Haskell:
Fun With Types

Eivind Jahren

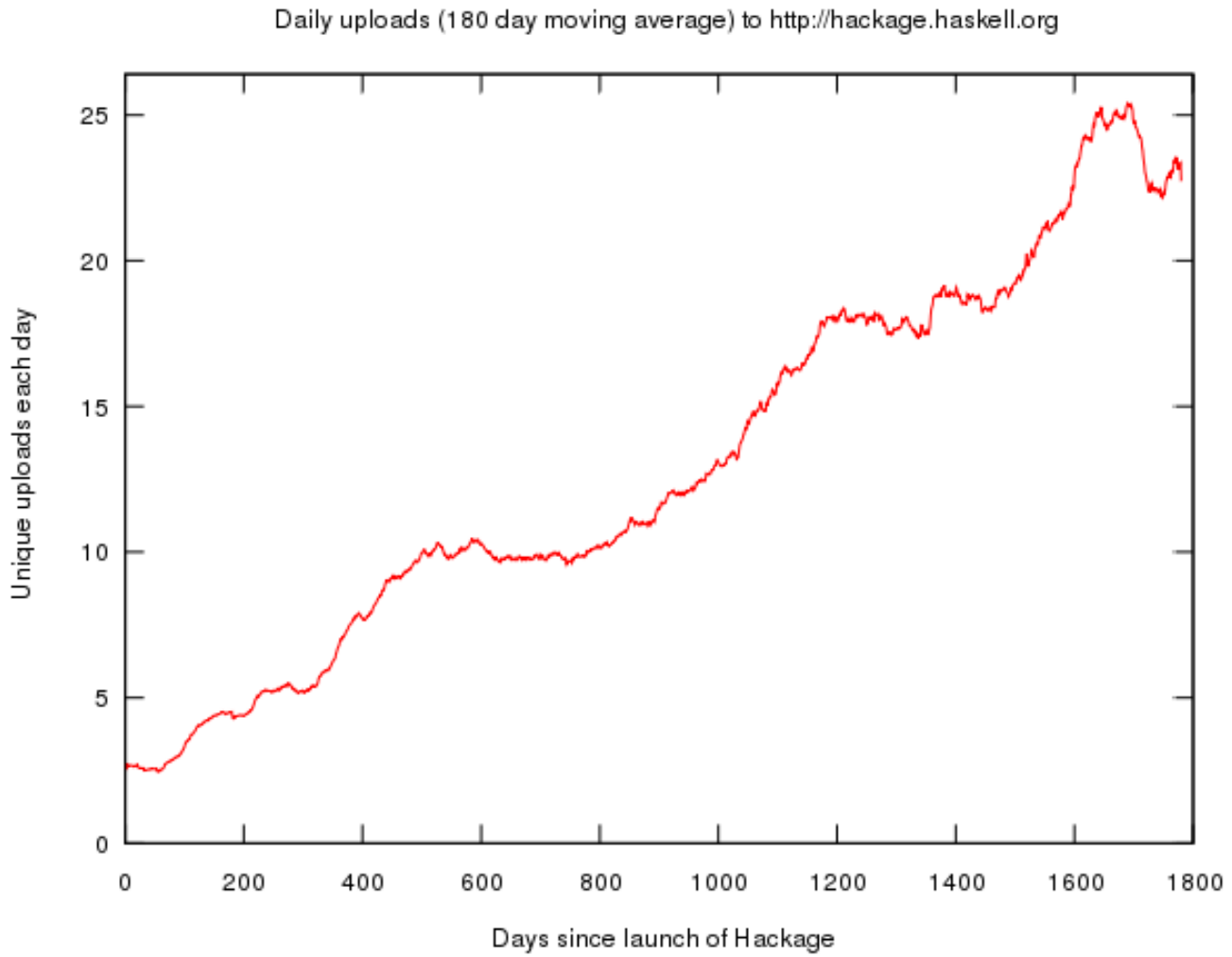
March 5, 2013

The life and times...

- Designed by a comittee.
- 1990: The first Haskell Report.
- Only a research toy for a very long time.

The life and times...

- Designed by a comittee.
- 1990: The first Haskell Report.
- Only a research toy for a very long time.



Why is Haskell Hawt?

- Lazy by default.

```
fst (a,b) = a

main = print (fst (1, killallhumans earth))
```

- Purely functional by default.
- Strong type system with inference.

```
main :: IO ()
main = putStrLn "Hello, World!"
```

Why is Haskell Hawt?

