Выбор языка программирования

Александр Чистяков

Clear Technology Group

Несколько слов о себе

- C{E,T}O @ Clear Technology Group
- Преподаватель @ avalon.ru
- Researcher @ ISST Lab, ITMO
- Координатор встреч DevOps-инженеров в Петербурге
- Пишу код

О чем это доклад?

- Я отвечу в самом конце

Что такое «DevOps»?

- Отличный вопрос для собеседования!

Что такое «DevOps»?

- Отличный вопрос для собеседования!
- Термин из прошлого десятилетия

Что такое «DevOps»?

- Отличный вопрос для собеседования!
- Термин из прошлого десятилетия
- Набор практик

Что такое «DevOps»?

- Отличный вопрос для собеседования!
- Термин из прошлого десятилетия
- Набор практик
- CAMS

CAMS

C - Culture

Раз-раз и высший класс!

CAMS

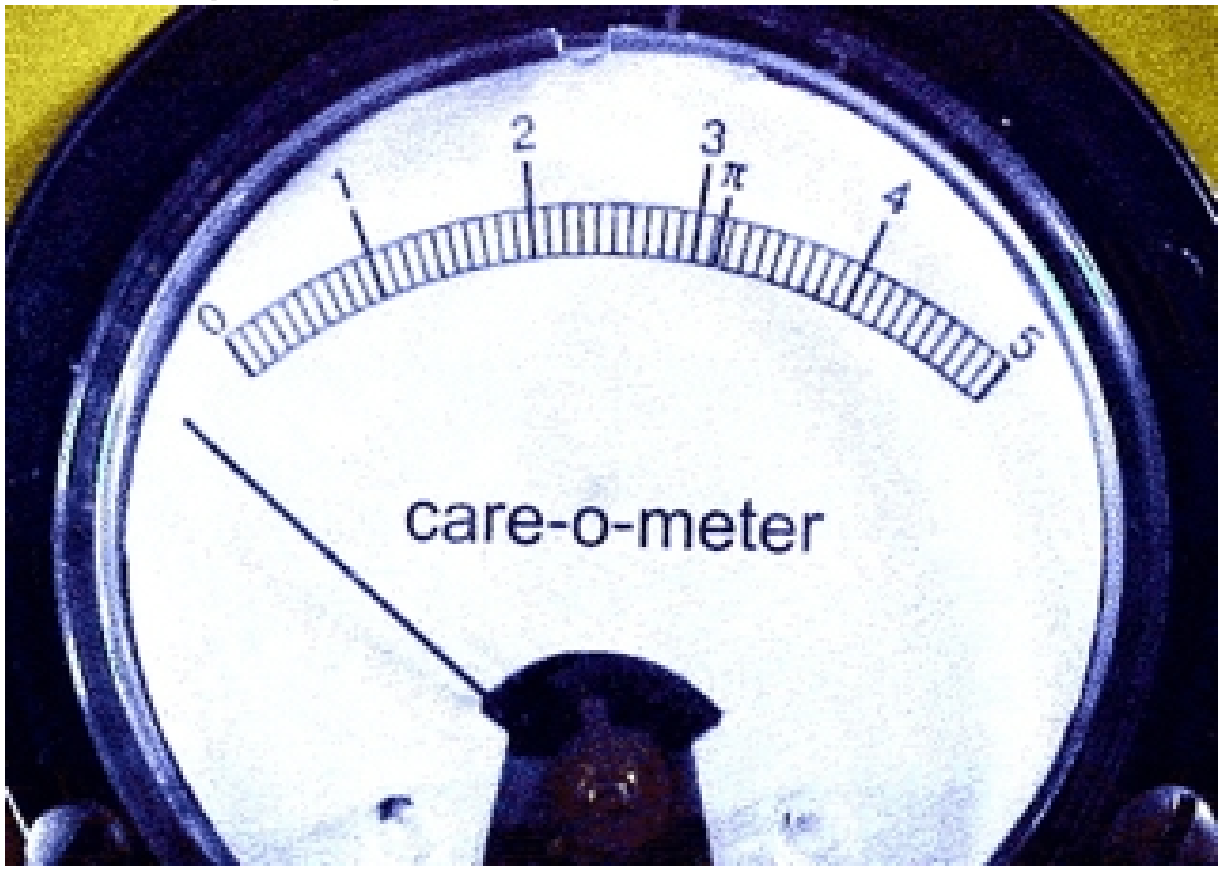
A - Automation

Humans need not apply

CAMS

M - Measurement

Запомните этот прибор!



