Why bother with FP?

More: https://github.com/LIttleAncientForestKami/why-bother-with-FP

Attempt at honest and backed by data answer

lambda DAλS

REACT DAY

SFI

λCON

Tomasz Borek, 2015

Who am I?

Tomasz Borek

Consultant @ Symentis

@LAFK_pl

FP is AWESOME!

- „Knowledge of FP is on the verge of becoming a must-have skill” (DrDobbs)
- „If you don't know it you're not a REAL programmer” (Devoxx UK 2014 discussion)
  - Real Programmers vs Quiche Eaters
- „The smartest programmers I know are functional programmers.” (Quora)

Why such talk?

„Mythical Man-Month”,
Frederick Brooks

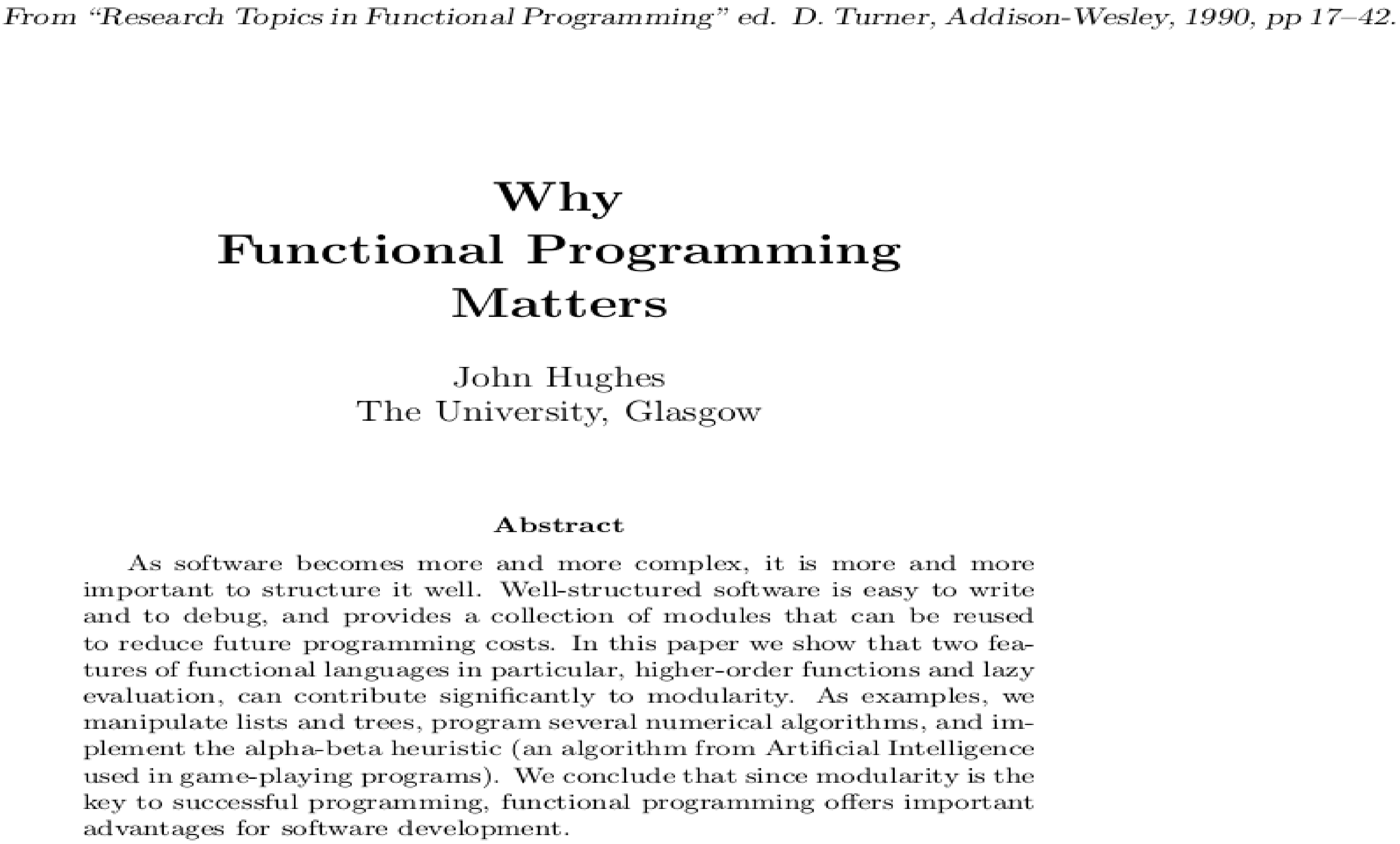
What will I talk about?

- About myself and the talk
- Better, means?
- The usual claims you hear and data on them
- The not so usual claims, like „good OO”
- So, why bother?

