

R⁶RS: Standard Language

Version 4.1.2

October 28, 2008

The The Revised⁶ Report on the Algorithmic Language Scheme defines a dialect of Scheme. We use *R⁶RS* to refer to both the standard and the language defined by the standard.

R⁶RS defines both *libraries* and *top-level programs*. Both correspond to PLT Scheme *modules* (see §6 “Modules”). That is, although R⁶RS defines top-level programs as entry points, you can just as easily treat a library as an entry point when using PLT Scheme. The only difference is that an R⁶RS top-level program cannot export any bindings to other modules.

Contents

1	Running Top-Level Programs	4
2	Installing Libraries	5
3	Libraries and Collections	6
4	Scheme Interoperability	7
5	R⁶RS Conformance	8
6	R⁶RS Libraries	10
6.1	(<code>rnrs base (6)</code>): Base	10
6.2	(<code>rnrs unicode (6)</code>): Unicode	10
6.3	(<code>rnrs bytevectors (6)</code>): Bytevectors	10
6.4	(<code>rnrs lists (6)</code>): List utilities	10
6.5	(<code>rnrs sorting (6)</code>): Sorting	10
6.6	(<code>rnrs control (6)</code>): Control Structures	10
6.7	(<code>rnrs records syntactic (6)</code>): Records: Syntactic	11
6.8	(<code>rnrs records procedural (6)</code>): Records: Procedural	11
6.9	(<code>rnrs records inspection (6)</code>): Records: Inspection	11
6.10	(<code>rnrs exceptions (6)</code>): Exceptions	11
6.11	(<code>rnrs conditions (6)</code>): Conditions	11
6.12	(<code>rnrs io ports (6)</code>): I/O: Ports	11
6.13	(<code>rnrs io simple (6)</code>): I/O: Simple	12
6.14	(<code>rnrs files (6)</code>): File System	12
6.15	(<code>rnrs programs (6)</code>): Command-line Access and Exit Values	12

6.16	<code>(rnrs arithmetic fixnums (6))</code> : Arithmetic: Fixnums	12
6.17	<code>(rnrs arithmetic flonums (6))</code> : Arithmetic: Flonums	12
6.18	<code>(rnrs arithmetic bitwise (6))</code> : Arithmetic: Bitwise	12
6.19	<code>(rnrs syntax-case (6))</code> : Syntax-Case	13
6.20	<code>(rnrs hashtables (6))</code> : Hashtables	13
6.21	<code>(rnrs enums (6))</code> : Enumerations	13
6.22	<code>(rnrs eval (6))</code> : Eval	13
6.23	<code>(rnrs mutable-pairs (6))</code> : Mutable Pairs	13
6.24	<code>(rnrs mutable-strings (6))</code> : Mutable Strings	13
6.25	<code>(rnrs r5rs (6))</code> : R5RS Compatibility	14

Index		15
--------------	--	-----------

1 Running Top-Level Programs

To run a top-level program, either:

- Use the `plt-r6rs` executable, supplying the file that contains the program on the command line:

```
plt-r6rs <program-file>
```

Additional command-line arguments are propagated as command-line arguments to the program (accessed via [command-line](#)).

To compile the file to bytecode (to speed future runs of the program), use `plt-r6rs` with the `--compile` flag:

```
plt-r6rs --compile <program-file>
```

The bytecode file is written in a "compiled" sub-directory next to *<program-file>*.

For example, if "hi.scm" contains

```
(import (rnrs))
(display "hello\n")
then
  plt-r6rs hi.scm
prints "hello."
```

- Prefix the program with `#!r6rs`, which counts as a comment from the R⁶RS perspective, but is a synonym for `#lang r6rs` from the PLT Scheme perspective. Such files can be run like any other PLT Scheme module, such as using `mzscheme`:

```
mzscheme <program-file>
```

or using DrScheme with the Module language. The file can also be compiled to bytecode using `mzc`:

```
mzc <program-file>
```

For example, if "hi.ss" contains

```
#!r6rs
(import (rnrs))
(display "hello\n")
then
  mzscheme hi.ss
```

prints "hello." Similarly, opening "hi.ss" in DrScheme and clicking Run prints "hello" within the DrScheme interactions window.

2 Installing Libraries

To reference an R⁶RS library from a top-level program or another library, it must be installed as a collection-based library in PLT Scheme.

