Coders At Work Reflections On The Craft Of Programming

If you ally habit such a referred **Coders At Work Reflections On The Craft Of Programming** ebook that will pay for you worth, get the agreed best seller from us currently from several preferred authors. If you want to entertaining books, lots of novels, tale, jokes, and more fictions collections are then launched, from best seller to one of the most current released.

You may not be perplexed to enjoy every book collections Coders At Work Reflections On The Craft Of Programming that we will categorically offer. It is not on the costs. Its practically what you habit currently. This Coders At Work Reflections On The Craft Of Programming, as one of the most in action sellers here will definitely be in the middle of the best options to review.

The Constitution of Algorithms Florian Jaton 2021-04-27 A laboratory study that investigates how algorithms come into existence. Algorithms--often associated with the terms big data, machine learning, or artificial intelligence--underlie the technologies we use every day, and disputes over the consequences, actual or potential, of new algorithms arise regularly. In this book, Florian Jaton offers a new way to study computerized methods, providing an account of where algorithms come from and how they are constituted, investigating the practical activities by which algorithms are progressively assembled rather than what they may suggest or require once they are assembled.

The Problem with Software Adam Barr 2018-10-23 An industry insider explains why there is so much bad software—and why academia doesn't teach programmers what industry wants them to know. Why is software so prone to bugs? So vulnerable to viruses? Why are software products so often delayed, or even canceled? Is software development really hard, or are software developers just not that good at it? In The Problem with Software, Adam Barr examines the proliferation of bad software, explains what causes it, and offers some suggestions on how to improve the situation. For one thing, Barr points out, academia doesn't teach programmers what they actually need to know to do their jobs: how to work in a team to create code that works reliably and can be maintained by somebody other than the original authors. As the size and complexity of commercial software have grown, the gap between academic computer science and industry has widened. It's an open secret that there is little engineering in software engineering, which continues to rely not on codified scientific knowledge but on intuition and experience. Barr, who worked as a programmer for more than twenty years, describes how the industry has evolved, from the era of mainframes and Fortran to today's embrace of the cloud. He explains bugs and why software has so many of them, and why today's interconnected computers offer fertile ground for viruses and worms. The difference between good and bad software can be a single line of code, and Barr includes code to illustrate the consequences of seemingly inconsequential choices by programmers. Looking to the future, Barr writes that the best prospect for improving software engineering is the move to the cloud. When software is a service and not a product, companies will have more incentive to make it good rather than “good enough to ship."

100 Things You Don't Wanna Know about Coders at Work Joseph Finning 2013-02 In this book, we have hand-picked the most sophisticated, unanticipated, absorbing (if not at times crackpot!), original and musing book reviews of "Coders at Work: Reflections on the Craft of Programming." Don't say we didn't warn you: these reviews are known to shock with their unconventionality or intimacy. Some may be startled by their biting sincerity; others may be spellbound by their unbridled flights of fantasy. Don't buy this book if: 1. You don't have nerves of steel. 2. You expect to get pregnant in the next five minutes. 3. You've heard it all.

Softwareentwicklung von Kopf bis Fuss Dan Pilone 2008-07-15 Was lernen Sie mit diesem Buch? Haben Sie sich schon einmal gefragt, was es mit testgetriebener Entwicklung auf sich hat? Oder auf welcher Basis es die richtig guten Consultants schaffen, gewaltige Stundensätze zu kassieren? Vielleicht sind Sie auch gerade an dem Punkt, an dem Sie Ihre Builds automatisieren wollen, Ihren Code in eine Versionskontrolle füttern, einem Refactoring unterziehen oder mit ein paar Entwurfsmustern anreichern wollen. Egal: Wenn Sie mit diesem Buch fertig sind, werden Sie ganz selbstverständlich Ihre Burndown-Rate verfolgen, den Durchsatz Ihres Teams berücksichtigen und sich erfolgreich Ihren Weg durch Anforderungen, Entwurf, Entwicklung und Auslieferung iterieren. Wieso sieht dieses Buch so anders aus? Wir gehen davon aus, dass Ihre Zeit zu kostbar ist, um mit neuem Stoff zu kämpfen. Statt Sie mit Bleiwüstentexten langsam in den Schlaf zu wiegen, verwenden wir für Softwareentwicklung von Kopf bis Fuß ein visuell und inhaltlich abwechslungsreiches Format, das auf Grundlage neuster Forschungsergebnisse im Bereich der Kognitionswissenschaft und der Lerntheorie entwickelt wurde. Wir wissen nämlich, wie Ihr Gehirn arbeitet.

Effektiv C++ programmieren Scott Meyers 2011

Code und andere Gesetze des Cyberspace Lawrence Lessig 2001

Die Kunst der JavaScript-Programmierung Marijn Haverbeke 2012-08-13 Das Buch ist eine Einführung in JavaScript, die sich auf gute Programmiertechniken konzentriert. Der Autor lehrt den Leser, wie man die Eleganz und Präzision von JavaScript nutzt, um browserbasierte Anwendungen zu schreiben. Das Buch beginnt mit den Grundlagen der Programmierung - Variablen, Kontrollstrukturen, Funktionen und Datenstrukturen -, dann geht es auf komplexere Themen ein, wie die funktionale und objektorientierte Programmierung, reguläre Ausdrücke und Browser-Events. Unterstützt von verständlichen Beispielen wird der Leser rasch die Sprache des Web fließend 'sprechen' können.

