Your First Haskell Terminal Application

A Gentle Project-Based Introduction to Haskell

Rebecca Skinner

<2022-11-08 Tue>

Prelude

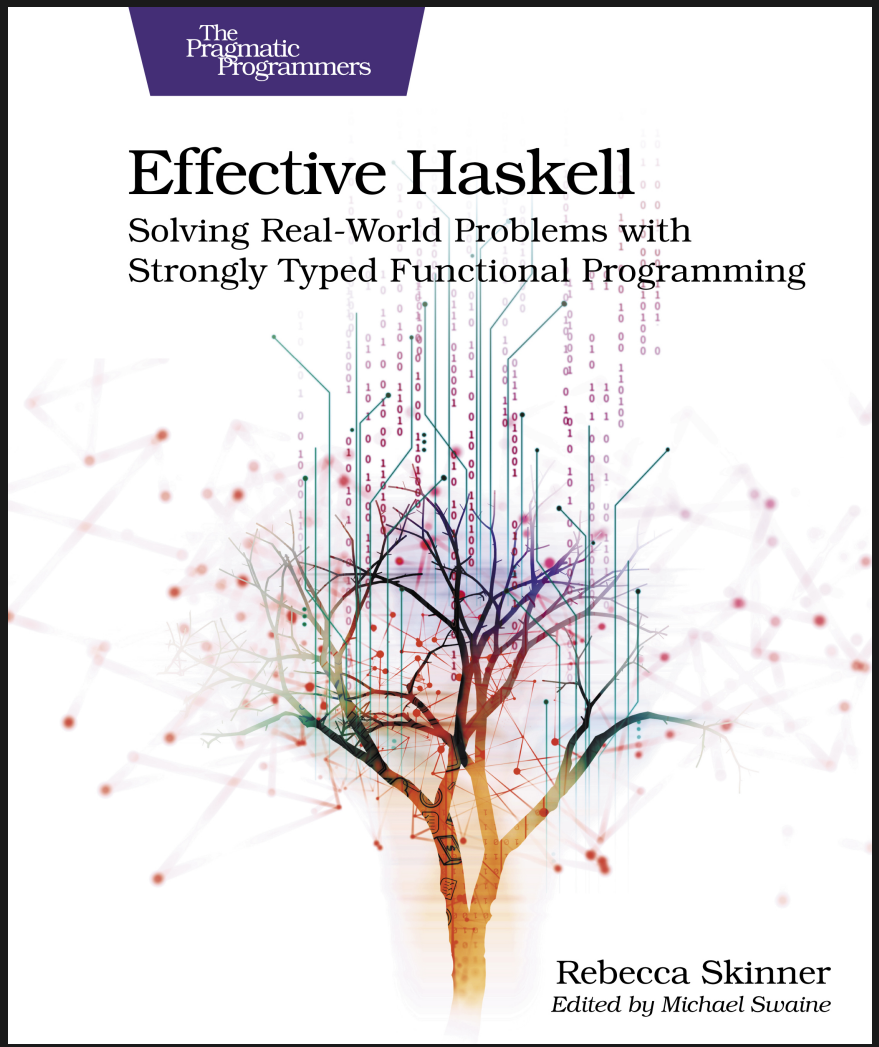
Hello, World

- About Me: Rebecca Skinner
  - Lead Software Engineer at Mercury
  - Author of Effective Haskell
- @cercerilla on Twitter and Cohost
- https://rebeccaskinner.net
- https://github.com/rebeccaskinner/

About This Talk

- This talk is inspired by the first half of Effective Haskell.
- We'll focus on one happy path, there are tools and libraries that are good but won't be covered.
- This talk will give you a "lay of the land" but we'll leave out some details.

Effective Haskell



https://tinyurl.com/2744kfu7

Now in Beta!

Build Your Tools!

Build Your Tools!

HCat: A Haskell Pager

A pager is a command line tool that lets you view long documents one page at a time. Some examples you might be familiar with incldue:

HCat: A Haskell Pager

A pager is a command line tool that lets you view long documents one page at a time. Some examples you might be familiar with incldue:

- less
- more
- bat

HCat: A Haskell Pager

A pager is a command line tool that lets you view long documents one page at a time. Some examples you might be familiar with incldue:

- less
- more
- bat

We're going to build our own pager in Haskell. I'm calling mine hcat.