One way to produce an R⁶RS installed library is to create in a collection a file that starts with `#!r6rs` and that contains a `library` form. For example, the following file might be created in a "hello.ss" file within a "examples" collection directory:

```
#!r6rs
(library (examples hello)
 (export greet)
 (import (rnrs)))

(define (greet)
 (display "hello\n"))
```

Alternately, the `plt-r6rs` executable with the `--install` flag accepts a sequence of `library` declarations and installs them into separate files in a collection directory, based on the declared name of each library:

```
plt-r6rs --install <libraries-file>
```

By default, libraries are installed into the user-specific collection directory (see `find-user-collects-dir`). The `--all-users` flag causes the libraries to be installed into the main installation, instead (see `find-collects-dir`):

```
plt-r6rs --install --all-users <libraries-file>
```

See §3 “Libraries and Collections” for information on how R⁶RS library names are turned into collection-based module paths, which determines where the files are written. Libraries installed by `plt-r6rs --install` are automatically compiled to bytecode form.

One final option is to supply a `++path` flag to `plt-r6rs`. A path added with `++path` extends the set of directories that are searched to find a collection (i.e., it sets `current-library-collection-paths`). If `<dir>` contains "duck" and "cow" sub-directories with "duck/feather.sls" and "cow/bell.sls", and if each file is an R⁶RS library prefixed with `#!r6rs`, then `plt-r6rs ++path <dir>` directs the R⁶RS library references `(duck feather)` and `(cow bell)` to the files. Note that this technique does not support accessing "duck.sls" directly within `<dir>`, since the library reference `(duck)` is treated like `(duck main)` for finding the library, as explained in §3 “Libraries and Collections”. Multiple paths can be provided with multiple uses of `++path`; the paths are searched in order, and before the installation’s collections.

3 Libraries and Collections

An R⁶RS library name is sequence of symbols, optionally followed by a version as a sequence of exact, non-negative integers. Roughly, such a name is converted to a PLT Scheme module pathname (see §6.3 “Module Paths”) by concatenating the symbols with a `/` separator, and then appending the version integers each with a preceding `-`. As a special case, when an R⁶RS path contains a single symbol (optionally followed by a version), a `main` symbol is effectively inserted after the initial symbol. See below for further encoding considerations.

When an R⁶RS library or top-level program refers to another library, it can supply version constraints rather than naming a specific version. Version constraints are always resolved at compile time by searching the set of installed files.

In addition, when an R⁶RS library path is converted, a file extension is selected at compile time based on installed files. The search order for file extensions is `".mzscheme.ss"`, `".mzscheme.sls"`, `".ss"`, and `".sls"`. When resolving version constraints, these extensions are all tried when looking for matches.

To ensure that all R⁶RS library names can be converted to a unique and distinct library module path, the following conversions are applied to each symbol before concatenating them:

- The symbol is encoded using UTF-8, and the resulting bytes are treated as Latin-1 encoded characters. ASCII letters, digits, `+`, `-`, and `_` are left as-is; other characters are replaced by `%` followed by two lowercase hexadecimal digits. Note that UTF-8 encodes ASCII letters, digits, etc. as themselves, so typical library names correspond to readable module paths.
- If the R⁶RS library reference has two symbol elements and the second one is `main` followed by any number of underscores, then an extra underscore is added to that symbol. This conversion avoids a collision between an explicit `main` and the implicit `main` when a library path has a single symbol element.

Examples (assuming a typical PLT Scheme installation):

```
(rnrs io simple (6)) means (lib "rnrs/io/simple-6.ss")
(rnrs)                means (lib "rnrs/main-6.ss")
(rnrs main)          means (lib "rnrs/main_.ss")
(rnrs (6))           means (lib "rnrs/main-6.ss")
(scheme base)        means (lib "scheme/base.ss")
(achtung!)            means (lib "achtung%21/main.ss")
(funco new-λ)        means (lib "funco/new-%ce%bb.ss")
```

4 Scheme Interoperability

Using the conversion rules in §3 “Libraries and Collections”, and R⁶RS library can refer to modules that are implemented in other dialects supported by PLT Scheme, and other PLT Scheme modules can refer to libraries that are implemented in R⁶RS.

Beware that a *pair* in R⁶RS corresponds to a *mutable pair* in `scheme/base`. Otherwise, R⁶RS libraries and `scheme/base` share the same datatype for numbers, characters, strings, bytevectors (a.k.a. byte strings), vectors, and so on. Hash tables are different. Input and output ports from `scheme/base` can be used directly as binary ports with R⁶RS libraries, and all R⁶RS ports can be used as ports in `scheme/base` programs, but only textual ports created via R⁶RS libraries can be used by other R⁶RS operations that expect textual ports.

5 R⁶RS Conformance

PLT Scheme's R⁶RS support does not conform with the standard in several known ways:

- When guard catches an exception that no clause matches, the exception is re-`raised` without restoring the continuation to the one that raised the exception.

