Clojure - Lisp для JVM, но зачем ?

Clojure - что это такое

- Lisp - семейство
- Динамическая типизация
- Сильный упор на ФП
- Многопоточность
- Интеграция с хост-платформой (изначально JVM), есть порты на CLR - ClojureCLR и JavaScript - ClojureScript
- Версия 1.0 была выпущена в 2009 году
- Язык практичный, с компромиссами и без стремления к чистоте

Lisp - Lots of insignificant parenthesis

- Печально известные))))))))))
- На самом деле скобочек не так уж и много, особенно если посчитать все дополнительные знаки
- Упрощают метапрограммирование

```java
public static List<String> someFunction() {
   List<String> list = new ArrayList<>();
   list.add("a");
   list.add("b");
   list.add("c");
   return list;
}
```

18

```clojure
(defn some-function []
 (doto (ArrayList.)
   (.add "a")
   (.add "b")
   (.add "c")))
```

14

Что вижу я

```
(defn lispy-function []
 (let [a (repeatedly #(rand-int 10))
       b (repeatedly #(rand-int 20))]
   (take 10 (map vector
                 (filter odd? a)
                 (filter even? b)))
```

Что видит человек,
впервый видящий Lisp

```
(defn lispy-function []
 (let [a (repeatedly #(rand-int 10))
       b (repeatedly #(rand-int 20))]

   (take 10 (map vector
                 (filter odd? a)
                 (filter even? b)))))
```

Пример кода для работы с REST API

```
(-> @(http/request
      {:request-method :get
       :url            "https://api.github.com/search/repositories"
       :headers        {"Accept"     "application/vnd.github.v3+json"
                        "User-Agent" "Awesome-Octocat-App"}
       :query-params   {"q"    "stars:>=100 fork:true language:clojure"
                        "sort" "stars"}})
   :body
   (io/reader)
   (json/parse-stream true)
   :items
   (first))
```

Рекламная пауза

Netflix

“A lot of the best programmers and the most productive programmers I know are writing everything in Clojure and swearing by it, and then just producing ridiculously sophisticated things in a very short time.”

Adrian Cockcroft

VP Cloud Architecture Strategy at AWS

https://thenewstack.io/the-new-stack-makers-adrian-cockcroft-on-sun-netflix-clojure-go-docker-and-more/

Рекламная пауза

Walmart

“Clojure uses anywhere from 5 to 10 times less code than other programming languages, speeding development time, reducing the instances of bugs, and lowering maintenance costs”

Anthony Marcar, Senior Architect - WalmartLabs

https://blog.cognitect.com/blog/2015/6/30/walmart-runs-clojure-at-scale

Недавнее (01.11.2018) обсуждение на HN

“I wrote and deployed (to production) some Clojure code at Netflix just yesterday. Among other things at Netflix the Mantis Query Language (MQL an SQL for streaming data) which ferries around approximately 2 trillion events every day for operational analysis (SPS alerting, quality of experience metrics, debugging production, etc) is written entirely in Clojure.

This runs in nearly every critical service, ~3000 ASGs and easily > 100k servers and Clojure allows us to also compile it for our NodeJS services as well”

https://news.ycombinator.com/item?id=18345243

Кто ещё пользуется Clojure

- Apple
- Atlassian
- IBM
- Puppet
- ThoughtWorks
- и другие организации

https://jobs.apple.com/en-us/details/114424334/senior-clojure-software-engineer
https://www.youtube.com/watch?v=BsLiPt90HDo
https://thoughtworks.github.io/p2/issue09/two-months-early/
https://puppetlabs.com/blog/introducing-puppetdb-put-your-data-to-work
https://www.youtube.com/watch?v=3QR8meTrh5g
https://clojure.org/community/success_stories

Где Clojure не прижилась

- Facebook
- Amazon

https://www.reddit.com/r/Clojure/comments/68r4lz/one_of_facebook_projects_migrating_from_clojure/
https://news.ycombinator.com/item?id=18346154

Недостатки Clojure

- Сложна для новичков: очень краткая документация, странные исключения и ошибки компилятора, уши JVM частенько вылезают.
- Не так широко распространена как другие языки для JVM
- Тяжелее найти работу/программистов
- Долгий старт (несколько секунд на запуск lein repl)
- Динамическая типизация

