R6RS: Scheme

Version 9.4.0.3

November 2, 2026

The The Revised^6 Report on the Algorithmic Language Scheme defines a dialect of Scheme. We use *R^{6}RS* to refer to both the standard and the language defined by the standard.

R^{6}RS defines both *libraries* and *top-level programs*. Both correspond to Racket *modules* (see §6 “Modules”). That is, although R^{6}RS 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 R^{6}RS top-level program cannot export any bindings to other modules.

See §23 “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 **8**

4.1 Using R6RS 8

4.2 The Implementation of R6RS 8

5 Libraries and Collections **9**

6 Language Interoperability **10**

7 R6RS Conformance **11**

8 R6RS Libraries **13**

8.1 `(rnrs base (6))`: Base 13

8.2 `(rnrs unicode (6))`: Unicode 13

8.3 `(rnrs bytevectors (6))`: Bytevectors 13

8.4 `(rnrs lists (6))`: List utilities 13

8.5 `(rnrs sorting (6))`: Sorting 13

8.6 `(rnrs control (6))`: Control Structures 13

8.7 `(rnrs records syntactic (6))`: Records: Syntactic 14

8.8 `(rnrs records procedural (6))`: Records: Procedural 14

8.9 `(rnrs records inspection (6))`: Records: Inspection 14

8.10 `(rnrs exceptions (6))`: Exceptions 14

8.11 (rnrs conditions (6)): Conditions 14
8.12 (rnrs io ports (6)): I/O: Ports 14
8.13 (rnrs io simple (6)): I/O: Simple 15
8.14 (rnrs files (6)): File System 15
8.15 (rnrs programs (6)): Command-line Access and Exit Values 15
8.16 (rnrs arithmetic fixnums (6)): Arithmetic: Fixnums 15
8.17 (rnrs arithmetic flonums (6)): Arithmetic: Flonums 15
8.18 (rnrs arithmetic bitwise (6)): Arithmetic: Bitwise 15
8.19 (rnrs syntax-case (6)): Syntax-Case 16
8.20 (rnrs hashtables (6)): Hashtables 16
8.21 (rnrs enums (6)): Enumerations 16
8.22 (rnrs eval (6)): Eval 16
8.23 (rnrs mutable-pairs (6)): Mutable Pairs 16
8.24 (rnrs mutable-strings (6)): Mutable Strings 16
8.25 (rnrs r5rs (6)): R5RS Compatibility 17

Index **18**

Index **18**

1 Using R^{6}RS with DrRacket

To run an R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS 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⟩
```

You may as well specify an arbitrary collections directory by using the `--collections` flag:

