

# Black Hole

Module system for Gambit

Bundled libraries documentation, revision 1. **\*\* DRAFT \*\***

Module system core copyright © 2008-2009 Per Eckerdal

Documentation copyright © 2009 Mikael More

Bundled libraries are copyrighted by their respective authors, details are found in each library's sourcecode and/or documentation.

Licensed under the MIT license, see the License section in the core documentation for details.

Documentation distributed in PDF, OpenOffice and Microsoft Word versions.

Please email any feedback, questions and suggestions to the authors.

## Table of Contents

<a href="#">Table of Contents, with all exports</a> .....	2
<a href="#">Introduction</a> .....	6
<a href="#">ds (data structures) directory</a> .....	6
<a href="#">queue – FIFO queue</a> .....	6
<a href="#">wt-tree – Weight balanced trees</a> .....	7
<a href="#">misc directory</a> .....	9
<a href="#">al – A-list helper library</a> .....	9
<a href="#">exception – Exception handling helper library</a> .....	9
<a href="#">mailbox – Single-element mailbox library</a> .....	9
<a href="#">match – Erlang-style list matcher</a> .....	9
<a href="#">optionals – let-optionals special form support</a> .....	10
<a href="#">splice – Splicer</a> .....	10
<a href="#">u8v – U8vector utility functions</a> .....	11
<a href="#">uuid – UUID generator</a> .....	11
<a href="#">net directory</a> .....	12
<a href="#">http-client – HTTP client</a> .....	12
<a href="#">http-common – Common HTTP features</a> .....	12
<a href="#">http-server – HTTP server</a> .....	12
<a href="#">http-session – HTTP session helper library</a> .....	12
<a href="#">tcpip – TCP/IP helper procedures</a> .....	13
<a href="#">uri – URI handling</a> .....	13
<a href="#">x-www-form-urlencoded - x-www-form-urlencoded encoding and decoding</a> .....	15
<a href="#">srfi directory</a> .....	16
<a href="#">1 – List processing</a> .....	16
<a href="#">13 – String</a> .....	16
<a href="#">14 – Character-sets</a> .....	16
<a href="#">16 – case-lambda special form support</a> .....	16
<a href="#">19 – Time data types and procedures</a> .....	16
<a href="#">95 – Sorting</a> .....	16
<a href="#">string directory</a> .....	16
<a href="#">base64 – Base64 processing library</a> .....	16
<a href="#">digest – Hash computation library</a> .....	17

<a href="#">sxml-to-xml – Fast SXML to XML generator</a> .....	22
<a href="#">util – String utility procedures</a> .....	23
<a href="#">xml-to-sxml – Fast XML string to SXML parser</a> .....	26
<a href="#">Index</a> .....	26

## Table of Contents, with all exports

Table of Contents, with all exports.....	2
<a href="#">Introduction</a> .....	6
<a href="#">ds (data structures) directory</a> .....	6
<a href="#">queue – FIFO queue</a> .....	6
<a href="#">(make-queue)</a> .....	6
<a href="#">(queue-size queue)</a> .....	6
<a href="#">(queue-push! queue element)</a> .....	6
<a href="#">(queue-pop! queue)</a> .....	6
<a href="#">(queue-empty? queue)</a> .....	6
<a href="#">(queue-empty! queue)</a> .....	6
<a href="#">(queue-front)</a> .....	7
<a href="#">wt-tree – Weight balanced trees</a> .....	7
<a href="#">(make-wt-tree-type)</a> .....	7
<a href="#">(make-wt-tree)</a> .....	7
<a href="#">(singleton-wt-tree)</a> .....	7
<a href="#">(alist-&gt;wt-tree)</a> .....	7
<a href="#">(wt-tree/empty?)</a> .....	7
<a href="#">(wt-tree/size)</a> .....	7
<a href="#">(wt-tree/add)</a> .....	7
<a href="#">(wt-tree/delete)</a> .....	7
<a href="#">(wt-tree/add!)</a> .....	7
<a href="#">(wt-tree/delete!)</a> .....	7
<a href="#">(wt-tree/member?)</a> .....	7
<a href="#">(wt-tree/lookup)</a> .....	7
<a href="#">(wt-tree/split&lt;)</a> .....	8
<a href="#">(wt-tree/split&gt;)</a> .....	8
<a href="#">(wt-tree/union)</a> .....	8
<a href="#">(wt-tree/intersection)</a> .....	8
<a href="#">(wt-tree/difference)</a> .....	8
<a href="#">(wt-tree/subset?)</a> .....	8
<a href="#">(wt-tree/set-equal?)</a> .....	8
<a href="#">(wt-tree/fold)</a> .....	8
<a href="#">(wt-tree/for-each)</a> .....	8
<a href="#">(wt-tree/index)</a> .....	8
<a href="#">(wt-tree/index-datum)</a> .....	8
<a href="#">(wt-tree/index-pair)</a> .....	8
<a href="#">(wt-tree/rank)</a> .....	8
<a href="#">(wt-tree/min)</a> .....	8
<a href="#">(wt-tree/min-datum)</a> .....	8
<a href="#">(wt-tree/min-pair)</a> .....	8
<a href="#">(wt-tree/delete-min)</a> .....	9
<a href="#">(wt-tree/delete-min!)</a> .....	9
<a href="#">(number-wt-type)</a> .....	9
<a href="#">(string-wt-type)</a> .....	9
<a href="#">misc directory</a> .....	9
<a href="#">al – A-list helper library</a> .....	9
<a href="#">exception – Exception handling helper library</a> .....	9
<a href="#">mailbox – Single-element mailbox library</a> .....	9
<a href="#">(make-empty-mailbox)</a> .....	9