Who here has read?

- **Why FP matters**
- **Out of the Tar Pit**
- **Can programming be liberated from von Neumann style?**

Why FP matters



From "Research Topics in Functional Programming" ed. D. Turner, Addison-Wesley, 1990, pp 17–42.

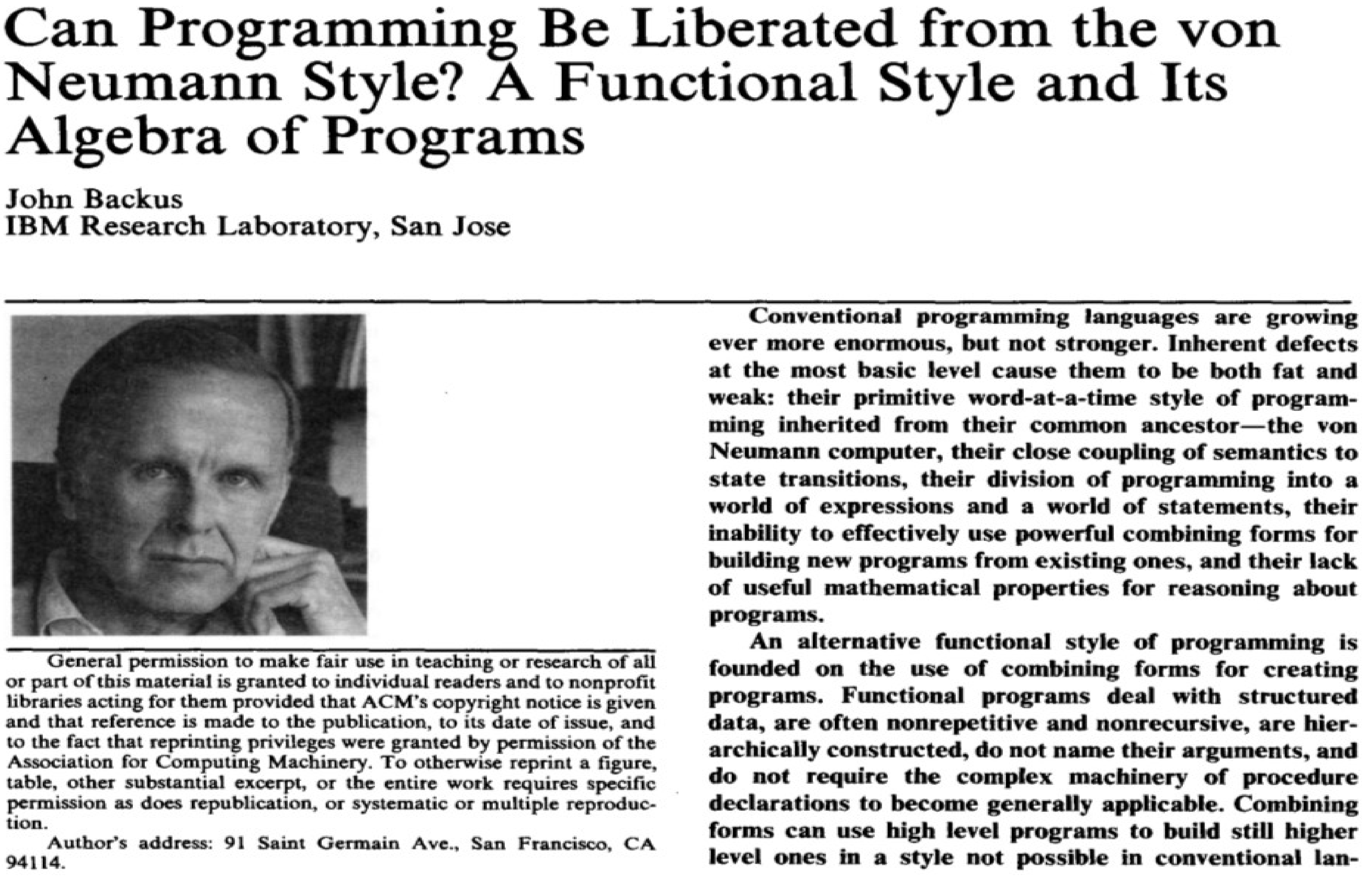
Why Functional Programming Matters

John Hughes
The University, Glasgow

Abstract

As software becomes more and more complex, it is more and more important to structure it well. Well-structured software is easy to write and to debug, and provides a collection of modules that can be reused to reduce future programming costs. In this paper we show that two features of functional languages in particular, higher-order functions and lazy evaluation, can contribute significantly to modularity. As examples, we manipulate lists and trees, program several numerical algorithms, and implement the alpha-beta heuristic (an algorithm from Artificial Intelligence used in game-playing programs). We conclude that since modularity is the key to successful programming, functional programming offers important advantages for software development.

