# Clojure: Designed for Performance

Tom Marble

January 30, 2013

## Contents

**1 Clojure: Designed for Performance** **3**
1.1 Overview: Topics to cover . . . 3

**2 Background** **3**
2.1 About Tom . . . 3
2.1.1 Sun: Java Performance Team . . . 4
2.1.2 Sun: OpenJDK . . . 5
2.1.3 Consulting . . . 6

**3 Why Common Lisp?** **6**
3.1 Avantages of Common Lisp . . . 7
3.1.1 homoiconic . . . 7
3.1.2 macros: code transformations at compile time . . . 7
3.1.3 Great for Domain Specific Languages . . . 8
3.1.4 REPL . . . 8
3.1.5 Lisp successes . . . 8
3.2 Disadvantages of Common Lisp . . . 9
3.2.1 There is a standard, but no compatibility test kit (as for Java) . . . 9
3.2.2 Having a standard is no guarentee of compatibility . . . 10
3.2.3 The "library" problem . . . 10
3.2.4 The porting problem . . . 10
3.2.5 The concurrency problem . . . 11

**4 Why Java?** **11**
4.1 Avantages of Java . . . 11
4.1.1 Cross Plaform . . . 11
4.1.2 Rich set of libraries . . . 11

4.1.3 Enterprise adoption . . . 11
4.1.4 Performance: Dynamic code optimization . . . 12
4.1.5 Performance: Garbage Collection . . . 12
4.2 Disadvantages of Java . . . 12
4.2.1 Mutation is (almost) required . . . 12
4.2.2 Single inheritence hierarchy . . . 13
4.2.3 Complex . . . 13
4.2.4 Java EE Containers . . . 13
4.2.5 Java EE APIs in the Web Container . . . 13
4.2.6 Java EE APIs in the EJB Container . . . 13
4.2.7 Java EE APIs in the Application Client Container . . . 13

**5 Why Clojure? 13**
5.1 Advantages of Clojure . . . 13
5.1.1 Easy interoperation with Java . . . 13
5.1.2 Leverages advantages of a Lisp . . . 14
5.1.3 Multimethods . . . 14
5.1.4 Lazy sequences . . . 14
5.1.5 Functional Programming . . . 14
5.1.6 Software Transactional Memory . . . 15
5.1.7 No spec, one implementation . . . 15
5.2 Disadvantages of Clojure . . . 16
5.2.1 The state of Clojure Contrib (is a challenge) . . . 16

**6 Concurrent Programming 16**
6.1 Threads . . . 16
6.2 Threading harness (for examples) . . . 17
6.3 Types of operations . . . 17
6.4 Refs . . . 17
6.5 Atoms . . . 17
6.6 Agents . . . 18

**7 Promising Future 18**
7.1 Java . . . 18
7.1.1 ARM looks very good for size, cost, heat . . . 18
7.1.2 We are seeing ARM everywhere in embedded devices . . . 19
7.1.3 Java as assembly language . . . 19
7.2 Bleeding Edge OpenJDK features . . . 20
7.2.1 Fork/Join . . . 20
7.2.2 Tail Call Optimization . . . 20

7.2.3 Invoke Dynamic . . . . . 20
7.2.4 Modularization (Jigsaw) . . . . . 21

**8 Conclusion** **21**

**9 Q/A** **21**

**10 Extra** **22**
10.1 What is that presentation tool? . . . . . 22
10.1.1 Note's on what I put in my .emacs.d/custom.el . . . . 24
10.2 The Tools I am using . . . . . 24
10.2.1 Maven . . . . . 24
10.2.2 Leiningen . . . . . 25
10.2.3 Jenkins . . . . . 25
10.2.4 Trac . . . . . 26