Unraveling Software Maintenance and Evolution Ervin Varga 2018-01-29 Software maintenance work is often considered a dauntingly rigid activity – this book proves the opposite: it demands high levels of creativity and thinking outside the box. Highlighting the creative aspects of software maintenance and combining analytical and systems thinking in a holistic manner, the book motivates readers not to blithely follow the beaten tracks of “technical rationality”. It delivers the content in a pragmatic fashion using case studies which are woven into long running story lines. The book is organized in four parts, which can be read in any order, except for the first chapter, which introduces software maintenance and evolution and presents a number of case studies of software failures. The “Introduction to Key Concepts” briefly introduces the major elements of software maintenance by highlighting various core concepts that are vital in order to see the forest for the trees. Each such concept is illustrated with a worked example. Next, the “Forward Engineering” part debunks the myth that being fast and successful during initial development is all that matters. To this end, two categories of forward engineering are considered: an inept initial project with a multitude of hard evolutionary phases and an effective initial project with multiple straightforward future increments. “Reengineering and Reverse Engineering” shows the difficulties of dealing with a typical legacy system, and tackles tasks such as retrofitting tests, documenting a system, restructuring a system to make it amenable for further improvements, etc. Lastly, the “DevOps” section focuses on the importance and benefits of crossing the development versus operation chasm and demonstrates how the DevOps paradigm can turn a loosely coupled design into a loosely deployable solution. The book is a valuable resource for readers familiar with the Java programming language, and with a basic understanding and/or experience of software construction and testing. Packed with examples for every elaborated concept, it offers complementary material for existing courses and is useful for students and professionals alike.

MATLAB Guide, Third Edition Desmond J. Higham 2016-12-27 MATLAB is an interactive system for numerical computation that is widely used for teaching and research in industry and academia. It provides a modern programming language and problem solving environment, with powerful data structures, customizable graphics, and easy-to-use editing and debugging tools. This third edition of MATLAB Guide completely revises and updates the best-selling second edition and is more than 30 percent longer. The book remains a lively, concise introduction to the most popular and important features of MATLAB and the Symbolic Math Toolbox. Key features are a tutorial in Chapter 1 that gives a hands-on overview of MATLAB; a thorough treatment of MATLAB mathematics, including the linear algebra and numerical analysis functions and the differential equation solvers; and a web page at http://www.siam.org/books/ot150 that provides example program files, updates, and links to MATLAB resources. The new edition contains color figures throughout; includes pithy discussions of related topics in new ?Asides" boxes that augment the text; has new chapters on the Parallel Computing Toolbox, object-oriented programming, graphs, and large data sets; covers important new MATLAB data types such as categorical arrays, string arrays, tall arrays, tables, and timetables; contains more on MATLAB workflow, including the Live Editor and unit tests; and fully reflects major updates to the MATLAB graphics system. This book is suitable for both beginners and more experienced users, including students, researchers, and practitioners.

Hacker-Manifest McKenzie Wark 2005

Projekt Phoenix Gene Kim 2015-08-22 Bill Palmer wird überraschend zum Bereichsleiter der IT-Abteilung eines Autoteileherstellers befördert und muss nun eine Katastrophe nach der anderen bekämpfen. Gleichzeitig läuft ein wichtiges Softwareprojekt und die Wirtschaftsprüfer sind auch im Haus. Schnell wird klar, dass »mehr Arbeiten, mehr Prioritäten setzen, mehr Disziplin« nicht hilft. Das ganze System funktioniert einfach nicht, eine immer schneller werdende Abwärtsspirale führt dazu, dass das Unternehmen kurz vor dem Aus steht. Zusammen mit einem weitsichtigen Aufsichtsratsmitglied fängt Bill Palmer an, das System umzustellen. Er organisiert Kommunikation und Workflow zwischen Abteilungen neu, entdeckt und entschärft Flaschenhälse und stimmt sich mit dem Management besser ab. Er schafft es damit, das Ruder herumzureißen. Das Buch zeigt, wie neue Ideen und Strategien der DevOps-Bewegung konkret umgesetzt werden können und zum Erfolg führen - und liest sich dabei wie ein guter Wirtschaftskrimi!

Think Like a Programmer - Deutsche Ausgabe V. Anton Spraul 2013-04-05 Typische Programmieraufgaben kreativ lösen am Beispiel von C++ Von der Aufgabe zur Lösung – so gehen Sie vor Probleme analysieren und schrittweise bearbeiten Systematisches Vorgehen lernen und anwenden Aus dem Inhalt: Strategien zur Problemlösung Eingabeverarbeitung Statusverfolgung Arrays Zeiger und dynamische Speicherverwaltung Klassen Rekursion Wiederverwendung von Code Rekursive und iterative Programmierung Denken wie ein Programmierer Die Herausforderung beim Programmieren besteht nicht im Erlernen der Syntax einer bestimmten Sprache, sondern in der Fähigkeit, auf kreative Art Probleme zu lösen. In diesem einzigartigen Buch widmet sich der Autor V. Anton Spraul genau jenen Fähigkeiten, die in normalen Lehrbüchern eher nicht behandelt werden: die Fähigkeit, wie ein Programmierer zu denken und Aufgaben zu lösen. In den einzelnen Kapiteln behandelt er jeweils verschiedene Programmierkonzepte wie beispielsweise Klassen, Zeiger und Rekursion, und fordert den Leser mit erweiterbaren Übungen zur praktischen Anwendung des Gelernten auf. Sie lernen unter anderem: Probleme in diskrete Einzelteile zerlegen, die sich leichter lösen lassen Funktionen, Klassen und Bibliotheken möglichst effizient nutzen und wiederholt verwenden die perfekte Datenstruktur für eine Aufgabenstellung auswählen anspruchsvollere Programmiertechniken wie Rekursion und dynamischen Speicher einsetzen Ihre Gendanken ordnen und Strategien entwickeln, um bestimmte Problemkategorien in Angriff zu nehmen Die Beispiele im Buch werden mit C++ gelöst, die dargestellten kreativen Problemlösungskonzepte gehen aber weit über die einzelnen Programmiersprachen und oft sogar über den Bereich der Informatik hinaus. Denn wie die fähigsten Programmierer wissen, handelt es sich beim Schreiben herausragender Quelltexte um kreative Kunst und der erste Schritt auf dem Weg zum eigenen Meisterwerk besteht darin, wie ein Programmierer zu denken. Über den Autor: V. Anton Spraul hat über 15 Jahre lang Vorlesungen über die Grundlagen der Programmierung und Informatik gehalten. In diesem Buch fasst er die von ihm dabei perfektionierten Verfahren zusammen. Er ist auch Autor von »Computer Science Made Simple«.

