

R6RS: Scheme

Version 5.1.3

August 15, 2011

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 Racket *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 Racket. The only difference is that an *R⁶RS* top-level program cannot export any bindings to other modules.

See §22 “Dialects of Racket and Scheme” for general information about different dialects of Scheme within Racket.

Contents

1	Using R⁶RS with DrRacket	4
2	Running Top-Level Programs	5
3	Installing Libraries	6
4	R⁶RS Module Language	7
4.1	Using R ⁶ RS	7
4.2	The Implementation of R ⁶ RS	7
5	Libraries and Collections	8
6	Language Interoperability	9
7	R⁶RS Conformance	10
8	R⁶RS Libraries	12
8.1	<code>(rnrs base (6))</code> : Base	12
8.2	<code>(rnrs unicode (6))</code> : Unicode	12
8.3	<code>(rnrs bytevectors (6))</code> : Bytevectors	12
8.4	<code>(rnrs lists (6))</code> : List utilities	12
8.5	<code>(rnrs sorting (6))</code> : Sorting	12
8.6	<code>(rnrs control (6))</code> : Control Structures	12
8.7	<code>(rnrs records syntactic (6))</code> : Records: Syntactic	13
8.8	<code>(rnrs records procedural (6))</code> : Records: Procedural	13
8.9	<code>(rnrs records inspection (6))</code> : Records: Inspection	13
8.10	<code>(rnrs exceptions (6))</code> : Exceptions	13

8.11	(<code>rnrs conditions</code> (6)):	Conditions	13
8.12	(<code>rnrs io ports</code> (6)):	I/O: Ports	13
8.13	(<code>rnrs io simple</code> (6)):	I/O: Simple	14
8.14	(<code>rnrs files</code> (6)):	File System	14
8.15	(<code>rnrs programs</code> (6)):	Command-line Access and Exit Values	14
8.16	(<code>rnrs arithmetic fixnums</code> (6)):	Arithmetic: Fixnums	14
8.17	(<code>rnrs arithmetic flonums</code> (6)):	Arithmetic: Flonums	14
8.18	(<code>rnrs arithmetic bitwise</code> (6)):	Arithmetic: Bitwise	14
8.19	(<code>rnrs syntax-case</code> (6)):	Syntax-Case	15
8.20	(<code>rnrs hashtables</code> (6)):	Hashtables	15
8.21	(<code>rnrs enums</code> (6)):	Enumerations	15
8.22	(<code>rnrs eval</code> (6)):	Eval	15
8.23	(<code>rnrs mutable-pairs</code> (6)):	Mutable Pairs	15
8.24	(<code>rnrs mutable-strings</code> (6)):	Mutable Strings	15
8.25	(<code>rnrs r5rs</code> (6)):	R5RS Compatibility	16

Index			17
--------------	--	--	-----------

1 Using R⁶RS with DrRacket

To run an R⁶RS program with DrRacket choose Use language declared in source from the language dialog box and add the following line to the top of your program. `#!r6rs`.

Here is a small example R⁶RS program that will work in DrRacket.

```
#!r6rs
(import (rnrs lists (6))
        (rnrs base (6))
        (rnrs io simple (6)))
(display (find even? '(3 1 4 1 5 9)))
```

2 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.sps" contains

```
(import (rnrs))
(display "hello\n")
```

then

```
plt-r6rs hi.sps
```

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 Racket perspective. Such files can be run like any other Racket module, such as using `racket`:

```
racket <program-file>
```

or using DrRacket. The file can also be compiled to bytecode using `raco make`:

```
raco make <program-file>
```

For example, if "hi.sps" contains

```
#!r6rs
(import (rnrs))
(display "hello\n")
```

then

```
racket hi.sps
```

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

3 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 Racket.

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.sls" 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 §5 “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 §5 “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.

4 R⁶RS Module Language

```
#lang r6rs
```

The `r6rs` language is usually used in the form `#!r6rs`, which is equivalent to `#lang r6rs` and is also valid R⁶RS syntax.

4.1 Using R⁶RS

See §1 “Using R⁶RS with DrRacket”, §2 “Running Top-Level Programs”, and §3 “Installing Libraries” for more information on writing and running R⁶RS programs with Racket.

4.2 The Implementation of R⁶RS

The R⁶RS language is itself implemented as a module within Racket. The details of that implementation, as provided in this section, are not normally relevant to programmers using R⁶RS; see the links in §4.1 “Using R⁶RS”, instead. The details may be relevant to programmers who are developing new tools or deriving variants of R⁶RS within Racket.