# 1 Clojure: Designed for Performance

file:Clojure-glyph.svg

## 1.1 Overview: Topics to cover

*...overview...*

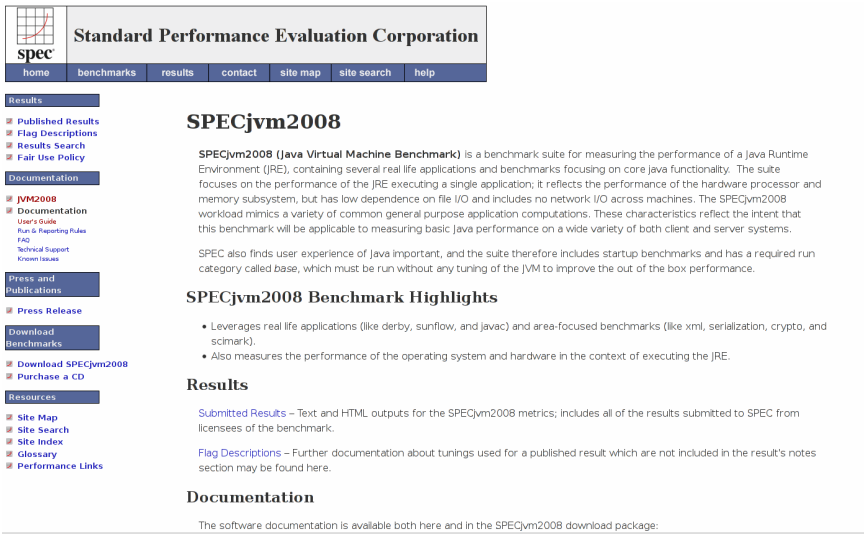
# 2 Background

## 2.1 About Tom

tmarble

### 2.1.1 Sun: Java Performance Team

Continuous *Performance* Integration for the JVM




### 2.1.2 Sun: OpenJDK

Part of the Java open source Core Strategy Team at Sun, first OpenJDK Ambassador (I went to a lot of conferences)

- FOSDEM
- FISL
- OSCON

ApacheCon 2006: Sun unBOF/Party

/Copyright 2006 Ted Leung: https://secure.flickr.com/photos/twleung/268116213/

### 2.1.3 Consulting

Cybersecurity
: Performance Analysis and Benchmarking
: Clojure
: Very interested in open, embedded systems

## 3 Why Common Lisp?

John McCarthy was old school: developed LISP in 1954

## 3.1 Avantages of Common Lisp

### 3.1.1 homoiconic

**code is data**

List

(def mylist (list 1 2 3))

Function

(def myadd (fn [a b] (+ a b)))

A lisp is defined in terms of the evaluation of data structures and not in terms of the syntax of files.

### 3.1.2 macros: code transformations at compile time

Macros offer hooks for syntactic abstraction and there is very little syntax.

(defmacro and ([] true) ([x] x) ([x & rest] `(let [and# ~x] (if and# (and ~@rest) and#))))

Allows code transformation **before** the reader does evaluation. In Clojure **defn** is a macro that makes defining functions a little simpler.

Code walkers are easy to write.

### 3.1.3 Great for Domain Specific Languages

LISP is the language of choice when writing Domain Specific Languages (DSL's).

Example from ILC '09 at MIT

- Alex Fukunaga (Tokyo University) spoke on The Satisfyability Problem
- A DSL for SAT algorithms
- Used a biological evolution inspired algorithm

### 3.1.4 REPL

The Read Eval Print Loop

Interactive code development

Instead of just dump a stack trace and die on an error... you can edit data and functions (they look the same) and continue your program!

