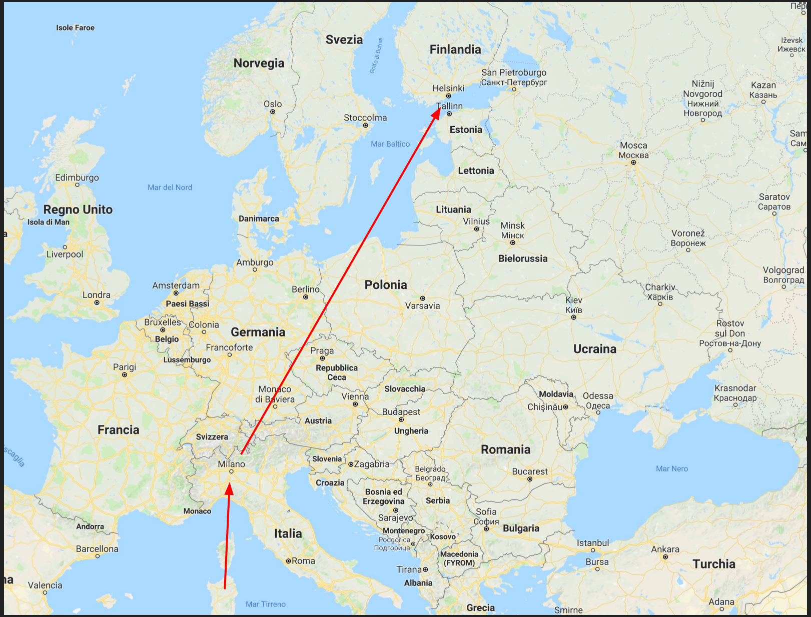
Haskell is mainstream
(finally)

A very excited industry report

Fabrizio Ferrai
@fabferrai
github.com/f-f

Who am I

- PoliMi alumnus, PPL alumnus
- Lead Developer at KSF Media in Helsinki, Finland
- We use the PHP stack: Postgres, Haskell, PureScript



Det ska bli lättare att starta ett medborgarinitiativ
29.11.2018 21.30

Finland är med i nya Hanseförbundet som fyller tomrummet efter brexit – "ingen vill...
29.11.2018 21.28

Helsingfors vill med ambitiösa åtgärder bekämpa klimatförändringen
29.11.2018 20.45

AKTUELLA ÄMNEN Filmrecensioner jämställdhet Notiser #dammenbrister Lucia 2018 TORSDAG 29 NOVEMBER

BORGMÄSTARENS SJÄLVSTÄNDIGHETSFEST

Borgmästarens fest är ett minne för livet – men saften var sur

Ställer du en fråga till fjärdeklassare kan du vara säker på att orden inte lindas in. Hallondrickan på borgmästarens stiliga fest var "lite sur", men vad gör det när du får vara med om en av grundskolans häftigaste upplevelser?
29.11.2018 - 20.24

HBL FEATURE
Cha-cha-cha får oss i extas

10 min live

TÄVLING
Vem bakar Svenskfinlands finaste pepparkakshus?

VEM BÄR LJUSKRONAN I ÅR?

Why am I here doing this

I work for a News company, so here’s some news:

- Finland is nice! (it’s a tradeoff..)
- Haskell is mainstream! (yes, really)
- Burnout in Software is a thing! (take care)

My actual agenda

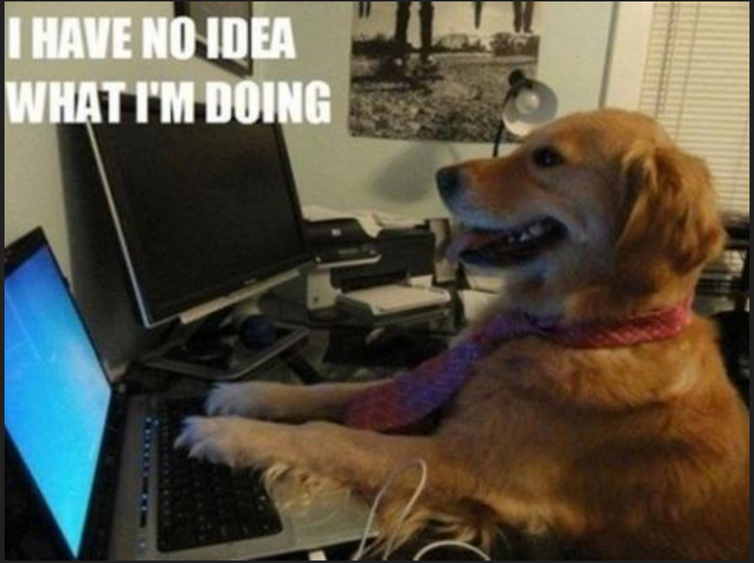
- Mostly, spread the love ❤️
- Backstory: I also want to fix Software Engineering

