SINBAD - Quick Reference - Racket

http://cs.berry.edu/sinbad

Basic Template

```
(require sinbad)

(define ds (sail-to "<URL>"
                    (manifest) ; can be invoked separately
                    (load)     ; afterwards, eg. (load ds)
                    ...<additional clauses>...))
(fetch ds "...")
```

Examining Available Data

```
(manifest ds)  ; can also be included in sail-to (above)
```

Test if field paths valid:

```
(has-fields? ds ".../..." ...)
```

List of available field names:

```
(field-list ds)    ;; OR   (field-list ds ".../..." )
```

Number of data records (in a list) available:

```
(data-length ds)   ;; OR   (data-length ds ".../...")
```

Other Connection Methods

Specify a data format ("CSV", "XML", "JSON"):

```
    (sail-to "<URL>"  (format "XML") ...))
```

Connect using a data specification file:

```
    (sail-to (spec "<URL>")  ...))
```

Connection (URL) Parameters

Some data sources may require additional parameters to construct the final URL to load data from:

```
    (sail-to "<URL>"  ...
             (param "<name>" "<value>")))
```

Data Format Options

Some data sources provide post-processing options to manipulate the data once it has been downloaded. The available options are format-specific and are listed in the *print_description()* information.

```
   (sail-to "<URL>"  ...  (option "<name>" "<value>")))
```

For example (with a CSV data source):

```
   (sail-to "<URL>"  ...
            (option "header"
                    "ID,Name,Country,Active")))
```

Selecting from .zip archive

```
   (sail-to "<URL>"  ...
            (format "...") ; often needed
            (option "file-entry" "FACTDATA.CSV")))
```

Sampling Data

```
  (sail-to "<URL>" ...   (sample <amt>)  ; instead of (load)
```

Sampled data is cached and reloaded from cache if the same code is run again. To force a fresh sample to be generated, use:

```
  (fresh-sample <amt>)   ;; OR   (fresh-sample <amt> <seed>)
```

Cache Control

Control frequency of caching (or disable it):

```
  (sail-to "<URL>" ...   (cache-timeout 300))  ; 300 seconds
;; OR   (cache-timeout NEVER-RELOAD)  -- always use cache
;; OR   (cache-timeout NEVER-CACHE)
```

Show where files are cached:

```
(cache-directory ds)
```

Clear all cache files (for *ds* and all data sources):

```
(clear-entire-cache ds)
```

Fetching Data

```
### GENERAL PURPOSE -----

(fetch ds)
     ;; fetches ALL available data (lists + assoc lists)

(fetch ds "path/to/field1" ...)
     ;; fetches (lists of, if appropriate) data
(fetch ds "path/to/field1" ... (base-path "loans"))
     ;; using optional base-path clause
(fetch ds (<proc>  "path/to/field1" ...) (base-path "loans"))
     ;; fetch and apply <proc> to each group of field values
     ;; base-path clause is optional

### RANDOM -----

(fetch-random ds ...)
    ;; same patterns as for (fetch ds ...) above
    ;; note: always returns the same result until (load ds)
             invoked again, e.g. (fetch-random (load ds) ...)

### POSITIONAL -----
     ;; same patterns as for (fetch ds ...) above
(fetch-first ds ...)
(fetch-second ds ...)
(fetch-third ds ...)
(fetch-ith ds i ...)        ;; 0 <= i < (data-length ds)

### TYPE CONVERTING —

(fetch-number       ds "path/to/field") ; fetch-first-number
(fetch-numbers      ds "path/to/field")
(fetch-ith-number   ds i "path/to/field")
(fetch-random-number ds "path/to/field")
;; all the same available for  fetch-...-boolean  as well

### FULL-API FETCH (undocumented, unsupported) —

(fetch* ds <sig-exp> ...)
; e.g.
   (fetch* Q `(path "features" "properties"
                    (,make-quake "place"
                                 (,quake-ts->timestr "time")
                                 "mag")))
```