nREPL and emacs

### 3.1.5 Lisp successes

Artificial Intelligence

Travel Planning

Google's $700 M acquisition of ITA

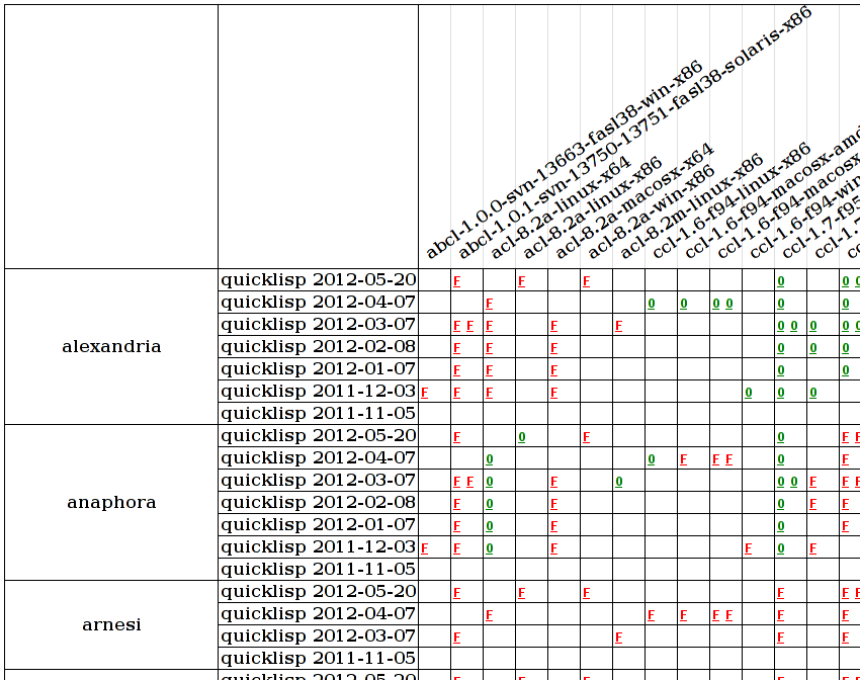
Scientific Computing Lisp

SciCL augments Common Lisp with an extensive library of aggregate-wise ("AG-wise") operations on arrays, providing the essential functionality of languages such as APL, Fortran 90, IDL and Matlab.

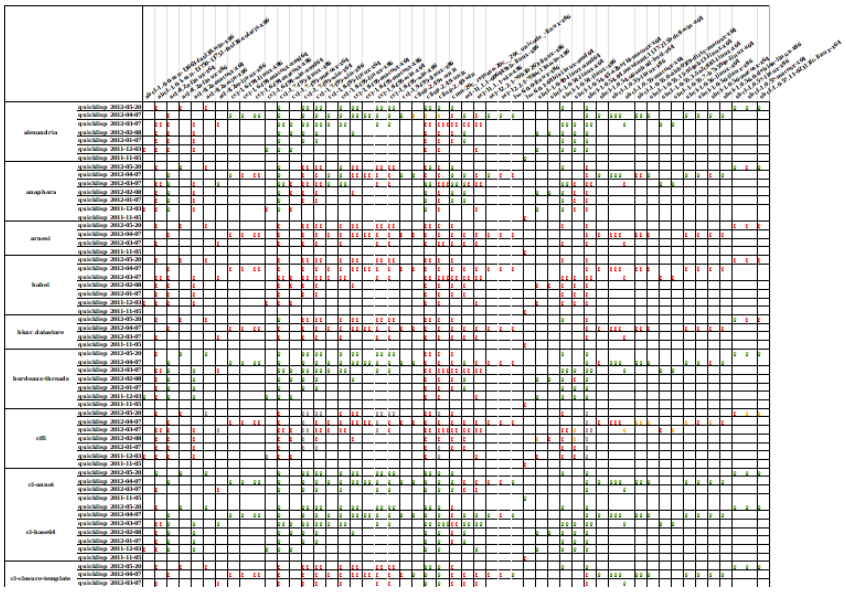
http://www.siginf.com/

## 3.2 Disadvantages of Common Lisp

### 3.2.1 There is a standard, but no compatibility test kit (as for Java)



### 3.2.2 Having a standard is no guarentee of compatibility



### 3.2.3 The "library" problem

There isn't a consistent disclipline within the LISP community about how best to package and distribute libraries.

There is ASDF which is a low-level approach and QuickLisp (which uses ASDF) to create a CPAN-like high-level approach. But not all the libraries are Quicklisp enabled (nor work on all Lisp implementations).

Writing Lisp code means making a committment to a set of implementations and a packaging approach – and porting missing bits.

### 3.2.4 The porting problem

The current Lisp implementions often use some assembly langauge at a low level... which limits their portability. For example here is the picture for SBCL:

| | X86 | AMD64 | PPC | SPARC | Alpha | MIPSbe | MIPSle |
|---|---|---|---|---|---|---|---|
| **Linux** | 1.0.57 *newest* | 1.0.57 *newest* | 1.0.28 | 1.0.28 | 1.0.28 | 1.0.23 | 1.0.28 |
| **Darwin (Mac OS X)** | 1.0.55 | 1.0.55 | 1.0.47 | | | | |
| **Solaris** | 1.0.56 | 1.0.55 | | 1.0.23 | | | |
| **FreeBSD** | 1.0.23 | 1.0.22 | | | | | |
| **NetBSD** | 1.0.22 | 1.0.56 | 1.0.23 | | | | |
| **OpenBSD** | 1.0.55 | 1.0.55 | 1.0.55 | | | | |
| **Windows** | 1.0.55 | | | | | | |

