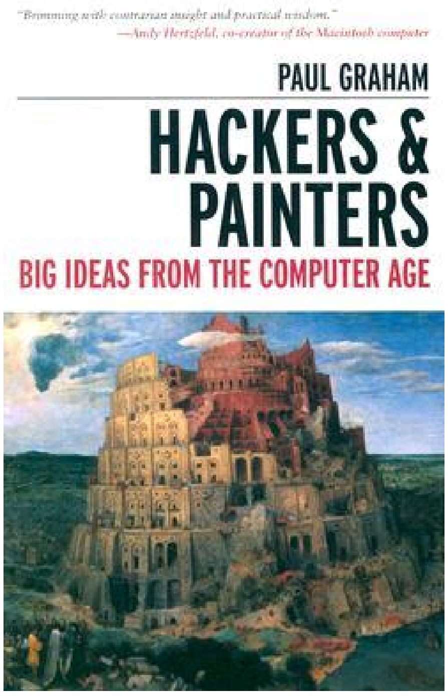
"Brimming with contrarian insight and practical wisdom."
—Andy Hertzfeld, co-creator of the Macintosh computer
PAUL GRAHAM
HACKERS & PAINTERS
BIG IDEAS FROM THE COMPUTER AGE

Hackers & Painters: Big Ideas from the Computer Age

Paul Graham , Allen Noren (Editor) , Matt Hutchinson (Editor)

Hackers & Painters: Big Ideas from the Computer Age

Paul Graham , Allen Noren (Editor) , Matt Hutchinson (Editor)

Hackers & Painters: Big Ideas from the Computer Age Paul Graham , Allen Noren (Editor) , Matt Hutchinson (Editor)
“The computer world is like an intellectual Wild West, in which you can shoot anyone you wish with your ideas, if you’re willing to risk the consequences.” —from *Hackers & Painters: Big Ideas from the Computer Age*, by Paul Graham

We are living in the computer age, in a world increasingly designed and engineered by computer programmers and software designers, by people who call themselves hackers. Who are these people, what motivates them, and why should you care?

Consider these facts: Everything around us is turning into computers. Your typewriter is gone, replaced by a computer. Your phone has turned into a computer. So has your camera. Soon your TV will. Your car was not only designed on computers, but has more processing power in it than a room-sized mainframe did in 1970. Letters, encyclopedias, newspapers, and even your local store are being replaced by the Internet.

Hackers & Painters: Big Ideas from the Computer Age, by Paul Graham, explains this world and the motivations of the people who occupy it. In clear, thoughtful prose that draws on illuminating historical examples, Graham takes readers on an unflinching exploration into what he calls “an intellectual Wild West.”

The ideas discussed in this book will have a powerful and lasting impact on how we think, how we work, how we develop technology, and how we live. Topics include the importance of beauty in software design, how to make wealth, heresy and free speech, the programming language renaissance, the open-source movement, digital design, internet startups, and more.

Hackers & Painters: Big Ideas from the Computer Age Details

Date	:	Published May 25th 2004 by O'Reilly Media (first published May 20th 2004)
ISBN	:	9780596006624
Author	:	Paul Graham , Allen Noren (Editor) , Matt Hutchinson (Editor)
Format	:	Hardcover 272 pages
Genre	:	Nonfiction, Computer Science, Programming, Business, Science, Technology

Download and Read Free Online Hackers & Painters: Big Ideas from the Computer Age Paul Graham , Allen Noren (Editor) , Matt Hutchinson (Editor)

From Reader Review Hackers & Painters: Big Ideas from the Computer Age for online ebook

Anusha Narasimhan says

A collection of essays that are thought provoking and insightful. Oh, and he makes nerds look super cool, so a big thumbs up from me. I recommend it to programmers and people interested in computer science.

Vijay says

I am a fan of PG's essays, so I was looking forward to reading this book. Unfortunately, it is just a collection of essays he has published online. If you have read the essays available on his website, you can safely skip this book.

