Real World Ocaml Functional Programming For The Masses Yaron Minsky

Haskell is one of the leading languages for teaching functional programming, enabling students to write simpler and cleaner code, and to learn how to structure and reason about programs. This introduction is ideal for beginners: it requires no previous programming experience and all concepts are explained from first principles via carefully chosen examples. Each chapter includes exercises that range from the straightforward to extended projects, plus suggestions for further reading on more advanced topics. The author is a leading Haskell researcher and instructor, well-known for his teaching skills. The presentation is clear and simple, and benefits from having been refined and class-tested over several years. The result is a text that can be used with courses, or for self-learning. Features include freely accessible Powerpoint slides for each chapter, solutions to exercises and examination questions (with solutions) available to instructors, and a downloadable code that's fully compliant with the latest Haskell release.
"This work strikes a balance between the pure functional aspects ofF# and the object-oriented and imperative features that make it souseful in practice, enable .NET integration, and make large-scaledata processing possible." —Thore Graepel, PhD, Researcher, Microsoft Research Ltd. Over the next five years, F# is expected to become one of theworld's most popular functional programming languages forscientists of all disciplines working on the Windows platform. F#is free and, unlike MATLAB® and other software withnumerical/scientific origins, is a full-fledged programminglanguage. Developed in consultation with Don Syme of Microsoft ResearchLtd.—who wrote the language—F# for Scientistsexplains and demonstrates the powerful features of this importantnew programming language. The book assumes no prior experience andguides the reader from the basics of computer programming to theimplementation of state-of-the-art algorithms. F# for Scientists begins with coverage of introductorymaterial in the areas of functional programming, .NET, andscientific computing, and goes on to explore: Program structure Optimization Data structures Libraries Numerical analysis Databases Input and output Interoperability Visualization Screenshots of development using Visual Studio are used toillustrate compilation, debugging, and interactive use, whilecomplete examples of a few whole programs are included to givereaders a complete view of F#'s capabilities. Written in a clear and concise style, F# for Scientistsis well suited for researchers, scientists, and developers who wantto program under the Windows platform. It also serves as an idealsupplemental text for advanced undergraduate and graduate studentswith a background in science or engineering.
This fast-moving tutorial introduces you to OCaml, an industrial-strength programming language designed for expressiveness, safety, and speed. Through the book’s many examples, you’ll quickly learn how OCaml stands out as a tool for writing fast, succinct, and readable systems code. Real World OCaml takes you through the concepts of the language at a brisk pace, and then helps you explore the tools and techniques that make OCaml an effective and practical tool. In the book’s third section, you’ll delve deep into the details of the compiler toolchain and OCaml’s simple and efficient runtime system. Learn the foundations of the language, such as higher-order functions, algebraic data types, and modules Explore advanced features such as functors, first-class modules, and objects Leverage Core, a comprehensive general-purpose standard library for OCaml Design effective and reusable libraries, making the most of OCaml’s approach to abstraction and modularity Tackle practical programming problems from command-line parsing to asynchronous network programming Examine profiling and interactive debugging techniques with tools such as GNU gdb
This book describes data structures and data structure design techniques for functional languages.
Functional C
Verified Functional Programming in Agda
Discrete Mathematics and Functional Programming
Practical OCaml
Learn to Program, One Game at a Time!
The Functional Approach to Programming
Your success—and sanity—are closer at hand when you work at a higher level of abstraction, allowing your attention to be on the business problem rather than the details of the programming platform. Domain Specific Languages—"little languages" implemented on top of conventional programming languages—give you a way to do this because they model the domain of your business problem. DSLs in Action introduces the concepts and definitions a developer needs to build high-quality domain specific languages. It provides a solid foundation to the usage as well as implementation aspects of a DSL, focusing on the necessity of applications speaking the language of the domain. After reading this book, a programmer will be able to design APIs that make better domain models. For experienced developers, the book addresses the intricacies of domain language design without the pain of writing parsers by hand. The book discusses DSL usage and implementations in the real world based on a suite of JVM languages like Java, Ruby, Scala, and Groovy. It contains code snippets that implement real world DSL designs and discusses the pros and cons of each implementation. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. What's Inside Tested, real-world examples How to find the right level of abstraction Using language features to build internal DSLs Designing parser/combinator-based little languages
Learn how to solve day-to-day problems in data processing, numerical computation, system scripting, and database-driven web applications with the OCaml multi-paradigm programming language. This hands-on book shows you how to take advantage of OCaml’s functional, imperative, and object-oriented programming styles with recipes for many real-world tasks. You’ll start with OCaml basics, including how to set up a development environment, and move toward more advanced topics such as the module system, foreign-function interface, macro language, and the ocamlbuild system. Quickly learn how to put OCaml to work for writing succinct and readable code.