This difference can be made visible using `dynamic-wind`. According to R⁶RS, the following program should print “in” and “out” twice, but each prints once using PLT Scheme:

```
(guard (exn [(equal? exn 5) 'five]))
  (guard (exn [(equal? exn 6) 'six]))
    (dynamic-wind
      (lambda () (display "in") (newline))
      (lambda () (raise 5))
      (lambda () (display "out") (newline))))))
```

Along similar lines, continuation capture and invocation within an exception handler is restricted. Unless the exception is raised through `raise-continuable`, a handler can escape only through a continuation that is a tail of the current continuation, and a continuation captured within the handler cannot be invoked after control escapes from the raise.

The initial exception handler does not return for non-`&serious` conditions, but `raise` and `raise-continuable` both install an uncaught-exception handler (via `parameterize` and `uncaught-exception-handler`) to one that returns for non-`&serious` conditions.

- Inexact numbers are printed without a precision indicator, and precision indicators are ignored on input (e.g., `0.5|7` is read the same as `0.5`).
- Word boundaries for `string-downcase`, `string-upcase`, and `string-titlecase` are not determined as specified by Unicode Standard Annex #29.
- When an identifier bound by `letrec` or `letrec*` is referenced before it is bound, an exception is not raised; instead, the reference produces `#<undefined>`.
- A custom textual port must represent positions using integers, and the positions must correspond to bytes in a UTF-8 encoding of the port's data. For custom ports (byte or character) that support both input and output, beware that buffered input can create a mismatch between the position implemented by the custom procedures and the port's current position; the result from a custom position procedure is automatically adjusted to account for buffering, and setting the port's position flushes all buffered bytes, but writing after a read does *not* automatically reset the port's position to counteract the effects of buffering.

- The bindings in a namespace produced by `null-environment` or `scheme-report-environment` correspond to R⁵RS bindings instead of R⁶RS bindings. In particular, `=>`, `else`, `_`, and `...` are not bound.

6 R⁶RS Libraries

6.1 `(rnrs base (6))`: Base

`(require rnrs/base-6)`

Original specification: Base

6.2 `(rnrs unicode (6))`: Unicode

`(require rnrs/unicode-6)`

Original specification: Unicode

6.3 `(rnrs bytevectors (6))`: Bytevectors

`(require rnrs/bytevectors-6)`

Original specification: Bytevectors

6.4 `(rnrs lists (6))`: List utilities

`(require rnrs/lists-6)`

Original specification: List utilities

6.5 `(rnrs sorting (6))`: Sorting

`(require rnrs/sorting-6)`

Original specification: Sorting

6.6 `(rnrs control (6))`: Control Structures

`(require rnrs/control-6)`

Original specification: Control Structures

6.7 `(nrns records syntactic (6))`: **Records: Syntactic**

`(require nrns/records/syntactic-6)`

Original specification: Records: Syntactic

6.8 `(nrns records procedural (6))`: **Records: Procedural**

`(require nrns/records/procedural-6)`

Original specification: Records: Procedural

6.9 `(nrns records inspection (6))`: **Records: Inspection**

`(require nrns/records/inspection-6)`

Original specification: Records: Inspection

6.10 `(nrns exceptions (6))`: **Exceptions**

`(require nrns/exceptions-6)`

Original specification: Exceptions

See also §5 “R⁶RS Conformance”.

6.11 `(nrns conditions (6))`: **Conditions**

`(require nrns/conditions-6)`

Original specification: Conditions

6.12 `(nrns io ports (6))`: **I/O: Ports**

`(require nrns/io/ports-6)`

Original specification: I/O: Ports

6.13 `(rnrs io simple (6))`: **I/O: Simple**

`(require rnrs/io/simple-6)`

Original specification: I/O: Simple

6.14 `(rnrs files (6))`: **File System**

`(require rnrs/files-6)`

Original specification: File System

6.15 `(rnrs programs (6))`: **Command-line Access and Exit Values**

`(require rnrs/programs-6)`

Original specification: Command-line Access and Exit Values

6.16 `(rnrs arithmetic fixnums (6))`: **Arithmetic: Fixnums**

`(require rnrs/arithmetic/fixnums-6)`

Original specification: Arithmetic: Fixnums

6.17 `(rnrs arithmetic flonums (6))`: **Arithmetic: Flonums**

`(require rnrs/arithmetic/flonums-6)`

Original specification: Arithmetic: Flonums

6.18 `(rnrs arithmetic bitwise (6))`: **Arithmetic: Bitwise**

`(require rnrs/arithmetic/bitwise-6)`

Original specification: Arithmetic: Bitwise

6.19 `(rnrs syntax-case (6))`: Syntax-Case

```
(require rnrs/syntax-case-6)
```

Original specification: Syntax-Case

6.20 `(rnrs hashtables (6))`: Hashtables

```
(require rnrs/hashtables-6)
```

Original specification: Hashtables

A hashtable is a dictionary in the sense of `scheme/dict`.