Can programming be liberated?



Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs

John Backus
IBM Research Laboratory, San Jose

Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: their primitive word-at-a-time style of programming inherited from their common ancestor—the von Neumann computer, their close coupling of semantics to state transitions, their division of programming into a world of expressions and a world of statements, their inability to effectively use powerful combining forms for building new programs from existing ones, and their lack of useful mathematical properties for reasoning about programs.

An alternative functional style of programming is founded on the use of combining forms for creating programs. Functional programs deal with structured data, are often nonrepetitive and nonrecursive, are hierarchically constructed, do not name their arguments, and do not require the complex machinery of procedure declarations to become generally applicable. Combining forms can use high level programs to build still higher level ones in a style not possible in conventional lan-

Author's address: 91 Saint Germain Ave., San Francisco, CA 94114.

Out of the Tar Pit

- Complexity:
  - Essential
  - Accidental
- Sources:
  - State mutation
  - State sharing
- Logic and FP marriage

SORRY!

MORE INFO

https://lafkblogs.wordpress.com/2015/03/07/why-bother-with-fp/

CONTRIBUTE

https://github.com/LIttleAncientForestKami/why-bother-with-FP

Better?

- Languages
- Designed with X in mind
- Is X better?

The usual claims

- It's the FUTURE / NEXT BIG thing
- Changes your thinking / another tool to have
- Shorter / terser code
- More power / Better abstractions / Convenient
- Complexity / State mutation
- Streams / no side-effects
- Reliability / proven
- Concurrency / Multicore

Define: FUTURE

- By fame
  - conferences and conference talks
  - articles
- By practicality
  - market share
  - who / where uses
  - job offers
- By language popularity / adoption level

The FUTURE by fame

- Conferences?
  - Lambda Days
  - LambdaCon
  - Strangeloop
- Articles:
  - Even in mainstream portals