Advanced text on how to program in the functional way; has exercises, solutions and code.
This book teaches functional programming using Haskell and examples drawn from multimedia applications.
Realm of Racket
Functional Programming for the Masses
Foundations of F#
Learning Functional Programming Through Multimedia
Real World OCaml
Domain Modeling Made Functional
Summary Functional Programming in JavaScript teaches JavaScript developers functional techniques that will improve extensibility, modularity, reusability, testability, and performance. Through concrete examples and jargon-free explanations, this book teaches you how to apply functional programming to real-life development tasks Purchase of the print book includes a free eBook in PDF, Kindle, and ePu Publications. About the Technology In complex web applications, the low-level details of your JavaScript code can obscure the workings of the system as a whole. As a coding style, functional programming (FP) promotes loosely coupled relationships among the components of your application, making the big picture easier to design, communicate, and maintain. About the Book Functional Programming in techniques to improve your web applications - their extensibility, modularity, reusability, and testability, as well as their performance. This easy-to-read book uses concrete examples and clear explanations to show you how to use functional programming in real life. If you're new to functional programming, you'll appreciate this guide's many insightful comparisons to imperative or object-oriented programmin functional design. By the end, you'll think about application design in a fresh new way, and you may even grow to appreciate monads! What's Inside High-value FP techniques for real-world uses Using FP where it makes the most sense Separating the logic of your system from implementation details FP-style error handling, testing, and debugging All code samples use JavaScript ES6 (ES 2015) About the with a solid grasp of JavaScript fundamentals and web application design. About the Author Luis Atencio is a software engineer and architect building enterprise applications in Java, PHP, and JavaScript. Table of Contents PART 1 THINK FUNCTIONALLY Becoming functional Higher-order JavaScript PART 2 GET FUNCTIONAL Few data structures, many operations Toward modular, reusable code Desig complexity PART 3 ENHANCING YOUR FUNCTIONAL SKILLS Bulletproofing your code Functional optimizations Managing asynchronous events and data
Learn from F#'s inventor to become an expert in the latest version of this powerful programming language so you can seamlessly integrate functional, imperative, object-oriented, and query programming style flexibly and elegantly to solve any programming problem. Expert F# 4.0 will help you achieve unrivaled levels of programmer productivity and program clarity across multiple platforms including Windo and iOS as well as HTML5 and GPUs. F# 4.0 is a mature, open source, cross-platform, functional-first programming language which empowers users and organizations to tackle complex computing problems with simple, maintainable, and robust code. Expert F# 4.0 is: A comprehensive guide to the latest version of F# by the inventor of the language A treasury of F# techniques for practical problem-solving applications and F# 4.0 concepts, syntax, and features Written by F#'s inventor and two major F# community members, Expert F# 4.0 is a comprehensive and in-depth guide to the language and its use. Designed to help others become experts, the book quickly yet carefully describes the paradigms supported by F# language, and then shows how to use F# elegantly for a practical web, data, parallel and ar world's experts in F# show you how to program in F# the way they do!
This book provides a distinct way to teach discrete mathematics. Since discrete mathematics is crucial for rigorous study in computer science, many texts include applications of mathematical topics to computer science or have selected topics of particular interest to computer science. This text fully integrates discrete mathematics with .....
