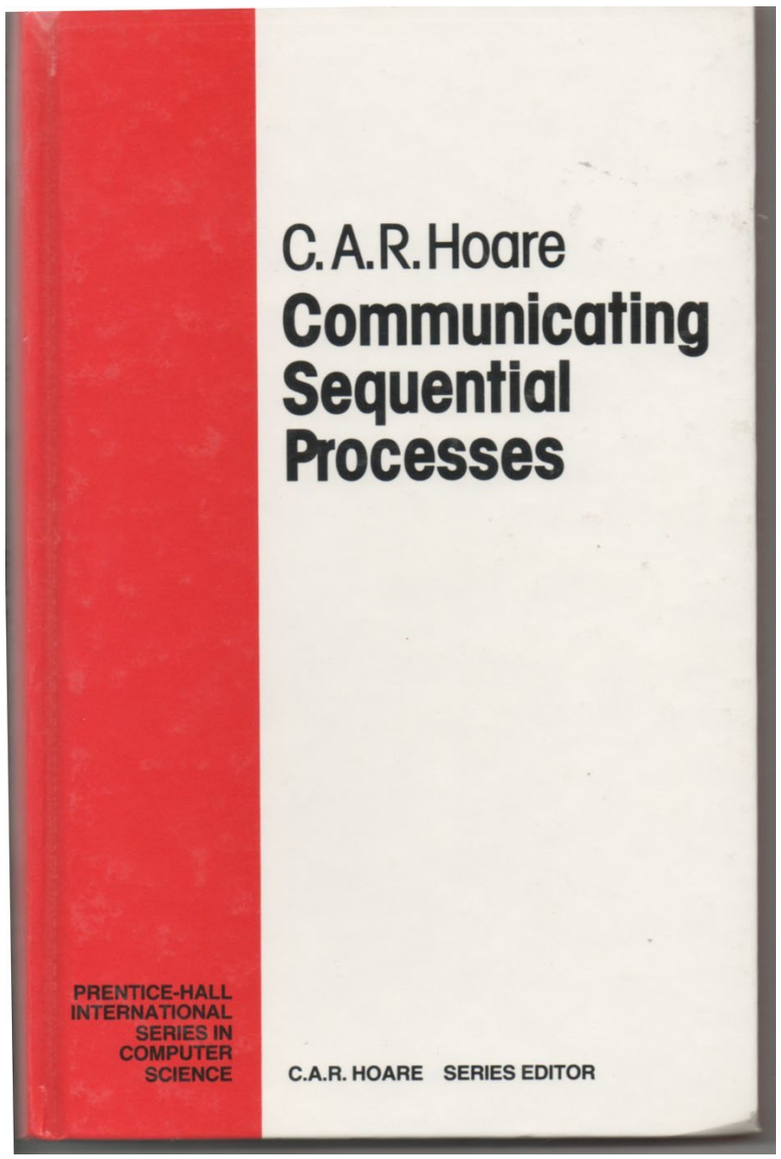
# Clojure core.async

Asynchronous programming using channels

# History




- Communicating Sequential Processes, originally described by Tony Hoare
- Currently popular with Go language

* Worthwhile read!

# Basics

```
(let [c (chan)]
  (go (>! c “hello”))
  (assert (= “hello” (<!! (go (<! c)))))
  (close! c))
```

# Timeout

```
(let [t (timeout 100)
      begin (System/currentTimeMillis)]
  (<!! t)
  (println “Waited “
    (- (System/currentTimeMillis) begin)))
```

# Select

```
(let [c (chan)
      t (timeout 100)
      [v sc] (alts!! [c t])]
  (condp = sc
    c (println “Read from channel”
    t (println “Gave up after 100ms”)))
```

# Buffers

```
;will block until read
(>! (chan) val)

;will put 10 values, then block until read
(>! (chan (buffer 10)) val)

;won't block, drops new values when full
(>! (chan (dropping-buffer 10)) val)

;won't block, drops old values when full
(>! (chan (sliding-buffer 10)) val)
```

# Threads & IO

```clojure
(defn blocking-get [url]
  (clj-http.client/get url))

(time
   (def data
     (let [c (chan)
           res (atom [])]
       ;; fetch em all
       (doseq [i (range 10 100)]
         (go (>! c (blocking-get (format "http://fssnip.net/%d" i)))))
       ;; gather results
       (doseq [_ (range 10 100)]
         (swap! res conj (<!! c)))
       @res)))

;; "Elapsed time: 11123.577 msecs"
```

examples via @martintrojer

# Threads & IO

```clojure
(defn blocking-get [url]
  (clj-http.client/get url))

(time
   (def data-thread
     (let [c (chan)
           res (atom [])]
       ;; fetch em all
       (doseq [i (range 10 100)]
         (thread (>!! c (blocking-get (format "http://fssnip.net/%d" i)))))
       ;; gather results
       (doseq [_ (range 10 100)]
         (swap! res conj (<!! c)))
       @res)))

;; "Elapsed time: 6523.782 msecs"
```

examples via @martintrojer

# Threads & IO

```clojure
(defn async-get [url result]
  (org.httpkit.client/get url #(go (>! result %))))

(time
 (def hk-data
   (let [c (chan)
         res (atom [])]
     ;; fetch em all
     (doseq [i (range 10 100)]
       (async-get (format "http://fssnip.net/%d" i) c))
     ;; gather results
     (doseq [_ (range 10 100)]
       (swap! res conj (<!! c)))
     @res)))

;; "Elapsed time: 6731.781 msecs"
```

examples via @martintrojer

# Examples

- Simulated Search
- 100,000 DOM Updates
- 10,000 Processes
- ClojureScript Dots Game

# More Reading

- CSP Book (Free Online)
- Go Lang Concurrency
- core.async release
- David Nolen's Blog
- Async & IO
- Walkthrough
- Detailed State Machine Walkthrough

# Questions?