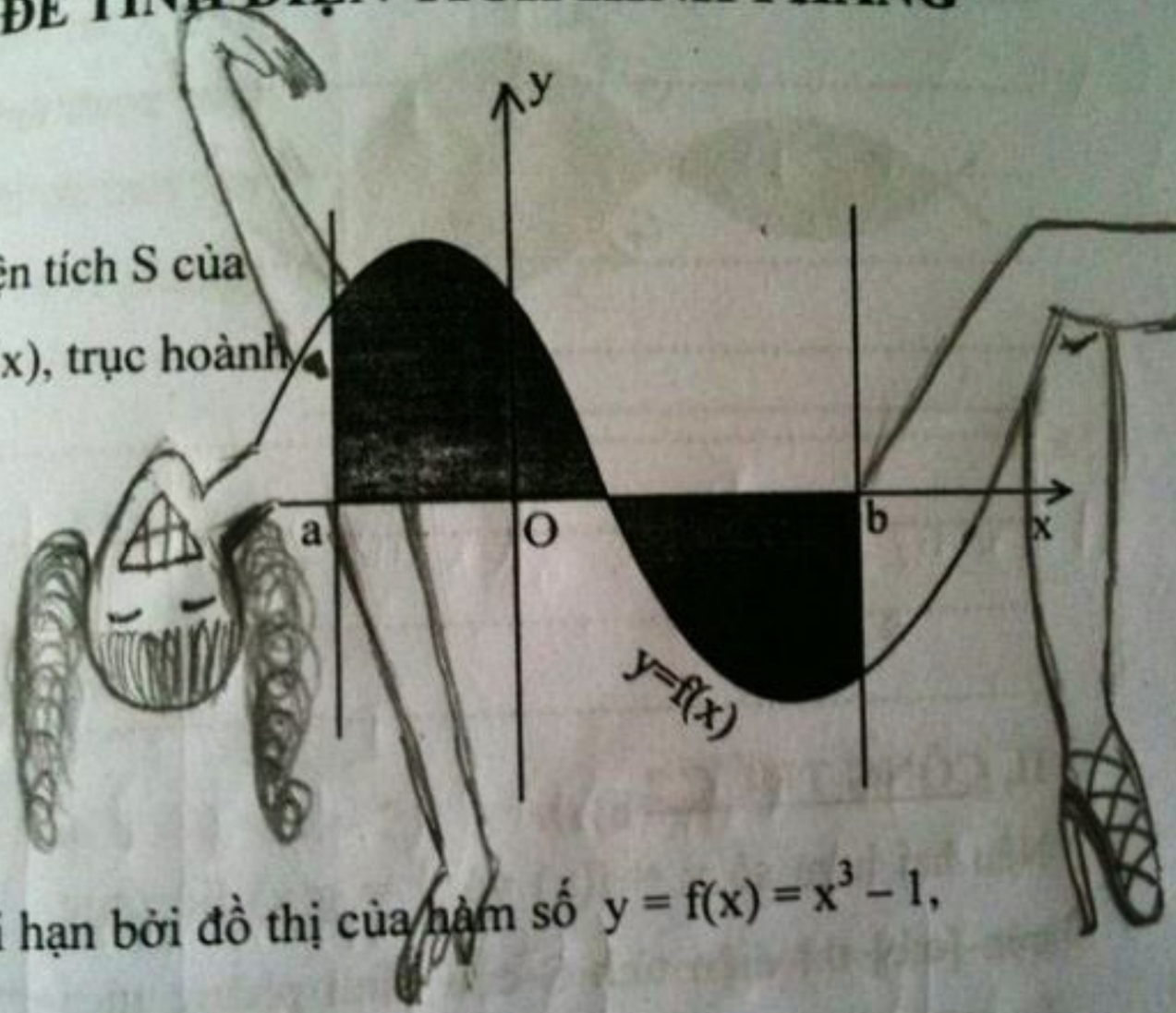
H PHÂN ĐỂ TÍNH DIỆN TÍCH HÌNH PHẲNG

[a;b] thì diện tích S của

m số y = f(x), trục hoành



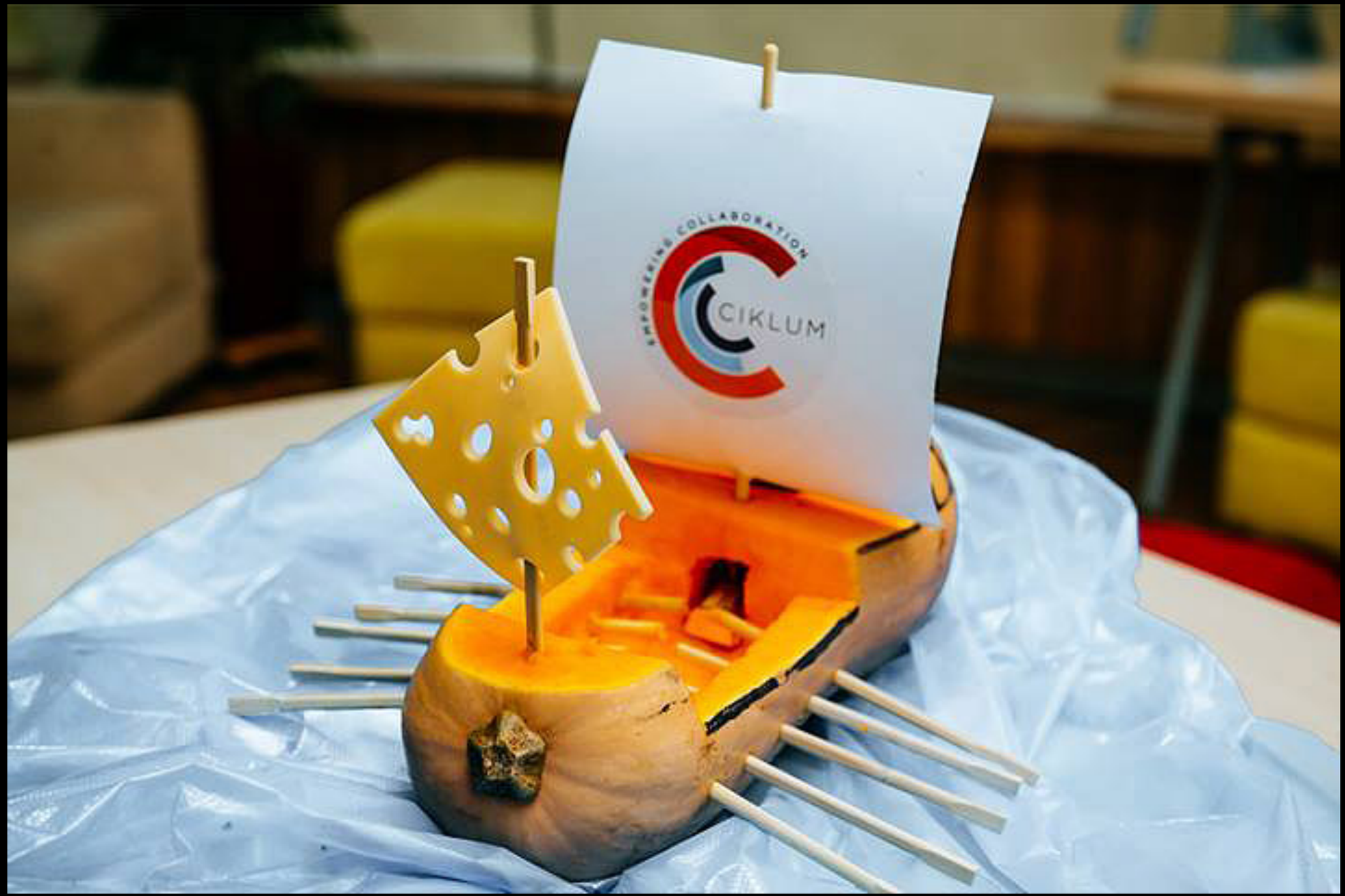
hẳng giới hạn bởi đồ thị của hàm số $y = f(x) = x^3 - 1$,

hoành.