Summary Type-Driven Development with Idris, written by the creator of Idris, teaches you how to improve the performance and accuracy of your programs by taking advantage of a state-of-the-art type system. This book teaches you with Idris, a language designed to support type-driven development. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publicat perfect for learning type-driven programming techniques you can apply in any codebase. About the Book Type-Driven Development with Idris teaches you how to improve the performance and accuracy of your code by taking advantage of a state-of-the-art type system. In this book, you'll learn type-driven development of real-world software, as well as how to handle side effects, interaction, state, and concu able to develop robust and verified software in Idris and apply type-driven development methods to other languages. What's Inside Understanding dependent types Types as first-class language constructs Types as a guide to program construction Expressing relationships between data About the Reader Written for programmers with knowledge of functional programming concepts. About the Author Edwin implementation of the Idris language. Table of Contents PART 1 - INTRODUCTION Overview Getting started with IdrisPART 2 - CORE IDRIS Interactive development with types User-defined data types Interactive programs: input and output processing Programming with first-class types Interfaces: using constrained generic types Equality: expressing relationships between data Predicates: expressing ass types Views: extending pattern matching PART 3 - IDRIS AND THE REAL WORLD Streams and processes: working with infinite data Writing programs with state State machines: verifying protocols in types Dependent state machines: handling feedback and errors Type-safe concurrent programming
The Haskell School of Expression
Learn You Some Erlang for Great Good!
Real-World Functional Programming
More OCaml
Real World Haskell
Practical Haskell
Functional programming languages like F#, Erlang, and Scala are attractingattention as an efficient way to handle the new requirements for programmingmulti-processor and high-availability applications. Microsoft's new F# is a truefunctional language and C# uses functional language features for LINQ andother recent advances. Real-World Functional Programming is a unique tutorial that explores thefunctional programming model through the F# and C# languages. The clearlypresented ideas and examples teach readers how functional programming differsfrom other approaches. It explains how ideas look in F#-a functionallanguage-as well as how they can be successfully used to solve programmingproblems in C#. Readers build on what they know about .NET and learn wherea functional approach makes the most sense and how to apply it effectively inthose cases. The reader should have a good working knowledge of C#. No prior exposure toF# or functional programming is required. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book.
In OCaml from the Very Beginning John Whitington takes a no-prerequisites approach to teaching a modern general-purpose programming language. Each small, self-contained chapter introduces a new topic, building until the reader can write quite substantial programs. There are plenty of questions and, crucially, worked answers and hints. OCaml from the Very Beginning will appeal both to new programmers, and experienced programmers eager to explore functional languages such as OCaml. It is suitable both for formal use within an undergraduate or graduate curriculum, and for the interested amateur.
This book constitutes revised selected papers from the 21st International Symposium on Trends in Functional Programming, TFP 2020, which was held in Krakow, Poland, during February 13-14, 2020. The 11 full papers presented in this volume were carefully reviewed and selected from 22 submissions. They were organized in topical sections named: domain-specific languages; debugging and testing; reasoning and effects; and parallelism.
This book uses a functional programming language (F#) as a metalanguage to present all concepts and examples, and thus has an operational flavour, enabling practical experiments and exercises. It includes basic concepts such as abstract syntax, interpretation, stack machines, compilation, type checking, garbage collection, and real machine code. Also included are more advanced topics on polymorphic types, type inference using unification, co- and contravariant types, continuations, and backwards code generation with on-the-fly peephole optimization. This second edition includes two new chapters. One describes compilation and type checking of a full functional language, tying together the previous chapters. The other describes how to compile a C subset to real (x86) hardware, as a smooth extension of the previously presented compilers.The examples present several interpreters and compilers for toy languages, including compilers for a small but usable subset of C, abstract machines, a garbage collector, and ML-style polymorphic type inference. Each chapter has exercises. Programming Language Concepts covers practical construction of lexers and parsers, but not regular expressions, automata and grammars, which are well covered already. It discusses the design and technology of Java and C# to strengthen students’ understanding of these widely used languages.
Programming Language Concepts
Learn Type-Driven Development
F# for Scientists
Functional Programming in C++
How to improve your JavaScript programs using functional techniques
Benefit from type systems to build reliable and safe applications using ReasonML 3
This easy-to-use, fast-moving tutorial introduces you to functional programming with Haskell. You'll learn how to use Haskell in a variety of practical ways, from short scripts to large and demanding applications. Real World Haskell takes you through the basics of functional programming at a brisk pace, and then helps you increase your understanding of Haskell in real-world issues like I/O, performance, dealing with data, concurrency, and more as you move through each chapter.
This book constitutes the refereed proceedings of the 23rd International Symposium on Practical Aspects of Declarative Languages, PADL 2021, held in Copenhagen, Denmark, in January 2021. The 10 full papers were carefully reviewed and selected from 21 submissions. The papers present original work emphasizing novel applications and implementation techniques for all forms of declarative concepts, including programming with sets, functions, logic, and constraints. The papers are organized in the following topical headings: Foundations and Programming Concepts; Applications of Declarative Languages, and Declarative Approaches to Testing and Debugging. Due to the Corona pandemic PADL 2021 was held as a virtual event.