As a Racket module, the `r6rs` module language provides only a `#!/module-begin` binding, which is used to process the entire body of a Racket module (see `module`). The `#!/module-begin` binding from `r6rs` allows the body of a module to use the syntax of either a R⁶RS library or a R⁶RS top-level program.

```
(#!/module-begin
  (library library-name
    (export export-spec ...)
    (import import-spec ...)
    library-body ...))
(#!/module-begin
  (import import-spec ...)
  program-body ...)
```

An `r6rs` module that contains a single `library` form defines an R⁶RS library, while a module body that starts with an `import` form defines an R⁶RS top-level program.

The `library`, `export`, and `import` identifiers are not exported by the `r6rs` library; they are recognized through equivalence to unbound identifiers.

5 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 Racket 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"`, `".sls"`, and `".rkt"`. 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 Racket installation):

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


6 Language Interoperability

Using the conversion rules in §5 “Libraries and Collections”, and R⁶RS library can refer to modules that are implemented in other dialects supported by Racket, and other Racket modules can refer to libraries that are implemented in R⁶RS.

Beware that a *pair* in R⁶RS corresponds to a *mutable pair* in `racket/base`. Otherwise, R⁶RS libraries and `racket/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 `racket/base` can be used directly as binary ports with R⁶RS libraries, and all R⁶RS ports can be used as ports in `racket/base` programs, but only textual ports created via R⁶RS libraries can be used by other R⁶RS operations that expect textual ports.