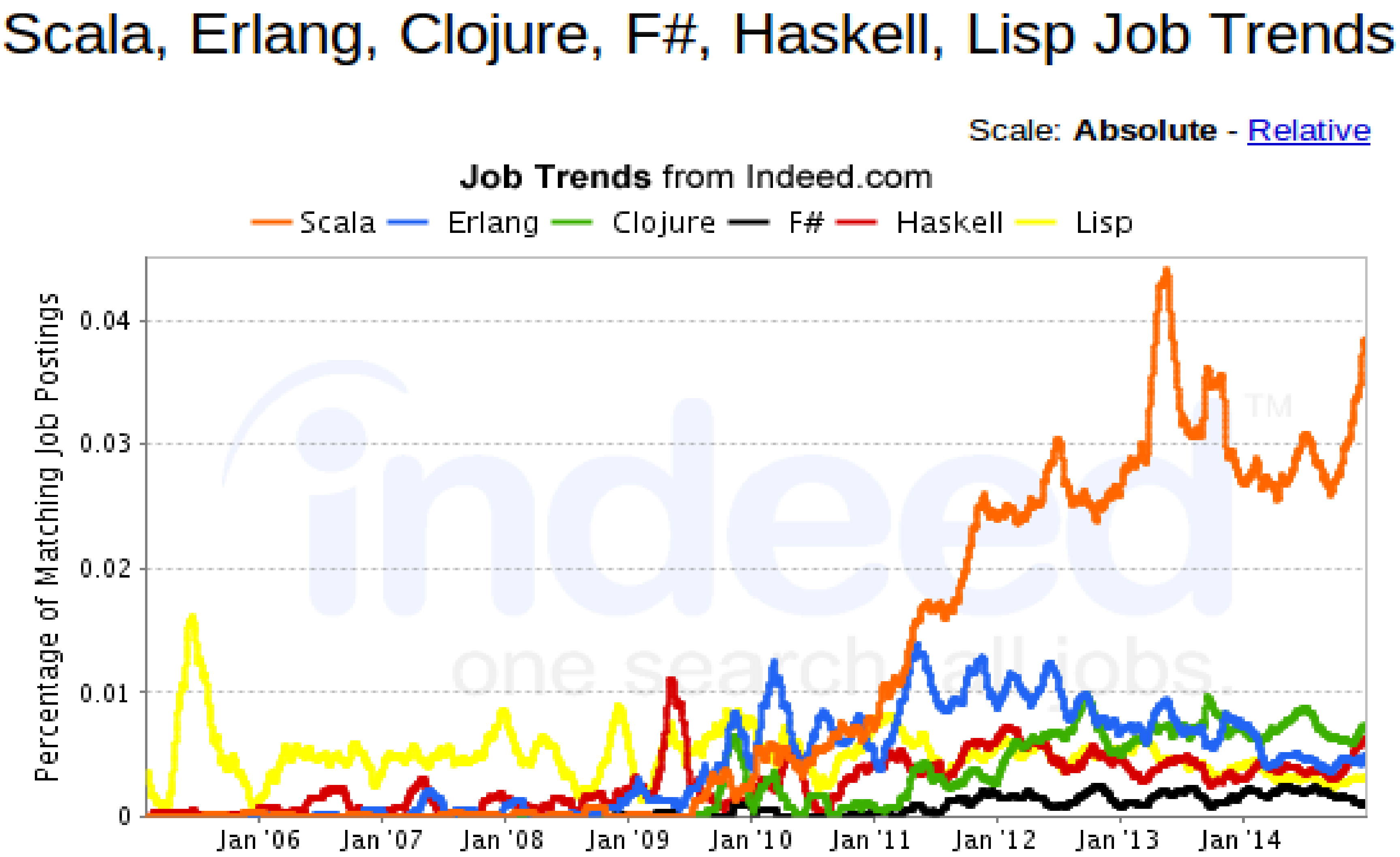
The FUTURE by practicality

- Market share
- Who / where uses
- Domains
- Quality / Quantity
- Job offers

Adoption level?

- Java 8 Lambdas
  - Supposed to be in Java 7
- C++
- C#
- Python Clojures
- JVM: Clojure, Scala, etc.
- SQL – long ago