Richard Bird takes a radical approach to algorithm design, namely, design by calculation. These 30 short chapters each deal with a particular programming problem drawn from sources as diverse as games and puzzles, intriguing combinatorial tasks, and more familiar areas such as data compression and string matching. Each pearl starts with the statement of the problem expressed using the functional programming language Haskell, a powerful yet succinct language for capturing algorithmic ideas clearly and simply. The novel aspect of the book is that each solution is calculated from an initial formulation of the problem in Haskell by appealing to the laws of functional programming. Pearls of Functional Algorithm Design will appeal to the aspiring functional programmer, students and teachers interested in the principles of algorithm design, and anyone seeking to master the techniques of reasoning about programs in an equational style.
This is the first book to bring F# to the world. It is likely to have many imitators but few competitors. Written by F# evangelist, Rob Pickering, and tech reviewed by F# ’ s inventor, Don Syme, it is an elegant, comprehensive introduction to all aspects of the language and an incisive guide to using F# for real-world professional development. It is detailed, yet clear and concise, and suitable for readers at any level of experience. Every professional .NET programmer needs to learn about Functional Programming (FP), and there ’ s no better way to do it than by learning F# — and no easier way to learn F# than from this book.
A Beginner's Guide
23rd International Symposium, PADL 2021, Copenhagen, Denmark, January 18-19, 2021, Proceedings
Haskell Programming from First Principles
Communicating Sequential Processes
21st International Symposium, TFP 2020, Krakow, Poland, February 13–14, 2020, Revised Selected Papers
OCaml from the Very Beginning
Introduces fundamental techniques for reasoning mathematically about functional programs. Ideal for a first- or second-year undergraduate course.
A fast paced guide for JavaScript developers for writing safe, fast, and reusable code by leveraging ResaonML’s strong static type system Key Features Reduce code errors with the power of type systems Employ static typechecking and genericity to promote code reuse and consistency Understand functional programming which is the foundation of type-driven development Book Description Type-driven development is an approach that uses a static type system to achieve results including safety and efficiency. Types are used to express relationships and other assumptions directly in the code, and these assumptions are enforced by the compiler before the code is run. Learn Type-Driven Development covers how to use these type systems to check the logical consistency of your code. This book begins with the basic idea behind type-driven development. You’ll learn about values (or terms) and how they contrast with types. As you progress through the chapters, you’ll cover how to combine types and values inside modules and build structured types out of simpler ones. You’ll then understand how to express choices or alternatives directly in the type system using variants, polymorphic variants, and generalized algebraic data types. You’ll also get to grips with sum types, build sophisticated data types from generics, and explore functions that express change in the types of values. In the concluding chapters, you’ll cover advanced techniques for code reuse, such as parametric polymorphism and subtyping. By end of this book, you will have learned how to iterate through a type-driven process of solving coding problems using static types, together with dynamic behavior, to obtain more safety and speed. What you will learn Use static types to capture information, making programs safer and faster Learn ReasonML from experienced type-driven developers Enhance safety by simply using basic types Understand the most important type-driven concepts with simple examples Explore a design space using static typing and find the best way to express your system rules Use static types and dynamic runtime in harmony to write even safer and faster code Who this book is for If you’re a programmer working with dynamically typed languages and are looking for ways to mitigate production runtime errors, Learn Type-Driven Development is for you. You’ll also find this book helpful if you’re a programmer working with statically typed languages looking for increased safety and improved performance.
In recent years, several formalisms for program construction have appeared. One such formalism is the type theory developed by Per Martin-Löf. Well suited as a theory for program construction, it makes possible the expression of both specifications and programs within the same formalism. Furthermore, the proof rules

can be used to derive a correct program from a specification as well as to verify that a given program has a certain property. This book contains a thorough introduction to type theory, with information on polymorphic sets, subsets, monomorphic sets, and a full set of helpful examples.