The Power of Ruby Rachel Keranen 2017-12-15 Although Yukihiro Matsumoto released Ruby in 1995, it would take more than a decade for the programming language to achieve widespread adoption. Today, Ruby is one of the fastest growing coding languages, and it is behind major sites like Hulu and Twitter. This book describes the history of the language, the relationship between Ruby and Ruby on Rails, and why Ruby is an ideal first language for new programmers, complete with tips on how readers can get started.

Coders Clive Thompson 2019-03-26 Hello, world. Facebook's algorithms shaping the news. Self-driving cars roaming the streets. Revolution on Twitter and romance on Tinder. We live in a world constructed of code--and coders are the ones who built it for us. From acclaimed tech writer Clive Thompson comes a brilliant anthropological reckoning with the most powerful tribe in the world today, computer programmers, in a book that interrogates who they are, how they think, what qualifies as greatness in their world, and what should give us pause. They are the most quietly influential people on the planet, and Coders shines a light on their culture. In pop culture and media, the people who create the code that rules our world are regularly portrayed in hackneyed, simplified terms, as ciphers in hoodies. Thompson goes far deeper, dramatizing the psychology of the invisible architects of the culture, exploring their passions and their values, as well as their messy history. In nuanced portraits, Coders takes us close to some of the great programmers of our time, including the creators of Facebook's News Feed, Instagram, Google's cutting-edge AI, and more. Speaking to everyone from revered "10X" elites to neophytes, back-end engineers and front-end designers, Thompson explores the distinctive psychology of this vocation--which combines a love of logic, an obsession with efficiency, the joy of puzzle-solving, and a superhuman tolerance for mind-bending frustration. Along the way, Coders thoughtfully ponders the morality and politics of code, including its implications for civic life and the economy. Programmers shape our everyday behavior: When they make something easy to do, we do more of it. When they make it hard or impossible, we do less of it. Thompson wrestles with the major controversies of our era, from the "disruption" fetish of Silicon Valley to the struggle for inclusion by marginalized groups. In his accessible, erudite style, Thompson unpacks the surprising history of the field, beginning with the first coders -- brilliant and pioneering women, who, despite crafting some of the earliest personal computers and programming languages, were later written out of history. Coders introduces modern crypto-hackers fighting for your privacy, AI engineers building eerie new forms of machine cognition, teenage girls losing sleep at 24/7 hackathons, and unemployed Kentucky coal-miners learning a new career. At the same time, the book deftly illustrates how programming has become a marvelous new art form--a source of delight and creativity, not merely danger. To get as close to his subject as possible, Thompson picks up the thread of his own long-abandoned coding skills as he reckons, in his signature, highly personal style, with what superb programming looks like. To understand the world today, we need to understand code and its consequences. With Coders, Thompson gives a definitive look into the heart of the machine.

Code Complete - Deutsche Ausgabe Steve McConnell 2005-01 Dieses Buch ist die deutsche Ubersetzung eines Klassikers der Programmierliteratur von Steve McConnell. Seine mit vielen Preisen ausgezeichneten Bucher helfen Programmierern seit Jahren, besseren und effizienteren Code zu schreiben. Das Geheimnis dieses Buches liegt in der Art, wie der Autor das vorhandene Wissen uber Programmiertechniken aus wissenschaftlichen Quellen mit den Erfahrungen aus der taglichen praktischen Arbeit am Code zusammenfuhrt und daraus die wesentlichen Grundvoraussetzungen der Softwareentwicklung und die effektivsten Arbeitstechniken ableitet. Verstandliche Beispiele und klare Anleitungen vermitteln dem Leser dieses Wissen auf unkomplizierte Weise. Dieses Buch informiert und stimuliert, ganz gleich, wie viel sie bereits uber Programmierung wissen, welche Entwicklungsumgebung und Sprache sie bevorzugen und welche Arten von Anwendungen sie normalerweise programmieren.

Contingent Computation M. Beatrice Fazi 2018-11-15 Contingent Computation offers a new theoretical perspective through which we can engage philosophically with computing. The book proves that aesthetics is a viable mode of investigating contemporary computational systems.

