HASKELL GETS A JOB
HELPING EVERYONE GET A JOB!

HASKELL AT SEEK

SIMON FENTON

» SEEK

» Search stream

» Developer / Tech lead

» Build back-end batch and streaming services

@SCREAMISH

HASKELL AT SEEK

1. FP + me
2. FP + SEEK
3. Haskell + my team
4. Haskell + SEEK

FP + ME

FP + ME

- Unimelb CompSci (2000)
  - Haskell
  - Prolog

“HASKELL IS NOT A DIFFICULT LANGUAGE TO USE. HASKELL IS DIFFICULT TO TEACH EFFECTIVELY.”

Haskell Book

'ENTERPRISE SOFTWARE'

» a.k.a. The OOP Tar-pit

» NullReferenceException

» Shared mutable state

» So many bugs AND so much time testing (manual and automated)

PEOPLE SHOW ME ANOTHER WAY

» F# community in London

» (small) successes with F# at JustGiving

» Tools to move TB of user content into new content service

FP + ME

» Immutable

» Terse

» Types

FP + SEEK

FP + SEEK (2014)

» Some F# in production

» Gain adoption via tooling and tests

» Programming Languages - Coursera

FP + SEEK (2015)

» Speed to market / cheap(er) experiments

» Microservices & Devops

» 'Cambrian explosion' of PL

» F# - Scala - Clojure - Haskell - Go - Javascript (Node)

» Internal F# Workshop

» ... Docker?

HASKELL + MY TEAM

HASKELL + MY TEAM

» Sept 2015 - No[1] Haskell experience

» Lunchtime group starts NICTA FP Course

» Greenfield F# on Mono 😕

» Brownfield F# on Mono[2] 😭

[1] Besides my uni experience and light dabbling in between

[2] What's up with mono?

CONSTRAINTS & CONTEXT

» AWS

» 12-factor App[3]

» Docker

[3] The Twelve-Factor App

RECKONING

» Stay with F# and go back to Windows?

» Internal Hackathon right around the corner

» Clojure?

» Scala?

» Haskell!

A PLAN FOR HASKELL

» What does success look like?

» Vertical slice of current F# app

» AWS SDKs critical (S3, DynamoDB)

» IO (CSV files, HTTP, CLI args)

» Tests (correctness measures)

» Docker deploy a la 12-factor App

DIVIDE & CONQUER

» 2 on core domain

» 1 on input CSV from S3

» 1 on output to HTTP

THE PLEASURES

» Stack

SERIOUSLY, STACK IS AMAZING!

STACK[5]

Fresh dev laptop

```
> brew install haskell-stack
> git clone http://the-thing.git
> stack setup
> stack test
```

[5] Haskell Stack

CROSS-PLATFORM

» Windows and OS X dev envs

» Linux in Docker for production

» Holy moly those Docker build times

» Docker build-caching to the rescue

AMAZONKA (AWS)

```haskell
downloadFile :: Region      -- ^ Region to operate in.
             -> BucketName
             -> ObjectKey   -- ^ The source object key.
             -> FilePath    -- ^ The destination file to save as.
             -> IO ()
downloadFile r b k f = do
    lgr <- newLogger Debug stdout
    env <- newEnv r Discover <&> envLogger .~ lgr

    runResourceT . runAWST env $ do
        rs <- send (getObject b k)
        view gorsBody rs `sinkBody` CB.sinkFile f
```

WREQ (HTTP)

» Awesome doco! Wreq Tutorial

» HTTP "hello world" was super quick

» Internal-only, legacy APIs with 'interesting' auth protocols

» Can we handle going off-road?

WREQ - JSON PARSING

```
instance FromJSON Response where
  parseJSON (Object o) =
          Response <$> o .: "status"
                   <*> o .: "nextHref"
                   <*> o .: "id"
  parseJSON _ = mzero
```

WREQ BROUGHT A FRIEND

```
import Control.Lens hiding ((.=))
import Data.Aeson.Lens

allIds = values . key "id" . _Integer
```

LENS

```
allIds :: (Integer -> Const (Endo [Integer]) Integer)
       -> LB.ByteString
       -> Const (Endo [Integer]) LB.ByteString
allIds = values . key "id" . _Integer
```

LENS

```
allIds :: (Integer -> Const (Endo [Integer]) Integer)
       -> LB.ByteString
       -> Const (Endo [Integer]) LB.ByteString

allIds :: Traversal' LB.ByteString Integer
allIds = values . key "id" . _Integer
```

THE PAINS

» Regex[4]

» String vs Text vs ByteString

» Text formatting (Text.Printf vs text-format vs ???)

[4] Regular Expressions @ Haskell Wiki

HACKATHON RETRO

» No major concerns

» Loving it!

» Let's plot a course for total cut-over

FULL-TIME HASKELL

HTTP TESTING

» Oh, easy, composable DSLs with our Free Monads!!

» Freer monads, etc...

» Stop. We really don't get this.

» What do we do?

ISOLATING HTTP

```
class Monad m => MonadHttp c m | m -> c where
  get :: Url -> Options -> m ByteString
  getSigned :: Url -> SignedOptions -> m ByteString
  getJson :: (MonadThrow m, FromJSON a) => Url -> Options -> m a
  sign :: Url -> Options -> m SignedOptions

newtype HttpT c m a = HttpT (ReaderT c m a)
  deriving (Functor, Applicative, Monad, MonadIO, MonadThrow, MonadCatch)

instance (HttpContext c, MonadIO m) => MonadHttp c (HttpT c m) where

newtype MockHttpT c m a =
    MockHttpT (ReaderT (MockHttpContext c m)
                       (WriterT [RecordedRequest] m) a)
  deriving (Functor, Applicative, Monad, MonadIO,
            MonadThrow, MonadCatch)
```

HTTP - PACT TESTING

FREE MONAD BITES BACK

» The 'naive' Free Monad performs terribly

» Saved by the Church encoded Free Monad

PERFORMANCE

» Mostly amazing

» But hard for us (still) to reason about

» Space leaks 😞

SOME NICE SURPRISES

» Haskell for scripting

» #! thanks to stack

» Most scripts promoted to apps sharing a lib

» Much fewer tests (property tests > unit tests)

» Really easy to drop into an unfamiliar part of codebase

» No bugs! (so far)

HASKELL TAKES OVER

» Launched the Haskell v2 in dry-run, side-by-side with F# v1 in March

» v2 and v1 change places in April

» Shutdown v1 in May

» No surprises, smooth sailing all the way

HASKELL + SEEK

HASKELL 2016?

THEN

```
default-extensions:
    OverloadedStrings
```

NOW

```
default-extensions:
    OverloadedStrings
  , TupleSections
  , FlexibleContexts
  , RecordWildCards
  , ScopedTypeVariables
```

HASKELL RESOURCES WE LIKE

» Haskell is easy

» Stackage

» 24 Days of Hackage on conscientiousprogrammer.com

» Lens over tea

» Haskell for all

HASKELLBOOK.COM
HASKELL PROGRAMMING FROM FIRST PRINCIPLES

“HASKELL IS NOT A DIFFICULT LANGUAGE TO USE. HASKELL IS DIFFICULT TO TEACH EFFECTIVELY.”

Haskell Book

SPACED REPETITION AND ITERATIVE DEEPENING

HASKELL + SEEK

» It's in production, doing serious business

» Interest growing

» YOW! Lambda Jam 2016

» Compose (Melbourne)

THANK YOU