# Vladilen Kozin

**Clojure(Script)**, **Racket**, **Emacs Lisp**, **Redex**, **JavaScript**, **OMeta**, **meta-programming**  
Fall’13 [Recurse Center](https://www.recurse.com/) (aka Hacker School) alum  
UK Tier 1 [Exceptional Talent visa](https://www.gov.uk/tier-1-exceptional-talent) holder

## Corporate ladder

Apr-Nov 2017

*Senior Programmer* at [Droit](http://droitfintech.com/) (London, UK)

Same as before but with obligatory daily commute.

2015-2017

*Programmer/Consultant* at [Droit](http://droitfintech.com/) (remote and New York, USA)

Building an expert system for compliant trading. Sneaking Clojure(Script) 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](https://company.yandex.com/) (Moscow, Russia).

Officially a member of *Search Interfaces Development Infrastructure* group, but mostly I wrote backend tools for source to source compilation - engines to write your template engines. If I were lucky and did it right frontend developers would get to use my work and take all the credit.

2009-2011

*Equity Derivatives & Structured Products Sales* at [Renaissance Capital](https://www.rencap.com/InvestmentBanking/About/) (Moscow, Russia).

2007-2009

*EM Structured Solutions and Derivatives Sales* at [Barclays Capital](http://investmentbank.barclays.com/markets.html) (London, UK).

## Projects

Emacs Lisp

*Author* of [multi.el](https://github.com/vkz/multi) - all things multiple dispatch for Emacs Lisp: type driven dispatch with *protocols*, ad-hoc polymorphism with *multi-methods*, pattern-matching and destructuring without noise with *multi-patterns*, case-dispatch with *multi-defuns*, benchmarking with *multi-benchmarks*.

Racket

*Author* of [tilda](https://github.com/vkz/tilda) an opinionated threading macro with self-documenting hole-markers, clause level keyword options and an implicit escape continuation.

*Author* of [prelude](https://github.com/vkz/prelude) that extends Racket with generic associative API, Lua-style meta-tables for prototypal inheritance and more.

*Author* of [ponzi](https://github.com/vkz/ponzi) - the beginnings of a clever Scheme for a discerning smart contract builder. WIP but it does implement the Ethereum Virtual Machine close enough to the Yellow Paper.

*Author* of [ometa-racket](https://github.com/vkz/ometa-racket), a mostly complete Racket implementation of [OMeta](http://en.wikipedia.org/wiki/OMeta) - OO pattern-matching language that extends PEGs with ability to handle left-recursive rules and match structured data.

*Author* of [skish](https://github.com/vkz/skish), a mostly futile attempt at porting Olin Shivers’ wonderful [scsh](http://scsh.net/) to Racket. scsh is a non-interactive Unix shell embedded within Scheme (originally Scheme48).

*Contributor* to Racket the language.

Clojure

*Author* of several closed-source products: FpML message parser, financial derivatives classifier based on ISDA taxonomies, legal annotation tools, PDF and XML content extractor and transformation tools.

*Author* of [bot](https://github.com/vkz/bot) - a crypto-currency arbitrager that could talk to several exchanges including Bitfinex and GDAX. It uses Clojure Spec to parse and validate protocol messages and [aleph](https://github.com/ztellman/aleph) for async communication and concurrency.

*Author* of [playrum](https://github.com/vkz/playrum) - just getting the taste for React in ClojureScript.

*Contributor* to [seqexp](https://github.com/cgrand/seqexp/pull/6), regular expressions for Clojure sequences.

JavaScript

*Author* of [bemhtml-syntax](https://github.com/vkz/bemhtml-syntax), a syntax converter for [BEMHTML](https://en.bem.info/) - an XSLT inspired templating language - part of [BEM methodology](https://en.bem.info/method/) of frontend development.

*Author* of [bemhtml-source-convert](https://github.com/vkz/bemhtml-source-convert), a *best effort* compiler from [BEMHTML](https://en.bem.info/) templates to [BH](https://github.com/bem/bh) templates.

*Author* of [xjst-more](https://github.com/vkz/xjst-more), an [XJST](https://github.com/veged/xjst)-based compiler for BEMHTML templates that facilitates incremental compilation of templates potentially on the Client. WIP.

*Contributor* to [ometa-js](https://github.com/veged/ometa-js), a JavaScript implementation of [OMeta](http://en.wikipedia.org/wiki/OMeta).

*Contributor* to [bem-xjst](https://github.com/bem/bem-xjst), XJST-based compiler for BEMHTML templates.