Преимущества Clojure

- Динамическая типизация
- Разработка в REPL заложена изначально
- Макросистема заложена изначально
- Эффективная реализация функциональных (неизменямых) коллекций
- Мультиметоды и протоколы для полиморфизма и интеграции с ООП
- Мощная стандартная библиотека для работы с данными
- Удобные примитивы для работы с многопоточными приложениями

Метапрограммирование в Clojure

```
(let [reader (io/reader (io/file "myfile.txt"))]
  (try
    (do-something reader)
    (finally
      (.close reader))))
```

Метапрограммирование в Clojure

```clojure
(defmacro with-open [[sym expr] & body]
 `(let [~sym ~expr]
    (try
      ~@body
      (finally
        (.close ~sym)))))
```

Метапрограммирование в Clojure

```
(with-open [reader (io/reader (io/file "myfile.txt"))]
  (do-something reader))
```

macroexpansion

```
(let [reader (io/reader (io/file "myfile.txt"))]
  (try
    (do-something reader)
    (finally
      (.close reader))))
```

Стандартная библиотека

- map, filter, remove, reduce
- group-by, sort-by, compare, partition
- count, distinct
- walk, prewalk, postwalk
- frequencies
- Функции для работы с множествами: union, intersection, difference, select
- И многое другое

Протоколы и мультиметоды

- Мультиметоды - способ реализации полиморфизма с произвольной функцией диспетчеризации
- Действуют по принципу “открытой системы” - можно расширять поведение, добавлять собственные обработчики
- Протоколы - упрощённая реализация мультиметодов, использующая особенности хост-платформы для увеличения производительности