6.21 `(rnrs enums (6))`: Enumerations

```
(require rnrs/enums-6)
```

Original specification: Enumerations

6.22 `(rnrs eval (6))`: Eval

```
(require rnrs/eval-6)
```

Original specification: Eval

6.23 `(rnrs mutable-pairs (6))`: Mutable Pairs

```
(require rnrs/mutable-pairs-6)
```

Original specification: Mutable Pairs

6.24 `(rnrs mutable-strings (6))`: Mutable Strings

```
(require rnrs/mutable-strings-6)
```

Original specification: Mutable Strings

6.25 `(rnrs r5rs (6))`: R5RS Compatibility

`(require rnrs/r5rs-6)`

Original specification: R5RS Compatibility

See also §5 “R⁶RS Conformance”.

Index

`&assertion`, 11
`&condition`, 11
`&error`, 11
`&i/o`, 11
`&i/o-decoding`, 11
`&i/o-encoding`, 11
`&i/o-file-already-exists`, 11
`&i/o-file-does-not-exist`, 11
`&i/o-file-is-read-only`, 11
`&i/o-file-protection`, 11
`&i/o-filename`, 11
`&i/o-invalid-position`, 11
`&i/o-port`, 11
`&i/o-read`, 11
`&i/o-write`, 11
`&implementation-restriction`, 11
`&irritants`, 11
`&lexical`, 11
`&message`, 11
`&no-infinities`, 12
`&no-nans`, 12
`&non-continuable`, 11
`&serious`, 11
`&syntax`, 11
`&undefined`, 11
`&violation`, 11
`&warning`, 11
`&who`, 11
`(rnrs arithmetic bitwise (6))`: Arithmetic: Bitwise, 12
`(rnrs arithmetic fixnums (6))`: Arithmetic: Fixnums, 12
`(rnrs arithmetic flonums (6))`: Arithmetic: Flonums, 12
`(rnrs base (6))`: Base, 10
`(rnrs bytevectors (6))`: Bytevectors, 10
`(rnrs conditions (6))`: Conditions, 11
`(rnrs control (6))`: Control Structures, 10
`(rnrs enums (6))`: Enumerations, 13
`(rnrs eval (6))`: Eval, 13
`(rnrs exceptions (6))`: Exceptions, 11
`(rnrs files (6))`: File System, 12
`(rnrs hashtables (6))`: Hashtables, 13
`(rnrs io ports (6))`: I/O: Ports, 11
`(rnrs io simple (6))`: I/O: Simple, 12
`(rnrs lists (6))`: List utilities, 10
`(rnrs mutable-pairs (6))`: Mutable Pairs, 13
`(rnrs mutable-strings (6))`: Mutable Strings, 13
`(rnrs programs (6))`: Command-line Access and Exit Values, 12
`(rnrs r5rs (6))`: R5RS Compatibility, 14
`(rnrs records inspection (6))`: Records: Inspection, 11
`(rnrs records procedural (6))`: Records: Procedural, 11
`(rnrs records syntactic (6))`: Records: Syntactic, 11
`(rnrs sorting (6))`: Sorting, 10
`(rnrs syntax-case (6))`: Syntax-Case, 13
`(rnrs unicode (6))`: Unicode, 10
`*`, 10
`+`, 10
`++path`, 5
`-`, 10
`...`, 10
`...`, 13
`/`, 10
`<`, 10
`<=`, 10
`=`, 10
`=>`, 10
`=>`, 11
`>`, 10
`>=`, 10
`→`, 10
`→`, 13
`abs`, 10
`acos`, 10

