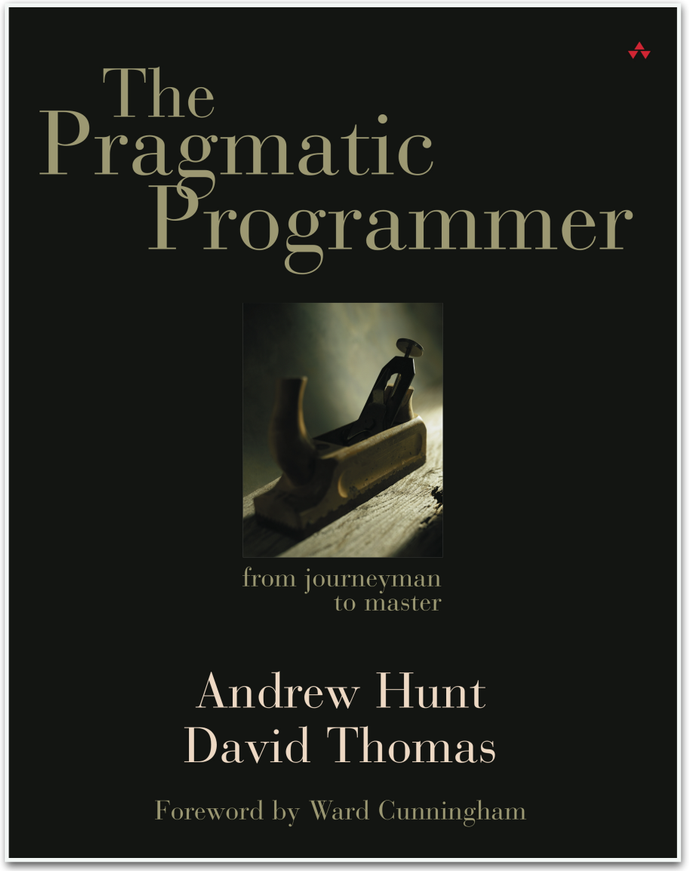
(why? clojure)

విజయ్ కిరణ్ (Vijay Kiran)
Lunatech



"Learn at least one new language every year"

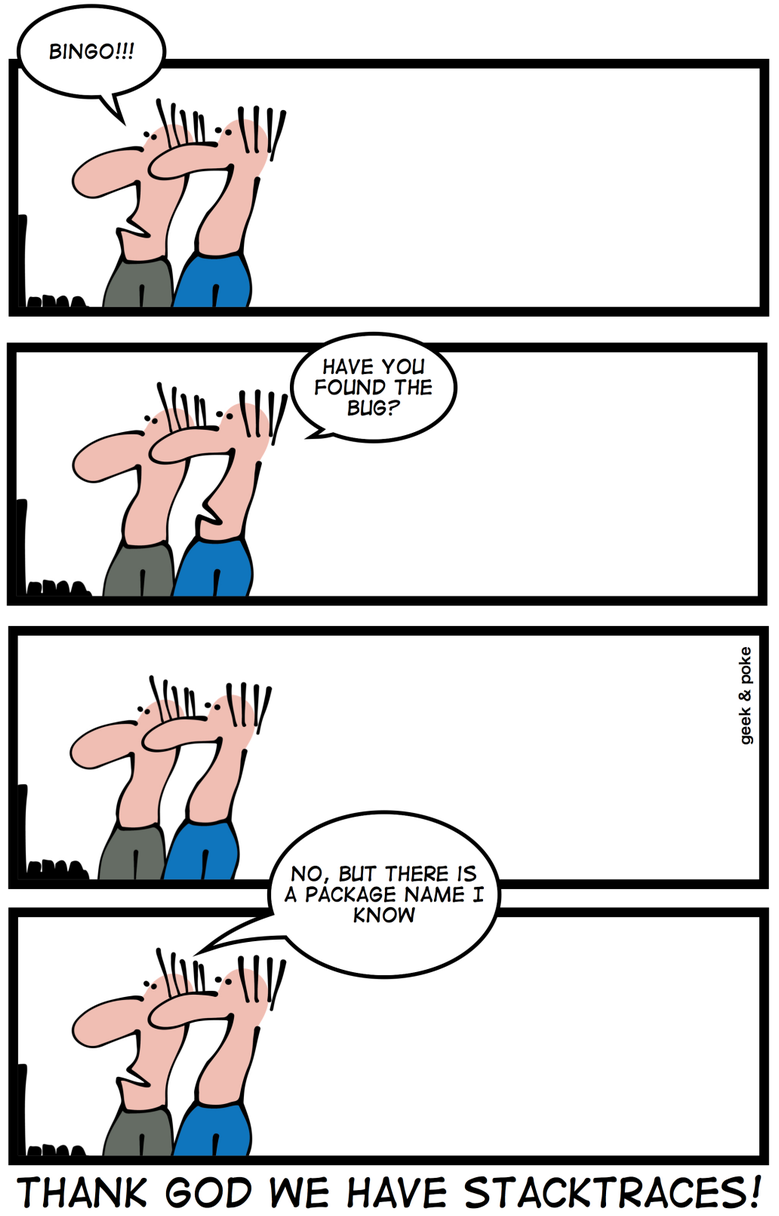
1994	BASIC
1998	C, Pascal
2002	Visual C++, Java
2003	Java, Perl
2005	Java, Ruby
2010	Java, Clojure
2015	Clojure, Scala