Language popularity

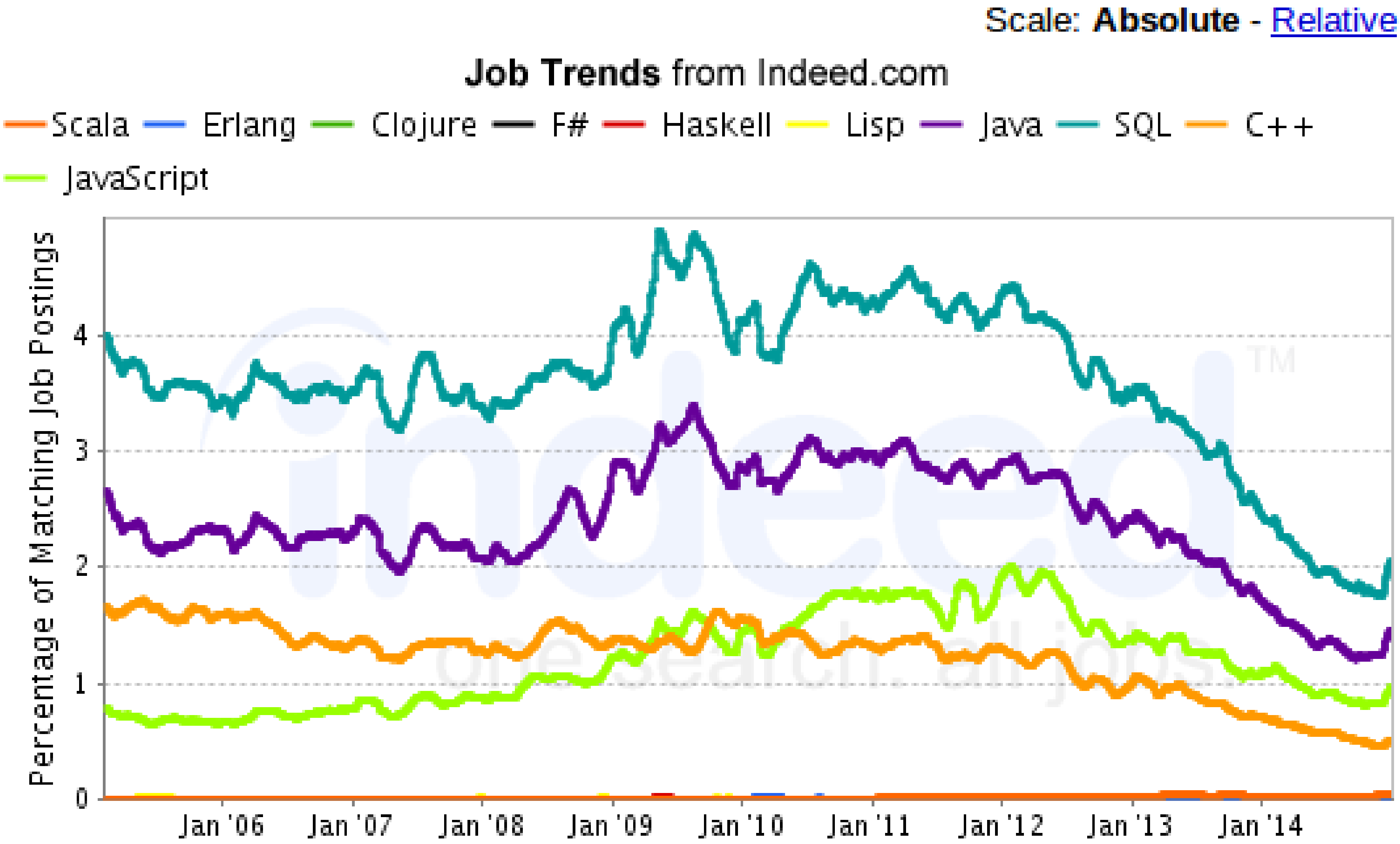


Scala, Erlang, Clojure, F#, Haskell, Lisp Job Trends

Scale: **Absolute** - Relative

Job Trends from Indeed.com

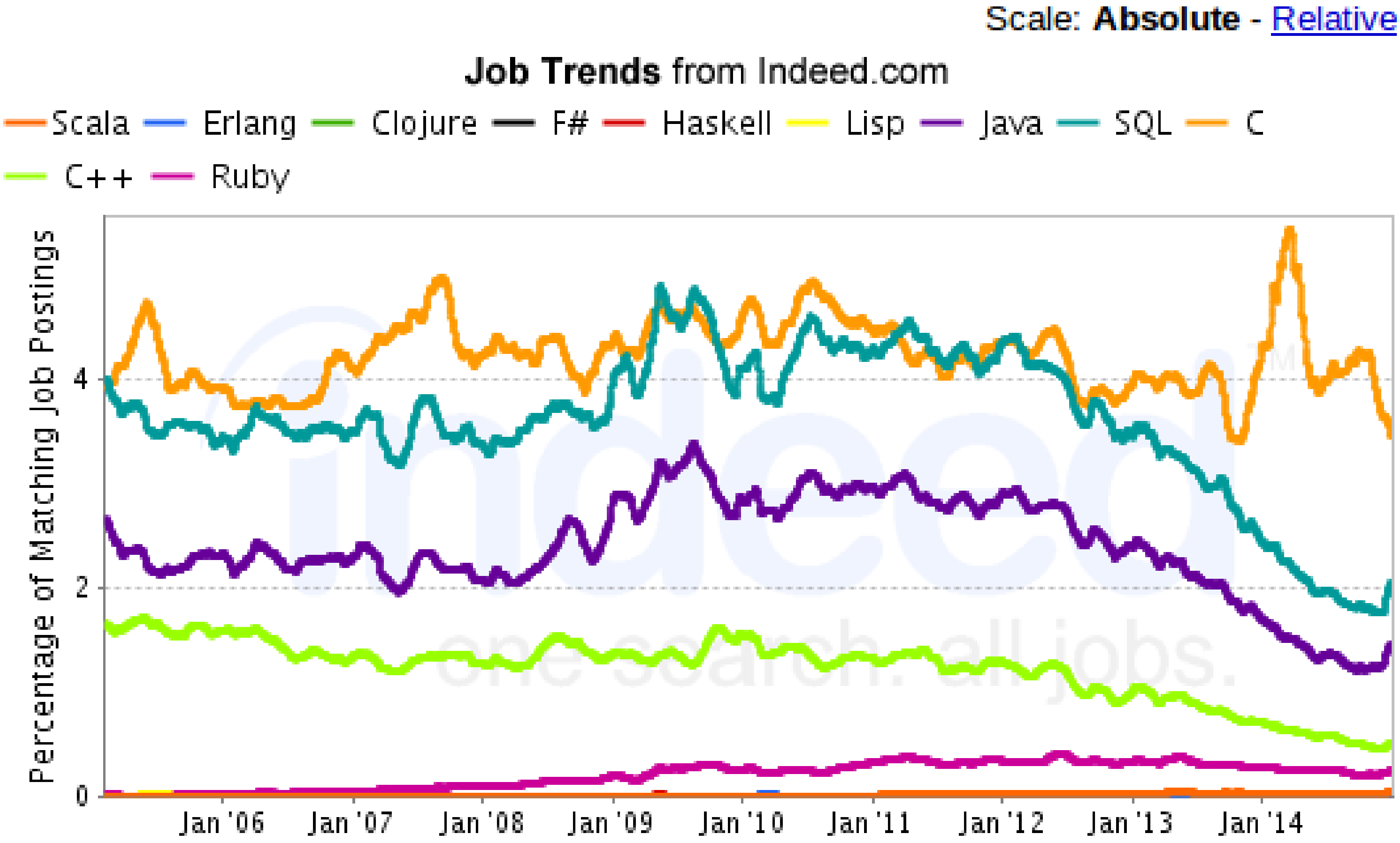
Language popularity in context



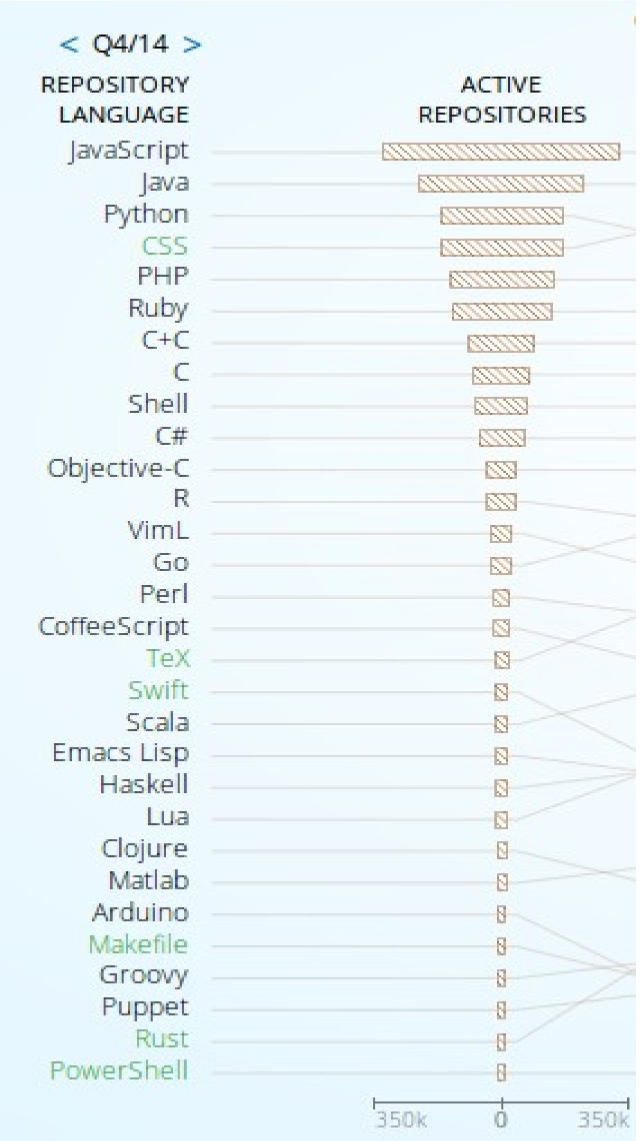
Scale: **Absolute** - Relative

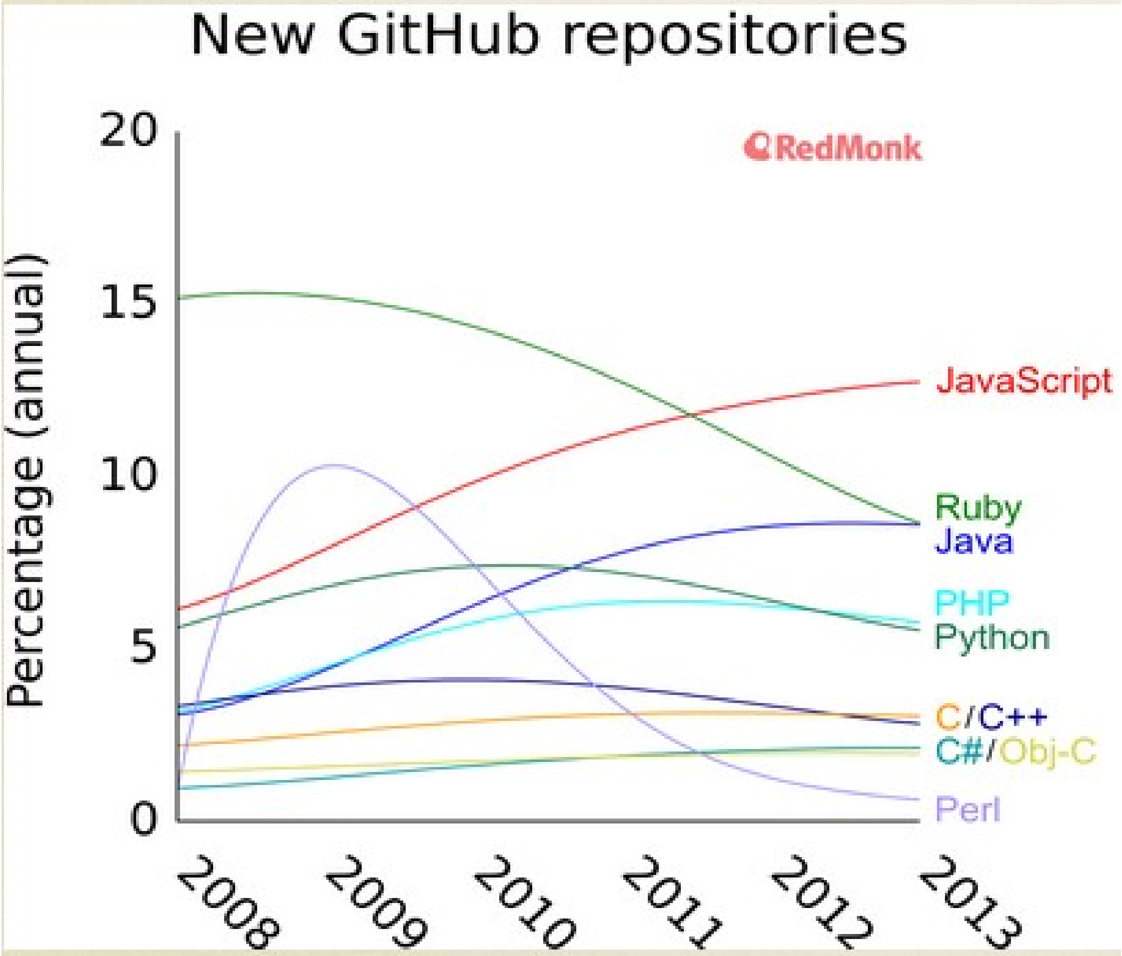