Coders at Work Peter Seibel 2009-09-16 Peter Seibel interviews 15 of the most interesting computer programmers alive today in Coders at Work, offering a companion volume to Apress’s highly acclaimed best-seller Founders at Work by Jessica Livingston. As the words “at work” suggest, Peter Seibel focuses on how his interviewees tackle the day-to-day work of programming, while revealing much more, like how they became great programmers, how they recognize programming talent in others, and what kinds of problems they find most interesting. Hundreds of people have suggested names of programmers to interview on the Coders at Work web site: www.codersatwork.com. The complete list was 284 names. Having digested everyone’s feedback, we selected 15 folks who’ve been kind enough to agree to be interviewed: Frances Allen: Pioneer in optimizing compilers, first woman to win the Turing Award (2006) and first female IBM fellow Joe Armstrong: Inventor of Erlang Joshua Bloch: Author of the Java collections framework, now at Google Bernie Cosell: One of the main software guys behind the original ARPANET IMPs and a master debugger Douglas Crockford: JSON founder, JavaScript architect at Yahoo! L. Peter Deutsch: Author of Ghostscript, implementer of Smalltalk-80 at Xerox PARC and Lisp 1.5 on PDP-1 Brendan Eich: Inventor of JavaScript, CTO of the Mozilla Corporation Brad Fitzpatrick: Writer of LiveJournal, OpenID, memcached, and Perlbal Dan Ingalls: Smalltalk implementor and designer Simon Peyton Jones: Coinventor of Haskell and lead designer of Glasgow Haskell Compiler Donald Knuth: Author of The Art of Computer Programming and creator of TeX Peter Norvig: Director of Research at Google and author of the standard text on AI Guy Steele: Coinventor of Scheme and part of the Common Lisp Gang of Five, currently working on Fortress Ken Thompson: Inventor of UNIX Jamie Zawinski: Author of XEmacs and early Netscape/Mozilla hacker

Python Crashkurs Eric Matthes 2017-04-19 "Python Crashkurs" ist eine kompakte und gründliche Einführung, die es Ihnen nach kurzer Zeit ermöglicht, Python-Programme zu schreiben, die für Sie Probleme lösen oder Ihnen erlauben, Aufgaben mit dem Computer zu erledigen. In der ersten Hälfte des Buches werden Sie mit grundlegenden Programmierkonzepten wie Listen, Wörterbücher, Klassen und Schleifen vertraut gemacht. Sie erlernen das Schreiben von sauberem und lesbarem Code mit Übungen zu jedem Thema. Sie erfahren auch, wie Sie Ihre Programme interaktiv machen und Ihren Code testen, bevor Sie ihn einem Projekt hinzufügen. Danach werden Sie Ihr neues Wissen in drei komplexen Projekten in die Praxis umsetzen: ein durch "Space Invaders" inspiriertes Arcade-Spiel, eine Datenvisualisierung mit Pythons superpraktischen Bibliotheken und eine einfache Web-App, die Sie online bereitstellen können. Während der Arbeit mit dem "Python Crashkurs" lernen Sie, wie Sie: - leistungsstarke Python-Bibliotheken und Tools richtig einsetzen – einschließlich matplotlib, NumPy und Pygal - 2D-Spiele programmieren, die auf Tastendrücke und Mausklicks reagieren, und die schwieriger werden, je weiter das Spiel fortschreitet - mit Daten arbeiten, um interaktive Visualisierungen zu generieren - Web-Apps erstellen und anpassen können, um diese sicher online zu deployen - mit Fehlern umgehen, die häufig beim Programmieren auftreten Dieses Buch wird Ihnen effektiv helfen, Python zu erlernen und eigene Programme damit zu entwickeln. Warum länger warten? Fangen Sie an!

Geheimnisse eines JavaScript-Ninjas Resig/Bibeault 2014-02-24 Bewährte Methoden bei der Entwicklung einer JavaScript-Bibliothek Anspruchsvolle Features von JavaScript Ausführliche Beschreibung browserübergreifender Programmierung Aus dem Inhalt: Assertionen und Debugging Funktionen und Objekte Closures Objektorientierung und Prototypen Reguläre Ausdrücke Umgang mit Threads und Timern Codeauswertung zur Laufzeit Die with-Anweisung Cross-Browser-Strategien Attribute, Eigenschaften und CSS Umgang mit Ereignissen Manipulation des DOMs CSS-Selector-Engines Anwendungen und Bibliotheken aus der richtigen Perspektive betrachtet Modernes JavaScript-Design Problemlösungen für die browserübergreifende Entwicklung Das Web ist heute ohne JavaScript undenkbar, doch seit der Entstehung dieser Scriptsprache hat sich einiges getan. Dieses Buch stellt moderne JavaScript-Konzepte vor, die für alle Webentwickler von Nutzen sind, die Ajax und JavaScript-Bibliotheken für interaktive Webseiten einsetzen. Der JavaScript-Experte John Resig, Autor der bekannten jQuery-Bibliothek, vermittelt im Buch das Insiderwissen der besten JavaScript-Programmierer. Das Buch richtet sich an fortgeschrittene Anfänger und weist dem Leser den Weg zur Programmierung moderner JavaScript-Anwendungen in drei Stufen: Entwurf, Entwicklung sowie Pflege und Wartung des Codes. Zunächst wird die Grundlage fortgeschrittenen JavaScript-Wissens gelegt. Danach lernt der Leser den Aufbau einer JavaScript-Bibliothek kennen: Hier werden die vielfältigen Aufgaben erläutert sowie Entwicklungsstrategien und Lösungsansätze aus der Praxis vorgestellt. Und natürlich werden auch die Vorgehensweisen zur Erstellung zukunftssicheren Codes thematisiert. Das Buch versetzt den Leser in die Lage, ausgezeichnete JavaScript-Anwendungen zu programmieren, eigene Bibliotheken zu schreiben und die verfügbaren JavaScript-Bibliotheken optimal zu verwenden. Über die Autoren: John Resig ist anerkannter JavaScript-Experte und Autor der JavaScript-Bibliothek jQuery. Bear Bibeault ist Webentwickler und Co-Autor von drei weiteren Büchern. Von zwei Meistern ihrer Zunft: über die Kunst, effektives browserübergreifendes JavaScript zu erschaffen. Glenn Stokol, Oracle Corporation Ganz nach der jQuery-Devise "Weniger Code schreiben, mehr erreichen". André Roberge, Universität Sainte-Anne Spannende und originelle Techniken. Scott Sauyet, Four Winds Software Wenn Sie dieses Buch gelesen haben, werden Sie nie wieder blindlings ein Codefragment übernehmen und sich fragen, wie es funktioniert – sondern verstehen, warum es funktioniert. Joe Litton, Collaborative Software Developer, JoeLitton.net Bringt Ihr JavaScript auf meisterliches Niveau. Christopher Haupt, greenstack.com

