DOWNLOAD PDF LEARNING TO PROGRAM STEVEN FOOTE

Chapter 1 : learning to program by steven foote ebooks preview

My first programming job was working in the systems department of the BYU Law Library. I worked on a wide variety of projects, including installing Firefox on law professors' computers and writing Perl scripts to process huge and disgustingly poorly formatted Oracle reports.

Create a Google Chrome Extension. Welcome to the world of programming! The program will be the first part of a larger project that we will build throughout this book. One word of warning before we get started: Not understanding what you are doing is for me at least a big part of programming computers. By the end of the book, all the mysteries will be revealed. For now, trust me. Choose a Text Editor One of the most important parts of programming is writing code but, as discussed in the sidebar, coding and programming are not exactly the same thing. When I say code, I mean instructions for a computer written in a programming language that the computer can understand. Code is written in plain-text files using a text editor. You have many text editors to choose from, so choose wisely. Coding The difference between programming and coding might be subtle, but it is significant. Coding is just a part of programming. In fact, coding is probably the easiest part of programming. Programming includes such tasks as creating an environment in which your code can successfully be executed, organizing your code in a logical way, testing your code to identify errors, debugging your code to find what is causing those errors, working with code and frameworks that others write, and packaging your code in a way that others can use it. When you know how to program, coding can be a lot more fun. TIP My first mistake in learning to program was my choice of text editors. I had previously used only Microsoft Word for putting my words and thoughts into a computer, so I tried programming in Word. Next, I opened up Notepad and used that for months before I discovered the amazing alternatives I could have been using. Learn from my mistakes. Core Features Several text editors are designed specifically for coding. The next section outlines some of the more popular text editors and their most important features. All these editors share a few core features: Monospace Type Font Monospace type font is a font style in which every character takes up the same amount of space. In other words, an i is as wide as a w and also as wide as a space. Syntax Highlighting Just as the English language has nouns, verbs, adjectives, and so on, programming languages are made up of different parts such as variables, reserved words, and strings. A good text editor differentiates the various parts of programming languages, usually with different text colors. Take a look at Figures 1. Even without knowing what the code means, you can see that Figure 1. Syntax highlighting can also help you find typos in your code. Is this not much easier to read and much prettier, too? Text Completion When writing code, you need to type the same words over and over again. For the code to work properly, the words must be typed exactly the same way every time for example, myCode is not the same as MyCode or mycode. Tiny misspellings and typos are hard to find and can cause major headaches. Using text completion helps you write code faster and can also help you avoid typos and other errors. Extensibility Text editors that are meant for programmers should allow those programmers to build extensions and plug-ins to enhance and modify the behavior of the editor. As you will see throughout this book, extensibility is important not only for text editors, but for nearly all software. Extensibility makes it possible for those features to be built by anyone who wants to build them.

Chapter 2 : Learning to Program - Steven Foote - pocket () | Adlibris Bokhandel

A good basic, beginners level book on computer programming. I stress, for absolute beginners. If you know nothing about programming, software, or data types, this is a great place to begin.