## Public Speaking

due Jul 2019

[talk](https://youtu.be/xSjk2PdQm5k?t=22405) at [RacketCon’19](https://con.racket-lang.org/#speakers) (Salt Lake City, USA)

due Sep 2019

talk accepted for [Strange Loop’19](https://www.thestrangeloop.com/2019/number-lang-wishful-thinking.html) (St. Louis, USA)

## Formal education

2004–2006

[Keldysh Institute of Applied Mathematics](http://keldysh.ru/index.en.shtml) (Moscow, Russia)  
*PhD track in Applied Mathematics, dropped out*

2004

[New Economic School](https://www.nes.ru/) (Moscow, Russia)  
*MS in Economics track with full scholarship, dropped out*

1999-2004

[Lomonosov Moscow State University](http://www.msu.ru/en/info/struct/depts/mechmath.html) (Moscow, Russia)  
*MS in Theoretical Mechanics and Applied Mathematics.*

## Autodidacticisms

2018

Language-oriented Programming and Language Building  
[The Racket Summer School 2018](https://summer-school.racket-lang.org/2018/) (Salt Lake City, USA)

2017

[Redex](https://redex.racket-lang.org/) for designing operational semantics  
[The Racket Summer School of Semantics and Languages](https://summer-school.racket-lang.org/2017/) (Salt Lake City, USA)

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?

2015

[Introduction to Probability](https://www.edx.org/course/introduction-probability-science-mitx-6-041x-0), [[Certificate](https://www.dropbox.com/s/egjo8b6ivigoqqj/Certificate%20-%20Intro%20to%20Probability%20%28MIT%20for%20EDX%29.pdf?dl=0) 94%]  
MIT for edX

Because it’s awesome.

2014

[Paradigms of Computer Programming 1](https://www.edx.org/course/paradigms-computer-programming-louvainx-louv1-1x-0), [[Certificate1](https://www.dropbox.com/s/043fwuco9fhbb09/Certificate%20-%20Paradigms%20of%20Computer%20Programming%20part1%20%28Louvain%20for%20EDX%29.pdf?dl=0) 94%]  
[Paradigms of Computer Programming 2](https://www.edx.org/course/paradigms-computer-programming-louvainx-louv1-2x-0), [[Certificate2](https://www.dropbox.com/s/awaogk8u5bsamqk/Certificate%20-%20Paradigms%20of%20Computer%20Programming%20part2%20%28Louvain%20for%20EDX%29.pdf?dl=0) 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](https://www.info.ucl.ac.be/~pvr/cvvanroy.html) and is based on his classical [Concepts, Techniques, and Models of Computer Programming](https://www.info.ucl.ac.be/~pvr/book).

2014

[Hardware/Software Interface](https://www.coursera.org/course/hwswinterface), [[Certificate](https://www.dropbox.com/s/ca393yfzxz9ymvi/Certificate%20-%20Hardware%20Software%20Interface%20%28Coursera%29.pdf?dl=0) 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](http://csapp.cs.cmu.edu/) by Bryant and O’Hallaron.

2012

[Programming Languages](http://cs.brown.edu/courses/cs173/2012/), [[Certificate](http://cs.brown.edu/courses/cs173/2012/OnLine/Certification/687898716/)]  
Brown University

How I was introduced to creating PLs. Taught by [Shriram Krishnamurthi](http://cs.brown.edu/~sk/) based on his wonderful [PLAI](http://cs.brown.edu/courses/cs173/2012/book/) text. [My solutions](https://github.com/vkz/PLAI) - a sequence of interpreters for progressively more complex languages: all the way to OOP, CPS transforms and type checkers.

2012

[How to Design Programs](http://www.ccs.neu.edu/home/matthias/HtDP2e/index.html) by Matthias Felleisen et al.

How I was introduced to programming. [Assorted solutions to HtDP](https://github.com/vkz/HtDP).

## Languages

Russian, English

Equally uncomfortable.

Clojure

What I get to use on the job. Can’t complain.

Racket

Favorite Lisp. Would be my weapon of choice were such choice ever offered.

Emacs Lisp

Unavoidable Lisp for any Emacs user. It is surprisingly fun to code.

JavaScript

Wrote fair amount, mostly backend compiler stuff with Node.js.

OMeta

Extensive experience writing parsers with complex and context dependent grammars.

Redex

Can implement executable semantics for your pet-language or DSL.

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, Rust, Shen

Toyed with but never used in earnest. I [ported](https://github.com/vkz/prelude/blob/master/tables.rkt) some good ideas from Lua to Racket and contributed a patch to [racer-rust](https://github.com/racer-rust/racer).

## 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](mailto:vladilen.kozin@gmail.com) • +44 7494979 626 • London, UK  
[pdf version](index.pdf) • [txt version](index.txt) • [doc version](index.docx) • [html version](index.html)