CAMS

S - Sharing

Information must be free!

CAMS(R)

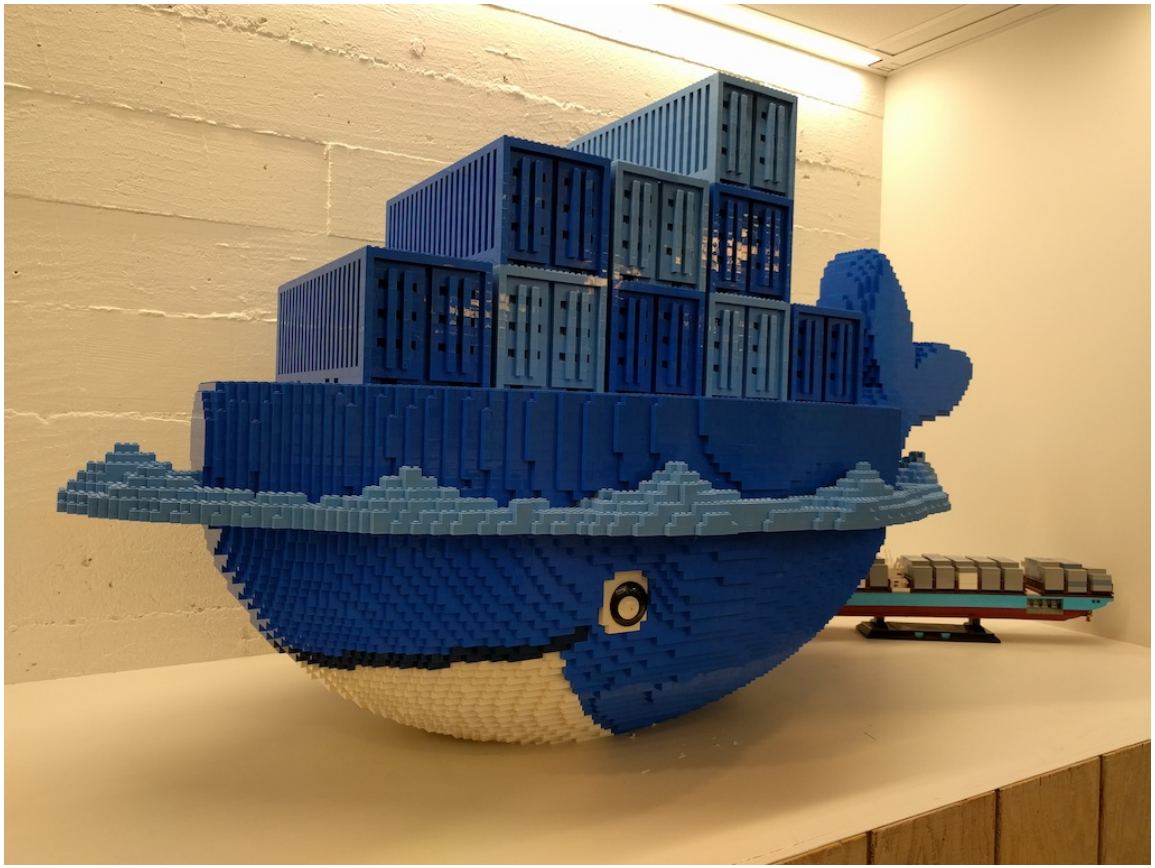
R - Repeatability (плачет в углу)

https://www.opennet.ru/opennews/art.shtml?num=46338

Repeatability

Docker

(на самом деле - нет)



Repeatability

Nixpkgs

(Nixpkgs + Docker)!

- Два варианта
- https://goo.gl/6uxL0M
- https://goo.gl/wkduPv
- Типичный случай аксиомы Эскобара

Немного истории

Что изображено на картинке?