<a href="#">(mailbox-put!)</a> .....	9
<a href="#">(mailbox-get!)</a> .....	9
<a href="#">match – Erlang-style list matcher</a> .....	9
<a href="#">(match value . handling-rules)</a> .....	9
<a href="#">optionals – let-optionals special form support</a> .....	10
<a href="#">splice – Splicer</a> .....	10
<a href="#">(splice . args)</a> .....	10
<a href="#">(unsplice-list list)</a> .....	10
<a href="#">u8v – U8vector utility functions</a> .....	11
<a href="#">(u8vector . args)</a> .....	11
<a href="#">(u8vector-reverse v)</a> .....	11
<a href="#">(subu8vector-&gt;hex-string u8vect start end), (u8vector-&gt;hex-string u8vect)</a> .....	11
<a href="#">(hex-string-&gt;u8vector s)</a> .....	11
<a href="#">(u8vector-invert! v)</a> .....	11
<a href="#">(hex-char-&gt;integer c)</a> .....	11
<a href="#">uuid – UUID generator</a> .....	11
<a href="#">(make-uuid)</a> .....	12
<a href="#">net directory</a> .....	12
<a href="#">http-client – HTTP client</a> .....	12
<a href="#">(http-get-url uri #!optional http-username http-password (headers '()) (follow #t) query) . (http-post-url uri #!optional http-username http-password (headers '()) (follow #t) query)</a> .....	12
<a href="#">http-common – Common HTTP features</a> .....	12
<a href="#">http-server – HTTP server</a> .....	12
<a href="#">http-session – HTTP session helper library</a> .....	12
<a href="#">(make-session-variable #!optional default-value)</a> .....	12
<a href="#">tcpip – TCP/IP helper procedures</a> .....	13
<a href="#">(call-with-tcp-client hostname port-num proc)</a> .....	13
<a href="#">(ipv4-&gt;u8vector ip)</a> .....	13
<a href="#">(tcp-socket-port-&gt;fd port)</a> .....	13
<a href="#">(device-port-rdevice-condvar port)</a> .....	13
<a href="#">(device-port-wdevice-condvar port)</a> .....	13
<a href="#">(port-rtimeout port)</a> .....	13
<a href="#">(port-wtimeout port)</a> .....	13
<a href="#">(port-wait-for-input port)</a> .....	13
<a href="#">(port-wait-for-output port)</a> .....	13
<a href="#">uri – URI handling</a> .....	13
<a href="#">(make-uri)</a> .....	13
<a href="#">(uri?)</a> .....	13
<a href="#">(uri-scheme)</a> .....	14
<a href="#">(uri-scheme-set!)</a> .....	14
<a href="#">(uri-authority)</a> .....	14
<a href="#">(uri-authority-set!)</a> .....	14
<a href="#">(uri-path)</a> .....	14
<a href="#">(uri-path-set!)</a> .....	14
<a href="#">(uri-query)</a> .....	14
<a href="#">(uri-query-set!)</a> .....	14
<a href="#">(uri-fragment)</a> .....	14
<a href="#">(uri-fragment-set!)</a> .....	14
<a href="#">(uri-userinfo)</a> .....	14
<a href="#">(uri-host)</a> .....	14
<a href="#">(uri-port)</a> .....	14
<a href="#">(uri-query-string)</a> .....	14
<a href="#">(parse-uri)</a> .....	14
<a href="#">(parse-uri-query)</a> .....	14
<a href="#">(string-&gt;uri)</a> .....	15
<a href="#">(uri-&gt;string)</a> .....	15
<a href="#">(string-&gt;uri-query)</a> .....	15
<a href="#">(encode-for-uri)</a> .....	15