and, 10
 angle, 10
 append, 10
 apply, 10
 asin, 10
 assert, 10
 assertion-violation, 10
 assertion-violation?, 11
 assoc, 10
 assp, 10
 assq, 10
 assv, 10
 atan, 10
 begin, 10
 binary-port?, 11
 bitwise-and, 12
 bitwise-arithmetic-shift, 12
 bitwise-arithmetic-shift-left, 12
 bitwise-arithmetic-shift-right, 12
 bitwise-bit-count, 12
 bitwise-bit-field, 12
 bitwise-bit-set?, 12
 bitwise-copy-bit, 12
 bitwise-copy-bit-field, 12
 bitwise-first-bit-set, 12
 bitwise-if, 12
 bitwise-ior, 12
 bitwise-length, 12
 bitwise-not, 12
 bitwise-reverse-bit-field, 12
 bitwise-rotate-bit-field, 12
 bitwise-xor, 12
 boolean=?, 10
 boolean?, 10
 bound-identifier=?, 13
 buffer-mode, 11
 buffer-mode?, 11
 bytevector->sint-list, 10
 bytevector->string, 11
 bytevector->u8-list, 10
 bytevector->uint-list, 10
 bytevector-copy, 10
 bytevector-copy!, 10
 bytevector-fill!, 10
 bytevector-ieee-double-native-ref,
 10
 bytevector-ieee-double-native-
 set!, 10
 bytevector-ieee-double-ref, 10
 bytevector-ieee-single-native-ref,
 10
 bytevector-ieee-single-native-
 set!, 10
 bytevector-ieee-single-ref, 10
 bytevector-length, 10
 bytevector-s16-native-ref, 10
 bytevector-s16-native-set!, 10
 bytevector-s16-ref, 10
 bytevector-s16-set!, 10
 bytevector-s32-native-ref, 10
 bytevector-s32-native-set!, 10
 bytevector-s32-ref, 10
 bytevector-s32-set!, 10
 bytevector-s64-native-ref, 10
 bytevector-s64-native-set!, 10
 bytevector-s64-ref, 10
 bytevector-s64-set!, 10
 bytevector-s8-ref, 10
 bytevector-s8-set!, 10
 bytevector-sint-ref, 10
 bytevector-sint-set!, 10
 bytevector-u16-native-ref, 10
 bytevector-u16-native-set!, 10
 bytevector-u16-ref, 10
 bytevector-u16-set!, 10
 bytevector-u32-native-ref, 10
 bytevector-u32-native-set!, 10
 bytevector-u32-ref, 10
 bytevector-u32-set!, 10
 bytevector-u64-native-ref, 10
 bytevector-u64-native-set!, 10
 bytevector-u64-ref, 10
 bytevector-u64-set!, 10
 bytevector-u8-ref, 10

bytevector-u8-set!, 10
 bytevector-uint-ref, 10
 bytevector-uint-set!, 10
 bytevector=?, 10
 bytevector?, 10
 caar, 10
 cadr, 10
 call-with-bytevector-output-port,
 11
 call-with-current-continuation, 10
 call-with-input-file, 12
 call-with-output-file, 12
 call-with-port, 11
 call-with-string-output-port, 11
 call-with-values, 10
 call/cc, 10
 car, 10
 case, 10
 case-lambda, 10
 cdddar, 10
 cddddr, 10
 cdr, 10
 ceiling, 10
 char->integer, 10
 char-alphabetic?, 10
 char-ci<=?, 10
 char-ci<?, 10
 char-ci=?, 10
 char-ci>=?, 10
 char-ci>?, 10
 char-downcase, 10
 char-foldcase, 10
 char-general-category, 10
 char-lower-case?, 10
 char-numeric?, 10
 char-title-case?, 10
 char-titlecase, 10
 char-upcase, 10
 char-upper-case?, 10
 char-whitespace?, 10
 char<=?, 10
 char<?, 10
 char=?, 10
 char>=?, 10
 char>?, 10
 char?, 10
 close-input-port, 12
 close-output-port, 12
 close-port, 11
 command-line, 12
 complex?, 10
 cond, 10
 condition, 11
 condition-accessor, 11
 condition-irritants, 11
 condition-message, 11
 condition-predicate, 11
 condition-who, 11
 condition?, 11
 cons, 10
 cons*, 10
 cos, 10
 current-error-port, 11
 current-input-port, 11
 current-output-port, 11
 datum->syntax, 13
 define, 10
 define-condition-type, 11
 define-enumeration, 13
 define-record-type, 11
 define-syntax, 10
 delay, 14
 delete-file, 12
 denominator, 10
 display, 12
 div, 10
 div-and-mod, 10
 div0, 10
 div0-and-mod0, 10
 do, 10
 dynamic-wind, 10
 else, 10
 else, 11
 endianness, 10

