zalando

Why Haskell

Engineers
Management
Company

DMITRY DOLGOV

10-05-2017

zalando

zalando

zalando

- ➔ For developers
- ➔ For management
- ➔ For company

zalando

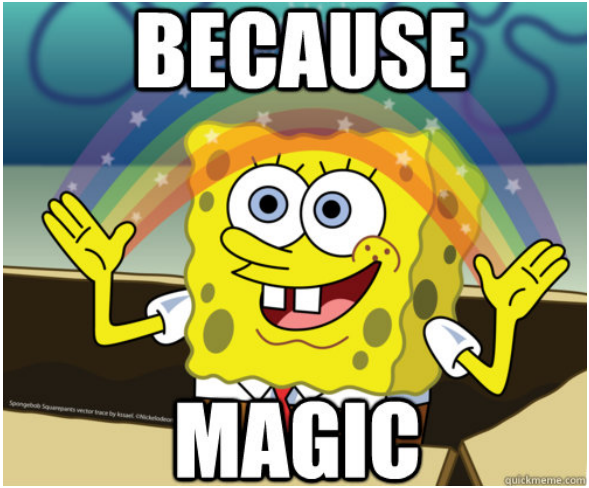
Developers

zalando

Compiler support

zalando

Advanced type system



```haskell
data USD
data EUR

newtype Currency a = Currency Double deriving Show

add :: Currency a → Currency a → Currency a
add (Currency a) (Currency b) = Currency $ a + b
```


zalando

```haskell
account :: Currency USD
account = Currency 5.0

transaction1 :: Currency USD
transaction1 = Currency 5.0

transaction2 :: Currency EUR
transaction2 = Currency 5.0

main = do
    print $ add account transaction1 -- Currency 10.0
    print $ add account transaction2 -- won't compile
```


zalando

```
$ ./currency_test.hs

currency_test.hs:23:25: error:
  • Couldn’t match type ‘EUR’ with ‘USD’
    Expected type: Currency USD
      Actual type: Currency EUR
  • In the second argument of ‘add’,
      namely ‘transaction2’
    In the second argument of ‘($)’,
      namely ‘add account transaction2’
    In a stmt of a ’do’ block:
      print $ add account transaction2
```


zalando

Computations are more explicit

```
gtod = scd->tick_gtod + __gtod_offset;
clock = gtod + delta;
min_clock = wrap_max(gtod, old_clock);
max_clock = wrap_max(old_clock, gtod + TICK_NSEC);
```


zalando

Explicit computation patters

Developer> I have a database query here
Developer> it may fail, so it’s kinde unsafe
Compiler> Don’t worry, I’ll check all the deps
Compiler> and let you know if some of them
Compiler> are not ok with that

zalando

```haskell
calculation1 :: Int → Maybe Int
calculation1 arg = Just (arg + 1)

calculation2 :: Int → Maybe Int
calculation2 arg = Nothing

calculation3 :: Int → Maybe Int
calculation3 arg = Just (arg + 2)
```


zalando

```
main = do
    print $ return 1 >>= calculation1 >>= calculation2
    print $ return 1 >>= calculation1 >>= calculation3
```


zalando

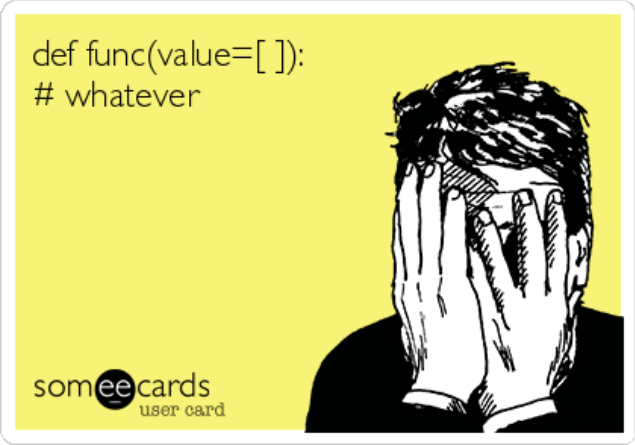
The same for state, IO, sequential calculations etc.

zalando

Referential transparency
a.k.a. refactoring without fear

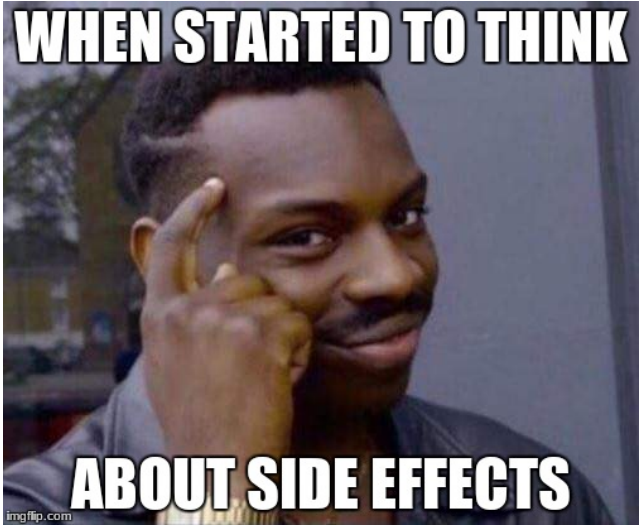
zalando

Immutability
a.k.a. you can trust this function



zalando

Explicit handling of side effects



zalando

Tooling support, great ecosystem

- ➔ stack
- ➔ ghc
- ➔ ghci
- ➔ editors and IDEs

Remarkable performance

Why is Haskell (GHC) so darn fast?

100

34

Haskell (with the `GHC` compiler) is a lot faster than you
to low-level languages. (A favorite thing for Haskellers
beat it, but that means you are using an inefficient C pr
My question is, why?

Haskell is declarative and based on lambda calculus. N
being based on turing machines, roughly. Indeed, Hask
order. Also, instead of dealing with machine data types

zalando

Do you need a parser?

zalando

Management

zalando

High quality code (less bugs, easy to maintain and to learn)

zalando

Awesome for agile development (because of referential transparency, easy to support and refactor large code base and create dsl)

zalando

Heavy boost for skills development inside your team

zalando

Rumored to be much cost efficient (Aaron Contorer)

zalando

You can attract a lot of talented people

zalando

Company

zalando

A lot of big companies are investing in this

zalando

Contributions is much more visible - not one of thousands nameless companies with java technology stack, you have a ~~dragon~~ Haskell in production

zalando

Great anchor for development of technical culture (CS and stuff)

zalando

Questions?

github.com/erthalion

@erthalion

9erthalion6 at gmail dot com

zalando