In many of the essays, PG makes statements such as "The time to code a program depends mainly on its length.." which are ridiculous. I know he is trying to appeal to a wider audience, but staying stuff like this without anything to back it up is ridiculous. Some of his arguments go like this:

1. Controversial statement
2. Because of #1, controversial statement #2
3. Because of #3, our conclusion! Viola!

Ekaterina Kiseki says

Haven't finished the book. The man may be very good at his job, but he sucks at writing. The book looks like a compilation of cheap motivation posts with catchy titles. However it may amuse those who are completely unrelated to IT.

Vignesh says

Starting from random opinionated views on how the world works, to interesting correlations about art and science ending with a strong evangelism on the programming language lisp, Paul forces us to put our thinking cap on.

Michelle Tran says

The articles on technology were decent (not great), but it was hard not to facepalm every couple of pages on his articles about social commentary.

Ilya Ivanov says

Great book, not only for developers. You don't need to agree with all Paul's points (I certainly didn't) in order to appreciate courage and creativity of authors ideas.

Einar says

I had serious problems with this book. So Paul Graham is a successful Lisp hacker who made a lot of money from his start-up. Good for him. To be sure, this earns him some credibility in discussing languages and start-ups. Unfortunately, he takes it upon himself to extrapolate from this single data point to universal laws of what makes you successful. Moreover, he seems to think that his success as a geek entrepreneur somehow lends validity to whatever unsubstantiated thoughts, feelings and prejudices he may cook up, including some completely ridiculous views on the general superiority of geeks over regular people. The only reason so many of his readers seem to accept these views must be that he's preaching to the choir: certainly his geek audience would dearly like them to be true. His arcane and naive notions of art and aesthetics are too embarrassing to even discuss. Oh, and the smugness is just insufferable.

Cody Django says

Meh. This started out promising. While it may provide inspirational fodder for young, technological entrepreneurs, everyone else might soon find the tone obnoxious and constant extrapolation tedious.

Graham is at his best when he sticks to what he knows: programming and business technology. As such, the best chapter is "programming languages explained." This chapter held the most accessible explanation on language analytic that I've ever come across, and is a pleasure to read. Other chapters, such as "how to make wealth" might be of interest to someone with little understanding of economies of wealth, but to anyone else, it pretty much comes off as an Ayn Rand diatribe. The hacker/painter metaphor wears thin pretty fast; thankfully, he doesn't return to it.

Not a waste of a book, but I expected more. Great for a first year comp sci/arts minor.

It's easily possible to pick and choose what you would like to read, for this book is written as a collection of essays.

If your looking for something in the same vein, but far more rewarding --> "godel escher bach."

Ross Siegel says

Self indulgent, self-congratulatory, vague concepts expounding on platitudes & trivialities.
Paul Graham is a badass, no doubt, but this book can be skipped.

Viet Nguyen says

A collection of essays from Paul Graham, a programmer who strongly advocates LISP programming. This book provides deep insights into nerd's life, hacker, entrepreneurship, and, which I enjoy the most, programming language. Paul showed why LISP is "the most powerful programming language" by comparing it with many other programming language: C, Java, Perl, Python, Ruby.

4 star only because the info is somehow out of date.

Here is my quick notes:

Chap 1. Reading about nerds in school made something inside me resonate.

Chap 2. Hackers and Painters
- Hacking and Painting have a lot in common.
- "Computer Science" is not a good term
- mathematicians - people in between - hackers
- Universities and research labs force hackers to be scientists, and companies force them to be engineers
- One way to build great software is to start your own startup. but 2 problems: have to do so much besides write software, 2) not much overlap between the kind of software that makes money and the kind that's interesting to write.

ex: hacking programming languages doesn't pay as well as figuring out how to connect some company's legacy database to their web server.
--> Solution: day jobs. you have one kind of work you do for money, and another for love.
- hackers learn to hack by doing
- hackers start original, and get good, and scientists start good, and get original
- hackers can learn to program by looking at good programs - not just what they do, but at the source code.
- hackers should have empathy for users, readers.