(remove-dot-segments).....	15
(uri-join).....	15
(uri-join*).....	15
x-www-form-urlencoded - x-www-form-urlencoded encoding and decoding.....	15
(encode-x-www-form-urlencoded fields).....	15
(decode-x-www-form-urlencoded str).....	15
(urlencode str).....	15
(urldecode str).....	15
(write-urlencoded str).....	15
(write-x-www-form-urlencoded fields).....	15
srfi directory.....	16
1 – List processing.....	16
13 – String.....	16
14 – Character-sets.....	16
16 – case-lambda special form support.....	16
19 – Time data types and procedures.....	16
95 – Sorting.....	16
string directory.....	16
base64 – Base64 processing library.....	16
(base64-string->u8vector str) , (base64-substring->u8vector str start end).....	16
(u8vector->base64-string u8vect #!optional (width 0)) , (subu8vector->base64-string u8vect start end #! optional (width 0)).....	17
digest – Hash computation library.....	17
(digest-file filename algorithm #!optional (result-type (digest-default-result-type))).....	17
(digest-subu8vector).....	17
(digest-u8vector).....	17
(digest-substring).....	17
(digest-string).....	17
(digest-update-u32-be).....	17
(digest-update-u32-le).....	17
(digest-update-u16-be).....	17
(digest-update-u16-le).....	17
(digest-update-u8).....	17
(get-zero-u8vector).....	17
zero-u8vector.....	17
(digest-update-subu8vector).....	17
(close-digest).....	18
(open-digest).....	18
(open-digest-sha-256).....	18
(open-digest-sha-224).....	18
(digest-update-sha-256).....	18
(hash-block-init-sha-256).....	18
(hash-block-init-sha-224).....	18
(open-digest-sha-1).....	18
(digest-update-sha-1).....	18
(hash-block-init-sha-1).....	18
(open-digest-md5).....	18
(digest-update-md5).....	18
(hash-block-init-md5).....	18
(digest-update-block-digest).....	18
(end-block-digest).....	18
(process-last-block).....	18
(convert-hash-block).....	19
(block-digest-width-set!).....	19
(block-digest-width).....	19
(block-digest-big-endian?-set!).....	19
(block-digest-big-endian?).....	19
(block-digest-bit-pos-set!).....	19

<a href="#">(block-digest-bit-pos)</a> .....	19
<a href="#">(block-digest-block-pos-set!)</a> .....	19
<a href="#">(block-digest-block-pos)</a> .....	19
<a href="#">(block-digest-block-set!)</a> .....	19
<a href="#">(block-digest-block)</a> .....	19
<a href="#">(block-digest-hash-set!)</a> .....	19
<a href="#">(block-digest-hash)</a> .....	19
<a href="#">(block-digest-hash-update-set!)</a> .....	19
<a href="#">(block-digest-hash-update)</a> .....	19
<a href="#">(block-digest?)</a> .....	20
<a href="#">(make-block-digest)</a> .....	20
<a href="#">(hash-block-&gt;u8vector)</a> .....	20
<a href="#">(hash-block-&gt;hex-string)</a> .....	20
<a href="#">(open-digest-crc32)</a> .....	20
<a href="#">(end-crc32)</a> .....	20
<a href="#">(digest-update-crc32)</a> .....	20
<a href="#">(crc32-table)</a> .....	20
<a href="#">(crc32-digest-lo16-set!)</a> .....	20
<a href="#">(crc32-digest-lo16)</a> .....	20
<a href="#">(crc32-digest-hi16-set!)</a> .....	20
<a href="#">(crc32-digest-hi16)</a> .....	20
<a href="#">(crc32-digest?)</a> .....	20
<a href="#">(make-crc32-digest)</a> .....	20
<a href="#">(digest-state-set!)</a> .....	20
<a href="#">(digest-state)</a> .....	20
<a href="#">(digest-update-set!)</a> .....	21
<a href="#">(digest-update)</a> .....	21
<a href="#">(digest-end-set!)</a> .....	21
<a href="#">(digest-end)</a> .....	21
<a href="#">(digest?)</a> .....	21
<a href="#">(make-digest)</a> .....	21
<a href="#">pregexp – Posix Regular Expressions matching engine library</a> .....	21
<a href="#">(pregexp s)</a> .....	21
<a href="#">(pregexp-match-positions pat str . opt-args)</a> .....	21
<a href="#">(pregexp-match pat str . opt-args)</a> .....	21
<a href="#">(pregexp-split pat str)</a> .....	21
<a href="#">(pregexp-replace pat str ins)</a> .....	21
<a href="#">(pregexp-replace* pat str ins)</a> .....	21
<a href="#">(pregexp-quote s)</a> .....	21
<a href="#">(pregexp-match? pattern string)</a> .....	22
<a href="#">sxml-to-xml – Fast SXML to XML generator</a> .....	22
<a href="#">(sxml&gt;&gt;html* item #!optional (port (current-output-port)) maybe-level)</a> .....	22
<a href="#">(sxml&gt;&gt;xhtml* item #!optional (port (current-output-port)) maybe-level)</a> .....	22
<a href="#">(sxml&gt;&gt;xml* item #!optional (port (current-output-port)) maybe-level)</a> .....	22
<a href="#">(sxml&gt;&gt;html-fast item #!optional (port (current-output-port)))</a> .....	22
<a href="#">(sxml&gt;&gt;html item #!optional (port (current-output-port)))</a> .....	22
<a href="#">(sxml&gt;&gt;xhtml-fast item #!optional (port (current-output-port)))</a> .....	22
<a href="#">(sxml&gt;&gt;xhtml item #!optional (port (current-output-port)))</a> .....	22
<a href="#">(sxml&gt;&gt;xml-fast item #!optional (port (current-output-port)))</a> .....	22
<a href="#">(sxml&gt;&gt;xml item #!optional (port (current-output-port)))</a> .....	22
<a href="#">(sxml&gt;&gt;pretty-xml-file item outpath #!optional noblanks)</a> .....	22
<a href="#">(sxml&gt;&gt;pretty-xhtml-file item outpath #!optional noblanks)</a> .....	22
<a href="#">(sxml-&gt;html-string-fragment item #!optional maybe-level)</a> .....	22
<a href="#">(sxml-&gt;xml-string-fragment item #!optional maybe-level)</a> .....	22
<a href="#">util – String utility procedures</a> .....	23
<a href="#">(char-&gt;string char)</a> .....	23
<a href="#">(string-strip str)</a> .....	23
<a href="#">(string-replace-char str from to)</a> .....	23
<a href="#">(join between args)</a> .....	23
<a href="#">(string-split chr str #!optional (sparse #f))</a> .....	23