Job Trends from Indeed.com

Language popularity in context 2



Language popularity: GitHub





Language popularity: GitHub 2

http://githut.info/

Language popularity: Tiobe

- Best source of data for language trends
- Contains:
  - What grew most in a year
  - Long range trends
  - Very long range... (1985)
  - Yearly index
- I looked at top 50 languages

Languages across 2014-2015

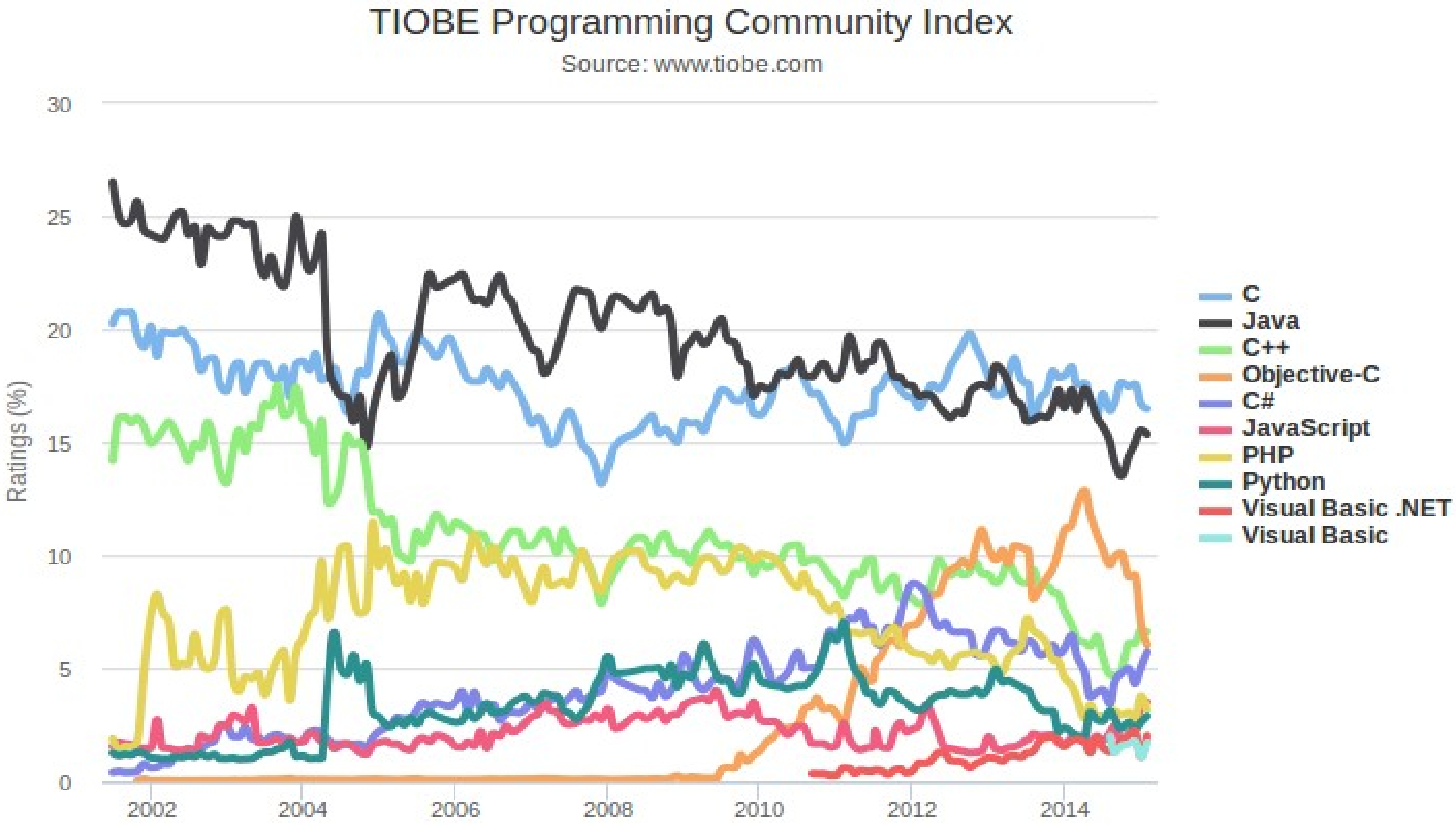
Place now	Previously	Language	Ratings (%)	Change (%)
1	1	C	16,488	-1,85
2	2	Java	15,345	-1,97
3	4	C++	6,612	-0,28
4	3	Objective C	6,024	-5,32
5	5	C#	5,738	-0,71
6	9	JavaScript	3,514	1,58
7	6	PHP	3,170	-1,05
8	8	Python	2,882	0,72
9	10	VB.NET	2,026	0,23
10	-	VB	1,718	1,72

