1. Record Nr. UNINA9910464206303321

Autore	Hebert Fred <1988->
Titolo	Learn you some Erlang for great good! [[electronic resource]] : a beginner's guide / / Fred Hebert ; [foreword by Joe Armstrong]
Pubbl/distr/stampa	San Francisco, : No Starch Press, 2013
ISBN	1-59327-504-8
Edizione	[1st edition]
Descrizione fisica	1 online resource (627 p.)
Disciplina	005.13/3
Soggetti	ERLANG (Computer program language) Electronic books.
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	First printing. Includes index.
Nota di contenuto	Intro -- Learn You Some Erlang For Great Good! -- Foreword -- Preface -- To the Foreigner -- To the Erlang Regular -- To the Person Who Has Read This Online -- Acknowledgments -- Introduction -- So What's Erlang? -- Don't Drink Too Much Kool-Aid -- What You Need to Dive In -- Where to Get Help -- 1. Starting Out -- Using the Erlang Shell -- Entering Shell Commands -- Exiting the Shell -- Some Erlang Basics -- Numbers -- Invariable Variables -- Atoms -- Boolean Algebra and Comparison Operators -- Tuples -- Lists -- List Comprehensions -- Working with Binary Data -- Bit Syntax -- Bitwise Binary Operations -- Binary Strings -- Binary Comprehensions -- 2. Modules -- What Are Modules? -- Creating Modules -- Compiling Code -- Compiler Options -- Defining Macros -- More About Modules -- Metadata -- Circular Dependencies -- 3. Syntax in Functions -- Pattern Matching -- Fancier Patterns -- Variables in a Bind -- Guards, Guards! -- What the If ?! -- In case ... of -- Which Should We Use? -- 4. Types (or Lack Thereof) -- Dynamite-Strong Typing -- Type Conversions -- To Guard a Data Type -- For Type Junkies -- 5. Hello Recursion! -- How Recursion Works -- Length of a List -- Length of a Tail Recursion -- More Recursive Functions -- A Duplicate Function -- A Reverse Function -- A Sublist Function -- A Zip Function -- Quick, Sort! -- More Than Lists -- Thinking Recursively -- 6. Higher-Order Functions -- Let's Get Functional -- Anonymous Functions -- More Anonymous Function

Power -- Function Scope and Closures -- Maps, Filters, Folds, and More -- Filters -- Fold Everything -- More Abstractions -- 7. Errors and Exceptions -- A Compilation of Errors -- Compile-Time Errors -- No, YOUR Logic Is Wrong! -- Runtime Errors -- Function Clause Errors -- Case Clause Errors -- If Clause Errors -- Bad Match Errors -- Bad Argument Errors -- Undefined Function Errors.
Bad Arithmetic Errors -- Bad Function Errors -- Bad Arity Errors -- System Limit Errors -- Raising Exceptions -- Error Exceptions -- When Not to Use Errors -- Custom Errors -- Exit Exceptions -- Throw Exceptions -- Dealing with Exceptions -- Handling Different Types of Exceptions -- After the Catch -- Trying Multiple Expressions -- Wait, There's More! -- Try a try in a Tree -- 8. Functionally Solving Problems -- Reverse Polish Notation Calculator -- How RPN Calculators Work -- Creating an RPN Calculator -- Testing the Code -- Heathrow to London -- Solving the Problem Recursively -- Writing the Code -- Running the Program Without the Erlang Shell -- 9. A Short Visit to Common Data Structures -- Records -- Defining Records -- Reading Values from Records -- Updating Records -- Sharing Records -- Key/Value Stores -- Stores for Small Amounts of Data -- Proplists -- Orddicts -- Larger Dictionaries: Dicts and GB Trees -- A Set of Sets -- Directed Graphs -- Queues -- End of the Short Visit -- 10. The Hitchhiker's Guide to Concurrency -- Don't Panic -- Concurrency Concepts -- Scalability -- Fault Tolerance -- Concurrency Implementation -- Not Entirely Unlike Linear Scaling -- So Long and Thanks for All the Fish! -- Spawning Processes -- Sending Messages -- Receiving Messages -- 11. More on Multiprocessing -- State Your State -- We Love Messages, But We Keep Them Secret -- Time Out -- Selective Receives -- The Pitfalls of Selective Receives -- More Mailbox Pitfalls -- 12. Errors and Processes -- Links -- It's a Trap! -- Old Exceptions, New Concepts -- Exceptions and Traps -- exit/2 Changes Everything -- Killing Me (Not So) Softly -- Monitors -- Naming Processes -- 13. Designing a Concurrent Application -- Understanding the Problem -- Defining the Protocol -- Lay Them Foundations -- An Event Module -- Events and Loops -- Adding An Interface -- The Event Server.
Handling Messages -- Hot Code Loving -- I Said, Hide Your Messages -- A Test Drive -- Adding Supervision -- Namespaces (or Lack Thereof) -- 14. An Introduction to OTP -- The Common Process, Abstracted -- The Basic Server -- Introducing the Kitty Server -- Generalizing Calls -- Generalizing the Server Loop -- Starter Functions -- Generalizing Kitty Server -- Specific vs. Generic -- Callback to the Future -- The init Function -- The handle_call Function -- The handle_cast Function -- The handle_info Function -- The terminate Function -- The code_change Function -- .BEAM Me Up, Scotty! -- 15. Rage Against the Finite-State Machines -- What Is a Finite-State Machine? -- Generic Finite-State Machines -- The init Function -- The StateName Function -- The handle_event Function -- The handle_sync_event Function -- The code_change and terminate Functions -- A Trading System Specification -- Show Me Your Moves -- Defining the State Diagrams and Transitions -- Game Trading Between Two Players -- The Public Interface -- FSM-to-FSM Functions -- The gen_fsm Callbacks -- That Was Really Something -- Fit for the Real World? -- 16. Event Handlers -- Handle This! *pumps shotgun* -- Generic Event Handlers -- The init and terminate Functions -- The handle_event Function -- The handle_call Function -- The handle_info Function -- The code_change Function -- It's Curling Time! -- The Scoreboard -- Game Events -- Alert the Press! -- 17. Who Supervises the Supervisors? -- Supervisor Concepts -- Using Supervisors -- Restart Strategies -- one_for_one -- one_for_all -- rest_for_one -- simple_one_for_one -- Restart Limits --