Hey wat wait what’s wrong with Soft Eng

We’re overly reliant on people computing things in their head

Can we let the compilers some more work for us?

YEES! And it’s better for everyone!



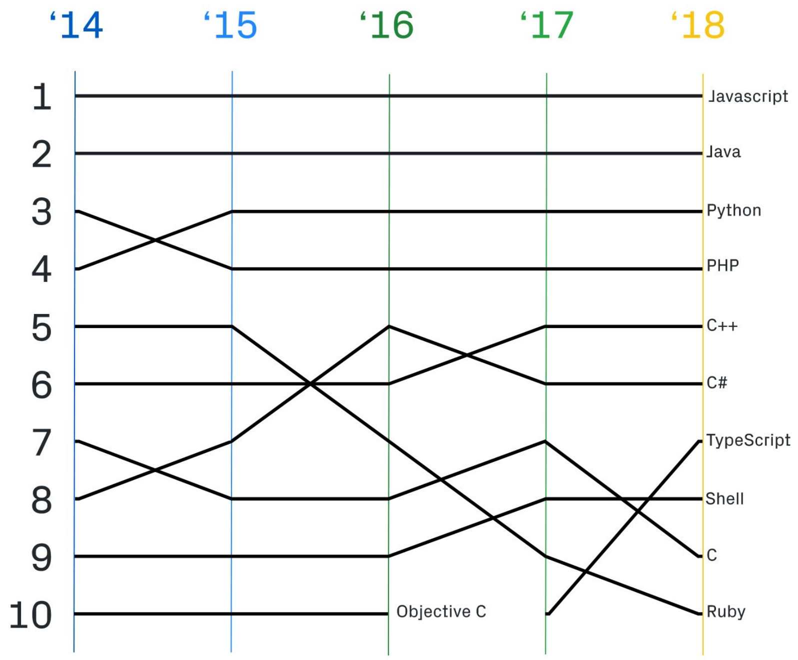
<Insert rant on consulting projects here>
<..and the industry in general>

Problem → Business requirements change all the time

Solution → Hey let’s make stuff actually maintainable!

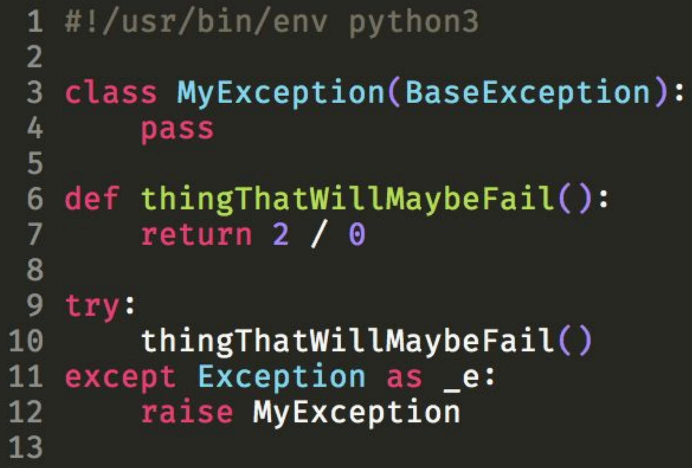
However...

