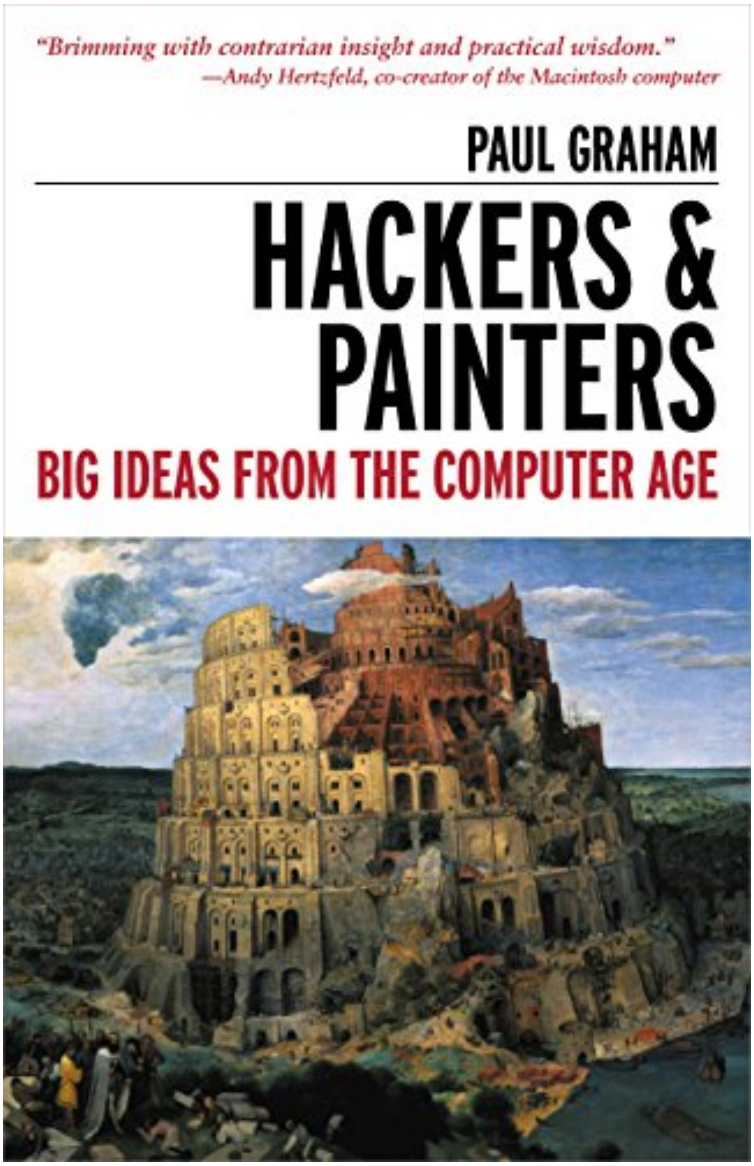
# Hackers Painters: Big Ideas from the Computer Age

*Von Paul Graham*
*DOC | *audiobook | ebooks | Download PDF | ePub*






Produktinformation -Verkaufsrang: #201267 in eBooksVerffentlicht am: 2004-05-18Erscheinungsdatum: 2008-07-14File Name: B0026OR2NQ | File size: 53.Mb

**Von Paul Graham : Hackers Painters: Big Ideas from the Computer Age** before purchasing it in order to gage whether or not it would be worth my time, and all praised Hackers Painters: Big Ideas from the Computer Age:

KundenrezensionenHilfreichste Kundenrezensionen8 von 8 Kunden fanden die folgende Rezension hilfreich. Ein Buch ber Computer-PhilosophieVon Jonas DohuHackers Painters ist ein Buch ber Computer-Philosophie. Es behandelt den Charakter von Hackern, das Wesen von Programmiersprachen und den vielleicht freiesten Weg zum

