Haskell Workshop

FINN.no

FINN.no

Haskell

- Haskell / GHC
- Purely functional
- No side-effects
- Lazy
- Mystic *

* has monads

In haskell

```
f :: a -> b

g :: c -> d

g . f
```

Wire input to output

```haskell
isPrime :: Natural -> Bool
isPrime 0 = False
isPrime 1 = False
isPrime 2 = True
isPrime 3 = True
isPrime 4 = False
isPrime 5 = True
isPrime 6 = False
isPrime 7 = True
```

Wat?

```haskell
validate :: Integer -> Bool
validate = isZeroMod10 . sum . doubleEveryOther . toDigits
```

A little bit of syntax

```haskell
data TrafficLight = Red | Yellow | Green

safe :: TrafficLight -> Bool
safe Red    = False
safe Yellow = False
safe Green  = True
```

Applying a function

```haskell
nextInt :: Integer -> Integer
nextInt x = succ x
```

No paranthesis!

Curried signatures

```
addTwoNumbers :: Integer -> Integer -> Integer
addTwoNumbers a b = a + b
```

If-then-else

```
isNine :: Integer -> Bool
isNine i = if i == 9
  then True
  else False
```

let/in-expressions

```
cylVolume :: Float -> Float -> Float
cylVolume diam h =
  let rad = diam / 2
      area = pi * rad^2
  in area * h
```

lists

```
listOfInts :: [Integer]
listOfInts  = [1,2,3]

concat :: [a] -> [a] -> [a]
concat as bs = as ++ bs
```

recursion on lists

```
uppercase :: [Char] -> [Char]
uppercase [] = []
uppercase (x:xs) = toUpper x :: uppercase xs
```

higher order functions on list

```
inc :: [Integer] -> [Integer]
inc numbers = map (+1) numbers
```

```haskell
inc2 :: [Integer] -> [Integer]
inc2 numbers = map (\x -> x + 1) numbes
```

higher order functions on list 2

```haskell
firstFive :: [a] -> [a]
firstFive as = take 5 as

onlyEven :: [Integer] -> [Integer]
onlyEven xs = filter even xs
```

What to do

- README.md