Most pop langs out there



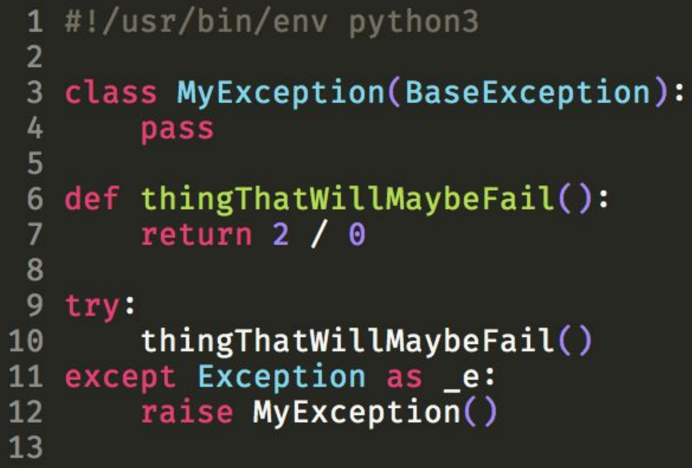
Source: Github's Octoverse

There’s a bug in here..



```python
#!/usr/bin/env python3

class MyException(BaseException):
    pass

def thingThatWillMaybeFail():
    return 2 / 0

try:
    thingThatWillMaybeFail()
except Exception as _e:
    raise MyException

```

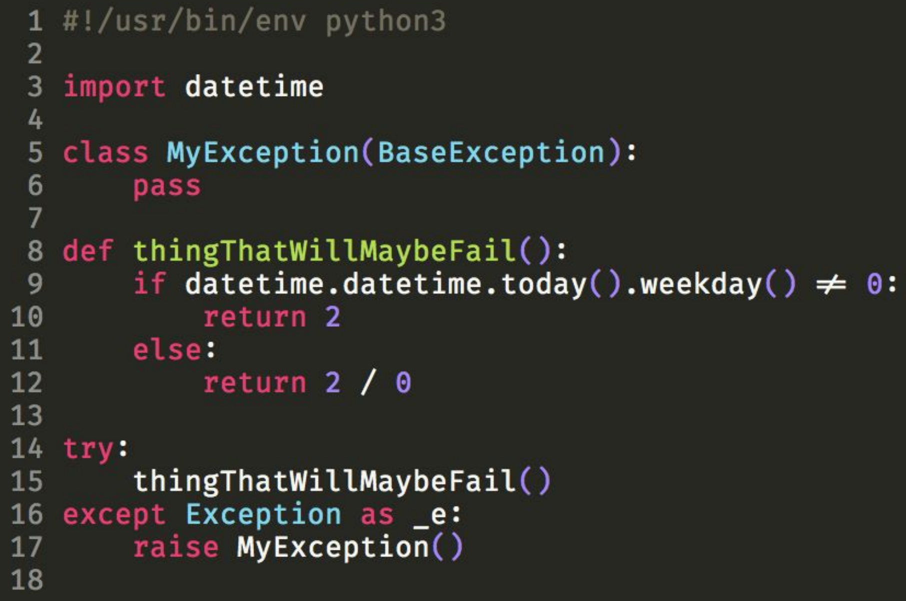
Yep



```python
#!/usr/bin/env python3

class MyException(BaseException):
    pass

def thingThatWillMaybeFail():
    return 2 / 0

try:
    thingThatWillMaybeFail()
except Exception as _e:
    raise MyException()

```

So maybe a typechecker makes sense

Let’s try again



```python
#!/usr/bin/env python3

import datetime

class MyException(BaseException):
    pass

def thingThatWillMaybeFail():
    if datetime.datetime.today().weekday() ≠ 0:
        return 2
    else:
        return 2 / 0

try:
    thingThatWillMaybeFail()
except Exception as _e:
    raise MyException()

```

So maybe controlling side effects makes sense

Let’s assume this makes sense

...can we do anything about it?

YES! Use Haskell! (or similar wizardry)

Wait, isn’t it academic?!?

