



**If we are going to transpile
JavaScript, why not use
ClojureScript?**

Shivek Khurana

About me

- Apps since 2008, Clojure since 2017
- Consultant at Juxt Ltd.
- Enterprise Apps
- Currently building a system for Vodafone

JUXT



Built an LMS from 2012 to 2016

Long projects == Living with your past mistakes

Built an LMS from 2012 to 2016

Long projects == Living with your past mistakes



JS Community Conundrum

LI + EC = HVOD + C

JUX^T



JS Community Conundrum

$$LI + EC = HVOD + C$$

Language
Imperfections

Enthusiastic
Community

High Velocity
Open Source
Development

Abandoned
Libs & Ideas



Clojure/Script

*All the best games are easy
to learn and difficult to
master.*

— Nolan Bushnell (Founder, Atari)

;; LISP – LISt Processing

```
'(1 2 3)
```

```
(+ 1 2 3)
```

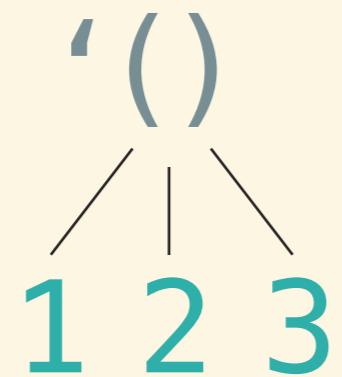
```
(defn sum [a b]  
  (+ a b))
```

JUXT

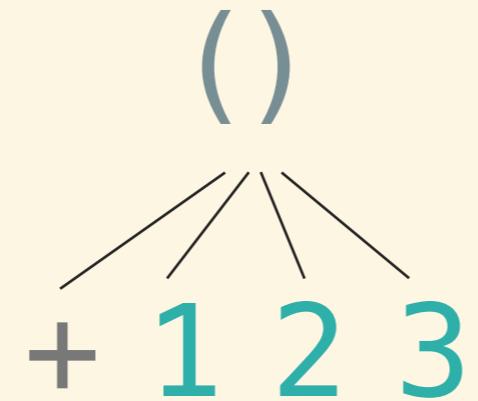


;; LISP – LISt Processing

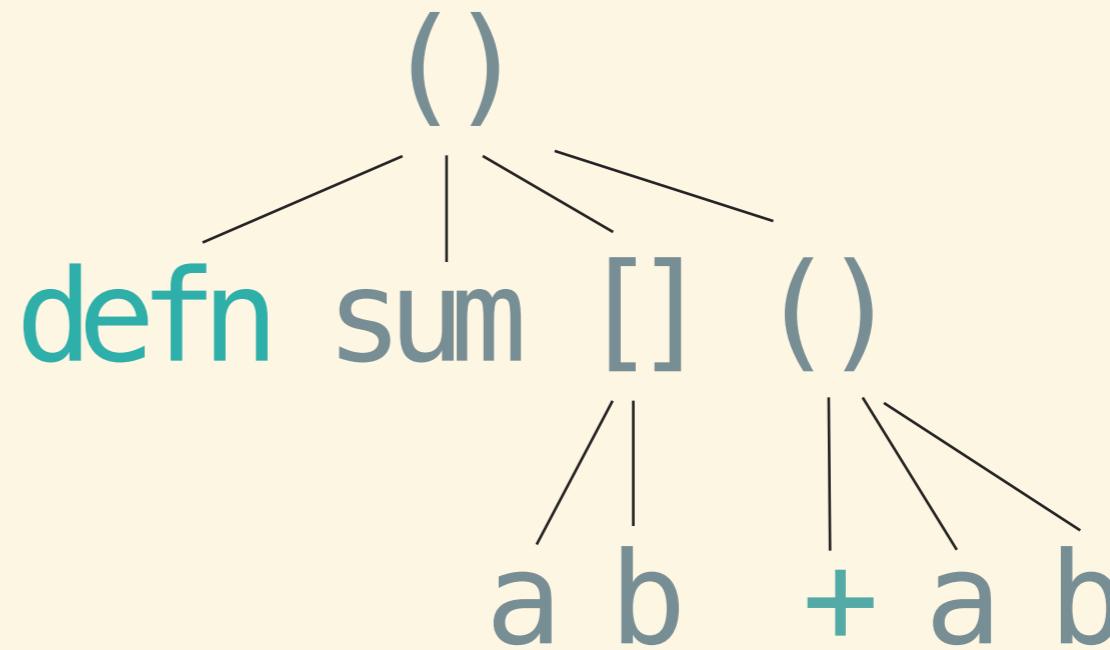
' (1 2 3)