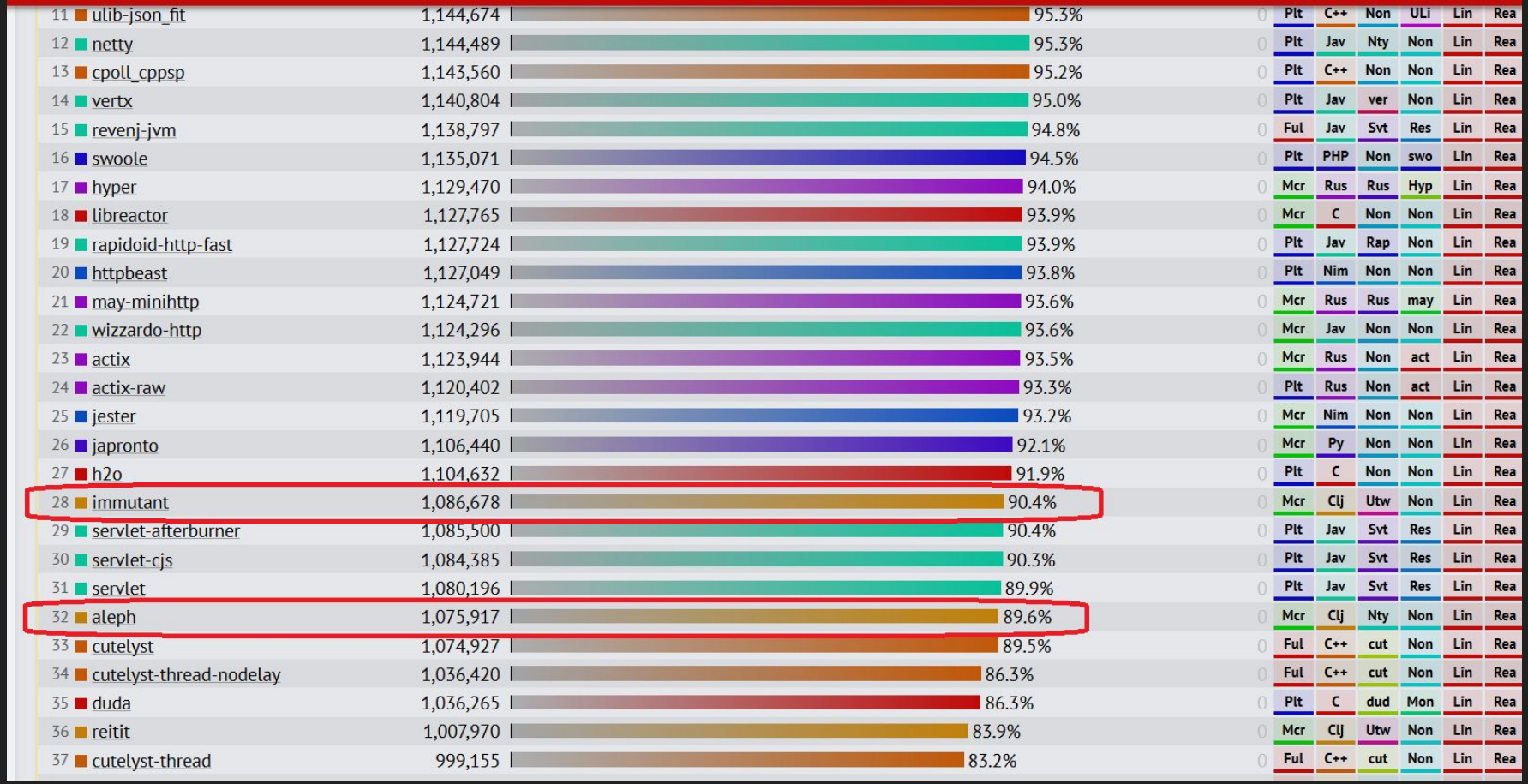
Многопоточность

- Все стандартные коллекции потокобезопасны
- ФП и упор на чистоту позволяет изолировать места, которые могут быть проблематичны с точки зрения многопоточности
- Удобные инструменты для работы с многопоточностью
- STM - Software Transactional Memory
- Atom - хранилище данных с CAS-семантикой
- Coroutines/Goroutines

Преимущества Clojure

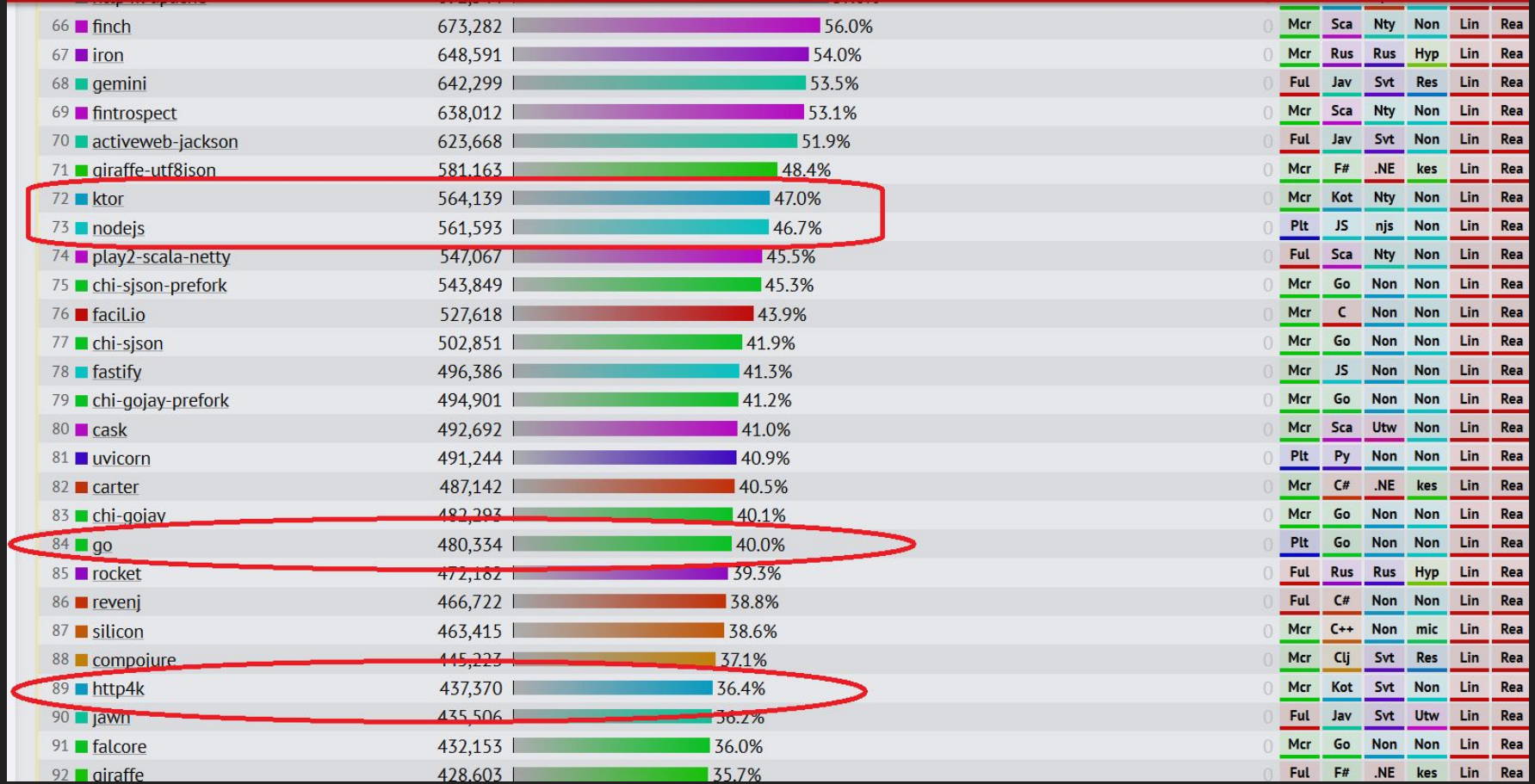
- Тесная интеграция с хост-платформой и простой interop
- Доступ к экосистеме JVM - профилировщики, анализаторы хипа и всё что работает на JVM можно использовать с Clojure
- Достойная производительность
- Упор на работу с данными и функциональное программирование
- Тенденция к малому количеству слоёв абстракции - предсказуемое поведение, легче понять работу системы.

