Vlad Kozin

Clojure, **ClojureScript**, **Racket**, **Redex**, **JavaScript**, **OMeta**, **meta-programming**
Fall'13 Recurse Center (aka Hacker School) alum

Corporate ladder

Since Apr 2017 — *Programmer* at Droit (London, UK)

Same as before but with obligatory daily commute.

2015-2017 — *Programmer/Consultant* at Droit (New York, USA - remote)

Building an expert system for compliant trading. Sneaking Clojure/ClojureScript into unsuspecting financial giants. On any given day I could be designing DSLs, implementing compilers, parsers, rule-based engines, putting together simple browser-based GUIs and whatever else the startup life would have me do.

2014-2015 — *Programmer* at Yandex (Moscow, Russia).

Officially a member of *Search Interfaces Development Infrastructure* group, but mostly I write backend tools that perform source to source compilation: parse, transform and generate code. Any given project will inevitably depend on: ometajs, esprima, estraverse, uglify-js, escodegen, xjst. If I'm lucky and do it right frontend developers get to use my work and get all the credit.

2009-2011 — *Equity Derivatives & Structured Products Sales* at Renaissance Capital (Moscow, Russia).
2007-2009 — *EM Structured Solutions and Derivatives Sales* at Barclays Capital (London, UK).

Projects

Racket — *Author* of ometa-racket, a mostly complete Racket implementation of OMeta - OO pattern-matching language that extends PEGs with ability to handle left-recursive rules and match structured data.

Author of skish, a mostly futile attempt at porting Olin Shivers' wonderful scsh to Racket. scsh is a non-interactive Unix shell embedded within Scheme (originally Scheme48).

Clojure — *Author* of several closed-source products: FpML message parser, financial derivatives classifier based on ISDA taxonomies, legal annotation tools, etc.

JavaScript — *Author* of bemhtml-syntax, a syntax converter for BEMHTML - an XSLT inspired templating language - part of BEM methodology of frontend development.

Author of bemhtml-source-convert, a *best effort* compiler from BEMHTML templates to BH templates.

Author of xjst-more, an XJST-based compiler for BEMHTML templates that facilitates incremental compilation of templates potentially on the Client. WIP.

Contributor to ometa-js, a JavaScript implementation of OMeta.

Contributor to bem-xjst, XJST-based compiler for BEMHTML templates.

Emacs Lisp — *Author* of jslime, a minor mode that sends JavaScript code to Node.js repl - silly little thing with very few features that does make iterative JavaScript development in Emacs sane.

Contributor to whatever little thing I find annoyingly broken in my daily Emacs use. Could be bug reports or tiny fixes to upstream packages.

Formal education

2004–2006 — Keldysh Institute of Applied Mathematics (Moscow, Russia)
PhD track in Applied Mathematics, dropped out

2004 — New Economic School (Moscow, Russia)
MS in Economics track with full scholarship, dropped out

1999-2004 — Lomonosov Moscow State University (Moscow, Russia)
MS in Theoretical Mechanics and Applied Mathematics.

Autodidacticisms

2012 — How to Design Programs by Matthias Felleisen et al.

How I was introduced to programming. Assorted solutions to HtDP.

2012 — Programming Languages, [Certificate]
Brown University

How I was introduced to creating PLs. Taught by Shriram Krishnamurthi based on his wonderful PLAI text. My solutions - a sequence of interpreters for progressively more complex languages: all the way to OOP, CPS transforms and type checkers.

2014 — Hardware/Software Interface, [Certificate 89.6%]
University of Washington for Coursera

How I was introduced to systems programming. Essentially an Introduction to Computer Systems course as taught at Carnegie Mellon with the same course-load and text Computer Systems: A Programmer's Perspective by Bryant and O'Hallaron.

2014 — Paradigms of Computer Programming 1, [Certificate1 94%]
Paradigms of Computer Programming 2, [Certificate2 97%]
Université catholique de Louvain for edX

How I was introduced to concurrency, multi-paradigm programming and delightful paradigms that so far seem to exist only in academic setting. Taught by Peter Van Roy and is based on his classical Concepts, Techniques, and Models of Computer Programming.

2015 — Introduction to Probability, [Certificate 94%]
MIT for edX

Because it's awesome.

2017 — Redex for designing operational semantics
The Racket Summer School of Semantics and Languages in Salt Lake City, Utah.

While targeted at PL PhDs a bunch of us non-academic types had been admitted. Learnt to create languages quickly and back them up with runnable reduction semantics - what's not to like?

Languages

Russian, English	Equally uncomfortable.
Clojure	What I get to use for my current projects.
Racket	Favorite Lisp. Would be my weapon of choice were such choice ever offered.
JavaScript	Wrote fair amount, mostly backend compiler stuff with Node.js.
OMeta	Extensive experience writing parsers and fairly complex grammars.
Emacs Lisp	Unavoidable Lisp.
Java	Enough to write a Clojure wrapper with necessary bindings.
C	Enough to pass a systems programming class but not nearly enough to actually use it.
Factor, OCaml, Lua	Toyed with but never used in earnest.

Activities and interests

Most of my activities and interests these days involve boxes with lights and buttons. Even so there were reports of me cycling, bouldering, surfing, roller-skating, skiing and more. Having owned a sports car I'll choose a bicycle every time.

Lived in the UK, US, Hungary, Spain and far more exotic places. Crossed the US from Mexico to Canada twice with the current state count of 19.

vladilen.kozin@gmail.com • +44 7494979 626 • London, UK
pdf version • txt version • doc version • html version