enum-set->list, 13
enum-set-complement, 13
enum-set-constructor, 13
enum-set-difference, 13
enum-set-indexer, 13
enum-set-intersection, 13
enum-set-member?, 13
enum-set-projection, 13
enum-set-subset?, 13
enum-set-union, 13
enum-set-universe, 13
enum-set=?, 13
environment, 13
eof-object, 11
eof-object?, 11
eol-style, 11
eq?, 10
equal-hash, 13
equal?, 10
eqv?, 10
error, 10
error-handling-mode, 11
error?, 11
eval, 13
even?, 10
exact, 10
exact->inexact, 14
exact-integer-sqrt, 10
exact?, 10
exists, 10
exit, 12
exp, 10
expt, 10
fields, 11
file-exists?, 12
file-options, 11
filter, 10
find, 10
finite?, 10
fixnum->flonum, 12
fixnum-width, 12
fixnum?, 12
fl*, 12
fl+, 12
fl-, 12
fl/, 12
fl<=?, 12
fl<?, 12
fl=?, 12
fl>=?, 12
fl>?, 12
flabs, 12
flacos, 12
flasin, 12
flatan, 12
flceiling, 12
flcos, 12
fldenominator, 12
fldiv, 12
fldiv-and-mod, 12
fldiv0, 12
fldiv0-and-mod0, 12
fleven?, 12
flexp, 12
flexpt, 12
flfinite?, 12
flfloor, 12
flinfinite?, 12
flinteger?, 12
fllog, 12
flmax, 12
flmin, 12
flmod, 12
flmod0, 12
flnan?, 12
flnegative?, 12
flnumerator, 12
flodd?, 12
flonum?, 12
floor, 10
flpositive?, 12
flround, 12
flsin, 12
flsqrt, 12

fltan, 12
 fltruncate, 12
 flush-output-port, 11
 flzero?, 12
 fold-left, 10
 fold-right, 10
 for-all, 10
 for-each, 10
 force, 14
 free-identifier=?, 13
 fx*, 12
 fx*/carry, 12
 fx+, 12
 fx+/carry, 12
 fx-, 12
 fx-/carry, 12
 fx<=?, 12
 fx<?, 12
 fx=?, 12
 fx>=?, 12
 fx>?, 12
 fxand, 12
 fxarithmetic-shift, 12
 fxarithmetic-shift-left, 12
 fxarithmetic-shift-right, 12
 fxbit-count, 12
 fxbit-field, 12
 fxbit-set?, 12
 fxcopy-bit, 12
 fxcopy-bit-field, 12
 fxdiv, 12
 fxdiv-and-mod, 12
 fxdiv0, 12
 fxdiv0-and-mod0, 12
 fxeven?, 12
 fxfirst-bit-set, 12
 fxif, 12
 fxior, 12
 fxlength, 12
 fxmax, 12
 fxmin, 12
 fxmod, 12
 fxmod0, 12
 fxnegative?, 12
 fxnot, 12
 fxodd?, 12
 fxpositive?, 12
 fxreverse-bit-field, 12
 fxrotate-bit-field, 12
 fxxor, 12
 fxzero?, 12
 gcd, 10
 generate-temporaries, 13
 get-bytevector-all, 11
 get-bytevector-n, 11
 get-bytevector-n!, 11
 get-bytevector-some, 11
 get-char, 11
 get-datum, 11
 get-line, 11
 get-string-all, 11
 get-string-n, 11
 get-string-n!, 11
 get-u8, 11
 greatest-fixnum, 12
 guard, 11
 hashtable-clear!, 13
 hashtable-contains?, 13
 hashtable-copy, 13
 hashtable-delete!, 13
 hashtable-entries, 13
 hashtable-equivalence-function, 13
 hashtable-hash-function, 13
 hashtable-keys, 13
 hashtable-mutable?, 13
 hashtable-ref, 13
 hashtable-set!, 13
 hashtable-size, 13
 hashtable-update!, 13
 hashtable?, 13
 i/o-decoding-error?, 11
 i/o-encoding-error-char, 11
 i/o-encoding-error?, 11
 i/o-error-filename, 11