7 R⁶RS Conformance

Racket'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 Racket:

```
(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 `racket-report-environment` correspond to R⁵RS bindings instead of R⁶RS bindings. In particular, `=>`, `else`, `_`, and `...` are not bound.
- Bindings for `#%datum`, `#%app`, `#%top`, and `#%top-interaction` are imported into every library and program, and at every phase level for which the library or program has imports.

8 R⁶RS Libraries

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

`(require rnrs/base-6)`

Original specification: Base

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

`(require rnrs/unicode-6)`

Original specification: Unicode

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

`(require rnrs/bytevectors-6)`

Original specification: Bytevectors

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

`(require rnrs/lists-6)`

Original specification: List utilities

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

`(require rnrs/sorting-6)`

Original specification: Sorting

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

`(require rnrs/control-6)`

Original specification: Control Structures

8.7 (nrns records syntactic (6)): Records: Syntactic

(require nrns/records/syntactic-6)

Original specification: Records: Syntactic

8.8 (nrns records procedural (6)): Records: Procedural

(require nrns/records/procedural-6)

Original specification: Records: Procedural

8.9 (nrns records inspection (6)): Records: Inspection

(require nrns/records/inspection-6)

Original specification: Records: Inspection

8.10 (nrns exceptions (6)): Exceptions

(require nrns/exceptions-6)

Original specification: Exceptions

See also §7 “R⁶RS Conformance”.

8.11 (nrns conditions (6)): Conditions

(require nrns/conditions-6)

Original specification: Conditions

8.12 (nrns io ports (6)): I/O: Ports

(require nrns/io/ports-6)

Original specification: I/O: Ports

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

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

Original specification: I/O: Simple

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

`(require rnrs/files-6)`

Original specification: File System

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

`(require rnrs/programs-6)`

Original specification: Command-line Access and Exit Values

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

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

Original specification: Arithmetic: Fixnums

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

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

Original specification: Arithmetic: Flonums

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

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

Original specification: Arithmetic: Bitwise

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

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

Original specification: Syntax-Case

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

`(require rnrs/hashtables-6)`

Original specification: Hashtables

A hashtable is a dictionary in the sense of `racket/dict`, and hash table operations interact with threads in the same way for hash tables created with `make-hash` (e.g., `hashtable-ref` and `hashtable-set!` are thread-safe).

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

`(require rnrs/enums-6)`

Original specification: Enumerations

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

`(require rnrs/eval-6)`

Original specification: Eval

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

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

Original specification: Mutable Pairs

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

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

Original specification: Mutable Strings

8.25 (nrns r5rs (6)): R5RS Compatibility

(require nrns/r5rs-6)

