have the word “computer” in their names, but the other fields inhabited by makers. Painting has been a much richer source of ideas than the theory of computation
- Programs should be malleable, allowing you to sketch roughly at first and debug over time
- If you want to make money at some point, remember this, because this is one of the reasons startups win. Big companies want to decrease the standard deviation of design outcomes because they want to avoid disasters. But when you damp oscillations, you lose the high points as well as the low. This is not a problem for big companies, because they don’t win by making great products. Big companies win by sucking less than other big companies. So if you can figure out a way to get in a design war with a company big enough that its software is designed by product managers, they’ll never be able to keep up with you. These opportunities are not easy to find, though. It’s hard to engage a big company in a design war, just as it’s hard to engage an opponent inside a castle in hand to hand combat. It would be pretty easy to write a better word processor than Microsoft Word, for example, but Microsoft, within the castle of their operating system monopoly, probably wouldn’t even notice if you did. The place to fight design wars is in new markets, where no one has yet managed to establish any fortifications. That’s where you can win big by taking the bold

approach to design, and having the same people both design and implement the product. Microsoft themselves did this at the start. So did Apple. And Hewlett-Packard. I suspect almost every successful startup has.

- One thing we can learn, or at least confirm, from the example of painting is how to learn to hack. You learn to paint mostly by doing it. Ditto for hacking. Most hackers don't learn to hack by taking college courses in programming. They learn to hack by writing programs of their own at age thirteen. Even in college classes, you learn to hack mostly by hacking. The other way to learn is from examples. For a painter, copying forces you to look closely at the way a painting is made. Writers do this too. Hackers, likewise, can learn to program by looking at good programs – not just at what they do, but the source code too. Another example we can take from painting is the way that paintings are created by gradual refinement. Paintings usually begin with a sketch. Gradually the details get filled in. But it is not merely a process of filling in. Sometimes the original plans turn out to be mistaken. Everyone by now presumably knows about the danger of premature optimization. I think we should be just as worried about premature design – deciding too early what a program should do. Hacking can also learn from painting not only how to manage our own work but how to work together. Painting has always utilized a master/apprentice model and worked on different parts of the painting. The right way to collaborate, I think, is to divide projects into sharply defined modules, each with a definite owner, and with interfaces between them that are as carefully designed and, if possible, as articulated as programming languages.
- Hackers start with original work and get good whereas scientists start good and get original
- This sounds like a paradox, but a great painting has to be better than it has to be. How hard one works on a painting doesn't depend at all on how closely one expects anyone to look at it. Be relentless. Relentlessness wins because, in the aggregate, unseen details become visible. All these unseen details combine to produce something that's just stunning, like a thousand barely audible voices all singing in tune
- In hacking, like painting, work comes in cycles. Sometimes you get excited about some new project and you want to work sixteen hours a day on it. Other times nothing seems interesting. To do good work, you have to take these cycles into account, because they're affected by how you react to them. Sometimes, backing off can prevent ambition from stalling. It's a good idea to save some easy tasks for moments when you would otherwise stall
    - *Think this is a smart and unique way of implementing the Pomodoro technique. Doing deep work for 25 minutes, or until you feel yourself losing focus, and then relax and celebrate for a couple minutes and take care of some easy tasks until your tank is again full for deep work*
- Like painting, most software is intended for human audience. And so hackers, like painters, must have empathy to do really great work. You have to be able to see things from the user's point of view. It turns out that looking at things from other people's point of view is practically the secret of success. It doesn't necessarily mean being self-sacrificing. Far from it. Understanding how someone else sees things doesn't imply that you'll act in his interest; in some situations – in war, for example – you want to do exactly the opposite. Most makers make things for a human audience. And to engage an audience you have to understand what they need. Empathy is probably the single most important difference

between a good hacker and a great one. One way to tell how good people are at empathy is to watch them explain a technical question to someone without a technical background

  - *"It turns out that looking at things from other people's point of view is practically the secret of success." Theoretically, knowing exactly how the world looks through all your counterparties' eyes will allow you to make zero mistakes and empathy and deep fluency are central to gaining these different points of view*

- In most fields the great work is done early on. Over and over we see the same pattern. A new medium appears, and people are so excited about it that they explore most of its possibilities in the first couple generations
- An example of applied empathy. At Viaweb, if we couldn't decide between two alternatives, we'd ask, what would our competitors hate most? At one point a competitor added a feature to their software that was basically useless, but since it was one of the few they had that we didn't, they made much of it in the trade press. We could have tried to explain that the feature was useless, but we decided it would annoy our competitor more if we just implemented it ourselves, so we hacked together our own version that afternoon

2004

What You Can't Say

- It's the nature of fashion to be invisible, in the same way that the movement of the earth is invisible to all of us riding on it. What scares me is that there are moral fashions too.
    - *Galilean Relativity at play again*
- The conformist test – do you have any opinions that you would be reluctant to express in front of a group of your peers? Almost certainly, there is something wrong with you if you don't think things you don't dare say out loud
- What can't we say? One way to find these ideas is simply to look at things people do say and get in trouble for. We are of course looking for things we can't say which are true, or at least have enough chance of being true that the question should remain open…I suspect the statements that make people maddest are those they worry might be true
- Another test is to see what is "heresy" or at least today's equivalent such as "divisive" or "racially insensitive." Can also see what today's labels are, such as "sexist"
- Can also look to the past to see what used to be acceptable and is now unthinkable. Might not even have to look to the past but merely to different cultures to see what they think is acceptable that you don't. My hypothesis is that the side that's shocked is most likely to be the mistaken one
- There are certain taboos which are universal, such as murder, but any idea that's considered harmless in a significant percentage of times and places, and yet taboo in ours, is a good candidate for something we're mistaken about
- To launch a taboo, a group has to be poised halfway between weakness and power and be nervous. A confident group doesn't need taboos to protect it. It's not considered improper to make disparaging remarks about Americans, or the English. And yet a group has to be powerful enough to enforce a taboo.
- There is nothing so wrong as the principles of the most recently defeated opponent
- Great work tends to grow out of ideas that others have overlooked, and no idea is so overlooked as one that's unthinkable. Training yourself to think unthinkable thoughts has advantages beyond the thoughts themselves. It's like stretching. When you stretch before running, you put your body into positions much more extreme than any it will assume during the run. If you think things so outside the box that they'd make people's hair stand on end, you'll have no trouble with the small trips outside the box that people call innovative.
- Argue with idiots, and you become an idiot
- When you find what you can't say, don't say it. Draw a sharp line between your thoughts and your speech. Inside your head, anything is allowed but keep it to yourself. The problem with keeping your thoughts secret, though, is that you lose the advantages of discussion. Talking about an idea leads to more ideas. So the optimal plan, if you can manage it, is to have a few trusted friends you can speak openly to. This is not just a way to develop ideas; it's also a good rule of thumb for choosing friends. The people you can say heretical things to without getting jumped on are also the most interesting to know.
- To see fashion in your own time requires a conscious effort. Without time to give you distance, you have to create distance yourself. Instead of being part of the mob, stand as

far away from it as you can and watch what it's doing. And pay especially close attention whenever an idea is being suppressed. And, it's not just the mob you need to learn to watch from a distance. You need to be able to watch your own thoughts from a distance. How can you see the wave, when you're the water? Always be questioning

  - *An antidote to Galilean Relativity is to always be questioning, being objective, taking nothing for granted, making no assumptions, challenging your own beliefs…*

The Word "Hacker"

- Hackers worry about government spying because it just leads eventually to a world in which bad ideas win

How to Make Wealth

- If you wanted to get rich, how would you do it? I think your best bet would be to start or join a startup. Economically, you can think of a startup as a way to compress your whole working life into a few years. The advantage of creating wealth, rather than getting rich, is not just that it's more legitimate but that it's more straightforward. You just have to do something people want
- If you want to create wealth, it will help to understand what it is. Wealth is not the same thing as money. Wealth is as old as human history. Far older, in fact; ants have wealth. Money is a comparatively recent invention. Wealth is the fundamental thing. Wealth is stuff we want: food, clothes, houses, cars, gadgets, travel to interesting places, and so on. You can have wealth without money. Wealth is what you want, not money. Money is simply a side effect of specialization. In a specialized society, most of the things you need, you can't make yourself and you need money to pay others to make it for you.
- Pie Fallacy – there is not a fixed amount of wealth in this world. Again, money is not wealth which may be fixed in a given period of time
- The top 5% of programmers probably write 99% of the good software
- A company that could pay all its employees so straightforwardly would be enormously successful. Many employees would work harder if they could get paid for it. More importantly, such a company would attract people who wanted to work especially hard. It would crush its competitors
- To get rich you need to get yourself in a situation with two things, measurement and leverage. You need to be in a position where your performance can be measured, or there is no way to get paid more by doing more. And you have to have leverage, in the sense that the decisions you make have a big effect. A good hint to the presence of leverage is the possibility of failure. However, you don't need to be a CEO or an athlete to have measurement and leverage. All you need to do is be part of a small group working on a hard problem.
- A startup is not merely ten people, but ten people like you. Steve Jobs once said that the success or failure of a startup depends on the first ten employees. I agree. Being small is not, in itself what makes startups kick butt, but rather that small groups can be select. Startups have leverage because they make money by inventing new technology. What is

technology? It's technique. It's the way we all do things. And when you discover a new way to do things, its value is multiplied by all the people who use it. Even giant corporations like McDonald's or Wal-Mart can be said to create technology. A McDonald's franchise is controlled by rules so precise that it is practically a piece of software. Write once, run everywhere. Ditto for Wal-Mart. Sam Walton got rich not by being a retailer, but by designing a new kind of store

- Use difficulty as a guide not just in selecting the overall aim of your company, but also at decision points along the way. At Viaweb, one of our rules of thumb was "run upstairs." Suppose you are a little, nimble guy being chased by a big, fat, bully. You open a door and find yourself a staircase. Do you go up or down? I say up. The bully can probably run downstairs as fast as you can. Going upstairs his bulk will be more of a disadvantage. Running upstairs is hard for you but even harder for him. What this meant in practice was that we deliberately sought hard problems, barriers to entry. Start by picking hard problems and then at every decision point, take the harder choice
- I think this is a good plan for life in general. If you have two choices, choose the harder. If you're trying to decide whether to go out running or sit home and watch TV, go running. Probably the reason this trick works so well is that when you have two choices and one is harder, the only reason you're even considering the other is laziness. You know in the back of your mind what the right thing to do is, and this trick merely forces you to acknowledge it.
- I think it's a good idea to get bought, if you can. Running a business is different from growing one. It is just as well to let a big company take over once you reach cruising altitude. It's also financially wiser, because selling allows you to diversify. What would you think of a financial advisor who put all his client's assets into one volatile stock?
- Getting bought is an art. For potential acquirers, the most powerful motivator is the prospect that one of their competitors will buy you. This, as we found, causes CEOs to take red-eyes. The second biggest worry is that, if they don't buy you now, you'll continue to grow rapidly and will cost more to acquire later, or even become a competitor. In both cases, what it all comes down to is the number of users you have
- One of the prime catalysts for industrialization was the spread of the rule of law. A necessary, if not sufficient, condition was that people who made fortunes be able to enjoy them in peace
- Don't let a ruling class of warriors and politicians squash the entrepreneurs. The same recipe that makes individuals rich makes countries powerful. Let the nerds keep their lunch money and you rule the world

Mind the Gap

- When people care enough about something to do it well, those who do it best tend to be far better than everyone else. There's a huge gap between Leonardo and second-rate contemporaries like Borgognone. Like chess or painting or writing novels, making money is a very specialized skill. But for some reason we treat this skill differently. No one complains when a few people surpass all the rest at playing chess or writing novels, but when a few people make more money than the rest, we get editorials saying this is wrong

- I think there are three reasons we treat making money as different: the misleading model of wealth we learn as children (confuse it with money, think there is a fixed amount, something that is distributed by authorities or parents and should be distributed equally, rather than something that has to be created and perhaps unequally); the disreputable way in which, till recently, most fortunes were accumulated (stolen); and the worry that great variations in income are somehow bad for society (may increase gap in income but decrease other gaps between rich and poor such as quality goods, quality of life but not brand). As far as I can tell, the first is mistaken, the second is outdated, and the third empirically false. Could it be that, in a modern democracy, variation in income is actually a sign of health? If technology doesn't bring about greater inequality it could be for three reasons: technical innovation has stopped, that the people who would create the most wealth aren't doing it or that they aren't getting paid for it
- To say a certain kind of work is underpaid is identical with saying that people want the wrong things
- You need rich people in your society not so much because in spending their money they create jobs, but because of what they have to do to get rich. I'm not talking about the trickle-down effect here. I'm not saying that if you let Henry Ford get rich, he'll hire you as a waiter at his next party. I'm saying that he'll make you a tractor to replace your horse
- Part of the reason this subject is so contentious is that some of the most vocal on the subject of wealth – university students, heirs, professors, politicians and journalists – have the least experience creating it
- One of the biggest differences between the Daddy Model [where effort = reward] and reality is that in reality effort does not necessarily correlate with how much wealth it brings. Painting a house with a toothbrush should not bring you more money just because it is harder

Great Hackers

- In many fields, the hard problem is not solving problems, but deciding what problems to solve
- Great hackers absolutely love to program. They think of it as something they do for fun, and which they're delighted to find people will pay them for. Money is not the main driver. Like all craftsmen, hackers like good tools and find it unbearable to use bad tools. Good hackers love open software because they insist on control. When something's broken, they need to fix it. Hackers also find their office to be of upmost importance. Long, uninterrupted periods of time to think and get into the flow of programming is vital. Making hackers work in a noisy, distracting environment is like having a paint factory where the air is full of soot. If companies want hackers to be productive, they should look at what they do at home. At home, hackers can arrange things themselves so they can get the most done. And when they work at home, hackers don't work in noisy, open spaces; they work in rooms with doors. They work in cozy, neighborhoody places with people around and somewhere to walk when they need to mull something over, instead of in glass boxes set in acres of parking lots. They have a sofa they can take a nap on when they feel tired, instead of sitting in a coma at their desk, pretending to work. There are no meetings or, God forbid, corporate retreats or team-building exercises.

- Hackers like to work for people with high standards. But it’s not enough to just to be exacting. You have to insist on the right things. Which usually means that you have to be a hacker yourself. If you aren’t a hacker, you can’t tell who the good hackers are. If you don’t have good taste, how are you going to recognize a good designer (having the ability and humility to admit you have no competence in an area and to hire someone who does is what keeps this obvious thing from happening). You can’t manage a process intended to produce beautiful things if you don’t know what beautiful is. American cars are ugly because American car companies are run by people with bad taste.
- Interesting projects have a few big, clear problems rather than a lot of little nasty ones. Working on nasty little problems makes you stupid and great hackers avoid them at all costs.
- Along with interesting problems, what good hackers like is other good hackers. Great hackers tend to clump together – sometimes spectacularly so, as at Xerox Parc. So you won’t attract good hackers in linear proportion to how good an environment you create for them. The tendency to clump means it’s more like the square of the environment. So, it’s winner take all
- The statistics about which languages are most popular can be misleading. What we ought to look at, if we want to know what tools are best, is what hackers choose when they can choose freely – that is, in projects of their own
- Good products, more than brand and dominating channels, win in the market
- With the amount of noise in the signal, it’s hard to tell good hackers when you meet them. I can’t tell, even now. You also can’t tell from their resumes. It seems like the only way to judge a hacker is to work with him on something. Because you can’t tell a great hacker except by working with him, hackers themselves can’t tell how good they are. This is true to a degree in most fields. I’ve found that people who are great at something are not so much convinced of their own greatness as mystified at why everyone else seems so incompetent.
- The key to being a good hacker may be to work on what you like. When I think about the great hackers I know, one thing they have in common is the extreme difficulty of making them work on anything they don’t want to. I don’t know if this is cause or effect; it may be both. To do something well you have to love it. The most common trait hackers have is curiosity, especially about how things work as well as their ability to concentrate – to “tune out everything outside their own heads.” Perhaps great hackers can load a large amount of context into their head, so that when they look at a line of code, they see not just that line but the whole program around it. John McPhee wrote that Bill Bradley’s success as a basketball player was due partly to his extraordinary peripheral vision. “Perfect” eyesight means about 47 degrees of vertical perception. Bill Bradley had 70; he could see the basket when he was looking at the floor. Maybe hackers have some similar inborn ability.
    - *How can you develop your own “peripheral vision”? Become deeply fluent through experience, by touching the medium and through reflection; see through the eyes of others; always be curious and learning; surround yourself with the best people possible; study the big ideas from different fields and ingrain them; take time to read, read, read, think, think, think*
- If it is possible to make yourself into a great hacker, the way to do it may be to make the following deal with yourself: you never have to work on boring projects and in return, you’ll never allow yourself to do a half-assed job

- *This goes with anything, of course. Find something you love to do and you will most likely do it well. Being able to say no, dozens of times potentially, so that you have the time, flexibility, space, mental capacity, passion, etc. to do the "hell yes" projects once they come up is so important*

The Python Paradox

- The Python paradox: if a company chooses to write its software in a comparatively esoteric language, they'll be able to hire better programmers, because they'll attract only those who cared enough to learn it. And for programmers the paradox is even more pronounced: the language to learn, if you want to get a good job is a language that people don't learn merely to get a job

The Age of the Essay

- The modern conception of writing essays seems to students a pointless exercise because we are now three steps removed from real work: the students are imitating English professors, who are imitating classical scholars, who are merely the inheritors of a tradition growing out of what was, 700 years ago, fascinating and urgently needed work
- Defending a position may be a necessary evil in a legal dispute, but it's not the best way to get at the truth, as I think lawyers would be the first to admit. It's not just that you miss subtleties this way. The real problem is that you can't change the question
- Good writing should be convincing, certainly, but it should be convincing because you got the right answers, not because you did a good job of arguing. When I give a draft of an essay to friends, there are two things I want to know: which parts bore them, and which seem unconvincing. The boring bits can usually be fixed by cutting. But I don't try to fix the unconvincing bits by arguing more cleverly. I need to talk the matter over. More often than not I have to change what I was saying as well. But the aim is never to be convincing per se. As the reader gets smarter, convincing and true become identical, so if I can convince smart readers I must be near the truth
- Montaigne penned the first essay or "essai" which in French means to try, an attempt. An essay is something you write to try to figure something out. Figure out what? You don't know yet. It begins with a question and you don't take a side and defend it. You notice door that's ajar, and you open it and walk in to see what's inside. Writing helps you to express ideas and form them. Indeed, helps is far too weak a word. Most of what ends up in my essays I only thought of when I sat down to write them. That's why I write them. Just as inviting people over forces you to clean up your apartment, writing something that other people will read forces you to think well. So it does matter to have an audience
- Questions aren't enough. An essay has to come up with answers. They don't always, of course. Sometimes you start with a promising question and get nowhere. But those you don't publish. Those are like experiments that get inconclusive results. An essay you publish ought to tell the reader something he didn't know
- While writing, flow towards the most interesting. Know what you want to write about generally but not the specific conclusions you want to reach. If you run up against a wall,

backtrack. Essays are interesting if they surprise and you should aim for maximum surprise. Learn the most and retain the most when something surprising comes up – ask people what surprised them most and what was different from what they expected. Surprises are things that you not only didn't know, but that contradict things you thought you knew. And so they're the most valuable sort of fact you can get

  - *This is one of the most powerful, revealing questions – "what surprised you about x…?"*

- You should only write about things which you have thought about a lot and still surprise you as this will probably surprise most readers. To write essays, you need two ingredients: a few topics you've thought about a lot and some ability to ferret out the unexpected. To ferret out the unexpected is somewhat like learning history. When you first learn history, it's just a whirl of names and dates. Nothing seems to stick. But the more you learn, the more hooks you have for new facts to stick onto – when means you accumulate knowledge at what's colloquially called an exponential rate. Collecting surprises is a similar process. The more anomalies you've seen, the more easily you'll notice new ones
  - *In other words, you build a latticework with which to hang these new ideas, mental models, etc. The more you learn the more you can learn as you have more context, more "hooks" with which to refer to, leading to non-linear learning*
- I find it especially useful to ask "why" about things that seem wrong. I have a hunch you want to pay attention not just to things that seem wrong, but things that seem wrong in a humorous way. I'm always pleased when I see someone laugh as they read a draft of an essay. But why should I be? I'm aiming for good ideas. Why should good ideas be funny? The connection may be surprise. Surprises make us laugh, and surprises are what one wants to deliver
  - *Power and strength lies in humor*
- I write down things that surprise me in notebooks. I never actually get around to reading them and using what I've written but I do tend to reproduce the same thoughts later. So the main value of notebooks may be what writing things down leaves in your head.
- People trying to be cool will find themselves at a disadvantage when collecting surprises because to be surprised means to be mistaken. If you want to find surprises, you should do the opposite. Study lots of different things, because some of the most interesting surprises are unexpected connections between different fields. Whatever you study, include history – but social and economic history, not political history. History seems to me so important that it's misleading to treat it as a mere field of study. Another way to describe it is all the data we have so far
- Above all, make a habit of paying attention to things you're not supposed to, either because they're "inappropriate" or not important or not what you're supposed to be working on. If you're curious about something, trust your instincts. Follow the threads that attract your attention. If there's something you're really interested in, you'll find they have an uncanny way of leading back to it anyway, just as the conversation of people who are especially proud of something always tends to lead back to it
- Gradualness is very powerful. And that power can be used for constructive or destructive purposes: just as you can trick yourself into looking like a freak with a comb over, you can trick yourself into creating something so grand you would never have dared to plan such a thing. Indeed, this is how most good software gets created. You start by writing a stripped-down kernel (how hard can it be?) and gradually it grows into a complete operating system

    - *Again, start with baby steps but make sure they are in the right direction and that you are intrigued/passionate about it. Almost all great ventures begin this way*
- Don't do what people expect of you. Don't do as you're told. Don't write the essay readers expect; one learns nothing from what one expects

What the Bubble Got Right

- In order to get a really big bubble you need to have something substantive at the center, so that even smart people are sucked in
- Over the long term, what bubbles get right is more important than what it gets wrong. In the case of the dot com bubble: retail VC (going public before having earnings), the Internet (genuinely game changing), increased choices from expansion of the Internet, the youth with grand visions will never go away, informality with substance over formality and shallow impressions, nerds who focus on substance, options to reward people, startups to develop technology and perhaps getting acquired, California will still be the hub of innovation and technology will keep improving productivity became illogically unfashionable
- Recognizing an important trend turns out to be easier than figuring out how to profit from it. The mistake investors always seem to make is to take the trend too literally. Since the Internet was the big new thing, investors supposed that the more Internettish the company, the better. Hence such companies as Pets.com. In fact, most of the money to be made from big trends is made indirectly. It was not the railroads that made the most money during the railroad boom, but the companies on either side, like Carnegie's steelworks, which made the rails, and Standard Oil, which used railroads to get oil to the East Coast, where it could be shipped to Europe
    - *Likewise, it is easier often to see who loses from a rising trend than who will win. As Buffett mentioned, it would have been easy to see that the horse and trolley would be in trouble when the car emerged but knowing which company would come to dominate the car industry would have been a much more difficult proposition*
- The Internet is shifting companies and people in the direction of quality. Google doesn't have to advertise because being the best is enough. The hard part, if you want to win by making the best stuff, is the beginning. Eventually everyone will learn by word of mouth that you're the best, but how do you survive to that point? The Internet helps because it lets anyone find you at almost zero cost and it dramatically speeds up the rate at which reputation spreads by word of mouth
- Dressing up is not so much bad in itself. The problem is the receptor it binds to: dressing up is inevitably a substitute for good ideas. It is no coincidence that technically inept business types are known as "suits." Nerds don't just happen to dress informally. They do it too consistently. Consciously or not, they dress informally as a prophylactic measure against stupidity.
    - *Graham's definition of a nerd is someone who doesn't expend any effort on marketing himself*
- What makes the Bay Area superior is the attitude of the people – they are both energetic and optimistic

- I'm surprised people still worry about losing jobs due to technology. After centuries of supposedly job-killing innovations, the number of jobs is within 10% of the number of people who want them. This can't be a coincidence. There must be some kind of balancing mechanism
- Good ideas always tend to win eventually. The problem is, it can take a very long time. It took decades for relativity to be accepted, and the greater part of a century to establish that central planning didn't work. So even a small increase in the rate at which good ideas win would be a momentous change – big enough, probably, to justify a name like the "new economy."

A Version 1.0

- The art of writing is rewriting.
- There seem to be several categories of cuts: things I got wrong, things that seem like bragging, flames, digressions, stretches of awkward prose and unnecessary words

Bradley's Ghost

- The Bradley Effect – people know what they believe but they also know what they're supposed to believe. And so, when a stranger asks them their opinion about something like gay marriage, they will not always say what they really think

It's Charisma, Stupid

- My theory doesn't require voters to be so superficial that they only care about charisma. I'm not proposing charisma is the only factor, just that it's the only one left after the efforts of the two parties cancel one another out
    - *Graham's theory is that in US presidential elections, the more charismatic candidate wins. Issues of course matter to voters but the major parties know so well which issues matter how much to how many voters, and adjust their message so precisely in response, that they tend to split the difference on the issues, leaving the election to be decided by the one factor they can't control: charisma*

Made in USA

- Americans are good at some things and bad at others. We're good at making movies and software, and bad at making cars and cities. And I think we may be good at what we're good at for the same reason we're bad at what we're bad at. We're impatient. We prefer to work quickly and iterate rather than slowly and meticulously as this is premature optimization. Better to get a prototype done fast, and see what new ideas it gives you. We don't value design and craftsmanship like the Japanese, we value getting the job done

- Instead of relying on their own internal design compass (like Henry Ford did), American car companies try to make what marketing people think consumers want. But it isn't working. American cars continue to lose market share. And the reason is that the customer doesn't want what he thinks he wants. Letting focus groups design your cars for you only wins in the short term. In the long term, it pays to bet on good design. Only great designers can surprise consumers by satisfying expectations they didn't even know they had
- I think speed is the reason Americans are the clear winners for generating wealth and technical innovations (which are practically the same thing). I think speed is the reason. It's hard to create wealth by making a commodity. The real value is in things that are new, and if you want to be the first to make something, it helps to work fast.

2005

What You'll Wish You'd Known

- *(This was a talk he was supposed to give to high school students and wanted to let them know what he wishes he had known in high school.)* First off, you don't have to know what you want to do with your life. What you need to do is discover what you like. You have to work on stuff you like if you want to be good at what you do. Often it is helpful not to have a set plan because the world changes so fast and the rate at which it changes is itself speeding up. Never giving up on your dreams can be a form of fixedness and premature optimization and can be synonymous with disaster. What people who say this really mean is don't get demoralized.
- One reason we like to believe in genius is that it gives us an excuse for being lazy. If the great inventors, businessmen, athletes, etc. of history did what they did only because of some magic Shakespeareness or Einsteinness, then it's not our fault we can't do something as good. I'm not saying there's no such thing as genius. But if you're trying to choose between two theories and one gives you an excuse for being lazy, the other is probably right.
- Instead of working back from a goal, work forward from promising situations. That is what most successful people actually do anyway. Don't commit to anything in the future, but just look at the options available now, and choose those that will give you the most promising range of options afterward. It's not so important what you work on, so long as you're not wasting your time. Work on things that interest you and increase your options, and worry later about which you'll take. So, I propose that as a replacement for "don't give up on your dreams," aim to stay "upwind." Look for smart people and hard problems. Smart people tend to clump together and if you can find such a clump, it's probably worthwhile to join it. This can be tricky as there is a lot of noise and fakery going on but the best protection is always to be working on hard problems. Hard means worry: if you're not worrying that something you're making will come out badly, or that you won't be able to understand something you're studying, then it isn't hard enough. There has to be suspense. In practice, "stay upwind" reduces to "work on hard problems."
    - *Interesting take. Another helpful model is to "work from the end of the maze." By knowing your ultimate goal and studying those who have achieved something similar, you can eliminate certain false leads. Combining this end of the maze model with staying upwind could be powerful – allowing you to keep the big-picture, long-term goal in mind while still knowing how to optimally act today, ideally by working on hard problems that interest and motivate you with other smart people*
- What's the difference between high school kids and adults? It's not making a living, it's taking responsibility for themselves, especially intellectual responsibility
- When I ask people what they regret most about high school, they nearly all say the same thing: they wasted so much time. If you're wondering what you're doing now that you'll regret later, that's probably it. The mental malaise one feels in high school stems from boredom, being ready for something else and something more. The second biggest regret was caring so much about unimportant things. And especially what other people thought

of them. Adults care about what others think of them but it is a selective group. Your peer group in high school is chosen by accident based on age and geography

- Great work, hard work, rarely translates into a line item on a college application
- The high school system is corrupt but you should not rebel. Rebellion is almost as stupid as obedience. In either case you let yourself be defined by what they tell you to do. The best plan, I think, is to step onto an orthogonal vector. Don't just do what they tell you, and don't just refuse to. Instead, treat school as a day job. As day jobs go, it's pretty sweet. You're done at 3 o'clock, and you can even work on your own stuff while you're there. And, what's your real job supposed to be? Unless you're Mozart, your first task is to figure that out. What are the great things to work on? Where are the imaginative people? And most importantly, what are you interested in? The word "aptitude" is misleading because it implies something innate. The most powerful sort of aptitude is a consuming interest in some question, and such interests are often acquired tastes
- Kid curiosity is broad and shallow; they ask why at random about everything. In most adults this curiosity dries up entirely. But in ambitious adults, instead of drying up, curiosity becomes deep and narrow. Curiosity turns work into play
- One of the most dangerous illusions you get from school is the idea that doing great things requires a lot of discipline. Many people who great work have little discipline. They're terrible procrastinators and find it almost impossible to make themselves do anything they're not interested in. I'm not saying you need zero self-discipline. You probably need about the amount you need to go running. They feel as if they don't work and have enough discipline to get themselves to their desks to start working. But once they get started, interest takes over and discipline is no longer necessary
    - *Use this insight to help you simply get started. Put your running shoes on. Sit at your desk. Make the difficult phone call. One small step…*
- To do great work you need a great curiosity about a promising question. It can take years to zero in on a productive question, because it can take years to figure out what a subject is really about. People who do great things look at the same world everyone else does, but notice some odd detail that's compellingly mysterious
- If it takes years to articulate great questions, what do you do now, at sixteen? Work toward finding one. Great questions don't suddenly appear. They gradually congeal in your head. And what makes them congeal is experience. So the way to find great questions is not to search for them – not to wander about thinking, what great discovery shall I make? You can't answer that; if you could you'd have made it. The way to get a big idea to appear in your head is not to hunt for big ideas, but to put in a lot of time on work that interests you, and in the process keep your mind open enough that a big idea can take root. Einstein, Ford, Beckenbauer all used this recipe. They all knew their work like a piano player knows the keys. So when something seemed amiss to them, they had the confidence to notice it. Put in time how and on what? Just pick a project that seems interesting: to master some chunk of material, or to make something, or to answer some question. Choose a project that will take less than a month, and make it something you have the means to finish. Do something hard enough to stretch you, but only just, especially at first. If you're deciding between two projects, choose whichever seems most fun. If one blows up in your face, start another. Repeat till, like an internal combustion engine, the process becomes self-sustaining and each project generates the next one. (This could take years).

