How to sell FP?

Denis Redozubov, @rufuse

December 9, 2016

Who should be interested?

- you're interested in showing your boss/colleagues that FP is the way to go
- you want to encourage you clients/investors to use FP

Quick answers

- Know your stuff and teach it to others
- Bring value
- Show that the brought value lies in F
- ???
- PROFIT

Disclaimer

I'll be talking primarily about statically typed pure functional languages.

Referential transparency

Expression always evaluates to the same result. Really basic stuff.

Persistent data structures

Making the best out of immutability. The most important building block for concurrent and parallel systems.

Equational reasoning

This is what we get with referential transparency and persistent data structures - an ability to reason about a piece of code ignoring everything else.
If you don't have this - it's super hard to reason about code, especially on paper.

Concurrency and parallelism

FP introduced STM, MVar and other primitives(such as LVar/IVar) that make life easier. Immutability and referential transparency makes it possible for the functional setting. Concurrency annotations in FP generally doesn't change program semantics!

Future-proof

CS concepts tend to go through the academical publications to mainstream. It took 3-4 decades for some concepts(e.g. Σ and Π-types), but this distance is closing fast.
Be the first ones to grok and use the new hotness:

- Refinement types
- Linear and affine types
- Deforestation/fusion
- Codata
- Advancements in Generic programming
- Homotopy type theory

and so on

Knowing your stuff

Essentially it makes you closer to the CS foundations and empowers you to read a lot of works on the subject.
For example, it's always good to recall π-calculus when you encounter a system of distributed processes.

Mature abstractions

The most documented side of programming

We have the best documented jargon in the world. It's all over the math books and CS papers. Really easy to get the idea across, if you know these concepts.
Much better than framework ad-hoc pattern names.

Battle-tested

- facebook use haskell for the anti-spam system
- twitter is powered by scala
- whatsapp uses erlang

We use haskell exclusively at typeable.io and we're really happy with it.

Business people are extremely smart

It's all about $'s

Businesses want "mostly correct" software that will be developed fast and can be updated cheaply. It's all about the "Bang for the buck".

As an advisor, teach the right questions that will lead to a solid software system.

Nowadays, you're fighting against the marketing teams when choosing technology, make this fact obvious.

Sell to people that want expertise

Great business people see why it's useful to hire people that will bring value and will generate income.
A lot of functional programmers nowadays are these people, they tend to escape routine and incomprehensible systems.

Go above the average

If the average is blub, just go above it. Get the secret weapon, just as Paul Graham advises. The value you bring by your choice of tech will bring great developers to you.

Battle of averages

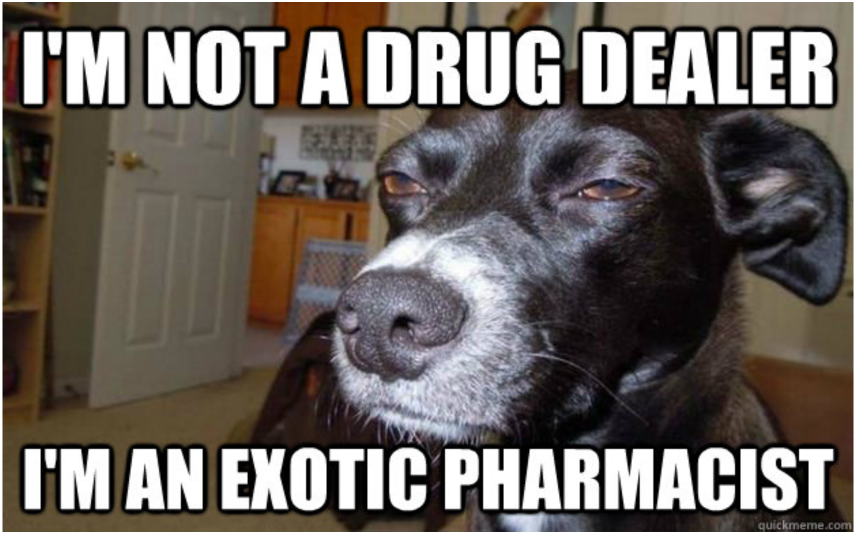
The concept of "average" is really important for business people. You should count on "average" person to do the job, not the geniuses. The fact is: average haskell developer is much more productive and insightful than Java/Go engineer.
It's possible to find brilliant people in any community though.

Go create some jobs

Show the goal to your boss:

- Performant and maintainable software
- Less time and costs on maintanance
- Attracting people that will bring value and even more talent
- Creating a culture where you can teach developers what they need

Drug Dealer Strategy



Critical software

Some software must be correct. Imagine:

- medical devices commiting malpractice
- buggy car brake automation systems
- tanker autopilot on collission course
- nodejs-powered drone falling on your face

So..

Traditional thought

It's a virtue that's traditionally regarded as secondary, but i'd like to argue that.
"We're ok with 'mostly correct'" (C)
No, thank you.

Criminal coding

- We have car brake systems controlled by software
- Oil tankers relying on autopilot
- Drones flying over our heads
- Medical diagnostics software

Criminal coding

- Martian Climate Orbiter Disaster
- Therac-25 - massive overdoses of radiation (at least 3 lethal cases)
- Both US and Russia reported false indication of incoming missile attacks
- Undetected hole in the ozone layer over Antarctic

Verified software

All theorem provers and verification systems are functional. FP gave us:

- CompCert
- Bedrock
- CertiCrypt

Good approximation

Using PL with a sound and expressive type system is a way to go too. Burden of proving every single thing about a program may be too high for the most cases.

Adoption in mainstream languages

Lambdas in mainstream languages. Map and filter functions everywhere.

Adoption in mainstream languages

It's not nearly good enough!

Overcoming obstacles

e.g. Halting problem, Rice's theorem.
We see FP solutions like Nix and Dhall.

Type systems

We have them all over the place, but few of them are good.

Type systems

The most advanced and expressive type system exist in purely functional languages.
Haskell, Agda, Idris, Coq, Epigram, F*, Lean etc.
There are less radical examples: OCaml, F#, Scala etc.

What constitutes an extraordinary software

If it does it's job and you don't have a total mess on your hands, it's considered good.

Questions?