Clojure-Gopher
CSP with idiomatic Clojure

Александр Хохлов

@apoint

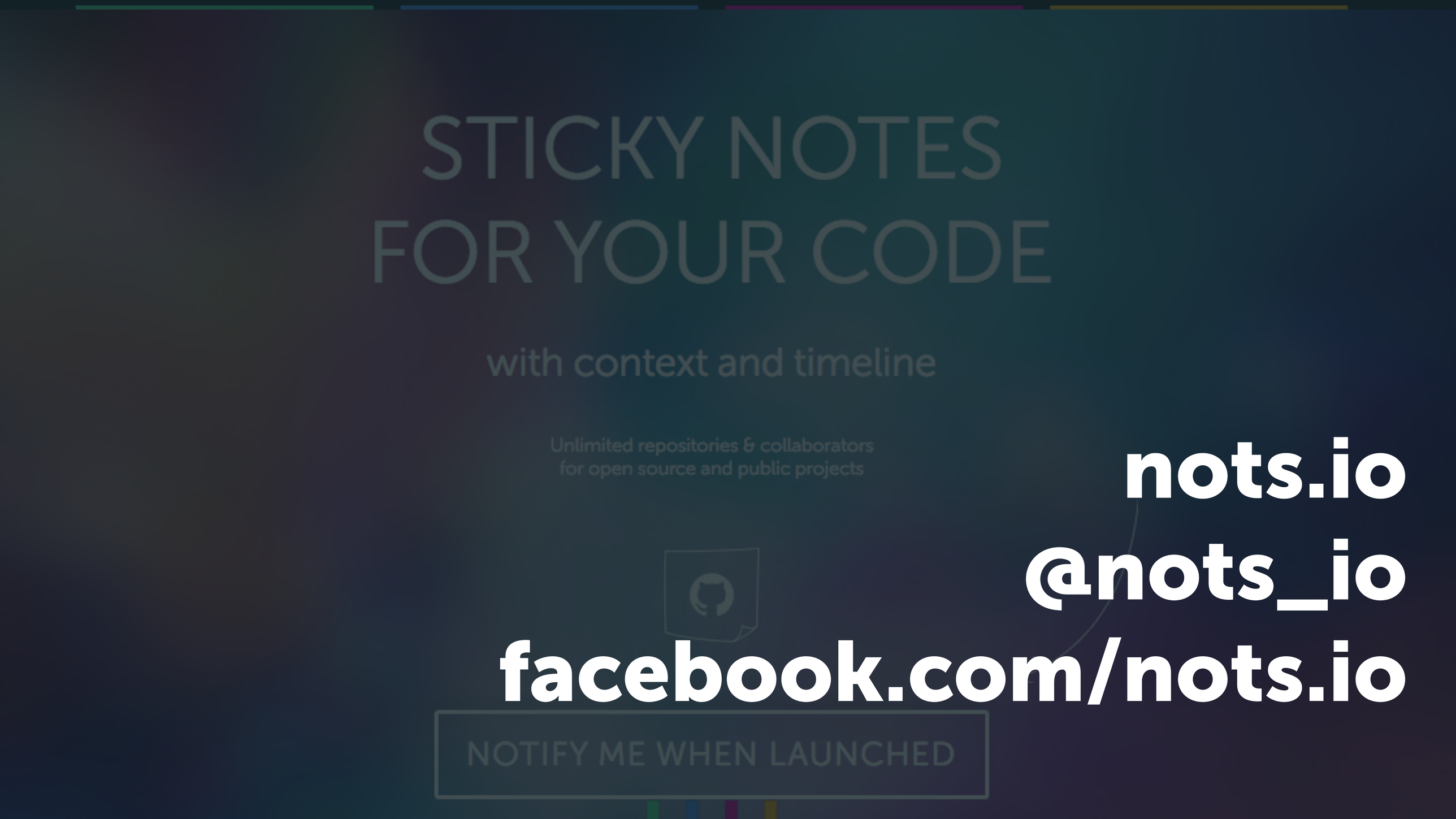

EMPOWERING COLLABORATION
CIKLUM

STICKY NOTES FOR YOUR CODE

with context and timeline

Unlimited repositories & collaborators
for open source and public projects

NOTIFY ME WHEN LAUNCHED



nots.io

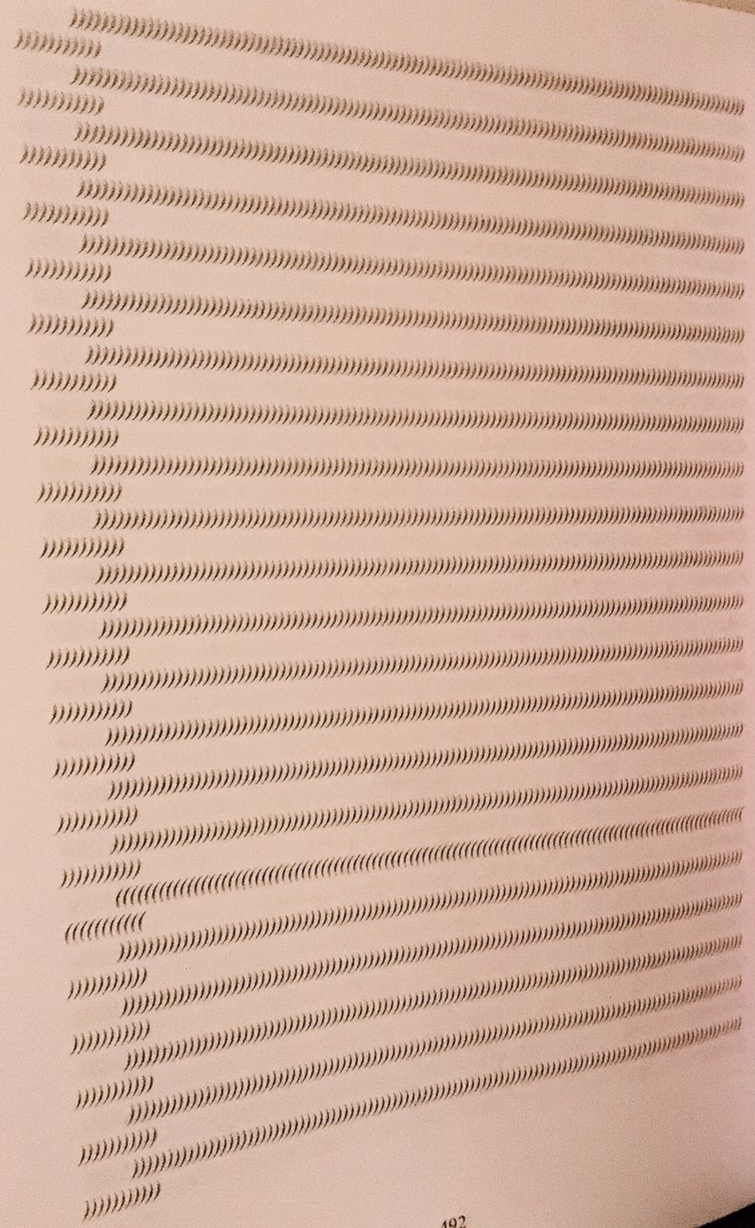
@nots_io

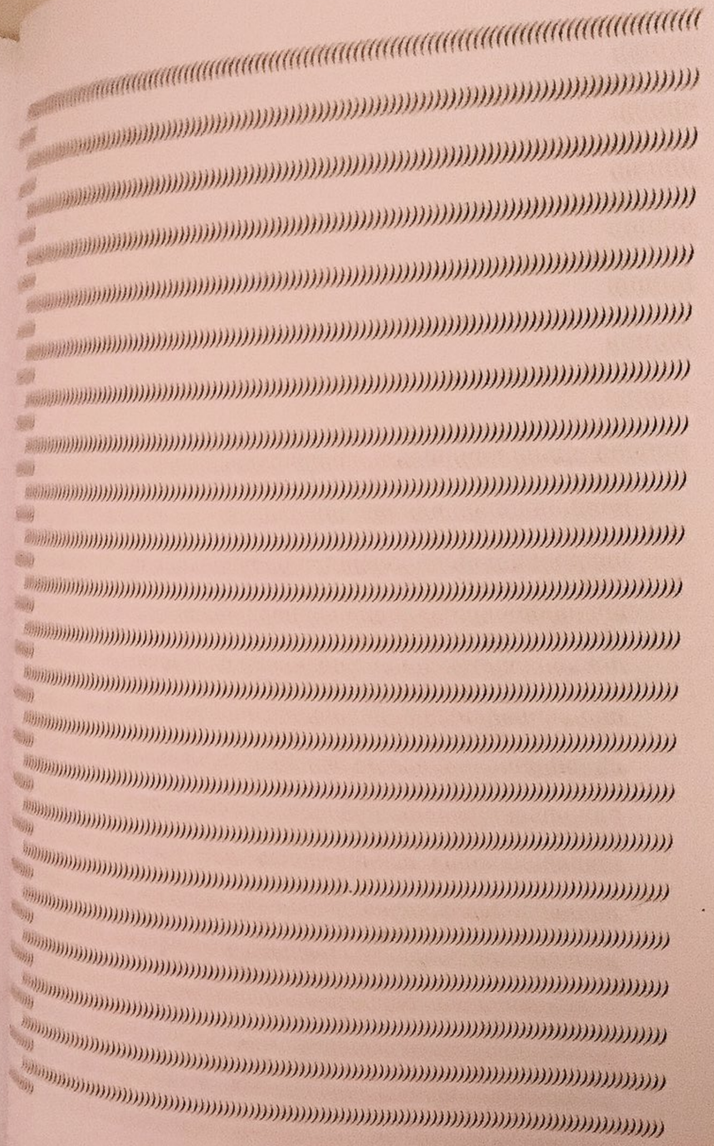
facebook.com/nots.io

Что ещё за Clojure?

