LISP ON

You know you want it.

An Advance Warning

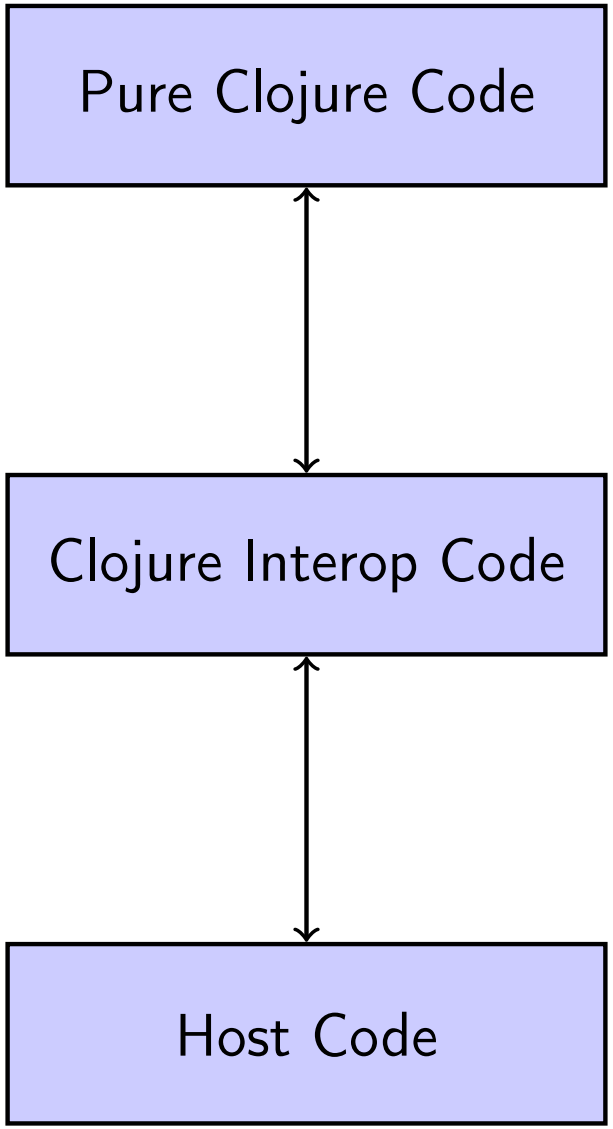
To paraphrase Aaron Bedra:

> *I'm going to sound like part of the 'Lisp weanie' crowd coming and telling you that Lisp will solve all of your problems.*

What is Clojure?

- A dialect of Lisp
- Designed and implemented by Rich Hickey
  - Initial release: 2007
- Originally written for the Java platform
- ClojureCLR extended it to .NET in 2009
- ClojureScript brought it to the browser in 2011
- IMPORTANT: Different implementations are very similar but not identical.

The Clojure System



The REPL

- Enables fast prototyping and testing of new code
- Java, CLR do not have any form of REPL
- JavaScript has a REPL available in most browsers
  - ClojureScript code can be debugged in-browser via a REPL. This is *despite* having been translated into optimized, minified JS. [1]

[1] http://swannodette.github.io/2013/09/15/source-maps/

Code as Data

Code is data, why shouldn't we treat it as such?

- Generation
  - Helps reduce boilerplate code
  - `(defsomething ...)` forms are common; eg `(defrecord ...)`, `(deftemplate ...)`, `(defactivity ...)`
- Transformation
  - cljx [2] transforms annotated code
    - Ex: `StringBuilder` (Java) vs. Google Closure's `StringBuffer` (JS)
  - core.async [3] has the `(go ...)` macro; transforms synchronous code into asynchronous

[2] https://github.com/lynaghk/cljx
[3] https://github.com/clojure/core.async

Functional Programming

For many problems, functional programming provides clean, easily undestandable solutions.

- In Clojure
  - First-class Functions (can be created and returned from other functions)
  - Function inputs are (mostly) immutable
  - State is avoided except when necessary
- In Java
  - No first-class functions (although *function object* pattern helps some)
  - Mutable function inputs
  - State commonly used

Not every problem can be easily solved in a functional manner. Clojure does not prevent using procedural methodology, one must merely be *explicit*.

The Big Downside

))
))
))
))
))
))
))
))
))
))
))
))
))
))
))
))
))
))

Why Clojure over other Lisps?

- Platform compatability (if you have to work on the JVM, you have to work on the JVM)
- Nice syntactic sugar
  - Contrast `(make-hash-table)` (Common Lisp) with `{:a 1, :b 2}` (Clojure)
- Extensive libraries
  - Aside from the host system libraries, many phenomenal Clojure libraries such as `core.async`, `om`, `compojure`, ...

Hello, World!

```
(ns example.hello-world)

(defn -main [& args]
    (println "Hello, world!"))
```

ClojureScript Case Study: React.js and Om

- React.js [4]: created by *Facebook*
  - Functional/Object-Oriented interface library
  - Used in Facebook's chat interface and for the entirety of Instagram
- Om [5]: created by David Nolen (@swannodette)
  - Clojure wrapper over Facebook's React.js
- Consistently 2-4X faster than Backbone.js on TodoMVC benchmarks [6]
- Why use Om over React.js?
  - Uses immutable structures: equality checks become ref. equality checks
  - Always batches updates onto frame renders; avoids unnecessary DOM hits that are never displayed

---

[4] http://facebook.github.io/react/

[5] https://github.com/swannodette/om

[6] http://swannodette.github.io/2013/12/17/the-future-of-javascript-mvcs/

Demo Code

Demo code is available at:

https://github.com/emallson/comment-example