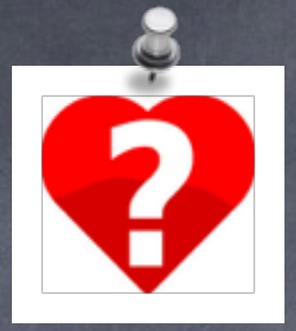




Scala & Clojure Playing Nice

David Pollak

Devoxx Poland ° June 2015

About @dpp

- Wrote some Spreadsheets
- Founded Lift/Wrote Beginning Scala
- Coding Clojure 3 Years
- Crazy Passionate Lawyer-trained Tech Dude

Preso Structure

- Background on Scala & Clojure
- Live Coding
- Thoughts & Questions

Scala

- Hybrid Functional/OO Language...
  All things to all people
- Gnarly Type System
- Java-like syntax
- Excellent Java Interopt

Clojure

- Modern Lisp/Functional
- Optional Type Systems
- Opinionated re: Immutability
- Super-Excellent Java Interopt

Both Compile to JVM ByteCode

... Can Subclass Java Classes

... And Implement Java Interfaces

Similarities

- Immutable Data & Collections
- Super easy to pass "functions" (really anonymous inner classes)
- Great for reducing complexity & concurrent systems
- Both address "Expression Problem"

Expression Problem

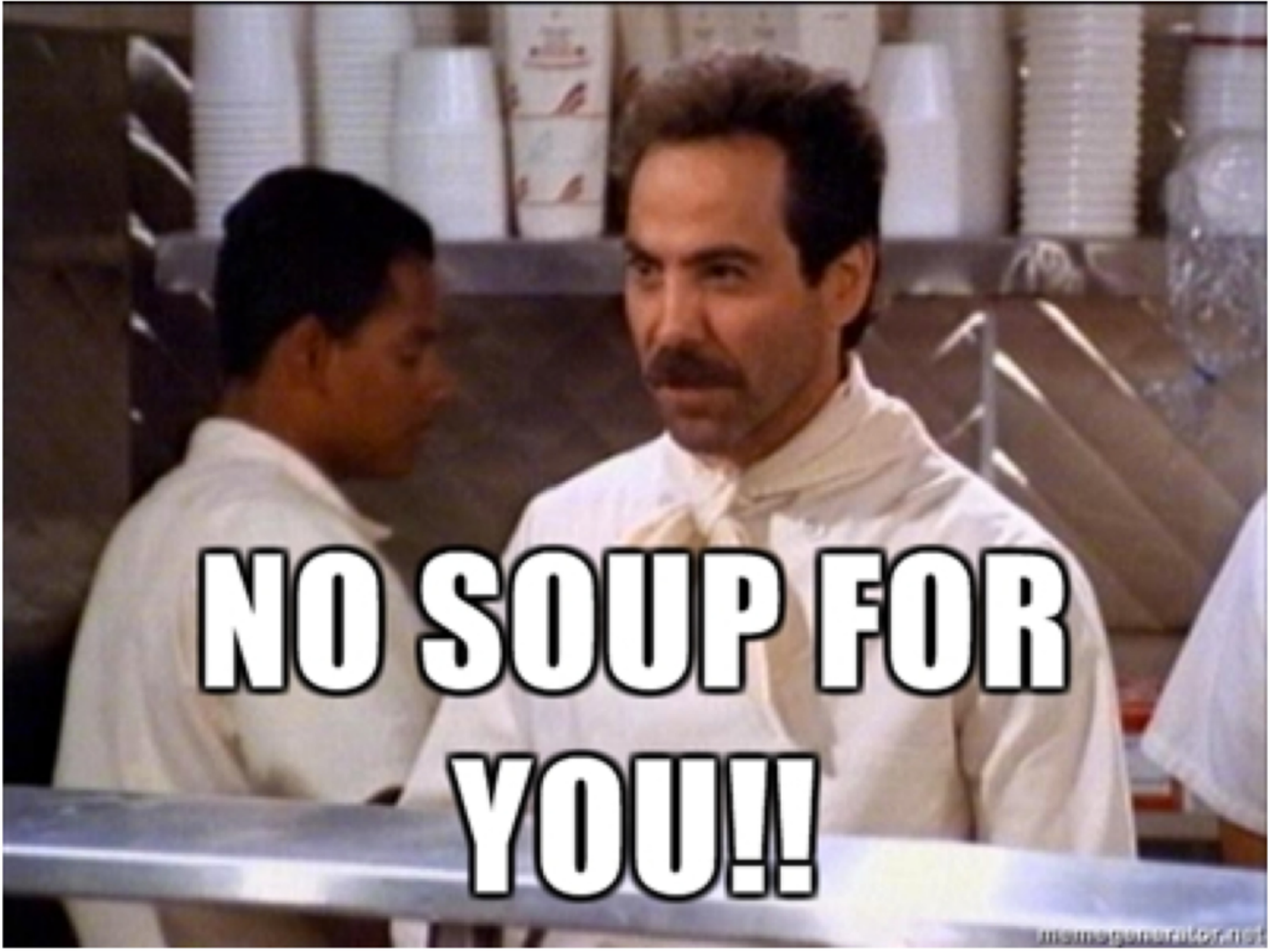
"The goal is to define a datatype by cases, where one can add new cases to the datatype and new functions over the datatype, without recompiling existing code."