Lisp для JVM

(и не только)





Excerpt from a typical LISP program:

```
) ) ) ) ) ) ) ) ) ) ) ) )
```

Excerpt from a typical Node.js program:

```
}); }); }); }); }); });
```

Язык общего назначения

Функциональный

Со строгой динамической типизацией

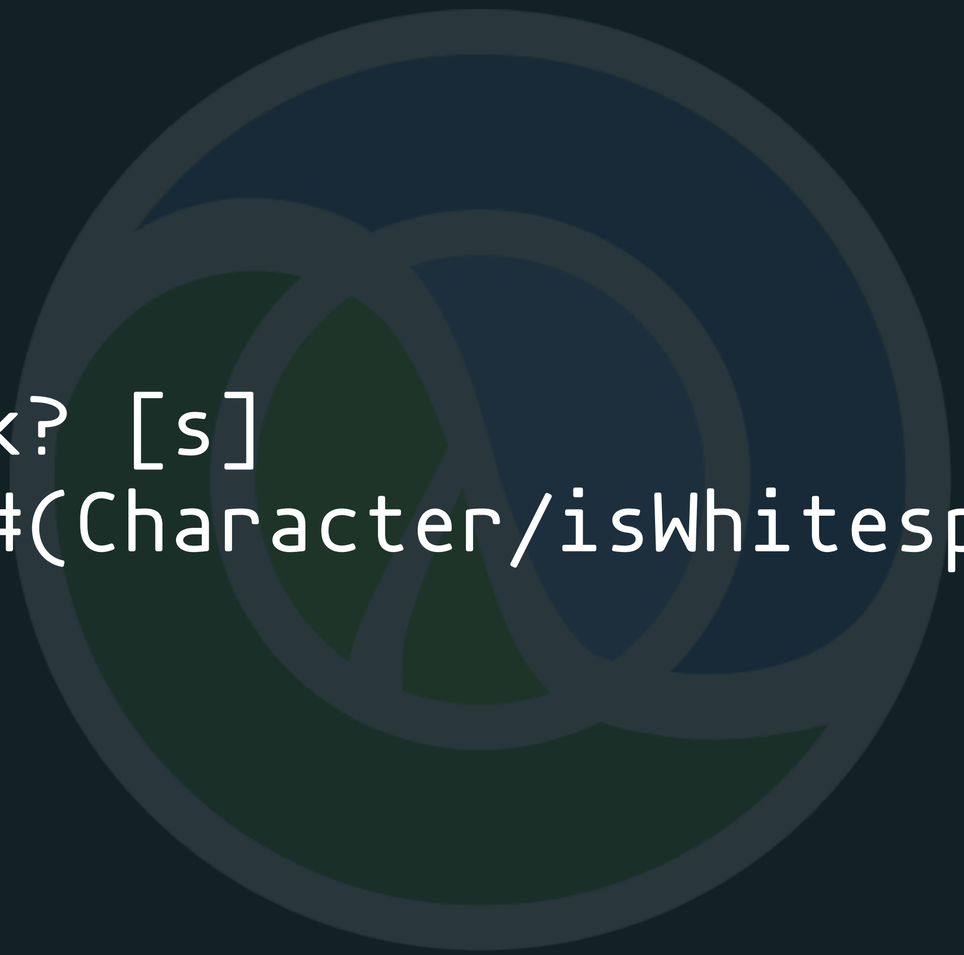
Прагматичный

Компактный синтаксис

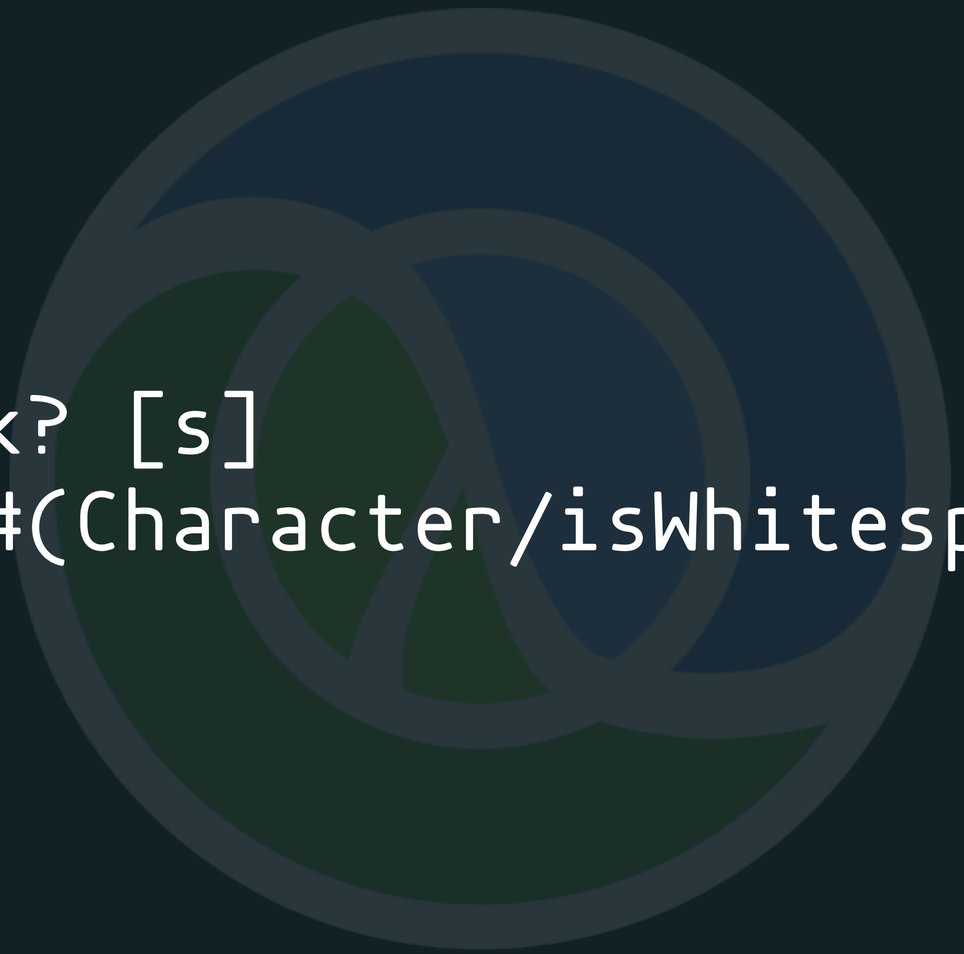
Полиморфизм без ООП