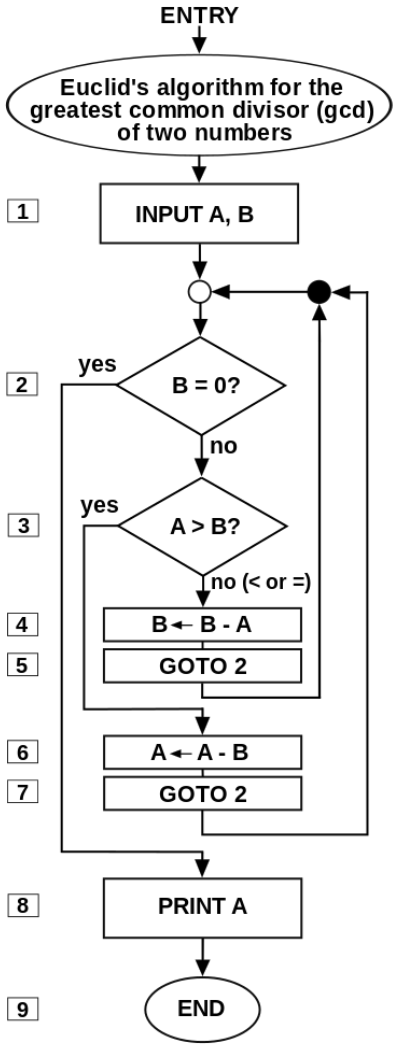
(Поговорим о реальном старье)

Немного истории

Носитель информации 30 лет назад

(Емкость примерно 200 килобайт)

ALGOL-60 и далее

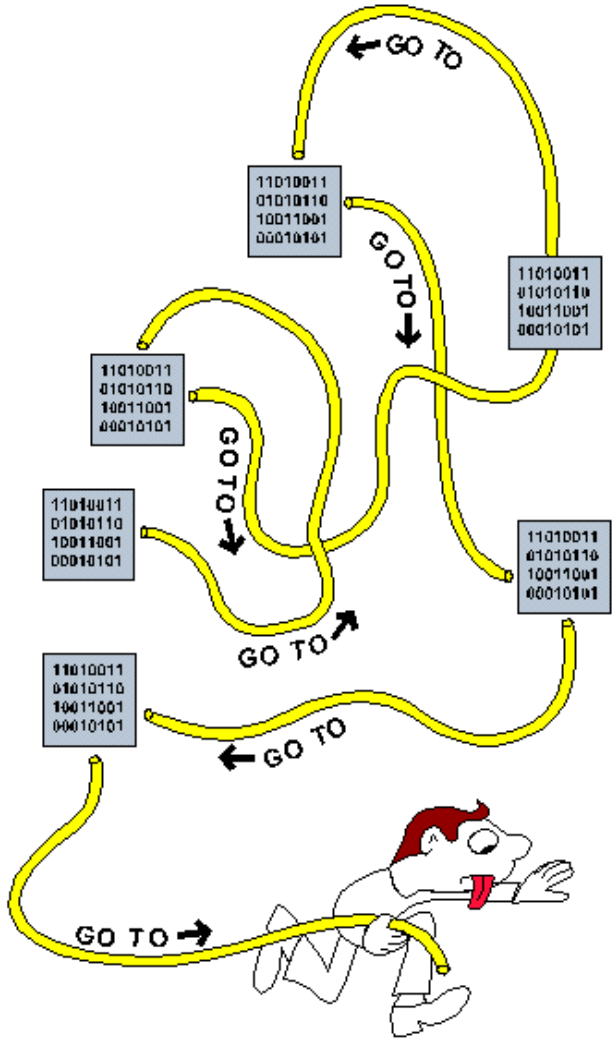


Структурное и процедурное программирование

Корень всех зол (нет, не goto)

Как C-программист под DSP пишет на C#?

В C# нет goto, но это не беда!



Зачем нужно ООР?

- Инкапсуляция, наследование, полиморфизм!
- Пенсия Гради Буча

Зачем на самом деле ООР?