<a href="#">(string-split-at-first at-char s #!optional (not-found (lambda () (error "String parameter didn't contain char."))))</a> .....	23
<a href="#">(string-split-at-first-nice at-char s)</a> .....	23
<a href="#">(string-name-split str)</a> .....	24
<a href="#">(string-set-first-char! fun str)</a> .....	24
<a href="#">(string-capitalize! str)</a> , <a href="#">(string-decapitalize! str)</a> .....	24
<a href="#">(string-constantize str)</a> , <a href="#">(string-camelize str)</a> , <a href="#">(string-dasherize str)</a> , <a href="#">(string-spaceize str)</a> , <a href="#">(string-humanize str)</a> .....	24
<a href="#">(string-remove-suffix haystack needle)</a> , <a href="#">(string-remove-prefix haystack needle)</a> .....	25
<a href="#">(string-pluralize str #!optional (num 0))</a> , <a href="#">(string-depluralize str)</a> .....	25
<a href="#">(symbol-append . syms)</a> .....	25
<a href="#">(string-invert str)</a> , <a href="#">(string-uninvert str)</a> .....	25
<a href="#">(dumps . args)</a> .....	26
<a href="#">xml-to-sxml – Fast XML string to SXML parser</a> .....	26
<a href="#">html-entity-unicode-numbers</a> .....	26
<a href="#">html-entity-unicode-chars</a> .....	26
<a href="#">(xml-string-&gt;sxml xml-string #!optional (namespace-prefix-assig '()))</a> .....	26
Index.....	26

## Introduction

All of the bundled libraries are in the `std` module-resolver. I.e., to import SRFI 13 and 14, evaluate `(import (std srfi/13 srfi/14))`.

## ds (data structures) directory

### queue – FIFO queue

© Per Eckerdal

Not threadsafe.

*(make-queue)*

*(queue-size queue)*

*(queue-push! queue element)*

*(queue-pop! queue)*

*(queue-empty? queue)*

*(queue-empty! queue)*

*(queue-front)*

**wt-tree – Weight balanced trees**

© 1993-1994 Stephen Adams

See [http://www.gnu.org/software/mit-scheme/documentation/mit-scheme-ref/Weight\\_002dBalanced-Trees.html](http://www.gnu.org/software/mit-scheme/documentation/mit-scheme-ref/Weight_002dBalanced-Trees.html) .

