Fun~~ctional Programming Languages~~
why, where, how

MF#K January 2017 Meetup

@Prosa 2017-01-31

Overview

- About me
- (fun _ → why, where, how)
- Summary
- Q & A

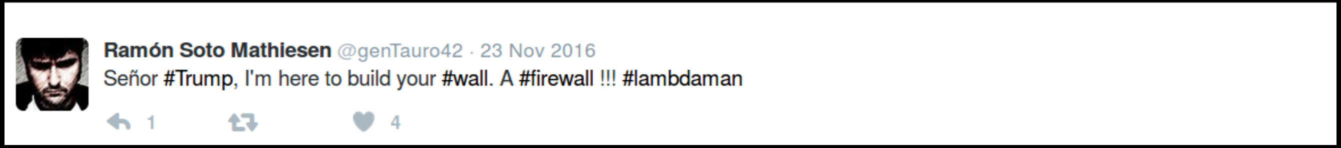
About me (very shortly)

- Ramón Soto Mathiesen
- MSc. Computer Science ***DIKU/Pisa*** and minors in Mathematics ***HCØ***
- ***CompSci*** @ SPISE MISU ApS
  - ***"If I have seen further it is by standing on the shoulders of giants"***
    -- Isaac Newton (Yeah Science, Bitch … Mostly mathematics)
  - ***Elm*** (***JS*** due to ports) with a bit of ***Haskell*** and a bit of ***F#*** (fast prototyping)
- Elm / Haskell / TypeScript / F# / OCaml / Lisp / C++ / C# / JavaScript
- Blog: http://blog.stermon.com/

(fun _ → why, where, how)

- In this second talk we will put emphasis on the ***fun*** part of programming languages
- You will all, mostly all, be coding functionally but without using a computer
- So given recent events, we are going to help build a …

(fun _ → why, where, how)


Ramón Soto Mathiesen @genTauro42 · 23 Nov 2016
Señor #Trump, I'm here to build your #wall. A #firewall !!! #lambdaman

1 4

(fun _ → why, where, how)

- In order to build a ***firewall***, we will need to have wall pieces of the same color ("***segregation***") with:
  - L × W × H: **1.6 cm** × **4.8 cm** × **4.75 cm** (≈ 4.92 cm top dots)
- Lets keep all those "***bad packages***" away

(fun _ → why, where, how)

- As mentioned before, most of you will be having ***fun*** while a very few will … Therefore we are dividing you up in ***two*** groups:
  - Team Functional
  - Team Imperative (***Claes***, ***Jannick*** and ***Oscar*** you go here)

Team Functional

- Will be working with immutable data structures (*)

(*) - Please don’t try to break them

Team Imperative

- Will be working with … Good luck, you will need it !!!

(fun _ → why, where, how)

- **Joakim** and myself will be the **final** “Acceptance Test”
  - We seem to have ***issues*** with our ***small hands***, that’s why ***our ruler*** is ***smaller*** than yours …
- Before you handle us a ***piece of wall***, you will need to ***perform*** your ***own tests***. There is ***only one ruler*** (to rule them all), so ***both*** teams will have to ***share it***
  - I’m guessing Team Imperative is going to use it the most

(fun _ → why, where, how)

(we will use ***15 – 30 minutes*** on the task)

Reminder:

Wall pieces of the same color with:

L × W × H: **1.6 cm** × **4.8 cm** × **4.75 cm** (≈ 4.92 cm top dots)

(fun _ → why, where, how)

/Nostradamus mode on

- What we have seen is that a lot of ***fun*** people could work with the same data, slicing colors and sizes, at the same time (concurrency/parallelism) while each ***imperative*** person had to sit with her/his small bucket of Play-Doh as a mixture of colors would be impossible to revert …

(fun _ → why, where, how)

/Nostradamus mode on

- Given the nature of the immutable data-blocks provided to the ***fun*** people, it was easy to combine them to the requested wall size while still providing the same robustness and immutability as the lesser blocks
- On the other hand, imperative people had to do everything on their own getting a much worse result, even though it was skilled people trying to provide some craftsmanship

(fun _ → why, where, how)

/Nostradamus mode on

- Some of Team Imperative suffered that our “Acceptance Test” sadly produced some ***awful side-effects*** on your data structure (a ***wall*** become a ***sphere***)
  - It wasn’t meant to be a ***cunt move*** (maybe it was) but we were only trying to show what happens in real life (*)

(*) Happened to me last week when having to work on some JavaScript Interoping with Elm:

```
function foobar(xs){
    xs.reverse()             // changes xs array
    xs.slice(0).reverse()    // clones xs and then reverse
}
```

(fun _ → why, where, how)

- It’s important to understand that “Play-Doh” might give you more freedom to do what you want but less reliability …
  - **Reliable**, adj: To deliver the same result every time.

(fun _ → why, where, how)

(fun _ → why, where, how)

(fun _ → why, where, how)

(fun _ → why, where, how)

- While “LEGO” still gives you artistic freedom, but with a few sound constraints that help you create reliable work every-single-time
  - **Reliable**, adj: To deliver the same result every time.

(fun _ → why, where, how)

(fun _ → why, where, how)

Summary

- We need reliability in our software solutions and this is something that is built-in to ***fun*** languages. I know you get more “freedom” with imperative languages (Example: ***C*** or ***JS***), but with that comes a lot of responsibility and lets face it, most developers can’t handle that.
  - **Reliable**, adj: To deliver the same result every time.
- Finally, so who paid for the ***firewall***? Sadly, I did :(

Summary

- Last but not least, Joakim and I have committed, in collaboration with PROSA, to provide two introductory courses in **Scala** (***Java*** people) and **F#** (***.NET*** people):
  - Date still to decide (most likely February or March)
  - Free for PROSA members and a fee for non-members

PROSA
Forbundet af It-professionelle

Q & A

Any Questions?

(and let’s go for beers @ Ørsted Ølbar)