- Инкапсуляция, наследование, полиморфизм!
- Пенсия Гради Буча
- Кошелек Миллера (спасибо Григорию Петрову)
- Закон Деметры
- SOLID

SOLID

- Single responsibility principle

SOLID

- Single responsibility principle
- Open/closed principle

SOLID

- Single responsibility principle
- Open/closed principle
- Liskov substitution principle

SOLID

- Single responsibility principle
- Open/closed principle
- Liskov substitution principle
- Interface segregation principle

SOLID

- Single responsibility principle
- Open/closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

Что-то пошло не так

Objects have failed* (OOPSLA 2002)

* на самом деле нет

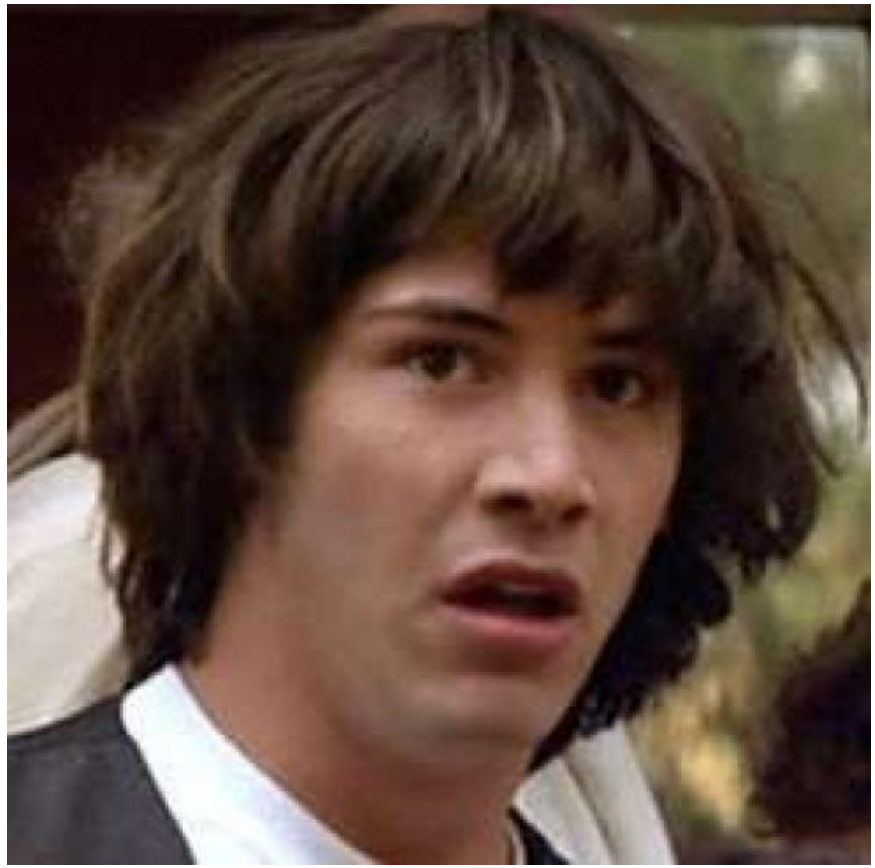
2002+15

Python - lingua franca индустрии

В Python есть всё

В Python есть всё

Зачем тогда что-то еще?



Отнять и поделить

Почему не декриминализуют легкие наркотики?

Хороший Язык Будущего

- Строгая типизация (PHP и JS - плохие)

Хороший Язык Будущего

- Строгая типизация (PHP и JS - плохие)
- (Опциональная) статическая типизация

Опциональная типизация

- PHP: type declarations, 5.0 => 7.0
- Python: type hints, PEP-484
- Python: mypy

Статические анализаторы

- mypy - статический анализатор кода

Статические анализаторы

- mypy - статический анализатор кода
- статический анализатор работает до запуска программы

Статические анализаторы

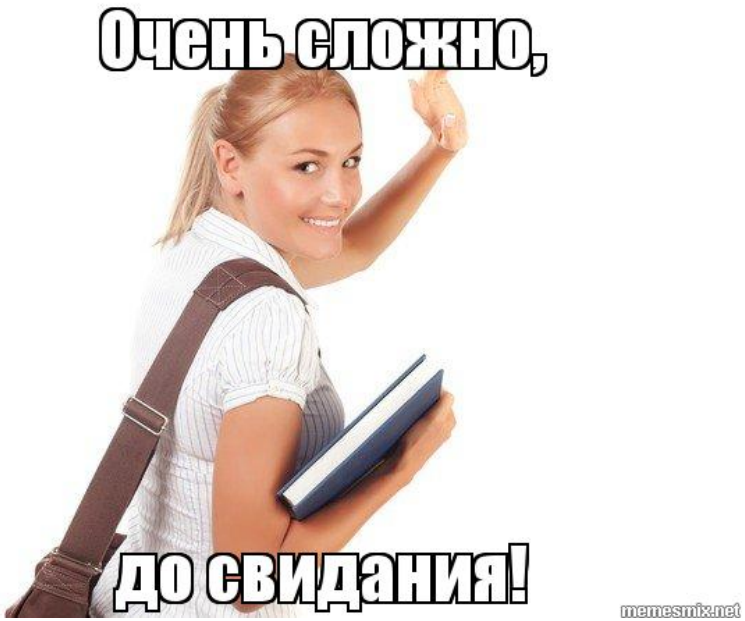
- mypy - статический анализатор кода
- статический анализатор работает до запуска программы
- статический анализатор обобщает идею статической типизации

