Foreword

This booklet is based on a challenge Will E. Byrd set himself in January 2024: to overcome procrastination by committing to writing and publishing 11 books and 1000 YouTube videos during the calendar year.

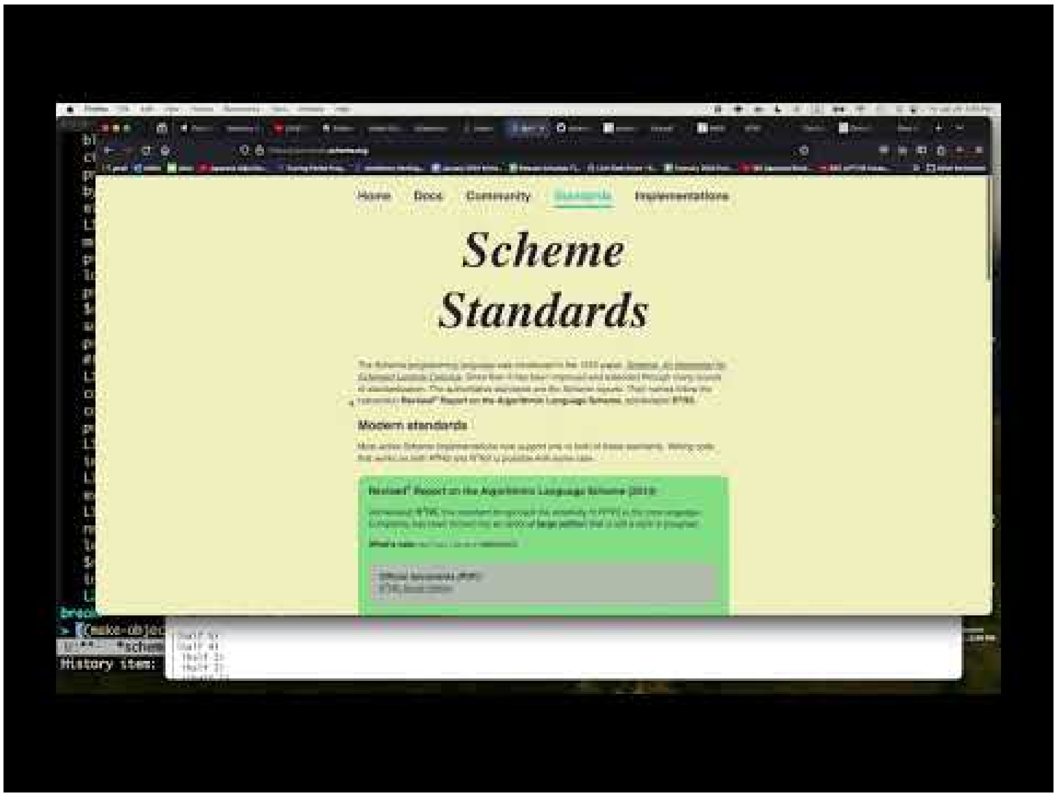
Shownotes

January 2024

January 26th - Scheme Learning Resources

In which Will Byrd shows us some starting points for diving into his favourite programming language

January 26th 2024: Scheme Learning Resources



Intro

Welcome to Episode IV of **Will's Guide**. This year, I've decided to make 1,024 videos - I've upped the number from a thousand, because I think two to the 10 or one kiloTube of videos is more fun!

I've also set up the Discord server, and we've had about a dozen people join. If you're interested in joining the Discord server *The Imperishable Wonderland of Infinite Fun*, please email me.

Today, I'd like to talk about something close to my heart: Scheme, a language which I love. And I'm just going to talk a little bit about how you might learn Scheme. I'm going to have a whole bunch of videos on Scheme (I hope), but how do you learn about Scheme? What are the resources available? Let's just talk about a few of them I found useful..

The Little Schemer by Dan Friedman and Matthias Felleisen

- The Little Schemer (MIT Press)
- Dan's Wikipedia Entry}}
- Felleisen.org Site}} and Wikipedia

One is called The Little Schemer by Daniel P Friedman{{footnote: and Matthias Felleisen{{footnote: . Dan was my adviser at Indiana University for my PhD, and he was also Matthias's adviser. You also might know Matthias from the Racket world, so you have two experts guiding you.