- Don’t disregard unseemly motivations. One of the most powerful is the desire to be better than other people at something. Another powerful motivator is the desire to do, or know, something you’re not supposed to. Closely related is the desire to do something audacious
    - *“I think I've been in the top 5% of my age cohort all my life in understanding the power of incentives, and all my life I've underestimated it. Never a year passes that I don't get some surprise that pushes my limit a little farther.” – Charlie Munger*
- Beware bad models. Especially when they excuse laziness
- A key ingredient in many projects, almost a project in its own, is to find good books. Most books are bad. Nearly all textbooks are bad. So don’t assume a subject is to be learned from whatever book on it happens to be closest. You have to search actively for the tiny number of good books
    - *“Any book which is at all important, should be reread immediately.” – Arthur Schopenhauer*
- The important thing is to get out there and do stuff. Instead of waiting to be taught, go out and learn
- The key to wasting time is distraction. Without distraction, it’s too obvious to your brain that you’re not doing anything with it and you start to feel uncomfortable. If you want to measure how dependent you’ve become on distractions, try this experiment: set aside a chunk of time on a weekend and sit alone and think. You can have a notebook to write your thoughts down in, but nothing else: no friends, TV, music, phone, IM, email, Web, games, books, newspapers or magazines. Within an hour most people will feel a strong craving for distraction

How to Start a Startup

- You need three things to create a successful startup: to start with good people, to make something customers actually want, and to spend as little money as possible. Most startups that fail do it because they fail at one of these. A startup that does all three will probably succeed. And that’s kind of exciting, when you think about it, because all three are doable. Hard, but doable. There is no magically difficult step that requires brilliance to solve
    - *Often, the most important things you can do cost nothing, you have to do them yourself, and nobody can stop you from doing them. Being curious, mastering something, pushing yourself…This is both exciting and frightening as you realize that more often than you care to admit you are your own limiting factor*
- I can think of several heuristics for generating ideas for startups, but most reduce to this: look at something people are trying to do, and figure out how to do it in a way that doesn’t suck
- Ideas are basically worth nothing. The market price is less than the inconvenience of signing an NDA. What matters is not ideas, but the people who have them. Good people can fix bad ideas, but good ideas can’t save bad people
- One of the best tricks I learned during our startup was a rule for deciding who to hire. Could you describe the person as an animal? Someone who takes their work a little too seriously; someone who does what they do so well that they pass right through professional and cross over into obsessive. This may not be ideal in big companies but they are necessary in a startup. As programmers are unique, we had three additional tests. Was the person

genuinely smart? If so, could they actually get things done? And finally, since a few good hackers have unbearable personalities, could we stand to have them around. When nerds are unbearable it's usually because they're trying too hard to seem smart. But the smarter they are, the less pressure they feel to act smart. So as a rule you can recognize genuinely smart people by their ability to say things like "I don't know," "Maybe you're right," and "I don't understand x well enough."

- Ideally you want between two and four founders. It would be hard to start with just one. One person would find the moral weight of starting a company hard to bear. If there are too many founders however, disagreements can linger and harden into factions. You don't want mere voting; you need unanimity
- On the business side, all you need to know to run a startup are commonsense things people knew before they were business schools or even universities. The rulers of technology business tend to come from technology, not business. However, business people can help understand what consumers want. If the technical founder can't understand users, you should either learn how or find a co-founder who can. That is the single most important issue for technology startups, and the rock that sinks more of them than anything else.
- The only way to make something customers want is to get a prototype in front of them and refine it based on their reactions
- It's worth trying very, very hard to make technology easy to use. A 10% improvement in ease of use doesn't just increase your sales 10%. It's more likely to double your sales
- How do you figure out what customers want? Watch them. One of the best places to do this is at trade shows. Trade shows don't pay as a way of getting new customers, but they are worth it as market research. Don't just give canned presentations, show people how to build real, working stores (or whatever your product or service is).
- If you start a startup that has to be a big consumer company, the odds of succeeding are steeper. The best odds are in niche markets. Since startups make money by offering people something better than they had before, the best opportunities are where things suck the most. Imbalance equals opportunity. The smaller markets are more valuable anyway. In technology, the low end always eats the high end. It's easier to make an inexpensive product more powerful than to make a powerful product cheaper. If you build the simple, inexpensive option, you'll not only find it easier to sell at first, but you'll also be in the best position to conquer the rest of the market
    - *Contrast can be your key. If everybody in x industry sucks, simply being good can allow you to dominate the market*
- The way to get rich from a startup is to maximize the company's chances of succeeding, not to maximize the amount of stock you retain. So if you can trade stock for something that improves your odds, it's probably a smart move
- I have no tricks for how to deal with fairly apportioning founder's ownership. All I can say is, try hard to do it right. I do have a rule of thumb of recognizing when you have, though. When everyone feels they're getting a slightly bad deal, that they're doing more than they should for the amount of stock they have, the stock is optimally apportioned.
- Don't do what we did. Before you consummate a startup, ask everyone about their previous IP history
- When raising seed money, the valuation isn't just the value of the code you've written so far, it is also the value of your ideas and for all the future work you'll do, which will be a lot

- You have more leverage negotiating with VCs than you realize. The reason is other VCs. I know a number of VCs now, and when you talk to them you realize that it's a seller's market. Even now there is too much money chasing too few good ideas. While connections and advice from the top firms is truly valuable, be skeptical about the VCs below that top tier. Basically, a VC is a source of money and you should be inclined to go with whoever offered the most money the soonest with the least strings attached
- Talk to as many VCs as you can, even if you don't want their money, because a) they may be on the board of someone who will buy you, and b) if you seem impressive, they'll be discouraged from investing in your competitors. The most efficient way to reach VCs, especially if you only want them to know about you and don't want their money, is at the conferences that are occasionally organized for startups to present to them.
- During the Bubble many startups tried to "get big fast." Ideally this meant getting a lot of customers fast. But it was easy for the meaning to slide over into hiring a lot of people fast. I think in most businesses the advantage of being first to market are not so overwhelmingly great. Google is again a case in point. Unless you're in a market as undifferentiated as cigarettes or vodka or laundry detergent, spending a lot on brand advertising is a sign of breakage.
- There is nothing more valuable, in the early stages of a startup, than smart users. If you listen to them, they'll tell you exactly how to make a winning product. And not only will they give you this advice for free, they'll pay you. That's the key to success as a startup. There is nothing more important than understanding your business. Design your product to please users first, and then think about how to make money from it. If you don't put users first, you leave a gap for competitors who do. To make something users love, you have to understand them. And the bigger you are, the harder that is. So I say "get big slow." The slower you burn through your funding, the more time you have to learn. The other reason to spend money slowly is to encourage a culture of cheapness. For most startups the model should be grad student, not law firm. Aim for cool and cheap, not expensive and impressive
    - *Your counterparties will teach you everything you need to know if you structure decisions to be win/win and are willing to listen to them*
- The key to productivity is for people to come back to work after dinner. Those hours after the phone stops ringing are by far the best for getting work done. Great things happen when a group of employees go out to dinner together, talk over ideas, and then come back to their offices to implement them. So you want to be in a place where there are a lot of restaurants around, not some dreary office park that's a wasteland after 6pm. Once a company shifts over into the model where everyone drives home to the suburbs for dinner, however late, you've lost something extraordinarily valuable. God help you if you actually start in that mode.
- The most important way to not spend money is by not hiring people. I may be an extremist, but I think hiring people is the worst thing a company can do. They are a recurring expense, tend to cause you to grow out of your space, slow you down. So, the fewer the people you can hire, the better. As with office space, the number of your employees is a choice between seeming impressive, and being impressive.
- So, who should start a startup? Someone who is a good hacker, between about 23 and 38 who wants to solve the money problem in one shot instead of getting paid gradually over a conventional working life. 23 because this gives young people a chance to work for an existing business before you try running your own. Negative lessons are just as valuable as

positive ones. Perhaps even more valuable: it's hard to repeat a brilliant performance, but it's straightforward to avoid errors

- Mainly what a startup buys you is time. That's the way to think about it if you're trying to decide whether to start one. If you're the sort of person who would like to solve the money problem once and for all instead of working for a salary for 40 years, then a startup makes sense.
- Grad school and starting a startup are not mutually exclusive. Two of our original hackers were in grad school the whole time, and both got their degrees. There are few sources of energy so powerful as a procrastinating grad student
- If you want to do it, do it. Starting a startup is not the great mystery it seems from outside. It's not something you have to know about "business" to do. Build something users love, and spend less than you make. How hard is that?

A Unified Theory of VC Suckage

- There's a reason VCs are the way they are. It's not so much that the business attracts jerks, or even that the power they wield corrupts them. The real problem is the way they're paid. They get paid a percentage of the money they manage so they asset gather and have to manage many deals, each for multiple millions of dollars. It explains why VCs take so agonizingly long to make up their minds and why their due diligence feels like a body cavity search. With so much at stake, they have to be paranoid. VCs are not intrinsically jerks. They are like car salesman or bureaucrats: the nature of their work turns them into jerks
- Mike Moritz and John Doerr don't behave like traditional VCs. But they work for the very best VC funds. And my theory explains why they'd tend to be different: just as the very most popular kids don't have to persecute nerds, the very best VCs don't have to act like VCs. They get the pick of all the best deals. So they don't have to be paranoid and sneaky.
    - *What others think are the characteristics which make you an anomaly, eccentric or the exception to the rule, are in fact the very things which make you dominant – your competitors simply can't understand them*

Undergraduation

- There are two main things you can do to become a good hacker: become very good at programming and learn a lot about specific, cool problems. These turn out to be equivalent, because each drives you to do the other. The way to be good at programming is to work a) a lot and b) on hard problems. And the way to make yourself work on hard problems is to work on some very engaging project. Odds are it won't be a class assignment
- Look around you and see what the smart people seem to be working on; there's usually a reason
- While you don't literally need math for most kinds of hacking, math is very much worth studying for its own sake. It's a valuable source of metaphors for almost any kind of work. I wish I'd studied more math in college for that reason

- The worthwhile departments, in my opinion, are math, the hard sciences, engineering, history (especially economic and social history, and the history of science), architecture and the classics. A survey course in art history may be worthwhile. Modern literature is important, but the way to learn about it is just to read. I don't know enough about music to say. You can skip the social sciences, philosophy and the various departments created recently in response to political pressures.
- On the whole, grad school is probably better than most alternatives. You meet a lot of smart people, and your glum procrastination will at least be a powerful common bond. And you have a PhD at the end.
- Apparently only recommendations really matter at the best schools. The professors and not the admissions officers decide and as they have to work with who they admit, they choose carefully. They're impressed by students who get good grades and want to be their research assistants because they're genuinely interested in the topic.

Writing Briefly

- I think it's far more important to write well than most people realize. Writing doesn't just communicate ideas; it generates them. If you're bad at writing and don't like to do it, you'll miss out on most of the ideas writing would have generated
- As for how to write well, here's the short version: write a bad version 1 as fast as you can; rewrite it over and over; cut out everything unnecessary; write in a conversational tone; develop a nose for bad writing, so you can see and fix it in yours; imitate writers you like; if you can't get started, tell someone what you plan to write about, then write down what you said; expect 80% of the ideas in an essay to happen after you start writing it, and 50% of those you start with to be wrong; be confident enough to cut; have friends you trust read your stuff and tell you which bits are confusing or drag; don't (always) make detailed outlines; mull ideas over for a few days before writing; carry a small notebook or scrap paper with you; start writing when you think of the first sentence; if a deadline forces you to start before that, just say the most important sentence first; write about stuff you like; don't try to sound too impressive; don't hesitate to change the topic on the fly; use footnotes to contain digressions; use anaphora to knit sentences together; read your essays out loud to see a) where you stumble over awkward phrases and b) which bits are boring (the paragraphs you dread reading); try to tell the reader something new and useful; work in fairly big quanta of time; when you restart, begin by rereading what you have so far; when you finish, leave yourself something easy to start with; accumulate notes for topics you plan to cover at the bottom of the file; don't feel obliged to cover any of them; write for a reader who won't read the essay as carefully as you do, just as pop songs are designed to sound ok on crappy car radios; if you say anything mistaken, fix it immediately; ask friends which sentence you'll regret most; go back and tone down harsh remarks; publish stuff online, because an audience makes you write more, and thus generate more ideas; print out drafts instead of just looking at them on the screen; use simple, Germanic words; learn to distinguish surprises from digressions; learn to recognize the approach of an ending, and when one appears, grab it.

Return of the Mac

- When it comes to computers, what hackers are doing now, everyone will be doing in ten years
- If you want to attract hackers to write software that will sell your hardware, you have to make it something that they themselves use. It's not enough to make it open. It has to be open and good. And open and good is what Macs are again, finally

Why Smart People Have Bad Ideas

- The Y Combinator experiment began in 2005 and they targeted younger founders. They thought there would be two camps: promising ideas and unpromising ideas. They found that they needed a third: promising people with unpromising ideas
- People have bad ideas for several reasons. One reason is that they tend to go with their first idea. You'd think people would spend a couple days thinking through different ideas before spending years on them. They don't. Part of the problem is that big projects tend to grow out of small ones. So the biggest cause of big ideas is the still life effect: you come up with a random idea, plunge into it, and then at each point (a day, a week, a month) feel you've put so much time into it that this must be a bad idea. How do we fix that? I don't think we should discard plunging. Plunging into a good idea is a good thing. The solution is at the other end: to realize that having invested time in something doesn't make it good.
- Don't go into business halfheartedly. Either be all in, wanting to make a great product or service and make money, or stay out of the game
- Don't be so timid. We lacked confidence in our ability to do a mysterious, undifferentiated thing we called "business." In fact, there is no such thing as "business." There's selling, promotion, figuring out what people want, deciding how much to charge, customer support, paying your bills, getting customers to pay you, getting incorporated, raising money, and so on. And the combination is not as hard as it seems, because some tasks (like raising money and getting incorporated) are a pain in the ass, whether you're big or small, and others (like selling and promotion) depend more on energy and imagination than any kind of special training.
- Reading the Wall Street Journal for a week should give anyone ideas for two or three new startups. The articles are full of descriptions of problems that need to be solved.
- Why did so few applications really think about what customers want? I think the problem with many, as with people in their early twenties generally, is that they've been trained their whole lives to jump through predefined hoops. They're good at solving problems, but bad at choosing them. It would be great if schools taught students how to choose problems as well as how to solve them, but I don't know how you'd run such a class in practice.
- That's the essence of a startup: having brilliant people do work that's beneath them. Big companies try to hire the right person for the job. Startups win because they don't – because they take people so smart that they would in a big company be doing "research," and set them to work instead on problems of the most immediate and mundane sort. Think Einstein designing refrigerators.
- If you want to learn what people want, read Dale Carnegie's *How to Win Friends and Influence People.* When a friend recommended this book, I couldn't believe he was serious.

But he insisted it was good, so I read it, and he was right. It deals with the most difficult problem in human experience: how to see things from other people's point of view, instead of thinking only of yourself. Most smart people don't do that very well. But adding this ability to raw brainpower is like adding tin to copper. The result is bronze, which is so much harder that it seems a different metal.
    - *One of the surest ways to succeed is through empathy – being able to see the world through other's eyes.*
    - *This may be the best personal advice you're ever going to get. Rather than trying to become a more "pure" version of yourself, mix in characteristics which aren't normally found in your personality type. If you're type-A and are ambitious, outgoing, confident, mix in some patience and empathy. If you're shy and intellectual, become a great public speaker and work on your social skills. This combination can lead to leaping emergent effects and be deadly*
- Trevor Blackwell presents the following recipe for a startup: "Watch people who have money to spend, see what they're wasting their time on, cook up a solution, and try selling it to them. It's surprising how small a problem can be and still provide a profitable market for a solution."
- You need to offer especially large rewards to get great people to do tedious work. That's why startups always pay equity rather than just salary

The Submarine

- PR is like a huge, lurking, quiet submarine beneath the news. Of the stories you read in traditional media that aren't about politics, crimes, or disasters, more than half probably come from PR firms
- PR is the news equivalent of search engine optimization; instead of buying ads, which readers ignore, you get yourself inserted directly into the stories
- PR is not dishonest. If anyone is dishonest, it's the reporters. The main reason PR firms exist is that reporters are lazy. Or, to put it more nicely, overworked

Hiring is Obsolete

- The less it costs to start a company, the less you need the permission of investors to do it. So a lot of people will be able to start companies now who never could have before
- The most interesting subset of founders may be those in their early twenties. The most promising group to be liberated by the new, lower threshold are those who have everything investors want, except experience
- It's hard to judge the young because a) they change rapidly, b) there is great variation between them, and c) they're individually inconsistent. That last one is a big problem. When you're young, you occasionally say and do stupid things even when you're smart
- When most people hear the word "startup," they think of the famous ones that have gone public. But most startups that succeed do it by getting bought. And usually the acquirer doesn't just want the technology, but the people who created it as well.

- The central problem in big companies, and the main reason they're so much less productive than small companies, is the difficulty of valuing each person's work. Buying larval startups solves that problem for them: the acquirer doesn't pay till the developers have proven themselves. Acquirers are protected on the downside, but still get most of the upside. Buying startups also solves another problem afflicting big companies: they can't do product development. Big companies are good at extracting the value from existing products, but bad at creating new ones. Why? To start with, most big companies have some kind of turf to protect, and this tends to warp their development decisions. Another reason is that the kind of people who do that tend not to have much power in big companies (unless they happen to be the CEO). Disruptive technologies are developed by disruptive people. And they either don't work for the big company, or have been outmaneuvered by yes-men and have comparatively little influence. Big companies usually only build one thing each, there are too many new ideas for companies to explore them all, they don't pay people the right way, everything happens slower in big companies and product development is something that has to happen fast, because you have to go through a lot of iterations to get something good.
- It's painful doing sales, but you learn much more from trying to sell people something than reading what they said in a focus group
- I think companies that acquire technology will gradually learn to go after earlier stage startups. They won't necessarily buy them outright. The solution may be some hybrid of investment and acquisition: for example, to buy a chunk of the company and get an option to buy the rest later
- Measured on the time scale of social change, what we have now is pre-beta. So we shouldn't assume the way startups work now is the way they have to work
- When I talk to undergrads, what surprises me most about them is how conservative they are. Not politically of course. I mean they don't seem to want to take risks. This is a mistake, because the younger you are the more risk you can take. Risk and reward are always proportionate. If you're young, you should take the riskiest investments you can find. Most undergrads may think they have nothing to invest, but that's not true: they have their time to invest, and the same rule about risk applies there. Your early twenties are exactly the time to take insane career risks. Even if your startup does tank, you won't harm your prospects with employers.
- I now have some data on what an inexperienced person is missing. I've said that every startup needs three things: to start with good people, to make something users want, and not to spend too much money. It's the middle one you get wrong when you're inexperienced. There are plenty of undergrads with enough technical skill to write good software, and undergrads are not especially prone to waste money. If they get something wrong, it's usually not realizing they have to make something people want. Usually, all it takes is to smack hackers on the side of the head and tell them: wake up. Don't sit here making up a priori theories about what users need. Go find some users and see what they need. Once you grasp that you have to build something users need, the next step is figuring out what those problems are. And that takes some effort, because the way software actually gets used, especially by the people who pay the most for it, is not at all what you might expect. For example, the stated purpose of PowerPoint is to present ideas. Its real role is to overcome people's fear of public speaking. It allows you to give an impressive-looking

talk about nothing, and it causes the audience to sit in a dark room looking at slides, instead of a bright one looking at you
    - *Avoid PowerPoint as much as possible. Have the talk internalized and use props, cues, stories in order to appeal to people's heart instead of their head. This will make your argument and presentation that much more memorable and convincing*

What Business Can Learn From Open Source

- I think the most important of the new principles business has to learn is that people work a lot harder on stuff they like. Well, that's news to no one. So how can I claim business has to learn it? When I say business doesn't know this, I mean the structure of the business doesn't reflect it
- Like rich food, idleness only seems desirable when you don't get enough of it
- If you depend on an oligopoly, as the print media has done for some time, you sink into bad habits that are hard to overcome when you suddenly get competition
- To me, the most demoralizing aspect of a traditional office is that you're supposed to be there at certain times. If you could measure how much work people did, many companies wouldn't need any fixed workday. You could just say: this is what you have to do. Do it whenever you like, wherever you like
- Meetings are like an opiate with a network effect. So is email, on a smaller scale. And in addition to the direct cost in time, there's cost in fragmentation – breaking people's day up into bits too small to be useful
- The other problem with pretend work is that it often looks better than real work. When I'm writing or hacking I spend as much time just thinking as I actually do typing. Half the time I'm sitting drinking a cup of tea, or walking around the neighborhood. This is a critical phase – this is where ideas come from – and yet I'd feel guilty doing this in most offices, with everyone else looking busy
- One of the key tenets of being a professional is that work and life are supposed to be separate. But that part, I'm convinced, is a mistake
- Another big lesson business can learn from open source and blogging is that ideas can bubble up from the bottom, instead of flowing down from the top. Open source and blogging both work bottom-up: people make what they want, and the best stuff prevails
- There are two forces that together steer design: ideas about what to do next, and the enforcement of quality. In the channel era, both flowed down from the top. For example, newspaper editors assigned stories to reporters, then edited what they wrote. However, today, open source shows there is a better way. For example, open source software is more reliable precisely because it's open source; anyone can find mistakes
- I'm not claiming companies can take these lessons, incorporate them and get smarter, just that the dumb ones will die
- It's demoralizing to be on the receiving end of a paternalistic relationship, no matter how cozy the terms. Just ask any teenager
- At the moment, even the smartest students leave school thinking they have to get a job. Actually, what they need to do is make something valuable. A job is one way to do that, but the more ambitious ones will ordinarily be better off taking money from an investor than an employer

- It’s a lot harder to create something people love than to take something people love and figure out how to make money from it
- That may be the greatest effect, in the long run, of the forces underlying open source and blogging: finally ditching the old paternalistic employer-employee relationship, and replacing it with a purely economic one, between equals

After the Ladder

- Thirty years ago, one was supposed to work one’s way up the corporate ladder. That’s less the rule now. Our generation wants to get paid up front. Instead of developing a product for some big company in the expectation of getting job security in return, we develop the product ourselves, in a startup, and sell it to the big company. At the very least we get options

Inequality and Risk

- There is of course a way to make the poor richer without simply shifting money from the rich. You could help the poor become more productive – for example, by improving access to education
- If you want to reduce economic inequality instead of just improving the overall standard of living, it’s not enough just to raise up the poor, you have to push down on the top as well as pushing up on the bottom. The only practical solution is to let people do the best work they can, and then (either by taxation or by limiting what they can charge) to confiscate whatever you deem to be surplus. So let’s be clear what reducing economic inequality means. It is identical with taking money from the rich, thereby decreasing the risk people are willing to take as the rewards are lower. If large payoffs aren’t allowed, you might as well play it safe and this would reduce the number of startups and innovation overall
    - o *See this in socialist-type countries where the rate of innovation and appetite for risk are suppressed relative to the US*
- The problem here is not wealth, but corruption. We don’t need to prevent people from being rich if we can prevent wealth from translating into power. And there has been progress on that front. How do you break the connection between wealth and power? Demand transparency
- I don’t think many people realize there is a connection between economic inequality and risk. I didn’t fully grasp it till recently. Likewise, it’s obvious empirically that a country that doesn’t let people get rich is headed for disaster. If you try to attack wealth, you end up nailing risk as well, and with it growth. If we want a fairer world, I think we’re better off attacking one step downstream, where wealth turns into power
- The variation in wealth in a non-corrupt country or organization will be inversely proportional to the prevalence of systems of seniority. So if you suppress variation in wealth, seniority will become correspondingly more important. Though in very corrupt countries you may get both simultaneously

What I Did This Summer

- One thing the young founders in the first Y Combinator class had in common was that they all worked ridiculously hard. People this age are commonly seen as lazy. I think in some cases it's not so much that they lack appetite for work, but that the work they're offered is unappetizing. Like good athletes, they don't work hard because the coach yells at them, but because they want to win
- The summer founders were as a rule very idealistic. They also wanted very much to get rich. These qualities may seem incompatible, but they're not. These guys wanted to get rich, but they want to do it by changing the world
- Here's a handy rule for startups: competitors are rarely as dangerous as they seem. Most will self-destruct before you can destroy them. And it certainly doesn't matter how many of them there are, any more than it matters to the winner of a marathon how many runners are behind him
- We advise groups to ignore issues like scalability, internationalization and heavy-duty security at first. I can imagine an advocate of "best practices" saying these ought to be considered from the start. And he'd be right, except that they interfere with the primary function of software in a startup: to be a vehicle for experimenting with its own design. Having to retrofit internationalization and scalability is a pain, certainly. The only bigger pain is not needing to, because your initial version was too big and rigid to evolve into something users wanted. I suspect this is another reason startups beat big companies. Startups can be irresponsible and release version 1s that are light enough to evolve. In big companies, all the pressure is in the direction of over-engineering
- It was surprising – slightly frightening even – how fast these groups learned. Just as happens in college, the summer founders learned a lot from each other – maybe more than they learned from us.
- Several groups said our weekly dinners saved them from a common problem afflicting startups: working so hard that one has no social life. I remember that part all too well. This way, they were guaranteed a social event at least once a week.
- One quality all the founders shared this summer was a spirit of independence. I've been wondering about that. Are some people just a lot more independent than others, or would everyone be this way if they were allowed to? As with most nature/nurture question, the answer is probably: some of each. But my main conclusion from this summer is that there's more environment in the mix than most people realize
- When we asked the summer founders what surprised them the most about starting a company, one said "the most shocking thing is that it worked." My guess is that if you put people in a position of independence, they develop the qualities they need
- Power is shifting from the people who deal with money to the people who create technology, and if our experience this summer is any guide, this will be a good thing

Ideas for Startups

- One question I receive the most often is how do you get good ideas for startups? What people usually say is not that they can't think of ideas, but that they don't have any. That's not quite the same thing. It could be the reason they don't have any is that they haven't tried to generate them. I think this is often the case. I think people believe that coming up with ideas for startups is very hard – that it must be very hard – and so they don't try to do it. They assume ideas are like miracles: they either pop into your head or they don't. I also have a theory about why people think this. They overvalue ideas. They think creating a startup is just a matter of implementing some fabulous initial idea. And since a successful startup is worth millions of dollars, a good idea is therefore a million-dollar idea. Actually, startup ideas are not million dollar ideas, and here's an experiment you can try to prove it: just try to sell one. Nothing evolves faster than markets. The fact that there's no market for startup ideas suggests there's no demand. Which means, in the narrow sense of the word, that startup ideas are worthless