EN - NL - FR

Telenet homespot

Wi-Free

Error executing macro: displayForm required parameter: roamingPartner is not specified. The problematic instruction: ---------- ==> macro displayForm [on line 106, column 1 in macro.ftl] in user-directive displayForm [on line 71, column 33 in login.ftl] ---------- Java backtrace for programmers: ---------- freemarker.template.TemplateException: Error executing macro: displayForm required parameter: roamingPartner is not specified. at freemarker.core.Macro$Context.sanityCheck(Macro.java:211) at freemarker.core.Macro$Context.runMacro(Macro.java:169) at freemarker.core.Environment.visit(Environment.java:614) at freemarker.core.UnifiedCall.accept(UnifiedCall.java:106) at freemarker.core.Environment.visit(Environment.java:221) at freemarker.core.MixedContent.accept(MixedContent.java:92) at freemarker.core.Environment.visit(Environment.java:221) at freemarker.core.Environment.process(Environment.java:199) at freemarker.template.Template.process(Template.java:237) at org.springframework.web.servlet.view.freemarker.FreeMarkerView.processTemplate(FreeMarkerView.java:366) at org.springframework.web.servlet.view.freemarker.FreeMarkerView.doRender(FreeMarkerView.java:283) at org.springframework.web.servlet.view.freemarker.FreeMarkerView.renderMergedTemplateModel(FreeMarkerView.java:233) at org.springframework.web.servlet.view.AbstractTemplateView.renderMergedOutputModel(AbstractTemplateView.java:167) at org.springframework.web.servlet.view.AbstractView.render(AbstractView.java:263) at org.springframework.web.servlet.DispatcherServlet.render(DispatcherServlet.java:1208) at org.springframework.web.servlet.DispatcherServlet.processDispatchResult(DispatcherServlet.java:992) at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:939) at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:856) at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:953) at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:844) at javax.servlet.http.HttpServlet.service(HttpServlet.java:734) at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:829) at javax.servlet.http.HttpServlet.service(HttpServlet.java:847) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:329) at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:248) at com.codahale.metrics.servlet.AbstractInstrumentedFilter.doFilter(AbstractInstrumentedFilter.java:97) at be.telenet.mijntelenet.status.api.metrics.MetricsServletFilter.doFilter(MetricsServletFilter.java:38) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:280) at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:248) at be.telenet.esp.homespot.portal.web.filter.SetHeadersFilter.doFilterInternal(SetHeadersFilter.java:27) at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:107) at org.springframework.web.filter.DelegatingFilterProxy.invokeDelegate(DelegatingFilterProxy.java:343) at org.springframework.web.filter.DelegatingFilterProxy.doFilter(DelegatingFilterProxy.java:260) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:280) at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:248) at org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter.java:88) at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:107) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:280) at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:248) at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:275) at




