LISP ON

ALL THE THINGS

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

Pure Clojure Code

Clojure Interop Code

Host Code
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. 

---

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](https://github.com/lynaghk/cljx) transforms annotated code
    - Ex: StringBuilder (Java) vs. Google Closure’s StringBuffer (JS)
  - [core.async](https://github.com/clojure/core.async) has the (go ...) macro; transforms synchronous code into asynchronous
Functional Programming

For many problems, functional programming provides clean, easily understandable 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 compatibility (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.
  - Always batches updates onto frame renders; avoids unnecessary DOM hits that are never displayed

---

⁵ [https://github.com/swannodette/om](https://github.com/swannodette/om)
Demo Code

Demo code is available at:

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