(+ 1 2 3)



(defn sum [a b]
(+ a b))



;; Functional

```
(def person
  {:_name "Vienaa"
   :age 24})  
  
(get person :name)  
(get person :age)
```

```
class Person {  
  init(name, age) {...}  
  getName() => {...}  
  getAge() => {...}  
}  
  
const v = Person("Viena", 24)  
v.getName()  
v.getAge()
```

JUST



;; Atoms
;; thread safe/ mutable constructs

```
(defonce count (atom 0))  
(swap! count inc) ;; => 1  
count ;; => 1  
(reset! count 4) ;; => 4  
count ;; => 4
```

JUST



;; Homiconic ;; Code is data

' (1 2 4)

{name: "Shivek Khurana"}

[:a :b]

["apples", "mangoes", 8]

(defn sum [a b]
(+ a b))

```
function sum(a, b) {  
    return a+b;  
}
```

JUXT





REPL - Developing inside the Runtime

Observation Driven Dev

Text Editor

Code



Transpile +
Hot Swap

Runtime (Browser)

Updated App

Perform Actions
Observe

Updated App

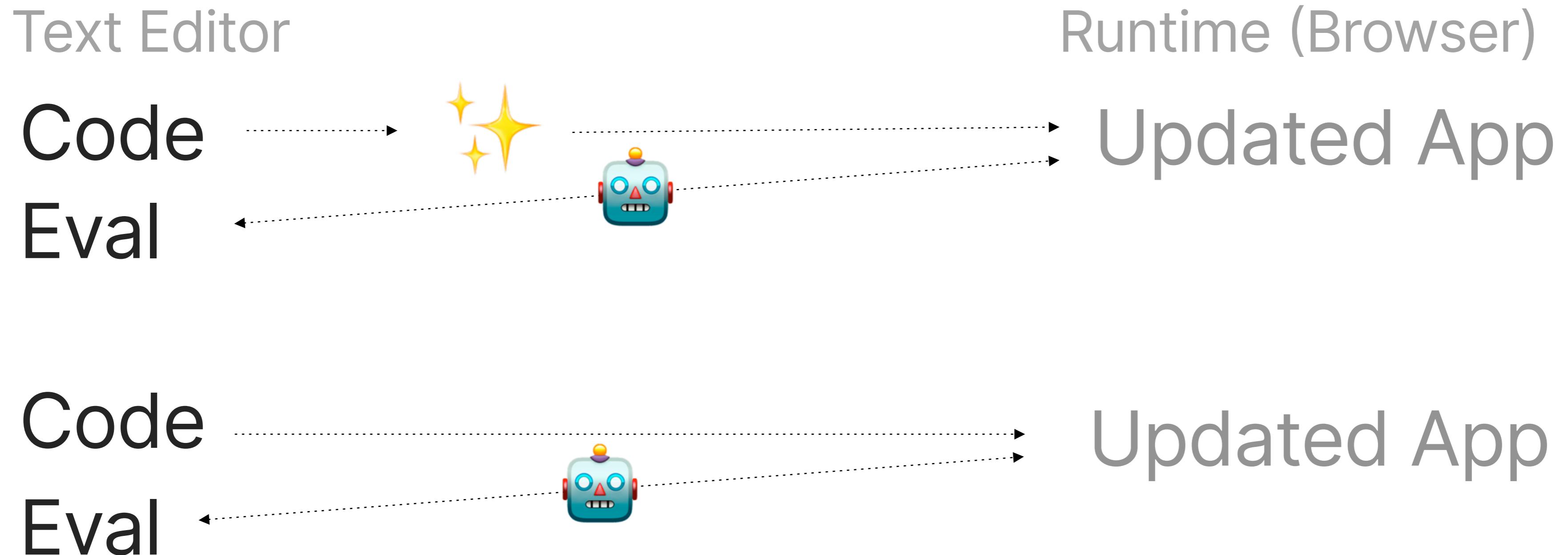
Perform Actions
Observe

Updated App

JUXT

Krim
Labs

REPL Driven Dev



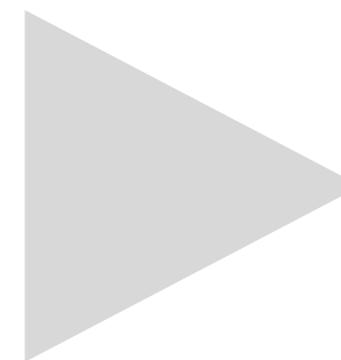
REPL Demo - Tic Tac Toe

State

Lattice

```
[[{:x nil nil}  
[nil :x nil]  
[nil nil :o]]
```

Turn :o



React

:o's turn

x	-	-
-	x	-
-	-	o

Reset

JUXT

Krim
Labs

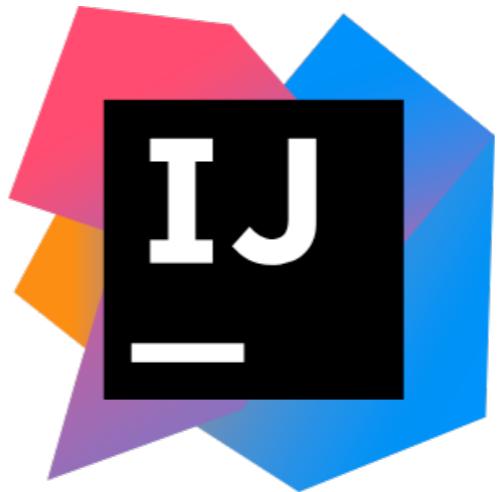
With REPL - Develop only a function

- Precisely load the relevant function
- Test with inline feedback