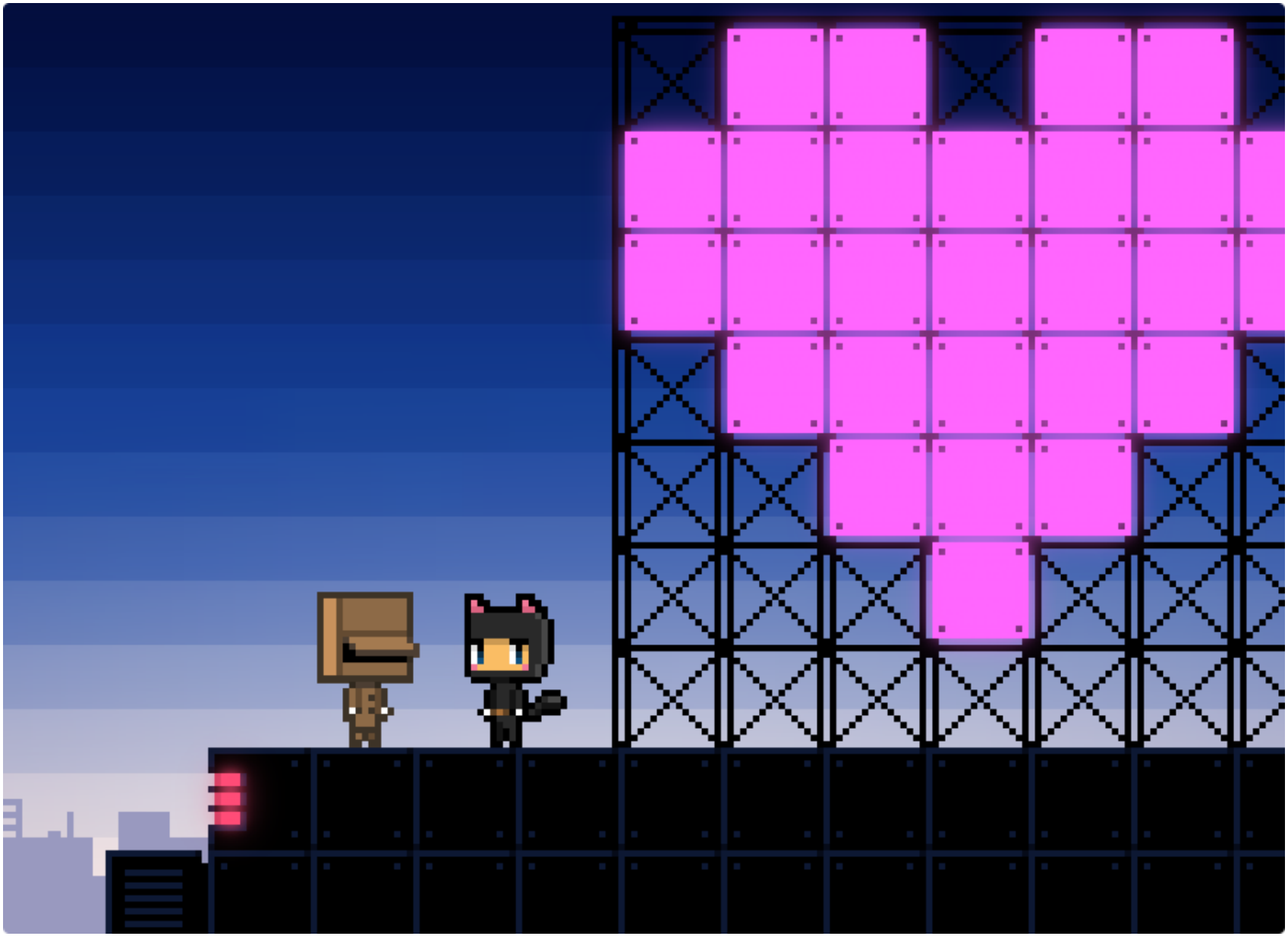
- Lazy by default.

```
fst (a,b) = a

main = print (fst (1, killallhumans earth))
```

- Purely functional by default.
- Strong type system with inference.

```
main = putStrLn "Hello, World!"
```



Enough sales talk, send me teh codez!

Task: Generate random values by type. User of the Library can declare objects that return a random value.

Requirement: Must be able to make random list whenever we can make random elements of that list.

```
class Random a where
  random :: IO a
```

```haskell
class Random a where
  random :: IO a

generateRandomInt :: IO Int
generateRandomInt = ...

instance Random Int where
  random = generateRandomInt
```

```haskell
class Random a where
  random :: IO a

generateRandomInt :: IO Int
generateRandomInt = ...

instance Random Int where
  random = generateRandomInt

instance (Random a) => Random [a] where
  random = do
    randLength <- generateRandomInt
```

```haskell
import Control.Monad(replicateM)

class Random a where
  random :: IO a

generateRandomInt :: IO Int
generateRandomInt = ...

instance Random Int where
  random = generateRandomInt

instance (Random a) => Random [a] where
  random = do
    randLength <- generateRandomInt
    replicateM randLength (random :: IO a)
```

```haskell
import Control.Monad(replicateM)

class Random a where
  random :: IO a

generateRandomInt :: IO Int
generateRandomInt = ...

instance Random Int where
  random = generateRandomInt

instance (Random a) => Random [a] where
  random = do
    randLength <- random
    replicateM randLength random
```