Original specification: R5RS Compatibility

See also §7 “R⁶RS Conformance”.

Index

`#!/module-begin`, 7
`&assertion`, 13
`&condition`, 13
`&error`, 13
`&i/o`, 13
`&i/o-decoding`, 13
`&i/o-encoding`, 13
`&i/o-file-already-exists`, 13
`&i/o-file-does-not-exist`, 13
`&i/o-file-is-read-only`, 13
`&i/o-file-protection`, 13
`&i/o-filename`, 13
`&i/o-invalid-position`, 13
`&i/o-port`, 13
`&i/o-read`, 13
`&i/o-write`, 13
`&implementation-restriction`, 13
`&irritants`, 13
`&lexical`, 13
`&message`, 13
`&no-infinities`, 14
`&no-nans`, 14
`&non-continuable`, 13
`&serious`, 13
`&syntax`, 13
`&undefined`, 13
`&violation`, 13
`&warning`, 13
`&who`, 13
`(rnrs arithmetic bitwise (6))`:
 Arithmetic: Bitwise, 14
`(rnrs arithmetic fixnums (6))`:
 Arithmetic: Fixnums, 14
`(rnrs arithmetic flonums (6))`:
 Arithmetic: Flonums, 14
`(rnrs base (6))`: Base, 12
`(rnrs bytevectors (6))`: Bytevectors,
 12
`(rnrs conditions (6))`: Conditions, 13
`(rnrs control (6))`: Control Structures,
 12
`(rnrs enums (6))`: Enumerations, 15
`(rnrs eval (6))`: Eval, 15
`(rnrs exceptions (6))`: Exceptions, 13
`(rnrs files (6))`: File System, 14
`(rnrs hashtables (6))`: Hashtables, 15
`(rnrs io ports (6))`: I/O: Ports, 13
`(rnrs io simple (6))`: I/O: Simple, 14
`(rnrs lists (6))`: List utilities, 12
`(rnrs mutable-pairs (6))`: Mutable
 Pairs, 15
`(rnrs mutable-strings (6))`: Mutable
 Strings, 15
`(rnrs programs (6))`: Command-line
 Access and Exit Values, 14
`(rnrs r5rs (6))`: R5RS Compatibility,
 16
`(rnrs records inspection (6))`:
 Records: Inspection, 13
`(rnrs records procedural (6))`:
 Records: Procedural, 13
`(rnrs records syntactic (6))`:
 Records: Syntactic, 13
`(rnrs sorting (6))`: Sorting, 12
`(rnrs syntax-case (6))`: Syntax-Case,
 15
`(rnrs unicode (6))`: Unicode, 12
`*`, 12
`+`, 12
`++path`, 6
`-`, 12
`...`, 12
`...`, 15
`/`, 12
`<`, 12
`<=`, 12
`=`, 12
`=>`, 12
`=>`, 13
`>`, 12
`>=`, 12
`_`, 12
`_`, 15
`abs`, 12

acos, 12
and, 12
angle, 12
append, 12
apply, 12
asin, 12
assert, 12
assertion-violation, 12
assertion-violation?, 13
assoc, 12
assp, 12
assq, 12
assv, 12
atan, 12
begin, 12
binary-port?, 13
bitwise-and, 14
bitwise-arithmetic-shift, 14
bitwise-arithmetic-shift-left, 14
bitwise-arithmetic-shift-right, 14
bitwise-bit-count, 14
bitwise-bit-field, 14
bitwise-bit-set?, 14
bitwise-copy-bit, 14
bitwise-copy-bit-field, 14
bitwise-first-bit-set, 14
bitwise-if, 14
bitwise-ior, 14
bitwise-length, 14
bitwise-not, 14
bitwise-reverse-bit-field, 14
bitwise-rotate-bit-field, 14
bitwise-xor, 14
boolean=?, 12
boolean?, 12
bound-identifier=?, 15
buffer-mode, 13
buffer-mode?, 13
bytevector->sint-list, 12
bytevector->string, 13
bytevector->u8-list, 12
bytevector->uint-list, 12
bytevector-copy, 12
bytevector-copy!, 12
bytevector-fill!, 12
bytevector-ieee-double-native-ref,
12
bytevector-ieee-double-native-
set!, 12
bytevector-ieee-double-ref, 12
bytevector-ieee-single-native-ref,
12
bytevector-ieee-single-native-
set!, 12
bytevector-ieee-single-ref, 12
bytevector-length, 12
bytevector-s16-native-ref, 12
bytevector-s16-native-set!, 12
bytevector-s16-ref, 12
bytevector-s16-set!, 12
bytevector-s32-native-ref, 12
bytevector-s32-native-set!, 12
bytevector-s32-ref, 12
bytevector-s32-set!, 12
bytevector-s64-native-ref, 12
bytevector-s64-native-set!, 12
bytevector-s64-ref, 12
bytevector-s64-set!, 12
bytevector-s8-ref, 12
bytevector-s8-set!, 12
bytevector-sint-ref, 12
bytevector-sint-set!, 12
bytevector-u16-native-ref, 12
bytevector-u16-native-set!, 12
bytevector-u16-ref, 12
bytevector-u16-set!, 12
bytevector-u32-native-ref, 12
bytevector-u32-native-set!, 12
bytevector-u32-ref, 12
bytevector-u32-set!, 12
bytevector-u64-native-ref, 12
bytevector-u64-native-set!, 12
bytevector-u64-ref, 12
bytevector-u64-set!, 12

bytevector-u8-ref, 12
 bytevector-u8-set!, 12
 bytevector-uint-ref, 12
 bytevector-uint-set!, 12
 bytevector=?, 12
 bytevector?, 12
 caar, 12
 cadr, 12
 call-with-bytevector-output-port,