Chap 3. What you can't say

Chap 4. Good Bad Attitude

Chap 5. The Other Road Ahead

Chap 6. How to make Wealth

Chap 10. Programming Language Explained

Chap 11. The Hundred-year Language

Chap 12. Beating the Averages

Chap 13. Revenge of the Nerds
- All languages are not equivalent
- Java, Perl, Python, Ruby

- Lisp - an effort to define a more convenient alternative to the Turing machine - John McCarthy
- 9 ideas of Lisp:
+ Conditionals
+ A function type
+ Recursion
+ Dynamic typing
+ Garbage-collection
+ Programs composed of expressions
+ A symbol type
+ A notation for code using trees of symbols and constants.
+ The whole language there all the time.

Chap 14. The Dream Language
- succintness is one place where statically typed language lose.

Chap 15. Design and Research

Saad El says

This book is a collection of essays by Paul Graham(co-founder of Y Combinator),
the essays are written in a really good style, the book is inspirational and thought provoking. It provide lots of insights on programming, startups, entrepreneurship, nerds, etc. The essays on programming languages are very interesting, Paul Graham clearly knows what he's talking about especially that he is co-creating a new Lisp dialect called Arc. But I didn't like it when he arguments on why Lisp is the most powerful programming language every now and then, it is probably because the essays weren't written with the idea of compiling them afterwards in a book, so you find that a lot of ideas are repeated throughout the book. The essays that I like the most were the ones where he recounts how he started Viaweb with Robert Morris before it got bought by Yahoo! a few years later, the story of Viaweb is such an inspiration.

Matt says

A fun-to-read mix of insight and ideology, Graham is someone we can learn from no matter which side of the box he's thinking on. His essay on nerds ("Why Nerds are Unpopular") is still a favorite, even while his essay on disparity of wealth ("Mind the Gap") is among the most unreflective apologies for anarcho-capitalism I've ever read.

I was, at least, inspired enough while reading Graham to put a few more thoughts together; those interested can find them here.

Shane says

The hackers and painters link is tenuous at best, and I didn't find much of the stuff in here revolutionary, but it was published in 2004 and I tend to agree with most of it. It seems to be mostly geared toward inspiring nerds to make more conscious decisions in the career, be it starting a business or otherwise even if it does claim to be aimed at anyone interested in learning about software and software systems. All that said, Graham is a decent writer. He adopts an authoritative tone which people might find annoying, but when it comes down to it these are opinion pieces and it's tough to expect otherwise.

Kevin Powe says

What I expected going in was interested parallels on the process of creating software versus other creative arts, and what Graham had learned across multiple disciplines. That I can dig.

What I got is a string of thinly justified essays that are lionising The Uber1337 Hacker as a misunderstood maverick agent for changing that is only being kept back by The Man.

Graham is a smart man - far smarter than me, and he's written a lot more software. But the tone of the book is grating, because:

a) he keeps coming back to that one point again and again
b) he never stoops to justifying his claims with a backing argument.

In Graham's view, the hacker is the central agent of change, of creating value, and while that may be true in his experience, it's a tremendously limited viewpoint, and he comes across as remarkably arrogant towards anything outside his experience. The biggest danger in this is people with half his intelligence justifying their own worldview via his writing.

I'd be a lot less annoyed with this book if Graham did himself a giant favour and didn't introduce his views on how economics interacts with society. He's obviously entitled to a viewpoint, but it's a remarkably cloistered one, and without justifying his opinion, it just comes off as another rich white guy wondering why people are griping instead of getting out there and Making Stuff (go get 'em tiger!)

The essays in here on programming and what he's learned from art are interesting - I'd love to read a whole book extrapolating on these points.

Max Nova says

Full review and highlights at https://books.max-nova.com/hackers-and-painters/

I was looking at my highlights for Paul Graham's "Hackers and Painters" and it seems like I basically highlighted the entire book. It's that good.

At its core, this is a book about how changes in technology (particularly computer tech) has changed economic and social realities... and the new breed of tech-savvy doers that these technological shifts have brought to the forefront of our society.

Graham begins at the beginning of the alpha-nerd's journey - middle school. He launches a withering salvo of criticism at the current educational system - a system which he fashions more of a prison than a temple of learning. He's a sharp critic of what he proclaims "the emptiness of school life" and he points out that "Misrule breeds rebellion" (in reference to troubled schools). Graham also contends that the total lack of real purpose in schools is the root of the crazy teenage drama that goes on in middle schools and high schools around the country.