- Rich Comments in git history

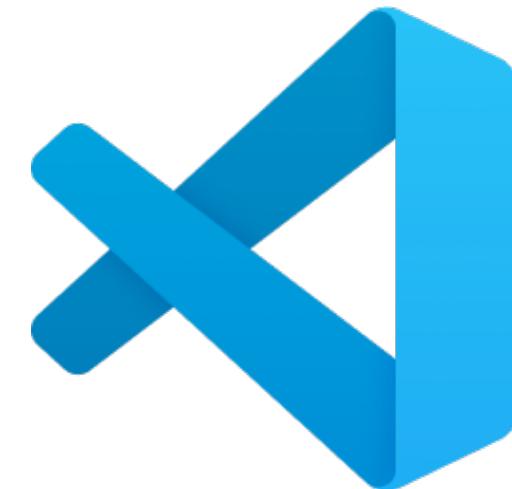
Editor Integration



(cider[[]])



Cursive



(()) *Calva*



Fireplace

Interop with JS ecosystem

Google Closure Compiler



- Open for 10+ years
- Google Closure Lib
- Cross Platform

;; Just add js/foo

```
(.getElementById js/document "root")
```

```
document.getElementById("root")
```

```
(new js/FormData)
```

```
new FormData()
```

```
(clj->js {:hello :world})
```

```
#js {:hello "world"}
```

JUST



;; Consuming NPM Packages

```
( :require ["react-dom/server" :refer [renderToString]] )  
( renderToString react-element)
```

JUXT



;; Write Node Libs

```
(ns demo.lib)
(defn shdILrnTypScrpt []
  ;; hope that ts people are not paying attention
  (prn "really?")
  false)

{...
  {:target :node-library
   :output-to "out/demo-library/lib.js"
   :exports {:hello demo.lib/shdILrnTypScrpt}}}
```

JUXT



;; Write NPM Packages

```
(defn ^:export shdILrnTypScrpt [] ...)
```

JUXT



;; Write Node Libs

```
$ cd out/demo-library
$ node
> var x = require('./lib');
undefined
> x.shdILrnTypScrpt()
really?
false
```

JUST



;; React using Hiccup

```
(defn Settings [username realname email]
  [:div#settings.margin-all-3
   [:h3 username]
   [:div realname]
   [:div email]
   [:button {:onClick #()} "Edit"]])
```