Scythe – Das Vermächtnis der Ältesten Neal Shusterman 2019-11-27 Endlich: das spektakuläre Finale der großen »Scythe«-Trilogie Drei Jahre sind vergangen, seit mit Scythe Goddard ein Scythe der neuen Ordnung die Macht ergriffen hat, und seit der Thunderhead verstummt ist – für alle Menschen, bis auf Grayson Tolliver. Gibt es Hoffnung auf ein Wiedersehen mit Citra und Rowan und auf ein wirkliches Happy End in der scheinbar perfekten Welt? Stell dir eine Welt vor, in der Armut, Krankheit und Tod besiegt sind. Aber auch in dieser perfekten Welt müssen Menschen sterben. Die Entscheidung über Leben und Tod treffen die Scythe: sie allein entscheiden, wer sterben muss. Und nicht alle Scythe haben halten sich an alten Regeln ...

Sieben Wochen, sieben Sprachen (Prags) Bruce A. Tate 2011-06-30 Mit diesen sieben Sprachen erkunden Sie die wichtigsten Programmiermodelle unserer Zeit. Lernen Sie die dynamische Typisierung kennen, die Ruby, Python und Perl so flexibel und verlockend macht. Lernen Sie das Prototyp-System verstehen, das das Herzstück von JavaScript bildet. Erfahren Sie, wie das Pattern Matching in Prolog die Entwicklung von Scala und Erlang beeinflusst hat. Entdecken Sie, wie sich die rein funktionale Programmierung in Haskell von der Lisp-Sprachfamilie, inklusive Clojure, unterscheidet. Erkunden Sie die parallelen Techniken, die das Rückgrat der nächsten Generation von Internet-Anwendungen bilden werden. Finden Sie heraus, wie man Erlangs "Lass es abstürzen"-Philosophie zum Aufbau fehlertoleranter Systeme nutzt. Lernen Sie das Aktor-Modell kennen, das das parallele Design bei Io und Scala bestimmt. Entdecken Sie, wie Clojure die Versionierung nutzt, um einige der schwierigsten Probleme der Nebenläufigkeit zu lösen. Hier finden Sie alles in einem Buch. Nutzen Sie die Konzepte einer Sprache, um kreative Lösungen in einer anderen Programmiersprache zu finden – oder entdecken Sie einfach eine Sprache, die Sie bisher nicht kannten. Man kann nie wissen – vielleicht wird sie sogar eines ihrer neuen Lieblingswerkzeuge.

Effektives Arbeiten mit Legacy Code Michael C. Feathers 2020-11-04 Können Sie Ihren Code leicht ändern? Können Sie fast unmittelbar Feedback bekommen, wenn Sie ihn ändern? Verstehen Sie ihn? Wenn Sie eine dieser Fragen mit nein beantworten, arbeiten Sie mit Legacy Code, der Geld und wertvolle Entwicklungszeit kostet. Michael Feathers erläutert in diesem Buch Strategien für den gesamten Entwicklungsprozess, um effizient mit großen, ungetesteten Code-Basen zu arbeiten. Dabei greift er auf erprobtes Material zurück, das er für seine angesehenen Object-Mentor-Seminare entwickelt hat. Damit hat er bereits zahlreichen Entwicklern, technischen Managern und Testern geholfen, ihre Legacy-Systeme unter Kontrolle zu bringen. Darüber hinaus finden Sie auch einen Katalog mit 24 Techniken zur Aufhebung von Dependencies, die Ihnen zeigen, wie Sie isoliert mit Programmelementen arbeiten und Code sicherer ändern können.

Implementation Patterns - Studentenausgabe Kent Beck 2010

Coders at Work Peter Seibel 2011

Hacking in the Humanities Aaron Mauro 2022-05-05 What would it take to hack a human? How exploitable are we? In the cybersecurity industry, professionals know that the weakest component of any system sits between the chair and the keyboard. This book looks to speculative fiction, cyberpunk and the digital humanities to bring a human - and humanistic - perspective to the issue of cybersecurity. It argues that through these stories we are able to predict the future political, cultural, and social realities emerging from technological change. Making the case for a security-minded humanities education, this book examines pressing issues of data security, privacy, social engineering and more, illustrating how the humanities offer the critical, technical, and ethical insights needed to oppose the normalization of surveillance, disinformation, and coercion. Within this counter-cultural approach to technology, this book offers a model of activism to intervene and meaningfully resist government and corporate oversight online. In doing so, it argues for a wider notion of literacy, which includes the ability to write and fight the computer code that shapes our lives.

