Vladilen Kozin

Clojure, **Racket**, **Emacs Lisp**, **Guile Scheme**, **TCL**, **Redex**, **OMeta**, etc
UK Tier 1 [Exceptional Talent visa](#) holder
UK [ILR](#) holder
Fall'13 [Recurse Center](#) alum
Former Yandex

I've worked on and implemented compilers, DSLs, rules and inference engines, parsers and parser combinator libraries, web-templating engines, data extraction and processing pipelines, backends for web-applications, React SPAs, IDEs and programmer productivity tools, formal semantics, blockchain DSLs and tooling, semi-autonomous and algorithmic trading and execution systems, OpenBanking data connectors and a gazillion other APIs. I built a [Funding Circle](#) competitor in under 5 months in a team of two. I've managed in-house server infrastructure and organized "homegrown" devops. Hired, trained and on-boarded engineers, successfully transitioned projects to new maintainers. I am a responsible and responsive professional who requires no micromanagement.

Corporate ladder

May 2023-now

> *Launched* https://git.ht (London, UK)
>
> Made tech blogging as simple as creating a Github gist. My [hoots](#) can be found at [vlad.git.ht](#).

Dec 2019-now

> *Director* and *CTO* at https://fullmeta.co.uk (London, UK)
>
> We are a Clojure SWAT team. Tightly knit, experienced, responsible and thoughtful. Hire one or a few of us to bring your vision to market. We take on contracts with startups and financial institutions. I can prototype, MVP, setup and get a project going, hire, on-board and hand off to your engineers when time comes. We can port and take on maintainance of existing products and take them to the next level. We communicate early and often, document our work and require little hand-holding from your senior staff.

Jul-Dec 2019

> *Senior Programmer* at [All Street Research](#) (London, UK)
>
> Building cognitive assistant for investment research in Clojure(Script). Front and back, AI, NLP, and more buzzwords here.

Apr-Nov 2017

> *Senior Programmer* at [Droit](#) (London, UK)
>
> Same as before but with obligatory daily commute.

2015-2017

> *Programmer/Consultant* at [Droit](#) (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](#) (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](#) (Moscow, Russia).

2007-2009
   EM Structured Solutions and Derivatives Sales at [Barclays Capital](#) (London, UK).

Sample Projects

Clojure

   Author of [fullmeta web](#) - Dynamic language deserves a dynamic web "framework": load www routes from Clojure namespaces "on the fly" - CGI style; render HTML and CSS. Utility-first local CSS vs selector-targeting is a stupid dichotomy - allow both! Other goodies: multi methods with per-position :default; helpful prelude functions, etc.

   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](#) - 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](#) for async communication and concurrency.

   Author of [playrum](#) - just getting the taste for React in ClojureScript.

   Contributor to [seqexp](#), regular expressions for Clojure sequences.

Emacs Lisp

   Author of [multi.el](#) - 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](#) an opinionated threading macro with self-documenting hole-markers, clause level keyword options and an implicit escape continuation.

   Author of [racket/tables](#) that extends Racket with first class Lua-style meta-tables for prototypal inheritance, generic associative API and more. Watch my [RacketCon'19](#) talk.

   Author of [FastCGI in Racket](#) that relies on my [racket/tables](#)

   Author of [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](#), 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).

Contributor to Racket the language.

Guile Scheme

I use GNU Guix OS for bit-for-bit reproducible packaging, containerized dev work and deployments. My entire OS and dev environments can be instantiated from the Scheme code I write.

Unless you're Google-scale I am positive I can do better than k8s and opaque Docker images with a dash of Scheme, Clojure and AWS API.

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.

Blockchain tech

I am most familiar with Chia (XCH) ecosystem and its own Chialisp smart-contract language. I have targeted EVM in the past but would prefer to stay away from Ethereum.

Public Speaking

Sep 2019

[talk](#) at [Strange Loop'19](#) (St. Louis, USA)

Jul 2019

[talk](#) at [RacketCon'19](#) (Salt Lake City, USA)

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

2018

Language-oriented Programming and Language Building
[The Racket Summer School 2018](#) (Salt Lake City, USA)

2017

[Redex](#) for designing operational semantics
[The Racket Summer School of Semantics and Languages](#) (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](#), [[Certificate](#) 94%]
MIT for edX

Because it's awesome.

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](#).

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.

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.

2012

[How to Design Programs](#) by Matthias Felleisen et al.

How I was introduced to programming. [Assorted solutions to HtDP](#).

Languages

Russian, English
What you'll hear me speak on Zoom calls.
Clojure
What I get to use on the job most often.
Racket
Used to be my favorite, but I tend to choose simpler tools now.

Emacs Lisp
    Unavoidable Lisp for a pro Emacs user. Surprisingly fun & productive.
Guile Scheme
    Unavoidable Scheme for a pro Guix user. Lively.
JavaScript
    Wrote fair amount, mostly backend Node.js. I prefer ClojureScript.
TCL
    Happy parallel universe with Shell scripts gone. Deserves more praise
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, Erlang (via LFE)
    Toyed with but never used in earnest. I ported some good ideas from Lua to Racket and contributed a patch to racer-rust. Would love to use Erlang via LFE professionally.

Infrastructure and Other tools

Managed and deployed OpenBSD and FreeBSD boxes but prefer Scheme code targeting GNU Guix. I programmed against Kafka, Elastic Search, Mongo, Postgres, MySQL, SQLite, Nginx, Jetty etc. Extensive hands on experience with on-prem Datomic and Datascript. Hands on and comfortable setting up, deploying and managing AWS resources (Ec2, Beanstalk, RDS, CloudWatch, etc) via console, cli, AWS API. I don't do YAML and k8s, nor should you - use Clojure AWS API directly and generate CloudFormation templates from EDN. I keep it simple.

On interviews

NB UK/EU recruiters: I am expensive

NB recruiters: liaise this section with your client

I hired and put together successful engineering teams from nothing. Interviewing is hard - I get it. That doesn't make throwing random puzzles at me a suitable interviewing technique, unless the job specifically calls for "tip of your fingers" algorithmic knowledge rather than implementing an algorithm Wikipedia claims best. I encourage you to look at my code and pair program a feature or debug something in one of my current projects. I'll happily give you a choice of interesting problems we can work on. You'll get to see me program computers and I'll enjoy a feature or bugfix. Take home problems are fine, but must be paid for at my usual contracting rate.

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.