JRuby
Scala
Ioke
Xtend
Groovy
Jython
FANTOM

Clojure

Dynamic

Functional & Immutable

Concurrency Support

Runs on JVM/JavaScript & CLR

Clojure

Flexible

Concise

Experimentation/Exploration

Focus on Business Problem

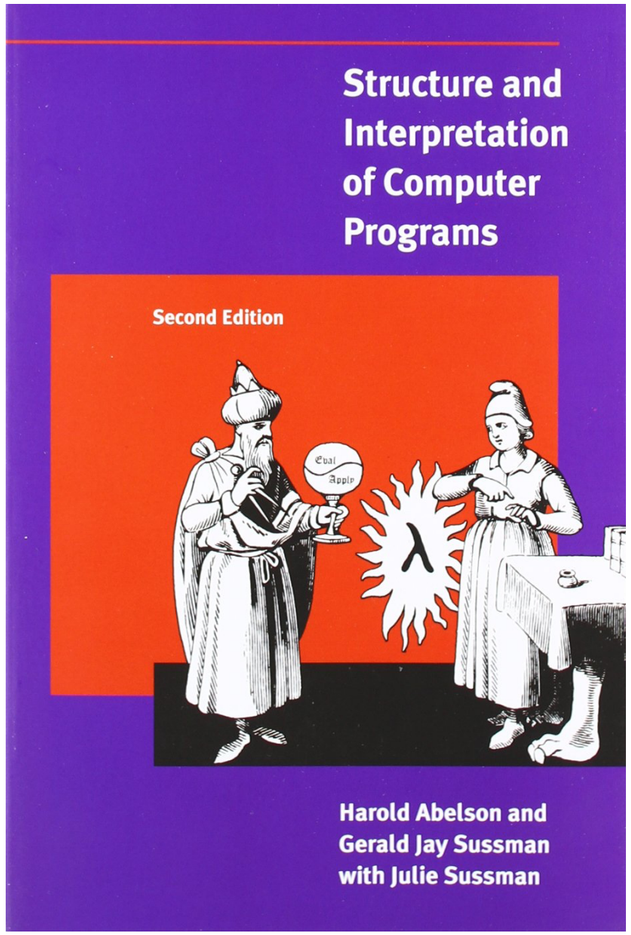
Clojure

- REPL: Read, Eval, Print, Loop
  - On-the-fly function definition
  - Interactive Development

Clojure is Simple & Elegant

Every powerful language has

- **Primitive Expressions**
- **Means of Combination**
- **Means of Abstraction**



- Integers, Doubles, Big Decimals & Ratios
  - **12345678987654, 7.8892, 1.2345678M & 5/3**
- Strings & Characters
  - **“Clojure” & \d \e \v \o \x \x**
- Symbols & Keywords
  - **hello & :hello**
- Booleans & Null
  - **true, false & nil**
- Regular Expression Patterns
  - **#“dev*x”**

- Lists – Singly Linked Lists, Prepend-friendly
  - **(list 1 “two” 3), (:hello “world”), (1 2 3 4 5)**
- Vectors – Indexed, Append-friendly
  - **[1 2 3], [“hello” 22/7 “world”]**
- Maps – Key/Val Pairs
  - **{:hello “Hallo”, :world “Wereld”} , {1 “one”}**
- Sets
  - **#{“devoxx.be”, “devoxx.uk”, “devoxx.pl”}**

```
1 (hello [names] (“hi” “everyone”))
2 [1 (2 3) [10 20 30]]
3 {:planets [“earth”, “mars”, “pluto”]}
```

All Data Structures are

Immutable and Persistent

Clojure has No Syntax*

*Almost :)

Data Structures ***are*** code

Code is a Data Structure

```
1 (ns helloworld)
2 (defn say-hello [event] (prn “Hello” event))

  => (say-hello “Devoxx”)
  Hello Devoxx
  nil
```

- Data Structures are the code
- Everything is just a ***List***
  - Declarations
  - Control Structures
  - Functions
  - Operators

(*operation* …)

- function expression
- macro (code that “writes” code)
- a few special forms

def, do, ., fn, if, let, loop, new, quote, recur,
set!, throw, try, var

stackoverflow

Questions

What are all the uses of an underscore in Scala?