Анализаторы разных языков

- Ruby: RuboCop
- Perl: Perl::Critic
- Python: Coala, Pylama, mypy
- PHP: PHPLint, PHP Mess Detector

Static Analysis Symposium

- Научная конференция
- Проходила уже 23 раза
- 23 сборника статей примерно по 400 страниц



Хороший Язык Будущего

- Строгая типизация (PHP и JS - плохие)
- (Опциональная) статическая типизация
- Package/vendoring manager

Package managers

- PHP: Composer
- Python: pip
- Perl: cpanminus
- Ruby: bundler

Хороший Язык Будущего

- Строгая типизация (PHP и JS - плохие)
- (Опциональная) статическая типизация
- Package/vendoring manager

Хороший Язык Будущего

- Строгая типизация (PHP и JS - плохие)
- (Опциональная) статическая типизация
- Package/vendoring manager
- Метапрограммирование

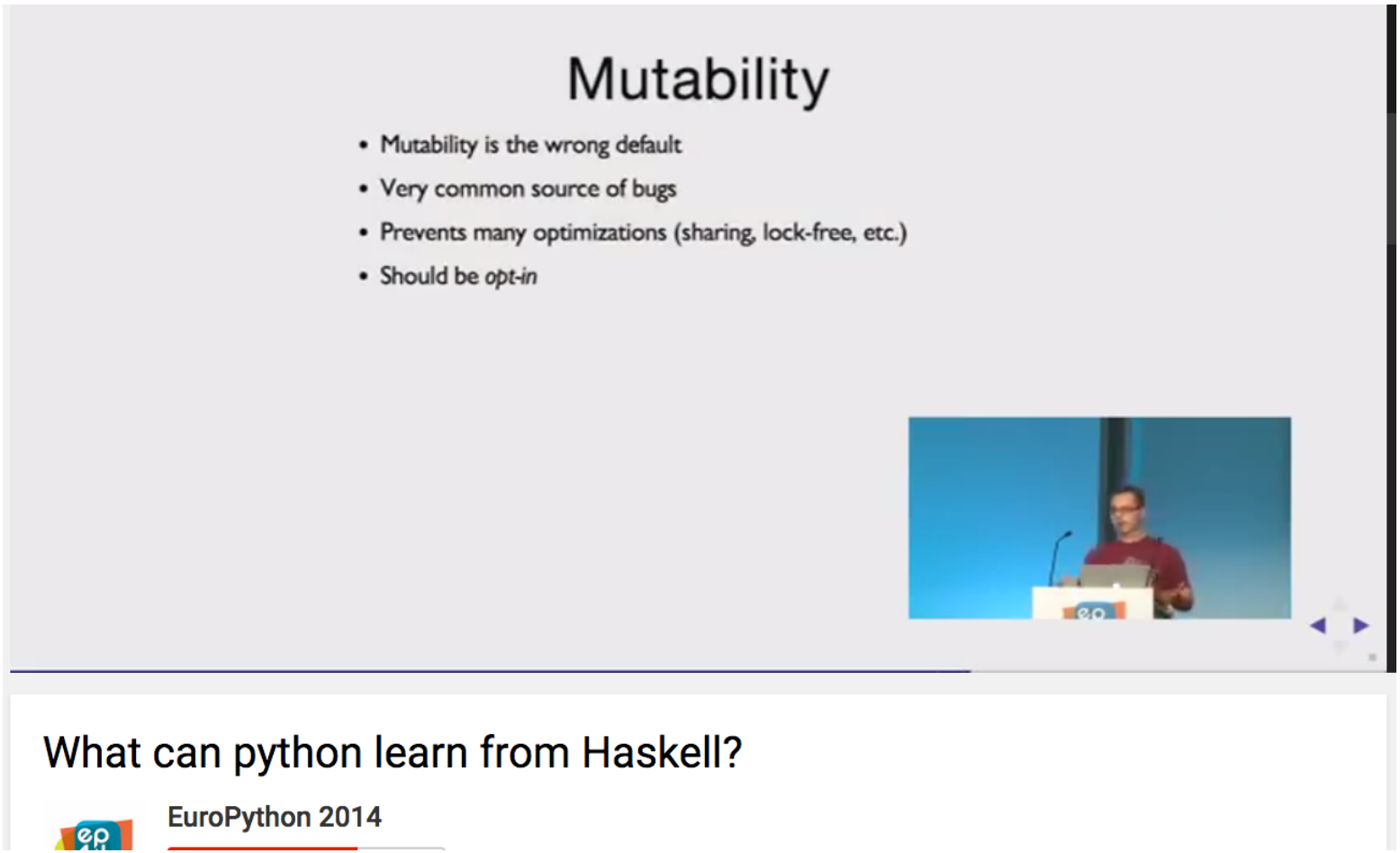
Хороший Язык Будущего

- Строгая типизация (PHP и JS - плохие)
- (Опциональная) статическая типизация
- Package/vendoring manager
- Метапрограммирование
- Иммутабельность

Иммутабельность

Доклад Боба Ипполито в 2014-м

верен и в 2017-м

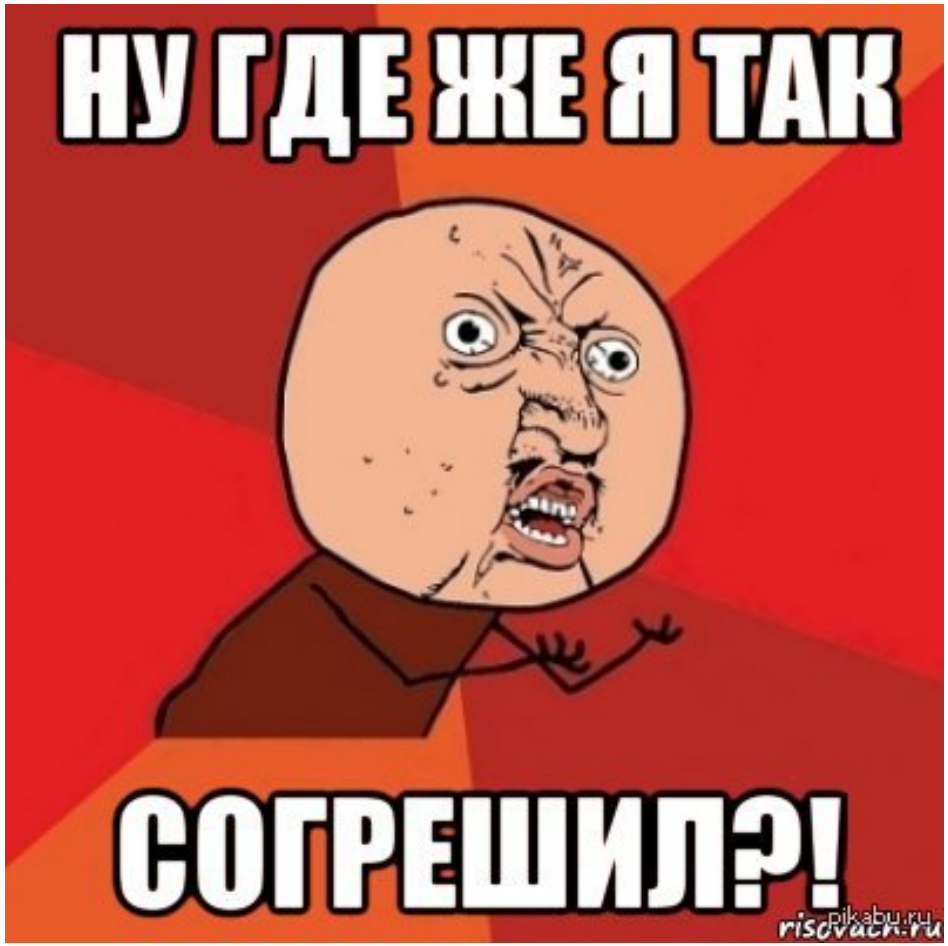


Хороший Язык Будущего

- Строгая типизация (PHP и JS - плохие)
- (Опциональная) статическая типизация
- Package/vendoring manager
- Метапрограммирование
- Иммутабельность
- Null-safety