Кодогенерация / макросы

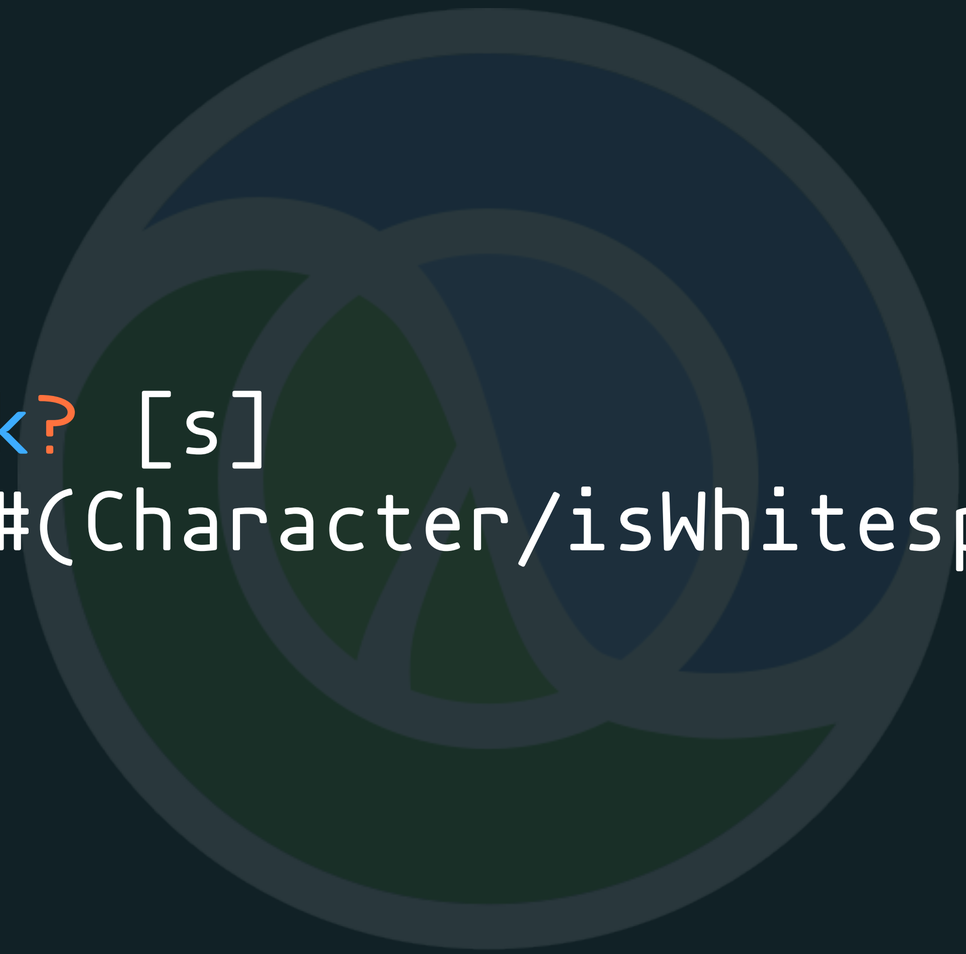

Иммутабельные персистентные структуры данных
Interop с Java/JavaScript
REPL-driven development



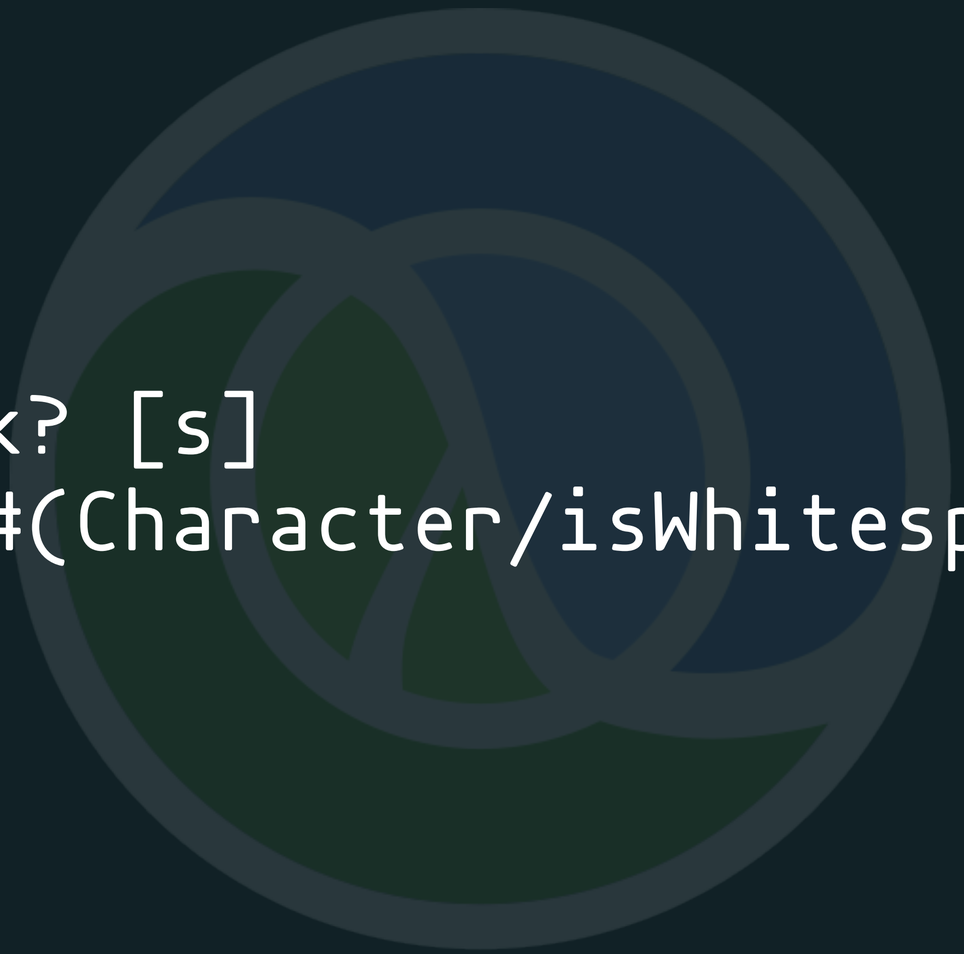
```
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```



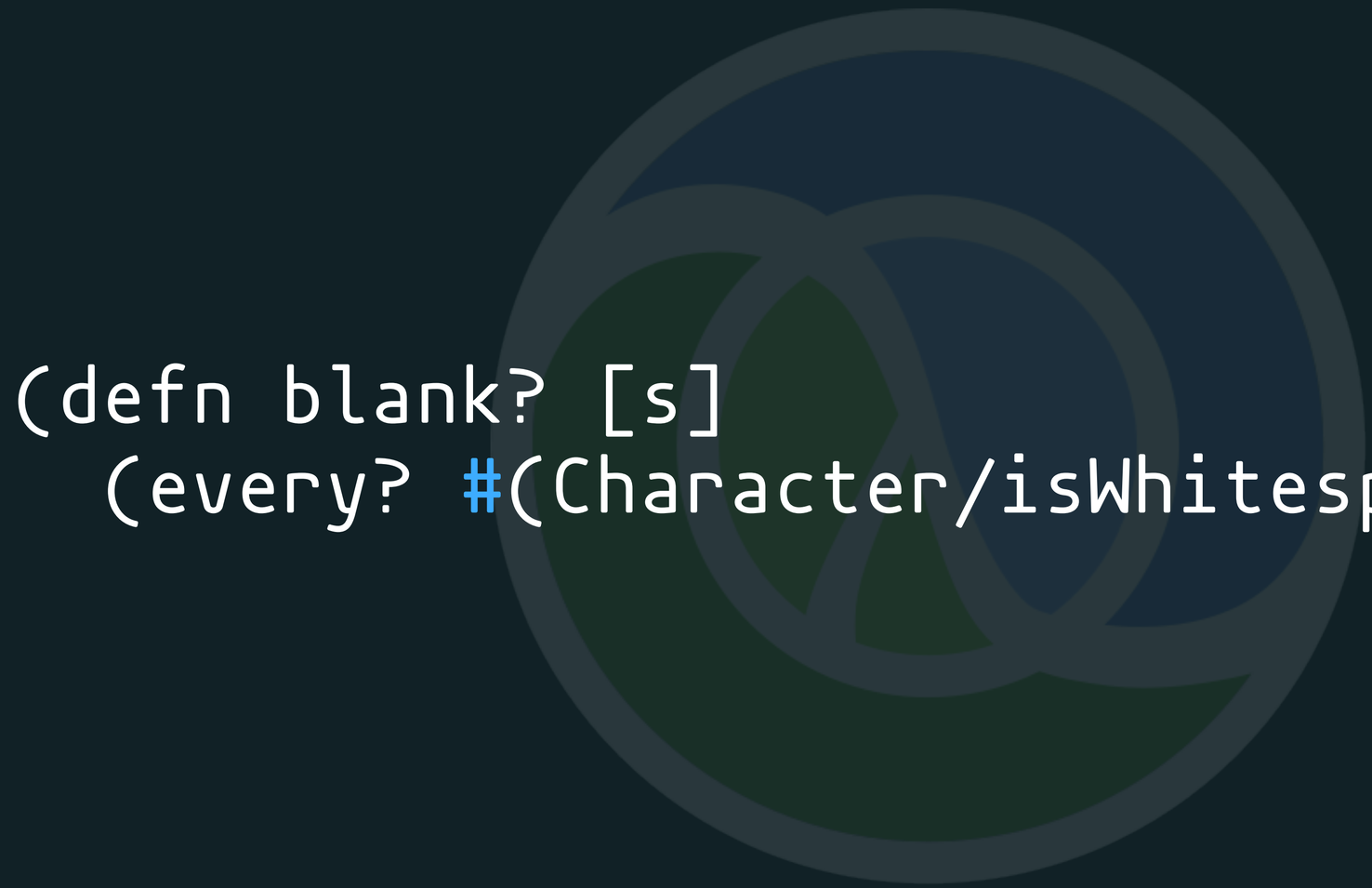
```
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```