Python von Kopf bis Fuß Paul Barry 2017-08-10 Was lernen Sie in diesem Buch? Haben Sie sich schon einmal gewünscht, Sie könnten mit nur einem Buch Python richtig lernen? Mit Python von Kopf bis Fuß schaffen Sie es! Durch die ausgefeilte Von-Kopf-bis-Fuß-Didaktik, die viel mehr als die bloße Syntax und typische How-to-Erklärungen bietet, wird es sogar zum Vergnügen. Python-Grundlagen wie Datenstrukturen und Funktionen verstehen Sie hier schnell, und dann geht es auch schon weiter: Sie programmieren Ihre eigene Web-App, erkunden Datenbank-Management, Ausnahmebehandlung und die Verarbeitung von Daten. Da Python häufig im Data-Science-Umfeld eingesetzt wird, haben in der 2. Auflage diejenigen Techniken ein stärkeres Gewicht bekommen, die in der Welt der Big Data genutzt werden. Wieso sieht dieses Buch so anders aus? In diesem Buch sind die neuesten Erkenntnisse der Kognitionswissenschaft und der Lerntheorie eingeflossen, um Ihnen das Lernen so einfach wie möglich zu machen. Statt einschläfernder Bleiwüsten verwendet dieses Buch eine Vielzahl von Abbildungen und Textstilen, die Ihnen das Wissen direkt ins Hirn spielen – und zwar so, dass es sitzt.

Solid Code Donis Marshall 2009

Grounded theory Barney G. Glaser 2010 Der Klassiker über den qualitativen Forschungsansatz der "Grounded Theory", geschrieben von den Entwicklern dieser Methode "Obwohl unser Buch sich in erster Linie an Soziologen richtet, glauben wir, dass es jedem nutzen kann, der daran interessiert ist, soziale Phänomene zu untersuchen politischer, pädagogischer, wirtschaftlicher, industrieller oder welcher Art auch immer." In der Tat ist "Grounded Theory" quer durch die wissenschaftlichen Disziplinen zum Standardbegriff der qualitativen Sozialforschung ge - worden. Die inzwischen zum Klassiker avancierte Arbeit von Glaser und Strauss eröffnet eine lehrbare und auch lernbare Forschungsstrategie. Das Konzept bietet eine Antwort auf das Problem, die oft schmerzlich empfundene Lücke zwischen empirischer Forschung und Theorie zu schliessen.

Coders at Work Peter Seibel 2009-12-21 Peter Seibel interviews 15 of the most interesting computer programmers alive today in Coders at Work, offering a companion volume to Apress’s highly acclaimed best-seller Founders at Work by Jessica Livingston. As the words “at work” suggest, Peter Seibel focuses on how his interviewees tackle the day-to-day work of programming, while revealing much more, like how they became great programmers, how they recognize programming talent in others, and what kinds of problems they find most interesting. Hundreds of people have suggested names of programmers to interview on the Coders at Work web site: www.codersatwork.com. The complete list was 284 names. Having digested everyone’s feedback, we selected 15 folks who’ve been kind enough to agree to be interviewed: Frances Allen: Pioneer in optimizing compilers, first woman to win the Turing Award (2006) and first female IBM fellow Joe Armstrong: Inventor of Erlang Joshua Bloch: Author of the Java collections framework, now at Google Bernie Cosell: One of the main software guys behind the original ARPANET IMPs and a master debugger Douglas Crockford: JSON founder, JavaScript architect at Yahoo! L. Peter Deutsch: Author of Ghostscript, implementer of Smalltalk-80 at Xerox PARC and Lisp 1.5 on PDP-1 Brendan Eich: Inventor of JavaScript, CTO of the Mozilla Corporation Brad Fitzpatrick: Writer of LiveJournal, OpenID, memcached, and Perlbal Dan Ingalls: Smalltalk implementor and designer Simon Peyton Jones: Coinventor of Haskell and lead designer of Glasgow Haskell Compiler Donald Knuth: Author of The Art of Computer Programming and creator of TeX Peter Norvig: Director of Research at Google and author of the standard text on AI Guy Steele: Coinventor of Scheme and part of the Common Lisp Gang of Five, currently working on Fortress Ken Thompson: Inventor of UNIX Jamie Zawinski: Author of XEmacs and early Netscape/Mozilla hacker

Produktiv programmieren Neal Ford 2009

Sieben Wochen, sieben Datenbanken Eric Redmond 2012

Recoding Gender Janet Abbate 2017-09-08 The untold history of women and computing: how pioneering women succeeded in a field shaped by gender biases. Today, women earn a relatively low percentage of computer science degrees and hold proportionately few technical computing jobs. Meanwhile, the stereotype of the male “computer geek” seems to be everywhere in popular culture. Few people know that women were a significant presence in the early decades of computing in both the United States and Britain. Indeed, programming in postwar years was considered woman's work (perhaps in contrast to the more manly task of building the computers themselves). In Recoding Gender, Janet Abbate explores the untold history of women in computer science and programming from the Second World War to the late twentieth century. Demonstrating how gender has shaped the culture of computing, she offers a valuable historical perspective on today's concerns over women's underrepresentation in the field. Abbate describes the experiences of women who worked with the earliest electronic digital computers: Colossus, the wartime codebreaking computer at Bletchley Park outside London, and the American ENIAC, developed to calculate ballistics. She examines postwar methods for recruiting programmers, and the 1960s redefinition of programming as the more masculine “software engineering.” She describes the social and business innovations of two early software entrepreneurs, Elsie Shutt and Stephanie Shirley; and she examines the career paths of women in academic computer science. Abbate's account of the bold and creative strategies of women who loved computing work, excelled at it, and forged successful careers will provide inspiration for those working to change gendered computing culture.