Erlang is the language of choice for programmers who want to write robust, concurrent applications, but its strange syntax and functional design can intimidate the uninitiated. Luckily, there's a new weapon in the battle against Erlang-phobia: Learn You Some Erlang for Great Good! Erlang maestro Fred Hébert starts slow and eases you into the basics: You'll learn about Erlang's unorthodox syntax, its data structures, its type system (or lack thereof!), and basic functional programming techniques. Once you've wrapped your head around the simple stuff, you'll tackle the real meat-and-potatoes of the language: concurrency, distributed computing, hot code loading, and all the other dark magic that makes Erlang such a hot topic among today's savvy developers. As you dive into Erlang's functional fantasy world, you'll learn about: –Testing your applications with EUnit and Common Test –Building and releasing your applications with the OTP framework –Passing messages, raising errors, and starting/stopping processes over many nodes –Storing and retrieving data using Mnesia and ETS –Network programming with TCP, UDP, and the inet module –The simple joys and potential pitfalls of writing distributed, concurrent applications Packed with lighthearted illustrations and just the right mix of offbeat and practical example programs, Learn You Some Erlang for Great Good! is the perfect entry point into the sometimes-crazy, always-thrilling world of Erlang.
Type-Driven Development with Idris
The Formal Semantics of Programming Languages
Trends in Functional Programming
Functional Programming in JavaScript
With examples in F# and C#
Code You Can Believe In

In More OCaml John Whitington takes a meandering tour of functional programming with OCaml, introducing various language features and describing some classic algorithms. The book ends with a large worked example dealing with the production of PDF files. There are questions for each chapter together with worked answers and hints. More OCaml will appeal both to existing OCaml programmers who wish to brush up their skills, and to experienced programmers eager to explore functional languages such as OCaml. It is hoped that each reader will find something new, or see an old thing in a new light. For the more casual reader, or those who are used to a different functional language, a summary of basic OCaml is provided at the front of the book.
Racket is a descendant of Lisp, a programming language renowned for its elegance, power, and challenging learning curve. But while Racket retains the functional goodness of Lisp, it was designed with beginning programmers in mind. Realm of Racket is your introduction to the Racket language. In Realm of Racket, you'll learn to program by creating increasingly complex games. Your journey begins with the Guess My Number game and coverage of some basic Racket etiquette. Next you'll dig into syntax and semantics, lists, structures, and conditionals, and learn to work with recursion and the GUI as you build the Robot Snake game. After that it's on to lambda and mutant structs (and an Orc Battle), and fancy loops and the Dice of Doom. Finally, you'll explore laziness, AI, distributed games, and the Hungry Henry game. As you progress through the games, chapter checkpoints and challenges help reinforce what you've learned. Offbeat comics keep things fun along the way. As you travel through the Racket realm, you'll: –Master the quirks of Racket's syntax and semantics –Learn to write concise and elegant functional programs –Create a graphical user interface using the 2htdp/image library –Create a server to handle true multiplayer games Realm of Racket is a lighthearted guide to some serious programming. Read it to see why Racketeers have so much fun!
Functional C teaches how to program in C, assuming that the student has already learnt how to formulate algorithms in a functional style. By using this as a starting point, the student will become a better C programmer, capable of writing programs that are easier to comprehend, maintain and that avoid common errors and pitfalls. All program code that appears in Functional C is available on our ftp server - see below. How to find a code fragment? To access a particular code fragment, use the book to locate the section or subsection in which the code fragment appears, then click on that section in the code index . This will open the appropriate page at the beginning of the section. The code fragment may then be selected using the copy/paste facilities of your browser. Each chapter is represented by a separate page, so as an alternative to the procedure above you can use the save-as menu of your browser to up-load all code fragments in a particular chapter at once. Also available on our ftp server is errata for Functional C.
Functional programming is a very powerful programming paradigm that can help us to write better code. This book presents essential functional and reactive programming concepts in a simplified manner using Typescript.
