

Vlad Kozin

Clojure, ClojureScript, Racket, Redex, JavaScript, OMeta, meta-programming Fall '13 Recurse Center (aka Hacker School) alum	
Corporate ladder	
Since Apr 2017	<i>Programmer</i> at Droid (London, UK) Same as before but with obligatory daily commute.
2015-2017	<i>Programmer/Consultant</i> at Droid (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	<i>Programmer</i> at Yandex (Moscow, Russia). Officially a member of <i>Search Interfaces Development Infrastructure</i> 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: <i>ometa.js</i> , <i>esprima</i> , <i>estreevis</i> , <i>uglify.js</i> , <i>escodegen</i> , <i>xsst</i> . If I'm lucky and do it right frontend developers get to use my work and get all the credit.
2009-2011 2007-2009	<i>Equity Derivatives & Structured Products Sales</i> at Renaissance Capital (Moscow, Russia). <i>EM Structured Solutions and Derivatives Sales</i> at Barclays Capital (London, UK).
Projects	
Racket	<i>Author</i> of <i>ometa-racket</i> , a mostly complete Racket implementation of OMeta - OO pattern-matching language that extends PCPS with ability to handle left-recursive rules and match structured data. <i>Author</i> of <i>skia</i> , a mostly futile attempt at porting Olin Shivers' <i>wonderfultsch</i> to Racket. <i>sch</i> is a non-interactive Unix shell embedded within Scheme (originally Scheme48).
Clojure	<i>Author</i> of several closed-source products: <i>FpML</i> message parser, financial derivatives classifier based on ISDA taxonomies, legal annotation tools, etc.
JavaScript	<i>Author</i> of <i>benhtml-syntax</i> , a syntax converter for BEMHTML - an XSLT inspired templating language - part of BEM methodology of frontend development. <i>Author</i> of <i>benhtml-source-convert</i> , a best effort compiler from BEMHTML templates to BH templates. <i>Author</i> of <i>xsst-more</i> , an XJST-based compiler for BEMHTML templates that facilitates incremental compilation of templates potentially on the Client. WIP. <i>Contributor</i> to <i>ometa-js</i> , a JavaScript implementation of OMeta.
	<i>Contributor</i> to <i>tem-xsst</i> , XJST-based compiler for BEMHTML templates.
Emacs Lisp	<i>Author</i> of <i>jslime</i> , 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. <i>Contributor</i> 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) <i>PhD track in Applied Mathematics</i> , dropped out
1999-2004	Lomonosov Moscow State University (Moscow, Russia) <i>MS in Theoretical Mechanics and Applied Mathematics</i> .
Autodidacticisms	
2012	How to Design Programs by Matthias Felleisen et al.
2012	How I was introduced to programming. Assorted solutions to HDP. <i>Programming Languages</i> , [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 Courses 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 <i>Computer Systems: A Programmer's Perspective</i> by Bryant and O'Hallaron.
2014	<i>Paradigms of Computer Programming 1</i> , [Certificate 94%] <i>Paradigms of Computer Programming 2</i> , [Certificate 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 <i>Concepts, Techniques, and Models of Computer Programming</i> .
2015	<i>Introduction to Probability</i> , [Certificate 94%] MIT for edX Because it's awesome.
2017	<i>Redex</i> 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.	