

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

Shivek Khurana

JUXT



About me

- Web apps since 2008
- Clojure for since 2017
- Consultant at Juxt Inc.
- Enterprise Apps
- Currently building a system for Vodafone
- 12m+ users

JUXT



Built an LMS from 2012 to 2016

Long projects == Living with your past mistakes

JUXT



Followed all best practices

Changed the build system thrice



JUST



JS ecosystem was maturing in 2012

Is it mature now?



PARCEL



rollup.js

JUXT

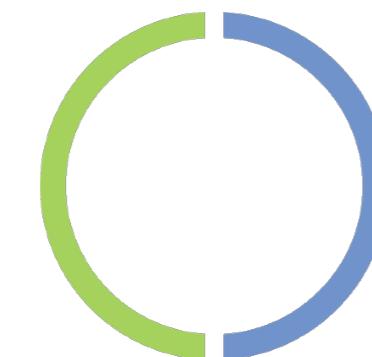


The JS community is blessed

But raw JS is not

JUXT





Clojure/Script

LISP Functional Immutable Homoiconic
Thread Safe Dynamically typed Hosted

Joy to use

JUXT



Clojure/Script Crash Course

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

— Nolan Bushnell (Founder, Atari)

JUXT



;; Native Data Types

;; Character

\a
\b
\newline
\tab

;; String

“hello”
“Line 1\nline2”

;; Keywords

:keyword

;; Regex

#"[a-z]+"

;; Bool

true, false

;; Nothing

nil

;; Number

255 ;; int
012 ;; octal
0xff ;; hex
2r1111 ;; radix
3.14 ;; standard float
1.35e-12 ;; scientific float
22/7 ;; fraction

JUST



;; List

;; function calls are written as a list
(max 1 2)

;; to not call the first element
(list 1 3 5)
'(1 3 5)
(quote (1 3 5))

JUXT



;; Vector, Sets & Maps

[1 2]

(vector 1 2)

#{1 3}

(set 1 2)

{:name "Shivek"
:age 24}

JUX^T



;; LISP

(1 2 3)

(+ 1 2 3)

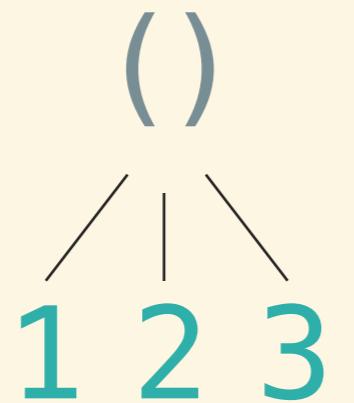
(defn sum [a b]
(+ a b))

JUXT

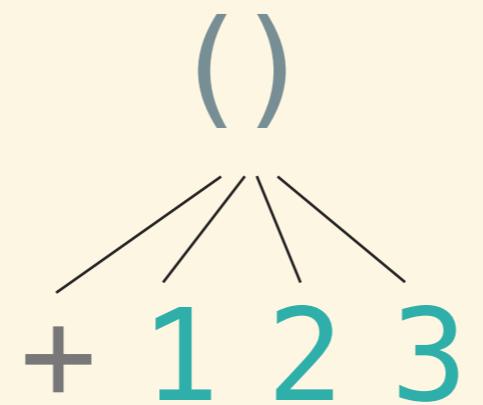


;; LISP

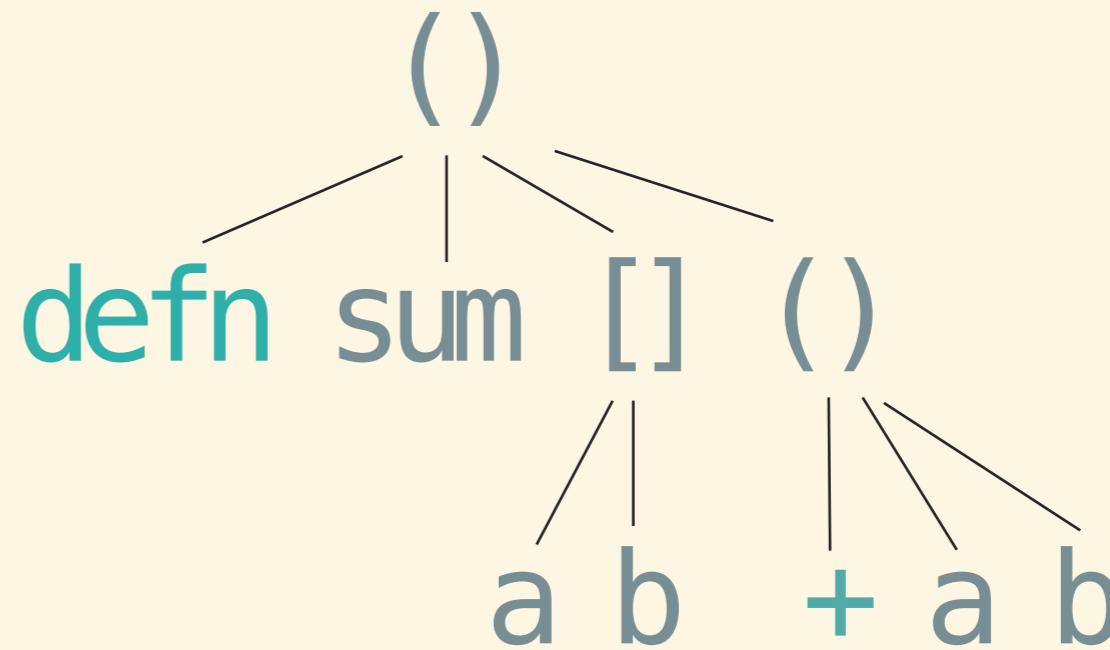
(1 2 3)