Programming in Haskell
Algorithms, Methods & Diversions
The Science of Functional Programming (draft version)
Functional programming for the masses
A Real World Guide to Programming
Purely Functional Data Structures

You want increased customer satisfaction, faster development cycles, and less wasted work. Domain-driven design (DDD) combined with functional programming is the innovative combo that will get you there. In this pragmatic, down-to-earth guide, you'll see how applying the core principles of functional programming can result in software designs that model real-world requirements both elegantly and concisely - often more so than an object-oriented approach. Practical examples in the open-source F# functional language, and examples from familiar business domains, show you how to apply these techniques to build software that is business-focused, flexible, and high quality. Domain-driven design is a well-established approach to designing software that ensures that domain experts and developers work together effectively to create high-quality software. This book is the first to combine DDD with techniques from statically typed functional programming. This book is perfect for newcomers to DDD or functional programming - all the techniques you need will be introduced and explained. Model a complex domain accurately using the F# type system, creating compilable code that is also readable documentation---ensuring that the code and design never get out of sync. Encode business rules in the design so that you have "compile-time unit tests," and eliminate many potential bugs by making illegal states unrepresentable. Assemble a series of small, testable functions into a complete use case, and compose these individual scenarios into a large-scale design. Discover why the combination of functional programming and DDD leads naturally to service-oriented and hexagonal architectures. Finally, create a functional domain model that works with traditional databases, NoSQL, and event stores, and safely expose your domain via a website or API. Solve real problems by focusing on real-world requirements for your software. What You Need: The code in this book is designed to be run interactively on Windows, Mac and Linux.You will need a recent version of F# (4.0 or greater), and the appropriate .NET runtime for your platform.Full installation instructions for all platforms at fsharp.org.
Get a practical, hands-on introduction to the Haskell language, its libraries and environment, and to the functional programming paradigm that is fast growing in importance in the software industry. This book contains excellent coverage of the Haskell ecosystem and supporting tools, include Cabal and Stack for managing projects, HUnit and QuickCheck for software testing, the Spock framework for developing web applications, Persistent and Esqueleto for database access, and parallel and distributed programming libraries. You'll see how functional programming is gathering momentum, allowing you to express yourself in a more concise way, reducing boilerplate, and increasing the safety of your code. Haskell is an elegant and noise-free pure functional language with a long history, having a huge number of library contributors and an active community. This makes Haskell the best tool for both learning and applying functional programming, and Practical Haskell takes advantage of this to show off the language and what it can do. What You Will Learn Get started programming with Haskell Examine the different parts of the language Gain an overview of the most important libraries and tools in the Haskell ecosystem Apply functional patterns in real-world scenarios Understand monads and monad transformers Proficiently use laziness and resource management Who This Book Is For Experienced programmers who may be new to the Haskell programming language. However, some prior exposure to Haskell is recommended.
Agda is an advanced programming language based on Type Theory. Agda's type system is expressive enough to support full functional verification of programs, in two styles. In external verification, we write pure functional programs and then write proofs of properties about them. The proofs are separate external artifacts, typically using structural induction. In internal verification, we specify properties of programs through rich types for the programs themselves. This often necessitates including proofs inside code, to show the type checker that the specified properties hold. The power to prove properties of programs in these two styles is a profound addition to the practice of programming, giving programmers the power to guarantee the absence of bugs, and thus improve the quality of software more than previously possible. Verified Functional Programming in Agda is the first book to provide a systematic exposition of external and internal verification in Agda, suitable for undergraduate students of Computer Science. No familiarity with functional programming or computer-checked proofs is presupposed. The book begins with an introduction to functional programming through familiar examples like booleans, natural numbers, and lists, and techniques for external verification. Internal verification is considered through the examples of vectors, binary search trees, and Braun trees. More advanced material on type-level computation, explicit reasoning about termination, and normalization by evaluation is also included. The book also includes a medium-sized case study on Huffman encoding and decoding.
The Formal Semantics of Programming Languages provides the basic mathematical techniques necessary for those who are beginning a study of the semantics and logics of programming languages. These techniques will allow students to invent, formalize, and justify rules with which to reason about a variety of programming languages. Although the treatment is elementary, several of the topics covered are drawn from recent research, including the vital area of concurency. The book contains many exercises ranging from simple to miniprojects.Starting with basic set theory, structural operational semantics is introduced as a way to define the meaning of programming languages along with associated proof techniques. Denotational and axiomatic semantics are illustrated on a simple language of while-programs, and fall proofs are given of the equivalence of the operational and denotational semantics and soundness and relative completeness of the axiomatic semantics. A proof of Godel's incompleteness theorem, which emphasizes the impossibility of achieving a fully complete axiomatic semantics, is included. It is supported by an appendix providing an introduction to the theory of computability based on while-programs. Following a presentation of domain theory, the semantics and methods of proof for several functional languages are treated. The simplest language is that of recursion equations with both call-by-value and call-by-name evaluation. This work is extended to lan guages with higher and recursive types, including a treatment of the eager and lazy lambda-calculi. Throughout, the relationship between denotational and operational semantics is stressed, and the proofs of the correspondence between the operation and denotational semantics are provided. The treatment of recursive types - one of the more advanced parts of the book - relies on the use of information systems to represent domains. The book concludes with a chapter on parallel programming languages, accompanied by a discussion of methods for specifying and verifying nondeterministic and parallel programs.