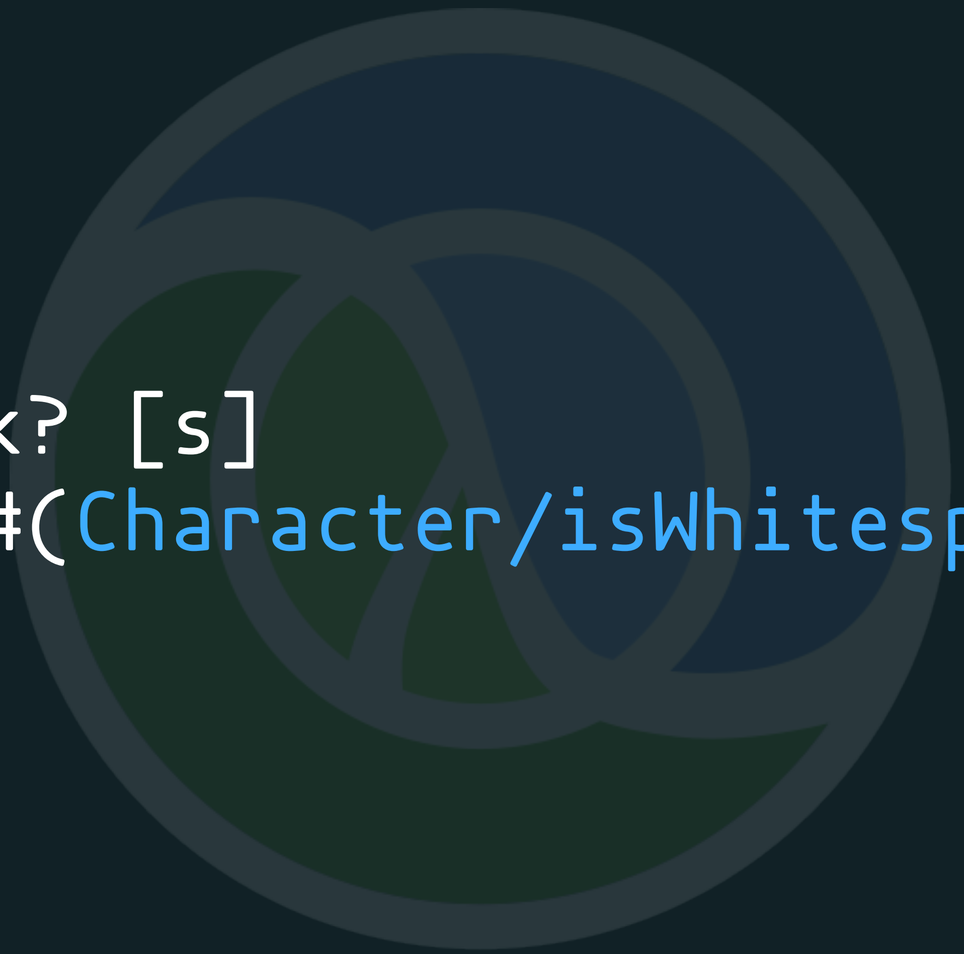
```clojure
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```



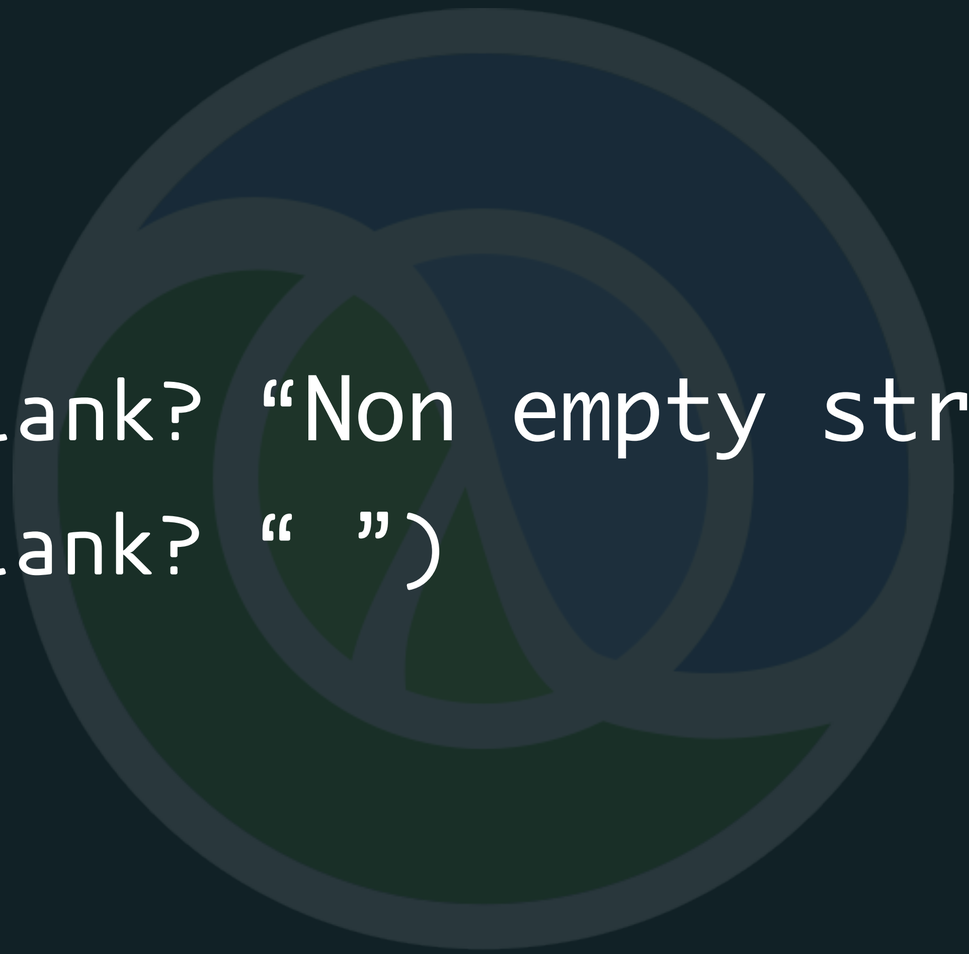
```
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```



```clojure
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```



```
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```



```
(blank? “Non empty string”)
(blank? “ ”)
```


[1 2 3]
(1 2 3)
{:a 1 :b 2}
(def set #{:x :y :z})

'(1 2 3 4) "Look Ma!
no commas"

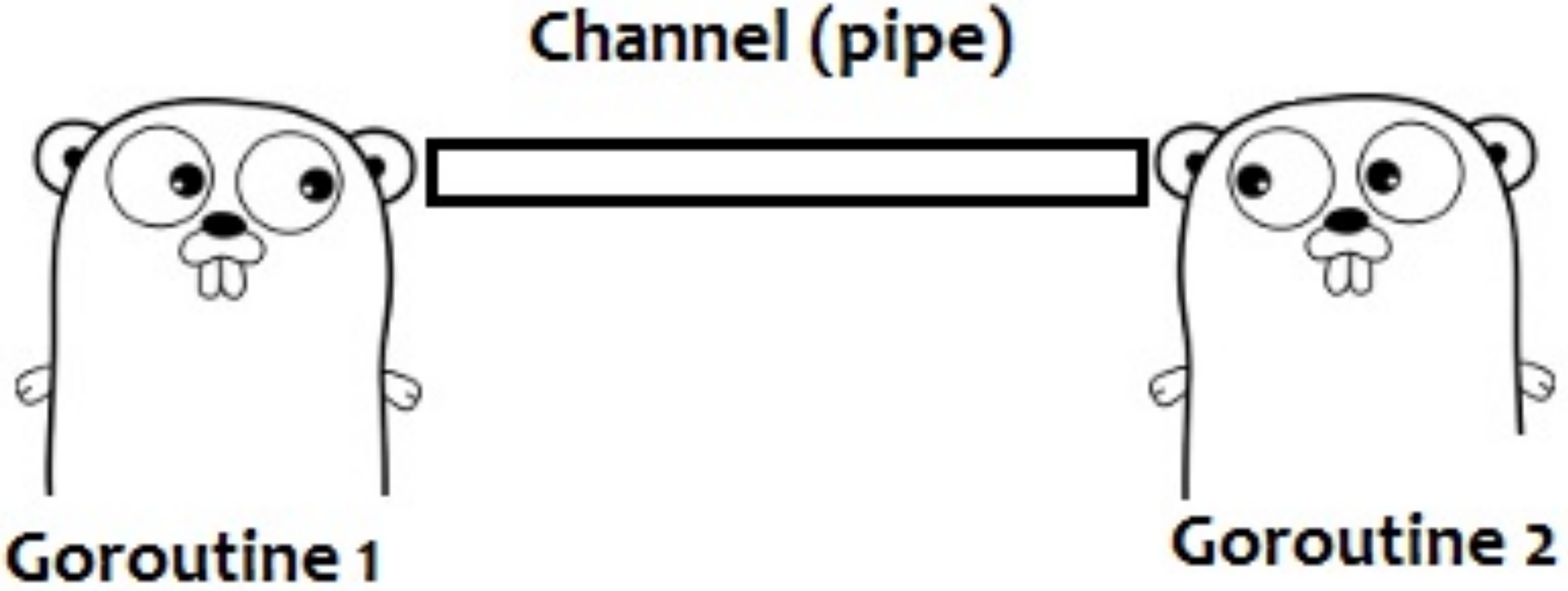
(- 8 2 2)