- The fact is, most startups end up nothing like the initial idea. It would be closer to the truth to say the main value of your initial idea is that, in the process of discovering it's broken, you'll come up with your real idea. This initial idea is just a starting point – not a blueprint, but a question. It might help if they were expressed that way. Instead of saying that your idea is to make a collaborative, web-based spreadsheet, say: could one make a collaborative, web-based spreadsheet? A few grammatical tweaks and a woefully incomplete idea becomes a promising question to explore. Treating a startup idea as a question changes what you're looking for. If an idea is a blueprint, it has to be right. But if it's a question, it can be wrong, so long as it's wrong in a way that leads to more ideas.
- So far, we've reduced the problem from thinking of a million-dollar idea to thinking of a mistaken question. That doesn't seem so hard, does it? To generate such questions, you need two things: to be familiar with promising new technologies, and to have the right kind of friends. New technologies are the ingredients startups are made of, and conversations with friends are the kitchen they're cooked in. Universities have both, and that's why so many startups grow out of them.
- Do you need other people? Can't you just think of new ideas yourself? The empirical answer is: no. Even Einstein needed people to bounce ideas off. Ideas get developed in the process of explaining them to the right kind of person. You need that resistance, just as a carver needs the resistance of the wood. This is one reason Y Combinator has a rule against investing in startups with only one founder. Practically every successful company has at least two. And because startup founders work under great pressure, it's critical they be friends. What these groups of co-founders do together is more complicated than just sitting down and trying to think of ideas. I suspect the most productive setup is a kind of together-alone-together sandwich. Together you talk about some hard problem, probably getting nowhere. Then, the next morning, one of you has an idea in the shower about how to solve it. He runs eagerly to tell the others, and together they work out the kinks. What happens in that shower? It seems to me that ideas just pop into my head. But can we say more than that? Taking a shower is like a form of meditation. You're alert, but there's nothing to distract you. It's in a situation like this, where your mind is free to roam, that it bumps into new ideas. What happens when your mind wanders? It may be like doodling. Most people have characteristic ways of doodling. This habit is unconscious, but not random: I found my doodles changed after I started studying painting. I started to make the kind of gestures I'd make if I were drawing from life. They were atoms of drawing, but arranged randomly. Perhaps letting your mind wander is like doodling with ideas. You have certain mental gestures you've learned in your work, and when you're not paying attention, you keep making these same gestures, but somewhat randomly. In effect, you call the same functions

on random arguments. That's what a metaphor is: a function applied to an argument of the wrong type. If new ideas are like doodles, this would explain why you have to work at something for a while before you have any. It's not just that you can't judge ideas till you're an expert in a field. You won't even generate ideas, because you won't have any habits of mind to invoke. Of course the habits of mind you invoke on some field don't have to be derived from working in that field. In fact, it's often better if they're not. You're not just looking for good ideas, but for good new ideas and you have a better chance of generating those if you combine stuff from distant fields. Math and any related fields (even if in unexpected ways) are worth studying to help with this process

- I find that to have good ideas I need to be working on some problem. You can't start with randomness. You have to start with a problem, then let your mind wander just far enough for new ideas to form. In a way, it's harder to see problems than their solutions. Most people prefer to remain in denial about problems. It's obvious why: problems are irritating. They're problems! Let me repeat that recipe: finding the problem intolerable and feeling it must be possible to solve it. Simple as it seems that's the recipe for a lot of startups
- Good ideas and valuable ideas are not quite the same thing; the difference is in individual tastes. But valuable ideas are very close to good ideas, especially in technology
- Another classic way to make something people want is to take a luxury and make it into a commodity. People must want something if they pay a lot for it. And it is a very rare product that can't be made dramatically cheaper if you try. When you make something cheaper you can sell more of them. But if you make something dramatically cheaper you often get qualitative changes, because people start to use it in different ways. For example, once computers got so cheap that most people can have one of their own, you can use them as communication devices. Often, to make something dramatically cheaper, you have to redefine the problem. The Model T didn't have all the features previous cars did. It only came in black, for example. But it solved the problem people cared about most, which was getting from place to place
- One of the most useful mental habits I know I learned from Michael Rabin: that the best way to solve a problem is often to redefine it. A lot of people use this technique without being consciously aware of it, but Rabin was spectacularly explicit. Redefining the problem is a particularly juicy heuristic when you have competitors, because it's so hard for rigid-minded people to follow. You can work in plain sight and they don't realize the danger. Don't worry about us. We're just working on search. Do one thing and do it well, that's our motto.
- Making things cheaper is actually a subset of a more general technique: making things easier. For a long time, it was most of making things easier, but now that the things we build are so complicated, there's another rapidly growing subset: making things easier to use. Simplicity takes effort – genius, even
- Another source of ideas is to look at big companies and think of what they should be doing, and do it yourself. Even if they already know it, you'll probably be done faster. Just be sure to make something multiple acquirers will want
- The Woz Rule – the most productive way to generate startup ideas is also the most unlikely-sounding: by accident. If you look at how famous startups got started, a lot of them weren't initially supposed to be startups. Apple got started because Steve Wozniak wanted to build microcomputers, and his employer, Hewlett-Packard, wouldn't let him do it at work. This is not the only way to start startups. You can sit down and consciously come up with an

idea for a company; we did. But measured in total market cap, the build-stuff-for-yourself model might be more fruitful. It certainly has to be the most fun way to come up with startup ideas. And since a startup ought to have multiple founders who were already friends before they decided to start a company, the rather surprising conclusion is that the best way to generate startup ideas is to do what hackers do for fun: cook up amusing hacks with your friends. It seems like it violates some kind of conservation law, but there it is: the best way to get a "million-dollar idea" is just to do what hackers enjoy doing anyway.

The Venture Capital Squeeze

- In the next few years, venture capital funds will find themselves squeezed from four directions. The four causes: open source, which makes software free; Moore's Law, which makes hardware geometrically closer to free; the Web, which makes promotion free if you're good; and better languages, which make development a lot cheaper. VCs are already awash with cash, there is too much money chasing too few deals and now those few deals want less money, because it's getting so cheap to start a startup
- During the Bubble, a lot of people predicted that startups would outsource their development to India. I think a better model for the future is David Heinemeier Hansson, who outsourced his development to a more powerful language instead.
- There are two things VCs need to do, one of which won't surprise them, and another that will seem an anathema. The obvious one is to lobby to get Sarbanes-Oxley loosened. This law was created to prevent future Enrons, not to destroy the IPO market. The second suggestion will seem shocking to VCs: let founders cash out partially in the Series A round. Most VCs have an almost religious rule against doing this. They don't want founders to get a penny till the company is sold or goes public. VCs are obsessed with control, and they worry that they'll have less leverage over the founders if the founders have any money. This is a dumb plan. In fact, letting the founders sell a little stock early would generally be better for the company, because it would cause the founders' attitudes toward risk to be aligned with the VCs'. This will help blunt the terrifying all-or-nothingness of a startup, which in its raw form is more a distraction than a motivator. If VCs are frightened at the idea of letting founders partially cash out, let me tell them something still more frightening: you are now competing directly with Google as they are acquiring companies near their Series A stage.

How to Fund a Startup

- Venture funding works like gears. A typical startup goes through several rounds of funding, and at each round you want to take just enough money to reach the speed where you can shift into the next gear. Few startups get it quite right. Many are underfunded. A few are overfunded, which is like trying to start driving in third gear
- I was surprised recently when I realized that all the worst problems we faced in our startup were due not to competitors, but investors. Dealing with competitors was easy by comparison
- There are five sources of startup funding:

- Friends and family – easy to find, already know them but the disadvantages are you mix together your business and professional life; they will probably not be as well connected as angels or venture firms; and they may not be accredited investors, which could complicate your life later
- Consulting – another way to fund a startup is to get a job. The best sort of job is a consulting project in which you can build whatever software you wanted to sell as a startup. Then you can gradually transform yourself from a consulting company into a product company, and have your clients pay your development expenses. There are millions of small businesses but only a few thousand startups. To be a startup, a company has to be a product business, not a service business. By which I mean not that it has to make something physical, but that it has to have one thing it sells to many people, rather than doing custom work for individual clients. Custom work doesn't scale. Fear of failure is an extraordinarily powerful force. Usually it prevents people from starting things, but once you publish some definite ambition, it switches directions and starts working in your favor. I think it's a pretty clever piece of jiu-jitsu to set this irresistible force against the slightly less immovable object of becoming rich
- Angel investors – angels are individual rich people. Angels who've made money in technology are preferable, for two reasons: they understand your situation, and they're a source of contacts and advice. The contacts and advice can be more important than the money. This is now professional funding which means you have to have an exit strategy – acquisition or going public – for angels to consider investing in you. Early on in a startup's process, the valuation number is just an artifact of the respective contributions of everyone evolved. However, as the company gets more established, its valuation gets closer to an actual market value. It's illegal to adjust the valuation up or down for each investor so if you're going to sell cheap stock to eminent angels, do it early, when it's natural for the company to have a low valuation. One of the dangers of taking investment from individual angels, rather than through an angel group or investment firm, is that they have less reputation to protect. Angels have a corresponding advantage, however: they're also not bound by all the rules that VC firms are. The best way to find angel investors is through personal introductions. You could try to cold call angel groups near you, but angels, like VCs, will pay more attention to deals recommended by someone they respect. Deal terms with angels vary a lot but one solution would be to have the startup's lawyer produce the agreement, instead of the angel's. With both investors and acquirers, the only leverage you have is competition. If an investor knows you have other investors lined up, he'll be a lot more eager to close – and not just because he'll worry about losing the deal, but because if other investors are interested, you must be worth investing in. It's the same with acquisitions. No one wants to buy you till someone else wants to buy you, and then everyone wants to buy you. The key to closing deals is never to stop pursuing alternatives.
- Seed Funding Firms – Seed firms are like angels in that they invest relatively small amounts at early stages, but like VCs in that they're companies that do it as a business, rather than individuals making occasional investments on the side. Because seed firms are companies rather than individual people, reaching them is

easier and the investment process is more standardized. Seed firms differ from angels and VCs in that they exclusively invest in the earliest phases – often when the company is still just an idea. The startup may completely redefine their idea during this time so seed investors usually care less about the idea than the people. Seed firms offer advice about technical as well as business problems

  - Venture Capital Firms – VC firms are like seed firms in that they're actual companies, but they invest other people's money, and much larger amounts of it. VC investments average several million dollars. So they tend to come later in the life of a startup, are harder to get, and come with tougher terms. There is a sharp drop off in performance among VC firms, because in the VC business both success and failure are self-perpetuating. In a sense, the lower-tier VC firms are a bargain for founders. They may not be quite as smart or as well connected as the big-name firms, but they are much hungrier for deals. This means you should be able to get better terms from them. There tends to be a fairly high break-rate with the lower-tier firms. One solution is to ask how many of their last 10 term sheets turned into deals when a VC offers you a term sheet. This will at least force them to lie outright if they want to mislead you. Would strongly advise against mailing your business plan randomly to VCs, because they treat this as evidence of laziness. Do the extra work of getting personal introductions. One of the most difficult problems for startup founders is deciding when to approach VCs. So when do you approach VCs? When you can convince them. Some VCs may defend what they're doing by saying it is the standard in the industry. Standard, schmandard. On that scale, every negotiation is unique.

- Every startup's rule should be: spend little, and work fast. Being slightly underfunded teaches founders an important lesson. For a startup, cheapness is power. The lower your costs, the more options you have
- Develop a skeleton business plan early on, addressing the five fundamental questions: what they're going to do, why users need it, how large the market is, how they'll make money, and who the competitors are and why this company is going to beat them
- How much stock do you give early employees? That varies so much that there's no conventional number. If you get someone really good, really early, it might be wise to give him as much stock as the founders. The one universal rule is that the amount of stock an employee gets decreases polynomially with the age of the company. In other words, you get rich as a power of how early you were. So if some friends want you to come work for their startup, don't wait several months before deciding.
- VCs all ask the same question: who else have you pitched to? VCs are like high school girls: they're acutely aware of their position in the VC pecking order, and their interest in a company is a function of the interest other VCs show in it.
- Closing is not what deals do. What deals do is fall through. If you're starting a startup you would do well to remember that. Birds fly; fish swim; deals fall through
- The very uncertainty of startups frightens away almost everyone. People overvalue stability – especially young people, who ironically need it least. And so in starting a startup, as in any really bold undertaking, merely deciding to do it gets you halfway there. On the day of the race, most of the other runners won't show up

Web 2.0

- The sites which use the power of democracy suggests that voters do a significantly better job than human editors
- Another element of Web 2.0 is the desire to not maltreat users. Don't make users register, unless you need to in order to store something for them. If you do make users register, never make them wait for a confirmation link in an email; in fact, don't even ask for their email address unless you need it for some reason. Don't ask them any unnecessary questions. Never send them email unless they explicitly ask for it. Never frame pages you link to, or open them in new windows. If you have a free version and a pay version, don't make the free version too restricted. And if you find yourself asking, "should we allow users to do x?" just answer "yes" whenever you're unsure. Err on the side of generosity.
- In How to Start a Startup, I advised startups never to let anyone fly under them, meaning never to let any other company offer a cheaper, easier solution. Another way to fly low is to give users more power. Let users do what they want. If you don't and a competitor does, you're in trouble
- The ultimate way to be nice to users is to give them something free that competitors charge for. The most successful sites are the ones that figure out new ways to give stuff away for free.
- Odd as it might sound, we tell startups that they should try to make as little money as possible. If you can figure out a way to turn a billion dollar industry into a fifty million dollar industry, so much the better, if all fifty million go to you. Though indeed, making things cheaper often turns out to generate more money in the end, just as automating things often turns out to generate more jobs.
- Web 2.0 means using the web the way it's meant to be used. Ajax ("JavaScript now works"), democracy and not dissing users
- Google is a prime example of Web 2.0. They don't try to force things to happen their way. They try to figure out what's going to happen, and arrange to be standing there when it does. That's the way to approach technology – and as business includes an ever larger technological component, the right way to do business

Good and Bad Procrastination

- The most impressive people I know are all terrible procrastinators. So could it be that procrastination isn't always bad? There are infinite number of things you could be doing. No matter what you work on, you're not working on everything else. So the question is not how to avoid procrastination, but how to procrastinate well. There are three variants of procrastination, depending on what you do instead of working on something: you could work on (a) nothing, (b) something less important, or (c) something more important. That last type, I'd argue, is good procrastination. That's the "absent-minded professor," who forgets to shave, or eat, or even perhaps look where he's going while he's thinking about some interesting question. His mind is absent from the everyday world because it's hard at work in another. That's the sense in which the most impressive people I know are all procrastinators. They're type-C procrastinators: they put off working on small stuff to work on big stuff. Good procrastination is avoiding errands to do real work. Good in a sense, at least. The people who want you to do the errands won't think it's good. But you probably have to annoy them if you want to get anything done. The mildest seeming people, if they want to do real work, all have a certain degree of ruthlessness when it comes to avoiding errands.

- The reason it pays to put off even urgent errands is that real work needs two things errands don't: big chunks of time and the right mood. If you get inspired by some project, it can be a net win to blow off everything you were supposed to do for the next few days to work on it.
- When I think of the people I know who've done great things, I don't imagine them dutifully crossing items off a to-do list. I imagine them sneaking off to work on some new idea.
- You probably only have to interrupt someone a couple times a day before they're unable to work on hard problems at all
- The most dangerous form of procrastination is unacknowledged type-B procrastination, because it doesn't feel like procrastination. You're "getting things done." Just the wrong things. Unless you're working on the biggest things you could be working on, you're type-B procrastinating, no matter how much you're getting done.
- Richard Hamming suggests you ask yourself three questions: what are the most important problems in your field? Are you working on one of them? Why not?
- The bigger the problem, the harder it is to get yourself to work on it. Why is it so hard to work on big problems? One reason is that you may not get any reward in the foreseeable future. If you work on something you can finish in a day or two, you can expect to have a nice feeling of accomplishment fairly soon. If the reward is indefinitely far in the future, it seems less real. Another reason people don't work on big projects is, ironically, fear of wasting time. What if they fail? Then all the time they spent on it will be wasted. (In fact it probably won't be, because work on hard projects almost always leads somewhere.) In addition, big problems are terrifying. There's an almost physical pain in facing them. It's like having a vacuum cleaner hooked up to your imagination. All your initial ideas get sucked out immediately, and you don't have any more, and yet the vacuum cleaner is still sucking. You can't look at a big problem too directly in the eye. You have to approach it somewhat obliquely. But you have to adjust the angle just right: you have to be facing the big problem directly enough that you catch some of the excitement radiating from it, but not so much that it paralyzes you. You can tighten the angle once you get going, just as a sailboat can sail closer to the wind once it gets underway. If you want to work on big things, you seem to have to trick yourself into doing it. You have to work on small things that could grow into big things, or work on successively larger things, or split the moral load with collaborators. It's not a sign of weakness to depend on such tricks. The very best work has been done this way
- I think the way to "solve" the problem of procrastination is to let delight pull you instead of making a to-do list push you. Work on an ambitious project you really enjoy, and sail as close to the wind as you can, and you'll leave the right things undone.

2006

How To Do What You Love

- Once, when I was about 9 or 10, my father told me I could be whatever I wanted when I grew up, so long as I enjoyed it. I remember that precisely because it seemed so anomalous. It was like being told to use dry water. Whatever I thought he meant, I didn't think he meant work could literally be fun – fun like playing. It took me years to grasp that
- Why is it conventional to pretend to like what you do? If you have to like something to do it well, then the most successful people will all like what they do
- It was not until I was in college that the idea of work finally broke free from the idea of making a living. The definition of work was now to make some original contribution to the world, and in the process not to starve
- Here's an upper bound: do what you love doesn't mean do what you would like to do most this second. The rule of doing what you love assumes a certain length of time. It doesn't mean, do what will make you happiest this second, but what will make you happiest over some longer period, like a week or a month.
    - *One's ability to opt for long-term gratification over short-term may be one of the biggest predictors of success*
- As a lower bound, you have to like your work more than any unproductive pleasure. You have to like what you do enough that the concept of "spare time" seems mistaken. Which is not to say you have to spend all your time working. You can only work so much before you get tired and start to screw up. Then you want to do something else – even something mindless. But you don't regard this time as the prize and the time you spend working as the pain you endure to earn it. I put the lower bound there for practical reasons. If your work is not your favorite thing to do, you'll have terrible problems with procrastination. You'll have to force yourself to work, and when you resort to that the results are distinctly inferior
- To be happy I think you have to be doing something you not only enjoy, but admire. You have to be able to say, at the end, wow that's pretty cool. This doesn't mean you have to make something. If you learn how to hang glide or to speak a foreign language fluently, that will be enough to make you say, at least for a while, wow, that's pretty cool. What there has to be is a test
- So one thing that falls just short of the standard, I think, is reading books. Except for some books in math and the hard sciences, there's no test of how well you've read a book, and that's why merely reading books doesn't quite feel like work. You have to do something with that you've read to feel productive
- I think the best test is one Gino Lee taught me: to try to do things that would make your friends say wow. But it probably wouldn't start to work properly till about age 22, because most people haven't had a big enough sample to pick friends from before then.
- What you should not do, I think, is worry about the opinion of anyone beyond your friends. Don't worry about prestige. It causes you to work not on what you like, but what you'd like to like. Prestige is just fossilized inspiration. If you do anything well enough, you'll make it prestigious. Prestige is especially dangerous to the ambitious. If you want to make ambitious people waste their time on errands, the way to do it is to bait the hook with prestige. That's the recipe for getting people to give talks, write forewords, serve on

committees, be department heads, and so on. It might be a good rule simply to avoid any prestigious task. If it didn't suck, they wouldn't have had to make it prestigious

- Similarly, if you admire two kinds of work equally, but one is more prestigious, you should probably choose the other. Your opinions about what's admirable are always going to be slightly influenced by prestige, so if the two seem equal to you, you probably have more genuine admiration for the less prestigious one
- The reason the young care so much about prestige is that the people they want to impress are not very discerning
- Money by itself is not that dangerous. The danger is when money is combined with prestige, as in, say, corporate law, or medicine. The test of whether people love what they do is whether they'd do it even if they weren't paid for it – even if they had to work at another job to make a living. How many corporate lawyers would do their current work if they had to do it for free, in their spare time, and take day jobs as waiters to support themselves?
- Parents tend to be more conservative for their kids than they would for themselves, simply because, as parents, they share risks more than rewards. If your eight year old son decides to climb a tree, or your teenage daughter decides to date the local bad boy, you won't get a share in the excitement, but if your son falls, or your daughter gets pregnant, you'll have to deal with the consequences.
- It is genuinely difficult work to find what you love and don't feel bad if you haven't succeeded yet. In fact, if you admit to yourself that you're discontented, you're a step ahead of most people, who are still in denial. If you're surrounded by colleagues who claim to enjoy work that you find contemptible, odds are they're lying to themselves. Not necessarily, but probably
- Doing great work takes less discipline than people think – because the way to do great work is to find something you like so much that you don't have to force yourself to do it – finding work you love does usually require discipline
- Plenty of people who will later do great things seem to be disappointments early on, when they're trying to find their niche
- Is there some test you can use to keep yourself honest? One is to try to do a good job at whatever you're doing, even if you don't like it. Then at least you'll know you're not using dissatisfaction as an excuse for being lazy. Perhaps more importantly, you'll get into the habit of doing things well
- Another test you can use is: always produce. Are you writing pages of fiction, however bad? As long as you're producing, you'll know you're not merely using the hazy vision of the grand novel you plan to write one day as an opiate. The view will be obstructed by the all too palpably flawed one you're actually writing. "Always produce" will discover your life's work the way water, with the aid of gravity, finds the hole in your roof
- When people can't do what they love, they protect themselves by saying I can't and by rationalizing doing unpleasant things due to money or prestige. Most unpleasant jobs would either get automated or go undone if no one were willing to do them
- There are generally two routes to finding work you love. The organic route: as you become more eminent, gradually to increase the parts of your job that you like at the expense of those you don't. The two-job route: to work at things you don't like to get money to work on things you do. The organic route is more common. It happens naturally to anyone who does good work. The disadvantage of this route is that it's slow and uncertain. Even tenure

is not real freedom. The two-job route is less common than the organic route, because it requires deliberate choice. It's also more dangerous. The advantage of the two-job route is that it lets you jump over obstacles. If you make money doing one thing and then work on another, you have more freedom of choice. Which route should you take? That depends on how sure you are of what you want to do, how good you are at taking orders, how much risk you can stand, and the odds that anyone will pay (in your lifetime) for what you want to do

- In the design of lives, as in the design of most things, you get better results if you use flexible media. So unless you're fairly sure what you want to do, your best bet may be to choose a type of work that could turn into either organic or two-job career. That was probably part of the reason I chose computers. You can be a professor, or make a lot of money, or morph it into any number of other kinds of work.
- Constraints give your life shape. Remove them and most people have no idea what to do: look at what happens to those who win lotteries or inherit money. Much as everyone things they want financial security, the happiest people are not those who have it, but those who like what they do. So a plan that promises freedom at the expense of knowing what to do with it may not be as good as it seems
    - *The first condition of freedom is its limitation. – Will Durant*

Why YC?

- The real reason we started Y Combinator is neither selfish nor virtuous. We didn't start it mainly to make money; we have no idea what our average returns might be, and won't know for years. Nor did we start YC mainly to help out young would-be founders, though we do like the idea, and comfort ourselves occasionally with the thought that if all our investments tank, we will thus have been doing something unselfish (it's oddly nondeterministic.) The real reason we started Y Combinator is one probably only a hacker would understand. We did it because it seems such a great hack. There are thousands of smart people who could start companies and don't, and with a relatively small amount of force applied at just the right place, we can spring on the world a stream of new startups that might otherwise not have existed. In a way this is virtuous, because I think startups are a good thing. But really what motivates us is the completely amoral desire that would motivate any hacker who looked at some complex device and realized that with a tiny tweak he could make it run more efficiently. In this case, the device is the world's economy, which fortunately happens to be open source.

6,631,372

- Graham acquired a patent for search by calculating bid times the number of transactions. Wasn't a big deal in 1998 when he was first working on Revenue Loop as advertisers were way overpaying but later on this became an important technology

Are Software Patents Evil?

- Since software patents are no different from hardware patents, people who say "software patents are evil" are saying simply "patents are evil."
- To be patentable, an invention has to be more than new. It also has to be non-obvious. The problem with Amazon's one-click patent, for example, is not that it's a software patent, but that it's obvious
- We tell startups we fund not to worry about infringing patents, because startups rarely get sued for patent infringement. There are only two reasons someone might sue you: for money, or to prevent you from competing with them. Startups are too poor to be worth suing for money. And in practice they don't seem to get sued much by competitors, either.
- When you read of big companies filing patent suits against smaller ones, it's usually a big company on the way down, grasping at straws. When you see a big company threatening patent suits, sell. When a company starts fighting over IP, it's a sign they've lost the real battle, for users.
- If a startup wants to get bought, they should apply for patents because patents are part of the mating dance with acquirers as acquirers care about patents. But even in the mating dance, patents are of secondary importance. It matters more to make something great and get a lot of users
- Why do patents play so small a role in software? I can think of three possible reasons. One is that software is so complicated that patents by themselves are not worth very much. Software is subtle and unpredictable that "qualified experts" don't get you very far. What protects little companies from being copied by bigger competitors is not just their patents, but the thousand little things the big company will get wrong if they try. The second reason patents don't count for much in our world is that startups rarely attack big companies head-on. In the software business, startups beat established companies by transcending them. Fortunately for startups, big companies are extremely good at denial. If you take the trouble to attack them from an oblique angle, they'll meet you half-way and maneuver to keep you in their blind spot. The third reason patents don't seem to matter very much in software is public opinion – or rather, hacker opinion. Good hackers care a lot about matters of principle, and they are highly mobile. If a company starts misbehaving, smart people won't work there. For some reason this seems to be more true in software than other businesses. Google's "don't be evil" policy may for this reason be the most valuable thing they've discovered. It's very constraining in some ways. If Google does something evil, they get doubly whacked for it: once for whatever they did, and again for hypocrisy. But I think it's worth it. It helps them to hire the best people, and its better, even from a purely selfish point of view, to be constrained by principles than by stupidity.
    - *Be constrained by principles rather than by stupidity. So good and applicable to all areas of life*
- Things always seem intangible when you don't understand them. Electricity seemed an airy intangible to most people in 1800. Who knew there was so much to know about it? So it is with design.
- Patent trolls are companies consisting mainly of lawyers whose whole business is to accumulate patents and threaten to sue companies who actually make things. Patent trolls, it seems safe to say, are evil.

- Whether patents encourage innovation or not, they were at least intended to. Before patents, people protected ideas by keeping them secret. With patents, central governments said, in effect, if you tell everyone your idea, we'll protect it for you. Patents, like police, are involved in many abuses. But in both cases the default is something worse. The choice is not "patents or freedom?" any more than it is "police or freedom?" The actual questions respectively are "patents or secrecy?" and "police or gangs?" Even worse than the spectacular abuses might be the overall decrease in efficiency that would accompany increased secrecy

See Randomness

- Plato quotes Socrates as saying "the unexamined life is not worth living." A lot of ancient philosophy focused on answering the question, "what is our purpose?" Today we'd ask why we even suppose we have a "purpose" in life. We may be better adapted for some things than others; we may be happier doing things we're adapted for; but why assume purpose?
- The history of ideas is a history of gradually discarding the assumption that it's all about us. The idea that we're the center of things is difficult to discard. So difficult that there's probably room to discard more. Richard Dawkins made another step in that direction only in the last several decades, with the idea of the selfish gene. No, it turns out, we're not even the protagonists: we're just the latest model vehicle our genes have constructed to travel around in. And having kids is our genes heading for the lifeboats. Reading that book snapped my brain out of its previous way of thinking the way Darwin's must have when it first appeared.
- If you want to discover things that have been overlooked till now, one really good place to look is in our blind spots: in our natural, naïve belief that it's all about us. And expect to encounter ferocious opposition if you do. Conversely, if you have to choose between two theories, prefer the one that doesn't center on you. Don't take things personally. Its more general version is our answer to the Greeks: Don't see purpose where there isn't. Or better still, the positive version: see randomness
    - *Worth repeating – If you want to discover things that have been overlooked till now, one really good place to look is in our blind spots: in our natural, naïve belief that it's all about us. If you have to choose between two theories, prefer the one that doesn't center on you. Don't take things personally. Its more general version is our answer to the Greeks: Don't see purpose where there isn't. Or better still, the positive version: see randomness*

The Hardest Lesson for Startups to Learn

- The counterintuitive lessons I've learned are the ones that I have to keep repeating
    - Release early
        - Get a version 1 out fast and then improve it based on users' reactions; not something full of bugs, but something minimal.