[i/o-error-port](#), 11
[i/o-error-position](#), 11
[i/o-error?](#), 11
[i/o-file-already-exists-error?](#), 11
[i/o-file-does-not-exist-error?](#), 11
[i/o-file-is-read-only-error?](#), 11
[i/o-file-protection-error?](#), 11
[i/o-filename-error?](#), 11
[i/o-invalid-position-error?](#), 11
[i/o-port-error?](#), 11
[i/o-read-error?](#), 11
[i/o-write-error?](#), 11
[identifier-syntax](#), 10
[identifier?](#), 13
[if](#), 10
[image-part](#), 10
[immutable](#), 11
[implementation-restriction-violation?](#), 11
[inexact](#), 10
[inexact->exact](#), 14
[inexact?](#), 10
[infinite?](#), 10
[input-port?](#), 11
[Installing Libraries](#), 5
[integer->char](#), 10
[integer-valued?](#), 10
[integer?](#), 10
[irritants-condition?](#), 11
[lambda](#), 10
[latin-1-codec](#), 11
[lcm](#), 10
[least-fixnum](#), 12
[length](#), 10
[let](#), 10
[let*](#), 10
[let*-values](#), 10
[let-syntax](#), 10
[let-values](#), 10
[letrec](#), 10
[letrec*](#), 10
[letrec-syntax](#), 10
[lexical-violation?](#), 11
[Libraries and Collections](#), 6
[list](#), 10
[list->string](#), 10
[list->vector](#), 10
[list-ref](#), 10
[list-sort](#), 10
[list-tail](#), 10
[list?](#), 10
[log](#), 10
[lookahead-char](#), 11
[lookahead-u8](#), 11
[magnitude](#), 10
[make-assertion-violation](#), 11
[make-bytevector](#), 10
[make-custom-binary-input-port](#), 11
[make-custom-binary-input/output-port](#), 11
[make-custom-binary-output-port](#), 11
[make-custom-textual-input-port](#), 11
[make-custom-textual-input/output-port](#), 11
[make-custom-textual-output-port](#), 11
[make-enumeration](#), 13
[make-eq-hashtable](#), 13
[make-eqv-hashtable](#), 13
[make-error](#), 11
[make-hashtable](#), 13
[make-i/o-decoding-error](#), 11
[make-i/o-encoding-error](#), 11
[make-i/o-error](#), 11
[make-i/o-file-already-exists-error](#), 11
[make-i/o-file-does-not-exist-error](#), 11
[make-i/o-file-is-read-only-error](#), 11
[make-i/o-file-protection-error](#), 11
[make-i/o-filename-error](#), 11
[make-i/o-invalid-position-error](#), 11
[make-i/o-port-error](#), 11
[make-i/o-read-error](#), 11

make-i/o-write-error, 11
 make-implementation-restriction-violation, 11
 make-irritants-condition, 11
 make-lexical-violation, 11
 make-message-condition, 11
 make-no-infinities-violation, 12
 make-no-nans-violation, 12
 make-non-continuable-violation, 11
 make-polar, 10
 make-record-constructor-descriptor, 11
 make-record-type-descriptor, 11
 make-rectangular, 10
 make-serious-condition, 11
 make-string, 10
 make-syntax-violation, 11
 make-transcoder, 11
 make-undefined-violation, 11
 make-variable-transformer, 13
 make-vector, 10
 make-violation, 11
 make-warning, 11
 make-who-condition, 11
 map, 10
 max, 10
 member, 10
 memp, 10
 memq, 10
 memv, 10
 message-condition?, 11
 min, 10
 mod, 10
 mod0, 10
 modulo, 14
 mutable, 11
 nan?, 10
 native-endianness, 10
 native-eol-style, 11
 native-transcoder, 11
 negative?, 10
 newline, 12
 no-infinities-violation?, 12
 no-nans-violation?, 12
 non-continuable-violation?, 11
 nongenerative, 11
 not, 10
 null-environment, 14
 null?, 10
 number->string, 10
 number?, 10
 numerator, 10
 odd?, 10
 opaque, 11
 open-bytevector-input-port, 11
 open-bytevector-output-port, 11
 open-file-input-port, 11
 open-file-input/output-port, 11
 open-file-output-port, 11
 open-input-file, 12
 open-output-file, 12
 open-string-input-port, 11
 open-string-output-port, 11
 or, 10
 output-port-buffer-mode, 11
 output-port?, 11
 pair?, 10
 parent, 11
 parent-rtd, 11
 partition, 10
 peek-char, 12
 port-eof?, 11
 port-has-port-position?, 11
 port-has-set-port-position!?, 11
 port-position, 11
 port-transcoder, 11
 port?, 11
 positive?, 10
 procedure?, 10
 protocol, 11
 put-bytevector, 11
 put-char, 11
 put-datum, 11
 put-string, 11