 13
 call-with-current-continuation, 12
 call-with-input-file, 14
 call-with-output-file, 14
 call-with-port, 13
 call-with-string-output-port, 13
 call-with-values, 12
 call/cc, 12
 car, 12
 case, 12
 case-lambda, 12
 cdddar, 12
 cddddr, 12
 cdr, 12
 ceiling, 12
 char->integer, 12
 char-alphabetic?, 12
 char-ci<=?, 12
 char-ci<?, 12
 char-ci=?, 12
 char-ci>=?, 12
 char-ci>?, 12
 char-downcase, 12
 char-foldcase, 12
 char-general-category, 12
 char-lower-case?, 12
 char-numeric?, 12
 char-title-case?, 12
 char-titlecase, 12
 char-upcase, 12
 char-upper-case?, 12
 char-whitespace?, 12
 char<=?, 12
 char<?, 12
 char=?, 12
 char>=?, 12
 char>?, 12
 char?, 12
 close-input-port, 14
 close-output-port, 14
 close-port, 13
 command-line, 14
 complex?, 12
 cond, 12
 condition, 13
 condition-accessor, 13
 condition-irritants, 13
 condition-message, 13
 condition-predicate, 13
 condition-who, 13
 condition?, 13
 cons, 12
 cons*, 12
 cos, 12
 current-error-port, 13
 current-input-port, 13
 current-output-port, 13
 datum->syntax, 15
 define, 12
 define-condition-type, 13
 define-enumeration, 15
 define-record-type, 13
 define-syntax, 12
 delay, 16
 delete-file, 14
 denominator, 12
 display, 14
 div, 12
 div-and-mod, 12
 div0, 12
 div0-and-mod0, 12
 do, 12
 dynamic-wind, 12
 else, 12
 else, 13

endianness, 12
enum-set->list, 15
enum-set-complement, 15
enum-set-constructor, 15
enum-set-difference, 15
enum-set-indexer, 15
enum-set-intersection, 15
enum-set-member?, 15
enum-set-projection, 15
enum-set-subset?, 15
enum-set-union, 15
enum-set-universe, 15
enum-set=?, 15
environment, 15
eof-object, 13
eof-object?, 13
eol-style, 13
eq?, 12
equal-hash, 15
equal?, 12
eqv?, 12
error, 12
error-handling-mode, 13
error?, 13
eval, 15
even?, 12
exact, 12
exact->inexact, 16
exact-integer-sqrt, 12
exact?, 12
exists, 12
exit, 14
exp, 12
expt, 12
fields, 13
file-exists?, 14
file-options, 13
filter, 12
find, 12
finite?, 12
fixnum->flonum, 14
fixnum-width, 14
fixnum?, 14
fl*, 14
fl+, 14
fl-, 14
fl/, 14
fl<=?, 14
fl<?, 14
fl=?, 14
fl>=?, 14
fl>?, 14
flabs, 14
flacos, 14
flasin, 14
flatan, 14
flceiling, 14
flcos, 14
fldenominator, 14
fldiv, 14
fldiv-and-mod, 14
fldiv0, 14
fldiv0-and-mod0, 14
fleven?, 14
flexp, 14
flexpt, 14
flfinite?, 14
flfloor, 14
flinfinite?, 14
flinteger?, 14
fllog, 14
flmax, 14
flmin, 14
flmod, 14
flmod0, 14
flnan?, 14
flnegative?, 14
flnumerator, 14
flodd?, 14
flonum?, 14
floor, 12
flpositive?, 14
flround, 14
flsin, 14

flsqrt, 14
 fltan, 14
 fltruncate, 14
 flush-output-port, 13
 flzero?, 14
 fold-left, 12
 fold-right, 12
 for-all, 12
 for-each, 12
 force, 16
 free-identifier=?, 15
 fx*, 14
 fx*/carry, 14
 fx+, 14
 fx+/carry, 14
 fx-, 14
 fx-/carry, 14
 fx<=?, 14
 fx<?, 14
 fx=?, 14
 fx>=?, 14
 fx>?, 14
 fxand, 14
 fxarithmetic-shift, 14
 fxarithmetic-shift-left, 14
 fxarithmetic-shift-right, 14
 fxbit-count, 14
 fxbit-field, 14
 fxbit-set?, 14
 fxcopy-bit, 14
 fxcopy-bit-field, 14
 fxdiv, 14
 fxdiv-and-mod, 14
 fxdiv0, 14
 fxdiv0-and-mod0, 14
 fxeven?, 14
 fxfirst-bit-set, 14
 fxif, 14
 fxior, 14
 fxlength, 14
 fxmax, 14
 fxmin, 14
 fxmod, 14
 fxmod0, 14
 fxnegative?, 14
 fxnot, 14
 fxodd?, 14
 fxpositive?, 14
 fxreverse-bit-field, 14
 fxrotate-bit-field, 14
 fxxor, 14
 fxzero?, 14
 gcd, 12
 generate-temporaries, 15
 get-bytevector-all, 13
 get-bytevector-n, 13
 get-bytevector-n!, 13
 get-bytevector-some, 13
 get-char, 13
 get-datum, 13
 get-line, 13
 get-string-all, 13
 get-string-n, 13
 get-string-n!, 13
 get-u8, 13
 greatest-fixnum, 14
 guard, 13
 hashtable-clear!, 15
 hashtable-contains?, 15
 hashtable-copy, 15
 hashtable-delete!, 15
 hashtable-entries, 15
 hashtable-equivalence-function, 15
 hashtable-hash-function, 15
 hashtable-keys, 15
 hashtable-mutable?, 15
 hashtable-ref, 15
 hashtable-set!, 15
 hashtable-size, 15
 hashtable-update!, 15
 hashtable?, 15
 i/o-decoding-error?, 13
 i/o-encoding-error-char, 13
 i/o-encoding-error?, 13

[i/o-error-filename](#), 13