I compare runtimes and interpreters to player pianos and piano players pianists? I love analogies, and I use them all the time to better understand the world, but I use them in software more than anything else. I am not alone. On a daily basis, software developers build pipes, trees, graphs, and objects, but none of them are real. In software development, nothing is real. As a new developer, I understood an "object" to be something realâ€”something that can be held, touched, or at least seen. A keyboard is an object, a mouse is an object, a CRT monitor is an object. But now I was learning that objects in programming are not really objects, but the idea of objects. And a class is a generic version of an imaginary objectâ€”the idea of an idea. See, a dog is a real object, and barking and eating are real things that real dogs really do. So, are these software objects real objects or not? Is object oriented programming just for programming robots? Is this tutorial teaching me to build a robot dog? Programming concepts can be hard to grasp, especially for those of us clinging to reality. Everything is an idea; nothing goes beyond the conceptual. Even code is just a codified representation of the idea; bugs are where the idea and code diverge. All of the analogies, the metaphors, the koans , the whiteboard drawings, the notebooks sketches, and rubber ducking are attempts to make the ideas in our heads more concrete and more understandable. Commit Messages Posted My intention with this post is to write a short and clear summary of what makes a good commit message. A commit message should be short and clear. If the description needs to be long and confusing, the commit is probably too large. A commit message should be in the present not past tense. Describe what the commit is doing, not the work you did. A past-tense commit message is also arrogant, because it implies that you think your code is so certain to be merged that you can describe it as if it has already happened. A commit message can be funny, given the humor does not make unclear the meaning in the message. The robots are not being programmed by robots; let your personality show. But my favorite projects by far were building web applications for law students and library employees to use. This story is about one of those web applications. Every semester the library circulation manager has to create a work schedule for all of the circulation employees. There are 2 desks that must be manned from 6: No student can close one night then open the next morning. More tenured student employees get more desirable shifts. If a student wants to work 12 hours per week, that student should not be scheduled for more than 14 hours, nor less than 10 hours. Shifts are diced up in 30 minute increments, with the minimum length of 2 hours and maximum length of 4 hours. I could keep going, but I think the point is quite clear: The process of creating the work schedule would generally take the circulation manager about 2 months i. The Challenge I was a new, overly confident programmer, certain that the computer could solve any problem. I saw the two month process of papers, spreadsheets, frustration, and tears, and I knew there was a better way. During the spring of , I took on the challenge. In less than 2 months, with the assistance of Object Oriented PHP, I created a web application that would create and display a fully compliant schedule. The 2 month process was about to become a 15 second process again, I was a new programmer, and I had never studied algorithms; all things considered I think 15 seconds was pretty good. I was jubilant as I watched her click the "Build the Schedule" button. The computers had won! Defeat Then my jubilee ended. My mind quickly scanned the possibility of adding employee compatibility to my model when I heard, "Oh, Anne is probably too shy to work the front desk. Unless Jess or Brandon is doing shelving at the same time In theory I could make an employee model detailed and accurate enough to handle these new constraints, but I knew these two new constraints were just the beginning and not the end. My program worked in a cold, mechanical sense, but it lacked a human touch. The Lesson I returned to my desk with a completely different view of the role that computers should play in our lives. In the five minute interaction I had with the circulation manager, I learned two important lessons: Computers work best when they are used to assist us humans in the tasks we are trying to perform. Calculations, data transfer, and even data analysis are perfect jobs for computers, but human decisions should be left to humans. A computer may sift through the many applicants for a job to surface the most qualified