JSON serialization



11	ulib-json_fit	1,144,674	95.3%	0	Plt	C++	Non	ULi	Lin	Rea
12	netty	1,144,489	95.3%	0	Plt	Jav	Nty	Non	Lin	Rea
13	cpoll_cppsp	1,143,560	95.2%	0	Plt	C++	Non	Non	Lin	Rea
14	vertx	1,140,804	95.0%	0	Plt	Jav	ver	Non	Lin	Rea
15	revenj-jvm	1,138,797	94.8%	0	Ful	Jav	Svt	Res	Lin	Rea
16	swoole	1,135,071	94.5%	0	Plt	PHP	Non	swo	Lin	Rea
17	hyper	1,129,470	94.0%	0	Mcr	Rus	Rus	Hyp	Lin	Rea
18	libreactor	1,127,765	93.9%	0	Mcr	C	Non	Non	Lin	Rea
19	rapidoid-http-fast	1,127,724	93.9%	0	Plt	Jav	Rap	Non	Lin	Rea
20	httpbeast	1,127,049	93.8%	0	Plt	Nim	Non	Non	Lin	Rea
21	may-minihttp	1,124,721	93.6%	0	Mcr	Rus	Rus	may	Lin	Rea
22	wizzardo-http	1,124,296	93.6%	0	Mcr	Jav	Non	Non	Lin	Rea
23	actix	1,123,944	93.5%	0	Mcr	Rus	Non	act	Lin	Rea
24	actix-raw	1,120,402	93.3%	0	Plt	Rus	Non	act	Lin	Rea
25	jester	1,119,705	93.2%	0	Mcr	Nim	Non	Non	Lin	Rea
26	japronto	1,106,440	92.1%	0	Mcr	Py	Non	Non	Lin	Rea
27	h2o	1,104,632	91.9%	0	Plt	C	Non	Non	Lin	Rea
28	immutant	1,086,678	90.4%	0	Mcr	Clj	Utw	Non	Lin	Rea
29	servlet-afterburner	1,085,500	90.4%	0	Plt	Jav	Svt	Res	Lin	Rea
30	servlet-cjs	1,084,385	90.3%	0	Plt	Jav	Svt	Res	Lin	Rea
31	servlet	1,080,196	89.9%	0	Plt	Jav	Svt	Res	Lin	Rea
32	aleph	1,075,917	89.6%	0	Mcr	Clj	Nty	Non	Lin	Rea
33	cutelyst	1,074,927	89.5%	0	Ful	C++	cut	Non	Lin	Rea
34	cutelyst-thread-nodelay	1,036,420	86.3%	0	Ful	C++	cut	Non	Lin	Rea
35	duda	1,036,265	86.3%	0	Plt	C	dud	Mon	Lin	Rea
36	reitit	1,007,970	83.9%	0	Mcr	Clj	Utw	Non	Lin	Rea
37	cutelyst-thread	999,155	83.2%	0	Ful	C++	cut	Non	Lin	Rea