        - No web startup knows their users and it's dangerous to guess what they'll like. Better to release something and let them tell you
        - Perhaps the most important reason to release early, though, is that it makes you work harder. When you're working on something that isn't released, problems are intriguing. In something that's out there, problems are alarming. There is a lot more urgency once you release. And I think that's precisely why people put it off. They know they'll have to work a lot harder once they do
        - The rate of improvement is more important to users than where you currently are
    - Keep pumping out features
        - By "feature" I mean one unit of hacking – one quantum of making users' lives better, not more complex
        - As with exercise, improvements beget improvements. So it is with hacking: the more ideas you implement the more ideas you'll have. You should make your system better at least in some way every day or two
        - This is not just a good way to get development done; it is also a form of marketing. Users love a site that's constantly improving
        - They'll like you even better when you improve in response to their comments, because customers are used to companies ignoring them
        - Assume that anything you've made is far short of what it could be. Force yourself, as a sort of intellectual exercise, to keep thinking of improvements. If your product seems finished, there are two possible explanations: it is finished, or you lack imagination. Experience suggests option two is a thousand times more likely
    - Make users happy
        - Improving constantly is an instance of a more general rule: make users happy
        - There are two things you have to do to make people pause. The most important is to explain, as concisely as possible, what the hell your site is about. How often have you visited a site that seemed to assume you already knew what they did? You probably shouldn't even start a company to do something that can't be described compellingly in one or two sentences. The other thing I repeat is to give people everything you've got, right away. If you have something impressive, try to put it on the front page, because that's the only one most visitors will see. Though indeed there's a paradox here: the more you push the good stuff toward the front, the more likely visitors are to explore further. In the best case these two suggestions get combined: you tell visitors what your site is about by showing them. Nothing will explain what your site does so well as using it
    - Fear the right things
        - Another thing I find myself saying a lot is "don't worry." Actually, it's more often "don't worry about this; worry about this instead." Startups are right to be paranoid, but they sometimes fear the wrong things
        - What you should fear, as a startup, is not the established players, but other startups you don't know exist yet. They're way more dangerous than

Google because, like you, they're cornered animals. Looking just at existing competitors can give you a false sense of security. You should compete against what someone else could be doing, not just what you can see people doing. A corollary is that you shouldn't relax just because you have no visible competitors yet. No matter what your idea, there's someone else out there working on the same thing

- There are a lot of ways for startups to hose themselves but the three main ones are internal disputes, inertia and ignoring users. Each is, by itself, enough to kill you. But if I had to pick the worst, it would be ignoring users.
- Almost everyone's initial plan is broken. If companies stuck to their initial plans, Microsoft would be selling programming languages, and Apple would be selling printed circuit boards. In both cases their customers told them what their business should be – and they were smart enough to listen
    - *Every counterparty will tell you how to operate and be successful, if you have a win/win mindset and are willing to listen*
- As Richard Feynman said, the imagination of nature is greater than the imagination of man. You'll find more interesting things by looking at the world than you could ever produce just by thinking. This principle is very powerful. It's why the best abstract painting still falls short of Leonardo, for example. And it applies to startups too. No idea for a product could ever be so clever as the ones you can discover by smashing a beam of prototypes into a beam of users
    - *Men of learning are those who have read the contents of books. Thinkers, geniuses, and those who have enlightened the world and furthered the race of men, are those who have made direct use of the book of the world. – Arthur Schopenhauer*

- Commitment is a self-fulfilling prophecy
    - I now have enough experience with startups to be able to say what the most important quality is in a startup founder, and it's not what you might think. The most important quality in a startup founder is determination. Not intelligence, determination.
    - The part that really demands determination is the backdrop of disaster always happening somewhere. So if you're the least bit inclined to find an excuse to quit, there's always one right there. But if you lack commitment, chances are it will have been hurting you long before you actually quit. If you lack commitment, it will seem to you that you're unlucky. Whereas if you're determined to stick around, people will pay attention to you, because odds are they'll have to deal with you later. You're a local, not just a tourist, so everyone has to come to terms with you. Both acquirers and investors judge you by your level of commitment. And you can't fake this. The only way to convince everyone that you're ready to fight to the death is actually to be ready to. You have to be the right kind of determined, though. I carefully chose the word determined rather than stubborn, because stubbornness is a disastrous quality in a startup. You have to be determined, but flexible, like a running back. A successful running back doesn't just put his head down and try to run through people. He improvises: if someone

appears in front of him, he runs around them; if someone tries to grab him, he spins out of their grip; he'll even run in the wrong direction briefly if that will help. The one thing he'll never do is stand still

        - *So true – "Both acquirers and investors judge you by your level of commitment. And you can't fake this. The only way to convince everyone that you're ready to fight to the death is actually to be ready to." This is a great filter and test for yourself, your ideas, your competitors, your employees…*

    - What really motivates investors, even big VCs, is not the hope of good returns, but the fear of missing out

- There is always room
    - There is always room for new stuff. At every point in history, even the darkest bits of the dark ages, people were discovering things that made everyone say "why didn't anyone think of that before?" the reason we don't see the opportunities all around us is that we adjust to however things are, and assume that's how things have to be
    - In particular, I don't think there's any limit to the number of startups. Startups make wealth, which means they make things people want, and if there's a limit on the number of things people want, we are nowhere near it.
- Don't get your hopes up
    - Startup founders are naturally optimistic. They wouldn't do it otherwise. But you should treat your optimism the way you'd treat the core of a nuclear reactor: as a source of power that's also very dangerous. You have to build a shield around it, or it will fry you. I think the place to draw the line is between what you expect of yourself, and what you expect of other people. It's ok to be optimistic about what you can do, but assume the worst about machines and other people
        - *This "dual optimism" is correct. You can push yourself and expect the most but it will only lead to frustration and ruined relationships if you expect that or try to push it on others. If they voluntarily do it, great, but otherwise have low expectations and make no assumptions*
    - Shielding your optimism is nowhere more important than with deals. If your startup is doing a deal, just assume it's not going to happen
    - The reason I warn startups not to get their hopes up is not to save them from being disappointed when things fall through. It's for a more practical reason: to prevent them from leaning their company against something that's going to fall over, taking them with it. Just run your company as if this deal didn't exist. Nothing is more likely to make it close. The way to succeed in a startup is to focus on the goal of getting lots of users, and keep walking swiftly toward it while investors and acquirers scurry alongside trying to wave money in your face.
    - VCs and corp dev guys are professional negotiators so don't even try to bluff them. The only way a startup can have any leverage in a deal is genuinely not to need it
- Speed, not money

        - There's nothing particularly grand about making money. That's not what makes startups worth the trouble. What's important about startups is the speed. By compressing the dull but necessary task of making a living into the smallest possible time, you show respect for life, and there is something grand about that.
            - *Nature seems "biased" towards velocity as* $K_e = \frac{1}{2}*mass*velocity^2$

How to Be Silicon Valley

- What would it take to recreate Silicon Valley? The right people. If you could get the right ten thousand people to move from Silicon Valley to Buffalo, Buffalo would become Silicon Valley. That's a striking departure from the past. Up till a couple decades ago, geography was destiny for cities. All great cities were located on waterways, because cities made money by trade, and water was the only economical way to ship. Now you could make a great city anywhere, if you could get the right people to move there. So the question of how to make a silicon valley becomes: who are the right people, and how do you get them to move?
- I think you only need two kinds of people to create a technology hub: rich people and nerds. So, you could reproduce Silicon Valley by making life good for both rich people and nerds in a new city. So what makes a place good to them? What nerds like is other nerds. Smart people will go where other smart people are. And in particular, to great universities. Creating one of the world's best universities may not be as hard as it seems. My professor friends, when they're deciding where they'd like to work, consider one thing above all: the quality of the other faculty. The university has to be in a town that has attractions other than the university. It has to be a place where investors want to live, and students want to stay after they graduate. They want to live somewhere with personality – well-preserved old neighborhoods, local shops, locally-owned restaurants. Most cities with personality are old but they don't have to be. Just have building codes that ensure density and ban large scale developments
- You don't build a silicon valley, you let one grow
- What you can't have, if you want to create a silicon valley, is a large, existing population of stodgy people. A place that tolerates oddness in the search for the new is exactly what you want in a startup hub, because economically that's what startups are. Most good startup ideas seem a little crazy; if they were obviously good ideas, someone would have done them already. That's the connection between technology and liberalism. But it's not because liberals are smarter that this is so. It's because liberal cities tolerate odd ideas, and smart people by definition have odd ideas
- Within the US, the two cities I think could most easily be turned into new silicon valleys are Boulder and Portland. Both have the kind of effervescent feel that attracts the young. They're each only a great university short of becoming a silicon valley, if they wanted to
- The companies that rule Silicon Valley now are all descended in various ways from Shockley Semiconductor. Startups beget startups. People who work for startups start their own. People who get rich from startups fund new ones. I suspect this kind of organic growth is the only way to produce a startup hub, because it's the only way to grow the expertise you need. That has two important implications. The first is that you need time to grow a

silicon valley. The university you could create in a couple years, but the startup community around it has to grow organically. The cycle time is limited by the time it takes a company to succeed, which probably averages about five years. The other implication of the organic growth hypothesis is that you can't be somewhat of a startup hub. You either have a self-sustaining reaction, or not. Of course, a would-be Silicon Valley faces an obstacle the original one didn't: it has to compete with Silicon Valley. Can that be done? Probably. A town that could exert enough pull over the right people could resist perhaps even surpass Silicon Valley

  - *What an incredibly fun, ambitious, potentially world-changing project this could be. If pulled off, it would have incredible ripple effects for everyone involved*

Why Startups Condense in America

- I've claimed that the recipe is a great university near a town smart people like. If you set up these conditions within the US, startups will form as inevitably as water droplets condense on a cold piece of metal. But when I consider what it would take to reproduce Silicon Valley in another country, it's clear the US is a particularly humid environment. Startups condense more easily here. It is by no means a lost cause to try to create a silicon valley in another country. There's room not merely to equal Silicon Valley, but to surpass it. But if you want to do that, you have to understand the advantages startups get from being in America: the US allows immigration, the US is a rich country, the US is not (yet) a police state, American universities are better, you can fire people in America, in America work is less identified with employment, America is not too fussy, America has a large domestic market, America has venture funding, America has dynamic typing for careers
- Imaginative people don't want to follow or lead. They're most productive when everyone gets to do what they want.
- Across industries and countries, there's a strong inverse correlation between performance and job security
  - *Creating a calm, safe, secure, stable environment allows people to go all-in as they aren't worried about their paycheck, getting fired, politicking or anything else. Their energy can actually be spent where it is supposed to – doing great work*
- Here's a tip for foreign governments that want to encourage startups: read the stories of existing startups, and then try to stimulate what would have happened in your country. When you have something that would have killed Apple, prune it off
- Incidentally, America's private universities are one reason there's so much venture capital. A lot of the money in VC funds come from their endowments. So another advantage of private universities is that a good chunk of the country's wealth is managed by enlightened investors
- Americans are said to be more entrepreneurial and less afraid of risk. But America has no monopoly on this. Indians and Chinese seem plenty entrepreneurial, perhaps more than Americans. Some say Europeans are less energetic, but I don't believe it. I think the problem with Europe is not that they lack balls, but that they lack examples
- To beat America, you can design a town that puts cars last as people in Silicon Valley like to get around by train, bicycle and on foot. It will be a while before any American city can bring itself to do that. There are also a couple things you could do to beat America at the

national level. One would be to have lower capital gains taxes. The other place you could beat the US would be with smarter immigration policy. There are huge gains to be made here. Silicon valleys are made of people, remember. It would be easy to do better. Imagine if, instead, you treated immigration like recruiting – if you made a conscious effort to seek out the smartest people and get them to come to your country

The Power of the Marginal

- One of California's hidden advantages is its weather: the mild climate means there's lots of marginal space. In cold places that margin gets trimmed off. There's a sharper line between outside and inside, and only projects that are officially sanctioned – by organizations, or parents, or wives, or at least by yourself – get proper indoor space. That raises the activation energy for new ideas. You can't just tinker. You have to justify
- Apple already had something few real companies ever have: a fabulously well designed product. You'd think they'd have had more confidence. But I've talked to a lot of startup founders, and it's always this way. They've built something that's going to change the world, and they're worried about some nit like not having proper business cards
- That's the paradox I want to explore: great new things often come from the margins, and yet the people who discover them are looked down on by everyone, including themselves. It's an old idea that new things come from the margins. I want to examine its internal structure. Why do great ideas come from the margins? What kind of ideas? And is there anything we can do to encourage the process?
    - *Where there is mystery, there is margin. Spaces which are seen as "uncivilized," "taboo," or "inappropriate" are prime areas to explore*
- One reason so many good ideas come from the margin is simply that there's so much of it
- A few of the disadvantages of insider projects: the selection of the wrong kind of people, the excessive scope, the inability to take risks, the need to seem serious, the weight of expectations, the power of vested interests, the undiscerning audience, and perhaps most dangerous, the tendency of such work to become a duty rather than a pleasure
- A world with outsiders and insiders implies some kind of test for distinguishing between them. And the trouble with most tests for selecting elites is that there are two ways to pass them: to be good at what they try to measure, and to be good at hacking the test itself. So the first question to ask about a field is how honest its tests are, because this tells you what it means to be an outsider. This tells you how much to trust your instincts when you disagree with authorities, whether it's worth going through the usual channels to become one yourself, and perhaps whether you want to work in this field at all. Tests are least hackable when there are consistent standards for quality, and the people running the tests really care about its integrity. One way to tell whether a field has consistent standards is the overlap between the leading practitioners and the people who teach the subject in universities. Don't learn things from teachers who are bad at them.
- Where the method of selecting the elite is thoroughly corrupt, most of the good people will be outsiders. In art, for example, the image of the poor, misunderstood genius is not just one possible image of a great artist: it's the standard image. If a test is corrupt enough, a test becomes an anti-test, filtering out the people it should select by making them do things only the wrong people would do. Popularity in high school seems to be such a test. I think

that's one big reason big companies are so often blindsided by startups. People at big companies don't realize the extent to which they live in an environment that is one large, ongoing test for the wrong qualities.
    - *Hugely important and getting a fresh perspective by bringing in outsiders is a great way to find out if this is a blind spot relevant to you*
- Even in a field with honest tests, there are still advantages to being an outsider. The most obvious is that outsiders have nothing to lose. They can do risky things, and if they fail, so what? Few will even notice. The eminent, on the other hand, are weighed down by their eminence. Eminence is like a suit: it impresses the wrong people, and it constrains the wearer. Outsiders should realize the advantage they have here. Being able to take risks is hugely valuable. Everyone values safety too much, both the obscure and the eminent. No one wants to look like a fool. But it's very useful to be able to. If most of your ideas aren't stupid, you're probably being too conservative. You're not bracketing the problem. The more complicated the world gets, the more valuable it is to be willing to look like a fool
- Lord Acton said you should judge talent at its best and character at its worst
- So, the best way to understand the advantages of being an outsider may be to look at the disadvantages of being an insider
- The eminent feel like everyone wants to take a bite out of them. The problem is so widespread that people pretending to be eminent do it by pretending to be overstretched. The lives of the eminent become scheduled, and that's not good for thinking. One of the great advantages of being an outsider is long, uninterrupted blocks of time. Obscurity is like health food – unpleasant, perhaps, but good for you. Whereas fame tends to be like the alcohol produced by fermentation. When it reaches a certain concentration, it kills off the yeast that produced it.
- In principle you could make any mark in any medium; in practice the medium steers you. And if you're no longer doing the work yourself, you stop learning from this. So if you want to beat those eminent enough to delegate, one way to do it is to take advantage of direct contact with the medium. In the arts it's obvious how: blow your own glass, edit your own films, and stage your own plays. And in the process pay close attention to accidents and to new ideas you have on the fly. This technique can be generalized to any sort of work: if you're an outsider, don't be ruled by plans. Planning is often just a weakness forced on those who delegate.
    - *The importance of staying engaged, of touching the medium. Deep fluency cannot come without this and it can disappear quite quickly when starved of it*
- Is there a general rule for finding problems best solved in one head? Well, you can manufacture them by taking any project usually done by multiple people and trying to do it all yourself. Wozniak's work was a classic example: he did everything himself, hardware and software, and the result was miraculous. He claims not one bug was ever found in the Apple II, in either hardware or software. Another good way to find problems to solve in one head is to focus on the grooves in the chocolate bar – the places where tasks are divided when they're split between several people. If you want to beat delegation, focus on a vertical slice: for example, be both writer and editor, or both design buildings and construct them. One especially good groove to span is the one between tools and things made with them.
- Much of the skill of experts is the ability to ignore false trails. But focus has drawbacks: you don't learn from other fields, and when a new approach arrives, you may be the last

one to notice. For outsiders this translates into two ways to win. One is to work on a variety of things. Since you can't derive as much benefit (yet) from a narrow focus, you may as well cast a wider net and derive what you can from similarities between fields. Just as you can compete with delegation by working on larger vertical slices, you can compete with specialization by working on larger horizontal slices – by both writing and illustrating your book, for example. The second way to compete with focus is to see what focus overlooks. In particular, new things. So if you're not good at anything yet, consider working on something so new that no one else is either. It won't have any prestige yet, if no one is good at it, but you'll have it all to yourself. The potential of a new medium is usually underestimated, precisely because no one has yet explored its possibilities. So in the future when you hear people say of a new platform: yeah, it's popular and cheap, but not ready yet for real work, jump on it
    - *Worth re-reading several times*
- The Achilles heel of successful companies is their inability to cannibalize themselves. Many innovations consist of replacing something with a cheaper alternative, and companies just don't want to see a path whose immediate effect is to cut an existing source of revenue. So if you're an outsider you should actively seek out contrarian projects. Instead of working on things the eminent have made prestigious, work on things that could steal that prestige. The really juicy new approaches are not the ones insiders reject as impossible, but those they ignore as undignified
- Both cheap and lightweight are good bets for growth: cheap things spread faster, and lightweight things evolve faster. Outsiders are free from all the pressures of working on big things. They can work on small things, and there's something very pleasing about small things. Small things can be perfect; big ones always have something wrong with them. But there's a magic in small things that goes beyond such rational explanations. All kids know it. Small things have more personality. Working on small things is also a good way to learn. The most important kinds of learning happen one project at a time. ("Next time I won't…") The faster you cycle through projects, the faster you'll evolve. So if you're an outsider, take advantage of your ability to make small and inexpensive things. Cultivate the pleasure and simplicity of that kind of work; one day you'll miss it
- What the old and eminent seem to miss the most is the lack of responsibilities. Responsibility is an occupational disease of eminence. How does responsibility constrain you? The worst thing is that it allows you not to focus on real work. Just as the most dangerous forms of procrastination are those that seem like work, the danger of responsibilities is not just that they can consume a whole day, but that they can do it without setting off the kind of alarms you'd set off if you spent a whole day sitting on a park bench.
- For insiders, work turns into a duty, laden with responsibilities and expectations. It's never so pure as it was when they were young. Work like a dog being taken for a walk, instead of an ox being yoked to the plow. That's what they miss
- A lot of outsiders make the mistake of doing the opposite; they admire the eminent so much that they copy even their flaws. Copying is a good way to learn but copy the right things. Half the distinguishing qualities of the eminent are actually disadvantages. Imitating these is not only a waste of time, but will make you seem a fool to your models, who are often well aware of it
    - *Take the "juice" – the habits, traits, characteristics, values, etc. – worth emulating and discard the "peel" – the flaws found in even the best exemplars*

- What are the genuine advantages of an insider? The greatest is an audience. If this is correct, we live in very exciting times. This is great news for the marginal, who retain the advantages of outsiders while increasingly being able to siphon off what had till recently been the prerogative of the elite. The big media companies shouldn't worry that people will post their copyrighted material on YouTube. They should worry that people will post their own stuff on YouTube, and audiences will watch that instead
- If I had to condense the power of the marginal into one sentence it would be: just try hacking something together. Hacking something together means deciding what to do as you're doing it, not a subordinate executing the vision of his boss. It implies the result won't be pretty, because it will be made quickly out of inadequate materials. It may work, but it won't be the sort of thing the eminent would want to put their name on. Something hacked together means something that barely solves the problem, or maybe doesn't solve the problem at all, but another you discovered en route. But that's ok, because the main value of that initial version is not of the thing itself, but what it leads to. Insiders who daren't walk through the mud in their nice clothes will never make it to the solid ground on the other side
- If I could go back and redo my twenties, that would be one thing I'd do more of: just try hacking things together. Like many people that age, I spent a lot of time worrying about what I should do. I also spent some time trying to build stuff. I should have spent less time worrying and more time building. If you're not sure what to do, make something. In life, as in books, action is underrated
    - *A bias for action seems to be a central tenet of the most successful*
- If you really want to score big, the place to focus is the margin of the margin: the territories only recently captured from the insiders. That's where you'll find the juiciest projects still undone, either because they seemed too risky, or simply because there were too few insiders to explore everything
- This leads to my final suggestion: a technique for determining when you're on the right track. You're on the right track when people complain that you're unqualified, or that you've done something inappropriate. If people are complaining, that means you're doing something rather than sitting around, which is the first step. And if they're driven to such empty forms of complaint, that means you've probably done something good. If you make something and people complain that it doesn't work, that's a problem. But if the worst thing they can hit you with is your own status as an outsider, that implies that in every other respect you've succeeded. Pointing out that someone is unqualified is as desperate as resorting to racial slurs. It's just a legitimate-sounding way of saying: we don't like your type around here. So that, I think, should be the highest goal for the marginal. Be inappropriate. When you hear people saying that, you're golden. And they, incidentally, are busted.

The Island Test

- What you imagine you would pack for a deserted island is what you're addicted to

Copy What You Like

- How do you avoid copying the wrong things? Copy only what you like. It can be hard to separate the things you like from the things you're impressed with. One trick is to ignore presentation.
- Another way to figure out what you like is to look at what you enjoy as guilty pleasures
- Even when you find genuinely good things to copy, there's another pitfall to be avoided. Be careful to copy what makes them good, rather than their flaws. It's easy to be drawn into imitating flaws because they're easier to see, and of course easier to copy too.
- You have to figure out for yourself what is good. You can't trust authorities

How to Present to Investors

- Startups are a counter example to the rule that haste makes waste. Too much money seems to be as bad for startups as too much time
- At first we give them just two goals: explain what you're doing and why users will want it
- Tips: explain what you're doing, get rapidly to the demo (start with the problem you're solving and then show how your product solves it), better a narrow description than a vague one, don't have the same person talk and drive the presentation, don't talk about secondary matters at length, don't get too quickly into business models, talk slowly and clearly at the audience, have one person talk, seem confident (show, not tell), don't try to seem more than you are, don't put too many words on slides, specific numbers are good, tell stories about the users (proxy for demand – what are people doing inadequately now that shows they need your product), make a sound bite stick in their heads
- Good writing is an elaborate effort to seem spontaneous

2007

A Student's Guide to Startups

- The problem with starting a startup while you're still in school is that you have a built in escape hatch
- For nearly everyone, the opinion of one's peers is the most powerful motivator or all - more powerful even than the nominal goal of most founders, getting rich. Consider this force like a wind and set up your boat accordingly. If you know your peers are going to push you in some direction, choose good peers, and position yourself so they push you in a direction you like
- As a young founder your strengths are: stamina, poverty, rootlessness, colleagues and ignorance
- The way successful startups find something that works is by trying things that don't. The worst thing you can do is have a rigid, pre-ordained plan and then start spending a lot of money to implement it
- Even more important than living cheaply is thinking cheaply
- The students who become successful startup founders are those who are smart and incurable builders. People who keep starting projects and finish at least some of them
- You need a certain activation energy to start a startup. So an employer who's fairly pleasant to work for can lull you into staying indefinitely, even if it would be a net win for you to leave
- Ignorance is a powerful force for young founders. Starting a startup is harder than you expect, but you're also capable of more than you expect, so they balance out
- Once you're past a basic level of intelligence, the deciding factor is how much you want to succeed
- Choosing something boring or unsexy can compensate for intelligence. So, no matter how far you think you fall short of Larry and Sergey, you can ratchet down the coolness of the idea far enough down to compensate
- When you know nothing, you have to reinvent stuff for yourself and if you're smart your reinventions may be better than what preceded them. This is especially true for fields where the rules change
- If you're free of a misconception that everybody else still shares, you're in a powerful position
- One of the most distinctive differences between school and the real world is that there is no reward for putting in a good effort. In fact, the whole concept of a "good effort" is a fake idea adults invented to encourage kids. It is not found in nature. Such lies seem helpful to kids. But unfortunately when you graduate they don't give you a list of all the lies they told you during your education. You have to get them beaten out of you by contact with the real world
- "Work experience" is not really some sort of expertise but the elimination of certain habits left over from childhood
- To someone who has learned from experience about the relationship between money and work, it translates to something way more important than Ferraris or prestige: it means you

get to opt out of the brutal equation that governs the lives of 99.99% of people. Getting rich means you can stop treading water
- You don't get money just for working, but for doing things other people want. Someone who's figured this out will automatically focus more on the user
- Be skeptical about classes and books about startups. The way to learn about startups is by watching them in action, preferably by working at one

The 18 Mistakes That Kill Startups

- It's easier to catch yourself doing something you shouldn't than always trying to remember to do something you should
- In a sense there's only one mistake that kills startups: not making something users want
- Single founder - letting one's friends down is one of the most powerful forces in human nature and single founders lack this
- Bad location
- Marginal niche - often subconscious won't even let you think of grand ideas so brainstorm what great ideas would be for others to do as a startup
- Derivative idea - few great startups are imitators, they solve some deep problem the founders identified that affect them personally. What do you complain about? What do you wish there was? Ironically, a good idea might be a Facebook exclusively for college students
- Obstinacy - being rigid and sticking to your vision works in some areas but not in startups. You need to follow the trail wherever it leads. If you're thinking of going some direction and your users seem excited about it, it's probably a good bet
- Hiring bad programmers
- Choosing the wrong platform - if you don't know, visit the top CS departments and see what they're using in research projects
- Slowness in launching - launching will force you to finish some quantum of work and it is only by launching that you can fully understand your ideas by bouncing them off your users
- Launching too early - ruins your reputation if too soon. Need to identify a core that's both: useful on its own and something that can be incrementally expanded into the whole project, and then get that done as soon as possible
- Having no specific user in mind - you can't build something users like without understanding them. Perhaps there's a rule here: perhaps you create wealth in proportion to how well you understand the problem you're solving, and the problems you understand best are your own
- Raising too little money - you should take more than you think you'll need, maybe 50-100% more, because software takes longer to write and deals longer to close than you expect. Give yourself maximum flexibility by spending practically nothing and make your initial goals simply to build a solid prototype
- Spending too much - hiring is usually the cause so don't do it if you can avoid it, pay by equity rather than salary to get committed people and only hire people that are going to write code or go out and get users

- Raising too much money - when you raise a lot of money your company moves to the suburbs and has kids. Raising higher amounts of money also takes much more time
- Poor investor management - pissing off and ignoring investors is probably less dangerous than caving into them
- Sacrificing users to (supposed) profit - business models should come after figuring out how to make something users want, the companies that win are those that put users first
- Not wanting to get your hands dirty - there's not much of a market for ideas, nobody trusts an idea until you embody it in a product and use that to grow a user base. Hackers can't spend all their time hacking, they will have to spend at least some time doing business stuff
- Fights between founders - don't suppress misgivings and don't compromise on people
- A halfhearted effort

How Art Can Be Good

- Good art (like good anything else) is art that achieves its purpose particularly well
- Tastes are like a series of concentric rings, like ripples in a pond. There are some things that will appeal to you and your friends, others that will appeal to most your age, others that will appeal to most humans and perhaps others that would appeal to most sentient beings
    - *The "big picture" outer rings of life are often neglected because one is too far in the weeds, leading to an unbalanced life or suboptimal choices. Start from the outermost concentric ring and filter down to the core. If it passes all rings of health, family, friends, work, spiritual growth, community service and personal growth, you can "back up the truck" on that decision*
- If good art is one which interests its audience, then when you talk about art being good, you also have to say for what audience
- It's good art if it consistently affects humans in a certain way
- For the average person, brand dominates all other factors in judgment of art. You can wean yourself off this by seeing a famous painting over and over and by standing close
- The way to not be vulnerable to tricks is to explicitly seek out and catalog them. Through this effort you can make yourself nearly immune to tricks. The way to do this is to travel widely, in both time and space
- There is such a thing as good art. It's art that interests its human audience and since humans have a lot in common, what interests them is not random. Since there's such a thing as good art, there's also such a thing as good taste, which is the ability to recognize it. These people tend to be hard to trick and don't just like whatever they grew up with. Taste is therefore not wholly subjective
- It's not for the people who talk about art that I want to free the idea of good art, but for the people who make it. So, the most important consequence of realizing there can be good art is that it frees artists to try and make it

Learning From Founders

- The less energy people expend on performance, the more they expend on appearances to compensate. More often than not, the energy they expend on seeming impressive makes their actual performance worse. A few years ago I read an article in which a car magazine modified the "sports" model of some production car to get the fastest possible standing quarter mile. You know how they did it? They cut off all the crap the manufacturer had bolted onto the car to make it look fast
    - *Being aware of this trick will help you recognize it in others and hopefully help you avoid it in yourself*
- The most productive companies are early stage startups where the employees are badly dressed, in offices strewn with junk. But no visitor would understand that. Not even investors, who are supposed to be able to recognize real productivity when they see it
- The time will come when instead of startups trying to seem more corporate, corporations will try to seem more like startups.

