Introducing myself…

- Nenad aka "DocKimbel" Rakocevic, 🇫🇷

- Programming for 25 years: C/C++, *Basic, ASM, REBOL, web client-side languages,…

- Founder of a software company in Paris: Softinnov

- Author of several libraries for REBOL:
  - MySQL, PostgreSQL, LDAP native drivers
  - UniServe: asynchronous, event-driven network engine
  - Cheyenne Web Server: full-featured web application server
  - CureCode: very fast web-based bug tracker (Mantis-like)
  - Various others tools, game, demos…
  - Was a happy Amiga user and registered BeOS developer
Why am I using REBOL for 11 years?

- Great scripting language
- Great prototyping tool
- Simple cross-platform graphic engine (View)
- Symbolic & Meta-programming
- Code / Data duality
- DSL-oriented
- Great designer behind: Carl Sassenrath
Why I don't want to use REBOL anymore?

- Closed source
- Slow (benchmark)
- No multithreading support
- Mostly glue language, not general-purpose enough
- Not (easily) embeddable in third-party apps
- Can't run on popular VM (JVM, CLR)
- Sometimes designed for "Bob the artist", rather than "John the programmer"
What is the state of REBOL world? (1/2)

How REBOL began 14 years ago…
What is the state of REBOL world? (2/2)

...and where it is today
What to do then?

- Give up and pick up another language?
- Build an alternative?

I chose the 2nd option!
My answer is: **Red!**

- Red[uced] REBOL dialect
- Fully open source (MIT/BSD)
- Statically compiled + JIT compiled
- Parallel programming support
- General purpose (system programming support)
- Can be used for scripting like REBOL (REPL console)
- Easily embeddable in other apps (think Lua)
- Built-in small & scalable web server
- Work in progress…started 3 months ago, but thinking about it for years!
Red Language Features Tour

- Syntax: strongly inspired by REBOL
- Semantic rules: most of REBOL
- Type system
  - rich, most of REBOL types
  - new types as pluggable modules (literal form accessible)
  - type inference, when possible
  - types mismatches caught at compile-time instead of runtime
- First-class functions and HOF support
- Meta-programming support (JIT-compiled code)
REBOL features not supported by Red

- Too "abstract" code
  - Foo: func [ a ][ a/b/c ] => "a" can be object!, function!, block!, ...

- Dynamic word binding
  - REBOL: can change the scope of a word! value dynamically
  - Red v1.0: static scoping only
  - REBOL-like word binding semantics could be added later at a higher level in Red
Red Architecture Overview

- **User Code**
  - User Scripts
  - User Applications
  - 3rd-party Libraries

- **Red Language Libraries**
  - Standard Library

- **Runtime**
  - Lexical Parser
  - Memory Manager

- **Executable Container**
  - Executable Image

- **Operating System**

- **Compilers**
  - Red (JIT) Compiler
  - Red Compiler
  - Red/System Compiler
  - Red/System Linker

- **Coded in Red Language**
- **Coded in RedSystem Dialect**
Red Memory Model

- Thread-local memory allocation
  - Arrays of 128-bit cells
- Possibility for shared immutable data structures
- Garbage collector
  - Compacting collector
  - Stop-the-thread GC model for v1.0
  - Incremental GC in v2
Red/System Language

- Purely imperative, C-level language, with a Red syntax
- Statically compiled (naïve compilation for now)
- Limited type system:
  - integer, struct, pointer, string (no 1st class functions)
  - No type inference
- Inlined ASM support
- Linker
  - Output types: Exe, DLL, Lib
  - Formats: PE, ELF, mach-o
- Targets: IA-32, ARM, x64, JVM, CLR
- Red/System as an inlined dialect in Red
Red Concurrent & Parallel programming

- "PPP challenge" (Intel)
  - We now live now in a multi-core CPU world
  - Window of opportunity for new solutions / languages
- Task parallelism
  - Execute several threads of code on multiple Cores at the same time
  - Red will provide an Actor-like abstraction
- Data parallelism
  - Process a data structure with several Cores at the same time
  - Red will provide a parallel series abstraction
Bootstrapping Red (chicken & egg problem)

1) Write Red/System compiler in REBOL  
2) Write Red linker in REBOL  
3) Write Red runtime in Red/System  
4) Write Red static compiler in REBOL  
5) Write Red standard library in Red  
6) Rewrite Red/System compiler in Red  
7) Rewrite Red static compiler in Red  
8) Write Red JIT-compiler in Red  
9) If still alive, take some good rest! 😊
Red IDE

- Mandatory for most programmers
- Code edition: Scintilla component
- Strong focus on debugging capabilities
  - step-by-step Red code debugging
  - step-by-step Parse rules debugger
  - I/O data streams capturing for inspection

- Code Profiler
- GUI in Red with an OS abstraction layer (SWT-like)
- Code bubbles support (v2)
Red Key Success Factors

- **Time to market**
  - As short as possible
  - Short iterations (no "tunnel" during months)
  - Critical for success

- **Community: reach a critical mass**
  - Keep community informed (web sites, blog, twitter,...)
  - Ease user contributions (github)
  - Be open (avoid "ivory tower" syndrom)
  - Goal: reach critical mass (get enough contributors)
Roadmap

- Sept. 2011:
  - beta of Red (no JIT)
  - alpha of ARM support
  - alpha of the IDE

- Dec. 2011:
  - v1.0 of Red (no JIT)
  - beta of the IDE

- Q1 2012:
  - beta of Red JIT-compiler
  - v1.0 of IDE
If you think this is not doable…watch me!

On Red's blog: http://red-lang.org

On Red's twitter channel: #red_lang

...see you next year!