In addition to the official SBCL, a Windows fork exists that improves support for the Windows platform, especially in the area of threads, I/O, and x86-64 support. Though is has not yet been incorporated into mainline, Windows users may want to consider using it in the meanwhile.

| | Key |
|---|---|
| | Available and supported |
| | Port in progress |
| | Not available (porters welcome!) |
| | No such system |

| | Processors |
|---|---|
| **X86** | X86 (32-bit Intel and compatible) |
| **AMD64** | 64-bit X86 (AMD64, EM64T, Via Nano) |
| **PPC** | PowerPC |
| **SPARC** | SPARC and UltraSPARC |
| **Alpha** | DEC Alpha |
| **MIPSbe** | MIPS (big endian mode) |
| **MIPSle** | MIPS (little endian mode) |

#### 3.2.5 The concurrency problem

The tools for managing threads and concurrent operations are not part of the ANSI Specification and thus left as an “exercise for the reader” :(

# 4 Why Java?

## 4.1 Avantages of Java

### 4.1.1 Cross Plaform

WORA = Write Once Run Anyware even with s/Run/Debug/ it is still better

The assembly language coding has been done for you (on many OS ARCH combinations)

Zero assembler JIT

### 4.1.2 Rich set of libraries

Many many libraries are available for Java

### 4.1.3 Enterprise adoption

Very popular

### 4.1.4 Performance: Dynamic code optimization

HotSpot Virtual Machine

- on the fly profiling,
- inlining, loop unrolling
- de-opt/reopt
- escape analysis
- dead code elimination

### 4.1.5 Performance: Garbage Collection

Several proven GC algorithms

- throughput
- pause time

## 4.2 Disadvantages of Java

### 4.2.1 Mutation is (almost) required

Graph of mutable, stateful objects are a nightmare to manage with concurrency

Unconcious mutation is a source of bugs

- passing mutable objects to functions
- using mutable objects as keys

Coping mechanisms

- copy constructurs "freeze state" in a snapshot
- deep copy
- collections offer a weak facade

#### 4.2.2 Single inheritence hierarchy

Object Oriented Programming is used for **everything** even when it doesn't make sense

- java.lang.Math has to gather up a bunch of static functions
- This leads to the "Kingdom of Nouns"

Interfaces are a soft attempt at multiple inheritance
Aspect oriented programming is an attempt to avoid code duplication in the face of strong typing.

#### 4.2.3 Complex

Java Fetishizes Complexity

#### 4.2.4 Java EE Containers

file:overview-architecture-cont.gif

#### 4.2.5 Java EE APIs in the Web Container

file:overview-architecture-web.gif

#### 4.2.6 Java EE APIs in the EJB Container

file:overview-architecture-ejb.gif

#### 4.2.7 Java EE APIs in the Application Client Container

file:overview-architecture-acc.gif

# 5 Why Clojure?

## 5.1 Advantages of Clojure

### 5.1.1 Easy interoperation with Java

(.toUpperCase "hello")
Embraces the power of the JVM

- also runs on the CLR and on JavaScript

Typing support without the burden of strong typing

- Common Lisp typing is a hint to the compiler!

(defn $\#^{P}ropertiesas{-}properties$"$Convertanyseqofpairstoajava.utils.Propertiesinstance.Usesas$
$strtoconvertbothkeysandvaluesintostrings.$"{: $tagProperties$}[$m$]($let$[$p$($Properties.$)]($doseq$[[$kv$]$m$](.$se$
$strk$)($as-strv$)))$p$))

### 5.1.2 Leverages advantages of a Lisp

Clojure models its data structures as immutable objects represented by interfaces

Many functions defined on few primary data structures (seq, map, vector, set).