ALL I want to do is add a method... to a library class!

...in Java

- Subclassing - can add new data
- But cannot add functions to existing data/classes


MT CYCLOPTIC COLLEAGUE TELLS ME
I CAN'T HAVE
"SHARKS WITH FRICKEN LASER BEAMS ATTACHED TO
THEIR HEADS
nemegenerator.net


NO SOUP FOR YOU!!

...in Scala

- Subclassing: Add new data
- implicits: "Scoped" adding new functionality to existing data – or "TypeClasses"

Scala Sample

```
"foo".toWombat()

class MyWombat(s: String) {
  def toWombat() …
}

implicit def asAWombat(s: String):
  MyWombat = new MyWombat(s)
```

...in Clojure

- Subclassing & Maps
- Protocols add functions to data

Clojure Sample

```
(defprotocol FromScala
 (to-c [x] "Scala -> Clojure"))

(extend Iterator FromScala
  {:to-c
   (fn [it](letfn [(build []
                  (if (.hasNext it)
                    (cons (to-c (.next it))
                          (lazy-seq (build)))
                    nil))]
       (build)))})

(defn seq-to [^Seq seq]
  (-> seq .iterator to-c))

(extend Seq FromScala
  {:to-c seq-to})
```

That Distributed & Concurrent Thing...

Distributed & Concurrent

- Easily Serializable
- Immutable
- Like REST: data in, answer out

Live Coding

Clojure Chat Server

```
(match
    (<! chat-server)
    [:add lst]
    (do
      (send! lst (take-last 40 @chats))
      (swap! listeners conj lst))

    [:remove lst]
    (swap! listeners (fn [info] (remove #(identical? lst %)
                                        info)))

    (msg :guard string?)
    (do
      (doseq [f @listeners] (send! f msg))
      (swap! chats conj msg))

    :else nil)
```

Lift — Set up xport

```
val clientProxy =
   session.serverActorForClient("sb.client.core.receive",
     shutdownFunc = Full(actor =>
                  postMsg.invoke('remove -> actor)),
     dataFilter = transitWrite(_))

postMsg.invoke('add -> clientProxy)

val serverActor = new LiftActor {
  override protected def messageHandler =
  {case JString(str) =>
       postMsg.invoke(ClojureInterop.transitRead(str))}}

Script(JsRaw("var sendToServer = " +
       session.clientActorFor(serverActor).toJsCmd).cmd)
```

HTML

```
<div id="app">Loading...</div>

<div data-lift="Actorize"></div>
```

HTML

```
<div>
    <h2>Chatting</h2>
    <ul>
        <li>Hi</li>
    </ul>
    <hr>
    <input id="in"> <button>Chat</
button>

</div>
```

Browser Receive

```
(defn receive [x]
  (let
    [msg (t-read x)]
    (cond
    (sequential? msg)
    (reset! chats (vec msg))

    (string? msg)
    (swap! chats conj msg)

    :else nil)))
```

Browser Render

```
(defn page []
  (xform page-text
         ["li" :* @chats]
         ["button"
          {:onclick
           (fn [] (some->
                   (swap-id! "in" "")
                   first
                   t-write
                   js/sendToServer))}]
  ))
```

Wrap-up

- Easy to convert between Scala & Clojure types
- Clojure & Scala do well for distributed apps
- JVM makes it easy to play together

Scala & Clojure

Play well together

Thanks!

Questions?