Child Specifications -- ChildId -- StartFunc -- Restart -- Shutdown -- Type -- Modules -- Band Practice -- Musicians -- Band Supervisor -- Dynamic Supervision -- Using Standard Supervisors Dynamically -- Using a simple_one_for_one Supervisor.
18. Building an Application -- A Pool of Processes -- The Onion Layer Theory -- A Pool's Tree -- Implementing the Supervisors -- Working on the Workers -- Writing a Worker -- Run Pool Run -- Cleaning the Pool -- 19. Building Applications the OTP Way -- My Other Car Is a Pool -- The Application Resource File -- Converting the Pool -- The Application Behavior -- From Chaos to Application -- Library Applications -- 20. The Count of Applications -- From OTP Application to Real Application -- The Application File -- The Application Callback Module and Supervisor -- The Dispatcher -- Returning Results through CPS -- Dispatching and Receiving -- The Counter -- Run App Run -- Included Applications -- Complex Terminations -- 21. Release Is the Word -- Fixing the Leaky Pipes -- Terminating the VM -- Updating the Application Files -- Compiling the Applications -- Releases with systools -- Creating a Boot File -- Packaging the Release -- Releases with Reltool -- Reltool Options -- Release-Only Options -- Release-wide and Application-wide Options -- Module-Specific Options -- All-levels Options -- That's Dense -- Reltool Recipes -- Development Versions -- Importing or Exporting Only Part of a Library -- Smaller Apps for Programmers with Big Hearts -- Released from Releases -- 22. Leveling Up in the Process Quest -- The Hiccups of Appups and Relups -- The Ninth Circle of Erl -- Process Quest -- The regis-1.0.0 Application -- The processquest-1.0.0 Application -- The sockserv-1.0.0 Application -- The Release -- Making Process Quest Better -- Updating code_change Functions -- Adding Appup Files -- Upgrading the Release -- Relup Review -- 23. Buckets of Sockets -- IO Lists -- UDP and TCP: Bro-tocols -- UDP Sockets -- TCP Sockets -- More Control with Inet -- Sockserv, Revisited -- Where to Go from Here? -- 24. EUnited Nations Council -- EUnit-What's an EUnit?.
Test Generators -- Fixtures -- More Test Control -- Test Documentation -- Testing Regis -- He Who Knits EUnits -- 25. Bears, ETS, Beets: In-Memory NoSQL for Free! -- Why ETS -- The Concepts of ETS -- ETS Phone Home -- Creating and Deleting Tables -- Inserting and Looking Up Data -- Meeting Your Match -- You Have Been Selected -- DETS -- A Little Less Conversation, a Little More Action, Please -- The Interface -- Implementation Details -- 26. Distribunomicon -- This Is My Boomstick -- Fallacies of Distributed Computing -- The Network Is Reliable -- There Is No Latency -- Bandwidth Is Infinite -- The Network Is Secure -- Topology Doesn't Change -- There Is Only One Administrator -- Transport Cost Is Zero -- The Network Is Homogeneous -- Fallacies in a Nutshell -- Dead or Dead-Alive -- My Other Cap Is a Theorem -- Consistency -- Availability -- Partition Tolerance -- Zombie Survivors and CAP -- Setting Up an Erlang Cluster -- Through the Desert on a Node with No Name -- Connecting Nodes -- More Tools -- Cookies -- Remote Shells -- Hidden Nodes -- The Walls Are Made of Fire, and the Goggles Do Nothing -- The Calls from Beyond -- The net_kernel Module -- The global Module -- The rpc Module -- Burying the Distribunomicon -- 27. Distributed OTP Applications -- Adding More to OTP -- Taking and Failing Over -- The Magic 8 Ball -- Building the Application -- The Supervisor Module -- The Server Module -- Making the Application Distributed -- 28. Common Test for Uncommon Tests -- What Is Common Test? -- Common Test Structure -- Creating a Simple Test Suite -- Running the Tests -- Testing with State -- Test Groups -- Defining Test Groups -- Test Group Properties -- The Meeting Room -- Test Suites Redux --

	Test Specifications -- Specification File Contents -- Creating a Spec File -- Running Tests with a Spec File -- Large-Scale Testing. Creating a Distributed Spec File.
Sommario/riassunto	Learn You Some Erlang for Great Good! is a hilariously illustrated guide to the concurrent functional programming language. As you laugh along with Hebert's brilliantly quirky drawings, you'll effortlessly pick up this complex language and have fun while you're at it..