Existential types

```
def foo(l: List[Option[_]]) = ...
```

Higher kinded type parameters

```
case class A[K[_],T](a: K[T])
```

Ignored variables

```
val _ = 5
```

Ignored parameters

```
List(1, 2, 3) foreach { _ => println("Hi") }
```

Wildcard patterns

```
Some(5) match { case Some(_) => println("Yes") }
```

Wildcard imports

```
import java.util._
```

Hiding imports

```
import java.util.{ArrayList => _, _}
```

Joining letters to punctuation

```
def bang_!(x: Int) = 5
```

Assignment operators

```
def foo_=(x: Int) { ... }
```

Placeholder syntax

```
List(1, 2, 3) map (_ + 2)
```

Partially applied functions

```
List(1, 2, 3) foreach println _
```

There may be others I have forgotten!

DEVOXX™

Clojure is functional

- easier to reason about
- easier to test
- immutability baked-in
- essential for concurrency

- functions are first-class
  - higher-order functions
- recursion
  - loop/recur
  - trampoline

Clojure has great support for concurrency

- No Locks
- Primitives
  - vars, atoms, agents
- Software Transaction Memory
  - MVCC based
  - refs

	Co-ordinated	*Asynchronous*	*Retriable*	*Thread-Local*
Var				✔
Atom			✔	
Agent		✔		
Ref	✔		✔	

core.async

- channel based concurrency
- first-class channels
- just a library
- also, works in JavaScript

Clojure is practical

Host (JVM) Friendly

- Strings are Java Strings
- Numbers are Java Numbers
- Collections implement **Collection**
- No ***conversions*** or ***asJava*** needed
- Functions implement ***Runnable***
- Using Java libraries is easy

ClojureScript

- Brings Clojure to JavaScript
- Great tools
  - Figwheel - live coding
  - react.js integration (OM, Reagent …)
- Macros!

Libraries over Frameworks

14199 Libraries

Simple build tools

Leiningen

boot

Editors & IDE Plugins

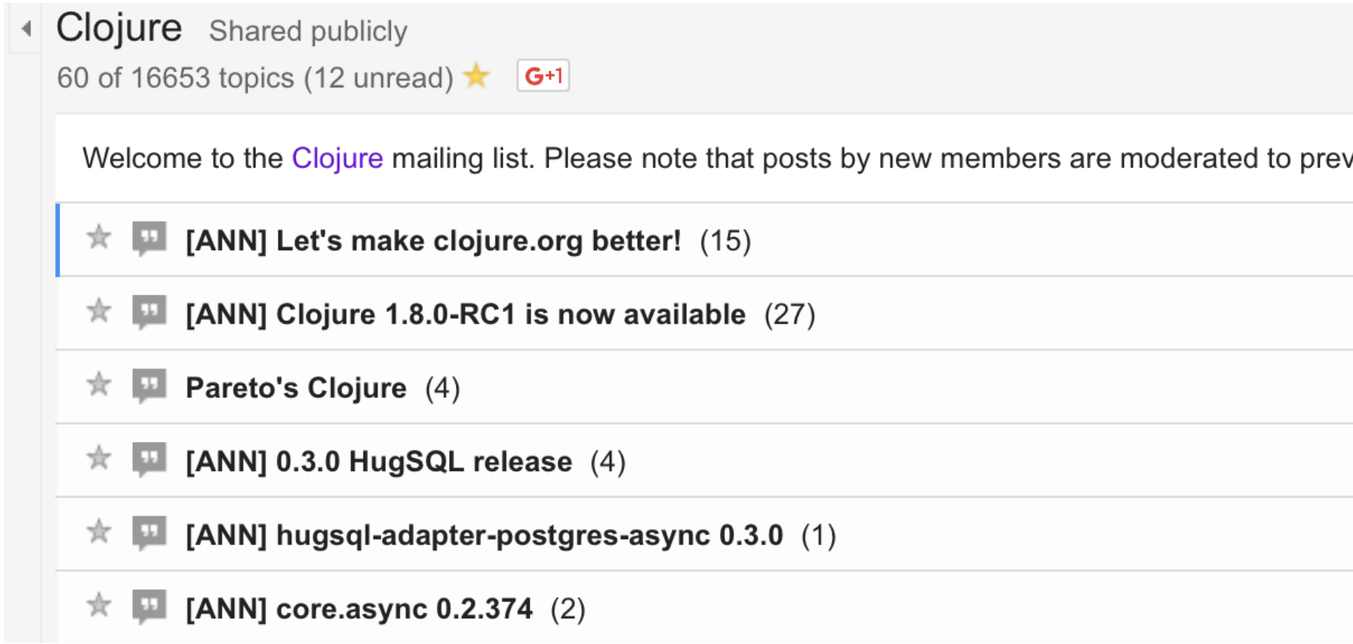
(cider[])