→ It started like that: an experimentation platform for FP

...but research eventually makes to the industry! 😎

→ 10 years anniversary for the Industrial Haskell movement

Where is Haskell used

Mostly in:

- Security
- Finance
- Telecom

More info:

- Haskell in industry
- haskell-companies repo

- Facebook
- GitHub - Semantic Analysis
- Google
- Barclays Capital
- Standard Chartered Bank
- JP Morgan
- Klarna
- IOHK (Cardano)
- Kaspersky Lab
- Awake Security
- Ericsson

Haskell in Helsinki

Futurice

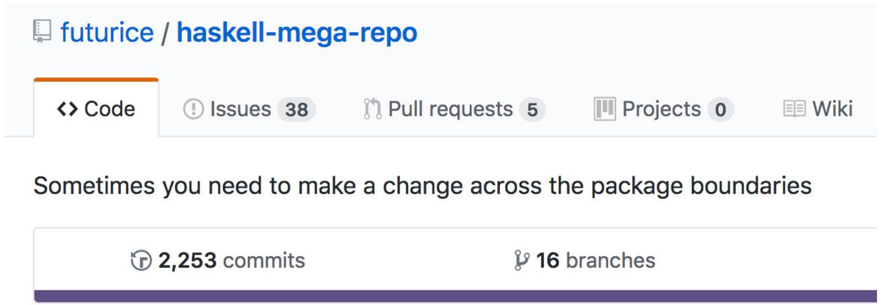
KSF Media

Tocoman

RELEX

Emblica

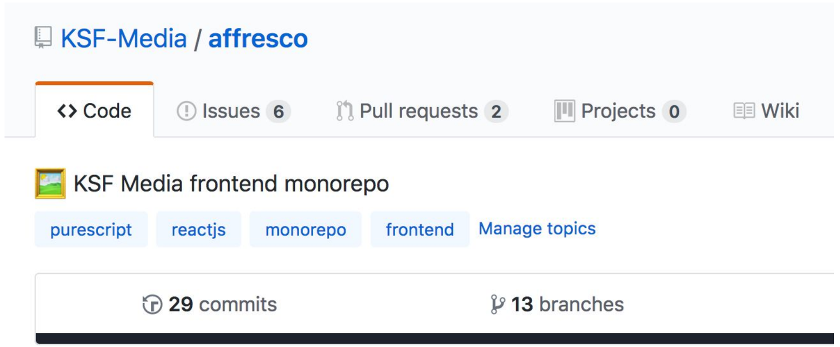
Zalando


futurice / haskell-mega-repo

Sometimes you need to make a change across the package boundaries

2,253 commits

16 branches

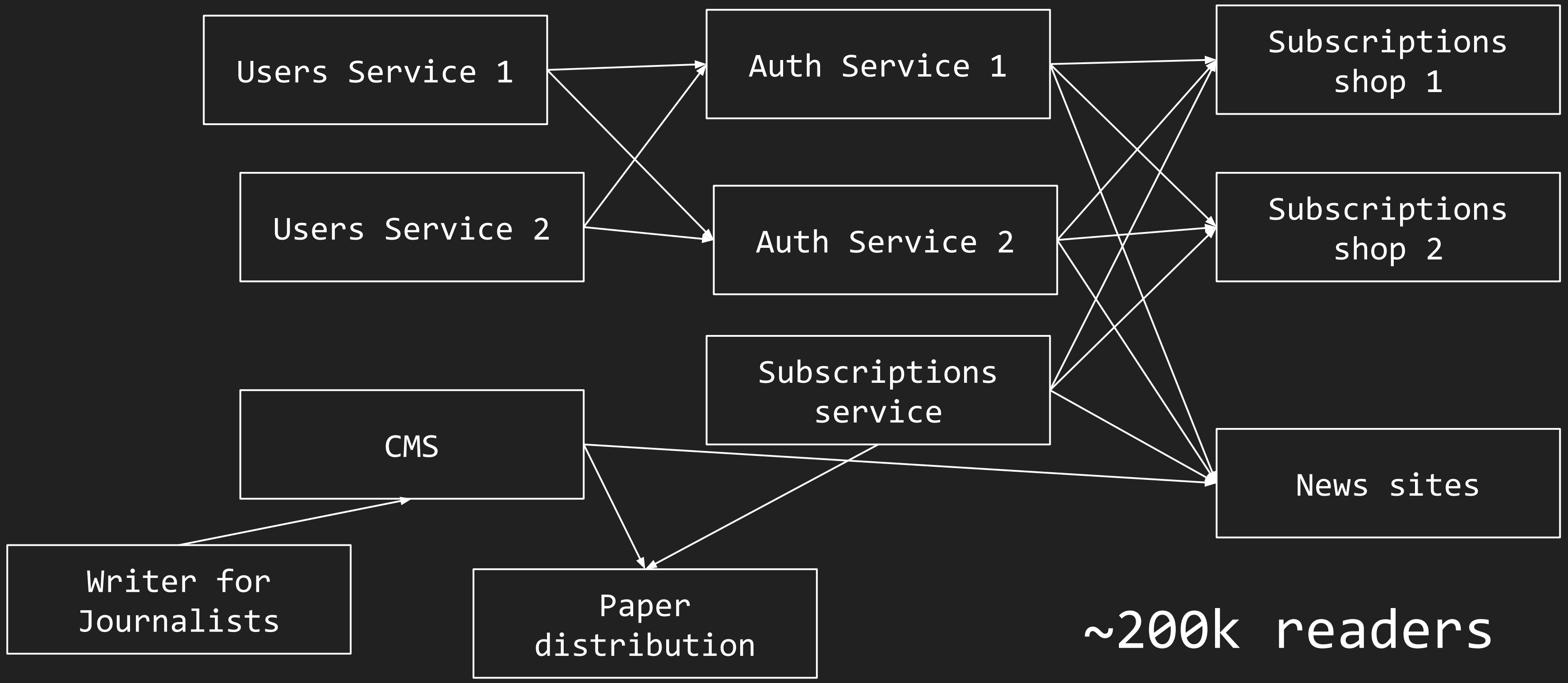

KSF-Media / affresco

KSF Media frontend monorepo

29 commits

13 branches

Right, so what do you actually do



~200k readers

Purely Functional Fullstack

NixOS	→ OS
Dhall	→ configurations
Haskell	→ backend
PureScript	→ frontend

Cool things