*(make-wt-tree-type)*

*(make-wt-tree)*

*(singleton-wt-tree)*

*(alist->wt-tree)*

*(wt-tree/empty?)*

*(wt-tree/size)*

*(wt-tree/add)*

*(wt-tree/delete)*

*(wt-tree/add!)*

*(wt-tree/delete!)*

*(wt-tree/member?)*

*(wt-tree/lookup)*

*(wt-tree/split<)*

*(wt-tree/split>)*

*(wt-tree/union)*

*(wt-tree/intersection)*

*(wt-tree/difference)*

*(wt-tree/subset?)*

*(wt-tree/set-equal?)*

*(wt-tree/fold)*

*(wt-tree/for-each)*

*(wt-tree/index)*

*(wt-tree/index-datum)*

*(wt-tree/index-pair)*

*(wt-tree/rank)*

*(wt-tree/min)*

*(wt-tree/min-datum)*

*(wt-tree/min-pair)*



*(wt-tree/delete-min)*

*(wt-tree/delete-min!)*

*(number-wt-type)*

*(string-wt-type)*

## **misc directory**

### **al – A-list helper library**

© Per Eckerdal. See sourcecode.

### **exception – Exception handling helper library**

© Christian Jaeger

### **mailbox – Single-element mailbox library**

© Marc Feeley. Taken from the Gambit manual with permission.

*(make-empty-mailbox)*

*(mailbox-put!)*

*(mailbox-get!)*

### **match – Erlang-style list matcher**

© Per Eckerdal

*(match value . handling-rules)*

Example:

```
> (match '(1 2 3) ((1 2 3) #t) (else #f))
#t
> (match 3 (3 #t) (else else))
#t
> (match 4 (3 #t) (else else))
4
> (define (test msg)
  (match msg
    (('no-args)
      'no-args)
    (('one-arg a1)
      `("The arg is " ,a1))
    (('two-args a1 a2)
      `("The two args are " ,a1 ,a2))
    (else
      (list 'nothing-matched: else))))

> (test '(two-args 1 2))
("The two args are " 1 2)
> (test 'bob)
(nothing-matched: bob)
```

## optionals – let-optionals special form support

© Olin Shivers

## splice – Splicer

© Per Eckerdal

### *(splice . args)*

Returns a conveniency splice data structure for containing the results of any splice process.

```
> (splice 1 2 3)
#<splice #2 data: (1 2 3)>
```

### *(unsplice-list list)*

A very simple data structure that is intended for use with SXML and similar data structures. Use like:

```
(unsplice-list `(element ,(splice 1 2 3) 4)) => (element 1 2 3 4)
```

The point is that you should be able to write functions that return multiple elements.

```
> (define (a-fun) (splice 1 2 3))
> (define (b-fun) 4)
> (unsplice-list `(element ,(a-fun) ,(b-fun)))
(element 1 2 3 4)
```

**u8v – U8vector utility functions**

© Mikael More

***(u8vector . args)***

```
> (u8vector 1 2 3)
#u8(1 2 3)
```

***(u8vector-reverse v)***

```
> (define u (make-u8vector 3)) (u8vector-set! u 1 1) (u8vector-set! u 2 2)
> > > (u8vector-reverse u)
#u8(2 1 0)
```

***(subu8vector->hex-string u8vect start end), (u8vector->hex-string u8vect)***

```
> (define u (make-u8vector 3)) (u8vector-set! u 1 1) (u8vector-set! u 2 2)
> > > (u8vector->hex-string u)
"000102"
```

***(hex-string->u8vector s)***

```
> (hex-string->u8vector "010203")
#u8(1 2 3)
```

***(u8vector-invert! v)***

```
> (define u (make-u8vector 3)) (u8vector-set! u 1 1) (u8vector-set! u 2 2)
> > > u
#u8(0 1 2)
> (u8vector-invert! u)
> u
#u8(255 254 253)
```

***(hex-char->integer c)***

Takes either of the characters 0-9 or a-f or A-F for argument, and returns the integer value 0-15.

```
> (map hex-char->integer ' (#\0 #\a #\F))
(0 10 15)
> (hex-char->integer #\g)
*** ERROR IN "(generated)"@1.1 -- Character not 0-9, A-F, a-f. #\g
1>
```

**uuid – UUID generator**

© Guillaume Germain

***(make-uuid)***

Example:

```
> (make-uuid)
"DB7F49A3-BBF6-40F3-95A1-038CCACE1D88"
```

Uses Gambit's random number facility.

**net directory****http-client – HTTP client**

© Per Eckerdal

The library contains mechanisms to receive request responses streaming, see sourcecode.

*(http-get-url uri #!optional http-username http-password (headers '()) (follow #t) query) , (http-post-url uri #!optional http-username http-password (headers '()) (follow #t) query)*

Non-streaming HTTP GET/POST routines.