Clojure multimethods decouple polymorphism from OO and types

- Supports multiple taxonomies
- Dispatches via static, dynamic or external properties, metadata, etc

### 5.1.3 Multimethods

Multimethod Examples...

file:multi-1.clj

file:multi-2.clj

### 5.1.4 Lazy sequences

All Clojure collection types are sequences (as are Java collections and Arrays)

A **lazy sequence** will only compute contents when they are consumed.

- Performance pro-tip: avoid doing work (until it's actually necessary)

file:lazy-seq-1.clj

file:lazy-seq-2.clj

### 5.1.5 Functional Programming

Immutable data + first-class functions, supporting recursion

Dynamic polymorphism

Emphasizes recursive iteration instead of side-effect based looping

user> (let [my-vector [1 2 3 4] my-map {:fred "ethel"} my-list (list 4 3 2 1)] (list (conj my-vector 5) (assoc my-map :ricky "lucy") (conj my-list 5) my-vector my-map my-list)) -> ([1 2 3 4 5] {:ricky "lucy", :fred "ethel"} (5 4 3 2 1) [1 2 3 4] {:fred "ethel"} (4 3 2 1))

### 5.1.6 Software Transactional Memory

Core data structures are immutable and can easily be shared between threads
Mutation **is** possible using locks to avoid conflicts

- dosync, ref, set, alter, et al, supports sharing changing state between threads in a synchronous and coordinated manner.
- The agent system supports sharing changing state between threads in an asynchronous and independent manner.
- The atoms system supports sharing changing state between threads in a synchronous and independent manner.
- The dynamic var system supports isolating changing state within threads.

### 5.1.7 No spec, one implementation

Disadvantages: All eggs in one basket
Advantages: Clojure works **everywhere** Innovation happens quickly Core data structures are extensible abstractions Vibrant community

### 5.2 Disadvantages of Clojure

#### 5.2.1 The state of Clojure Contrib (is a challenge)

"Modularization of Contrib"

http://dev.clojure.org/display/doc/Clojure+Contrib

The idea is that everything that hasn't been modularized yet is supposedly either low quality or in low demand

This is improving...

## 6 Concurrent Programming

### 6.1 Threads

(def long-calculation (future (apply + (range 1e8)))) @long-calculation

(def bg (future (Thread/sleep 5000) (println "done"))) @bg

### 6.2 Threading harness (for examples)

file:futures.clj

### 6.3 Types of operations

Coordinated: multiple actors must cooperate to produce correct results

Synchronous: caller blocks evaluation

| Operations | `Coordinated` | `Uncoordinated` |
|---|---|---|
| `Synchronous` | Refs | Atoms |
| `Asynchronous` | | Agents |

NOTE: as the focus of Clojure is in-process concurrency the Coordinated - Asynchronous case is not implemented directly in the language (e.g. more for databases)

### 6.4 Refs

Coordinated + Synchronous

STM has ACID properties (except D):

- Atomic
- Consistent
- Isolated
- (Durability)

(dosync ;; the body is a transaction (alter myref f arg1 arg2)) ;; mutation of a reference

file:refs.clj

### 6.5 Atoms

Uncoordinated + Synchronous

Safe mutation within a thread: compare and set

;; The function f will be re-tried of the value ;; of myatom changed during the call (swap! myatom f)