```
plt-r6rs --install --collections ⟨directory⟩ ⟨libraries-file⟩
```

See §5 “Libraries and Collections” for information on how R^{6}RS 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 R^{6}RS library prefixed with `#!r6rs`, then `plt-r6rs ++path` ⟨*dir*⟩ directs the R^{6}RS 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 R^{6}RS Module Language

`#lang r6rs` package: `r6rs-lib`

The `r6rs` language is usually used in the form `#!r6rs`, which is equivalent to `#lang r6rs` and is also valid R^{6}RS syntax.

4.1 Using R^{6}RS

See §1 "Using R^{6}RS with DrRacket", §2 "Running Top-Level Programs", and §3 "Installing Libraries" for more information on writing and running R^{6}RS programs with Racket.

4.2 The Implementation of R^{6}RS

The R^{6}RS 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 R^{6}RS; see the links in §4.1 "Using R^{6}RS", instead. The details may be relevant to programmers who are developing new tools or deriving variants of R^{6}RS 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 R^{6}RS library or a R^{6}RS 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 R^{6}RS library, while a module body that starts with an `import` form defined an R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS 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 R^{6}RS.

Beware that a *pair* in R^{6}RS corresponds to a *mutable pair* in `racket/base`. Otherwise, R^{6}RS 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 R^{6}RS libraries, and all R^{6}RS ports can be used as ports in `racket/base` programs, but only textual ports created via R^{6}RS libraries can be used by other R^{6}RS operations that expect textual ports.

7 R^{6}RS Conformance

Racket's R^{6}RS support does not conform with the standard in several known ways:

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

  This difference can be made visible using `dynamic-wind`. According to R^{6}RS, 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.

- 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 R^{5}RS bindings instead of R^{6}RS 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.

Changed in version 6.0.1.4: When an identifier bound by `letrec` or `letrec*` is referenced before it is initialized, an exception is raised, instead of producing `#<undefined>`.

8 R⁶RS Libraries

8.1 (rnrs base (6)): Base

(require rnrs/base-6) package: r6rs-lib

Original specification: Base

8.2 (rnrs unicode (6)): Unicode

(require rnrs/unicode-6) package: r6rs-lib

Original specification: Unicode

8.3 (rnrs bytevectors (6)): Bytevectors

(require rnrs/bytevectors-6) package: r6rs-lib

Original specification: Bytevectors

8.4 (rnrs lists (6)): List utilities

(require rnrs/lists-6) package: r6rs-lib

Original specification: List utilities

8.5 (rnrs sorting (6)): Sorting

(require rnrs/sorting-6) package: r6rs-lib

Original specification: Sorting

8.6 (rnrs control (6)): Control Structures

(require rnrs/control-6) package: r6rs-lib

Original specification: Control Structures

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

`(require rnrs/records/syntactic-6)` package: `r6rs-lib`

Original specification: Records: Syntactic

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

`(require rnrs/records/procedural-6)` package: `r6rs-lib`

Original specification: Records: Procedural

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

`(require rnrs/records/inspection-6)` package: `r6rs-lib`

Original specification: Records: Inspection

8.10 (rnrs exceptions (6)): Exceptions

`(require rnrs/exceptions-6)` package: `r6rs-lib`

Original specification: Exceptions

See also §7 "R^{6}RS Conformance".

8.11 (rnrs conditions (6)): Conditions

`(require rnrs/conditions-6)` package: `r6rs-lib`

Original specification: Conditions

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

`(require rnrs/io/ports-6)` package: `r6rs-lib`

Original specification: I/O: Ports

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

`(require rnrs/io/simple-6)` package: `r6rs-lib`

Original specification: I/O: Simple

8.14 (rnrs files (6)): File System

`(require rnrs/files-6)` package: `r6rs-lib`

Original specification: File System

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

`(require rnrs/programs-6)` package: `r6rs-lib`

Original specification: Command-line Access and Exit Values

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

`(require rnrs/arithmetic/fixnums-6)` package: `r6rs-lib`

Original specification: Arithmetic: Fixnums

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

`(require rnrs/arithmetic/flonums-6)` package: `r6rs-lib`

Original specification: Arithmetic: Flonums

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

`(require rnrs/arithmetic/bitwise-6)` package: `r6rs-lib`

Original specification: Arithmetic: Bitwise

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

(require rnrs/syntax-case-6) package: r6rs-lib

Original specification: Syntax-Case

8.20 (rnrs hashtables (6)): Hashtables

(require rnrs/hashtables-6) package: r6rs-lib

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) package: r6rs-lib

Original specification: Enumerations

8.22 (rnrs eval (6)): Eval

(require rnrs/eval-6) package: r6rs-lib