JUST



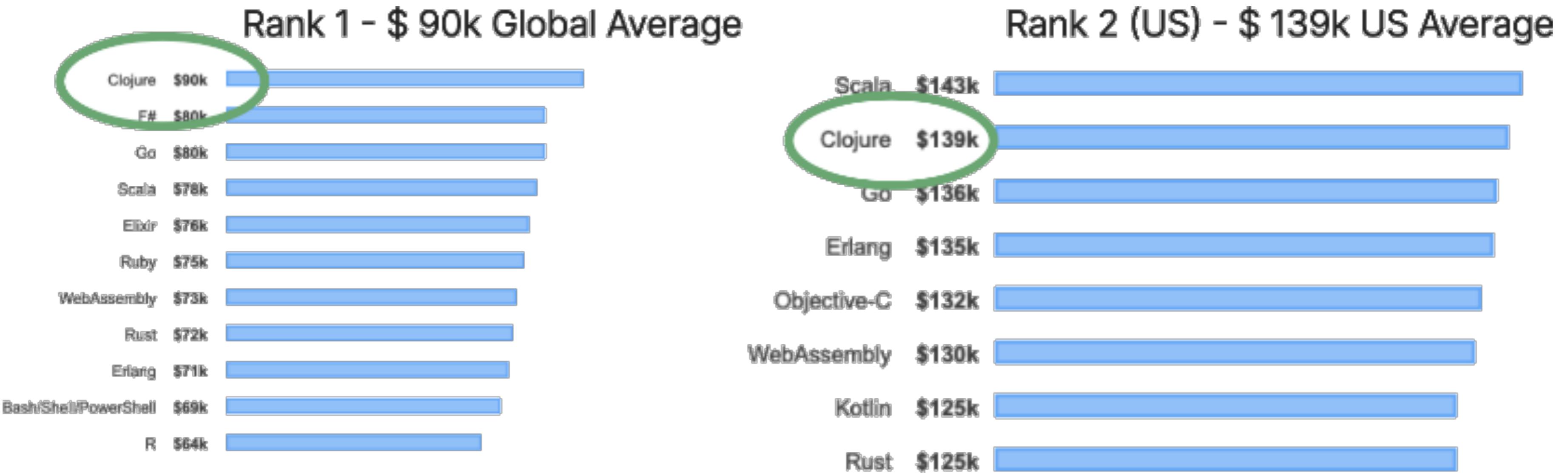
;; JS Equivalent

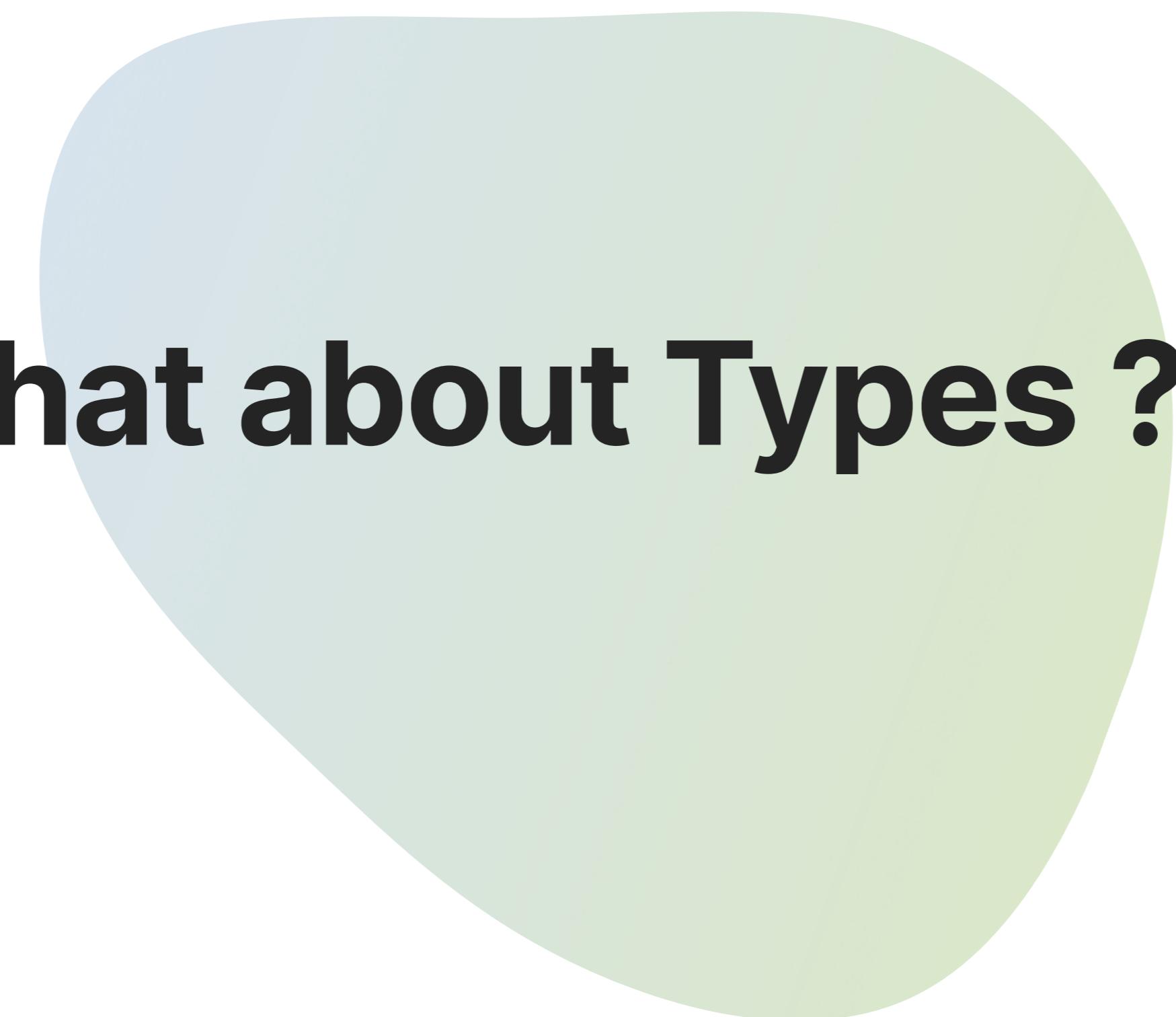
```
function Settings(username, realname, email) {
  return (<div id="settings" class="margin-all-3">
    <h3>{username}</h3>
    <div>{realname}</div>
    <div>{email}</div>
    <button onClick={() => {...}}>Edit</button>
  </div>);
}
```