This book is really about thinking computationally, and specifically thinking about *recursion* - recursion over lists and trees - those sorts of things, and also a little interpreter work. So, if you think is is going to be a book that's a big thick manual on Scheme - the programming language and the semantics or whatever - well, you might be disappointed. But this really in another way is at the heart of thinking about Scheme computationally and thinking about computation and recursion, so it's a great book to affect your thinking and make sure that you're really comfortable with ideas about recursion.

So this is recommended by me if you want to learn how to think recursively at least a certain way of thinking recursively. If you're not comfortable with recursion, if you read this book and if you learn Scheme in general, you're likely to become very comfortable with recursion, because Scheme is really based on recursion. There's no loop constructs like there are in most languages (or anything does look like a loop, it's actually tail recursion which we'll talk about later)..

So it's sort of the secret that Scheme doesn't really have loops in the way most languages do, although it turns out you can Implement Loops that are equivalent to a certain type of recursion..

The Structure and Interpretation of Computer Programs by Hal Abelson, Jerry and Julie Sussman

- The Structure and Interpretation of Computer Programs

Another great resource is the Structure and Interpretation of Computer Programs by Harold Abelson and Gerald J Sussman with Julie Sussman. this is very famous as maybe the 2:44 greatest textbook in the history of computer science was used as the introductory textbook at MIT for many 2:52 years it has all sorts of interesting ideas about computation and interpreters 2:57 how to write interpreters interpreters how you write a program that interprets another program those sorts of ideas 3:04 computational models great book full of deep fascinating ideas once again not a 3:11 manual on scheme scheme is introduced as needed as a computational notation which 3:17 really gets back to the heart of lisp the lisp family of languages you you 3:23 need to have some sort of notation to talk about computation so let's have a 3:29 very elegant simple minimal notation that's compositional has nice properties 3:35 and we can think about that's at the heart of scheme and that's one reason I really like 3:40 scheme there's a series of videos based on the ideas in the books called the 3:46 sicp lectures and if you're interested in the book you 3:53 might like these videos I will say that the videos can be in the book can be a 3:59 little mathy especially in terms of the examples um you know they aim towards 4:05 the MIT you know entering class or you know in this case um for these lectures I 4:14 don't know if these are supposed to be more experienced programmers or or whatever but um you know it can turn 4:21 people off a little bit so if you find that this is a little Hardo maybe start 4:27 with something like the little schemer first which is really written for the bright high school student but the ideas 4:33 are very deep and powerful and can you know can benefit anyone U so maybe start with something 4:39 like the little schemer and if you feel that's not enough you could try the reason schemer which is a much harder 4:46 book um but in any case you could build yourself up towards the ideas in sicp 4:52 over time

MIT 6.001 1986 Lecture Series

- MIT 6.001 1986 Lectures on YouTube

Software Design for Flexibility by Chris Hanson and Jay Sussman

- Software Design for Flexibility

Simply Scheme: Introducing Computer Science by Brian Harvey and Matthew Wright

- Simply Scheme: Introducing Computer Science

Scheme and Functional Programming Workshop

- Scheme and Functional Programming Workshop

Lambda the Ultimate

- Lambda the Ultimate (papers page)

Scheme.org

- Scheme.org

SchemeDoc Bibliography

- SchemeDoc Bibliography

R5RS Page

- R5RS Page
- R5RS Standard PDF

R6RS

- R6RS.org

R7RS

- R7RS.org

The Scheme Programming Language by R. Kent Dybvig

- The Scheme Programming Language

The Chez Scheme User's Guide

- The Chez Scheme User's Guide

Chez Scheme on Github

- Chez Scheme on Github
- Chez Scheme Github Repo

Scheme Related Papers by CollectRobot

- Scheme Related Papers by CollectRobot
- How to Debug Scheme Programs
- Generation-Friendly Eq Hash Tables

Appendix A - Dev Notes for this Booklet Project

To upload my repo to Github, I ran:

```
$ gh repo create
```

To create a `gh-pages` branch in Github, I followed this advice:

```
git checkout --orphan gh-pages
git reset --hard
git commit --allow-empty -m "Initializing gh-pages branch"
git push origin gh-pages
git checkout main # was master
```

I added entries to the Makefile provided here

Now, I run

```
git commit -a -m "publish"
git push
make deploy
```