NixOS

“The Purely Functional Linux Distribution”

What if we could configure a system declaratively?

What if we could have deterministic dependencies?

And a global shared cache of precompiled binaries?

And have atomic upgrades?

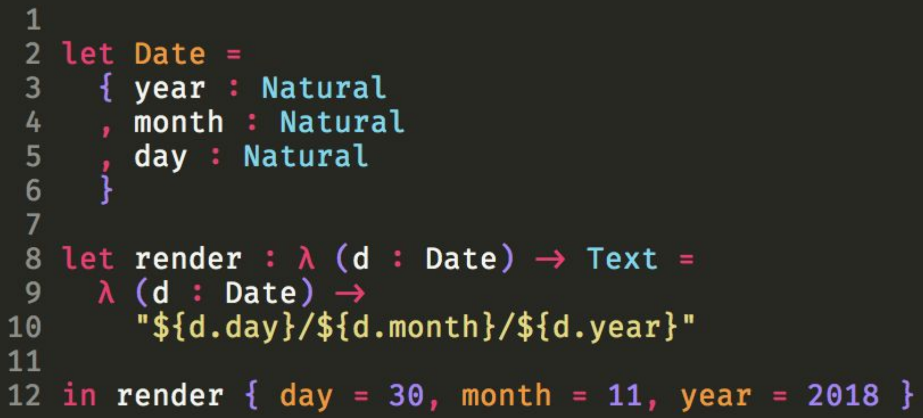
```
programs = {
  vim.defaultEditor = true;
  bash.enableCompletion = true;
  java.enable = true;
  zsh.enable = true;
  ssh.startAgent = true;
};


virtualisation = {
  docker.enable = true;
  docker.autoPrune = {
    enable = true;
    dates = "monthly";
  };
};
```

Dhall

Total Functional Programming language → always terminates!

Strongly typed and strongly normalizing → useful for configs

Looks like: JSON/YAML + functions + imports + types + templating



```

let Date =
  { year : Natural
  , month : Natural
  , day : Natural
  }

let render : λ (d : Date) → Text =
  λ (d : Date) →
    "${d.day}/${d.month}/${d.year}"

in render { day = 30, month = 11, year = 2018 }
```

=> “30/11/2018”

Servant

Formalizing API definitions with types

Get for free:

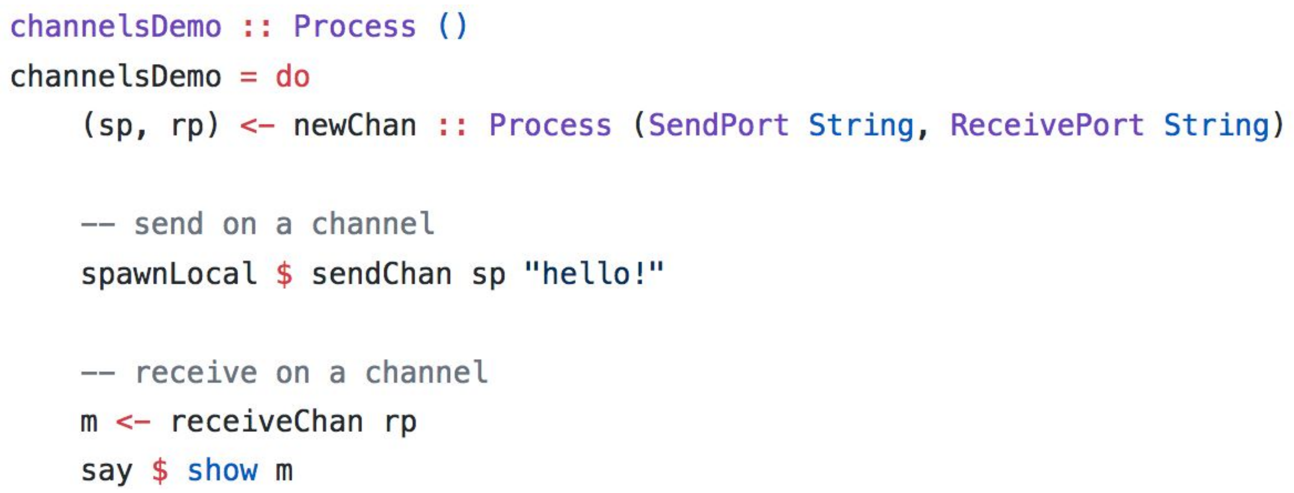
- Type safety
- Documentation
- Clients generation

```haskell
data LoginData = LoginData
  { username   :: EmailAddress
  , password   :: Password
  } deriving (Show, Eq, Generic, Data, ToJSON, FromJSON, ToSchema)

data LoginResponse = LoginResponse
  { token    :: AccessToken
  , uuid     :: UUID
  } deriving (Show, Eq, Generic, Data, ToJSON, FromJSON, ToSchema)

type Login =
  Summary "Login with email and password"
    :> JsonReqBody LoginData
    :> Throws 403 "invalid_credentials"
    :> Throws 500 "internal_server_error"
    :> Post '[JSON] LoginResponse

server :: ServerT Login (RIO Env)
server = login

login :: LoginData → RIO Env LoginResponse
login = undefined
```

Cloud Haskell

Like Erlang, but typesafe:

not only send/receive, but Typed Channels!

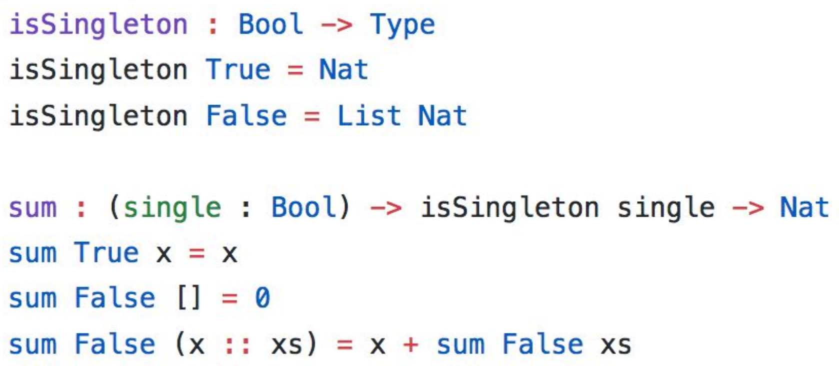


```haskell
channelsDemo :: Process ()
channelsDemo = do
    (sp, rp) <- newChan :: Process (SendPort String, ReceivePort String)

    -- send on a channel
    spawnLocal $ sendChan sp "hello!"

    -- receive on a channel
    m <- receiveChan rp
    say $ show m
```