IntelliJ IDEA

Cursive

Clojure has
vibrant & friendly community

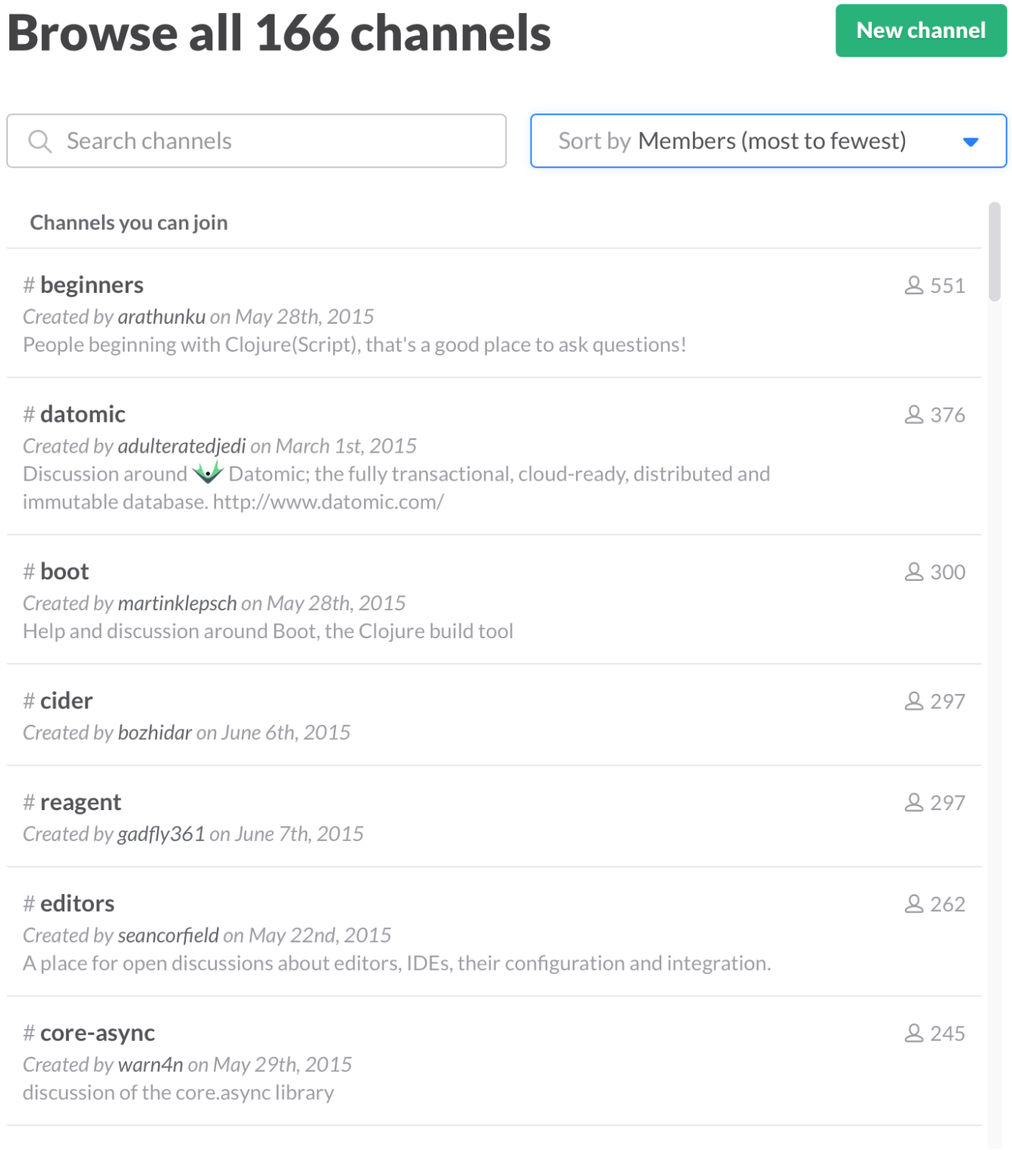
10569
Members



~ 3500 members

clojurians.slack.com

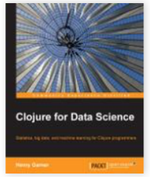
irc.freenode.net#clojure



Clojure Recipes
by Julian Gamble

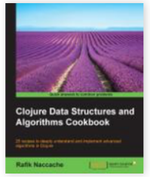
Clojure High Performance Programming - Second Edition
by Shantanu Kumar