[i/o-error-port](#), 13
[i/o-error-position](#), 13
[i/o-error?](#), 13
[i/o-file-already-exists-error?](#), 13
[i/o-file-does-not-exist-error?](#), 13
[i/o-file-is-read-only-error?](#), 13
[i/o-file-protection-error?](#), 13
[i/o-filename-error?](#), 13
[i/o-invalid-position-error?](#), 13
[i/o-port-error?](#), 13
[i/o-read-error?](#), 13
[i/o-write-error?](#), 13
[identifier-syntax](#), 12
[identifier?](#), 15
[if](#), 12
[imag-part](#), 12
[immutable](#), 13
[implementation-restriction-violation?](#), 13
[inexact](#), 12
[inexact->exact](#), 16
[inexact?](#), 12
[infinite?](#), 12
[input-port?](#), 13
[Installing Libraries](#), 6
[integer->char](#), 12
[integer-valued?](#), 12
[integer?](#), 12
[irritants-condition?](#), 13
[lambda](#), 12
[Language Interoperability](#), 9
[latin-1-codec](#), 13
[lcm](#), 12
[least-fixnum](#), 14
[length](#), 12
[let](#), 12
[let*](#), 12
[let*-values](#), 12
[let-syntax](#), 12
[let-values](#), 12
[letrec](#), 12
[letrec*](#), 12
[letrec-syntax](#), 12
[lexical-violation?](#), 13
[Libraries and Collections](#), 8
[list](#), 12
[list->string](#), 12
[list->vector](#), 12
[list-ref](#), 12
[list-sort](#), 12
[list-tail](#), 12
[list?](#), 12
[log](#), 12
[lookahead-char](#), 13
[lookahead-u8](#), 13
[magnitude](#), 12
[make-assertion-violation](#), 13
[make-bytevector](#), 12
[make-custom-binary-input-port](#), 13
[make-custom-binary-input/output-port](#), 13
[make-custom-binary-output-port](#), 13
[make-custom-textual-input-port](#), 13
[make-custom-textual-input/output-port](#), 13
[make-custom-textual-output-port](#), 13
[make-enumeration](#), 15
[make-eq-hashtable](#), 15
[make-eqv-hashtable](#), 15
[make-error](#), 13
[make-hashtable](#), 15
[make-i/o-decoding-error](#), 13
[make-i/o-encoding-error](#), 13
[make-i/o-error](#), 13
[make-i/o-file-already-exists-error](#), 13
[make-i/o-file-does-not-exist-error](#), 13
[make-i/o-file-is-read-only-error](#), 13
[make-i/o-file-protection-error](#), 13
[make-i/o-filename-error](#), 13
[make-i/o-invalid-position-error](#), 13

make-i/o-port-error, 13
 make-i/o-read-error, 13
 make-i/o-write-error, 13
 make-implementation-restriction-violation, 13
 make-irritants-condition, 13
 make-lexical-violation, 13
 make-message-condition, 13
 make-no-infinities-violation, 14
 make-no-nans-violation, 14
 make-non-continuable-violation, 13
 make-polar, 12
 make-record-constructor-descriptor, 13
 make-record-type-descriptor, 13
 make-rectangular, 12
 make-serious-condition, 13
 make-string, 12
 make-syntax-violation, 13
 make-transcoder, 13
 make-undefined-violation, 13
 make-variable-transformer, 15
 make-vector, 12
 make-violation, 13
 make-warning, 13
 make-who-condition, 13
 map, 12
 max, 12
 member, 12
 mexp, 12
 memq, 12
 memv, 12
 message-condition?, 13
 min, 12
 mod, 12
 mod0, 12
 modulo, 16
 mutable, 13
 nan?, 12
 native-endianness, 12
 native-eol-style, 13
 native-transcoder, 13
 negative?, 12
 newline, 14
 no-infinities-violation?, 14
 no-nans-violation?, 14
 non-continuable-violation?, 13
 nongenerative, 13
 not, 12
 null-environment, 16
 null?, 12
 number->string, 12
 number?, 12
 numerator, 12
 odd?, 12
 opaque, 13
 open-bytevector-input-port, 13
 open-bytevector-output-port, 13
 open-file-input-port, 13
 open-file-input/output-port, 13
 open-file-output-port, 13
 open-input-file, 14
 open-output-file, 14
 open-string-input-port, 13
 open-string-output-port, 13
 or, 12
 output-port-buffer-mode, 13
 output-port?, 13
 pair?, 12
 parent, 13
 parent-rtd, 13
 partition, 12
 peek-char, 14
 port-eof?, 13
 port-has-port-position?, 13
 port-has-set-port-position!?, 13
 port-position, 13
 port-transcoder, 13
 port?, 13
 positive?, 12
 procedure?, 12
 protocol, 13
 put-bytevector, 13
 put-char, 13

- [put-datum](#), 13
- [put-string](#), 13
- [put-u8](#), 13
- [quasiquote](#), 12
- [quasisyntax](#), 15
- [quote](#), 12
- [quotient](#), 16
- [r6rs](#), 7