JUST



Clojure Devs are best paid





What about Types ?

clojure.spec

- Optional type library
- Define types as predicates

;; Spec domain entities

```
(s/def ::first-name string?)  
(s/def ::last-name string?)  
(s/def ::phone-number (s/and int?  
                           #(<= % 9999999999)  
                           #(>= % 10000000) ))
```

```
(s/def ::person  
  (s/keys :req-un [::first-name ::last-name]  
          :opt-un [::phone-number]))
```

JUST



Spec Demo

State

```
:first-name  
:last-name  
:phone-number
```

Validation

```
(s/valid? ::person %)
```

API (Stub)

```
POST /person
```

Spec Benefits

- Optional opt in
- Generative testing inside source
- Just predicates
- Spec code (not just data)*

*<https://clojure.org/guides/spec>

;; Spec - going crazy

@danielcompton/defn-spec

```
(s/def ::int int?)  
  
(ds/defn my-inc :- ::int  
  [a :- ::int]  
  (+ a 1))
```

@jeaye/orchestra

```
(defn-spec my-inc int?  
  [a int?]  
  (+ a 1))
```

JUST





Going crazy demo

Building on strong foundations

- Simple & robust semantics
- Language level functionality as lib

Will stay in prod after 10 years ?

Getting started

Build Systems



Lein + Figwheel

<https://leinigen.org/>

<https://figwheel.org/>



Shadow CLJS

<https://shadow-cljs.org>

React Wrappers



Reagent

<https://reagent-project.github.io/>

*Reagent is what
React hopes to be
one day*

— Luke Whorton
(<https://medium.com/@lwhorton/>)