Is It Worth Being Wise?

- What is Wisdom? I'd say it's knowing what to do in a lot of situations
- "Wise" and "smart" are both ways of saying someone knows what to do. The difference is that wise means one has a high average outcome across all situations and smart means one does spectacularly well in a few. That's how the two are related: they're the two different senses in which the same curve can be high. So a wise person knows what to do in most situations. While a smart person knows what to do in situations where few others could
- As we progress, our abilities are tested in an ever widening range of situations. Wisdom, learning and intelligence have thus been diverging over time. As knowledge gets more specialized, there are more points on the curve and the distinction between the spikes and the average becomes sharper, like a digital image rendered with more pixels
- There are reasons to believe that at some point you have to choose between wisdom and intelligence
- Human knowledge seems to grow fractally. Time after time, something that seemed a small and uninteresting area - experimental error, even - turns out, when examined up close, to have as much in it as all knowledge up to that point. Several of the fractal buds that have exploded since ancient times involve inventing and discovering new things. Math, for example, used to be something a handful of people did part time. Now it's the career of thousands
- To me it was just a relief just to realize it might be ok to be discontented as some of the old rules don't apply to new situations. The idea that a successful person should be happy has thousands of years of momentum behind it. If I was any good, why didn't I have the easy confidence winners are supposed to have? But that, I now believe, is like a runner asking "if I'm such a good athlete, why am I so tired?" Good runners still get tired; they just get tired at higher speeds. People whose work is to invent or discover things are in the same position as the runner. There's no way for them to do the best they can as there's no limit to what they could do. The closest you can come is to compare yourself to others but the better you do, the less this matters

- Intelligence has become increasingly important relative to wisdom because there is more room for spikes
- Wisdom seems to come largely from curing childish qualities and intelligence largely from cultivating them. The wise are much alike in their wisdom but very smart people tend to be smart in distinctive ways
- The path to wisdom is through discipline and the path to intelligence through carefully selected self-indulgence. Wisdom is universal and intelligence idiosyncratic. And while wisdom yields calmness, intelligence much of the time leads to discontentment. If you feel exhausted, it's not necessarily because there's something wrong with you. Maybe you're just running fast

Why to Not Not Start a Startup

- The big mystery to me is why don't more people start startups? If nearly everyone who does it prefers it to a regular job, and a significant percentage get rich, why doesn't everyone want to do this?
- The reasons why most people don't but should ignore: being too young, too inexperienced (even failure will get you to the ultimate goal faster than getting a job), not determined enough (are you sufficiently driven to be working on your own projects?), not smart enough (the decisive factor in most startups is effort, not brains), know nothing about business (make things users love, seek out ideas that would be popular but hard to profit from), no cofounder (having a committed cofounder is more important than anything else), no idea (founders far more important than the idea, find something that is missing in your own life and fill that need), no room for more startups, family to support (if you have a family, I wouldn't advise starting a startup), independently wealthy (what makes a startup founder so lethal is his willingness to endure infinite schleps), not ready for commitment, need for structure, fear of uncertainty (even big companies prefer to hire those who failed at a startup than those who spent time working at big companies), don't realize what you're avoiding (most regular jobs suck), parents want you to be a doctor (parents share more in children's ill fortune than good fortune which makes them conservative), a job is the default (defaults are so powerful precisely because they operate without any conscious choice)
- The most valuable truths are the ones most people don't believe

Microsoft is Dead

- Microsoft cast a shadow over the tech world for 20 years but nobody is afraid of them anymore, they're not dangerous
- 4 things killed them - google, death of desktops as apps moved to the web, broadband internet (further reducing need for desktops), and Apple
- Microsoft, in theory, could bounce back because of their enormous cash on hand and potentially very lucrative acquisitions. By buying all the good Web 2.0 startups and locking them far away from Redmond they could bounce back. Microsoft would never do this however as they don't yet realize how much they suck

Two Kinds of Judgment

- There are two different ways people judge you. Sometimes judging you correctly is the end goal such as in classes and competitions and sometimes judging is only a means to something else as in hiring and investment decisions. Our early training and our self-centeredness combine to make us believe that every judgment of us is about us. In fact, most aren't. Once you realize how little most people judging you care about judging you accurately, you won't take rejection so personally. And curiously enough, taking rejection less personally may help you get rejected less often
    - *Always try your best, don't take things personally, be impeccable with your word, don't make any assumptions. – Don Miguel Ruiz*

The Hacker's Guide to Investors

- The world of investors is a foreign one to most hackers and these are some of the more surprising things I've learned about investors
    - The investors are the limiting factor and are what make the startup hub
    - Angel investors are the most critical
    - Angels don't like publicity
    - Most investors, especially VCs, are not like founders
    - Most investors are momentum investors - often investing in stuff they don't truly understand
    - Most investors are looking for big hits
    - VCs want to invest in large amounts
    - Valuations are fiction - since valuations change on how much has been invested, it shows how far they are from reflecting any kind of value in the company
    - Investors look for founders like the current stars - what investors still don't get is how clueless and tentative great founders can seem at the very beginning
    - The contribution of investors tends to be underestimated - the best investors set themselves apart by giving the best advice. Founders tend to be overestimated as their support structure is not publicized
    - VCs are afraid of looking bad - they can't do anything that will look bad to the doofuses running pension funds
    - Being turned down by investors doesn't mean much
    - Investors are emotional
    - The negotiation never stops till the closing
    - Investors like to co-invest - as a rule, one of the top things VCs care about is what other VCs think
    - Investors collude
    - Large scale investors care about their portfolio, not any individual company
    - Investors have different risk profiles than founders
    - Investors vary greatly
    - Investors don't realize how much it costs to raise money from them - time sink, opportunity cost, not building the product. Opportunity for companies to form a

new type of VC fund where they invest smaller amounts at lower valuations but promises to close or say no very quickly
    - Investors don't like to say no - as it limits optionality
    - You need investors
    - Investors like it when you don't need them - always have a backup plan for getting started if a given investor says no. Having one is the best insurance against needing one

An Alternative Theory of Unions

- In a rapidly growing market you don't worry too much about efficiency. It's more important to grow fast
- Manufacturing in the 20th century was the fast growth industry and overpaid for infrastructure. Today, we simply overspend on different things

The Equity Equation

- The equation for taking money from investors and hiring is 1/(1-n) where n is % of the company. You give up n% if it improves your average outcome enough that the (100-n%) you have left is worth more than the whole company was before

Stuff

- Stuff has gotten a lot cheaper but our attitudes toward it haven't. We overvalue stuff
- Most of the stuff we have is worthless because we don't need it. What I didn't understand was that the value of some new acquisition wasn't the difference between its retail price and what I paid for it. It was the value I derived from it
- Stuff, worse than being worthless, eventually comes to own you
- Before buying anything ask, "is this going to make my life noticeably better?" and "will this be something I use constantly?"
- Everything you own takes energy away from you as it requires thought and planning. Some give more than they take. Those are the only things worth having
- I've now stopped accumulating stuff. Except for books if I want to spend my money on some kind of treat, I'll take services over goods any day
    - *We shape our buildings and thereafter they shape us. – Winston Churchill.*

Holding a Program in One's Head

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

    - *This applies to any project, problem or thought. If you can't hold it in your head and manipulate it at will, like a 3D image or hologram, to see the ripple effects of certain decisions, you don't understand it well enough*
- Things you can do to help load it into your head - avoid distractions, work in long stretches (the optimum is not the longest stretch you can physically endure), use succinct languages, keep rewriting your program, write rereadable code, work in small groups, don't have multiple people editing the same piece of code, start small
- It's amazing how often hackers hit all 8 points and how often officially sanctioned projects miss all of them
- The opportunity to chip away at large competitors is to find and stack the kinds of problems that have to be solved in one big brain

How Not to Die

- In most startups, if you can avoid dying you often get rich
- Most startups simply crawl off somewhere and die, there is no bang. Most die because they get demoralized
- Staying in touch with YC and other YC-funded companies may be a great start to avoid death
- Expect there to be great lows and to feel like things aren't working
- Make something that at least some small group really loves. This will give you morale and show you what to focus on. What is it about you that you love? Can you do more of it? Where can you find more people who love that sort of thing?
    - *You are your own startup. Apply this principle to how you operate, what you learn, what you do. Keep iterating, learning, improving, refining what you do, how you do it and whom you do it with*
- Distraction is fatal to startups. Don't do other things, like grad school
- People are more afraid of looking bad than the hope of getting millions of dollars. So, put yourself in a position where failure will be public and humiliating. Remove every excuse and be committed to the death

News From the Front

- A few weeks ago I had a thought so heretical that it really surprised me. It may not matter all that much where you go to college
- The test applies to a startup is among the purest of real world tests. A startup succeeds or fails depending almost entirely on the efforts of the founders. Success is decided by the market: you only succeed if users like what you've built. And users don't care where you went to college
- Graduates from elite colleges are great at doing what they're asked and they're confident. Both these traits make them fit right into big organizations and make them safe bets
- What we've found is that the variation between schools is so much smaller than the variation between individuals that it's negligible by comparison. We can learn more from someone in the first minute of talking to them than by knowing where they went to school

- You can't tell where a person went to college just by talking to them. How much you learn depends a lot more on you than the college
- The other students are the biggest advantage of going to an elite college; you learn more from them than the professors
- The tragedy of the situation is that by far the greatest liability of not having gone to the college you'd have liked is your own feeling that you're thereby lacking something
- What matters is what you make of yourself. Not what college you went to

How To Do Philosophy

- In high school I decided to study philosophy in college. I thought it would be a shortcut straight to Wisdom as I would be learning what was really what
- I learned that something went wrong with philosophy and I have some ideas on how we might fix it.
- Philosophy doesn't really have a subject matter like math or history or other majors
- A key lesson is that concepts we use in everyday life are fuzzy, and break down if pushed too hard. Even a concept as dear to us as "I"
- Everyday words are inherently imprecise and, unfortunately for philosophy, most debates are not merely directed but driven by confusions over words. This is a central fact of philosophy: words break if you push them too far
- Aristotle's goal was to find the most general of the general principles
- Math is the precise half of the most abstract ideas, and philosophy the imprecise half. It's probably inevitable that philosophy will suffer by comparison, because there's no lower bound to its precision. Bad math is merely boring, whereas bad philosophy is nonsense. And yet there are some good ideas in the imprecise half. The trend is clear: the more general the knowledge, the more admirable it is. But then he makes a mistake - possibly the most important mistake in the history of philosophy. He has noticed that theoretical knowledge is often acquired for its own sake, out of curiosity, rather than for any practical need. So he proposed there are two kinds of theoretical knowledge: some that's useful in practical matters and some that isn't. Since people interested in the latter are interested in it for its own sake, it must be more noble. So he sets as his goal in Metaphysics the exploration of knowledge that has no practical use. Which means no alarms go off when he takes on grand but vaguely understood questions and ends up getting lost in a sea of words. His mistake was to confuse motive and result. It's very valuable in practice to have a deep understanding of what you're doing; even if you're never called on to solve advanced problems, you can see shortcuts in the solution of simple ones, and your knowledge won't break down in edge cases, as it would if you were relying on formulas you didn't understand. Knowledge is power. That's what makes theoretical knowledge prestigious. So while ideas don't have to have immediate practical implications to be interesting, the kinds of things we find interesting will surprisingly often turn out to have practical implications. Aristotle's mistake made the exploration and dedication to vague and useless problems admirable
- I propose we try again, to discover the most general truths, but approach it from a different direction. The test of utility I propose is whether we cause people who read what we've written to do anything differently afterward. Knowing we have to give definite (if implicit) advice will keep us from straying beyond the resolution of the words we're using. These

seem to me what philosophy should look like: quite general observations that would cause someone who understood them to do something differently
- Civilization always seems old because it's always the oldest it has ever been. The only way to say whether something is really old or not is by looking at structural evidence, and structurally philosophy is young; it's still reeling from the unexpected breakdown of words

The Future of Web Startups

- Startups are undergoing the same transformation that technology does when it becomes cheaper. As they get cheaper to build, many more get built and as a result they can be used in new ways
- This trend will lead to lots of startups, standardization always follows when technology makes things dramatically cheaper (when you make things in large volumes, you tend to standardize everything that doesn't need to change), new attitude to acquisition by big companies as startups proliferate, riskier strategies are possible as initial investment can be much less, younger and nerdier founders will arise, startup hubs will persist and perhaps become even more important as face to face meetings will always be important, better judgment by investors and acquirers will be needed to pick winners, college will matter less as degrees will become increasingly less important for founders, there will be lots of competitors but it is not a zero sum game, technology will evolve faster as startups will implement ideas faster,
    - *Recommends companies to have a Chief Acquisition Officer in charge of acquiring startups as this becomes an increasingly important and common part of business*
- You don't beat the incumbents, you redefine the problem to make them irrelevant

Why to Move to a Startup Hub

- If another country wanted to establish a rival to Silicon Valley, the single best thing they could do might be create a special visa for startup founders. US immigration policy is one of Silicon Valley's biggest weaknesses
- The quality of the investors may be the main advantage of startup hubs
- The way you get big returns is not by trying to avoid losses, but by trying to ensure you get some of the big hits. And the big hits often look risky at first. Empirically, boldness wins. West Coast investors are confident enough of their judgment to act boldly
- Startup hubs are also markets and markets are usually centralized. It's hard to say exactly what it is about face to face contact that makes deals happen, but whatever it is, it hasn't yet been duplicated by technology

2008

Six Principles for Making New Things

- My m.o. is the same across all my projects – I like simple solutions, to overlooked problems, that actually need to be solved and deliver them as informally as possible, starting with a very crude version 1, then iterating rapidly
    - *Yes! – for everything*
- Like a contrarian investment fund, this recipe often garners a contemptuous initial reaction. This technique is successful in the long term because it gives you all the advantages other people forgo by trying to seem legit
- If you release a crude version 1 and then iterate, you can benefit from the imagination if nature, which, as Feynman pointed out, is more powerful than your own
- Great ideas are all around you but you ignore them because they look wrong

Trolls

- I think trolling in the broader sense has four causes. The most important is distance, anonymity. Second is the type of people who are attracted to computers, third cause of trolling is incompetence and lastly trolls are like children in that they test how much will be tolerated
- Graffiti happens at the intersection of ambition and incompetence: people want to make their mark on the world but have no other way to do it than literally making a mark on the world
- Anything becomes art if you do it well enough

A New Venture Animal

- When you scale animals you can't just keep everything in proportion. For example, volume grows as the cube of linear dimension but surface area only as the square. So as they get bigger, they have trouble radiating heat. That's why mice and rabbits are furry and elephants and hippos aren't. You can't get a mouse by scaling down an elephant. YC represents a new, smaller kind of animal – so much smaller that all the rules are different. All good investors supply a combination of money and help. But these scale differently, just as volume and scale do. Late stage investors supply huge amounts of capital and comparatively little help
- What we really do at YC is get startups launched straight. One of many metaphors you could use for YC is a steam catapult on an aircraft carrier. We get startups airborne. Barely airborne, but enough that they can accelerate fast
- If what we do is so useful, why wasn't anyone doing it before? There are two answers to them. One is that people were doing it before, just haphazardly on a smaller scale. Before us, seed funding came primarily from individual angel investors. But raising money from angels is hit or miss. The other reason no one was quite doing what we do is that till recently

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

  - *Being creative and finding ways to expand the pie rather than trying to play a zero-sum game makes the venture win/win and sustainable and in fact potentially even more rewarding*

- It's natural that the new niche would at first be described, even by its inhabitants, in terms of the old one. But really YC is not in the startup funding business. Really we are more of a small, furry catapult

You Weren't Meant to Have a Boss

- Technology tends to separate normal from natural. Our bodies weren't designed to eat the foods that people in rich countries eat, or to get so little exercise. There may be a similar problem with the way we work: a normal job may be as bad for us intellectually as white flour or sugar is for us physically. I suspect that working for oneself feels better to humans much in the same way that living in the wild must feel better to a wide-ranging predator like a lion. Life in a zoo is easier, but it isn't the life they were designed for
- Humans aren't meant to work in such large groups. Each species of animals thrived in groups of a certain size. For humans, groups of 8 works well, by 20 they're getting hard to manage and a group of 50 is really unwieldy. However, companies sometimes have hundreds or thousands of employees and to coordinate everything they introduced something new: bosses
  - *Must respect Dunbar's number. As human group sizes grow, the communication, cohesion and intimacy breakdown and the group and its performance suffers*
- In practice a group of people are never able to act as if they were one person. But in a large organization divided into groups this way, the pressure is always in that direction. A group of 10 people within a large organization is a type of fake tribe. The number of people you interact with is about right. But something is missing: individual initiative
- Food is an excellent metaphor to explain what's wrong with the usual sort of job. If "normal" food with white flour and refined sugar is so bad for us, why is it so common? There are two main reasons: it has immediate appeal though we feel bad an hour after eating that pizza and the other is economies of scale. Junk food scales but fresh vegetables doesn't. So, junk food can be very cheap and it's worth spending a lot to market it. If people have to choose between something that's cheap, heavily marketed and appealing in the short term, and something that's expensive, obscure, and appealing in the long term, which do you think most will choose? It's the same with work. Working at Google or Microsoft

is prestigious and safe but the drawbacks will only become apparent later, and then only in a vague sense of malaise
    - *Yes! This is where the opportunities to stand out and get decisive advantages lies. Being able to act on your "long-term gratification gene" where others can't offers enormous opportunity for contrast. Being aware and respecting that the further out the payoff the more difficult it will be for most people to do and hence the bigger the benefit for you doing it. The most valuable truths are the ones most people don't believe OR know but can't/don't act on*
- An obstacle downstream propagates upstream. If you're not allowed to implement new ideas, you stop having them. And vice versa: when you can do whatever you want, you have more ideas about what to do
- The only way I can see large organizations avoid the tree structure and slowing down as they grow would be to have no structure: to have each group actually be independent, and to work together the way components of a market economy do. Another way to avoid this is to stay small. This is especially important for technology companies and makes hiring that much more important.
    - *This is a fascinating idea worth exploring. Some companies might lean this way but would be eye-opening to see if any do this consciously and to the extreme*

How to Disagree

- Disagreement hierarchy – name calling, ad hominem responding to tone, contradiction, counter argument, refutation
- The most convincing form of disagreement is refutation. It's also the rarest because it's the most work. Indeed, the disagreement hierarchy forms a kind of pyramid, in the sense that the higher you go the fewer instances you find. You probably have to quote them and then refute what you think is mistaken. The most powerful form of disagreement is to refute someone's central point
    - *To truly have an opinion, you should know the other side's argument better than they do*
- A truly effective refutation would look like: the author's main point seems to be x. As he says...But this is wrong for the following reasons...
- The most obvious advantage of classifying the forms of disagreement is that it will help people to evaluate what they read and see through intellectually dishonest arguments. This may help writers too in avoiding unintentional intellectual dishonesty

Some Heroes

- My test was to think of someone and ask "is this person my hero?" It often returned surprising answers. When I thought about what it meant to call someone a hero, it meant I'd decide what to do by asking what they'd do in the same situation. That's a stricter standard than admiration
- Everyone on the list had two qualities: they cared almost excessively about their work and they were absolutely honest. By honest I meant that they never say or do something because

that's what the audience wants. They are all fundamentally subversive for this reason, though they conceal it to varying degrees

- Jack Lambert - defenseman on 1970s Steelers, he almost cared too much
- Kenneth Clark - best nonfiction writer I know of, on any subject. What really makes him stand out is the quality of his ideas
- Larry Mihalko - great teacher as he was intellectually curious and liked kids
- Leonardo - most of the best stuff isn't made for audiences but for oneself. One of his best qualities was that he did so many things that were admirable
- Robert Morris - he's never wrong. It might seem this would require you to be omniscient but actually it's surprisingly easy. Simply never say anything unless you're fairly sure of it. The trick is to pay careful attention to how you qualify what you say. It doesn't seem like that much extra work to pay as much attention to the error on an idea as to the idea itself but it is so hard
- PG Wodehouse - one of the great writers as he wrote exactly what he wanted and didn't care what others thought of him
- Alexander Calder - he's on this list because his art makes me happy. The point of art is to engage the viewer and his sculptures never get boring, they always radiate optimism. The happiness in his work seems to be his own happiness shining through
- Jane Austen - to me she seems the best novelist of all time. In her novels I can't see the gears at work, I can't see how she makes her choices. She's so good that her stories don't seem made up. I feel like I'm reading a description of something that actually happened
- John McCarthy - the inventor of Lisp and head of MIT's and Stanford's CS departments
- The Spitfire - a machine is my hero because it is optimism embodied, it was at the edge of what could be manufactured. Taking the high road worked. In the air, beauty had the edge, just
- Steve Jobs - many people are afraid of him at Apple which is a bad sign but he compels admiration. He has taste, which is rare in CEOs, and has shown the world how important taste is
- Isaac Newton - he worked on big things most of his life and is the one I reproach myself with. It is so easy to get distracted working on small things. This is the route to well-deserved obscurity. To do really great things, you have to seek out questions people didn't even realize we're questions. "Paradigm shift" is overused now but you know more are out there, separated from us by what will later seem a surprisingly thin wall of laziness and stupidity

Why There Aren't More Googles

- Many startups don't grow to become Google because they get bought too early. Why didn't that happen to Google? Because they had a deep sense of purpose: a conviction to change the world for the better. However, they also didn't sell out because they wanted more than acquirers were willing to pay
- Tip for acquirers: when a startup turns you down, consider raising your offer because there's a good chance the outrageous price they want will later seem like a bargain. Another tip: if you went to get all that value, don't destroy the startup after you buy it. Give the

founders enough autonomy that they can grow through acquisition into what it would have become

- Corporate M&A is a strange business in that respect. They consistently lose the best deals, because turning down reasonable offers is the most reliable test you could invent for whether a startup will make it big
- Money guys consistently undervalue the most innovative startups. The real reason there aren't more Googles is not that investors encourage innovative startups to sell out, but that they won't even fund them. The low cost of starting a startup means the average good bet is a riskier one, but most VCs are still operating as if they were investing in hardware startups in 1985
- “Don't worry about people stealing your ideas. If your ideas are any good, you'll have to ram them down people's throats.” – Howard Aiken
- The more people who have to like a new idea, the more outliers you lose
- The exciting thing about market economies is that stupidity equals opportunity. The gap between $20k seed funding and $2m VC funding is an increasingly valuable opportunity

Be Good

- Make something people want and don't worry too much about the business model. When you put these two ideas together you get something surprising. What you've got is the description of a charity. When you get an unexpected result like this, it could either be a bug or a new discovery. Either businesses aren't supposed to be like charities, and we've been proven by reductio ad absurdum that one or both of the principles we began with is false. Or we have a new idea. I suspect it's the latter, because as soon as this thought occurred to me, a whole bunch of other things fell into place. For example, Craigslist. It's not a charity but they run it like one. Craigslist is basically "upwind of potential profits." 50 years ago the market would be shocked by a public company to not pay dividends. Today most tech companies don't. Maybe the next step is for the market to get comfortable with potential earnings. VCs already are. A company with rapid, genuine growth is valuable and eventually markets learn how to value valuable things
- From either direction we get to the same spot. If you start from successful startups, they often behave like nonprofits. And if you start from ideas for nonprofits, you find they often make good startups
- Could you grow a successful startup out of curing an unfashionable but deadly disease like malaria? Maybe an organization that helped lift its weight off a country could benefit from the resulting growth.
    - *What a fascinating idea and way to align principles and incentives – helping to lift a country, city, region or whatever and getting a % of their future growth.*
- One way to guess how far an idea extends is to ask yourself at what point you'd bet against it. The thought of betting against benevolence is alarming in the same way that saying something is technically impossible is. You're just asking to be made a fool of because these are such powerful forces
    - *Benevolence can be a great mental model or filter for deciding which companies to fund, invest or work for.*

- When you're small you can't bully customers so you have to charm them. Whereas when you're big you can maltreat them at will, and you tend to, because it's easier than satisfying them. You grow big by being nice but you can stay big by being mean. You get away with it till the underlying conditions change, and then all your victims escape. Google's elixir of Don't be Evil is freely available to any company
    - *The only truly sustainable permutation to relationships is win/win*
- Being good seems to help startups in three ways: it improves their morale, it makes other people want to help them and, above all, it helps them be decisive, to have a compass during hard times. If you feel like you're helping people, you'll keep working even when it seems like your startup is doomed
- "The Tamogatchi Effect" – once you have users you care about and who rely on you, you're forced to figure out what will make them happy and that's actually very valuable information
- If you're benevolent, other people will rally behind you: investors, customers, other companies and potential employees. In the long term, the most important may be potential employees
- Startups have thousands of choices and decisions to make. How to choose? Do whatever is best for your users. This is particularly powerful because it is stateless. It's like telling the truth. The trouble with lying is that you have to remember everything you've said in the past to make sure you don't contradict yourself
    - *Finding these "invariant strategies" – ones which are optimal regardless of context, timing or circumstance – are so powerful*
- Don't just not be evil. Be good

Lies We Tell Kids

- By studying the ways adults lie to kids, we may be able to clear our heads of lies we were told
- The conspiracy is so broad and so thorough that most kids who discover it do so only by discovering internal contradictions in what they're told. It can be traumatic for those who wake up during the operation
- Parents lie to protect their kids. Misleading the child is just a byproduct of trying to conceal certain frightening things
- Many kids get the sex talk but few get the cocoon talk which explains how their fake world differs from the real world
- Parents lie about sex and drugs for obvious reasons but they also mislead in that it can also cause great pleasure. Parents want to instill confidence and great judgment but lie to kids because they believe their judgment to be lousy.
- Parents also lie to preserve innocence - sex, drugs, curse words. Words are interesting and causes discomfort because a jaded 10 year old who curses has cut off his room for growth so early – innocence is open mindedness. Very smart adults seem unusually innocent and I don't think this is a coincidence. They deliberately avoid learning about certain things
- Parents lie about death because it is the ultimate threat
- Parents lie about identity. Telling them they are an X and whatever specific lies Xes differentiate themselves by believing. Modern religions and doctrines are a combination of

the useful and the bizarre. The bizarre half is what makes the religion stick and the useful half is the payload

- One of the least excusable lies parents tell their kids is to maintain authority over their kids
- The most confident people are the most willing to say "I don't know"
- Kids get lied to in schools mostly to simplify ideas but too often propaganda gets slipped into this simplification. As subjects get softer, the lies get more frequent. Probably the biggest lie told in schools, though, is that the way to succeed is through following "the rules." In fact most such rules are just hacks to manage large groups effectively
- However, people most often lie to keep the peace, because the response to truth would be too violent. If you freak out when people tell you things, they won't tell them to you in the future
- Much like sprinters enter an "oxygen debt" in a race, most kids enter adulthood with a kind of truth debt. Nobody is going to tell you which were lies, this hard work has to be done by yourself. Few do. I've found that whenever I've been able to undo a lie I was told, a lot of other things fall into place. The first step in clearing your head is to realize how far you are from a neutral observer. It's not enough to consider your mind a blank slate. You have to consciously erase it

Disconnecting Distraction

- Procrastination feeds on distraction. Most people avoid work by doing something else. You beat procrastination by starving distraction but this is difficult as distraction seeks you out, they evolve with technology. They're often nefarious as they so closely resemble work
- The biggest ingredient in most bad habits is denial. When you slip, it has to set off alarms
- My solution to the internet addiction has been to set up a separate computer just for using the internet. I've slipped back but this was a useful experiment

Cities and Ambition

- Great cities attract ambitious people but each one seems to send a different message. NY tells you to be richer, Cambridge to be smarter and Silicon Valley to be more powerful. These forces are more powerful than people think. The people, conversations, businesses all affect you. It is hard to keep working on something when nobody around you cares about it. Peers and encouragement matter more than we think
- In most ambitious kids, ambition seems to precede anything specific to be ambitious about

The Pooled Risk Company Management

- The main economic motives of startup founders seem to be freedom and security
- If want to live off revenues of your company rather than selling but don't want to devote full time to it, hiring a company-management company which pooled your risk could work, theoretically. This does exist however, when you get acquired by a public company. The problem is, the acquirer doesn't think of themselves this way

A Fundraising Survival Guide

- Raising money is the second hardest part of starting a startup. The hardest part is making something people want
- Investors tend not to truly understand what they're investing in and this makes them very skittish
- Startups live and die on morale. Don't let fundraising kill you
- Many are bootstrapped or consulting companies at first and gradually transform themselves into product companies. Not as painful as raising money but it lasts for years
- To survive raising money you need a set of techniques mostly orthogonal to the ones used in convincing investors, just as mountain climbers need to know survival techniques that are mostly orthogonal to those used in physically getting up and down mountains
    - *The power of a broad, multidisciplinary education and focus. Everything you learn interlocks and feeds upon each other, allowing you to make connections and see things that others may not. A thousand little advantages may be more robust and powerful than one massive advantage*
- Have low expectations and realize raising money is much harder than you expect. What kills you is disappointment. One of our secondary mantras is deals fall through. Deals do not have a trajectory like most other human interactions, where shared plans solidify linearly over time. Deals often fall through at the last moment
- Keep working on your startup as you raise money. This may sound obvious but raising money often sucks up all your time. Often the best tactic is to pick one founder to deal with this as everyone else keeps working. Realize it always takes longer than you expect. Squeeze investor meetings in between building things rather than the other way around
- Be conservative - if someone offers you funding on reasonable terms, do it
- Be flexible - if asked how much you want to raise say there are several routes but raising more lets you get to the goal faster
- Be independent - investors flock to those who don't truly need them. Toughness, adaptability and determination are needed
- Don't take rejection personally. Handle rejection with precision. The odds are against you but always try to learn why they said no
- If raising isn't going well, downshifting to consulting, if appropriate, may be a good short term solution. Realize this never scales
- Avoid inexperienced investors
- Know where you stand, how likely you are to get funded. The most dangerous thing about investors is their indecisiveness. Convincing the hot investors is the best way to get the lukewarm ones on board
- YC's process is 20 minutes. So either existing investors will start to make up their minds faster or new investors will emerge who do.
- The biggest danger is surprise
- Oddly enough, the best VCs tend to be the least VC-like
    - *This isn't a bug, it's a feature. Paradox is where the opportunity lies*

Why to Start a Startup in a Bad Economy

- The state of the economy doesn't matter much for startups. If you create something people want, you'll be ok.
- Should always run the startup as cheaply as possible. Be the cockroaches of the corporate world - the hardest to kill. Recessions only make things cheaper so the number one cause of death, running out of money, is less likely
- There's less competition

The Other Half of "Artist's Ship"

- 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 but the real costs are the ones you never hear about. Every check should be analyzed for both its costs and benefits
- The cost of checks may actually be increasing too. As software plays an increasingly large role in companies, checks slow down programmers and their ideas. Programmers love working hard and it is my impression that they'd trade up to 50% of their acquisition price simply to be able to release software immediately
- If you don't let people ship, you won't have any artists working for you

The High-Res Society

- 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. Today 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
- 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 and then switches polarity? The millennia-long run 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 to discard. The place to look is where the spread of smallness began: in the world of startups
- Like science, wealth seems to expand fractally
- The most dynamic part of the economy always sets the tone for the rest. In everything from salaries to standards to dress. Not just because of prestige but because the principles underlying the most dynamic part of the economy tend to be the 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

Could VC be a Casualty of the Recession?