JSON serialization



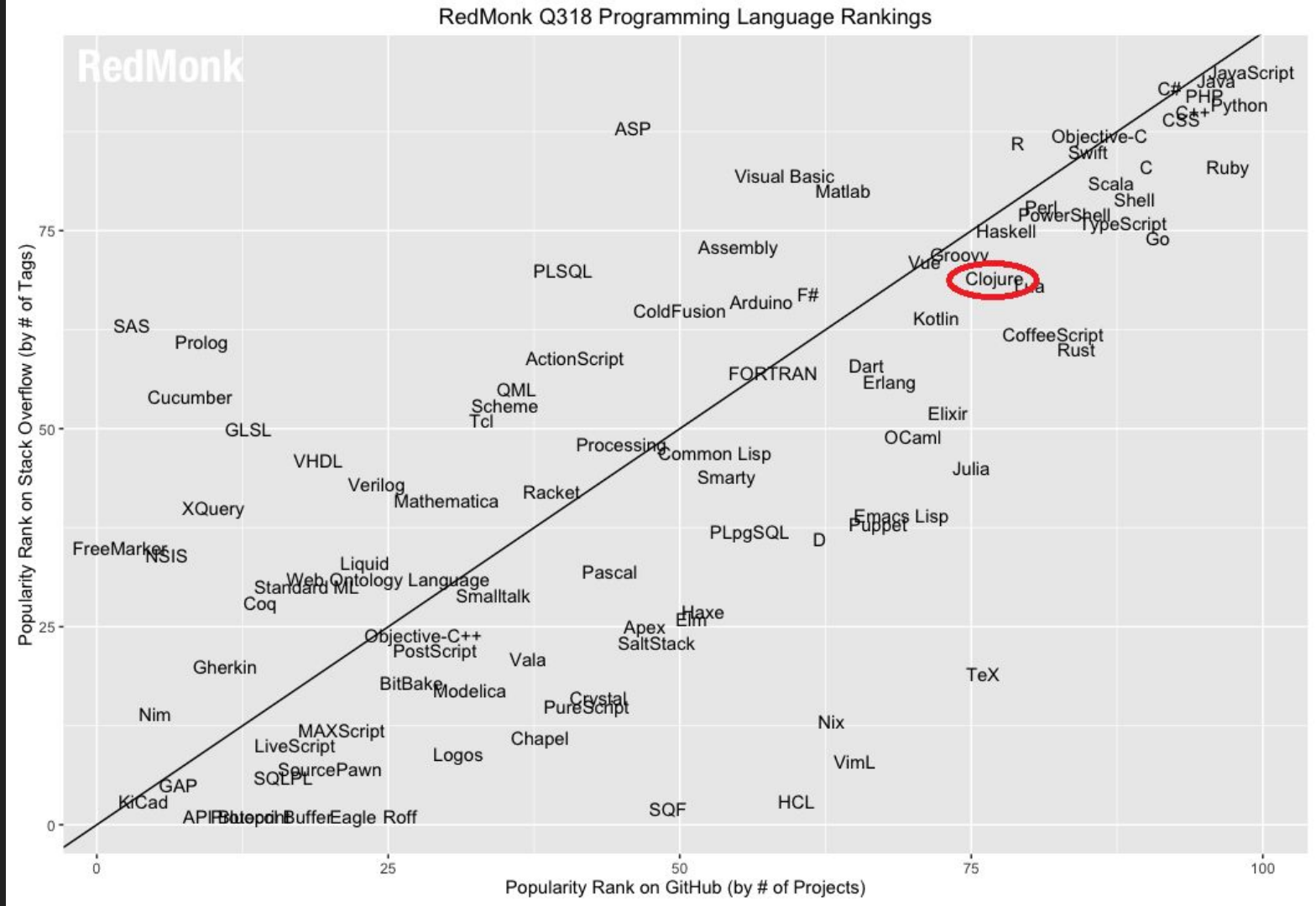
66	finch	673,282	56.0%	0	Mcr	Sca	Nty	Non	Lin	Rea
67	iron	648,591	54.0%	0	Mcr	Rus	Rus	Hyp	Lin	Rea
68	gemini	642,299	53.5%	0	Ful	Jav	Svt	Res	Lin	Rea
69	fintrospect	638,012	53.1%	0	Mcr	Sca	Nty	Non	Lin	Rea
70	activeweb-jackson	623,668	51.9%	0	Ful	Jav	Svt	Non	Lin	Rea
71	giraffe-utf8ison	581,163	48.4%	0	Mcr	F#	.NE	kes	Lin	Rea
72	ktor	564,139	47.0%	0	Mcr	Kot	Nty	Non	Lin	Rea
73	nodejs	561,593	46.7%	0	Plt	JS	njs	Non	Lin	Rea
74	play2-scala-netty	547,067	45.5%	0	Ful	Sca	Nty	Non	Lin	Rea
75	chi-sjson-prefork	543,849	45.3%	0	Mcr	Go	Non	Non	Lin	Rea
76	facil.io	527,618	43.9%	0	Mcr	C	Non	Non	Lin	Rea
77	chi-sjson	502,851	41.9%	0	Mcr	Go	Non	Non	Lin	Rea
78	fastify	496,386	41.3%	0	Mcr	JS	Non	Non	Lin	Rea
79	chi-gojay-prefork	494,901	41.2%	0	Mcr	Go	Non	Non	Lin	Rea
80	cask	492,692	41.0%	0	Mcr	Sca	Utw	Non	Lin	Rea
81	uvicorn	491,244	40.9%	0	Plt	Py	Non	Non	Lin	Rea
82	carter	487,142	40.5%	0	Mcr	C#	.NE	kes	Lin	Rea
83	chi-gojay	482,293	40.1%	0	Mcr	Go	Non	Non	Lin	Rea
84	go	480,334	40.0%	0	Plt	Go	Non	Non	Lin	Rea
85	rocket	472,182	39.3%	0	Ful	Rus	Rus	Hyp	Lin	Rea
86	revenj	466,722	38.8%	0	Ful	C#	Non	Non	Lin	Rea
87	silicon	463,415	38.6%	0	Mcr	C++	Non	mic	Lin	Rea
88	compojure	445,223	37.1%	0	Mcr	Clj	Svt	Res	Lin	Rea
89	http4k	437,370	36.4%	0	Mcr	Kot	Svt	Non	Lin	Rea
90	jawn	435,506	36.2%	0	Ful	Jav	Svt	Utw	Lin	Rea
91	falcore	432,153	36.0%	0	Mcr	Go	Non	Non	Lin	Rea
92	giraffe	428,603	35.7%	0	Ful	F#	.NE	kes	Lin	Rea