He moves on to discussing the role of "makers" in society - from the eponymous hackers to painters. He makes a great observation - which is that while both of these professions involve creating things, painting has a far longer tradition of training and educating its practitioners. Graham notes that almost all great programmers are self-taught, but the lack of a good training regimen for programmers means that society misses out on a lot of potentially great hackers.

Graham touches on the often subversive, counter-authority, and contrarian culture of hackers - noting, "Whatever the reason, there seems a clear correlation between intelligence and willingness to consider shocking ideas." He goes on a bit of a (justified) rant against political correctness and moral fashions - noting that, "when people are bad at open mindedness, they don't know it. In fact they tend to think the opposite. Remember, it's the nature of fashion to be invisible. It wouldn't work otherwise"

He also emphasizes how goddam patriotic it is to be a hacker: "There is such a thing as American-ness. There's nothing like living abroad to teach you that. And if you want to know whether something will nurture or squash this quality, it would be hard to find a better focus group than hackers, because they come closest of any group I know to embodying it." He pulls in a great Jefferson quote too:

"When you read what the founding fathers had to say for themselves, they sound more like hackers. "The spirit of resistance to government," Jefferson wrote, "is so valuable on certain occasions, that I wish it always to be kept alive." Imagine an American president saying that today. Like the remarks of an outspoken old grandmother, the sayings of the the founding fathers have embarrassed generations of their less confident successors. They remind us where we come from. They remind us that it is the people who break rules that are the source of America's wealth and power."

In the next section of the book, Graham discusses the nature of writing code itself. He emphasizes the design and complexity of software - "designing web-based software is like designing a city rather than a building: as well as buildings you need roads, street signs, utilities, police and fire departments, and plans for both growth and various kinds of disasters." He notes that, "with the rise of industrialization there are fewer and fewer craftsmen. One of the biggest remaining groups is computer programmers"

Graham's next topic is on wealth creation - and why startups are so good at it. He claims that "I think every one who gets rich by their own efforts will be found to be in a situation with measurement and leverage. Everyone I can think of does: CEOs, movie stars, hedge fund managers, professional athletes. A good hint to the presence of leverage is the possibility of failure." He also puts forth some pretty bold historical analysis: "Understanding this may help to answer an important question: why Europe grew so powerful. Was it something about the geography of Europe? Was it that Europeans are somehow racially superior? Was it their religion? The answer (or at least the proximate cause) may be that the Europeans rode on the crest of a powerful new idea: allowing those who made a lot of money to keep it."

He pulls out a few great examples of technology fundamentally reshaping society, noting "But it was not till the Industrial Revolution that wealth creation definitively replaced corruption as the best way to get rich. In England, at least, corruption only became unfashionable (and in fact only started to be called "corruption") when there started to be other, faster ways to get rich... Technology had made it possible to create wealth faster than you could steal it. The prototypical rich man of the nineteenth century was not a courtier but an industrialist."

In regards to the increasing income gap, Graham says, "Will technology increase the gap between rich and poor? It will certainly increase the gap between the productive and the unproductive. That's the whole point of technology... Technology should increase the gap in income, but it seems to decrease other gaps. A hundred years ago, the rich led a different kind of life from ordinary people. They lived in houses full of servants, wore elaborately uncomfortable clothes, and travelled about in carriages drawn by teams of horses which themselves required their own houses and servants. Now, thanks to technology, the rich live more like the average person."

A great money quote from Graham is, "It's absolute poverty you want to avoid, not relative poverty. If, as the evidence so far implies, you have to have one or the other in your society, take relative poverty. 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."

The rest of the book is a rant on programming languages and a lot of love for the esoteric Lisp programming language. Probably not of general interest.

Overall though, this book really blew me away. Everything he says seems obvious in retrospect, but that's because he's a genius. The way he approaches this immense topic is totally unique among all the stuff I've read and Graham certainly has the credentials to back it up. Required reading for citizens of the 21st century.