Book review

Review of "Learn you some Erlang for great good! A beginner's guide", by Fred Hébert, No Starch Press, 2013, £26.80 (paperback), ISBN: 978-1-59327-435-1
doi:10.1017/S0956796815000234

This book is marketed as "a beginners guide" to Erlang and it does do a brilliant job at teaching the basics; however through a perfect blend of humour and nitty-gritty details, Hébert has written an outstanding guide to learning Erlang. This is a book written for people that have heard of Erlang or the power of functional languages, but have not yet "dipped their toes into" it. Throughout the book, there are sections titled "Don't drink too much kool-aid" which are used to explore the common misconceptions and pitfalls that are often encountered when learning Erlang. As a newcomer to the language, I found these sections particularly interesting because Hébert used the sections to be very forthcoming about the advantages and disadvantages of the language.

The production quality of the book is high, with illustrations and sketched diagrams on most pages. It is part of the same series as "Learn you a Haskell for Great Good" and so has a brightly coloured front page with a purple multitasking octopus that sets the tone of the entire book: fun and packed full of knowledge. The inner pages are all black and white, supposedly to keep the costs down, but it is laid out in such a way that the lack of colour is really not an issue, the illustrations are just as entertaining in black and white. At just under 600 pages long, it is not a quick read but Hébert's conversational style of writing makes it easy and fun to read. As the book is fairly large, it is often not suitable for travelling so I found it very useful that the entire book is freely available online at http://learnyousomeerlang.com/content.

As a textbook, it has a good order of topics and a logical flow in how it explains the concepts and syntax. For example, in the first chapter it starts with using the shell as a calculator and quickly gets into explaining how variables cannot be changed and that atoms are the same value as their name. From this point, he is able to quickly explain pattern matching within tuples because of the way he already explained the nature of variables in Erlang. It also has a good approach to learning, as each section has parts (such as the kool-aid sections) that the reader can either skip or read to gain more insight as they wish. Many books for beginners tend either to stay at a very basic level or to have a steep learning curve. In this book, each chapter has a smooth learning curve, in the same way he quickly went from variables to pattern matching, taking the reader from the absolute basics all the way through to unit testing, sockets and the organisation of large programs. Throughout the book, Hébert focuses on explaining the concepts and makes it feel that the syntax that is scary to a beginner is actually just a quirk of the language. There are no exercises for the reader to do alone, but there are many small little examples that make sure the reader understands the concepts. This constant encouragement and focus on the concepts makes the language far easier to learn. That said, it may not be for everybody. This book is conversational and has a narrative from beginning to end, it is meant to be read all the way through, therefore it is not quite so good as a reference guide. The index pages do include syntax from operators to binary strings and match specifications, and this makes it easy to dip into the book when you have forgotten the quirky syntax.

Once the basics are taught, Hébert goes through a number of small projects such as a Reverse Polish Notation calculator and Heathrow to London, which is essentially a case of finding the shortest path in a weighted graph although Hébert keeps the difficulty low and does not call it that. These are the same projects as in Lipovača's Learn You a Haskell for Great Good, Hébert explains that the reason for this is to show how it is relatively easy to use the same functional concepts between languages. Shortly after these projects, Hébert starts teaching the theory of concurrency and explores how to use the functional concepts already taught alongside the concepts of concurrency. The concurrency chapters are at quite a high level, so it does feel like the same book but it may throw some people off. Ideally, the book could have had a couple of small projects that make use of the concepts in the same way that the Reverse Polish Notation calculator and Heathrow to London were good small examples of the functional concepts. Throughout the book, and particularly in these chapters, he often enforces the idea of Fault Tolerance and encourages code to be written in a way that expects errors instead failing because of them. The book is for beginners but it gets into some advanced areas, particularly Erlang's frameworks such as the Open Telecom Platform (OTP) and Dialyzer, a static analysis tool. Despite the difficulty, Hébert makes it feel like a perfectly smooth transition from absolute basics to complicated real world Erlang.

This book targets its audience perfectly. It is about 600 pages, full to the brim with knowledge and useful insights into how Erlang was designed as a language, as well as being fun and easy to read. The cover makes it stand out on the bookshelf and it is organised in such a way that makes you want to read it in a single sitting. Each chapter extends upon the previous, purposely enhancing the reader's ability to write real world applications. For anybody who wants to add Erlang to their repertoire, Learn You Some Erlang for Great Good is the book to get.

SHANE HUDSON
Canterbury, UK

https://doi.org/10.1017/S0956796815000234 Published online by Cambridge University Press