Tiobe ratings

- Query: +"<language> programming"
- 25 search engines
- „The counted hits are normalized for each search engine for all languages in the list.”
  - All languages in the list add up to 100%
- Confidence factor says how many results we take in (filtering)

FP languages @ Tiobe

- Currently none makes it past 14th place (F#)
- None became „Programming Language of the Year” (at least not since 2003)
- They hold > 1% rating, except for F#'s 1,179%
- Reigning languages are still C and Java.
- A number didn't made it to top 50 (Erlang)


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
C
Java
C++
Objective-C
C#
JavaScript
PHP
Python
Visual Basic .NET
Visual Basic

Summarizing fame

- On the rise – yes
- Dominating, must have – no
  - Including the startup advantage somewhat
  - Not strong enough market share
  - There are some backers though

Revolution

- 10101010100100100
- Assembly
- Fortran
- High-level languages
- OO
- FP?

Thought-bending / Another tool

- General truth:
  - Learning shapes mind
  - We think in patterns
- Number of personal statements
  - „I learned Haskell and I suffer”
  - SO questions
- First language FP vs imperative (universities)
- Number of authority statements
  - Including from different fields (Miyamoto Musashi)

Shorter / terser code

- No FP book / tutorial / FAQ I found had data
  - One or two examples
  - „Usually leads to”, „Most of the times it has”
- Examples are just examples
- **No studies to prove the claim**
- Is shorter really better? (Hint: APL)
- Sometimes used as in...

More power

- More power
- Better expressive power
- Better abstractions
- Convenient for the programmer
- **Paul Graham: Blub Paradox**
  - Ties with „makes you think differently”

The Blub Paradox

- „Programming languages vary in power”
- Who uses Assembly now?
- Blub – middle of language power axis
  - Less powerful is obvious
  - More powerful is weird
- Why don't you use most powerful language?
  - Comfort zone
  - Experience

Less bugs!

- Complexity
  - Accidental
  - Essential
- State = state mutation vs immutability
- Referential transparency, aka no (little) side-effects
- **Papers: yes, studies: no**

You can reason about it (algebra of programs)

You can reason about it (algebra of programs)

But you won't

Solves multicore problem

- Immutable
- Less state, less deadlocks
- Referential transparency, aka no side-effects
- Impure things can be contained within pure (monads)
- **Papers: yes, studies: no**

Unusual claims

- Hybrid approaches
- How good OOP works similarly to FP

Summarizing

- What is logical, can be inferred:
  - Concurrency
  - Complexity
- What you can back up with data
  - Prooving it (Coq, Yoneda lemma, program algebra)
- What you can argue:
  - Power, expresiveness
  - Gives you another perspective, changes thinking
- Not really: terser code, market share, popular...

How I see it

- Concurrency is easier
- Less state = less worries
- Makes you into a better programmer
- Streams make for a better thought-flow

- You can reason about it, but will you?
- Yes, on the rise, but not a „must-have”
- More power... well

So, why bother?

Your case, you tell me!

Version 2, WIP

- Work in progress: sorry!
- Previous installments:
- Info page on my blog:

https://lafkblogs.wordpress.com/2015/03/07/why-bother-with-fp/

- Contribute (pull-requests welcome):

https://github.com/LIttleAncientForestKami/why-bother-with-FP