Transducers and core.async

Clojure Exchange 2017 - Workshops

core.async

- Core.async is a library (not shipped with core Clojure)
- It nicely complement already existent concurrency models
- It is based on Hoare's Communicating Sequential Processes (CSP) model (see resources)
- "go-loop" avoids blocking by parking threads in the background (when threads are non-blocking).
- One of the main driver for transducers was to re-use the same sequential processing on top of core.async "channels"

Transducers can be used with core.async

- Items flowing through channels are similar to items in a collection
- On entering/exiting the channel they can be transformed, aggregated, etc.
- No need to create a special "async/map" implementation
- Just reuse the same sequential processing on channels

core.async essentials for the lab

```
; creates a channel with buffer size = 1
(def in (chan 1))

; also takes an optional transducer chain "xform"
(def out (chan 1 xform))

; claims back resources and stops the running loops
(close! in)

; Writes into a channel
>!

; Reads from a channel
<!
```

Resources

- Communicating Sequential Processes (CSP) paper
- Core.async walk-through
- Extended guide

Lab 03
Transducers and core.async

Goal of Lab3

- **Task 1**: create channels and orchestrate them so incoming products end up in a local cache
- **Task 2**: use the “prepare-data” transducer from lab01 to transform incoming products.

Open transducers-workshop.lab03 to start.