Clojure concurrency cheat sheet
===============================

Refs, altered only inside dosync

```
(def ref-name (ref initial-state)) => the ref
@ref-name   or   (deref ref-name) => the current value

(dosync (ref-set ref-name new_value) ...) => last value
(dosync (alter ref-name fn args) ...)  =>
  (alter ref-name fn args) sets ref-name to (fn ref-name args)
  ex. (dosync (alter my-list conj new-first))
```

Atoms, don't need dosync to alter

```
(def atom-name (atom initial-value)))
@atom-name   or   (deref atom-name)
(reset! atom-name new-value)  => new-value
(swap! atom-name fun args) calls (fun atom-name args) => new-value
```

Agents

```
(def agent-name (agent initial-state))
@agent-name   or   (deref agent-name)
(send agent-name update-function arguments) => agent
(send-off agent-name update-function arguments)
(await  agent-name-1 ... agent-name-N)
(await-for timeout-millis agent-name-1 ... agent-name-N)
(agent-errors agent-name)
(clear-agent-errors agent-name)
(shutdown-agents)
```

Watches (identity in {atom, ref, agent, var}

```
(add-watch identity key watch-function)
(defn watch-function-name [key identity old-val new-val] expressions)
(remove-watch identity key)
```

Futures

```
(def  future-name  (future  expressions))
@future-name   or   (deref future-name)
(future-done?  future-name); also, future-cancel, future-cancelled?,
future?
```

Promises

```
(def promise-name (promise))
@promise-name   or   (deref promise-name)
(deliver promise-name value)
```

```
Structs
-------

(defstruct struct-name key ... key)
(def my-struct (struct-map struct-name key value ... key value))
(assoc map key value ... key value)
(dissoc map key ... key)
(contains? map key)

Shark example
-------------

(defstruct shark :name :age :hunger)
(ref sherman (ref (struct-map shark :name "Sherman" :hunger 5)))
(defn feed [fish]
    (dosync
        (ref-set fish (assoc @fish :hunger (dec (@fish :hunger))))))
(feed sherman)

Misc.
-----

(Thread/sleep millis)
(rand)
```