(+ 1 2 3)



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



;; Functional

```
(defonce person
  {:_name "Vienaa"
   :age 24})
```

```
(:name person)
(:age person)
```

```
class Person {
  init(name, age) {...}
  getName() => {...}
  getAge() => {...}
}
```

```
const v = Person("Viena", 24)
v.getName()
v.getAge()
```

JUXT



;; Immutable

```
(defonce count 0)
(inc count) ;; => 1
count ;; => 0
```

JUXT



;; Atoms
;; thread safe/ mutable constructs

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

JUXT



;; Homoiconic
;; Code is data

(1 2 4)

{:name "Shivek" :age 24}

[2 1]

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

JUX^T



Compiles to JavaScript using Google Closure Compiler



JUXT



Google Closure Compiler

- In prod for 10+ years
- Compiles JS to better JS
- Parse > Analyse > Remove Dead Code
- Minify & Rewrite what's left

JUXT



Google Closure Library

- Google's Standard Lib for JS
- Cross Platform

JUXT



;; Interop with JS

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

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

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

```
new FormData()
```

```
(.-height js/window)
```

```
window.height
```

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

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

JUXT



;; Consuming NPM Packages

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

JUXT



;; Write Node Libs

```
(ns demo.lib)
(defn hello []
  (prn "hello")
  "hello")
```

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

JUXT



;; Write Node Libs

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

JUXT



;; Write NPM Packages

```
(defn ^:export my-function [] ...)
```

JUXT



;; HTML/React Components ;; using Hiccup

```
(defn Settings [username realname email]
  [:div {:class "pa2 ma3"}
    [:h3 username]
    [:div realname]
    [:div email]
    [:button {:onClick #()} "Edit"]])
```

JUXT



;; JS Equivalent

```
function Settings(username, realname, email) {  
  return (<div class="pa2 ma3">  
    <h3>{username}</h3>  
    <div>{realname}</div>  
    <div>{email}</div>  
    <button onClick={() => {...}}>Edit</button>  
  </div>);  
}
```

JUXT



Switching stacks

Weird Syntax
Immutable
No classes
Small Community
Talent/ Training

JUXT



Switching stacks

Weird Syntax
Immutable
No classes
Small community
Talent/ Training

Brackets become invisible
Immutable ❤️
Functional ❤️
Small but brilliant community
Things are changing

JUXT



Differences from JS

Built-in objects

Classes

Functions

Prototypes

Mutability

Statements (Syntax)

JUXT



Differences from JS

Built-in Objects
Classes

Functions
Prototypes

Mutability
Statements (Syntax)

Built in Data Structures



Functions



Immutability



JUXT



REPL

Developing inside the Runtime

JUXT



Observation Driven Dev

Text Editor

Code



Transpile +
Hot Swap

Runtime (Browser)