Return (list response-code response-headers response-data).

**http-common – Common HTTP features**

© Marc Feeley, Per Eckerdal

**http-server – HTTP server**

© Marc Feeley

**http-session – HTTP session helper library**

© Per Eckerdal

Implementation of HTTP sessions, for use together with http-server.

*(make-session-variable #!optional default-value)*

In order to create a session variable, say `user-id`, do

```
(define user-id (make-session-variable [default value, defaults to #f]))
```

In the code that responds to HTTP requests, do `(user-id)` to get the value of the session variable, and `(user-id [new value])` to set its value.

THE CURRENT VERSION DOESN'T LET OVER OLD VALUES TO THE GARBAGE COLLECTOR, LEAKS 100% MEMORY.

### **tcpip – TCP/IP helper procedures**

© Mikael More

*(call-with-tcp-client hostname port-num proc)*

*(ipv4->u8vector ip)*

*(tcp-socket-port->fd port)*

*(device-port-rdevice-condvar port)*

*(device-port-wdevice-condvar port)*

*(port-rtimeout port)*

*(port-wtimeout port)*

*(port-wait-for-input port)*

*(port-wait-for-output port)*

### **uri – URI handling**

© Marc Feeley

*(make-uri)*

*(uri?)*

*(uri-scheme)*

*(uri-scheme-set!)*

*(uri-authority)*

*(uri-authority-set!)*

*(uri-path)*

*(uri-path-set!)*

*(uri-query)*

*(uri-query-set!)*

*(uri-fragment)*

*(uri-fragment-set!)*

*(uri-userinfo)*

*(uri-host)*

*(uri-port)*

*(uri-query-string)*

*(parse-uri)*

*(parse-uri-query)*

*(string->uri)*

*(uri->string)*

*(string->uri-query)*

*(encode-for-uri)*

*(remove-dot-segments)*

*(uri-join)*

*(uri-join\*)*

**x-www-form-urlencoded - x-www-form-urlencoded encoding and decoding**

© Marc Feeley

*(encode-x-www-form-urlencoded fields)*

*(decode-x-www-form-urlencoded str)*

*(urlencode str)*

*(urldecode str)*

*(write-urlencoded str)*

*(write-x-www-form-urlencoded fields)*

## **srfi directory**

### **1 – List processing**

© 1998, 1999 Olin Shivers

See <http://srfi.schemers.org/srfi-1/srfi-1.html> .

### **13 – String**

© 1988-1994 Massachusetts Institute of Technology, © 1998, 1999, 2000 Olin Shivers. All rights reserved.

See <http://srfi.schemers.org/srfi-13/srfi-13.html> .

### **14 – Character-sets**

© 1988-1995 Massachusetts Institute of Technology

See <http://srfi.schemers.org/srfi-14/srfi-14.html> .

### **16 – case-lambda special form support**

Public domain.

See <http://srfi.schemers.org/srfi-16/srfi-16.html> .

### **19 – Time data types and procedures**

© 2000, 2002, 2003 I/NET, Inc. All Rights Reserved.

See <http://srfi.schemers.org/srfi-19/srfi-19.html> .

### **95 – Sorting**

By Richard A. O'Keefe. Public domain.

See <http://srfi.schemers.org/srfi-95/srfi-95.html> .

## **string directory**

### **base64 – Base64 processing library**

© Marc Feeley

*(base64-string->u8vector str) , (base64-substring->u8vector str start end)*



*(u8vector->base64-string u8vect #!optional (width 0)) , (subu8vector->base64-string u8vect start end #!optional (width 0))*

**digest – Hash computation library**

© Marc Feeley

*(digest-file filename algorithm #!optional (result-type (digest-default-result-type)))*