Original specification: Eval

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

(require rnrs/mutable-pairs-6) package: r6rs-lib

Original specification: Mutable Pairs

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

(require rnrs/mutable-strings-6) package: r6rs-lib

Original specification: Mutable Strings

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

`(require rnrs/r5rs-6)` package: `r6rs-lib`

Original specification: R5RS Compatibility

See also §7 "R^{6}RS Conformance".

Index

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

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

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

enum-set-member?
enum-set-projection
enum-set-subset?
enum-set-union
enum-set-universe
enum-set=?
environment
eof-object
eof-object?
eol-style
eq?
equal-hash
equal?
eqv?
error
error-handling-mode
error?
eval
even?
exact
exact->inexact
exact-integer-sqrt
exact?
exists
exit
exp
expt
fields
file-exists?
file-options
filter
find
finite?
fixnum->flonum
fixnum-width
fixnum?
fl*
fl+
fl-
fl/
fl<=?
fl<?
fl=?
fl>=?
fl>?
flabs
flacos
flasin
flatan
flceiling
flcos
fldenominator
fldiv
fldiv-and-mod
fldiv0
fldiv0-and-mod0
fleven?
flexp
flexpt
flfinite?
flfloor
flinfinite?
flinteger?
fllog
flmax
flmin
flmod
flmod0
flnan?
flnegative?
flnumerator
flodd?
flonum?
floor
flpositive?
flround
flsin
flsqrt
fltan
fltruncate
flush-output-port
flzero?
fold-left
fold-right

for-all
for-each
force
free-identifier=?
fx*
fx*/carry
fx+
fx+/carry
fx-
fx-/carry
fx<=?
fx<?
fx=?
fx>=?
fx>?
fxand
fxarithmetic-shift
fxarithmetic-shift-left
fxarithmetic-shift-right
fxbit-count
fxbit-field
fxbit-set?
fxcopy-bit
fxcopy-bit-field
fxdiv
fxdiv-and-mod
fxdiv0
fxdiv0-and-mod0
fxeven?
fxfirst-bit-set
fxif
fxior
fxlength
fxmax
fxmin
fxmod
fxmod0
fxnegative?
fxnot
fxodd?
fxpositive?
fxreverse-bit-field
fxrotate-bit-field
fxxor
fxzero?
gcd
generate-temporaries
get-bytevector-all
get-bytevector-n
get-bytevector-n!
get-bytevector-some
get-char
get-datum
get-line
get-string-all
get-string-n
get-string-n!
get-u8
greatest-fixnum
guard
hashtable-clear!
hashtable-contains?
hashtable-copy
hashtable-delete!
hashtable-entries
hashtable-equivalence-function
hashtable-hash-function
hashtable-keys
hashtable-mutable?
hashtable-ref
hashtable-set!
hashtable-size
hashtable-update!
hashtable?
i/o-decoding-error?
i/o-encoding-error-char
i/o-encoding-error?
i/o-error-filename
i/o-error-port
i/o-error-position
i/o-error?
i/o-file-already-exists-error?
i/o-file-does-not-exist-error?
i/o-file-is-read-only-error?

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

make-no-infinities-violation
make-no-nans-violation
make-non-continuable-violation
make-polar
make-record-constructor-descriptor
make-record-type-descriptor
make-rectangular
make-serious-condition
make-string
make-syntax-violation
make-transcoder
make-undefined-violation
make-variable-transformer
make-vector
make-violation
make-warning
make-who-condition
map
max
member
memp
memq
memv
message-condition?
min
mod
mod0
modulo
mutable
nan?
native-endianness
native-eol-style
native-transcoder
negative?
newline
no-infinities-violation?
no-nans-violation?
non-continuable-violation?
nongenerative
not
null-environment
null?
number->string
number?
numerator
odd?
opaque
open-bytevector-input-port
open-bytevector-output-port
open-file-input-port
open-file-input/output-port
open-file-output-port
open-input-file
open-output-file
open-string-input-port
open-string-output-port
or
output-port-buffer-mode
output-port?
pair?
parent
parent-rtd
partition
peek-char
port-eof?
port-has-port-position?
port-has-set-port-position!?
port-position
port-transcoder
port?
positive?
procedure?
protocol
put-bytevector
put-char
put-datum
put-string
put-u8
quasiquote
quasisyntax
quote
quotient
r6rs

R6RS Conformance
R6RS Libraries
R6RS Module Language
R6RS: Scheme
raise
raise-continuable
rational-valued?
rational?
rationalize
read
read-char
real->flonum
real-part
real-valued?
real?
record-accessor
record-constructor
record-constructor-descriptor
record-field-mutable?
record-mutator
record-predicate
record-rtd
record-type-descriptor
record-type-descriptor?
record-type-field-names
record-type-generative?
record-type-name
record-type-opaque?
record-type-parent
record-type-sealed?
record-type-uid
record?
remainder
remove
remp
remq
remv
reverse
rnrs/arithmetic/bitwise-6
rnrs/arithmetic/fixnums-6
rnrs/arithmetic/flonums-6
rnrs/base-6
rnrs/bytevectors-6
rnrs/conditions-6
rnrs/control-6
rnrs/enums-6
rnrs/eval-6
rnrs/exceptions-6
rnrs/files-6
rnrs/hashtables-6
rnrs/io/ports-6
rnrs/io/simple-6
rnrs/lists-6
rnrs/mutable-pairs-6
rnrs/mutable-strings-6
rnrs/programs-6
rnrs/r5rs-6
rnrs/records/inspection-6
rnrs/records/procedural-6
rnrs/records/syntactic-6
rnrs/sorting-6
rnrs/syntax-case-6
rnrs/unicode-6
round
Running Top-Level Programs
scheme-report-environment
sealed
serious-condition?
set!
set-car!
set-cdr!
set-port-position!
simple-conditions
sin
sint-list->bytevector
sqrt
standard-error-port
standard-input-port
standard-output-port
string
string->bytevector
string->list
string->number
string->symbol

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

```
with-input-from-file
with-output-to-file
with-syntax
write
write-char
zero?
```