ClojureScript Cheat Sheet

http://github.com/clojure/clojurescript

Documentation

http://github.com/clojure/clojurescript/wiki

Listing 1: Example Namespace Declaration

```
(ns my-cool-lib
  (:require [some-lib :as lib])
  (:use [another-lib :only (a-func)])
  (:require-macros [my.macros :as macs]))
```

Rich Data Literals

Maps:	`{:key1 :val1, :key2 :val2}`
Vectors:	`[1 2 3 4 :a :b :c 1 2]`
Sets:	`#{:a :b :c 1 2 3}`
Truth and nullity:	`true, false, nil`
Keywords:	`:kw, :a-2, :prefix/kw, ::pi`
Symbols:	`sym, sym-2, prefix/sym`
Characters:	`\a, \u1123, \space`
Int, Float, String:	same as in JavaScript

Frequently Used Functions

Math:	`+ - * / quot rem mod inc dec max min`
Comparison:	`= == not= < > <= >=`
Tests:	`nil? identical? zero? pos? neg? even? odd? true? false? nil?`
Keywords:	`keyword keyword?`
Symbols:	`symbol symbol? gensym`
Data Processing:	`map reduce filter partition split-at split-with`
Data Create:	`vector vec hash-map set list list* for`
Data Examination:	`first rest count get nth get get-in contains? find keys vals`
Data Manipulation:	`seq into conj cons assoc assoc-in dissoc zipmap merge merge-with select-keys update-in`
Arrays:	`into-array to-array aget aset amap areduce alength`

More information

http://clojuredocs.org

Frequently Used Macros

Defining:	`defmacro`
Macros:	`if if-let cond and or -> ->> doto when when-let ..`
Implementation:	Must be written in Clojure
Emission:	Must emit ClojureScript

Abstraction (http://clojure.org/protocols)

Protocols

Definition:	`(defprotocol Slicey (slice [at]))`
Extend:	`(extend-type js/String Slicey (slice [at] ...))`
Extend null:	`(extend-type nil Slicey (slice [_] nil))`
Reify:	`(reify Slicey (slice [at] ...))`

Records

Definition:	`(defrecord Pair [h t])`
Access:	`(:h (Pair. 1 2)) ;=> 1`
Constructing:	`Pair. ->Pair map->Pair`

Types

Definition:	`(deftype Pair [h t])`
Access:	`(.h (Pair. 1 2)) ;=> 1`
Constructing:	`Pair. ->Pair`
With Method(s):	`(deftype Pair [h t] Object (toString [] ...))`

Multimethods

Definition:	`(defmulti my-mm dispatch-function)`
Method Define:	`(defmethod my-mm :dispatch-value [args] ...)`

JS Interop (http://fogus.me/cljs-js)

Method Call:	`(.meth obj args)`
Method Call:	`(. obj (meth args))`
Property Access:	`(. obj -prop)`
Property Access:	`(.-prop obj)`
Set Property:	`(set! (.-prop obj) val)`
JS Direct Access:	`js/something`
JS this:	`(this-as me (.method me))`
Create JS Object:	`(js-obj)`

Compilation (http://fogus.me/cljsc)

Simple Compile:	`cljsc src-home '{:optimizations :simple :pretty-print true}'`
Advanced Compile:	`cljsc src-home '{:optimizations :advanced}'`

Extra ClojureScript Libraries

`clojure.{string set zipper}`
`clojure.browser.{dom event net repl}`

Other Useful Libraries

App Sample:	http://clojurescriptone.com
Client/Server:	http://github.com/ibdknox/fetch
D3:	http://github.com/lynaghk/cljs-d3
DOM:	http://github.com/levand/domina
Framework:	http://github.com/ibdknox/pinot
jQuery:	http://github.com/ibdknox/jayq

`$Revision: 1.0, $Date: Feb 08, 2012`
`Fogus (fogus -at- clojure -dot- com)`