("one" "two" "three") (+ 1 1)

[:foo :bar] {:key1 "one" :key2 2}

2016-01-17


core.async


Channel (pipe)
Goroutine 1
Goroutine 2

```
(ns meetup.core
  (:require [clojure.core.async :refer :all :as async]))
```

```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (async/chan))
```

```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (chan))
(def c2 (chan 10))
```

```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (chan))
(def c2 (chan 10))
(def c3 (chan (dropping-buffer 1))) ;dropping newest
```

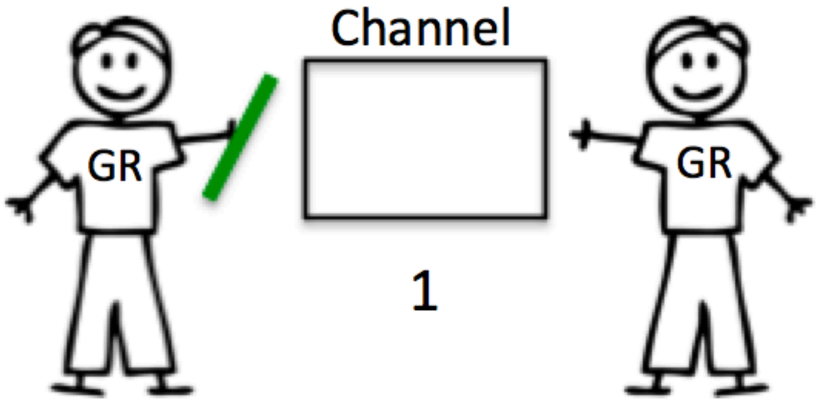
```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (chan))
(def c2 (chan 10))
(def c3 (chan (dropping-buffer 1))) ;dropping newest
(def c3 (chan (sliding-buffer 1))) ;dropping oldest
```

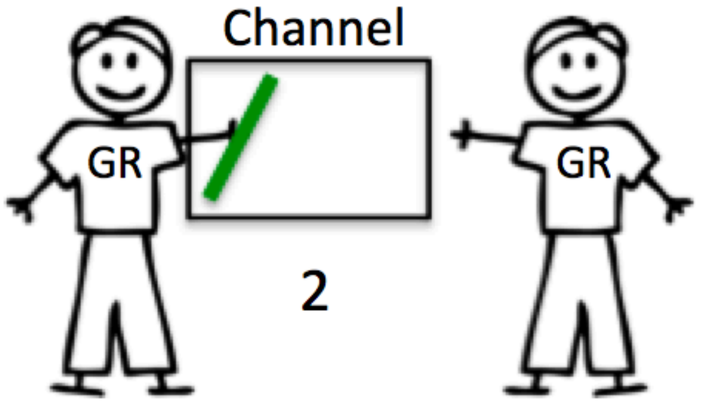
```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (chan))

(take! c #(println %))
(put!  c 42)
```

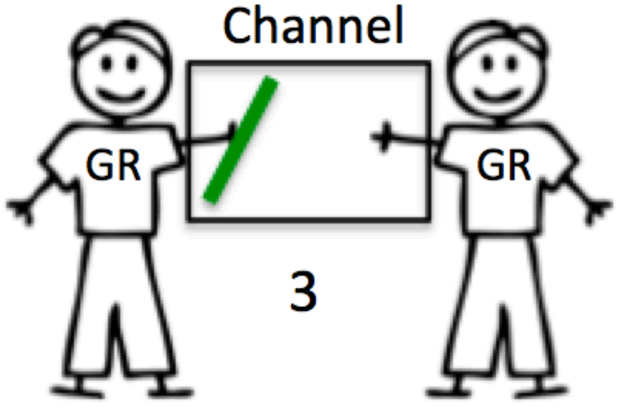
```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (chan 1))

(<!! c)
(>!! c 42)
;; used for thread coordination mainly
```

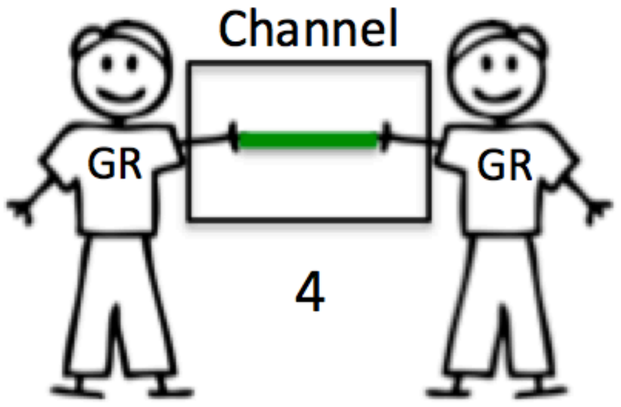
```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (chan 1))

(thread (println (<!! c)))
(>!! c 42)
;; used for thread coordination mainly
```

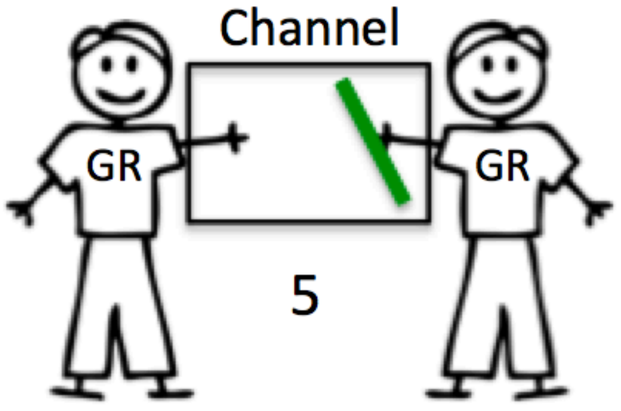
```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def c  (chan))

(go (println (<! c)))
(go (>! c “Hello”))
```