*(digest-subu8vector)*

*(digest-u8vector)*

*(digest-substring)*

*(digest-string)*

*(digest-update-u32-be)*

*(digest-update-u32-le)*

*(digest-update-u16-be)*

*(digest-update-u16-le)*

*(digest-update-u8)*

*(get-zero-u8vector)*

*zero-u8vector*

*(digest-update-subu8vector)*

*(close-digest)*

*(open-digest)*

*(open-digest-sha-256)*

*(open-digest-sha-224)*

*(digest-update-sha-256)*

*(hash-block-init-sha-256)*

*(hash-block-init-sha-224)*

*(open-digest-sha-1)*

*(digest-update-sha-1)*

*(hash-block-init-sha-1)*

*(open-digest-md5)*

*(digest-update-md5)*

*(hash-block-init-md5)*

*(digest-update-block-digest)*

*(end-block-digest)*

*(process-last-block)*

*(convert-hash-block)*

*(block-digest-width-set!)*

*(block-digest-width)*

*(block-digest-big-endian?-set!)*

*(block-digest-big-endian?)*

*(block-digest-bit-pos-set!)*

*(block-digest-bit-pos)*

*(block-digest-block-pos-set!)*

*(block-digest-block-pos)*

*(block-digest-block-set!)*

*(block-digest-block)*

*(block-digest-hash-set!)*

*(block-digest-hash)*

*(block-digest-hash-update-set!)*

*(block-digest-hash-update)*

*(block-digest?)*

*(make-block-digest)*

*(hash-block->u8vector)*

*(hash-block->hex-string)*

*(open-digest-crc32)*

*(end-crc32)*

*(digest-update-crc32)*

*(crc32-table)*

*(crc32-digest-lo16-set!)*

*(crc32-digest-lo16)*

*(crc32-digest-hi16-set!)*

*(crc32-digest-hi16)*

*(crc32-digest?)*

*(make-crc32-digest)*

*(digest-state-set!)*

*(digest-state)*

*(digest-update-set!)*

*(digest-update)*

*(digest-end-set!)*

*(digest-end)*

*(digest?)*

*(make-digest)*

*pregexp – Posix Regular Expressions matching engine library*

© Dorai Sitaraam 1999

See <http://www.ccs.neu.edu/home/dorai/pregexp/pregexp.html> .

*(pregexp s)*

*(pregexp-match-positions pat str . opt-args)*

*(pregexp-match pat str . opt-args)*

*(pregexp-split pat str)*

*(pregexp-replace pat str ins)*

*(pregexp-replace\* pat str ins)*

*(pregexp-quote s)*

*(pregexp-match? pattern string)*

### **sxml-to-xml – Fast SXML to XML generator**

© 2005-2008 Christian Jaeger

*(sxml>>html\* item #!optional (port (current-output-port)) maybe-level)*

*(sxml>>xhtml\* item #!optional (port (current-output-port)) maybe-level)*

*(sxml>>xml\* item #!optional (port (current-output-port)) maybe-level)*

*(sxml>>html-fast item #!optional (port (current-output-port)))*

*(sxml>>html item #!optional (port (current-output-port)))*

*(sxml>>xhtml-fast item #!optional (port (current-output-port)))*

*(sxml>>xhtml item #!optional (port (current-output-port)))*

*(sxml>>xml-fast item #!optional (port (current-output-port)))*

*(sxml>>xml item #!optional (port (current-output-port)))*

*(sxml>>pretty-xml-file item outpath #!optional noblanks)*

*(sxml>>pretty-xhtml-file item outpath #!optional noblanks)*

*(sxml->html-string-fragment item #!optional maybe-level)*

*(sxml->xml-string-fragment item #!optional maybe-level)*

**util – String utility procedures**

© Per Eckerdal and Mikael More

***(char->string char)***

```
> (char->string #\a)
"a"
```

***(string-strip str)***

Removes all leading and ending (char-whitespace?):s (that's a R5RS procedure).

```
> (string-strip "\tyo \r\n")
"yo"
```

***(string-replace-char str from to)***

```
> (string-replace-char "hullo" #\u #\e)
"hello"
```

***(join between args)***

```
> (join "a" '(1 2 3))
(1 "a" 2 "a" 3)
> (apply string-append (join " " '("a" "b" "c")))
"a b c"
```

***(string-split chr str #!optional (sparse #f))***

```
> (string-split #\space " wag the dog. ")
(" " "wag" "the" "dog." "")
> (string-split #\space " wag the dog. " #t)
("wag" "the" "dog." "")
```

***(string-split-at-first at-char s #!optional (not-found (lambda () (error "String parameter didn't contain char."))))***

```
> (string-split-at-first #\space "wag the dog")
("wag" . "the dog")
> (string-split-at-first #\space "wag" (lambda () #f))
#f
> (string-split-at-first #\space "wag")
*** ERROR IN "(generated)"@1.1 -- String parameter didn't contain char.
1>
```

***(string-split-at-first-nice at-char s)***

```
> (string-split-at-first-nice #\space "wag the dog")
("wag" . "the dog")
> (string-split-at-first-nice #\space "wag")
#f
```

*(string-name-split str)*

```
> (string-name-split "Bob Doe")
("Bob" "Doe")
> (string-name-split "Bob      Doe ")
("Bob" "Doe")
> (string-name-split " Bob-Doe")
("Bob" "Doe")
```

*(string-set-first-char! fun str)*

```
> (define s "Test")
> (string-set-first-char! (lambda (c) (print "c is " c "\n") #\F) s)
c is T
"Fest"
> s
"Fest"
```