HCat: A Screenshot

```
{-# LANGUAGE RecordWildCards #-}
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE LambdaCase #-}
{-# LANGUAGE TypeApplications #-}
module HCat where

import qualified System.IO.Error as IOError
import qualified Control.Exception as Exception

import qualified System.Environment as Environment
import qualified System.Exit as Exit
import qualified Data.List as List
import qualified Data.Maybe as Maybe
import qualified Data.Char as Char
import qualified Text.Printf as Printf
import qualified Data.Text as Text
import qualified Data.Text.IO as TextIO
import qualified Data.ByteString as BS
import System.IO ( openFile
                 , hPutStr
                 , hFlush
                 , hClose
                 , hFileSize
                 , hGetContents
                 , stdin
                 , stdout
                 , hGetChar
                 , hSetBuffering
                 , hGetBuffering
                 , hSetEcho
                 , BufferMode (..)
                 , Handle
                 , IOMode (..) )
import qualified System.Info
import qualified System.Process as Process
import Text.Read
import Control.Monad
import qualified Data.Time.Clock as Clock
import qualified Data.Time.Format as TimeFormat

-- import qualified System.Posix.User as PosixUser
import qualified System.Directory as Directory


--START:FileInfo
./src/HCat.hs | permissions: rw- | 19337 bytes | modified: 2022-01-25 04:15:03 | page: 1 of 15
```

Re-Inventing a Thousand Wheels

Why write something boring like a pager?

Re-Inventing a Thousand Wheels

Why write something boring like a pager?

- Teaches you how to do basic IO and work with the local system

Re-Inventing a Thousand Wheels

Why write something boring like a pager?

- Teaches you how to do basic IO and work with the local system
- Implementing word wrap and pagination will help you think about problems in a functional way

Re-Inventing a Thousand Wheels

Why write something boring like a pager?

- Teaches you how to do basic IO and work with the local system
- Implementing word wrap and pagination will help you think about problems in a functional way
- Building something similar to tools you use ever day will help you make good decisions, and give you something you can benefit from

Choosing Haskell for Developer Tooling

Why choose Haskell to build your developer tooling?