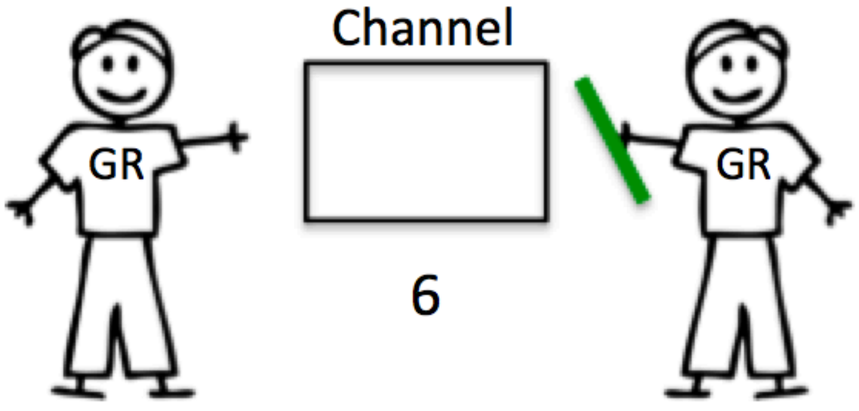

Channel
GR
GR
1


Channel
GR
GR
2


Channel
GR
GR
3

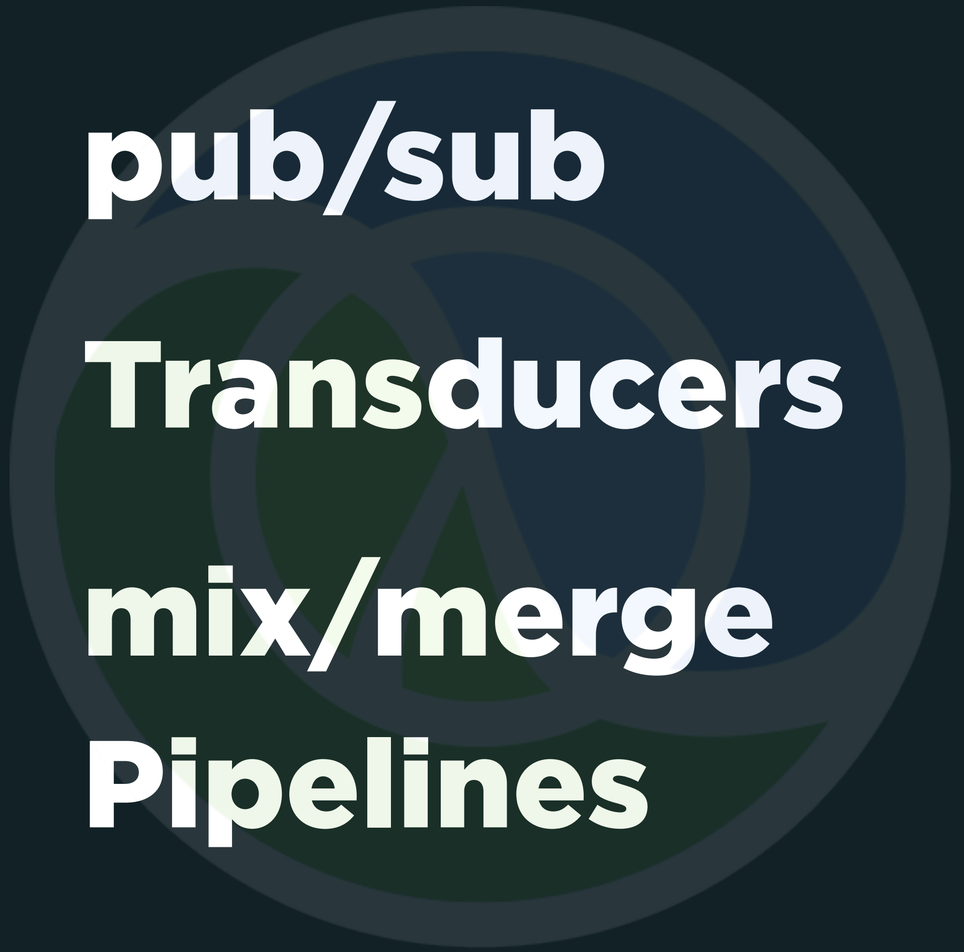

Channel
GR
GR
4

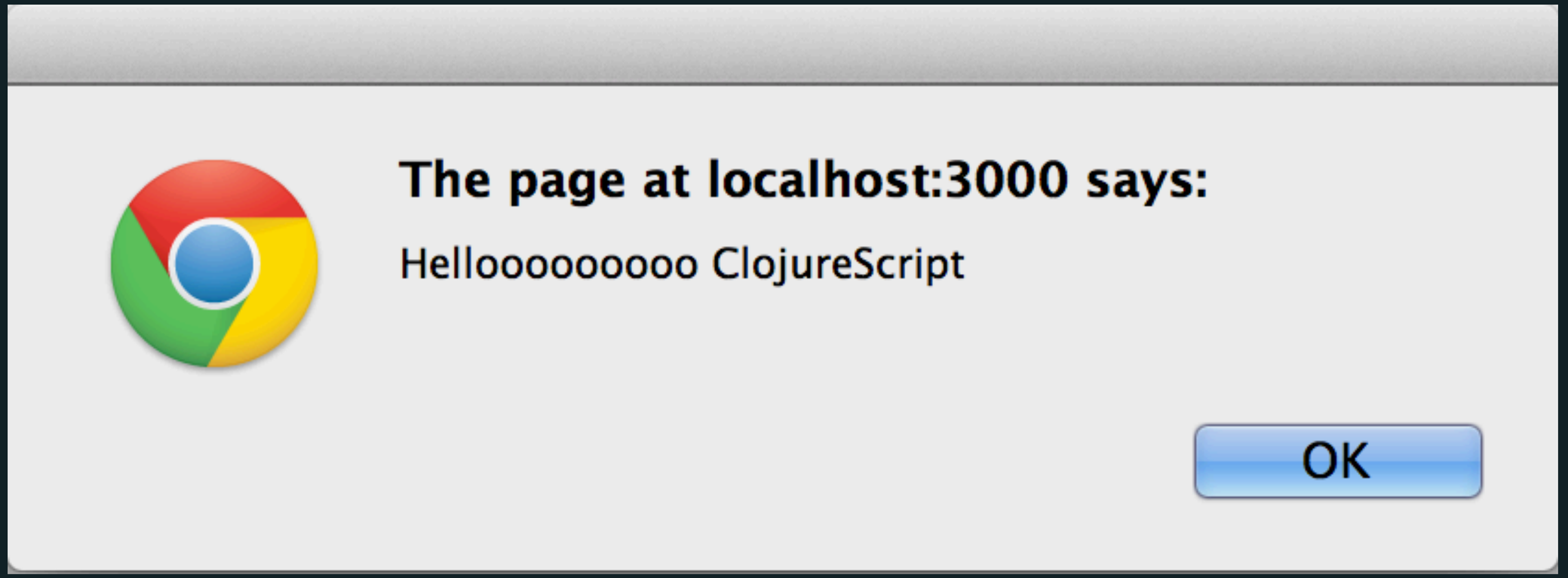

Channel
GR
GR
5


Channel
GR
GR
6

```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def a  (chan))
(def b  (chan))

(go (alts! [a b])) ;; returns [value chan]
```

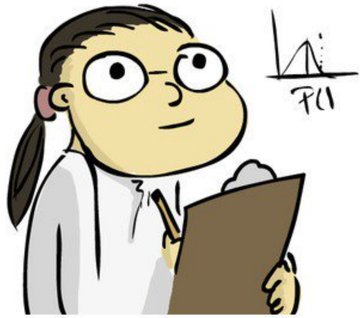
```
(ns meetup.core
   (:require [clojure.core.async :refer :all :as async]))

(def a  (chan))
(def b  (chan))

(go (alts! [a b (timeout 1000)]))
;; returns [value chan] or closes after 1s
```