Reichtum: die eigene Firma. Man findet hier vor allem Denkanste und neue Betrachtungsweisen, deren Nhe zur Wahrheit sich nur im Laufe der Zeit herausstellen kann.Jedes Kapitel ist einer von fnfzehn eigenstndigen Aufstzen, die zum Teil auch online verfgbar sind. Der titelgebende Aufsatz ([...]) hat mich nach Jahren des verwirrten Nrgelns ber das Informatikstudium sehr angesprochen. Er beschftigt sich mit den hnlichkeiten von Hackern und Malern, wobei "Hacker" im ursprnglichen Sinne zu verstehen ist und im Grunde begeisterte Programmierer wie Kernel-Hacker meint - nicht Leute, die in andere Computer einbrechen und Schaden anrichten. Genaueres findet man in Eric S. Raymonds How To Become A Hacker ([...]Die fr mich interessantesten Aufstze:"Hackers and painters", also Hacker und Maler, werden wie Komponisten, Architekten und Autoren als Macher bezeichnet - als Menschen, die gute Dinge mit vorhandenen Mitteln erschaffen wollen. Whrend Forscher sich intensiv mit den vorhandenen Mitteln und Wegen beschftigen, um sie zu verstehen und zu verbessern, tun Macher dies hchstens, wenn das Vorhandene ihren Ansprchen nicht gengt.Daraus ergeben sich fr den Hacker Probleme. Whrend Architekten in ihrem Studium hufig praktisch ttig sind, kommt man als Informatik-Student so gut wie nie dazu, Programme nach eigenem Gutdnken zu erschaffen. Wissen ist fr einen Macher kein Selbstzweck, er braucht nur so viel, um seine Ziele verwirklichen zu knnen. Im Studium hat man allerdings nur begrenzten Freiraum in Bezug auf den Lernstoff. Mchte ich etwas ber relationale Datenbanken erfahren und gleichzeitig Fortschritt in meinem Studium, so muss ich auch den Rest der Vorlesung hren, fr den ich eigentlich gerade nicht empfnglich bin. Natrlich hat man so auch die Chance, auf bessere Alternativen zu stoen, aber wenn ich nur eine einzige Schraube in ein Brett drehen mchte, wre ich mit einem Schraubenzieher schneller fertig, als wenn ich vorher noch die Geschichte des Akkubohrers studieren msste - so toll dieses Werkzeug auch sein mag.Im Arbeitsleben sieht die Situation anders aus, aber nicht notwendigerweise besser. Wer mehr Freiheiten bei seiner Arbeit will, muss die Karriereleiter hochklettern und dafr am besten auch mehr als eine Ausbildung absolviert haben. Ist es dann endlich soweit, beschrnken sich die Freiheiten immer noch auf den Rahmen der Firma. Paul Graham schlgt als Ausweg aus diesem Dilemma den Day Job vor: eine Arbeit um Geld zu verdienen und eine andere aus Leidenschaft. Die Mehrheit der Open Source Programmierer drfte dies praktizieren."The other road ahead" spielt auf Bill Gates' Buch "The road ahead" an und erzhlt die Geschichte der Firma Viaweb, die Paul Graham 1995 mit gegrndet hat. Deren Software zur Verwaltung von Online-Shops sei eine der ersten Web-Anwendungen berhaupt gewesen. Graham schwrmt von den Vorteilen solch Server-basierter Software gegenber klassischen Desktop-Applikationen sowohl fr den Anwender als auch fr den Betreiber - insbesondere bei Startups. Ihm zufolge msse man fr die Grndung einer Firma nur zwei Ratschlge beachten: Baue etwas, das die Anwender lieben und nimm mehr ein als du ausgibst. Den Rest lerne man unterwegs."How to make wealth" ist eine interessante Auseinandersetzung mit der Vermehrung von Reichtum ("Dinge, die wir wollen") und Geld ("Ein Weg, Reichtum zu bewegen"). Der klassischen Denkweise zufolge ist das Erhhen des eigenen Reichtums lediglich durch das Senken des Reichtums an anderer Stelle mglich. Allerdings erhht etwa das Verwenden vorhandener Bretter und Ngel zum Bau eines Mbelstcks den eigenen Reichtum ganz ohne Opfer: ein benutzbarer Stuhl ist mehr wert als die dafr verwendeten Rohstoffe, ohne dass irgendjemand bei der Verarbeitung gelitten htte. Die Herstellung eines Stuhls macht den Erbauer also reicher, entweder um eine Sitzgelegenheit oder einen durch Verkauf erlsten Gegenwert."Mind the gap" behauptet, dass sich zwischen den Besten in einer Disziplin und dem Rest eine groe Lcke befinde. Normalerweise wird solch Talent bewundert, doch wenn sich dieses auf die Anhufung von Reichtum bezieht, betrachte man den Erfolg als unfair. Graham diskutiert drei Grnde dafr: das "Daddy model of wealth", demzufolge Reichtum nicht erzeugt werden kann, sondern von den Eltern kommt; der kriegerischen Art und Weise, mit der in der Geschichte Reichtum vergrert wurde und der Sorge, dass eine ungleiche Verteilung von Einkommen schlecht fr die Gesellschaft sei. Er betrachtet ersteres als Irrtum, das zweite als berholt und letzteres als empirisch falsch und begrndet dies ausfhrlich."Programming languages explained" behandelt die Geschichte und den Zweck von Programmiersprachen in ihren verschiedenen Ausprgungen. Graham revidiert die bekannte Aussage, dass alle Programmiersprachen gleich mchtig seien und spricht weitere Konfliktherde an, wie etwa statische gegenber dynamischer Programmierung und den Grad, mit dem eine Programmiersprache seine Benutzer vor Dummheiten schtzen will."Revenge of the nerds" geht tiefer auf die Entstehung von Lisp ein und verneint erneut die bekannte Ansicht ber die gleiche Mchtigkeit von Programmiersprachen. Manager wrden dies in der Regel nicht beachten, laut Graham sollten sie dies aber. Die interessanteste Aussage des Aufsatzes ist ein Zitat, nmlich "Greenspun's Tenth Rule: Any sufficiently complicated C or Fortran program contains an ad hoc informally-specified bug-ridden slow implementation of half of Common Lisp.""The dream language" untersucht die fr den Erfolg von Programmiersprachen verantwortlichen Faktoren. Graham beschftigt sich unter anderem mit den Aspekten Kompaktheit, Hackbarkeit, Wegwerf-Programmen, Befehls-Bibliotheken, Effizienz, Zeit und Redesign. Eine interessante Aussage ist, dass die Ansichten von Hackern und Sprachdesignern auseinandergehen. Wer sich also an die Entwicklung einer neuen Programmiersprache wagen mchte, findet hier mglicherweise ein paar gute Hinweise.Fr mich waren aufgrund moralischer Bedenken die Ansichten zum Thema Reichtum am interessantesten. Die Aussagen ber Startups besttigen mich in der Annahme, dass mir diese Arbeitsform unter den richtigen Umstnden besser gefallen kann als jeder normale Job. Neugierig auf Lisp war ich schon vor der Lektre, nun bin ich es noch etwas mehr - insbesondere aufgrund der Aussage, Ruby sei Lisp mit Syntax, denn Ruby gefllt mir sehr. Und ich bekomme zunehmend Lust, mich mit der Technologie hinter Programmiersprachen