Freie Kultur Lawrence Lessig 2006

Refactoring to patterns Joshua Kerievsky 2005

Visionäre der Programmierung Federico Biancuzzi 2009 In Visionare der Programmierung - Die Sprachen und ihre Schopfer werden exklusive Interviews mit den Entwicklern von historischen wie auch von hoch aktuellen Programmiersprachen veroffentlicht. In dieser einzigartigen Zusammenstellung erfahren Sie uber die Hintergrunde, die zu den spezifischen Design-Entscheidungen in den Programmiersprachen gefuhrt haben und uber die ursprungliche Ziele, die die Entwickler im Kopf hatten, als sie eine neue Programmiersprache entwarfen. Ebenso konnen Sie lesen, wieso Abweichungen zum ursprunglichen Design entstanden und welchen Einflu die jeweilige Sprache auf die heutige Softwareentwicklung noch besitzt. Adin D. Falkoff: APL Thomas E. Kurtz: BASIC Charles H. Moore: FORTH Robin Milner: ML Donald D. Chamberlin: SQL Alfred Aho, Peter Weinberger und Brian Kernighan: AWK Charles Geschke und John Warnock: PostScript Bjarne Stroustrup: C++ Bertrand Meyer: Eiffel Brad Cox und Tom Love: Objective-C Larry Wall: Perl Simon Peyton Jones, Paul Hudak, Philip Wadler und John Hughes: Haskell Guido van Rossum: Python Luiz Henrique de Figueiredo und Roberto Ierusalimschy: Lua James Gosling: Java Grady Booch, Ivar Jacobson und James Rumbaugh: UML Anders Hejlsberg: Delphi-Entwickler und fuhrender Entwickler von C#

Managing the Unmanageable Mickey W. Mantle 2012-09-16 “Mantle and Lichty have assembled a guide that will help you hire, motivate, and mentor a software development team that functions at the highest level. Their rules of thumb and coaching advice are great blueprints for new and experienced software engineering managers alike.” —Tom Conrad, CTO, Pandora “I wish I’d had this material available years ago. I see lots and lots of ‘meat’ in here that I’ll use over and over again as I try to become a better manager. The writing style is right on, and I love the personal anecdotes.” —Steve Johnson, VP, Custom Solutions, DigitalFish All too often, software development is deemed unmanageable. The news is filled with stories of projects that have run catastrophically over schedule and budget. Although adding some formal discipline to the development process has improved the situation, it has by no means solved the problem. How can it be, with so much time and money spent to get software development under control, that it remains so unmanageable? In Managing the Unmanageable: Rules, Tools, and Insights for Managing Software People and Teams , Mickey W. Mantle and Ron Lichty answer that persistent question with a simple observation: You first must make programmers and software teams manageable. That is, you need to begin by understanding your people—how to hire them, motivate them, and lead them to develop and deliver great products. Drawing on their combined seventy years of software development and management experience, and highlighting the insights and wisdom of other successful managers, Mantle and Lichty provide the guidance you need to manage people and teams in order to deliver software successfully. Whether you are new to software management, or have already been working in that role, you will appreciate the real-world knowledge and practical tools packed into this guide.

Kuckucksei Clifford Stoll 2015-11-16 ›Kuckucksei‹ schildert bis ins Detail die hochdramatische Jagd nach deutschen Hackern, die in

Downloaded from krr-law.com on August 10, 2022 by guest

amerikanische Computernetze eingedrungen waren. Es ist der autobiografische Report eines amerikanischen Computercracks, der leidenschaftlich für die Sicherheit der Datennetze kämpft. (Dieser Text bezieht sich auf eine frühere Ausgabe.)

Clean Coder Robert C. Martin 2014-03-24 Verhaltensregeln für professionelle Programmierer Erfolgreiche Programmierer haben eines gemeinsam: Die Praxis der Software-Entwicklung ist ihnen eine Herzensangelegenheit. Auch wenn sie unter einem nicht nachlassenden Druck arbeiten, setzen sie sich engagiert ein. Software-Entwicklung ist für sie eine Handwerkskunst. In Clean Coder stellt der legendäre Software-Experte Robert C. Martin die Disziplinen, Techniken, Tools und Methoden vor, die Programmierer zu Profis machen. Dieses Buch steckt voller praktischer Ratschläge und behandelt alle wichtigen Themen vom professionellen Verhalten und Zeitmanagement über die Aufwandsschätzung bis zum Refactoring und Testen. Hier geht es um mehr als nur um Technik: Es geht um die innere Haltung. Martin zeigt, wie Sie sich als Software-Entwickler professionell verhalten, gut und sauber arbeiten und verlässlich kommunizieren und planen. Er beschreibt, wie Sie sich schwierigen Entscheidungen stellen und zeigt, dass das eigene Wissen zu verantwortungsvollem Handeln verpflichtet. In diesem Buch lernen Sie: Was es bedeutet, sich als echter Profi zu verhalten Wie Sie mit Konflikten, knappen Zeitplänen und unvernünftigen Managern umgehen Wie Sie beim Programmieren im Fluss bleiben und Schreibblockaden überwinden Wie Sie mit unerbittlichem Druck umgehen und Burnout vermeiden Wie Sie Ihr Zeitmanagement optimieren Wie Sie für Umgebungen sorgen, in denen Programmierer und Teams wachsen und sich wohlfühlen Wann Sie Nein sagen sollten – und wie Sie das anstellen Wann Sie Ja sagen sollten – und was ein Ja wirklich bedeutet Großartige Software ist etwas Bewundernswertes: Sie ist leistungsfähig, elegant, funktional und erfreut bei der Arbeit sowohl den Entwickler als auch den Anwender. Hervorragende Software wird nicht von Maschinen geschrieben, sondern von Profis, die sich dieser Handwerkskunst unerschütterlich verschrieben haben. Clean Coder hilft Ihnen, zu diesem Kreis zu gehören. Über den Autor: Robert C. Uncle Bob Martin ist seit 1970 Programmierer und bei Konferenzen in aller Welt ein begehrter Redner. Zu seinen Büchern gehören Clean Code – Refactoring, Patterns, Testen und Techniken für sauberen Code und Agile Software Development: Principles, Patterns, and Practices. Als überaus produktiver Autor hat Uncle Bob Hunderte von Artikeln, Abhandlungen und Blogbeiträgen verfasst. Er war Chefredakteur bei The C++ Report und der erste Vorsitzende der Agile Alliance. Martin gründete und leitet die Firma Object Mentor, Inc., die sich darauf spezialisiert hat, Unternehmen bei der Vollendung ihrer Projekte behilflich zu sein.