JSON serialization

191	tio-mvc	139,657	11.6%	0	Mcr	Jav	t-i	Non	Lin	Rea
192	pyramid	138,391	11.5%	0	Ful	Py	Non	Mei	Lin	Rea
193	vapor	135,394	11.3%	0	Ful	Swi	Non	Non	Lin	Rea
194	http-kit	133,936	11.1%	0	Plt	Clj	Rin	Non	Lin	Rea
195	django-py3	129,911	10.8%	0	Ful	Py	Non	Mei	Lin	Rea
196	servlet3-sync	125,801	10.5%	0	Plt	Jav	Svt	tom	Lin	Rea
197	ringojs	117,201	9.8%	0	Plt	JS	Jty	Non	Lin	Rea
198	activeweb	112,369	9.4%	0	Ful	Jav	Svt	Non	Lin	Rea
199	ktor-jetty	108,863	9.1%	0	Mcr	Kot	Jty	Non	Lin	Rea
200	phalcon-micro	106,490	8.9%	0	Mcr	PHP	Non	ngx	Lin	Rea
201	falcon-pypy2	104,825	8.7%	0	Mcr	Py	Non	Tor	Lin	Rea
202	spring	104,566	8.7%	0	Ful	Jav	tom	Non	Lin	Rea
203	duct-aleph	104,454	8.7%	0	Mcr	Clj	Nty	Non	Lin	Rea
204	express-chakra	101,263	8.4%	0	Mcr	JS	Non	Non	Lin	Rea
205	weppy-pypy2	100,806	8.4%	0	Ful	Py	Tor	Non	Lin	Rea
206	sinatra-sequel-postgres-torquebox-jruby	100,090	8.3%	0	Mcr	Rby	Rac	Tor	Lin	Rea
207	sinatra-sequel-torquebox-jruby	98,329	8.2%	0	Mcr	Rby	Rac	Tor	Lin	Rea
208	bottle-pypy2	96,269	8.0%	0	Mcr	Py	Tor	Non	Lin	Rea
209	sinatra-sequel-postgres	94,138	7.8%	0	Mcr	Rby	Rac	Pum	Lin	Rea
210	sinatra-sequel	94,136	7.8%	0	Mcr	Rby	Rac	Pum	Lin	Rea
211	duct-httpkit	93,061	7.7%	0	Mcr	Clj	Rin	Non	Lin	Rea
212	sinatra	92,266	7.7%	0	Mcr	Rby	Rac	Pum	Lin	Rea
213	sinatra-postgres	91,721	7.6%	0	Mcr	Rby	Rac	Pum	Lin	Rea
214	kitura-gcd	88,546	7.4%	0	Mcr	Swi	kit	kit	Lin	Rea
215	kitura-gcd-mongodb	87,743	7.3%	0	Mcr	Swi	kit	kit	Lin	Rea
216	api_hour	87,462	7.3%	0	Mcr	Py	asy	Gun	Lin	Rea
217	sinatra-sequel-postgres-unicorn-mri	86,944	7.2%	0	Mcr	Rby	Rac	Uni	Lin	Rea