candidates, but a hiring manager would be loath to relinquish the interviewing and final hiring decision to the computer. Humans enjoy doing work. People want the satisfaction of doing work more than they want the free time available by having computers do the work for them. I reused the Object Oriented code I had written for the fully automated schedule builder to create a schedule building interface that the circulation manager could use to add a human touch to the process. As far as I know, the rewritten program is still being used today. How the Web Works Posted The Web has changed the way we shop , the way we watch TV and movies , the way we work , the way we read the news , the way we socialize , the way we find jobs and show our work experience ; in short, the Web has changed the way we live. Considering the fundamental role the web now plays in our lives, I believe we should all have at least a basic understanding of how it works. The Web is a collection of interconnected documents that can be requested using HTTP , and are generally viewed using a Web browser. In recent years, the Web has grown beyond a simple collection of documents to a collection of interconnected applications. HTML contains both the content to be displayed and a description of the content. Web browsers us CSS to decide how things on a web page look and JavaScript to decide how things on a web page act. Modern web servers often create the HTML document as it is requested e. There are different types of requests for different purposes. When you want to view a blog, you make a GET request. POST A post request is used when you want to send or "post" some data to the server. When you want to "post" an entry on your blog, you make a POST request. When you try to go to www. When you send your request, the first place the request goes is to a DNS Domain Name System which is like a big phone book that maps domains like google. Once the IP adress is found, the request bounces through a bunch of routers before finally finding the server. Routers tend to send requests in generally the right geographic direction, depending on a number of factors including other network traffic. The request is sent with a return IP address, so the server knows where to send the response. Browsers Bring it All Together When you type "cnn. Then routers shepherd the request to the correct server. The server finds the document that is being requested and sends a response, which is routed back to the computer or phone that made the original request. With so many steps and so many places where things could go wrong, it may seem like a wonder that the web works at all. But it does, and now you know a bit more about how. If you are interested in learning more about the Web, computers, and programming, check out my book, Learning to Program. At first it seemed strange to me that anyone would not innately understand the GitHub workflow. But then, as I explained what I seemed to know instinctively about branches, remotes, origins, and upstreams, I realized that the GitHub workflow is neither intuitive nor straightforward. Now that it has become a part of me, it seems obvious. No one was born understanding complex data structures, Git, the US tax system, or any of the other overly complicated things we humans have come up with. Even Wayne Gretzky had to learn how to ice skate. Even Michael Phelps started by blowing bubbles in swimming lessons. Over the next few weeks, I will be writing about the aspects of web development that seem obvious to me now, but did not always. If you have any topic suggestions or recommendations let me know. If you would like to start learning to program, check out my book. The title and content for this post were inspired by the book Everything is Obvious Out in the Wild Posted Seth is a good friend. As of a few weeks ago that book, Learning to Program is officially available, out in the wild. I am really excited, but also quite nervous. With this book, I have put myself and my knowledge out there in the public eye to be scrutinized and judged. I wrote my first line of code less than 7 years ago. I have never taken programming course unless Udacity counts, and I think it does. Now I am out there telling the world how to learn to program. I did a lot of studying for this book, to make sure my understanding and explanations were as close to the truth as possible. I continue to learn every day.

Chapter 3 : Learning to Program: the book

Steven Foote is a web developer at LinkedIn. A self-taught programmer who loves technology, especially the Web, he has a Bachelor's degree and Master's degree in Accountancy from.

Chapter 4 : Learning to Program â€“ CoderProg

DOWNLOAD PDF LEARNING TO PROGRAM STEVEN FOOTE

Steven Foote is the author of Learning To Program (avg rating, 37 ratings, 4 reviews, published) and Learning to Program (avg rating, 8 ra.

Chapter 5 : Learning to Program : Steven Foote :

Author Steven Foote taught himself to program, figuring out the best ways to overcome every obstacle. Now a professional web developer, he'll help you follow in his footsteps. He teaches concepts you can use with any modern programming language, whether you want to program computers, smartphones, tablets, or even robots.

Chapter 6 : Steven Foote (Author of Learning To Program)

Pdf file is about learning to program by steven foote is available in several types of edition. This pdf document is presented in digital edition of learning to program by steven foote and it can be searched throughout the net in such search engines as google, bing and yahoo.

Chapter 7 : Learning to Program â€“ ScanLibs

Title: Learning To Program Steven Foote theinnatdunvilla.com Author: Book PDF Subject: Free Download Learning To Program Steven Foote Pdf Book PDF Keywords: Free DownloadLearning To Program Steven Foote Pdf Book PDF, read, reading book, free, download, book, ebook, books, ebooks, manual.

Chapter 8 : Foote, Learning to Program | Pearson

Learning to program takes a lot of practice, and you shouldn't expect to be a pro after reading one book, even this book. This book is about building the foundation that you need to become a great programmer.

Chapter 9 : Learning to Program - E-bok - Steven Foote () | Bokus

About the author. Steven Foote is a web developer at LinkedIn. A self-taught programmer who loves technology, especially the Web, he has a Bachelor's degree and Master's degree in Accountancy from Brigham Young University.