Clojure

Usable Lisp

http://github.com/djwhitt/cposc2009

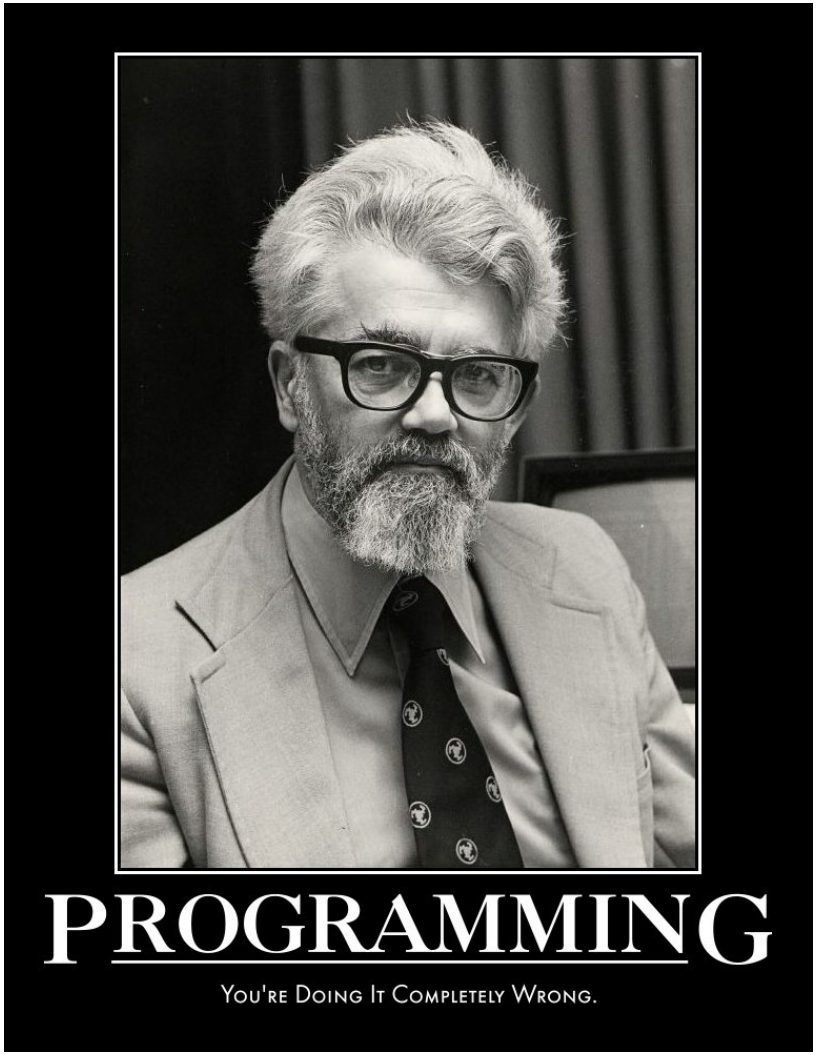
Agenda

- Why Clojure
- Clojure intro (actual code + hacking)
- Cool bonus material
- Questions are welcome at any point (especially during the code section later) but I may have to move on if they take too long

What is Clojure?

- Lisp dialect
- JVM hosted
- Functional
- Deep concurrency support
- Created by Rich Hickey →

Why Lisp?



- Simple syntax
- Few special cases
- Syntactic abstraction
- Interactive development
- Lisp is fun

Expressions

(fn arg1 arg2 … argn)

Compilation and Evaluation

Most languages:
Text → [compiler] → bytecode

Lisp:
Text → [reader] → data structure → [compiler] → bytecode

This is what enables macros.

Macros

```
(macroexpand '(and false 123))
```

→

```
(let* [and__3307__auto__ false] (if
and__3307__auto__ (clojure.core/and 123)
and__3307__auto__))
```

The JVM is your friend

- Lots of libraries
- Lots of languages
- Very good performance
- Sorry, the classpath still sucks

Why not Scheme or CL?

- No literal for maps (or vectors in CL afaik)
- Vectors, sets, and maps are generally harder to work with in Scheme and CL (not persistent, no sequence abstraction)
- JVM has lots of libraries
- Working with the JVM is simpler using Clojure compared with other JVM Lisps
- Clojure is designed for concurrency

Identity and State

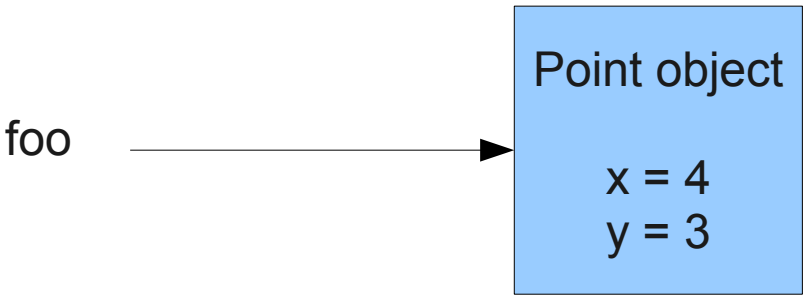
Process - program that includes change over time

State - value of identity at a time

Identity - succession of values over time

Identity and State

State and identity in your favorite OO language

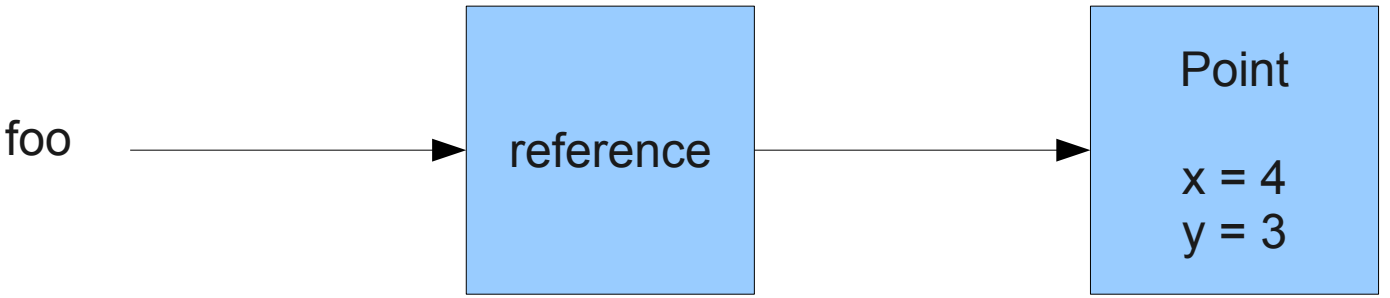


How do you change this while preserving identity?

Typically you mutate this thing

Identity and State

State and identity in Clojure



State is changed by atomically replacing referenced values.

Hickey's hair