Метапрограммирование

- Было в C - #ifdef



Метапрограммирование

- Было в C - #ifdef
- Было в Java - аннотации

Метапрограммирование

- Было в C - #ifdef
- Было в Java - аннотации
- Было в LISP - макросы

Сферический в вакууме

- Языку нужна среда исполнения

Сферический в вакууме

- Языку нужна среда исполнения
- JVM

Сферический в вакууме

- Языку нужна среда исполнения
- JVM
- V8

Сферический в вакууме

- Языку нужна среда исполнения
- JVM
- V8
- BEAM

Сферический в вакууме

- Языку нужна среда исполнения
- JVM
- V8
- BEAM
- Golang runtime (not a VM, but...)

A quest for my next PL

https://goo.gl/MS1UfB

Language	Strengths	Weaknesses	Opportunities	Threats
Ruby-like (Ruby, CoffeeScript, Elixir, Crystal)	Neat syntax, one can write a DSL easily	Runtimes are total crap (maybe w/exception of Elixir/BEAM), RoR is a "component-oriented" Delphi-like env	RoR developers seem to be paid well enough (not in Russia, of course)	My personal experience w/Ruby projects was awful (don't ask!)
Nim	Syntax, FP, Strong typing, per-thread GC	Observability should be close to 0 (Nim transpiles to C)	A first Nim developer on a local scene gets everything (can start a meetup, a podcast and an elite dating site)	Does Nim have enough momentum? in 7 years it reached version 0.14
Rust	No GC at all			
SBCL	The Lord used LISP to build Earth		It's easier to beat the averages with something LISP-based (ask Paul Graham!)	There are 3 or so Lispers in SPb and most of them drink too much
Clojure	STM, immutable data structures	JVM is too heavy for infrastructure tasks (?), Clojure is dynamically typed		Can't think of any (is Clojure ideal in terms of number of threats?)
Scala	Scala is multiparadigm enough to keep me learning it next 10 years	Learning curve is steep	A good Scala developer can apply to an EJB2 related position in Luxoft	One can really spend a whole life learning Scala and not become a decent Scala developer at the end of the day
F#	Good IDE,comparing to scala: more functional, full Hindley-Milner style type inference	more bulky syntax than scala	10x less jobs than scala	
Kotlin	Dunno	Dunno	A good Kotlin developer can apply to a position in JB	Does anybody use it outside of JetBrains?
OCaml	It's ML and ML is a	Batteries are included but	A good OCaml developer can	There are no many new projects

Sheet1

Буду гиперполиглотом

http://hyperpolyglot.org

Почему не Golang?

- Очень простой: 25 ключевых слов

Почему не Golang?

- Очень простой: 25 ключевых слов
- Нет метапрограммирования

Почему не Golang?

- Очень простой: 25 ключевых слов
- Нет метапрограммирования
- Нет иммутабельности

Почему не Golang?

- Очень простой: 25 ключевых слов
- Нет метапрограммирования
- Нет иммутабельности
- Нет null-safety

Почему не Golang?

- Очень простой: 25 ключевых слов
- Нет метапрограммирования
- Нет иммутабельности
- Нет null-safety
- Из Golang легко сделать Python

Почему не Golang?

- Очень простой: 25 ключевых слов
- Нет метапрограммирования
- Нет иммутабельности
- Нет null-safety
- Из Golang легко сделать Python
- С вендорингом какая-то боль

Что реально успел?

- Clojure: dynamic, strong
- Elixir: dynamic, strong
- Nim: static, strong, null-unsafe
- Rust: static, strong, null-safe

Как ощущения?

Use libraries, not frameworks!

- Clojure: dynamic, strong
- Elixir: dynamic, strong
- Nim: static, strong, null-unsafe
- Rust: static, strong, null-safe

Use libraries, not frameworks!

- Везде генерируется scaffolding
- Везде есть порт Sinatra
- Везде есть ORM tool

Use libraries, not frameworks!

