Clojure is Becoming Popular

- Popular*
- Functional
- Lisp
- Not terribly novel
How Popular?

- 8000+ member mailing list
  82,000+ messages
- 700+ on IRC
- #2 (behind Scala) non-Java JVM server lang
- Many companies use in production
  Startups to Fortune 50
- > Dozen books, several 2nd editions
- 3 Clojure conferences per year
<table>
<thead>
<tr>
<th>Rank</th>
<th>Lang</th>
<th>New repos</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Scala</td>
<td>6918</td>
</tr>
<tr>
<td>16</td>
<td>Go</td>
<td>6884</td>
</tr>
<tr>
<td>17</td>
<td>Prolog</td>
<td>5829</td>
</tr>
<tr>
<td>18</td>
<td>Clojure</td>
<td>4904</td>
</tr>
<tr>
<td>19</td>
<td>Haskell</td>
<td>4681</td>
</tr>
<tr>
<td>20</td>
<td>Lua</td>
<td>4048</td>
</tr>
</tbody>
</table>
Context

- JVM
- PHP, Python, Perl, Ruby
  broke C-lineage stranglehold
  demonstrated dynamic lang agility
- Paul Graham’s essays
- Crushing complexity of stateful OO
- JSON, XML
Right Place, Right Time?

- In fall 2007 when announced, no one had heard of me nor Clojure
- 18,000 hits the first day
Survey 2012 - Growth

1,372 respondents

From Where?

Java: 30
Ruby: 10
Python: 15
C#: 5
Erlang/Scala/Haskell: 2
Other: 20

0 10 20 30 40
Why?
Practical

• Target - use it wherever you could use Java both reach and performance
• Tool, for professional programmers hobbyists welcome
• General purpose programming
Small

- A basic Lisp evaluation model
- No continuations, condition systems etc
Simple

- Few, orthogonal axes
- Functions, data, state model, evaluation model, macros etc
FP Benefits, Distilled

• Pure functions + immutable data
• Made practical and idiomatic
• Efficient persistent data structures the default
• Functional core library
• No mutable local variables
• Clearly separate state constructs
Opinionated

- Fewer choices
- Set of decisions made and encoded
- Everything works together
Eschews Types for Information

- Simple data - lists, vectors, maps, sets
- Associative collections in lieu of objects
- First-class names (keywords)
Minimal Type-Specific Code

• just generic map-manipulation code
• therefore, much less code
• more reuse
• better interop and libs
• facilitates systems building
• less compiler help
Runtime Polymorphic

- People coming from OO understand and expect this
- Abstractions for everything (even invocation)
- Polymorphism constructs are ‘open’
Concurrency?

- Not so much
- Part of initial story
- Has to be there
  - but rarely needed
Hosted

- Dominant hosts (JVM, JS)
- Great interop
- Libraries out of the gate
- No migration pain
- Easy to sneak in

“just another library”
Lisp

- Still has appeal
- Small core, rest is libs
- Runtime redef, code loading, macros
- Code-as-data, read/print, REPL
- but enhancements matter
  fp, collections, syntax, abstractions
Permissive

- “Consenting adults” language
- Support the right thing
  - vs disallow the wrong
Support and Evangelism

- 3 years x 10 hours/day
  Mailing list and IRC
- Lots of talks and travel
Stability

- All programs from first book (2009) still work
- Always a high priority
The Blub Paradox?

• People don’t know what they are missing?
• They know what they are getting
  • Simpler and more robust programs
  • High productivity
  • Resulting systems work, perform well, and are flexible
• Clojure is an effective tool
Gateway Drug?

If Clojure disappeared, then what?

Scala 30
Haskell 30
Python 25
Common Lisp 20
Ruby 15
Java 10
Scheme 10
Erlang 0

“too depressing to contemplate”
Thanks!