Hello, Startup Yevgeniy Brikman 2015-10-21 This book is the "Hello, World" tutorial for building products, technologies, and teams in a startup environment. It's based on the experiences of the author, Yevgeniy (Jim) Brikman, as well as interviews with programmers from some of the most successful startups of the last decade, including Google, Facebook, LinkedIn, Twitter, GitHub, Stripe, Instagram, AdMob, Pinterest, and many others. Hello, Startup is a practical, how-to guide that consists of three parts: Products, Technologies, and Teams. Although at its core, this is a book for programmers, by programmers, only Part II (Technologies) is significantly technical, while the rest should be accessible to technical and non-technical audiences alike. If you’re at all interested in startups—whether you’re a programmer at the beginning of your career, a seasoned developer bored with large company politics, or a manager looking to motivate your engineers—this book is for you.

Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code Robert C. Martin 2013-12-18 h2> Kommentare, Formatierung, Strukturierung Fehler-Handling und Unit-Tests Zahlreiche Fallstudien, Best Practices, Heuristiken und Code Smells Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code Aus dem Inhalt: Lernen Sie, guten Code von schlechtem zu unterscheiden Sauberen Code schreiben und schlechten Code in guten umwandeln Aussagekräftige Namen sowie gute Funktionen, Objekte und Klassen erstellen Code so formatieren, strukturieren und kommentieren, dass er bestmöglich lesbar ist Ein vollständiges Fehler-Handling implementieren, ohne die Logik des Codes zu verschleiern Unit-Tests schreiben und Ihren Code testgesteuert entwickeln Selbst schlechter Code kann funktionieren. Aber wenn der Code nicht sauber ist, kann er ein Entwicklungsunternehmen in die Knie zwingen. Jedes Jahr gehen unzählige Stunden und beträchtliche Ressourcen verloren, weil Code schlecht geschrieben ist. Aber das muss nicht sein. Mit Clean Code präsentiert Ihnen der bekannte Software-Experte Robert C. Martin ein revolutionäres Paradigma, mit dem er Ihnen aufzeigt, wie Sie guten Code schreiben und schlechten Code überarbeiten. Zusammen mit seinen Kollegen von Object Mentor destilliert er die besten Praktiken der agilen Entwicklung von sauberem Code zu einem einzigartigen Buch. So können Sie sich die Erfahrungswerte der Meister der Software-Entwicklung aneignen, die aus Ihnen einen besseren Programmierer machen werden – anhand konkreter Fallstudien, die im Buch detailliert durchgearbeitet werden. Sie werden in diesem Buch sehr viel Code lesen. Und Sie werden aufgefordert, darüber nachzudenken, was an diesem Code richtig und falsch ist. Noch wichtiger: Sie werden herausgefordert, Ihre professionellen Werte und Ihre Einstellung zu Ihrem Beruf zu überprüfen. Clean Code besteht aus drei Teilen:Der erste Teil beschreibt die Prinzipien, Patterns und Techniken, die zum Schreiben von sauberem Code benötigt werden. Der zweite Teil besteht aus mehreren, zunehmend komplexeren Fallstudien. An jeder Fallstudie wird aufgezeigt, wie Code gesäubert wird – wie eine mit Problemen behaftete Code-Basis in eine solide und effiziente Form umgewandelt wird. Der dritte Teil enthält den Ertrag und den Lohn der praktischen Arbeit: ein umfangreiches Kapitel mit Best Practices, Heuristiken und Code Smells, die bei der Erstellung der Fallstudien zusammengetragen wurden. Das Ergebnis ist eine Wissensbasis, die beschreibt, wie wir denken, wenn wir Code schreiben, lesen und säubern. Dieses Buch ist ein Muss für alle Entwickler, Software-Ingenieure, Projektmanager, Team-Leiter oder Systemanalytiker, die daran interessiert sind, besseren Code zu produzieren. Über den Autor: Robert C. »Uncle Bob« Martin entwickelt seit 1970 professionell Software. Seit 1990 arbeitet er international als Software-Berater. Er ist Gründer und Vorsitzender von Object Mentor, Inc., einem Team erfahrener Berater, die Kunden auf der ganzen Welt bei der Programmierung in und mit C++, Java, C#, Ruby, OO, Design Patterns, UML sowie Agilen Methoden und eXtreme Programming helfen.

Downloaded from krr-law.com on August 10, 2022 by guest