Updated App

Perform Actions
Observe

Code

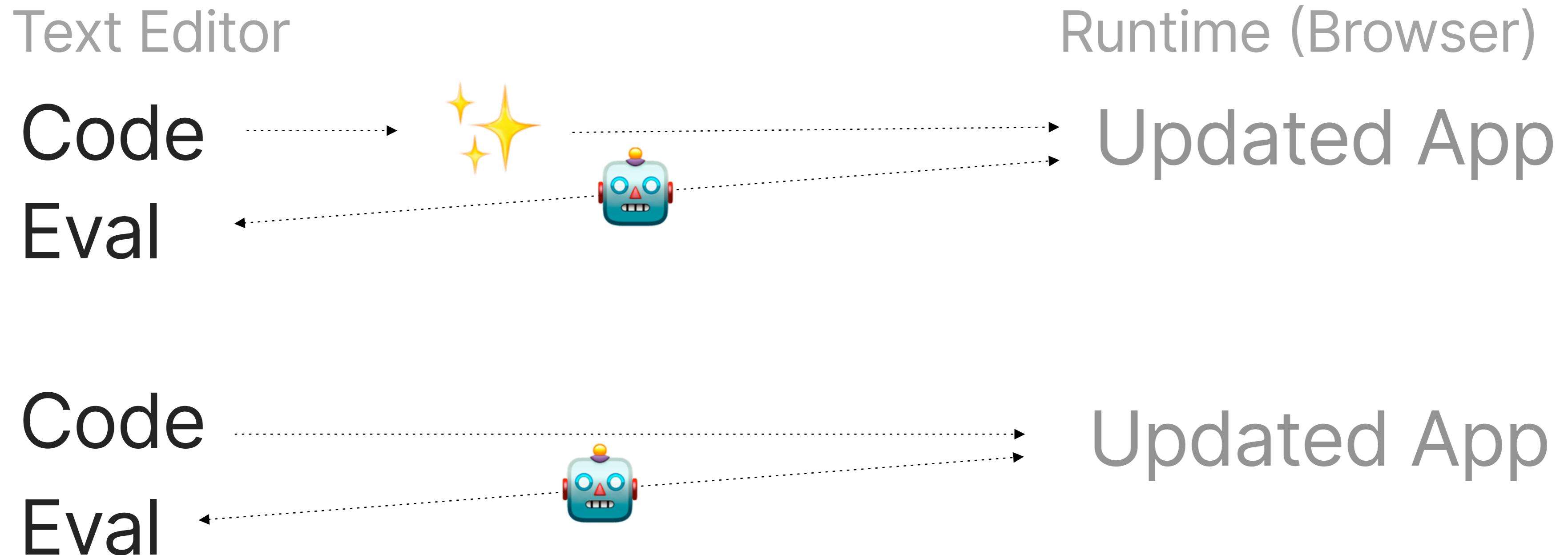
Updated App

Perform Actions
Observe

JUXT



REPL Driven Dev



JUXT



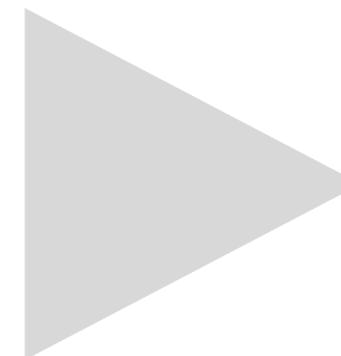
REPL Demo - Tic Tac Toe

State

Lattice

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

Turn :o



React

:o's turn

x	-	-
-	x	-
-	-	o

Reset

JUXT



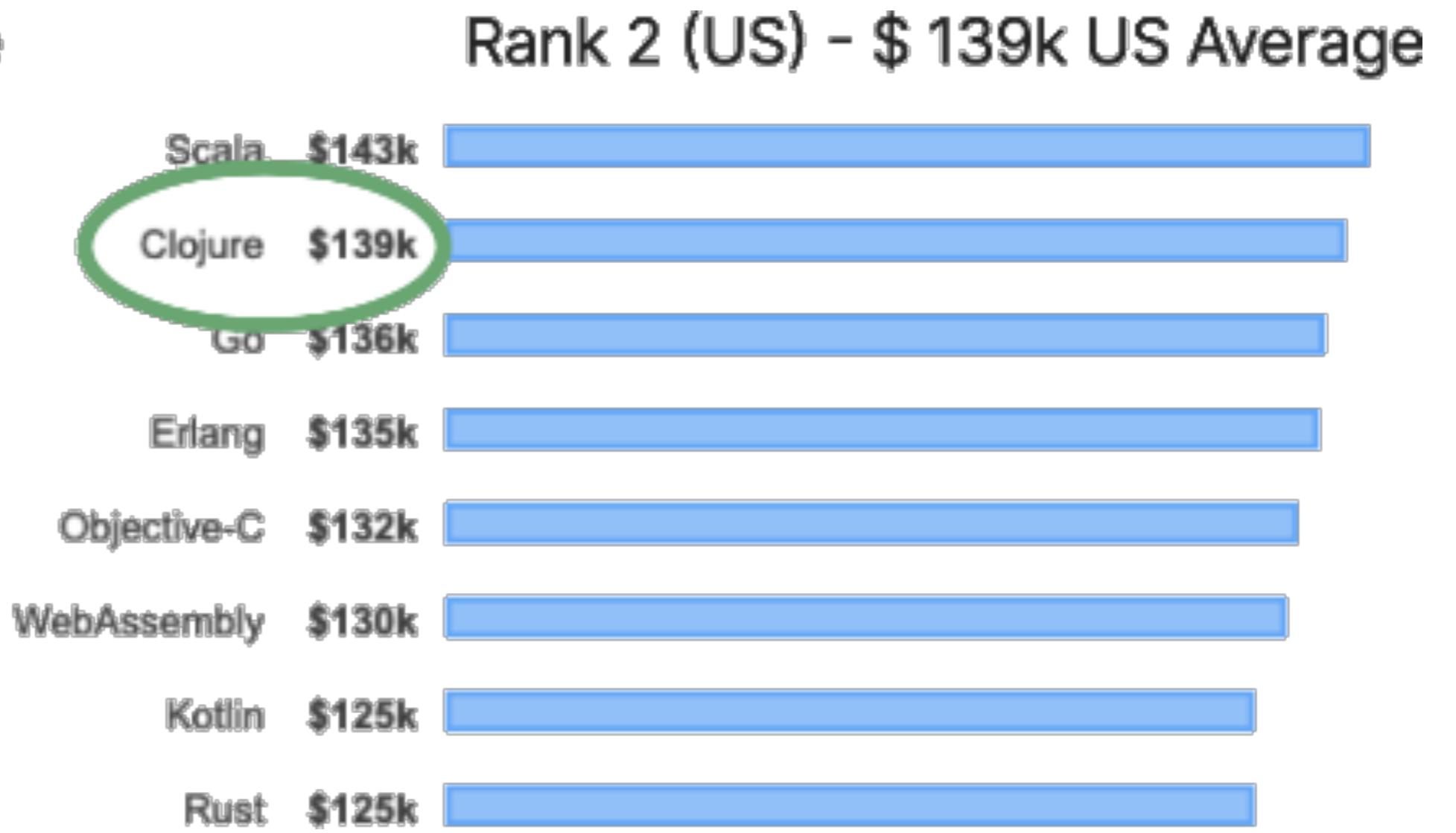
With REPL - Develop only a function

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

JUXT



Clojure Devs are best paid



core.spec

- Optional type library
- Define types as predicates
- Assert/ Validate/ Generate/ Conform

JUXT



Spec demo

JUXT



Build Systems



Lein + Figwheel

