Why Clojure Matters

Creighton Kirkendall

Principal Consultant SEI

Email: ckirkendall@gmail.com

Twitter: @crkirkendall

Github: https://github.com/ckirkendall

Why Clojure Matters

- Lisp Refresher
- Concurrency & State
  - Java Concurrency and Mutability
  - Identity vs State
  - Immutability & Persistent Data Structures
  - Clojure Example
- Web Development
  - The Death of MVC
  - Pinot Noir (thirsty?)

Why Clojure Matters

Clojure Syntax (lisp)

Name	**Clojure Syntax**	**Java Equivalent**
Symbols	atom, foo-bar, *foo*, etc.	Variables Names
Literals	42, "foo", nil, true, false, \c, :foo	Same
Keywords	:foo (like symbols, prefixed with colon)	None
Lists	(a b c) & '(a b c)	LinkedList
Vectors	[a b c]	Array
Maps (hashes)	{:a 1 :b 1} or {:a 1, :b 2}	Map
Sets	#{:a :b :c}	Set

Why Clojure Matters

Clojure Syntax (lisp) – defining a variable

Clojure Syntax	Java Equivalent
(def a “test”)	String a=”test”;

Why Clojure Matters

Clojure Syntax (lisp) – defining a function

Clojure Syntax	Java Equivalent
(fn [x] (println x)) #(println %1)	No parallel for in-line functions

(defn tmp [x] (println x))	public void tmp(Object x){
	 System.out.println(x);
	}

Why Clojure Matters

Clojure Syntax (lisp) – calling a function

Clojure Syntax	**Java Equivalent**
(test “test”)	test(“test”);

Why Clojure Matters

Java and Concurrency

It's the mutable state, stupid. All concurrency issues boil down to coordinating access to mutable state. The less mutable state, the easier it is to ensure thread safety.

Java Concurrency in Practice

Why Clojure Matters

Java and Concurrency

Immutable objects are automatically thread-safe. Immutable objects simplify concurrent programming tremendously. They are simpler and safer, and can be shared freely without locking or defensive copying.

Java Concurrency in Practice

Why Clojure Matters

Java and Concurrency

A program that accesses a mutable variable from multiple threads without synchronization is a broken program

Java Concurrency in Practice

Why Clojure Matters

Java and Concurrency

Example

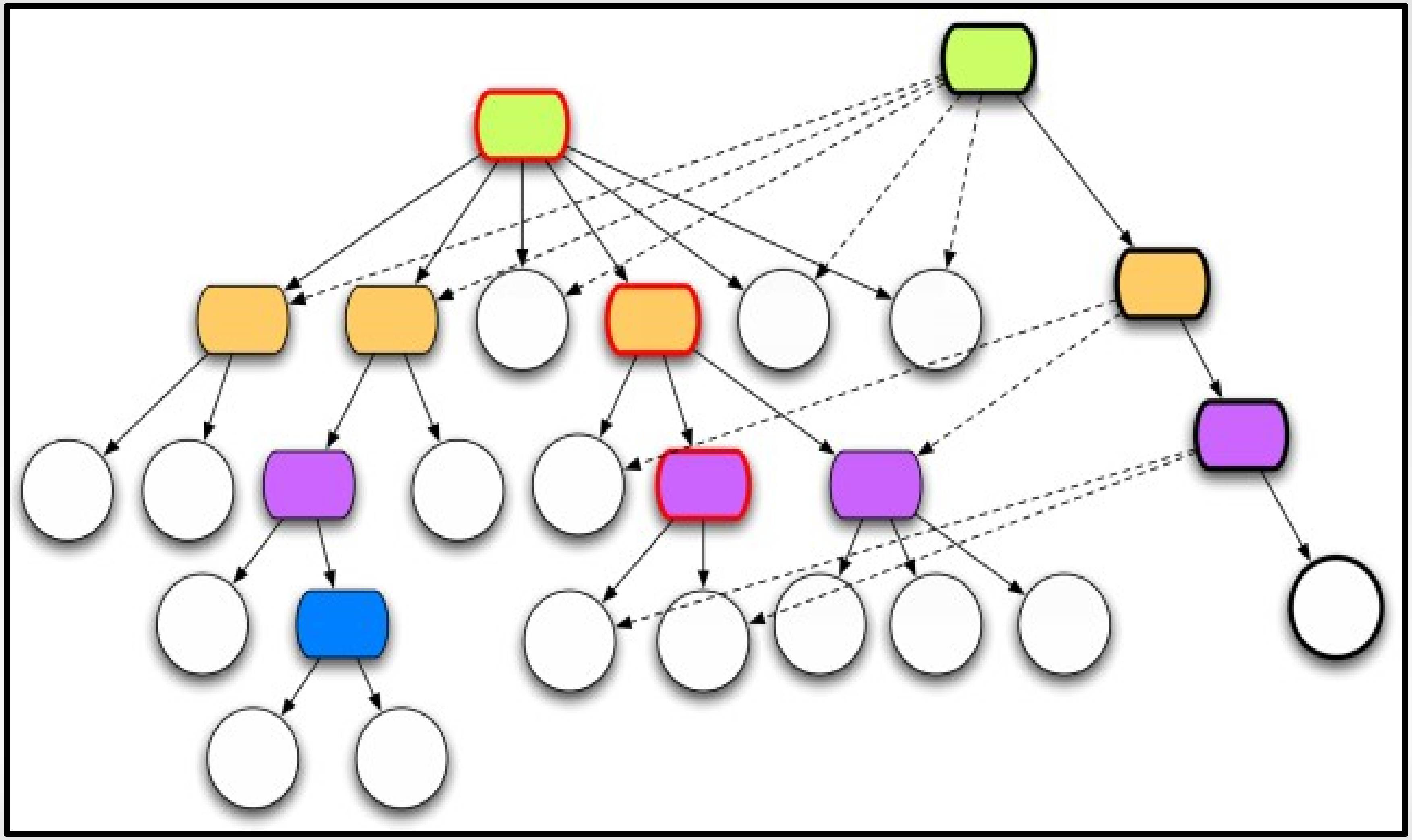
Why Clojure Matters

Identity vs State

- Java
  - Identity ← 1 to 1 → state
- Clojure
  - Identity ← 1 to N → state
  - Version Control

Why Clojure Matters

Immutability & Persistent Data Structures



Why Clojure Matters

Clojure Mutable State

All mutable state access is thread safe!

- Atoms
  - swap!
- Refs
  - dosync, alter, commute, ref-set
- Agents
  - send

Why Clojure Matters

Clojure Concurrency

Example

Why Clojure Matters

The Death Of MVC

- What is wrong with MVC.
  - Limited User Experience
  - Resource Skill Set Limitations
- The Rise of Service Oriented UIs (AJAX, JSON)
  - Rich UI
  - Rise of the JavaScript Developer
  - Rise of SOA

Why Clojure Matters

Clojure and Clojurescript

Example

Why Clojure Matters

Q & A