file:atoms.clj

### 6.6 Agents

Uncoordinated + Asynchronous
    file:agents.clj

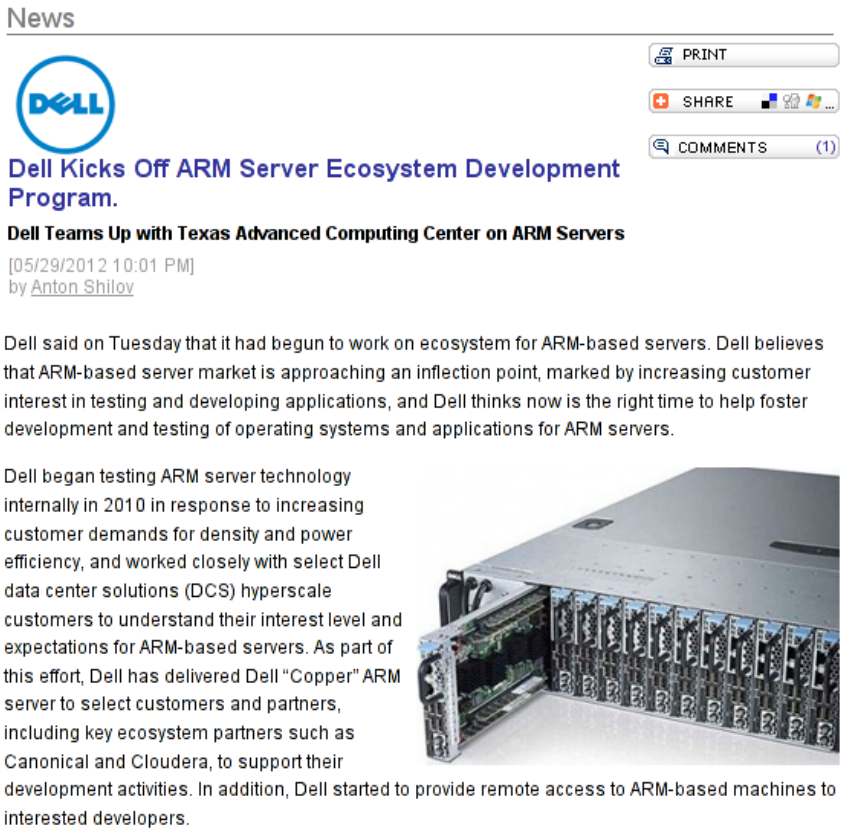
## 7 Promising Future

### 7.1 Java

Moore's law in combination with new architectures makes Java very attractive from mobile to super computers.
    Sun originally wanted Java to enable customers to use SPARC
    Today many Enterprises run on Intel architectures
    But what about tomorrow?

#### 7.1.1 ARM looks very good for size, cost, heat

Maybe we will see ARM in the data center? Red Hat is now porting OpenJDK to arm64!!!



News

PRINT

SHARE

COMMENTS (1)

Dell Kicks Off ARM Server Ecosystem Development Program.

**Dell Teams Up with Texas Advanced Computing Center on ARM Servers**

[05/29/2012 10:01 PM]
by Anton Shilov

Dell said on Tuesday that it had begun to work on ecosystem for ARM-based servers. Dell believes that ARM-based server market is approaching an inflection point, marked by increasing customer interest in testing and developing applications, and Dell thinks now is the right time to help foster development and testing of operating systems and applications for ARM servers.

Dell began testing ARM server technology internally in 2010 in response to increasing customer demands for density and power efficiency, and worked closely with select Dell data center solutions (DCS) hyperscale customers to understand their interest level and expectations for ARM-based servers. As part of this effort, Dell has delivered Dell "Copper" ARM server to select customers and partners, including key ecosystem partners such as Canonical and Cloudera, to support their development activities. In addition, Dell started to provide remote access to ARM-based machines to interested developers.

### 7.1.2 We are seeing ARM everywhere in embedded devices

Raspberry Pi

### 7.1.3 Java as assembly language

For these reasons Clojure is one of many vibrant, alternative languages on the JVM which include:

- JRuby
- Scala
- Jython
- IKVM.NET
- Gosu

- Smalltalk
- JavaScript

## 7.2 Bleeding Edge OpenJDK features

NOT yet truly being used by Clojure

### 7.2.1 Fork/Join

Bring Doug Lea's Fork/Join framework into Clojure

Primary example **pmap**

- using the shortest map/reduce tutorial ever

  user> (def mylist '(1 2 3 4 5 6)) #'user/mylist user> (map even? mylist) (false true false true false true) user> (reduce 'or (map even? mylist)) true

David Liebke: "From Concurrency to Parallelism" http://incanter.org/downloads/fjclj.pdf

### 7.2.2 Tail Call Optimization

Save space on the stack:

call factorial (3) call fact (3 1) call fact (2 3) call fact (1 6) call fact (0 6) return 6 return 6 return 6 return 6 return 6

call factorial (3) call fact (3 1) replace arguments with (2 3), jump to "fact" replace arguments with (1 6), jump to "fact" replace arguments with (0 6), jump to "fact" return 6 return 6

NOTE: Clojure does have **recur** and **trampoline** but the JVM itself lacks a generic optimization for TCO (but there is an older patch in the MVLM repo).