Есть опасность написать свой фреймворк

Чего еще нет в Python?

Скорости!

Что насчет скорости?

Динамические языки

- JIT compilers

Что насчет скорости?

Динамические языки

- JIT compilers
- GraalVM

Что насчет скорости?

Динамические языки

- JIT compilers
- GraalVM
- Truffle framework

Что насчет скорости?

Статически типизированные языки

- Zero-cost abstractions

Haskell

Как открыть ВАЗ 2101 без ключа?

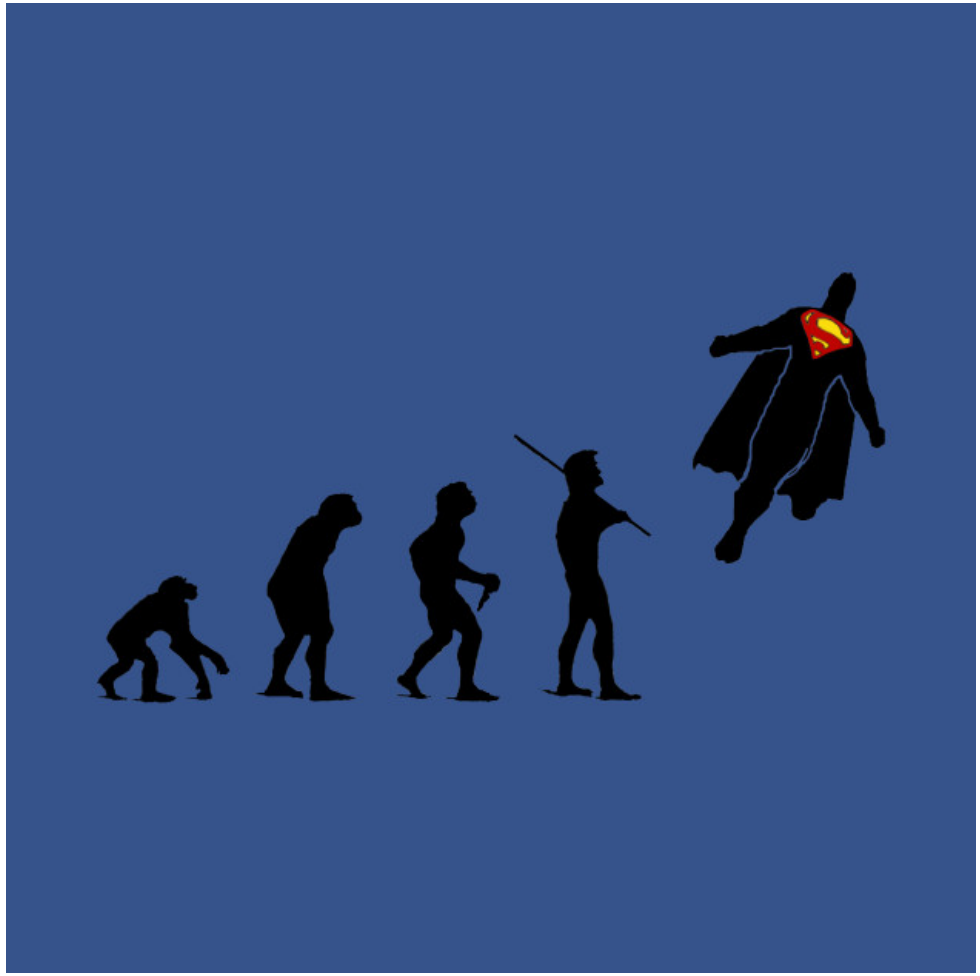
(Гораздо легче, чем пройти курс по Haskell*)

Героическое фэнтези

Типичный состав команды

- Лоховатый главный герой (с потенциалом)
- Дева в беде (муза героя)
- Болтливый друг героя (без потенциала)
- Воительница

Мужские роли играю я



Выводы

- DevOps - из прошлого десятилетия
- LISP - из 1958-го года
- Я не знаю, что будет дальше
- Я не знаю, какой язык лучший
- Поэтому писать надо на всем
- «Но не пиши на Коболе, если можешь этого избежать»

Вопросы, пожалуйста?

- ...?
- ...?
- ...?

That’s all, folks!

- alex@cleartech.group
- https://telegram.me/lhommequipleure