pub/sub
Transducers
mix/merge
Pipelines


The page at localhost:3000 says:
Hellooooooooo ClojureScript
OK

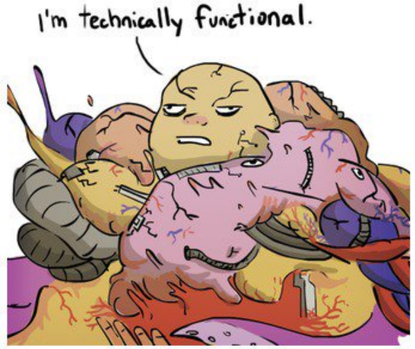
Python

R



Java

JavaScript



PHP

Haskell

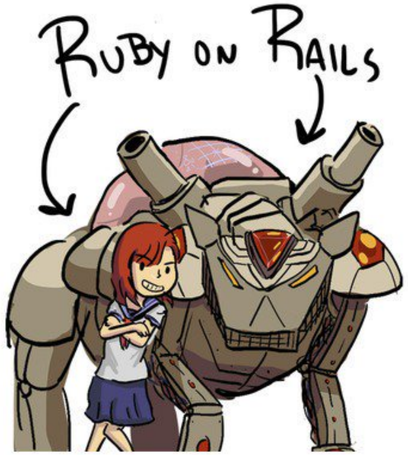
Perl

C

C++

C#

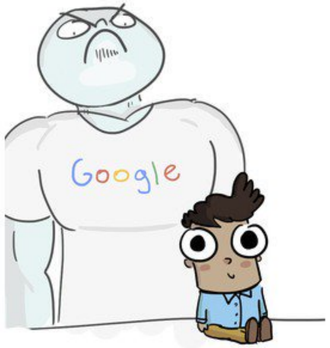
Ruby on Rails



Assembly

Erlang

Go



Brainfuck

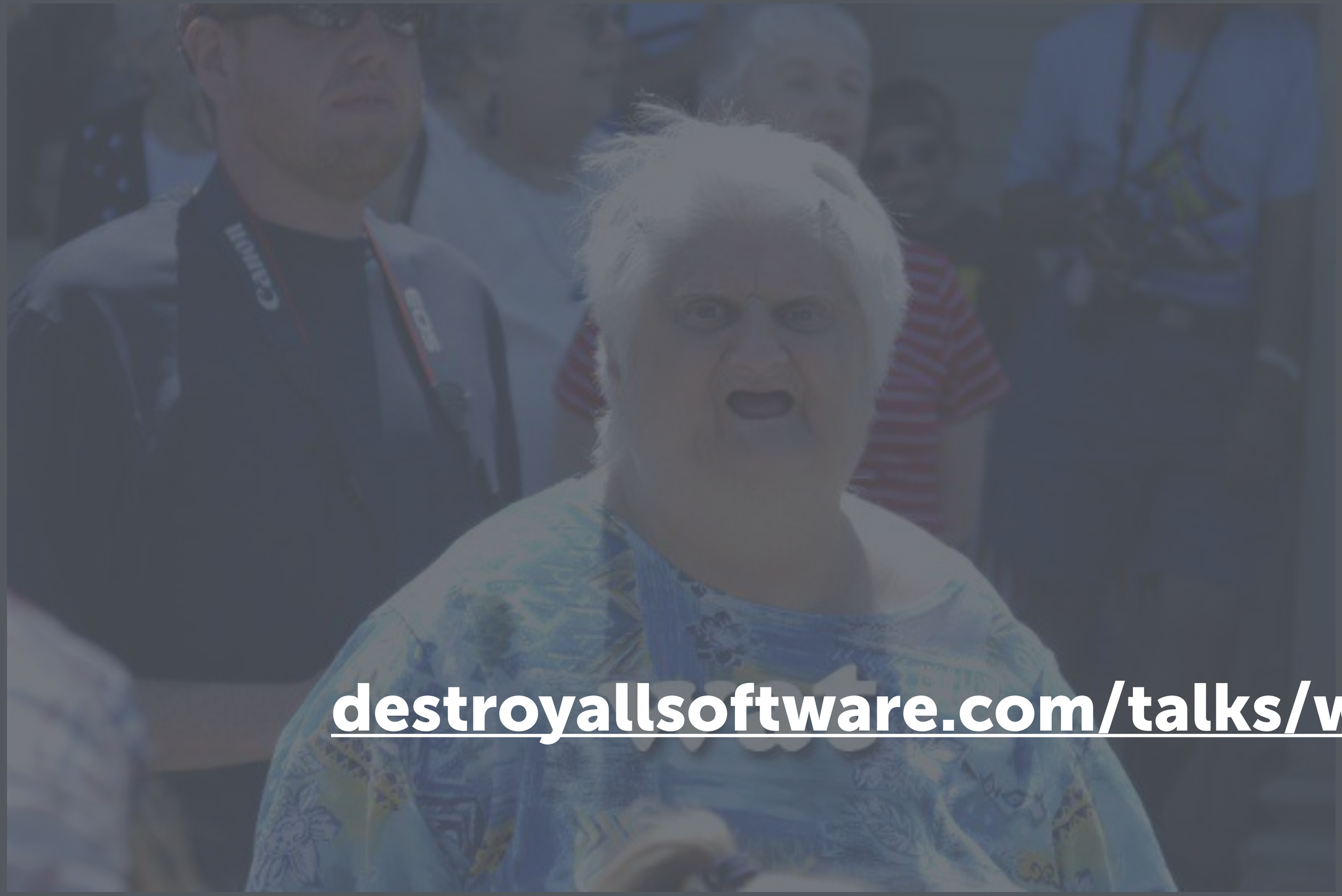
D.marino

leftoversalad.tumblr.com

Rust

JAVASCRIPT


wat



destroyallsoftware.com/talks/wat

ClojureScript

Состояние

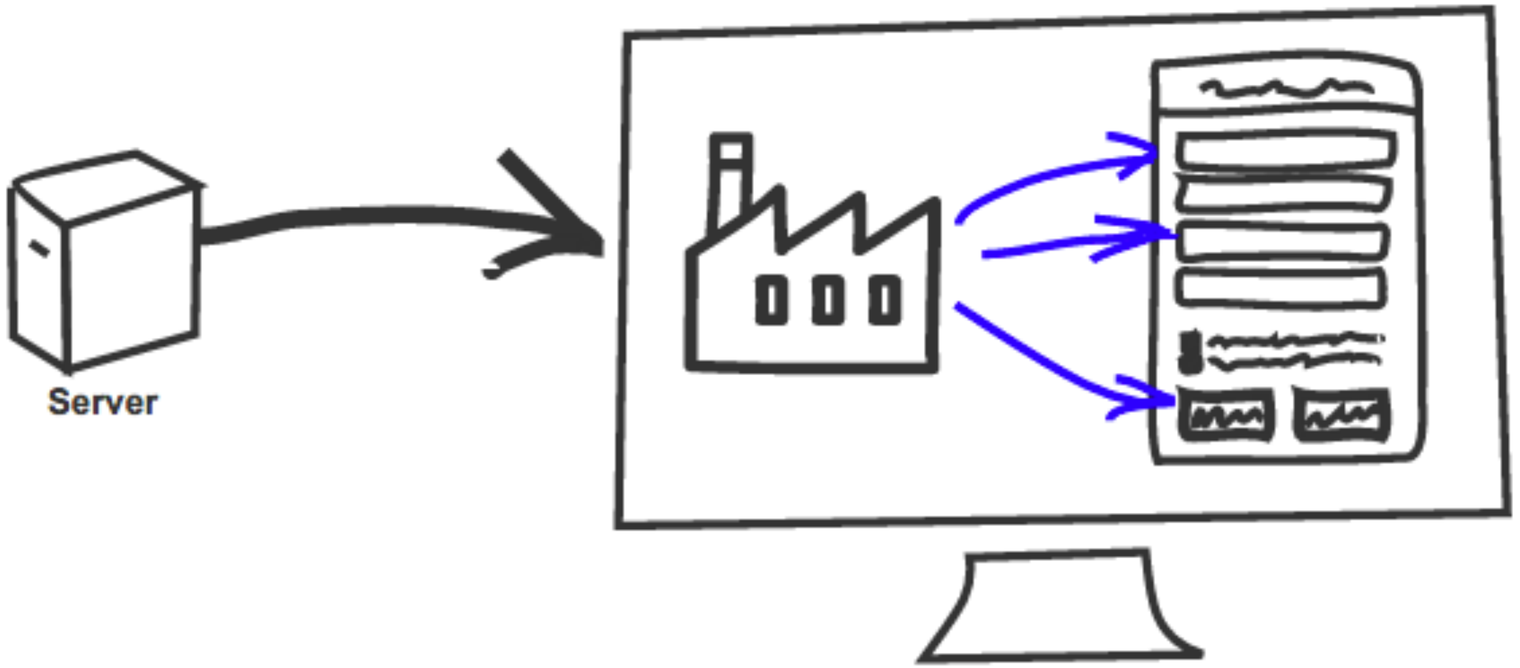
Иммутабельность

Макросы

core.async

```
(ns meetup-cljs.core
  (:require-macros [cljs.core.async.macros :refer [go]])
  (:require [cljs.core.async :as async))

(def state (atom {}))
(def state-watch []
  (let [c (async/chan 1)]
    (add-watch state :w (fn [_ _ _ n] (go (!> c n))))
  c))
```

```
(def state (atom {}))
(def watch-state []
  (let [c (async/chan 1)]
    (add-watch state :w (fn [_ _ _ n] (go (!> c n))))
  c))

(go-loop [] (js/console.log (async/<! (watch-state)))
  (recur))
;; do other stuff
(reset! state 42)
```


Server

```
(defn get [payload]
  (let [push (.push ws-fetch-channel "fetch" payload)
        ok-chan (async/chan)
        error-chan (async/chan) ]
    (.receive push “ok" #(put! ok-chan %))
    (.receive push “error" #(put! error-chan %))
    [chan error-chan]))
```

```
(defn get-kitty []
  (go
   (let [[ok-c error-c] (get :kitty)
         [value c] (alts! [ok-c error-c (async/timeout 1000)])]
     (condp = c
       ok-c :>> #(do-smth value)
       error-c :>> #(show-error value)
       (show-error “Timeout”)))))
```

- core.async
- Source maps
- Tooling
- Documentation
- Figwheel
- cljx/cljc
- Om
- re-frame
- CLJSJS
- Transit
- Bootstrap
- Planck
- Devcards
- ...


Спасибо за
внимание!