zu beschftigen. Sollte dies in eigenen Sprachen mnden, finde ich hier ebenfalls guten Rat.9 von 9 Kunden fanden die folgende Rezension hilfreich. Mehr drin, als ich zunchst dachteVon L. W.Paul Graham stellt in "Hackers and Painters" mehrere Essays zusammen, die zum Teil in hnlicher Form auch auf seiner Homepage verffentlicht sind. Er beschreibt in sehr leicht zu lesendem, lockerem Stil verschiedene Themen, die zum Teil nur im weiteren Sinne mit dem Thema Computer zusammen hngen. Der rote Faden, der sich durch alle Essays zieht ist der Erfolg, der sich mit unkonventionellen Ideen und (Programmier-)Werkzeugen und bei harter Arbeit einstellt. Fr mich war der Band sehr unterhaltsame Urlaubslektre, die mir viele neue Einsichten vermittelt und meine Gedanken sehr beflgelt hat.5 von 5 Kunden fanden die folgende Rezension hilfreich. In unkonventionellen Bahnen denkenVon Stefan RoockIch habe gerade das Buch "Hackers and Painters" von Paul Graham durchgelesen. In dem Buch ist eine Reihe von Essays verffentlicht, die (fast?) alle auch online auf auf der Web-Site von Paul Graham verfgbar sind. Trotzdem lohnt sich das Buch, um es in der Bahn oder im Urlaub zu lesen.Die Essays decken verschiedene Themenbereiche ab begonnen vom typischen Hacker-Charakter ber Innovationen und Startups in der IT bis hin zum Programmiersprachenentwurf. Dabei vertritt Paul Graham hufig ungewhnliche Ansichten und Ideen und regt damit zum Denken auerhalb der gngigen Bahnen an (warum die nchste Webanwendung nicht in Lisp schreiben?).Mir hat das Buch jedenfalls sehr gut gefallen.

Kurzbeschreibung"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 GrahamWe 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.