Clojure for the Brave and True
by Daniel Higginbotham



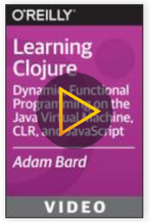
Clojure for Data Science
by Henry Garner

Clojure Applied
by Ben Vandgrift,Alex Miller



Clojure Data Structures and Algorithms Cookbook
by Rafik Naccache

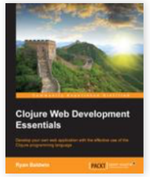
Building Microservices with Clojure
by Scott Rehorn



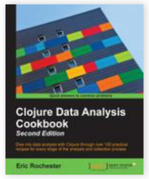
Learning Clojure
by Adam Bard

Living Clojure
by Carin Meier

Clojure Reactive Programming
by Leonardo Borges



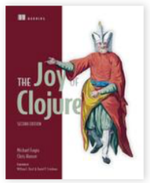
Clojure Web Development Essentials
by Ryan Baldwin



Clojure Data Analysis Cookbook - Second Edition
by Eric Rochester

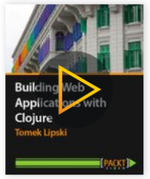
Building Web Applications in Clojure
by Ryan Neufeld

Mastering Clojure Macros
by Colin Jones



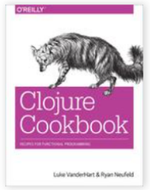
The Joy of Clojure, Second Edition
by Michael Fogus and Chris Houser

Mastering Clojure Data Analysis
by Eric Rochester



Building Web Applications with Clojure
by Tomek Lipski

Clojure for Machine Learning
by Akhil Wali

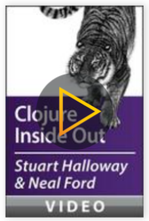


Clojure Cookbook
by Luke VanderHart,Ryan Neufeld

Web Development with Clojure
by Dmitri Sotnikov

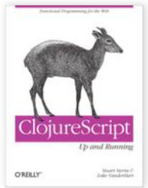
Clojure for Domain-specific Languages
by Kelker Ryan

Clojure High Performance Programming
by Shantanu Kumar



Clojure Inside Out
by Neal Ford,Stuart Halloway

Clojure Data Analysis Cookbook
by Eric Rochester

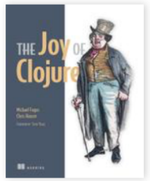


ClojureScript: Up and Running
by Stuart Sierra,Luke VanderHart

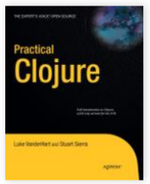
Clojure Programming
by Chas Emerick,Christophe Grand,Brian Carper

Programming Clojure, 2nd Edition
by Stuart Halloway,Aaron Bedra

Clojure in Action
by Amit Rathore



The Joy of Clojure
by Chris Houser,Michael Fogus



Practical Clojure
by Stuart Sierra,Luke VanderHart

DEVOXX™

#Devoxx #WhyClojure @vijaykiran

Why Clojure?

- is Simple & Elegant
- is functional
- has great support for concurrency
- is practical
- has vibrant & friendly community

Why *not* Clojure?

	Impure	*Pure*	*Lazy*	*Strict*	*Static*	*Dynamic*
Clojure	😄		😄			😄
Frege		😄	😄		😄	
Scala	😱	😱	😱	😱	😱	😱

DEVOXX™

Related Talks

Joe Kutner 17:50 - 18:50 Room 7	**4 JVM Web Frameworks in 40 Minutes**
Lutz Huehnken 17:50 - 18:50 Room 6	**New Concurrency Models on the JVM: Fibres, Verticles, Channels and Actors.**

DEVOXX™

Thanks for Listening!

Questions?