https://en.wikipedia.org/wiki/Tail_call

### 7.2.3 Invoke Dynamic

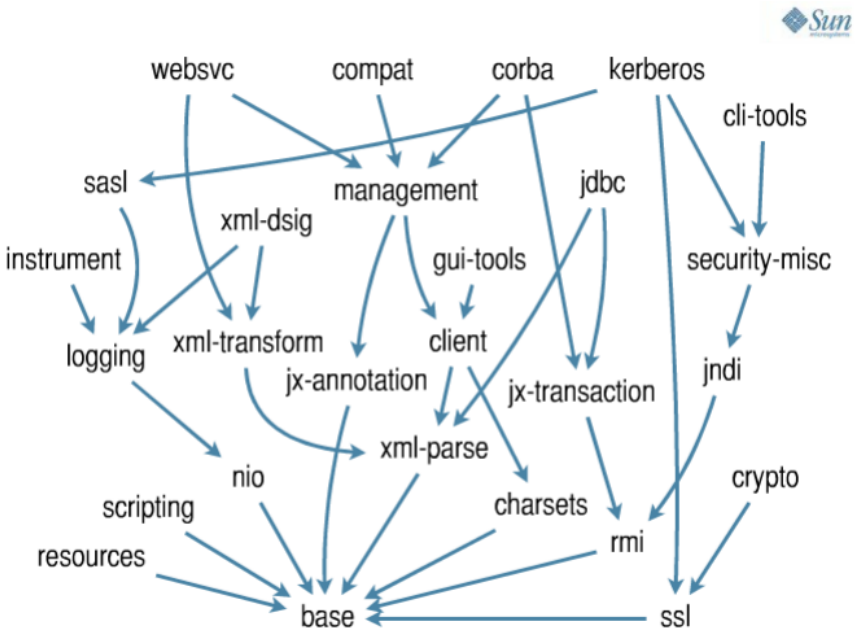
JSR 292

Enables the HotSpot VM to **see** into your "JVM Language" code and optimize it!

Why Clojure Doesn't Need Invokedynamic (Unless You Want It to be More Awesome) http://blog.headius.com/2011/10/why-clojure-doesnt-need-invokedynamic.html

### 7.2.4 Modularization (Jigsaw)

Better startup time Finer grained dependencies Smaller footprint (embedded)




## 8 Conclusion

LISP is incredibly powerful (don't be afraid of the parens)

Clojure is the best LISP now (because of the JVM)

Java means future proof for platforms in the cloud and the "Internet of Things".

There are **still** many optimizations waiting to be made

The #1 reason to use Clojure: productivity.



Will post a pointer to slides on {twitter,identi.ca} @tmarble

## 9 Q/A

Questions?

# 10 Extra

## 10.1 What is that presentation tool?

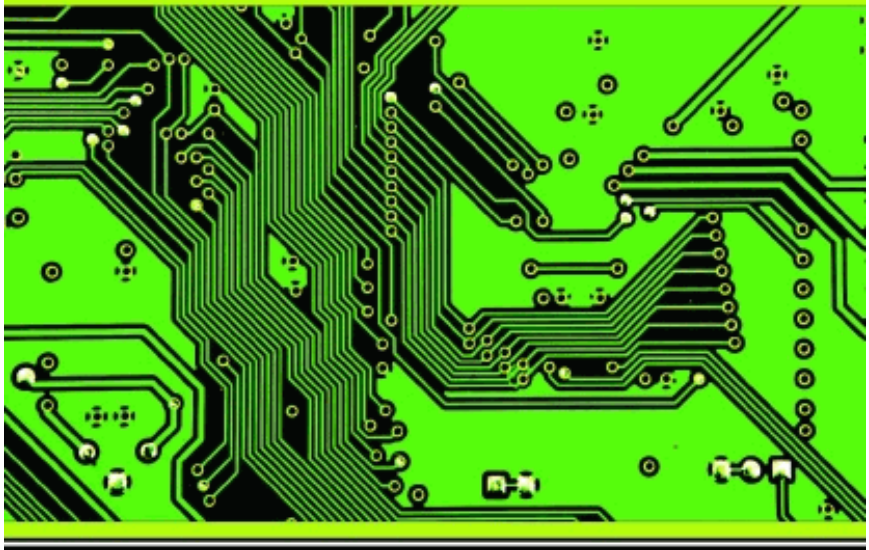
Emacs!