- Startups aren't as tied to VCs as they were 10 years ago. They're cheaper to start because: 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 means development teams can be smaller. As they've become so cheap to run, the threshold to profitability is so low and now many internet startups don't need VC scale investments like they used to. There's a sense of "investors aren't worth the trouble" and this could spread and this would be very bad for VCs
- 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

After Credentials

- What crams 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
- History suggests that, all 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 society get the best man for the job but the parents' ambitions are diverted from direct methods to indirect ones - to actually trying to raise their kids well
- Can solve the leaks by seeing where the holes are. You're succeeding in fixing them when things like cram schools, hacks to credentials, become less popular. A more general solution would be to push for increased transparency in the credentials.
- Instead of trying to make credentials harder to hack, we can also make them matter less. Credentials are a way to try to predict performance. If you could measure actual performance, you wouldn't need them
- In an economy run by a few large organizations, like the US in the 60s or South Korea today, credentials matter a lot
- A good test for economic competitiveness is seeing if young people get paid market rates for the work they do. All it takes is a few to break rank and pay for performance. Measurement spreads like heat
- The best way to block the transmission of power between generations is to encourage the trend toward an economy made of more, smaller units as performance can be quickly and better measured

2009

Keep Your Identity Small

- Politics and religion yield such useless discussions as people don't feel they need to have any particular expertise to have opinions about it. All they need is strongly held beliefs
- What these two may have in common that cause this is that they are part of people's identity and people can never have fruitful arguments about something that's part of their identity
- The most intriguing things about this theory, if it is 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
- There is a step beyond thinking of yourself as x and tolerating y: not even to consider yourself an x. The more labels you have for yourself, the dumber they make you

Startups in 13 Sentences

- If forced to choose one, it is understand your users. That's the key
- It's better to make a few people really happy than make a lot of people semi-happy
- Pick good cofounders
- Launch fast - pretext for engaging users
- Let your ideas evolve
- Understand your users - hard part is seeing something that users lack. The better you understand them the better your odds of doing so
- Better to make a few users love you than a lot ambivalent - easier to expand userwise than satisfactionwise
- Offer surprisingly good customer service - especially at first, even if it doesn't scale as it helps you learn about your users
- You make what you measure - corollary: be careful what you measure
- Spend little
- Get ramen profitable - makes just enough money to pay for the founders' living expenses. Great for morale and changes relationship with investors
- Avoid distractions
- Don't get demoralized
- Don't give up
- Deals fall through - don't get your hopes up

What I've Learned From Hacker News

- “The key to performance is elegance, not battalions of special cases.” – McIlroy and Bentley
- It's bad behavior you want to keep out more than bad users. Expectations on behavior are powerful (broken window theory)

- 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
- Important for editors and community to show what links and posts get killed

Can You Buy a Silicon Valley? Maybe.

- Long way would be to establish a top tier university in a place where rich people want to live. That's how SV happened. But you could possibly shortcut the process by funding startups. But this would take more money to force startups to stick around. This is an interesting prospect as with $1b, or about the price of a football stadium, you could turn a nice city into a first rate startup hub. This could maybe be done in 5 years and if successful, become a self-reinforcing chain like the one that drives SV
    - *This would be such a fascinating experiment if one had the capacity*
- Bad VCs tend to pick startups that are good at what they are: presenting
- What you ideally want is a pool of local angels but that hardly exists anywhere. What you could do is make a list of the most eminent angels and offer the startups they invested in $1m to move
- Don't be cheap, go for at least 30 startups. Let them work where they want and don't impose any other restrictions

Why TV Lost

- TV lost to computers because the internet is an open platform and innovates at hacker speed rather than big company speed. Second is Moore's law, third is piracy and lastly, social applications
- When a new medium arises that's powerful enough to make incumbents nervous, then it's probably powerful enough to win and the best incumbents can do is jump in immediately
    - *A bit late but interesting to note the recent news of Disney taking their catalogs off Netflix and starting their own steaming service*
- The internet dissolves the two cornerstones of broadcast media: synchronicity and locality
- TV networks are trying to create synchronicity by creating more live shows. Instead of delivering what viewers want, they're trying to force them to change their habits to suit the networks obsolete business model. This never works unless you have a monopoly or cartel to enforce it, and even then it only works temporarily

How To Be An Angel Investor

- Investing in the right startups is so much more important than anything else, than any contract or term could ever be
- Typically angels syndicate and band together with a lead negotiator. The easiest way to get started is find a friend who is an angel in a syndicate
- When you negotiate with a startup there are two numbers you care about: how much money you're putting in and the valuation of the company

- A typical round is about $150,000 from 5 people
- The valuation at these stages reflects nothing more than the strength of the company's bargaining position
- Agree in advance how much, if at all, you'll help them
- Investor mantra - pick the startup that makes things people want
- Good founders are the opposite of hapless. They make things happen the way they want. They are relentlessly resourceful
- There are differing opinions on whether to back great markets or great people but I think people is more important
- The best way to hear about startups is through referrals and you get those by being a useful, decisive, reliable investor who is a good person. Good angels are rare and there aren't more than a couple hundred in the whole Valley even though they may be the single most important ingredient in making the Valley what it is. Angels are the limiting reagent in startup formation
- Can hack being decisive by ratcheting down the amount you invest into something you won't stress about losing. Start by making 3-4 of these types of investments. Nothing will teach you like experience. Treat the first few as educational and $60,000 is less than a lot of graduate programs
- The best investors tend to be good people. They simply try to help everyone as much as possible and assume good things will eventually flow back to them somehow
    - *The world works by reciprocation – timing, magnitude and location yet to be determined*

Relentlessly Resourceful

- Boiling down being a good startup founder into two words: relentlessly resourceful
- Surprisingly often it can be taught
- Make something people want is the destination but being relentlessly resourceful is how you get there

Five Founders

- The five most influential founders for me
    - Steve Jobs - his sense of design is what makes him unique
    - TJ Rodgers - may be the best writer in the valley and have learned more from him about the startup way of thinking than anyone else - brutally candid, aggressively garbage collecting outdated ideas and yet driven by pragmatism rather than ideology
    - Larry & Sergey - pushed the idea that all you need are the best hackers to new heights
    - Paul Buchheit - responsible for GMail, AdSense, Google's mantra "don't be evil" and popularized the notion of a passionate small base of customers over a lukewarm large one

    - Sam Altman - the best founder in terms of strategy or ambition, he gets what he wants

The Founder Visa

- The single biggest thing that 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

Why Twitter is a Big Deal

- Because it's a new messaging protocol, and new protocols are rare. Even rarer is a protocol owned by a private company
- Lack of current monetization has allowed it to spread faster than it would have otherwise

A Local Revolution?

- Startups may represent a new economic phase on the scale of the Industrial Revolution
- Startups are a type of business that flourish in certain places that specialize in it. They don't seem to spread so well because they're mostly a social rather than a technical phenomenon and partly because they're not tied to geography
- The combination that has yielded Silicon Valley is a world class research university that is nice enough that rich people want to live there. Need both f
- ounders and investors

Maker's Schedule, Manager's Schedule

- Programmers dislike meetings so much because they're on a different type of schedule. Meetings cost them more. The manager's schedule is the one most people follow, where the day is chopped up into one hour intervals. However, maker's schedules are generally blocks of half a day. An hour is barely enough to get started. Therefore, a one hour meeting can ruin a whole day for a maker as now they don't have a stretch of time long enough to do anything meaningful
- Each schedule works fine by itself but problems arise when they meet
- These two ideas, combined, would be unique in that they wouldn't spread as other revolutions have

Ramen Profitable

- Ramen profitable means a startup makes just enough money to pay the founders' living expenses

- This gives the founders some flexibility as it might be nice to raise money but not required. This makes you more attractive to investors as it shows you're committed, already make something people like and can keep expenses low
- What keeps people from starting startups isn't the long hours or risk but the fear of having so much responsibility.
- Raising money is distracting and ramen profitability means you may be able to avoid it

The Trouble With the Segway

- People look smug when they ride the Segway because they seem not to be working hard enough
- Segway may be more successful if they made something that doesn't look so easy for the rider

What Kate saw in Silicon Valley

- What surprised Kate Courteau, designer of the YC office and member of the team, the most about the startup world: how many startups fail, how much startups' ideas change, how little money it can take to start a startup, how scrappy founders are (threatening and undignified), how tech-saturated SV is, speakers at YC are so consistent in their advice (launch fast, listen to users and then iterate; resilience is required to weather the emotional storm; most VCs are sheep), how casual founders are, how important it is for founders to have people they can seek advice from, what a solitary task startups are
- 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

The Anatomy of Determination

- The most important predictor of success is determination. 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
- The simplest form of determination is sheer willfulness. When you want something, you must have it, no matter what. In this, nature seems more important than nurture. Being strong willed has to be coupled and balanced by discipline. The word balance is key. 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. 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. A corollary: the more willful you are, the more dangerous it is to be undisciplined

  - *Another great example of the power of "blending" in characteristics not often found in your personality type to achieve leaping emergent effects – like in blending tin and copper to achieve bronze*

- Determination is interesting as the more you achieve the more temptations will arise and the more you'll have to build your determination or else slip backwards and your achievements revert to the mean. That's why Julius Caesar thought thin men so dangerous. They weren't tempted by the minor perquisites of power
- 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
- Unfortunately ambition seems somewhat rare and probably most ambitious people are starved for the sort of encouragement they'd get from ambitious peers, whatever their age. Which means one if the best ways to help society generally is to create events and institutions that bring ambitious people together
- 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: willfulness balanced by determination, aimed by ambition. And fortunately at least two of these qualities can be cultivated
- 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 do it, it's what you'd do anyway

The List of N Things

- Readers love "the list of n things" articles so much because they're easier to read than a regular article. Some of the work of reading an article is understanding its structure, its outline. A great reader probably has some such outline after reading an article but with lists, the work is already done for you
- It's easier for writers too as the threat that their ideas will run out won't kill their essay
- The damage is capped but it limits new thought. The whole point of Essays, when done right, is the new ideas you have while doing it

Post-Medium Publishing

- Almost every form of publishing has been organized as if the medium was what they were selling, and the content was irrelevant. Now that the medium is evaporating, publishers have nothing left to sell.
- If audiences were willing to pay more for better content, why wasn't anyone already selling it to them? 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 optimal way to make money from the written word probably requires different words written by different people

- I'm not too worried though as disruptions bring in as many exciting new forms as they kill
- When you see something that is merely reacting to new technology in an attempt to preserve some existing source of revenue, you're probably looking at a loser

Persuade xor Discover

- There are certain conventions which are harmless and even nice, such as saying “pleased to meet you” when meeting someone new. There are, however, some which are more insidious as they interact with the ideas. We're starting to move from social lies to real lies
- 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 up against reality
- Most writers write to persuade but I write to figure things out. I write to persuade a hypothetical perfectly unbiased reader
- 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
- Maybe I'm excessively tied to conciseness. I write essays the way I write code, reviewing it over and over to see what I can cut. You don't know what the ideas are until you get them down to the fewest words. The danger of pandering is not merely that it's longer but that the spin and ideas begin to get mixed together in your head
- 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' 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
- When I notice something surprising, it's usually very faint at first. There's nothing more than a slight feeling of discomfort. I don't want anything to get in the way of noticing it consciously

What Startups are Really Like

- I asked startups what surprised them which amounts to what I got wrong because if I'd explained things well enough, nothing should have surprised them
- Be careful with cofounders - careful who you pick and you have to work hard to maintain your relationship. Choose character and commitment over ability. This relationship has to be built of top quality materials and carefully maintained. It's the basis of everything
- Startups take over your life - it's so consuming it seems to stretch time so that a month is a huge time interval
- It's an emotional roller coaster - the highs higher and lows lower

- It can be fun - the freedom is amazing and working on something that is challenging, creative and something you believe in is huge. It's fun the way a survivalist training course is. Fun, if you're into that sort of thing and not at all if you're not
- Persistence is the key - if you are persistent, even problems that seem out of your control seem to work themselves out   I'm surprised over and over again how much more important persistence is than intelligence, ability
- Think long term - everything takes longer than you expect. There's a shocking amount of sheer stress at every point where a startup touches a more bureaucratic organization, like a big company or VC fund. 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. Expecting it to take longer takes pressure off and because you're relaxed, it's much easier to have fun and do good work. You can focus on doing what's best for users, employees, the product and the company overall
- Lots of little things - people rarely realize how rare it is to make it because they hit on some magic idea. You have to do lots of different, little things. Much more of a grind than glamorous. Never 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. 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
    - *Think this is what separates Amazon and Facebook. The thousands of experiments and iterations they do every single day. The willingness for 99% of these to be useless or even slightly damaging but it makes the 1% of surprising finds worth the time and energy*
- Start with something minimal - It's a mantra now but launch fast and iterate. People don't because of pride. Over-engineering is poison
- Engage users - the only way to do this initially is to launch your product. All products should be considered experiments. When you let customers tell you what they're after, they will often reveal amazing details about what they find valuable as well as what they're willing to pay for
    - *Again, being open and willing to listen and learn from each counterparty can teach you exactly how to behave and what to make. It can take so much risk and uncertainty out of the equation as they are telling you what they want and willing to pay for*
- Change your idea - the idea is a hypothesis and not a blueprint. Most times if something is hard you can just work harder. In startups I'd advise finding a problem that is easy for you to solve. You can never tell what will work. You simply have to do whatever seems best at each point
    - *"I don't look to jump over 7-foot bars, I look around for 1-foot bars I can step over." – Warren Buffett*
- Don't worry about competitors - one reason people overreact to competitors is because they overvalue ideas. But, it's usually execution that matters
- It's hard to get users - the question to ask is not whether your product is good but whether it's good enough to supply the activation energy required
- Expect the worst with deals

- Investors are clueless - angels tend to be much more knowledgeable as they often have technical experience
- 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. VCs get money from LPs by seeming confident, founders get money by seeming confident to VCs. This is the same phenomenon you see with defense contractors or fashion brands. The dumber the customers, the more effort you expend on the price of selling things to them, rather than making the things you sell
- Things change as you grow - the biggest change is that you get to program less
- Unconsciously, everyone expects a startup to be like a job and that explains most of the surprises. You probably can't overcome anything as pervasive as the model of work you grew up with. So the best solution is to be consciously aware of that

Apple's Mistake

- The App Store is an ongoing karma leak and ruining their reputation with programmers. They got into this mess because they fundamentally don't get software. Forcing software to fit their channel doesn't work like it does for music as developers must launch fast and iterate whereas Apple wants a finished product the first time.
- The real problem when you treat people poorly and establish a bad reputation is that you never get the best people and they flock to your competitors. Programmers are fussy about morale because they can work anywhere and believe that evil begets stupidity
    - *Treating employees poorly not only leads to worse products, lower morale and a worse culture but the most insidious part is that it culls people from the top. The best don't want to work in those types of environments and since they have the option to work pretty much anywhere, they will congregate in companies which treat them well. The best people will eventually vote with their feet*
- Developers usually build best when they actually use it and as almost all of them have iPhones, it seems like the vast majority will keep developing for the Apple platform. 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

2010

Organic Startup Ideas

- The best way to come up with startup ideas is to ask yourself the question: what do you wish someone would make for you? What's missing or broken in your daily life? Surprisingly often if you ask the question you'll get immediate answers. 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. Don't be discouraged if what you produce initially is something other people dismiss as a toy. In fact, that's a good sign.
- 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 have a gold mine. As with a 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
- There are two types of startups: 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
- I'd encourage anyone starting a startup to become one of its users, however unnatural it seems
- Organic ideas are preferable to the made up kind, but particularly so when the founders are young

How to Lose Time and Money

- The way most fortunes are lost is not through excessive spending but through bad investments. In most people's minds, spending money on luxuries sets off alarms that making investments doesn't. Luxuries seem self-indulgent. The solution is to develop new alarms. But this is difficult and often counterintuitive
- Losing time is similar in that the most dangerous way to lose it is not spending it having fun, but to spend it doing fake work. Things which superficially seem like real work are the most dangerous. The most dangerous traps 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

The Top Idea in Your Mind

- I realized recently that what one thinks about in the shower 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. There's a kind of thinking you do without thinking that is very powerful. This type of "ambient thought" is necessary for solving hard problems but the tricky part is you can't control it
- When most people's minds drift it goes towards their top problem in their mind. So it's a disaster to let the wrong ides become the top one
- Thus has been the lesson to 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

- There are two types of problems to especially avoid. Money and disputes. They push everything else out and are a time and attention sink. These two hurt you twice: first the injury itself and second by taking your time afterward thinking about it. If you can learn to ignore injuries you can at least avoid the second half. These things don't deserve space in your head

The Acceleration of Addictiveness

- The acceleration of addictiveness is scary because we don'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. 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
- One sense of normal is statistically normal: what everyone else does. The other is the sense we mean when we talk about normal operating range of a piece of machinery: what works best. This will only become more pronounced over time so that someone who is simply trying to live well would seem eccentrically abstemious in most of the US. 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
- It may actually become a reasonable strategy (or a more reasonable strategy) to suspect everything new as this trend accelerates
- My strategy to temper my internet addiction is long hikes. I used to like running because it took less time but I now see the slowness as an advantage as it gives me more time to think without interruption
- Things will always seem eccentric when you're trying to solve problems where there are no customs to guide you. We'll increasingly be defined by what we say no to

The Future of Startup Funding

- A couple years ago an unexploited area was angel-sized investments but this gap has closed in the last several years. This may in fact be good for angels as startups may turn to them even if they could turn to VCs. VCs are more prestigious but quality of investor is way more important than the type of round
- The power is shifting to founders and over time everything founders dislike will be eliminated. Speed of decisions and simple deals with simple terms are the future
    - *Again, as always, simply see through the eyes of the key stakeholders and you will not make (many) mistakes*
- Is it better or worse for founders to have more power? I had never considered the question. It has to be better if people with more knowledge have more power
- When investors stop trying to squeeze more out of current deals they'll end up net ahead actually because they'll encourage other deals to appear

What Happened to Yahoo!

- Yahoo! had two problems early on which Google didn't: easy money and ambivalence about being a technology company. They called themselves a media company as they made money by selling ads and if they called themselves a software company, they'd be pitting themselves against Microsoft. This caused an ambiguous identity and to not take programming seriously enough. They weren't obsessed with hiring the best programmers. Good programmers want to work with other good programmers so this leads to a virtuous or vicious cycle. Once the quality of programmer's slips, the death spiral starts from which there is no recovery. However, there is one way around this. To buy good programmers by buying their startups. But so far the only companies smart enough to do this are companies smart enough not to need to
- Any company that has to have good software needs to have a hacker-centric culture

High Resolution Fundraising

- The fact that the most important data point for investors is who else is investing leads to deadlock.
- The future lies in different terms for different investors. Markets always evolve towards higher resolution
- Another issue was that startups had to decide beforehand how much they wanted to raise. This isn't reasonable as picking an optimal round size depends on the reactions of investors and this is impossible to predict

Where to See Silicon Valley

- Stanford, University Ave, the Lucky Office, old Palo Alto, Sand Hill Road, Castro Street, Google, Skyline Drive, 280

The New Funding Landscape

- Super angels - invest more than angels and less than VCs. They're forcing VCs to make some angel-style investments themselves. They arose because it is what founders demanded. They invest quickly and with other people's money. What super angels really are is a new form of a 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
- Raising an angel round is not only quicker but you get feedback as it progresses
- Big returns come from a few big successes and it therefore matters far more which startups you picked than how much you paid for them. This might make sense for super angels because they're hoping for a quick buyout rather than an IPO like VCs

What We Look For in Founders

- Determination
- Flexibility - need to be able to modify your dreams on the fly
- 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. Most startups seem like bad ideas initially because if they were obvious others 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
- Naughtiness - good people with a gleam in their eye. They delight in breaking rules but not rules that matter. Ask people for times when they hacked something to their advantage - hacked in the sense of beating the system, not breaking into computers
- Friendship - relationship between the founders has to be strong

Tablets

- Mobile devices are better thought of as tablets.
- Many if not most of the special purpose objects around us are going to be replaced by apps running on tablets. I think people will be surprised what and how many things get replaced. 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. No one who has studied history would want to underestimate the power of this force. Apple has this tremendous force behind it
    - *This rings incredibly true for me. It's almost a compulsion for me to cull as much as possible. From possessions to habits to negative or useless thoughts. Every year I "shed" a little less as my current habits, thoughts, actions, etc. have stood up to the challenge but I never stop looking for false or beloved ideas to debunk*
- The power of platforms is that the more versatile the tool the more surprising directions developers can take them

Founder control

- It is becoming more common for founders to retain control of the board after series A rounds

2011

Subject: AirBnb

- Fascinating email exchange between PG and Fred Wilson regarding AirBnb at an early stage

The Patent Pledge

- Bad uses of patents seem to be increasing so there is an increasing call for patent reform. One way that using patents that clearly does not encourage innovation is when established companies with bad products use patents to suppress small competitors with good products. The way to solve for this without getting the government involved 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
- The pledge - no first use of software patents against companies with less than 25 people
- Though constraining, like Google’s don't be evil, this attracts the most productive people as they are attracted to employers who hold themselves to a higher standard than the law requires
- Companies that use patents on startups are attacking innovation at the root

Why Startup Hubs Work

- If you're sufficiently good at generating your own morale, you can survive without external encouragement. For mere mortals, however, being in a hub is a great source of support. Death is the default for startups and most towns don't save them
- 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
- There is a difference between starting a business and a startup: a startup is a new business designed to scale
- Having people around you care about what you're doing is an extraordinarily powerful force
- Chance meetings produce miracles to compensate for the disasters that characteristically befall startups. If you're in a startup hub, unexpected good things will happen to you, especially if you deserve them. Chance meetings let your acquaintances drift in the same way taking a shower lets your thoughts drift. The critical thing in both cases is that they drift the right amount
- 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
    - *Establishing this kind of culture and mindset would be beneficial in any organization. You have to get the environment right, make people feel secure, safe,*

calm and stable, in order for them to "fill others' cups" with no expectation of getting anything in return

- You need a lot of people to start a startup hub because once you get 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

Snapshot: Viaweb, June 1998

- Reminiscing about Viaweb and how much things have changed since then

2012

Schlep Blindness

- Startup ideas are constantly right under our noses but we don't see them because of schlep blindness. This tedious work is unpleasant but shouldn't be avoided 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
- 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
- The most valuable antidote to schlep blindness is ignorance
- A trick is to ask yourself' "what problem do I wish somebody else would solve for me?" Rather than "what problem should I solve?"