Expert F# 4.0
Programming in Martin-Löf's Type Theory
An Introduction
DSLs in Action
Explore functional and reactive programming to create robust and testable TypeScript applications
Practical Aspects of Declarative Languages

Haskell Programming makes Haskell as clear, painless, and practical as it can be, whether you're a beginner or an experienced hacker. Learning Haskell from the ground up is easier and works better. With our exercise-driven approach, you'll build on previous chapters such that by the time you reach the notorious Monad, it'll seem trivial.
Haskell is the world's leading lazy functional programming language, widely used for teaching, research, and applications. The language continues to develop rapidly, but in 1998 the community decided to capture a stable snapshot of the language: Haskell 98. All Haskell compilers support Haskell 98, so practitioners and educators alike have a stable base for their work.This book constitutes the agreed definition of Haskell 98, both the language itself and its supporting libraries, and should be a standard reference work for anyone involved in research, teaching, or application of Haskell.
Summary Functional Programming in C++ teaches developers the practical side of functional programming and the tools that C++ provides to develop software in the functional style. This in-depth guide is full of useful diagrams that help you understand FP concepts and begin to think functionally. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Well-written code is easier to test and reuse, simpler to parallelize, and less error prone. Mastering the functional style of programming can help you tackle the demands of modern apps and will lead to simpler expression of complex program logic, graceful error handling, and elegant concurrency. C++ supports FP with templates, lambdas, and other core language features, along with many parts of the STL. About the Book Functional Programming in C++ helps you unleash the functional side of your brain, as you gain a powerful new perspective on C++ coding. You'll discover dozens of examples, diagrams, and illustrations that break down the functional concepts you can apply in C++, including lazy evaluation, function objects and invokables, algebraic data types, and more. As you read, you'll match FP techniques with practical scenarios where they offer the most benefit. What's inside Writing safer code with no performance penalties Explicitly handling errors through the type system Extending C++ with new control structures Composing tasks with DSLs About the Reader Written for developers with two or more years of experience coding in C++. About the Author Ivan Čukić is a core developer at KDE and has been coding in C++ since 1998. He teaches modern C++ and functional programming at the Faculty of Mathematics at the University of Belgrade. Table of Contents Introduction to functional programming Getting started with functional programming Function objects Creating new functions from the old ones Purity: Avoiding mutable state Lazy evaluation Ranges Functional data structures Algebraic data types and pattern matching Monads Template metaprogramming Functional design for concurrent systems Testing and debugging
The book provides a description of the Standard ML (SML) Basis Library, the standard library for the SML language. For programmers using SML, it provides a complete description of the modules, types and functions composing the library, which is supported by all conforming implementations of the language. The book serves as a programmer's reference, providing manual pages with concise descriptions. In addition, it presents the principles and rationales used in designing the library, and relates these to idioms and examples for using the library. A particular emphasis of the library is to encourage the use of SML in serious system programming. Major features of the library include I/O, a large collection of primitive types, support for internationalization, and a portable operating system interface. This manual will be an indispensable reference for students, professional programmers, and language designers.
Tackle Software Complexity with Domain-Driven Design and F#
The Revised Report
Hands-On Functional Programming with TypeScript
The Standard ML Basis Library
Haskell 98 Language and Libraries
Pearls of Functional Algorithm Design

Objective Caml (OCaml) is an open source programming language that utilizes both functional and object oriented programming. Practical OCaml teaches Objective Caml in a straightforward manner, teaching all the features of this functional programming language by example. You will learn how to utilize OCaml to create a simple database, do reporting, and create a spam filter. You will also learn how to do complex log file scanning, create your own network servers by creating a ShoutCast server, and create a web crawler. By the book's conclusion, you will be well on your way to creating your own applications with OCaml.
Thinking Functionally with Haskell