This is **org-tree-slide** from https://github.com/takaxp/org-tree-slide
For more on org mode see http://orgmode.org/org.html
Yes I will share my “slides” on my website http://tmarble.info9.net

A GNU MANUAL

# The Org Mode 7 Reference Manual

Organize your life with GNU Emacs



Carsten Dominik and others

PUBLISHED BY NETWORK THEORY LTD

### 10.1.1 Note's on what I put in my .emacs.d/custom.el

(require 'org-tree-slide)
(global-set-key (kbd "<f1>") 'show-all) (global-set-key (kbd "<f5>") 'text-scale-decrease) (global-set-key (kbd "<f6>") 'text-scale-increase) (global-set-key (kbd "<f8>") 'org-tree-slide-mode) (global-set-key (kbd "<f9>") 'org-tree-slide-content) (global-set-key (kbd "<f10>") 'hide-sublevels)
Printing to PDF: C-c C-e p (org-export-as-pdf)

## 10.2 The Tools I am using

### 10.2.1 Maven

Finding dependencies: `mvn dependency:tree -DoutputFile=dependency.txt`
my-website:my-website:jar:0.1.0-SNAPSHOT +- org.clojure:clojure:jar:1.3.0:compile noir:noir:jar:1.2.2-SNAPSHOT:compile +- compojure:compojure:jar:1.0.0-RC2:compile

+- org.clojure:core.incubator:jar:0.1.0:compile
+- org.clojure:tools.macro:jar:0.1.0:compile
+- clout:clout:jar:1.0.0:compile
ring:ring-core:jar:1.0.1:compile
+- commons-io:commons-io:jar:1.4:compile
+- commons-fileupload:commons-fileupload:jar:1.2.1:compile
javax.servlet:servlet-api:jar:2.5:compile

+- org.clojure:tools.namespace:jar:0.1.0:compile

org.clojure:java.classpath:jar:0.1.0:compile

+- clj-json:clj-json:jar:0.4.3:compile

org.codehaus.jackson:jackson-core-asl:jar:1.5.0:compile

+- ring:ring:jar:1.0.1:compile

+- ring:ring-devel:jar:1.0.1:compile
ns-tracker:ns-tracker:jar:0.1.1:compile
+- ring:ring-jetty-adapter:jar:1.0.1:compile
+- org.mortbay.jetty:jetty:jar:6.1.25:compile
org.mo
org.mortbay.jetty:jetty-util:jar:6.1.25:compile
ring:ring-servlet:jar:1.0.1:compile

+- hiccup:hiccup:jar:0.3.7:compile +- clj-stacktrace:clj-stacktrace:jar:0.2.3:compile
+- ring-reload-modified:ring-reload-modified:jar:0.1.1:compile +- net.java.dev.jets3t:jets3t:jar:0.8.1:comp

+- commons-codec:commons-codec:jar:1.3:compile
+- commons-logging:commons-logging:jar:1.1.1:compile
+- commons-httpclient:commons-httpclient:jar:3.1:compile
com.jamesmurty.utils:java-xmlbuilder:jar:0.4:compile

org.mindrot:jbcrypt:jar:0.3m:compile

### 10.2.2 Leiningen

Leiningen is awesome https://github.com/technomancy/leiningen
Use with nREPL https://github.com/kingtim/nrepl.el

### 10.2.3 Jenkins

Continuous Integration Server: http://jenkins-ci.org/
Amazing Plugins: https://wiki.jenkins-ci.org/display/JENKINS/Plugins
The ones that I use:

- Trac Publisher
- Dependency Graph Viewer
- IM
- Pathignore (essential for big git repo)
- SSH Slaves
- Thin Backup
- Build Result Trigger

Fun ones

- Gravatar
- Emotional Jenkins

#### 10.2.4 Trac

http://trac.edgewall.org/

- Tickets (bugs, tasks), Reports, Browse code, Timeline, Wiki
- Can now use git (yeah!)
- Integration with Jenkins http://trac-hacks.org/wiki/XmlRpcPlugin