A Word to the Resourceful

- 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. The least successful ones tend to be hard to talk to. It seemed like they never quite understood what I was saying
- A word to the wise is sufficient. You don't have to explain things in detail, they'll chase down all the implications
- Great groups soak in everything you say with fresh eyes, even if it's dismissed. Poor groups seem to have made up their minds already and make what you say conform to what they've already decided or just outright dismiss it

Frighteningly Ambitious Startup Ideas

- The biggest startup ideas are terrifying. And not just because they'd be a lot of work. The biggest ideas seem threatening to your identity: you wonder if you'd have enough ambition to carry them through
- You'd expect big startup ideas to be attractive, but actually they tend to repel you. And this 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. This is 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
- A new search engine – there seem to be some cracks in Google's fortress and now search results seem to bring up results inspired by the Scientologist principle that what's true is what's true for you. The way to win here is to build the search engine all the hackers use,

whose users consisted of the top 10,000 hackers and no one else would be in a very powerful position despite its smalls size

- Replace email – email was not designed to be used the way we use it now. It is not a messaging but a to-do list. This new protocol should be a to-do list protocol, not a messaging protocol. It should give the recipient the power to what someone else can put on my to-do list. It is impossible to kill a protocol but in this case the most powerful people would change to a better one as they too are at the mercy of email today. Even if it is simply Gmail but faster, I would pay a lot. I'd probably be justified in spending $1000 considering how much time I spend on it and it would be a relatively cheap way to make my life better
- Replace universities – they won't be replaced wholesale but they'll lose the de facto monopoly on certain types of learning they once had. There will be many different ways to learn different things, and some may look quite different from universities. Learning is such a big problem that changing the way people do it will have a wave of secondary effects, such as decoupling credentials from the university
- Internet drama – 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
- The next Steve Jobs – empirically you can't seem to get visionary CEOs by hiring them. 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. This sounds preposterous but they have one advantage: Apple and Steve Jobs have 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
- Bring back Moore's Law – 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?
- Ongoing diagnosis – it will seem crazy in the future that people had to wait for symptoms before getting diagnosed. There will be resistance as it goes against how things have been done for millennia.
- One of my tricks for generating startup ideas is to imagine the ways in which we'll seem backward to future generations
- If you're going to take on one of these massive problems, don't make a direct frontal attack on it. If you're going to replace email, don't say so at first. Instead, say you're building to-do-list-software. People can notice you've replaced email when it's a fait accompli. Empirically the way to do really big things seems to be to start with deceptively small things. Maybe it's a bad idea to have really big ambitions initially, because the bigger your ambition, the longer it will take to get there and the further you project into the future, the more likely you'll get it wrong. Start with what you know works and when you expand,

expand in the right general direction. 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

Defining Property

- What counts as property depends on what works to treat as property. And that not only can change but has changed. Humans may 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 now
- 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
- 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

Writing and Speaking

- Although I'm not a great speaker, 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
- Reading talks out loud is very helpful in that it can expose awkward parts
- 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
- Talks aren't as good as writing as a source for ideas but they're the closest most of us can get to having a conversation with someone like the president. They're also motivating. This may have been their original purpose. The emotional reactions you can elicit with a talk can be a powerful force

The Top of My To-Do List

- If you had to compress the top 5 mistakes Bronnie Ware compiled from the dying 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 inverted these 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

Black Swan Farming

- I don't know of another business as counterintuitive as startup investing. Effectively all returns are concentrated in a few big winners and the best ideas look initially like bad ideas
- 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. You need to do what you know intellectually to be right, even though it feels wrong. 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 altitude 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
- Financially, it's truly not the chance of succeeding that matters but the chance of succeeding really big. 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
- The sweet spot for startup ideas are ideas which seem bad but are good ideas.
- When I first heard about Facebook it sounded lame to me. 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
- I can logically lay out and understand the reason for taking more risk and still not do it. The real reason we're as conservative as we are is that we haven't assimilated the fact of 1000x variation in returns

Startup = Growth

- 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 to take venture funding, or have some sort of "exit." The only essential thing is growth. Everything else we associate with startups follows from growth. 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
- 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

- 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 whereas with startups, you're probably going to have to think of something fairly novel.
- Successful startup founders aren't obviously making a conscious effort to find ideas everyone else has overlooked. Usually successful startups happen because the founders are sufficiently different from other people that ideas few others can see seem obvious to them.
- 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. 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. 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
- The growth of a successful startup usually has three phases: (1) 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 and growth will slow partly due to its internal limits and partly because the company is starting to bump up against the limits of the markets it serves
- The one number every founder should always know is the company's growth rate. That's the measure of a startup. If you don't know that number, you don't know if you're doing well or badly. The best thing to measure the growth rate of is revenue. The next best, for startups that aren't charging initially, is active users
- 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." Programmers will recognize what we're doing here. We're turning starting a startup into an optimization problem. 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
- This forces founders to act and acting versus not acting is the high bit of succeeding
- Most fairly good ideas are adjacent to even better ones
- 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
- 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

- 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. 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

How to Get Startup Ideas

- 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 could build, and that few others realize are worth doing. Microsoft, Apple, Yahoo!, Google, Facebook all began this way
- By trying to solve a problem you yourself have you ensure the problem really exists. Too many times people invent models of the world that don't correspond to reality and work from that.
- Always choose to make something a small number of people want in a large amount than something that a large amount wants in a small amount. Build a startup that is like a well – narrow and deep. You get narrowness as a byproduct of optimizing for depth (and speed). But you almost always do get it. Who will use version one even if it's a crappy version made by a two person startup they've never heard of. The users' need has to give them sufficient activation energy to start using whatever you make, which can vary a lot
- How to escape from the narrowness is often impossible to see at first or at least vague. Bill, Mark and the AirBnb guys probably had no idea at the beginning how big of a space they were tapping into. They just knew they were onto something. So how do you choose between ideas if you can't see a way out? The truth is disappointing but interesting: if you're the right person you have the right hunches.
- 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." Paul Buccheit 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. Most things that are missing will take some time to see. You almost have to trick yourself into seeing the ideas around you. Pay particular attention to things that chafe you. 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 yourself
- 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. The verb you want to be using with respect to startup ideas is not "think up" but "notice." These are organic startup ideas
- You need to loosen up your mind and not make too much of a direct frontal attack – i.e., sitting down at a desk and trying to think of ideas. The best plan may just be 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. Have this second self keep a journal and each night make a brief entry listing the gaps and anomalies you'd noticed that day. Not startup ideas, just the raw gaps and anomalies. Taking time to come up with your ideas 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

- 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. Working on things that could be dismissed as "toys" often produces good startup idea. Toys have everything except being important – users love them, it's cool, it just doesn't matter. But if you're living in the future and build something cool that users love, it may matter more than outsiders think. An even better version of "live in the future and build what's missing" is "live in the future and build what seems interesting"
- The clash of domains is an especially fruitful source of idea. In fact, you're doubly like to find good problems in another domain: people don't have your expertise in your main field and since you're totally ignorant of this new domain, you don't even know what the status quo is. If you're an undergrad, don't take a class on entrepreneurialism. Get a summer job working at a biotech company or a new domain of your choice
- It's no surprise both Microsoft and Facebook were built in January. At Harvard it was reading period, when students have no classes to attend because they're supposed to be studying for finals
- However, don't feel like you need to build things that will become startups. That's premature optimization. Just build things
    - *There is no bigger waste of time than doing something efficiently which shouldn't be done at all. The most insidious distractions seem productive but truly aren't*
- The more a project has to rely on research, the less likely it is something that could be turned into a startup
- 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. It's exceptionally rare for startups to be killed by competitors – so rare that you can almost discount the possibility. If you're uncertain, ask users. Whether you succeed depends far more on you than on your competitors. So better a good idea with competitors than a bad one without
- A crowded market is actually a good sign, because it means there's both demand and that none of the existing solutions are good enough
- 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. 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. Stripe benefited from schlep blindness and overall might have experienced less pain than most startups because the fear of dealing with payments kept most others away

- While organic ideas are often more promising, coming up with ideas is useful, you just have to be more disciplined. When searching for ideas, look in areas where you have some expertise. If you think you have a good idea in an unrelated area, you most likely don't. You can't think of a good idea in your expertise because your standards are higher here. Your ideas in other areas 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. It's especially good if you're different in a way people will increasingly be. A particularly promising way to be unusual is to be young
- Trying to sell something bad can often be a source of better ideas. When you see nobody wants to buy what you're about to build, you very often come back with a real idea that you discovered in the process of trying to sell the bad idea
- The next best thing to an unmet need of your own is an unmet need of someone else. Try talking to everyone about the gaps they find in the world. What's missing/what would they like to do but 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
- One way to bypass the schlep and 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?
- 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 a mixture of disdain in the big players' attitude, because that often misleads them
- Another good tactic is to ride some wave bigger than yourself. What are we unconsciously ruling out as impossible that will soon be possible?

2013

Startup Investing Trends

- I think investors will make more money in the future. It is becoming cheaper to start a startup and startups are becoming a more normal thing to do
- Founders are getting the upper hand over investors but they still need their energy and imagination, even if they don't need as much money from them. However, there will still be some massive winners and there are potentially 10-50x more good founders out there and now that the barrier is not as intimidating, they may join as well. Returns don't come from investing at low valuations. They come from investing in the companies that do really well. This may lead to even larger variability between VCs
- 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
- Big hits won't grow linearly with the total number of new startups because of idea clashes
- 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. VCs want to give too much money in series A rounds and often make companies take more money than they want in order to get a larger percentage of the company. This is an opportunity as the first VC to break rank and start to do series A rounds for as much equity as founders want to sell (and with no "option pool" that comes from the founders' shares) stands to reap huge benefits. This will get the firm all the best startup deals as it is better for founders
    - *This can lead to a virtuous cycle once you become the "benevolent black hole" where all founders want to work with you as you act in a way which better serves them*
- If you want to find new opportunities for investing, look for things that founders complain about

Do Things That Don't Scale

- One of the most common types of advice we give at YC 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
- 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. There are two reasons founders resist going out and recruiting users individually. One is a combination of shyness and laziness. Usually the CEO will have to spend a lot of time on sales and marketing. 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.

- 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."
- 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.
- How do you find users to recruit manually? If you build something to solve your own problem, 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
- You should take extraordinary measures not just to acquire users, but also to make them happy. 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. 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 companies they've been customers of, which are mostly big ones. That's one advantage of being small: you can provide a level of service no big company can. 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. Once you realize that existing conventions are not the upper bound on user experiences, it's interesting in a very pleasant way to think about how far you could go to delight your users
    - *Take the juice and leave the rind. Every hero, great company, etc. has both pros and cons. Don't holistically try to imitate them but adopt the positives and, to the extent possible, mitigate the negatives*
    - *AirBnb thought experiment of what a "10/10" experience would look like – parades, all of a guest's favorite things are known and provided, etc. Though not possible in real life, it provides a starting point from which you can work backwards. All of a sudden providing an 8/10 experience seems feasible and actionable*
- Your attention to users should be, as Steve Jobs said, "insanely great." He was 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. At the larval stage, it's not the product that should be insanely great, but the experience of

being your user. The feedback you get from engaging directly with your earliest users will be the best you ever get.

- Sometimes the right unscalable trick is to focus on a deliberately narrow market. It's like keeping a fire constrained at first to get it really hot before adding more logs. It's always worth asking if there's a subset of the market in which you can get a critical mass of users quickly. 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
- 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.
- 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.
- One sort of initial tactic that usually doesn't work: the Big Launch. I occasionally met 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.
- 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 not enough just to do something extraordinary initially. You have to make an extraordinary effort initially. Any strategy that omits effort – whether it's expecting a big launch to get 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 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 in two dimensions. 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' 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.

How to Convince Investors

- Inexperienced founders try to convince investors 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. 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 founders is the most important. A formidable person is one who seems like they'll get what they want, regardless of whatever obstacles are in their way. Formidable is close to confident, except that someone could be confident and mistaken. Formidable is roughly justifiably confident. 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. The way to seem most formidable as an inexperienced founder is to stick to the truth. The secret is to convince yourself first 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 raise money. You don't have to be a smooth presenter if you understand something well and tell the truth about it. When you try to convince investors, 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
- 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
- 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? Besides formidable founders, you need a plausible path to owning a big piece of a big market. But the market doesn't have to be big yet, nor do you necessarily have to be in it yet. There just has to be some plausible sequence of hops that leads to dominating a big market a few years down the line. 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 you, 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
- 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 condiment, which they like, and you'll probably also do a better job of presenting that aspect of your startup

- So here’s the recipe for impressing investors when you’re not already good at seeming formidable: make something worth investing in, understand why it’s worth investing in, explain that clearly to investors

Investor Herd Dynamics

- The biggest component in most investors’ opinion of you is the opinion of other investors. Which is of course a recipe for exponential growth. 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. 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
- 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

How to Raise Money

- Fundraising is hard in two senses: hard like lifting a heavy weight and hard like solving a puzzle
- 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 the 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
- Rules
    - Don’t raise money unless you want it and it wants you
    - Be in fundraising mode or not, because it is so distracting
    - Get introductions to investors – the best kind is from a well-known investor who has just invested in you and the next best is from a founder of a company they’ve funded
    - Hear no till you hear yes – the worst ones lead you on and never say no
    - Do breadth-first search weighed by expected value – always talk to investors in parallel rather than serially but give higher priority to the more promising ones

    - Know where you stand – look at investors’ actions rather than their words. Ask what more do they need in order to decide, what happens next, 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? A good proxy to tell where you stand is the resources the investors expend on you after the first meeting
    - Get the first commitment
    - Close committed money
    - Avoid investors who don’t “lead”
    - Have multiple plans – you shouldn’t be planning to raise a specific amount but have multiple plans depending on how much you can raise
    - Underestimate how much you want – a good metaphor here is angle of attack. If you try to fly at too steep an angle of attack, you just stall. 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 – not being desperate by being profitable quickly makes you a more attractive candidate
    - Don’t optimize for valuation – most important thing to understand about valuation is that it’s not that important. The real test is revenue
    - Yes/no before valuation – resist giving potential investors a valuation if it isn’t yet set. Tell them valuation isn’t the most important thing to you and that you haven’t thought much about it and that you are looking for investors you want to partner with and who want to partner with you
    - Beware “valuation sensitive” investors
    - Accept offers greedily – 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 now. You can’t predict the future and your priority should be to finish fundraising and get back to work
    - Stop fundraising when it stops working
    - Don’t get addicted to fundraising
    - Don’t raise too much – raising money in subsequent rounds at lower valuations is seen as a bad sign so be conservative initially. But, more dangerous than the valuation may be the money. The more you raise, the more you spend and spending a lot of money can be disastrous for an early stage startup
    - Be nice – it’s a mistake to behave arrogantly to investors. In fact, investors who reject you are some of your warmest leads for future fundraising
    - The bar will be higher next time – every subsequent round will be held to a higher standard
    - Don’t make things complicated – here is fundraising in one sentence: 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
- The founder who handles fundraising should be the CEO, who should in turn be the most formidable of the founders

- 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
    - *So often you have customers or any of the counterparties* telling *you what they want and, yet, many don't listen*
- 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 in order to win

Before the Startup

- 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. 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 – 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 YC I used to joke that our function was to tell founders things they would ignore. It's really true. 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. 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. You can, however, trust your instincts about people
    - Expertise – 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. In fact, I worry it's not merely unnecessary to learn in great detail about the mechanics of startups, but possibly somewhat dangerous
    - Game – people are trained their whole lives to play games rather than doing the real thing and young founders' first impulse on starting a startup is to try to figure out the tricks for winning at this new game. They want to know the tricks 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. Young founders make things so complicated because 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. There is no boss or teacher 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. 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. It's exciting that there even exists 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

  - *Don't make the marginal the core nor the core marginal. Intelligent people weigh every decision by opportunity cost and spending your time in the right area, in the right way, at the right time is make or break*

- All-Consuming – the difficulty of being a successful startup founder is concealed from almost everyone except those who've done it. The total volume of worry never decreases; if anything it increases. Starting a startup is intrinsically something you can only really learn by doing it. Do not start a startup in college as it will take over your life and you will no longer be a startup. How to start a startup is just a subset of a bigger problem you're trying to solve: how to have a good life
- Try – you can't tell if starting a startup is too hard. What you're trying to estimate is not just what you are, but what you could grow into, and who can do that? I may have more experience than anyone in trying to predict how tough and ambitious founders would become and I can tell you how much an expert knows about it, and the answer is: not much. I've read that the same is true in the military as in early startup founders – 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
- Ideas – the sixth counterintuitive point is that the way to get startup ideas is to try to think of startup ideas. 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. 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. So how do you find things that matter? 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. I'm not sure how others do it but this seems to be my compass. Indulging in these energetically is the best way to prepare yourself for a startup. And indeed, probably also the best way to live life. I did in fact 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?

    - One way to develop a 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 Buccheit 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
    - 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.
    - 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 deadline. For unambitious people this sort of things is the dreaded "failure to launch" but for the ambitious ones it can be an incomparably valuable sort of exploration
        - *This resonates a lot with me after my ~4 months of travel in summer 2017. Seeing new cultures and gaining a deeper appreciation for these people and how they view the world has been priceless. A broad spectrum of experiences and interactions is a sign of a rich life and you get this more extensively through travel than perhaps any other way*

Mean People Fail

- It struck me recently how few of the most successful people I know are mean. There are exceptions, but remarkably few. What's going on here? Are meanness and success inversely correlated? Part of what's going on is of course selection bias. I only know people who work in certain fields: startup founders, programmers, professors.
- So, why do mean people tend to fail? 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

  - *Being able to maintain equanimity in these difficult times will help you make better decisions, see the big picture, think long term, act rather than react…*

- Another reason mean founders lose is that they can't get the best people to work for them. The best people have other options and vote with their feet. 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. 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
  - *Today the vicious 3G business model and GE's mantra of firing the bottom 10% are taught and revered yet the ripple effects are hardly ever talked about or recognized – these models don't only cull from the bottom but also from the top. The best people don't want to work in this type of an environment so they opt to work for a benevolent company with high aspirations. The best can and do (eventually) vote with their feet*
- 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 which we got by fighting. For most of history, success meant success at a zero-sum game. 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. In order for this to become possible, you need a certain degree of civil order. People need to feel that what they create can't be stolen

The Fatal Pinch

- 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. That last sentence is the fatal one.
- What bites them the second time they try to raise money is a confluence of three forces: the company is spending more now than it did the first time it raised money, investors have much higher standards for companies that have already raised money and the company is now starting to read as a failure

- 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
- If you're already in the fatal pinch you have zero chance of raising more money. Three options remain: you can shut down the company, you can increase how much you make, and you can decrease how much you spend
- Changing the question 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

How You Know

- 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
- 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
    - *"Any book which is at all important should be reread immediately." – Arthur Schopenhauer*

How to be an Expert in a Changing World

- 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 you 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. (My trick is to talk about aspects of the present that most people haven't noticed yet). 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
    - *Does a tree have any idea which direction to grow? No. It is "aggressively open-minded" and grows in the direction which it can. It makes the most of opportunities that present themselves. One cannot predict how it will look and which direction it will grow, but it will be optimal given the circumstances because it adapted to its circumstances and didn't have a rigid plan it tried to impose on a changing world. "If we do not let the world teach us, it teaches us a lesson." – Joseph Tussman*
- 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. 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. 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. Within YC, 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
- Anyone who must in some sense bet on ideas rather than merely commenting on them has similar incentives. Which means anyone who wants to have such incentives can, 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
- 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 to be 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 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

Let the Other 95% of Great Programmers In

- 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. This is what anti-immigration people don't understand. 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.
- The US is so lucky right now that if we let great programmers come, they'll be breaking down the door. However, this may not always be the case and if this trend ceases, the US is screwed. And unlike other potential mistakes on this scale, it costs nothing to fix. So please, get on with it

Don’t Talk to Corp Dev

- 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 you want to sell your company right now and you’re sufficiently likely to get an offer at an acceptable price
- 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
- 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. 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

What Doesn’t Seem Like Work?

- 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 a chapter. To my father 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
- 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 all 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?

  - *For me I think it would be doing deep work on topics that fascinate me. "Pre-paying" the effort today by deeply reading, digesting, organizing, and indexing books and other resources so that I ingrain it and internally organize it for future recall and easy reference. The long-term gratification gene is also something I too often take for granted as is my willingness to fail and not worrying about standing out, being different or looking stupid.*

The Ronco Principle

- 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 to even 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. 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. 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. 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

What Microsoft Is this the Altair Basic of?

- 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

Change Your Name

- 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 x.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. Even good founders can be in denial about this. There's nothing intrinsically great about your current name. Nearly all your attachment to it comes from it being attached to you

Why It's Safe for Founders to Be Nice

- 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
- When anything grows at the rate of a successful startup, the Y axis will take care of itself, even if the initial numbers are smaller
- 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

- 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
    - *If you're at a "table" – an industry, company, etc. – where rapacity is the name of the game, it might be better to switch tables than compete in that arena. It certainly is one way to make money and achieve some semblance of "success" but do you really want to spend your one life fighting others to make money rather than celebrating and building something meaningful with great people?*

Default Alive or Default Dead?

- 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's 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 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? So few founders know this because early on it is meaningless. It is like asking 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. 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
- 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. VCs may push to overhire because their risk profile is different from yours, they're protected by the portfolio effect. VCs want to blow you up, in one sense of the phrase or the other. But as the founder your incentives are different. You want above all to survive
- At the early stages, the product needs to evolve more than to be "built out," and that's usually easier with fewer people. AirBnb waited 4 months after raising money at the end of YC 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.

- Steep revenue or usage growth interests 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

Write Like You Talk

- 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 on 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.
- 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
- 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
- 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 an 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. For drastic cases, you might not be able to fix it 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

A Way to Detect Bias

- You can use this technique to detect bias in a selection process without knowing anything about the applicant people. You need (a) to 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. 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

Jessica Livingston

- Much of what's most novel about YC is due to Jessica. If you don't understand her, you don't understand YC. So let me tell you a little bit about Jessica
- 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. 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
- 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
- Jessica doesn't tend to ask many questions, but 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
- 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 be 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 egregious, they were going to be part of it for life. This alumni feature is certainly among the most valuable features of YC
    - *The most impressive thing about having a social radar is having the guts to listen to it and trust it and not let your instinct be stilled because of external, financial or other short-term motivating factors.*

- *How does one develop this skill to the highest level? Empathy. Acting in such a way so that you deserve to be treated well. Trust. Knowing what you want (to be loved, respected, cared for, listened to…) and look for in others (trust, leadership, empathy…) and simply exhibiting those qualities at* all *times. Awareness. Presence. Active listening. Patience...*

- 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.
- 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. She also hates bragging. Her horror of ostentation is so visceral it's almost a phobia. She also hates fighting. She can't do it; she just shuts down. So although Jessica more than anyone made YC unique, the very qualities that enabled her to do it means she tends to get written out of YC's history.

2016

The Refragmentation

- 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. And much more. 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. Both the wars and large corporations caused both social and economic cohesion. 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.
- The rise of national corporations didn't just compress us culturally as they tried to make us all similar. It compressed us economically too, and on both ends. They overpaid union workers and underpaid top management
- 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. 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
- 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 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. 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
- 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 disintegrated 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 to 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 JP Morgan's world as the natural state of things, began to realize it wasn't the last word after all

- Computers reduce the transaction costs that Coase argued are the raison d'etre of corporations. That is a fundamental change
- 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
- 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
- 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. 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. 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
- Baumol's cost disease (or the Baumol effect) is the rise of salaries in jobs that have experienced no increase of labor productivity, in response to rising salaries in other jobs that have experienced labor productivity growth. 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

Economic Inequality

- I'm interested in the topic of income inequality 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 startups 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. So have we just shown, by reductio ad absurdum, that it's false that economic inequality should be decreased? That doesn't sound right either. Surely it's bad that some people are born practically locked into poverty, while at the other extreme fund managers exploit tax loopholes to cut their income taxes in half. The solution to this puzzle is to realize that economic inequality is not just one thing. It consists of some things that are very bad, like kids with no chance of reaching their potential, and others that 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 naïve version of which is the one based on the pie fallacy: that the rich get rich by taking money from the poor
- 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. 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. 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
- 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. 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
- 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.
- I think rising 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. The US is the bellwether. The situations we face here, the rest of the world will sooner or later

- You do not want to design your society in a way that's incompatible with the exponential technology curve. The evolution of technology is one of the most powerful forces in history
- 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
- Poverty and inequality are not identical. When the city is turning off your water because you can't pay the bill, it doesn't make any different what Larry Page's net worth is compared to you. He might only be a few times richer than you, and it would still be just as much of a problem that you water was getting turned off
- One of the most important measures 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. 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
- 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. If our goal is to decrease economic inequality, then it is equally important to prevent people from becoming rich and to prevent them becoming poor. I believe it's far more important to prevent people becoming poor. And that therefore decreasing economic inequality should not be our goal.

Life is Short

- 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. 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 insures 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.

- 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 on them 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, its better most of the time not to defend yourself. Otherwise these people are literally taking your life.
- 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.

  - *One heuristic for distinguishing stuff that matters is to ask yourself whether you'll care about it in the future.*

- 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.
  - *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*
- 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.

   - *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.*

How to Make Pittsburgh a Startup Hub

- When I was a kid, Pittsburgh 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 people in a typical startup is right in that 25 to 29 bracket
- 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 suburb. 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, but 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.

- 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 it's in Siberia. 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 very 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.

This Year We Can End the Death Penalty in California

- When I was younger I used to think the debate about the death penalty was about when it's ok to take a human life. Is it ok to kill a killer? But that is not the issue here. The real world does not work like the version I was shown on TV growing up. The police often arrest the wrong person. Defendants' lawyers are often incompetent. And prosecutors are often motivated more by publicity than justice. In the real world, about 4% of people sentenced to death are innocent. So this is not about whether it's ok to kill killers. This is about whether it's ok to kill innocent people. A child could answer that one for you. This year, in California, you have a chance to end this, by voting yes on Proposition 62. But beware, because there is another proposition, Proposition 66, whose goal is to make it easier to execute people. So yes on 62, no on 66. It's time.

2017

The Risk of Discovery

- 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.

Charisma/Power

- 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.

General and Surprising

- The most valuable insights are both general and surprising. $F = ma$ for example. But general and surprising is a hard combination to achieve. That territory tends to be picked clean, precisely because those insights are so valuable. Ordinarily, the best that people can do is one without the other: either surprising without being general (e.g. gossip), or general without being surprising (e.g. platitudes). Where things get interesting is the moderately valuable insights. You get those from small additions of whichever quality was missing. The more common case is a small addition of generality: a piece of gossip that's more than just gossip, because it teaches something interesting about the world. But another less common approach is to focus on the most general ideas and see if you can find something new to say about them. Because these start out so general, you only need a small delta of novelty to produce a useful insight. A small delta of novelty is all you'll be able to get most of the time. Which means if you take this route your ideas will seem a lot like ones that already exist. Sometimes you'll find you've merely rediscovered an idea that did already exist. But don't be discouraged. Remember the huge multiplier that kicks in when you do manage to think of something even a little new. Corollary: the more general the ideas you're talking about, the less you should worry about repeating yourself. If you write enough, it's inevitable you will. Your brain is much the same from year to year and so are the stimuli that hit it. I feel slightly bad when I find I've said something close to what I've said before, as if I were plagiarizing myself. But rationally one shouldn't. You won't say something exactly the same way the second time, and that variation increases the chance you'll get that tiny but critical delta of novelty. And of course, ideas beget ideas. (That sounds familiar.) An idea with a small amount of novelty could lead to one with more. But only if you keep going. So it's doubly important not to let yourself be discouraged by people who say there's not much new about something you've discovered. "Not much new" is a real achievement when you're talking about the most general ideas. Maybe if you keep going, you'll discover more. It's not true that there's nothing new under the sun. There are some domains where there's almost nothing new. But there's a big difference between nothing and almost nothing, when it's multiplied by the area under the sun.

Appendix

Essays

- Paul Graham's Essays

Books

- *Hackers and Painters: Big Ideas From the Computer Age* by Paul Graham

Other

- Y-Combinator Home Page
- Y-Combinator YouTube Channel
- Y-Combinator Podcast
- How to Start a Startup YouTube Series

Hackers and Painters: Big Ideas From the Computer Age by Paul Graham

Summary

1. Paul Graham, founder of Viaweb and Y-Combinator, gives an inside view on the computer world and the motivations behind the people in it. Discussing how we think, how we work, how we develop technology and how we live

Key Takeaways

1. Why Nerds are Unpopular - Their minds are not on the game
    1. So, if intelligence itself is not a factor in popularity, why are smart kids so consistently unpopular? The answer, I think, is that they don't really want to be popular. Not enough, anyway. There was something else they wanted more: to be smart, to make great things. The main reasons nerds are unpopular is that they have other things to think about.
    2. Another reason kids persecute nerds is to make themselves feel better. When you tread water, you lift yourself up by pushing water down. Likewise, in any social hierarchy, people unsure of their own position will try to emphasize it by maltreating those they think rank below. I've read that is why poor whites in the United States are the group most hostile to blacks
    3. I think the important thing about the real world is not that it's populated by adults, but that it's very large, and the things you do have real effects. That's what school, prison, and ladies-who-lunch all lack. The inhabitants of all those worlds are trapped in little bubbles where nothing they do can have more than a local effect. Naturally these societies degenerate into savagery. They have no function for their form to follow. When the things you do have real effects, it's no longer enough just to be pleasing. It starts to be important to have the right answers, and that's where nerds show to advantage. The other thing that's different about the real world is that it's much larger. In a large enough pool, even the smallest minorities can achieve a critical mass if they clump together
    4. What bothers me is not that the kids are kept in "prisons" but that (a) they aren't told about it, and (b) the prisons are mostly run by the inmates. Life in this twisted world is stressful for the kids. And not just for the nerds. Like any way, it's damaging even to the winners.
    5. As far as I can tell, the concept of the hormone-crazed teenager is coeval with suburbia. I don't think this is a coincidence. I think teenagers are driven crazy by the life they're made to lead. Teenage apprentices in the Renaissance were working dogs. Teenagers now are neurotic lapdogs. Their craziness is the craziness of the idle everywhere.
    6. Teenagers seem to have respected adults more in the past because the adults were the visible experts in the skills they were apprenticing in. Now most kids have little idea what their parents do in their distant offices, and see no connection (indeed, there is precious little) between schoolwork and the work they'll do as

adults. What happened? We're up against a hard one here. The cause of this problem is the same as the cause of so many present ills: specialization. As jobs become more specialized, we have to train longer for them and teenagers are now useless, except as cheap labor in industries like fast food, which evolved explicitly to exploit this fact

7. Misrule breeds rebellion; this is not a new idea. And yet the authorities still for the most part act as if drugs were themselves the cause of the problem

8. It's important for nerds to realize, too, that school is not life. School is a strange, artificial thing, half sterile and half feral. It's all-encompassing, like life, but it isn't the real thing. It's only temporary, and if you look, you can see beyond it even while you're still in it

o *The power of Galilean Relativity and stepping "above time" to gain perspective*

1. Hackers and Painters - Hackers are makers, like painters or architects or writers

    1. Hacking and painting have a lot in common. In fact, of all the different types of people I've known, hackers and painters are among the most alike. What hackers and painters have in common is that they're both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is make good things. They're not doing research per se though if in the course of trying to make good things they discover some new technique, so much the better.

2. For hackers, computers are just a medium of expression, as concrete is for architects or paint for painters.

3. There is a small but real chasm between architects and engineers: architects decide what to do, and engineers figure out how to do it. What and how should not be kept too separate. You're asking for trouble if you try to decide what to do without understanding how to do it.

4. The way to create something beautiful is often to make subtle tweaks to something that already exists, or to combine existing ideas in a slightly new way. This kind of work is hard to convey in a research paper

5. There is nothing so tempting as an easy test that kind of works

6. The only external test is time. Over time, beautiful things tend to thrive, and ugly things tend to get discarded. Unfortunately, the amounts of time involved can be longer than human lifetimes

7. I've found that the best sources of ideas are not the other fields that have the word "computer" in their names, but the other fields inhabited by makers. Painting has been a much richer source of ideas than the theory of computation

8. Programs should be malleable, allowing you to sketch roughly at first and debug over time

9. If you want to make money at some point, remember this, because this is one of the reasons startups win. Big companies want to decrease the standard deviation of design outcomes because they want to avoid disasters. But when you damp oscillations, you lose the high points as well as the low. This is not a problem for big companies, because they don't win by making great products. Big companies win by sucking less than other big companies. So if you can figure out a way to get

in a design war with a company big enough that its software is designed by product managers, they'll never be able to keep up with you. These opportunities are not easy to find, though. It's hard to engage a big company in a design war, just as it's hard to engage an opponent inside a castle in hand to hand combat. It would be pretty easy to write a better word processor than Microsoft Word, for example, but Microsoft, within the castle of their operating system monopoly, probably wouldn't even notice if you did. The place to fight design wars is in new markets, where no one has yet managed to establish any fortifications. That's where you can win big by taking the bold approach to design, and having the same people both design and implement the product. Microsoft themselves did this at the start. So did Apple. And Hewlett-Packard. I suspect almost every successful startup has.