<https://leinigen.org/>

<https://figwheel.org/>



Shadow CLJS

<https://shadow-cljs.org>

JUXT



React Wrappers



Reagent

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

*Reagent is what
React hopes to be
one day*

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

JUXT



React Wrappers



Reagent

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

hx

hx

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

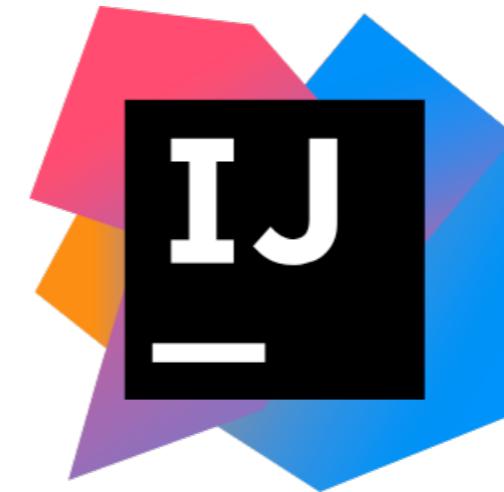
JUXT



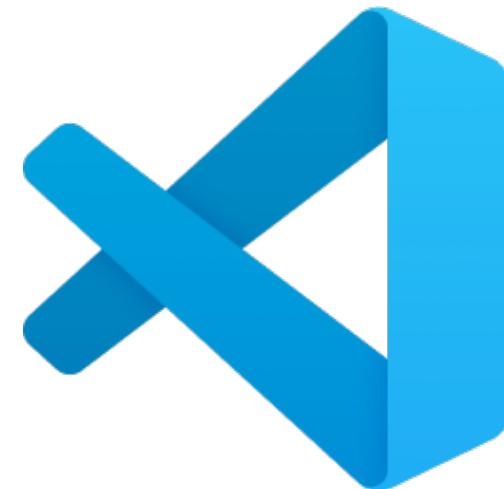
Editors



(cider[])



Cursive



(()) *Calva*



Fireplace (?)

JUXT



People

JUXT





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

JUXT