[put-u8](#), 11
[quasiquote](#), 10
[quasisyntax](#), 13
[quote](#), 10
[quotient](#), 14
[R⁶RS Conformance](#), 8
[R⁶RS Libraries](#), 10
R⁶RS: Standard Language, 1
[raise](#), 11
[raise-continuable](#), 11
[rational-valued?](#), 10
[rational?](#), 10
[rationalize](#), 10
[read](#), 12
[read-char](#), 12
[real->flonum](#), 12
[real-part](#), 10
[real-valued?](#), 10
[real?](#), 10
[record-accessor](#), 11
[record-constructor](#), 11
[record-constructor-descriptor](#), 11
[record-field-mutable?](#), 11
[record-mutator](#), 11
[record-predicate](#), 11
[record-rtd](#), 11
[record-type-descriptor](#), 11
[record-type-descriptor?](#), 11
[record-type-field-names](#), 11
[record-type-generative?](#), 11
[record-type-name](#), 11
[record-type-opaque?](#), 11
[record-type-parent](#), 11
[record-type-sealed?](#), 11
[record-type-uid](#), 11
[record?](#), 11
[remainder](#), 14
[remove](#), 10
[remp](#), 10
[remq](#), 10
[remv](#), 10
[reverse](#), 10
[rnrs/arithmetic/bitwise-6](#), 12
[rnrs/arithmetic/fixnums-6](#), 12
[rnrs/arithmetic/flonums-6](#), 12
[rnrs/base-6](#), 10
[rnrs/bytevectors-6](#), 10
[rnrs/conditions-6](#), 11
[rnrs/control-6](#), 10
[rnrs/enums-6](#), 13
[rnrs/eval-6](#), 13
[rnrs/exceptions-6](#), 11
[rnrs/files-6](#), 12
[rnrs/hashtables-6](#), 13
[rnrs/io/ports-6](#), 11
[rnrs/io/simple-6](#), 12
[rnrs/lists-6](#), 10
[rnrs/mutable-pairs-6](#), 13
[rnrs/mutable-strings-6](#), 13
[rnrs/programs-6](#), 12
[rnrs/r5rs-6](#), 14
[rnrs/records/inspection-6](#), 11
[rnrs/records/procedural-6](#), 11
[rnrs/records/syntactic-6](#), 11
[rnrs/sorting-6](#), 10
[rnrs/syntax-case-6](#), 13
[rnrs/unicode-6](#), 10
[round](#), 10
[Running Top-Level Programs](#), 4
[Scheme Interoperability](#), 7
[scheme-report-environment](#), 14
[sealed](#), 11
[serious-condition?](#), 11
[set!](#), 10
[set-car!](#), 13
[set-cdr!](#), 13
[set-port-position!](#), 11
[simple-conditions](#), 11
[sin](#), 10
[sint-list->bytevector](#), 10
[sqrt](#), 10
[standard-error-port](#), 11
[standard-input-port](#), 11
[standard-output-port](#), 11

string, 10
 string->bytevector, 11
 string->list, 10
 string->number, 10
 string->symbol, 10
 string->utf16, 10
 string->utf32, 10
 string->utf8, 10
 string-append, 10
 string-ci-hash, 13
 string-ci<=?, 10
 string-ci<?, 10
 string-ci=?, 10
 string-ci>=?, 10
 string-ci>?, 10
 string-copy, 10
 string-downcase, 10
 string-fill!, 13
 string-foldcase, 10
 string-for-each, 10
 string-hash, 13
 string-length, 10
 string-normalize-nfc, 10
 string-normalize-nfd, 10
 string-normalize-nfkc, 10
 string-normalize-nfkd, 10
 string-ref, 10
 string-set!, 13
 string-titlecase, 10
 string-upcase, 10
 string<=?, 10
 string<?, 10
 string=?, 10
 string>=?, 10
 string>?, 10
 string?, 10
 substring, 10
 symbol->string, 10
 symbol-hash, 13
 symbol=?, 10
 symbol?, 10
 syntax, 13
 syntax->datum, 13
 syntax-case, 13
 syntax-rules, 10
 syntax-violation, 13
 syntax-violation-form, 11
 syntax-violation-subform, 11
 syntax-violation?, 11
 tan, 10
 textual-port?, 11
 transcoded-port, 11
 transcoder-codec, 11
 transcoder-eol-style, 11
 transcoder-error-handling-mode, 11
 truncate, 10
 u8-list->bytevector, 10
 uint-list->bytevector, 10
 undefined-violation?, 11
 unless, 10
 unquote, 10
 unquote-splicing, 10
 unsyntax, 13
 unsyntax-splicing, 13
 utf-16-codec, 11
 utf-8-codec, 11
 utf16->string, 10
 utf32->string, 10
 utf8->string, 10
 values, 10
 vector, 10
 vector->list, 10
 vector-fill!, 10
 vector-for-each, 10
 vector-length, 10
 vector-map, 10
 vector-ref, 10
 vector-set!, 10
 vector-sort, 10
 vector-sort!, 10
 vector?, 10
 violation?, 11
 warning?, 11
 when, 10

`who-condition?`, 11
`with-exception-handler`, 11
`with-input-from-file`, 12
`with-output-to-file`, 12
`with-syntax`, 13
`write`, 12
`write-char`, 12
`zero?`, 10