- [R⁶RS Conformance](#), 10
- [R⁶RS Libraries](#), 12
- [R⁶RS Module Language](#), 7
- [R6RS: Scheme](#), 1
- [raise](#), 13
- [raise-continuable](#), 13
- [rational-valued?](#), 12
- [rational?](#), 12
- [rationalize](#), 12
- [read](#), 14
- [read-char](#), 14
- [real->flonum](#), 14
- [real-part](#), 12
- [real-valued?](#), 12
- [real?](#), 12
- [record-accessor](#), 13
- [record-constructor](#), 13
- [record-constructor-descriptor](#), 13
- [record-field-mutable?](#), 13
- [record-mutator](#), 13
- [record-predicate](#), 13
- [record-rtd](#), 13
- [record-type-descriptor](#), 13
- [record-type-descriptor?](#), 13
- [record-type-field-names](#), 13
- [record-type-generative?](#), 13
- [record-type-name](#), 13
- [record-type-opaque?](#), 13
- [record-type-parent](#), 13
- [record-type-sealed?](#), 13
- [record-type-uid](#), 13
- [record?](#), 13
- [remainder](#), 16
- [remove](#), 12
- [remp](#), 12
- [remq](#), 12
- [remv](#), 12
- [reverse](#), 12
- [rnrs/arithmetic/bitwise-6](#), 14
- [rnrs/arithmetic/fixnums-6](#), 14
- [rnrs/arithmetic/flonums-6](#), 14
- [rnrs/base-6](#), 12
- [rnrs/bytevectors-6](#), 12
- [rnrs/conditions-6](#), 13
- [rnrs/control-6](#), 12
- [rnrs/enums-6](#), 15
- [rnrs/eval-6](#), 15
- [rnrs/exceptions-6](#), 13
- [rnrs/files-6](#), 14
- [rnrs/hashtables-6](#), 15
- [rnrs/io/ports-6](#), 13
- [rnrs/io/simple-6](#), 14
- [rnrs/lists-6](#), 12
- [rnrs/mutable-pairs-6](#), 15
- [rnrs/mutable-strings-6](#), 15
- [rnrs/programs-6](#), 14
- [rnrs/r5rs-6](#), 16
- [rnrs/records/inspection-6](#), 13
- [rnrs/records/procedural-6](#), 13
- [rnrs/records/syntactic-6](#), 13
- [rnrs/sorting-6](#), 12
- [rnrs/syntax-case-6](#), 15
- [rnrs/unicode-6](#), 12
- [round](#), 12
- [Running Top-Level Programs](#), 5
- [scheme-report-environment](#), 16
- [sealed](#), 13
- [serious-condition?](#), 13
- [set!](#), 12
- [set-car!](#), 15
- [set-cdr!](#), 15
- [set-port-position!](#), 13
- [simple-conditions](#), 13
- [sin](#), 12
- [sint-list->bytevector](#), 12
- [sqrt](#), 12

standard-error-port, 13
 standard-input-port, 13
 standard-output-port, 13
 string, 12
 string->bytevector, 13
 string->list, 12
 string->number, 12
 string->symbol, 12
 string->utf16, 12
 string->utf32, 12
 string->utf8, 12
 string-append, 12
 string-ci-hash, 15
 string-ci<=?, 12
 string-ci<?, 12
 string-ci=?, 12
 string-ci>=?, 12
 string-ci>?, 12
 string-copy, 12
 string-downcase, 12
 string-fill!, 15
 string-foldcase, 12
 string-for-each, 12
 string-hash, 15
 string-length, 12
 string-normalize-nfc, 12
 string-normalize-nfd, 12
 string-normalize-nfkc, 12
 string-normalize-nfkd, 12
 string-ref, 12
 string-set!, 15
 string-titlecase, 12
 string-upcase, 12
 string<=?, 12
 string<?, 12
 string=?, 12
 string>=?, 12
 string>?, 12
 string?, 12
 substring, 12
 symbol->string, 12
 symbol-hash, 15
 symbol=?, 12
 symbol?, 12
 syntax, 15
 syntax->datum, 15
 syntax-case, 15
 syntax-rules, 12
 syntax-violation, 15
 syntax-violation-form, 13
 syntax-violation-subform, 13
 syntax-violation?, 13
 tan, 12
 textual-port?, 13
 The Implementation of R⁶RS, 7
 transcoded-port, 13
 transcoder-codec, 13
 transcoder-eol-style, 13
 transcoder-error-handling-mode, 13
 truncate, 12
 u8-list->bytevector, 12
 uint-list->bytevector, 12
 undefined-violation?, 13
 unless, 12
 unquote, 12
 unquote-splicing, 12
 unsyntax, 15
 unsyntax-splicing, 15
 Using R⁶RS, 7
 Using R⁶RS with DrRacket, 4
 utf-16-codec, 13
 utf-8-codec, 13
 utf16->string, 12
 utf32->string, 12
 utf8->string, 12
 values, 12
 vector, 12
 vector->list, 12
 vector-fill!, 12
 vector-for-each, 12
 vector-length, 12
 vector-map, 12
 vector-ref, 12
 vector-set!, 12

[vector-sort](#), 12
[vector-sort!](#), 12
[vector?](#), 12
[violation?](#), 13
[warning?](#), 13
when, 12
[who-condition?](#), 13
[with-exception-handler](#), 13
[with-input-from-file](#), 14
[with-output-to-file](#), 14
with-syntax, 15
write, 14
[write-char](#), 14
[zero?](#), 12