Состояние языка и экосистемы

- Редакторы: Cursive, emacs, vscode, vi
- Системы управления зависимостями (Leiningen, Boot, deps)
- Большое количество библиотек для всевозможных задач
- Возможность использовать библиотеки для JVM


RedMonk Q318 Programming Language Rankings
RedMonk
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)
0
25
50
75
100
JavaScript
Java
C#
PHP
Python
C++
CSS
R
Objective-C
Swift
C
Ruby
Scala
Shell
Perl
PowerShell
TypeScript
Go
Haskell
Groovy
Vue
Clojure
F#
Kotlin
CoffeeScript
Rust
Dart
Erlang
Elixir
OCaml
Julia
Emacs Lisp
Puppet
D
PLpgSQL
Haxe
Elm
Apex
SaltStack
TeX
Nix
VimL
HCL
SQF
Crystal
PureScript
Chapel
Logos
Vala
Pascal
Smalltalk
Racket
Mathematica
Common Lisp
Smarty
Processing
Assembly
Visual Basic
Matlab
ASP
PLSQL
ColdFusion
Arduino
ActionScript
FORTRAN
QML
Scheme
Tcl
SAS
Prolog
Cucumber
GLSL
VHDL
Verilog
XQuery
FreeMarker
NSIS
Liquid
Web Ontology Language
Standard ML
Coq
Objective-C++
PostScript
Gherkin
BitBake
Modelica
Nim
MAXScript
LiveScript
SourcePawn
SQLPL
GAP
KiCad
Eagle
Roff

JVM Ecosystem Report 2018

What is the principal JVM language you use for your main applications?

- Java - 90%
- Clojure - 3%
- Kotlin - 2.42%
- Groovy - 2.36%
- Scala - 1.83%
- Other - 0.60%

https://snyk.io/blog/jvm-ecosystem-report-2018

Что можно (нужно) разрабатывать на Clojure

- REST/CQRS бэкенды
- Системы для обработки и анализа данных
- Практически любая backend-разработка которая по каким-то причинам не попала в предыдущие два пункта
- SWING приложения
- Fullstack разработка на Clojure + ClojureScript

Что не стоит разрабатывать на Clojure

- Системы реального времени
- Числодробилки

- Программы, которые совершают много манипуляций с байтами и битами (сдвиги, побитовые операции)
- Программы, где есть много массивов

- Скрипты и консольные утилиты
- Android-приложения (впрочем некоторые пишут на React Native + ClojureScript)

Интерактивная разработка

- Тесная интеграция с редактором/IDE
- Не тратится время на пересборку/деплой/перезапуск
- Заставляет писать код маленькими, легко тестируемыми функциями
- Отладчик становится практически не нужен
- Работа “внутри” программы - нет перезапусков, не теряется состояние
- Возможность подключаться к работающим системам и проверять их состояние, а также менять поведение без перезапуска
- Можно встроить nREPL даже в Java-приложение

Live REPL development demo

https://jum.dshilov.me

Заключение

- Этим пользуются и весьма успешно
- Не надо бояться скобок
- В языке есть интересные идеи, которые помогают в разработке программ
- Русскоязычное сообщество в Telegram: @clojure_ru

Репозиторий с исходниками чата и полезными ссылками:

github.com/shilder/jum