React Wrappers



Reagent

<https://reagent-project.github.io/>

hx

hx

<https://github.com/Lokeh/hx>

People



Rich Hickey
Creator of Clojure



Stuart Halloway
Core Team Member



Alex Miller
Core Team Member



Stuart Sierra
Component; Reloaded



Malcolm Sparks
Yada/Bidi/Tick



James Reeves
Integrant/ Hiccup/ Medley



Bruce Hauman
Figwheel



Bozhidar Batsov
nrepl

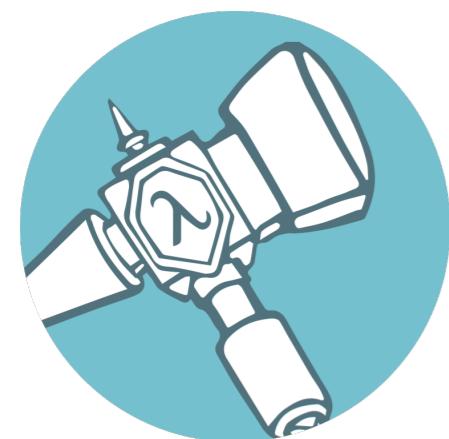


David Nolen
ClojureScript



David Miller
ClojureCLR

Blogs & Docs



bravecjure.com

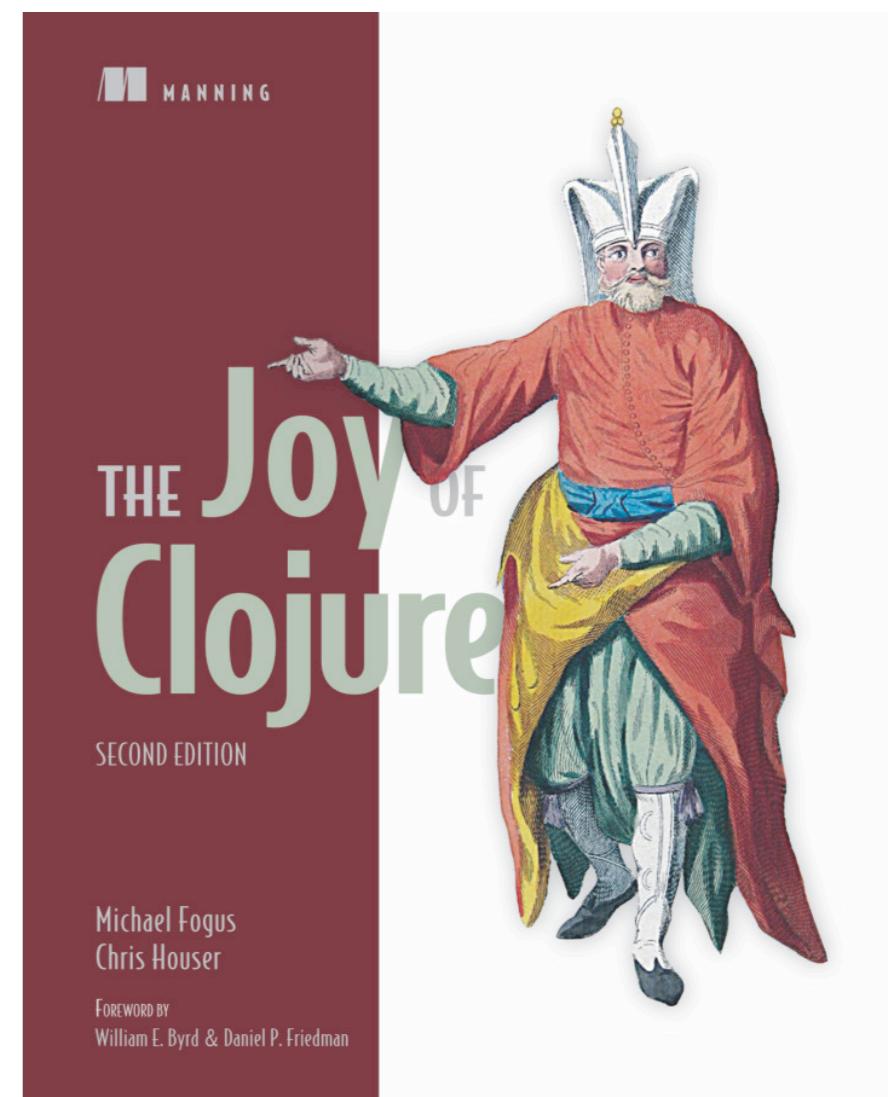
JUXT
juxt.pro/blog

PurelyFunctional.tv
purelyfunctional.tv



CLOJURE KOANS

clojurekoans.com
clojurescriptkoans.com



The Joy of Clojure;
By Fogus & Houser

JUXT
 Krim
Labs



Thank You

Shivek Khurana

Clojure Consultant at **JUXT Inc**

Independant Consultant at **Krim Labs**

Around the web

medium.com/@shivekkhurana

github.com/shivekkhurana

twitter.com/shivek_khurana

