testeez egg

A lightweight unit test mechanism for Scheme.
Extension for Chicken Scheme
Version 0.3

Neil W. Van Dyke

Table of Contents

1 About this egg

1.1 Version history

`0.3` Synchronize with upstream 0.3, expose test/equiv (Zbigniew)

`0.1.1` test-define with low-level macros didn't show description

`0.1` Initial port of upstream 0.1 (Felix)

1.2 Usage

Load this egg like so:

```
(require-extension testeez)
```

2 Documentation

See the official documentation. There are a couple of additions for Chicken:

`test/equiv` [test clause]

```
(test/equiv DESC EXPR EXPECTED (PRED ...))
```

Additional clause form which allows you to specify one or more custom equivalence predicates. You should either specify one predicate, which will be checked against every value expected; or N predicates, where N is the number of values expected. In the latter case, each predicate is checked only against its corresponding value.

The form `(test/eq DESC EXPR EXPECTED)` is exactly the same as `(test/equiv DESC EXPR EXPECTED (eq?))`.

This clause is non-public in the upstream version, but has been exposed since it seems useful. Its syntax is subject to change.

`%testeez:self-test` [procedure]

```
(%testeez:self-test)
```

See example below.

3 Examples

The following definition is made available as `%testeez:self-test` if the `debug` feature is registered at runtime. For example, run `csi -D debug -R testeez -eval "(%testeez:self-test)"`.

```
(define (%testeez:self-test)
  (testeez
   "Foo Station"

   (test/equal "Put two and two together" (+ 2 2) 4)

   (test-define "Bar function" bar (lambda (x) (+ x 42)))

   (test/equal "Bar scene" (bar 69) 0)  ;; will fail

   (test/eqv   "Full circle" (* (bar -21) 2) 42)

   (test/equal "Multiple"
               (values (+ 2 2) (string #h #i) (char-upcase #p))
               (values 4 "hi" #P))

   (test/equiv "Multiple predicates"
               (values (+ 2 2) (string #h #i) (char-upcase #p))
               (values 4 "hi" #P)
               (eqv? string=? char=?))))
```

Output:

```
#;1> (%testeez:self-test)

;;; BEGIN "Foo Station" TESTS

;; 1. Put two and two together
(+ 2 2)
;; ==> 4
;; Passed.

;; DEFINE: Bar function
(define bar (lambda (x) (+ x 42)))

;; 2. Bar scene
(bar 69)
;; ==> 111
;; FAILED!  Expected:
;;     0
```

```
;; 3. Full circle
(* (bar -21) 2)
;; ==> 42
;; Passed.

;; 4. Multiple
(values (+ 2 2) (string #h #i) (char-upcase #p))
;; ==> 4
;;     "hi"
;;     #P
;; Passed.

;; 5. Multiple predicates
(values (+ 2 2) (string #h #i) (char-upcase #p))
;; ==> 4
;;     "hi"
;;     #P
;; Passed.

;;; END "Foo Station" TESTS: FAILED
;;;     (Total: 5  Passed: 4  Failed: 1)
```

4 License

Index

%

T