*(string-capitalize! str)* , *(string-decapitalize! str)*

Sets first character to its R5RS `char-upcase` and `char-downcase` correspondent, respectively.

```
> (define s "ttt")
> (string-capitalize! s)
"Ttt"
> s
"Ttt"
> (string-decapitalize! s)
"ttt"
> s
"ttt"
```

*(string-constantize str)* , *(string-camelize str)* , *(string-dasherize str)* , *(string-spaceize str)* , *(string-humanize str)*

```
> (define (test s)
  (for-each
   (lambda (f)
     (eval `(print "(" ',f "\t\t" ,s ")" => "(" (,f ,s) "\"\n"))
     '(string-constantize string-camelize string-dasherize
        string-spaceize string-humanize)))
   s))
> (test "helloWorld")
(string-constantize "helloWorld") => "HelloWorld"
(string-camelize "helloWorld") => "helloWorld"
(string-dasherize "helloWorld") => "hello-world"
(string-spaceize "helloWorld") => "hello world"
(string-humanize "helloWorld") => "Hello world"
> (test "hello-world")
```



```

(string-constantize "hello-world") => "HelloWorld"
(string-camelize    "hello-world") => "helloWorld"
(string-dasherize   "hello-world") => "hello-world"
(string-spaceize    "hello-world") => "hello world"
(string-humanize    "hello-world") => "Hello world"
> (test "hello world")
(string-constantize "hello world") => "HelloWorld"
(string-camelize    "hello world") => "helloWorld"
(string-dasherize   "hello world") => "hello-world"
(string-spaceize    "hello world") => "hello world"
(string-humanize    "hello world") => "Hello world"

```

The point of these procedures is to provide you with an easy way to benefit from multiple naming conventions.

For instance, in Scheme code, you may want variables `string-dasherize:d` (in-this-way), in a database, you may want column names to be in `camel-case` (inThisWay), and in error messages you may want them `string-humanize:d` (In this way).

*(string-remove-suffix haystack needle) , (string-remove-prefix haystack needle)*

```

> (string-remove-suffix "Sr Bob Sr" "Sr")
" Sr Bob "
> (string-remove-prefix "Sr Bob Sr" "Sr")
" Bob Sr"

```

*(string-pluralize str #!optional (num 0)) , (string-depluralize str)*

```

> (string-pluralize "Fish" 1)
"Fish"
> (string-pluralize "Fish" 2)
"Fishes"
> (string-depluralize "Fishes")
"Fish"
> (string-depluralize "Boats")
"Boat"

```

*(symbol-append . syms)*

```

> (symbol-append 'a 'b 'c)
abc

```

*(string-invert str) , (string-uninvert str)*

The inverted form is hexadecimalized, utf8-encoded, inverted. This makes inverted strings sort in the reverse order of regular strings, both as they are or in utf8-encoded hexadecimalized form.

```

> (string-invert "\0abc")
"ff9e9d9c"
> (string-uninvert "ff9e9d9c")
"\0abc"

```

***(dumps . args)***

Produces a fair string representation of the arguments.

```
> (dumps 'a)
"a"
> (dumps "hi")
"hi"
> (dumps '(1 2 3))
"(1 2 3)"
> (dumps (make-u8vector 2))
"#u8(0 0)"
> (dumps "Two is " 2 " and a cons cell is " (cons 1 2))
"Two is 2 and a cons cell is (1 . 2)"
```

**xml-to-sxml – Fast XML string to SXML parser**

© Mikael More

Uses the XML to SXML parser found in SSAX-SXML released public domain by Oleg Kiselyovs. Optimized for speed. Features the complete set of HTML entities.

***html-entity-unicode-numbers***

A-list of HTML entity references and their corresponding unicode numbers, i.e. '(amp . 38) etc.

***html-entity-unicode-chars***

Same as html-entity-unicode-numbers, but car is string rather than symbol, and cdr is a one character long string containing the unicode character, rather than the its integer unicodenumber.

***(xml-string->sxml xml-string #!optional (namespace-prefix-assig '()))***

**Index**

char->string.....	18	string-remove-prefix.....	20
dumps.....	21	string-remove-suffix.....	20
join.....	18	string-replace-char.....	18
string-camelize.....	19	string-set-first-char!.....	19
string-capitalize!.....	19	string-split.....	18
string-constantize.....	19	string-split-at-first.....	18
string-decapitalize!.....	19	string-split-at-first-nice.....	19
string-depluralize.....	20	string-strip.....	18
string-invert.....	21	string-uninvert.....	21
string-name-split.....	19	string/util module (String utility procedures)...	18
string-pluralize.....	20	symbol-append.....	20