10. One thing we can learn, or at least confirm, from the example of painting is how to learn to hack. You learn to paint mostly by doing it. Ditto for hacking. Most hackers don't learn to hack by taking college courses in programming. They learn to hack by writing programs of their own at age thirteen. Even in college classes, you learn to hack mostly by hacking. The other way to learn is from examples. For a painter, copying forces you to look closely at the way a painting is made. Writers do this too. Hackers, likewise, can learn to program by looking at good programs – not just at what they do, but the source code too. Another example we can take from painting is the way that paintings are created by gradual refinement. Paintings usually begin with a sketch. Gradually the details get filled in. But it is not merely a process of filling in. Sometimes the original plans turn out to be mistaken. Everyone by now presumably knows about the danger of premature optimization. I think we should be just as worried about premature design – deciding too early what a program should do. Hacking can also learn from painting not only how to manage our own work but how to work together. Painting has always utilized a master/apprentice model and worked on different parts of the painting. The right way to collaborate, I think, is to divide projects into sharply defined modules, each with a definite owner, and with interfaces between them that are as carefully designed and, if possible, as articulated as programming languages.

11. Hackers start with original work and get good whereas scientists start good and get original

12. This sounds like a paradox, but a great painting has to be better than it has to be. How hard one works on a painting doesn't depend at all on how closely one expects anyone to look at it. Be relentless. Relentlessness wins because, in the aggregate, unseen details become visible. All these unseen details combine to produce something that's just stunning, like a thousand barely audible voices all singing in tune

13. In hacking, like painting, work comes in cycles. Sometimes you get excited about some new project and you want to work sixteen hours a day on it. Other times nothing seems interesting. To do good work, you have to take these cycles into account, because they're affected by how you react to them. Sometimes, backing off can prevent ambition from stalling. It's a good idea to save some easy tasks for moments when you would otherwise stall

   - *Think this is a smart and unique way of implementing the Pomodoro technique. Doing deep work for 25 minutes, or until you feel* *self losing focus, and then relax and celebrate for a couple minutes and take care of some easy tasks*

14. Like painting, most software is intended for human audience. And so hackers, like painters, must have empathy to do really great work. You have to be able to see things from the user's point of view. It turns out that looking at things from other people's point of view is practically the secret of success. It doesn't necessarily mean being self-sacrificing. Far from it. Understanding how someone else sees things doesn't imply that you'll act in his interest; in some situations – in war, for example – you want to do exactly the opposite. Most makers make things for a human audience. And to engage an audience you have to understand what they need. Empathy is probably the single most important difference between a good hacker and a great one. One way to tell how good people are at empathy is to watch them explain a technical question to someone without a technical background
15. In most fields the great work is done early on. Over and over we see the same pattern. A new medium appears, and people are so excited about it that they explore most of its possibilities in the first couple generations
16. An example of applied empathy. At Viaweb if we couldn't decide between two alternatives, we'd ask, what would our competitors hate most? At one point a competitor added a feature to their software that was basically useless, but since it was one of the few they had that we didn't, they made much of it in the trade press. We could have tried to explain that the feature was useless, but we decided it would annoy our competitor more if we just implemented it ourselves, so we hacked together our own version that afternoon

1. What You Can't Say - How to think heretical thoughts and what to do with them

    1. It's the nature of fashion to be invisible, in the same way that the movement of the earth is invisible to all of us riding on it. What scares me is that there are moral fashions too.
    2. The conformist test – do you have any opinions that you would be reluctant to express in front of a group of your peers? Almost certainly, there is something wrong with you if you don't think things you don't dare say out loud
    3. What can't we say? One way to find these ideas is simply to look at things people do say and get in trouble for. We are of course looking for things we can't say which are true, or at least have enough chance of being true that the question should remain open…I suspect the statements that make people maddest are those they worry might be true
    4. Another test is to see what is "heresy" or at least today's equivalent such as "divisive" or "racially insensitive." Can also see what today's labels are, such as "sexist"
    5. Can also look to the past to see what used to be acceptable and is now unthinkable. Might not even have to look to the past but merely to different cultures to see what they think is acceptable that you don't. My hypothesis is that the side that's shocked is most likely to be the mistaken one

6. There are certain taboos which are universal, such as murder, but any idea that's considered harmless in a significant percentage of times and places, and yet taboo in ours, is a good candidate for something we're mistaken about
7. To launch a taboo, a group has to be poised halfway between weakness and power and be nervous. A confident group doesn't need taboos to protect it. It's not considered improper to make disparaging remarks about Americans, or the English. And yet a group has to be powerful enough to enforce a taboo.
8. There is nothing so wrong as the principles of the most recently defeated opponent
9. Great work tends to grow out of ideas that other shave overlooked, and no idea is so overlooked as one that's unthinkable. Training yourself to think unthinkable thoughts has advantages beyond the thoughts themselves. It's like stretching. When you stretch before running, you put your body into positions much more extreme than any it will assume during the run. If you think things so outside the box that they'd make people's hair stand on end, you'll have no trouble with the small trips outside the box that people call innovative.
10. Argue with idiots, and you become an idiot
11. When you find what you can't say, don't say it. Draw a sharp line between your thoughts and your speech. Inside your head, anything is allowed but keep it to yourself. The problem with keeping your thoughts secret, though, is that you lose the advantages of discussion. Talking about an idea leads to more ideas. So the optimal plan, if you can manage it, is to have a few trusted friends you can speak openly to. This is not just a way to develop ideas; it's also a good rule of thumb for choosing friends. The people you can say heretical things to without getting jumped on are also the most interesting to know.
12. To see fashion in your own time requires a conscious effort. Without time to give you distance, you have to create distance yourself. Instead of being part of the mob, stand as far away from it as you can and watch what it's doing. And pay especially close attention whenever an idea is being suppressed. And, it's not just the mob you need to learn to watch from a distance. You need to be able to watch your own thoughts from a distance. How can you see the wave, when you're the water? Always be questioning
    - *Antidote to Galilean Relativity*

1. Good Bad Attitude - Like Americans, hackers win by breaking rules

1. Those in authority tend to be annoyed by hackers' general attitude of disobedience. But that disobedience is a byproduct of the qualities that make them good programmers. They may laugh at the CEO when he talks in generic corporate newspeech but they also laugh at someone who tells them a certain problem can't be solved. Suppress one, and you suppress the others
2. The next generation of computer technology has often - perhaps more often than not - been developed by outsiders
    1. *As with most other fields, true breakthroughs taken an outsider's perspective and willingness to fail and look stupid that most entrenched incumbents or insiders aren't willing to take*

3. Hackers are unruly. That is the essence of hacking. And it is also the essence of American-ness. It is no accident that Silicon Valley is in America, and not France, or Germany, or England, or Japan. In those countries, people color inside the lines...It is greatly to America's advantage that it is a congenial atmosphere for the right sort of unruliness - that it is a home not just for the smart, but for smart-alecks. And hackers are invariably smart-alecks. If we had a national holiday, it would be April 1st.

4. Civil liberties make countries rich. If you made a graph of GNP per capita vs. civil liberties, you'd notice a definite trend. Could civil liberties really be a cause, rather than just an effect? I think so. I think a society in which people can do and say what they want will also tend to be one in which the most efficient solutions win, rather than those sponsored by the most influential people. Authoritarian countries become corrupt; corrupt countries become poor; and poor countries are weak.

1. The Other Road Ahead - Web-based software offers the biggest opportunity since the arrival of the microcomputer

1. Most people, most of the time, will take whatever choice requires least work – death before inconvenience

2. When you have competitors, “you can” means “you must” because if you don’t take advantage of this possibility, your competitors will

3. Turned negatives into positives and win/wins – ability to release fixes continuously lead to developers being genuinely interested in customer support because they would often learn about bugs and customers would feel important and even triumphant in finding these bugs

4. Focusing on “bottom half of the roster” – made customer support and developer relationships win/win as customer support’s honor was on the line when bringing in new bugs and they always knew which users features wanted more. This kind of relationship rather than a contentious, hierarchical one

5. Working to implement one idea gives you more ideas. Shelving ideas probably even inhibits new ideas. At Viaweb, often didn’t have future plans because plans are just another word for ideas on the shelf which they didn’t allow to happen. When they thought of good ideas, they implemented them

6. Paying attention is more important to reliability than moving slowly. Because he pays close attention, a Navy pilot can land a 40,000 lb. aircraft at 140 miles per hour on a pitching carrier deck, at night, more safely than the average teenager can cut a bagel

7. Complexity and bureaucracy scales exponentially with the size of the group. “We never had more to say at any one time [with 13 programmers] than we could say as we were walking to lunch.”

8. Software should do what users think it will. But you can’t have any idea what users will be thinking, believe me, until you watch them. By watching users, you can often tell when they’re in trouble and since the customer is always right, that’s a sign of something you need to fix.

9. Software is ideally suited for price discrimination as the marginal cost is close to zero
10. You'll sell more of something when it's easy to buy. Make it difficult to buy and give people a chance to second guess themselves and they will buy much less
11. There is always a tendency for rich customers to buy expensive solutions, even when cheap solutions are better, because the people offering expensive solutions can spend more to sell them. There is nothing you can do about this conundrum, so the best plan is to go for the smaller customers first. The rest will come in time
12. If a company wants to make a platform that startups will build on, they have to make it something that hackers themselves will want to use. That means it has to be inexpensive and well-designed
13. If not willing to disrupt or cannibalize yourself – "A cash cow can be a damned heavy monkey on your back"
14. "Just good enough" is often a powerful stepping stone to outweigh any cons or awkwardness in usage
15. Most hackers don't start their startup because they think they don't know anything about business and are afraid of competition. Neither of these fences have any "current" in them. There are only two things you have to know about business: build something users love and make more than you spend. If you can get these two right, you'll be ahead of most startups. You can figure out the rest as you go…Start by making something clean and simple that you would want to use yourself. The customer is always right but about different things; the least sophisticated users show you what you need to clarify and simplify and the most sophisticated tell you what features you need to add.
16. The standard to compare your software to is what it could be, not what your current competitors happen to have
17. Companies often wonder what to outsource and what not to. One possible answer: outsource any job that's not directly exposed to competitive pressure, because outsourcing it will thereby expose it to competitive pressure
18. There is always room for something new if the current options such enough. Make sure it works on all the free OSes first – new things start with their users

1. How to Make Wealth - The best way to get rich is to create wealth. And startups are the best way to do that
    1. If you wanted to get rich, how would you do it? I think your best bet would be to start or join a startup. Economically, you can think of a startup as a way to compress your whole working life into a few years. The advantage of creating wealth, rather than getting rich, is not just that it's more legitimate but that it's more straightforward. You just have to do something people want
    2. If you want to create wealth, it will help to understand what it is. Wealth is not the same thing as money. Wealth is as old as human history. Far older, in fact; ants have wealth. Money is a comparatively recent invention. Wealth is the fundamental thing. Wealth is stuff we want: food, clothes, houses, cars, gadgets, travel to interesting places, and so on. You can have wealth without money. Wealth is what

you want, not money. Money is simply a side effect of specialization. In a specialized society, most of the things you need, you can't make yourself and you need money to pay others to make it for you.

3. Pie Fallacy – there is not a fixed amount of wealth in this worlds. Again, money is not wealth which may be fixed in a given period of time
4. The top 5% of programmers probably write 99% of the good software
5. A company that could pay all its so straightforwardly would be enormously successful. Many employees would work harder if they could get paid for it. More importantly, such a company would attract people who wanted to work especially hard. It would crush its competitors
6. To get rich you need to get yourself in a situation with two things, measurement and leverage. You need to be in a position where your performance can be measured, or there is no way to get paid more by doing more. And you have to have leverage, in the sense that the decisions you make have a big effect. A good hint to the presence of leverage is the possibility of failure. However, you don't need to be a CEO or an athlete to have measurement and leverage. All you need to do is be part of a small group working on a hard problem.
7. A startup is not merely ten people, but ten people like you. Steve Jobs once said that the success or failure of a startup depends on the first ten employees. I agree. Being small is not, in itself what makes startups kick butt, but rather that small groups can be select. Startups have leverage because they make money by inventing new technology. What is technology? It's technique. It's the way we all do things. And when you discover a new way to do things, its value is multiplied by all the people who use it. Even giant corporations like McDonald's or Wal Mart can be said to create technology. A McDonald's franchise is controlled by rules so precise that it is practically a piece of software. Write once, run everywhere. Ditto for Wal-Mart. Sam Walton got rich not by being a retailer, but by designing a new kind of store
8. Use difficulty as a guide not just in selecting the overall aim of your company, but also at decision points along the way. At Viaweb, one of our rules of thumb was "run upstairs." Suppose you are a little, nimble guy being chased by a big, fat, bully. You open a door and find yourself a staircase. Do you go up or down? I say up. The bully can probably run downstairs as fast as you can. Going upstairs his bulk will be more of a disadvantage. Running upstairs is hard for you but even harder for him. What this meant in practice was that we deliberately sought hard problems, barriers to entry. Start by picking hard problems and then at every decision point, take the harder choice
9. I think this is a good plan for life in general. If you have two choices, choose the harder. If you're trying to decide whether to go out running or sit home and watch TV, go running. Probably the reason this trick works so well is that when you have two choices and one is harder, the only reason you're even considering the other is laziness. You know in the back of your mind what's the right thing to do, and this trick merely forces you to acknowledge it.
10. I think it's a good idea to get bought, if you can. Running a business is different from growing one. It is just as well to let a big company take over once you reach cruising altitude. It's also financially wiser, because selling allows you to diversify.

What would you think of a financial advisor who put all his client's assets into one volatile stock? Getting bought is an art. For potential acquires the most powerful motivator is the prospect that one of their competitors will buy you. This, as we found, causes CEOs to take red-eyes. The second biggest worry is that, if they don't buy you now, you'll continue to grow rapidly and will cost more to acquire later, or even become a competitor. In both cases, what it all comes down to is the number of users you have

11. One of the prime catalysts for industrialization was the spread of the rule of law. A necessary, if not sufficient, condition was that people who made fortunes be able to enjoy them in peace
12. Don't let a ruling class of warriors and politicians squash the entrepreneurs. The same recipe that makes individuals rich makes countries powerful. Let the nerds keep their lunch money and you rule the world

1. Mind the Gap - Could "unequal income distribution" be less of a problem than we think?
    1. When people care enough about something to do it well, those who do it best tend to be far better than everyone else. There's a huge gap between Leonardo and second-rate contemporaries like Borgognone Like chess or painting or writing novels, making money is a very specialized skill. But for some reason we treat this skill differently. No one complains when a few people surpass all the rest at playing chess or writing novels, but when a few people make more money than the rest, we get editorials saying this is wrong
    2. I think there are three reasons we treat making money as different: the misleading model of wealth we learn as children (confuse it with money, think there is a fixed amount, something that is distributed by authorities or parents and should be distributed equally, rather than something that has to be created and perhaps unequally); the disreputable way in which, till recently, most fortunes were accumulated (stolen); and the worry that great variations in income are somehow bad for society (may increase gap in income but decrease other gaps between rich and poor such as quality goods, quality of life but not brand). As far as I can tell, the first is mistaken, the second is outdated, and the third empirically false. Could it be that, in a modern democracy, variation in income is actually a sign of health? If technology doesn't bring about greater inequality it could be for three reasons: technical innovation has stopped, that the people who would create the most wealth aren't doing it or that they aren't getting paid for it
    3. To say a certain kind of work is underpaid is identical with saying that people want the wrong things
    4. You need rich people in your society not so much because in spending their money they create jobs, but because of what they have to do to get rich. I'm not talking about the trickle-down effect here. I'm not saying that if you let Henry Ford get rich, he'll hire you as a waiter at his next party. I'm saying that he'll make you a tractor to replace your horse
    5. Part of the reason this subject is so contentious is that some of the most vocal on the subject of wealth – university students, heirs, professors, politicians and journalists – have the least experience creating it

6. One of the biggest differences between the Daddy Model and reality is that in reality effort does not necessarily correlate with how much wealth it brings. Painting a house with a toothbrush should not bring you more money just because it is harder

1. A Plan for Spam - Till recently most experts thought spam filtering wouldn't work. This proposal changed their minds

    1. Began using probability filtering so that emails which entailed words associated with spam mail could get flagged and their frequency lowered over time

1. Taste for Makers - How do you make great things?

    1. For those of us who design things, these are not just theoretical questions, if there is such a thing as beauty, we need to be able to recognize it. We need good taste to make good things
    2. Once you start to examine the question, it’s surprising how much different fields’ ideas of beauty have in common. The same principles of good design crop up again and again. Good design is simple. Everything above simplicity is evasion. When you can’t deliver ornament, you have to deliver substance. Good design is timeless. Good design solves the right problem. Good design is suggestive. Good design is often slightly funny. I think it is because humor is related to strength. To have a sense of humor is to be strong: to keep one’s sense of humor is to shrug off misfortunes, and to lose one’s sense of humor is to be wounded by them. Good design is hard. If function is hard enough, form is forced to follow it, because there is no effort to spare for error. Wild animals are beautiful because they have hard lives. Good design looks easy. Like great athletes, great designers make it look easy. Mostly this is an illusion. In science and engineering, some of the greatest discoveries seem so simple that you say to yourself, I could have thought of that! In most fields the appearance of ease seems to come with practice. Perhaps what practice does is train your unconscious mind to handle tasks that used to require conscious thought. In some cases you literally train your body. An expert pianist can play notes faster than the brain can send signals to his hand. Likewise, an artist, after a while, can make visual perception flow in through his eye and out through his hand as automatically as someone tapping his foot to a beat. When people talk about being in “the zone,” I think what they mean is that the spinal cord has the situation under control. Your spinal cord is less hesitant, and it frees conscious thought for the hard problems. Good design uses symmetry (repetition and recursion). Nature uses them a lot, which is a good sign. The danger of symmetry, and repetition especially, is that it can be used as a substitute for thought. Good design resembles nature. It’s not so much that resembling nature is intrinsically good as that nature has had a long time to work on the problem. It’s not cheating to copy. Good design is redesign. It’s rare to get things right the first time. Experts expect to throw away some early work. Mistakes are natural. Instead of treating them as disasters, make them easy to acknowledge and easy to fix. Good design can copy. It is more important to be right than original. Unknowing imitation is almost a recipe

for bad design. The ambitious are not content to imitate. The second phase in the growth of taste is a conscious attempt at originality but are selfless and confident enough to take from anyone without feeling that their own vision will be lost in the process. Good design is often strange. The only style worth having is the one you can't help. And this is especially true for strangeness. Good design happens in chunks. Nothing is more powerful than a community of talented people working on related problems. Genes count for little by comparison: being a genetic Leonardo was not enough to compensate for having been born near Milan instead of Florence. Good design is often daring, teetering on the border of ostracism. This problem afflicts not just every era, but in some degree every field. Today's experimental error is tomorrow's new theory. If you want to discover great new things, then instead of turning a blind eye to the places where conventional wisdom and truth don't quite meet, you should pay particular attention to them.

3. Problems can be improved as well as solutions. In software, an intractable problem can usually be replaced by an equivalent one that's easy to solve. Physics progressed faster as the problem became predicting observable behavior, instead of reconciling it with scripture

4. As a practical matter, I think it's easier to see ugliness than to imagine beauty. Most of the people who've made beautiful things seem to have done it by fixing something that they thought ugly. Intolerance for ugliness is not in itself enough. You have to understand a field well before you develop a good nose for what needs fixing. You have to do your homework. But as you become an expert in a field, you'll start to hear little voices saying, What a hack! There must be a better way. Don't ignore those voices. Cultivate them. The recipe for great work is: very exacting taste, plus the ability to gratify it.

1. The Programming Languages Explained - What a programming language is and why they are a hot topic now
    1. The complete list of things a computer can do is machine language
    2. The more you have to say to get something done, the harder it will be to see bugs
    3. Open source software gives you the source code and the ability to modify it
    4. The biggest debate in language design is probably the one between those who think a language should prevent programmers from doing stupid things, and those who think programmers should be allowed to do whatever they want

1. The Hundred-Year Language - How will we program in a hundred years? Why not start now?

1. Staying close to the main evolutionary tree branches which will carry on is a useful heuristic for finding languages that will be good to program in now and to make better decisions about language design

o *Spend the most time on ideas, decisions, languages, etc. that will have the longest shelf life*

2. There's good waste, and bad waste. I'm interested in good waste – the kind where, by spending more, we can get simpler designs. How will we take advantage of the opportunities to waste cycles that we'll get from new, faster hardware? The

desire for speed is so deeply ingrained in us, with our puny computers, that it will take conscious effort to overcome it. In language design, we should be consciously seeking out situations where we can trade efficiency for even the smallest increase in convenience…Inefficient software isn't gross. What's gross is a language that makes programmers do needless work. Wasting programmer time is the true inefficiency, not wasting machine time. This will become ever more clear as computers get faster.

3. When you're working on language design, I think it is good to have such a target and to keep it consciously in mind. When you learn to drive, one of the principles they teach you is to align the car not by lining up the hood with the stripes painted on the road, but by aiming at some point in the distance. Even if all you care about is what happens in the next ten feet, this is the right answer. I think we can and should do the same with programming languages.

o *The importance of having a "true north" with morals, values, behavior, goals, "life vision," and more. If you have the big picture, end goal in mind, making decisions, saying no, taking advantage of opportunities becomes that much easier. You are able to "stand above time" and make optimal decisions when you have this end of the track vision and mindset*

1. Beating the Averages - For web-based applications you can use whatever language you want. So can your competitors

1. Graham chose to use Lisp in startup Viaweb because he thought that with Lisp he could get features done faster than competitors, could do things in their software that the competitors couldn't do, it would cost less so they could provide a better product for less money and still make a profit

o *The power of iteration, velocity, nimbleness, agility. Even if competitors introduced features they didn't currently have, they could quickly implement them*

2. In business, there is nothing more valuable than a technical advantage your competitors don't understand. In business, as in war, surprise is worth as much as force.

3. A startup should give its competitors as little information as possible. If they didn't know what language our software was written in, or didn't care, I wanted to keep it that way.

4. Programming languages are not merely technologies, but habits of mind as well, and nothing changes slower. The only programmers in a position to see all the differences in power between the various languages are those who understand the most powerful one. You can't trust the opinions of others, because of the " Blub paradox" – they're satisfied with whatever language they happen to use, because it dictates the way they think about programs. Can use technology that competitors don't understand like aikido – using their unwillingness to change and adapt against them

o *Programming languages are different, they are not just technology. They are half technology and half religion*

5. A suspicious programmer might begin to wonder if there was some correlation here. A big chunk of our code was doing things that are very hard to do

in other languages besides Lisp. The resulting software did things our competitors' software couldn't do. Maybe there was some kind of connection. I encourage you to follow that thread. There may be more to that old man hobbling along on his crutches than meets the eye. Lisp's power is multiplied by the fact that your competitors don't get it

6. A good tool to evaluate what competitors are doing is to view their job listings. The rest can be made pretty but they have to be real and authentic about their job listings or else they'll get the wrong candidates.

o *Graham used this technique over many years to see which competitors he had to worry about and which not to*

o *Graham's partner, Robert Morris, said later that he didn't have to be so secretive about Lisp because even if the competitors had known they were using Lisp, they wouldn't have understood why. "If they were that smart, they'd already be programming in Lisp."*

7. The old adage, "you can't tell a book by its cover," originated in the times when books were sold in plain cardboard covers, to be bound by each purchaser according to his taste. In those days, you couldn't tell a book by its cover. But publishing has advanced since then – present day publishers work hard to make the cover something you can tell a book by. Graham has spent enough time around technology that he can look at the "cover" of technology and know which to avoid

8. Reasons to dislike Java - really good products don't need to be promoted or over hyped; languages designed for other people to use have been bad and the good languages have been those that were designed for their own creators; ulterior motives are bad; no one loves it; people are forced to use it; it has too many cooks; it's bureaucratic; it's pseudo-hip; it's designed for large organizations; the wrong people like it; Sun's business model is being undermined; the DoD likes it

9. Revenge of the Nerds - In technology, "industry best practice" is a recipe for losing

10. The pointy-haired boss miraculously combines two qualities that are common by themselves, but rarely seen together: (a) he knows nothing whatsoever about technology, and (b) he has very strong opinions about it. He sticks to "industry best practice" to avoid responsibility and blame in case the company loses. However, it never gets you the best, merely the average

11. The main reason many ideas and practices are widespread is because they are comfortable

12. Lisp allows one to express the language in its own data structures which turns out to be a very powerful feature

o *Much like DNA which holds its own data and its own instructions*

13. One tends to magnify risks one doesn't truly understand

14. Often, what differentiates you or your company and gives you a competitive advantage seems like an anomaly to outsiders but is in fact the cause and effect of your moat

1. The Dream Language - A good programming language is one that lets hackers have their way with it

1. Designing a good programming language can be found by looking at hackers and learning what they want. Programming languages are for hackers, and a programming language is good as a programming language (rather than, say, an exercise in denotational semantics or compiler design) if and only if hackers like it
2. Good programming languages have to feature very powerful abstractions

2. Programming languages become popular if hackers use them because this tiny minority designs all the good software and their influence is such that the rest of the programmers will tend to use whatever language they use

3. Thinks that languages have to be popular to be good and that getting the first twenty users may be harder than going from twenty to a thousand. Best way to convince new users is through a Trojan horse – give people an application they want which happens to be written in a new language

4. A new language must have free implementation, a book, something to hack, must be brief, must be able to do what they want (can never guess all the ways a language will be used), dirty and clean – cleanly designed but let's hackers have their way with it, ability to write quick/throwaway programs but these tend to stick around and be the foundation for great programs (big things tend to be too scary to start but starting small and quick is easy and benefits from evolution), interactive, available, start up quickly, large libraries for manipulating strings, hackers will wait until they're more sure that the language will be around for some time and simple repetition often solves the problem (it's not when people notice you're there that they pay attention; it's when they notice you're still there), early adopters are demanding and help flush out problems quickly, the most important design is ability to redesign, no language designed by committee as they yield bad design and interfere with redesign

5. A friend of mine rarely does anything the first time someone asks him. He knows that people sometimes ask for things that they turn out not to want. To avoid wasting his time, he waits till the third or fourth time he's asked to do something; by then, whoever's asking him may be fairly annoyed, but at least they probably really do want whatever they're asking for.

6. There are typically two ways new technology gets introduced – the organic growth method (two guys in a garage) and the big-bang (VC-backed). Most of the garage boys are envious of the big-bang buys but more often than not it is the organic growth technology which yields better technology and richer founders

7. To write good software you must simultaneously keep two opposing ideas in your head. You need the young hacker's naïve faith in his abilities, and at the same time the veteran's skepticism. You have to be able to think how hard can it be? With one half of your brain while thinking it will never work with the other. The trick is to realize there's no real contradiction here. You want to be optimistic and skeptical about two different things. You have to be optimistic about the possibility of solving the problem, but skeptical about the value of whatever solution you have so far. People who do good work often think that whatever they're working on is no good. Others see what they've done and are full of wonder, but the creator is full of worry. This pattern is no coincidence: it is the worry that

made the work good. If you can keep hope and worry balanced, they will drive a project forward the same way your two legs drive a bicycle forward

- *Have heard a definition of intelligence as being able to hold two opposing thoughts in your head at the same time while still being able to function*
- *This constant worry and self-doubt seems ubiquitous in people trying to do great things. Love the connection that it is in fact this worrying which makes the work great, not in spite of it*

8. Users are a double-edged sword. They can help you improve your language, but they can also deter you from improving it. So choose your users carefully, and be slow to grow their number. Having users is like optimization: the wise course is to delay it. Also, as a general rule, you can at any given time get away with changing more than you think. Introducing a change is like pulling off a bandage: the pain is a memory almost as soon as you feel it.

1. Design and Research - Research has to be original. Design has to be good

    1. The difference between design and research seems to be a question of new versus good. Design doesn't have to be new, but it has to be good. Research doesn't have to be good, but it has to be new
    2. You have to design for the user, but you have to design what the user needs, not simply what he says he wants
    3. Must choose your group of users and almost always this group must include the designer himself
    4. There are two broad strategies: Worse is Better and the Hail Mary strategy. The Worse is Better gets a prototype in front of users as quickly as possible and slowly refines along the way. A prototype doesn't have to be just a mode; you can refine it into the finished product. I think you should always do this when you can. It lets you take advantage of new insights you have along the way. But perhaps even more important, it's good for morale and morale is key in design because it keeps you engaged. The Hail Mary tries to create the complete, finished, product in one long touchdown pass. As far as I know, this is a recipe for disaster
    5. Notice all this time I've been talking about "the designer." Design usually has to be under the control of a single person to be any good. And yet it seems to be possible for several people to collaborate on a research project. This seems to me one of the most interesting differences between research and design. Design by committee is a synonym for bad design. Good design requires a dictator. One reason is that good design has to be all of a piece. Design is not just for humans, but for individual humans. If a design represents an idea that fits in one person's head, then the idea will fit in the user's head too.
    6. There's nothing more valuable than the advice of someone whose judgment you trust

What I got out of it

1. So much to be gotten out of these essays even if you have no interest in either hacking or painting. A multi-disciplinary thinker who takes a different and interesting take on a variety of topics