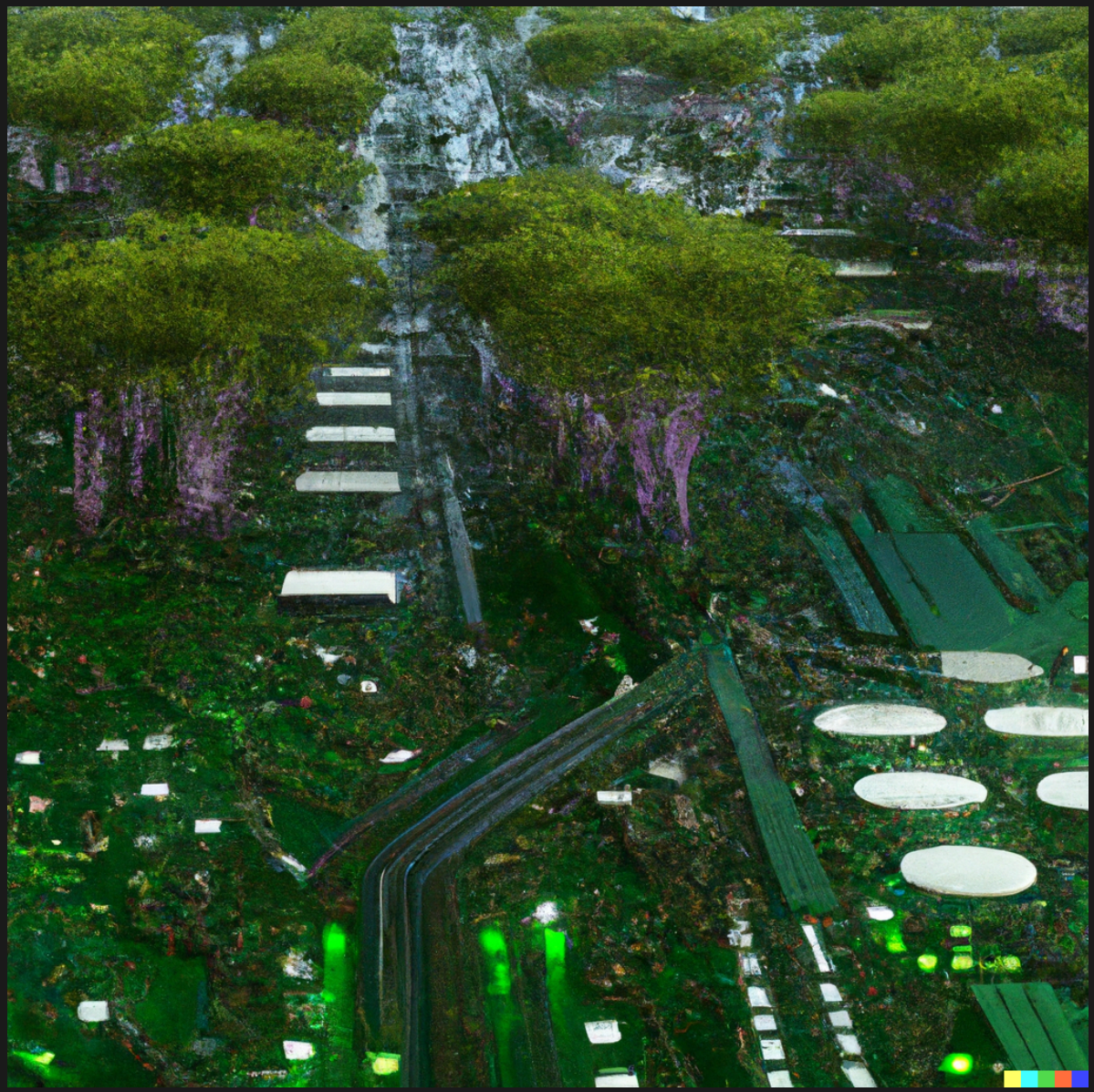
Choosing Haskell for Developer Tooling

Why choose Haskell to build your developer tooling?

- You want to learn Haskell, and building things for yourself is a good way to learn
- Haskell is a compiled language, so you don't have to deal with the startup times of a JIT language like Java
- Haskell is garbage collected (fewer memory errors)
- Haskell is statically typed so you'll have fewer runtime errors
- Haskell has good performance
  - Frequently, but not always slower than C and C++ programs
  - About as fast as Java programs while using a bit less memory, and having no JIT warmup cost
  - Nearly always faster than Python programs

Setting Up Your Environment

Setting Up Your Environment



Operating System

Haskell works on all of the major operating systems.

- Linux
  - x86 and arm. Best on x86
- macOS
  - Intel and Apple hardware are well supported
- Windows
  - x86. Native and WSL supported, but best with WSL.

Operating System

Haskell works on all of the major operating systems.

- Linux
  - x86 and arm. Best on x86
- macOS
  - Intel and Apple hardware are well supported
- Windows
  - x86. Native and WSL supported, but best with WSL.

I use NixOS. Haskell works great with NixOS, but it's a steep learning curve.

Operating System

Installing Haskell

The best way to install Haskell is with ghcup:
https://www.haskell.org/ghcup/

Installing Haskell

The best way to install Haskell is with ghcup:
https://www.haskell.org/ghcup/

- Managing Haskell with nixpkgs works well, but is a steep learning curve.
- Avoid using linux distro packages, homebrew, etc.
- Installing Haskell with Stack is no longer recommended, but you can install stack with ghcup if you like.

Configuring Your Editor

VSCode is the most popular editor for working with Haskell. It will help you set up additional tooling and offers the most IDE-like experience. Emacs and vim also have good Haskell support, and can be configured with more IDE like features if you want.

Linting

hlint is the most popular Haskell linter by far. You should try to use hlint when you are developing.

- hlint will sometimes tell you things that you might not have learned yet, try not to worry about that too much.
- hlint sometimes makes bad suggestions, but most of the time it's suggestions are good and will help you learn to write better Haskell programs.
- hlint is very configurable, so you can tune it if there are things that you find annoying.

Pretty Printing

There are a lot of different pretty printers for Haskell. None of them do a perfect job, and they'll all make your code worse on occasion, so you should treat them as a tool to help you format your code, rather than something that should be enforced.

Pretty Printing

There are a lot of different pretty printers for Haskell. None of them do a perfect job, and they'll all make your code worse on occasion, so you should treat them as a tool to help you format your code, rather than something that should be enforced.

- Fourmolu is modestly configurable, and tends to work reliably. It's also fairly aggressive and is more likely to uglify your code.
- Stylish Haskell tends to lag a bit in supporting newer GHC versions, and can be a little flaky, but is less intrusive and it's style is a bit more idiomatic to classic haskellers

Pretty Printing

There are a lot of different pretty printers for Haskell. None of them do a perfect job, and they'll all make your code worse on occasion, so you should treat them as a tool to help you format your code, rather than something that should be enforced.

- Fourmolu is modestly configurable, and tends to work reliably. It's also fairly aggressive and is more likely to uglify your code.
- Stylish Haskell tends to lag a bit in supporting newer GHC versions, and can be a little flaky, but is less intrusive and it's style is a bit more idiomatic to classic haskellers

There are a bunch of other options you can explore if you want

IDEs and Not-IDEs

Developing with Nix

Picking a GHC Version

Getting Help

Browse Documentation on Hackage

Search Libraries with Hoogle

Local Documentation

Getting Help From the Haskell Community

Starting a New Project

Haskell Packaging Overview

Cabal Projects

App, Src, and Test

Managing Dependencies

Putting the M in MVP

Read File, Write File

Zoom, Enhance, Refactor

That's No Moon: Working With IO

A Tale of Two Languages

Working With Text, and Other Hard Problems

A Tale of Three Text Types

String

ByteString

Text

Questions?