Erlang and Node.JS
or, better,

Criteria for evaluating technology

Lev Walkin, CTO Echo
@levwalkin

Task

- Library coverage
- Community size and quality
- Traditional tasks solved by community

Domain

- Languages have objective differences
  - Math → R, Matlab
  - Strings → Perl
  - Syntax trees → ML, LISP
  - Client-side programming: VBScript

Language

- Power of abstraction
- Support for useful practices
- Discouragement of negative practices (global variables)

Barriers to entry

- Simplicity as in "number of moving parts"
- Simplicity as in "simple made easy, hard things possible"
- Simplicity as in "number of abstractions to the bottom"
- Simplicity as in "reusable prior knowledge"

Code evolution

- Support for growing codebase
- Coverage, testing, refactoring
- Testability (= light coupling)
- QuickCheck / PropEr

Running the service

- Introspection
- Code deployment
- Hot upgrade
- Running in presense of failures

Hiring arguments

- Ability to find skilled people...
- Yet, **diversity** is an overlooked argument

Programming in groups

- Unclear dependencies
- Non-formalized conventions
- Code does not fit a single brain
- Understanding peer's code
- Involuntary data corruption
- Code review

Ecosystem maturity

- Mature system to support ever-changing requirements?

Enjoyment (1)

- Streamlined learning process (physiology)
- Speeds up habit formation
- Huge support for intrinsic motivation

Enjoyment (2)

- Objective advantage can turn into objective deficiency... enjoyment helps to pull through.

Enjoyment (3)

- Can be instilled and modulated!
- Enjoyment is contagious!
- Helps form communities

Erlang

- Erlang was not designed the web
- yet, web server side is a system for serving ton of people (same approximate domain)
- Supports good practices
- Runs in presense of programmer and environment errors
- Great introspection and debugging

JavaScript (1)

- JavaScript !designed for the modern web
- Somewhat indifferent to good practices (global vars, namespaces, memory isolation)
- Lots of folklore used to keep code neat
- Great frameworks for its primary tasks and domain

JavaScript (2)

- The most important component of the modern Web ecosystem
- LOTS of useful complexity on the client
- Lots of people know it (really?)
- Hard to find and train JavaScript programmers... but easier to find than Erlang ones (See Hiring argument)

Node

- Fast VM: beats Erlang at many benchmarks
- Not very stable yet, but keeps getting better
- JavaScript lures client-side folks by promising code reuse
- Async-IO technique requires next slide

IO story

- Linear flow (open(); do(); close();)
- State machines (switch(STEP2) { ... })
- Coroutines = Packaging FSM
- Green threads → back to Linear
- Linear is GOOD!

Node.JS

- Linear flow is a desired property!
- New languages and pre-processors provide resemblance of linear flow, therefore no code reuse
- Frameworks mask necessity to do IO, yet still...

Node.JS

- No significant code reuse.
- People do not move to Node.JS from client side (shortage of such people, anyway).
- People move to Node.JS because they **think** there are many client side people who will support the ecosystem.

Node.JS

- Node.JS is a platform in its own right!
- Ruby and Python and Java programmers chose Node.JS
- Node.JS introduces frameworks for the next web (real time, Socket.IO, NowJS)
- People are EXCITED about it.

Erlang and Node.JS

- Clear new-ish niche for Erlang: the real time, always on web.
- Node.JS has increasing footprint in this space
- **Technical merits may not count when people enjoy stuff**

Enjoyment with Node

- Programmers moving from LAMP to Node enjoy low barriers to entry
- Programmers moving from LAMP enjoy web (and real-time web)
- Excited people develop the ecosystem and overcome technical hurdles

Enjoyment with Erlang

- Erlang folks enjoy language, not so much a domain (web)
- Erlang folks see Node.JS as a competitor
- Erlang community is at risk of losing the good domain

Answers!

@levwalkin

lionet @ LJ