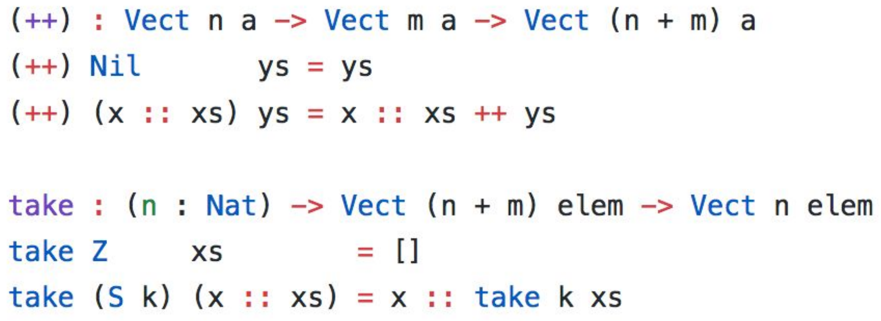
Idris

Dependently typed language

Crash course dependent types: types that depend on values



```
isSingleton : Bool -> Type
isSingleton True = Nat
isSingleton False = List Nat

sum : (single : Bool) -> isSingleton single -> Nat
sum True x = x
sum False [] = 0
sum False (x :: xs) = x + sum False xs
```



```
(++) : Vect n a -> Vect m a -> Vect (n + m) a
(++) Nil       ys = ys
(++) (x :: xs) ys = x :: xs ++ ys

take : (n : Nat) -> Vect (n + m) elem -> Vect n elem
take Z     xs         = []
take (S k) (x :: xs) = x :: take k xs
```

Where is the world going?

We’re getting rid of boring programming!

1. **Formal Proofs**: write a specification, which is executable, prove it correct (very useful for DistSys)
2. **Declarative programming**: say what should be done, not how (SQL)
3. **Encoding constraints in Type Systems**: like (2) but happening at compile time, so like (1)

TL;DR: Production Haskell

😍

- I get to keep my sanity
- Refactoring is sweet
- Types prevent tons of bugs
- Solid ecosystem
- Tight, lovely community
- Faster than Java/Node/Go
- Parallelism/Concurrency

😕

- It’s “different” (lazy, FP)
  Need to relearn many things
- Tough learning curve, getting started alone is hard

So I’d like to work in Haskell, wat do?

- Let’s have a chat
- Get a normal job, convince everyone it’s worth it
- reddit.com/r/haskell
- Come to conferences! E.g. ZuriHac
- Follow Haskell peeps on Twitter
- Open Source projects in Haskell
- Master Thesis in Haskell! Let’s improve GHC!

Hope it’s useful!

Thanks! ❤️

@fabferrai

github.com/f-f

Links 1/2

Haskell ecosystem and patterns

- github.com/Gabriel439/post-rfc/blob/master/sotu.md
- haskellforall.com/2014/10/how-to-desugar-haskell-code.html
- two-wrongs.com/a-gentle-introduction-to-monad-transformers
- mylifeecho.com/dev/telegram-bot-tutorial/
- parsonsmatt.org/2018/03/22/three_layer_haskell_cake.html
- github.com/bitemyapp/learnhaskell/blob/master/specific_topics.md
- dev.stephendiehl.com/hask
- wiki.haskell.org/Typeclassopedia

Idris

- docs.idris-lang.org/en/latest/tutorial/index.html
- docs.idris-lang.org/en/latest/tutorial/typesfuns.html
- manning.com/books/type-driven-development-with-idris
- youtube.com/watch?v=Yxd9_kNtoZg

Links 2/2

Cloud Haskell

- haskell-distributed.github.io/tutorials/1ch.html
- stackbuilders.com/tutorials/haskell/cloud-haskell/
- haskell-distributed.github.io/documentation.html#typed-channels

NixOS

- nixos.org

Dhall

- github.com/dhall-lang/dhall-lang
- youtube.com/watch?v=UHp6nEF5m2o

Servant

- haskell-servant.readthedocs.io/en/stable/tutorial/

PureScript

- purescript.org
- leanpub.com/purescript/read
- github.com/f-f/purescript-react-basic-todomvc