

R6RS: Scheme

Version 5.3

August 6, 2012

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 §21 “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 R ⁶ RS	8
4.2	The Implementation of R ⁶ RS	8
5	Libraries and Collections	9
6	Language Interoperability	10
7	R⁶RS Conformance	11
8	R⁶RS Libraries	13
8.1	(<code>rnrs base (6)</code>): Base	13
8.2	(<code>rnrs unicode (6)</code>): Unicode	13
8.3	(<code>rnrs bytevectors (6)</code>): Bytevectors	13
8.4	(<code>rnrs lists (6)</code>): List utilities	13
8.5	(<code>rnrs sorting (6)</code>): Sorting	13
8.6	(<code>rnrs control (6)</code>): Control Structures	13
8.7	(<code>rnrs records syntactic (6)</code>): Records: Syntactic	14
8.8	(<code>rnrs records procedural (6)</code>): Records: Procedural	14
8.9	(<code>rnrs records inspection (6)</code>): Records: Inspection	14
8.10	(<code>rnrs exceptions (6)</code>): Exceptions	14

8.11	<code>(rnrs conditions (6))</code> : Conditions	14
8.12	<code>(rnrs io ports (6))</code> : I/O: Ports	14
8.13	<code>(rnrs io simple (6))</code> : I/O: Simple	15
8.14	<code>(rnrs files (6))</code> : File System	15
8.15	<code>(rnrs programs (6))</code> : Command-line Access and Exit Values	15
8.16	<code>(rnrs arithmetic fixnums (6))</code> : Arithmetic: Fixnums	15
8.17	<code>(rnrs arithmetic flonums (6))</code> : Arithmetic: Flonums	15
8.18	<code>(rnrs arithmetic bitwise (6))</code> : Arithmetic: Bitwise	15
8.19	<code>(rnrs syntax-case (6))</code> : Syntax-Case	16
8.20	<code>(rnrs hashtables (6))</code> : Hashtables	16
8.21	<code>(rnrs enums (6))</code> : Enumerations	16
8.22	<code>(rnrs eval (6))</code> : Eval	16
8.23	<code>(rnrs mutable-pairs (6))</code> : Mutable Pairs	16
8.24	<code>(rnrs mutable-strings (6))</code> : Mutable Strings	16
8.25	<code>(rnrs r5rs (6))</code> : R5RS Compatibility	17
	Index	18

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>
```

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 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):

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

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-[raiseed](#) 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 `(rnrs r5rs (6))`: R5RS Compatibility

`(require rnrs/r5rs-6)`

Original specification: R5RS Compatibility

See also §7 “R⁶RS Conformance”.

Index

`#!/module-begin`, 8
`&assertion`, 14
`&condition`, 14
`&error`, 14
`&i/o`, 14
`&i/o-decoding`, 14
`&i/o-encoding`, 14
`&i/o-file-already-exists`, 14
`&i/o-file-does-not-exist`, 14
`&i/o-file-is-read-only`, 14
`&i/o-file-protection`, 14
`&i/o-filename`, 14
`&i/o-invalid-position`, 14
`&i/o-port`, 14
`&i/o-read`, 14
`&i/o-write`, 14
`&implementation-restriction`, 14
`&irritants`, 14
`&lexical`, 14
`&message`, 14
`&no-infinities`, 15
`&no-nans`, 15
`&non-continuable`, 14
`&serious`, 14
`&syntax`, 14
`&undefined`, 14
`&violation`, 14
`&warning`, 14
`&who`, 14
`(rnrs arithmetic bitwise (6))`: Arithmetic: Bitwise, 15
`(rnrs arithmetic fixnums (6))`: Arithmetic: Fixnums, 15
`(rnrs arithmetic flonums (6))`: Arithmetic: Flonums, 15
`(rnrs base (6))`: Base, 13
`(rnrs bytevectors (6))`: Bytevectors, 13
`(rnrs conditions (6))`: Conditions, 14
`(rnrs control (6))`: Control Structures, 13
`(rnrs enums (6))`: Enumerations, 16
`(rnrs eval (6))`: Eval, 16
`(rnrs exceptions (6))`: Exceptions, 14
`(rnrs files (6))`: File System, 15
`(rnrs hashtables (6))`: Hashtables, 16
`(rnrs io ports (6))`: I/O: Ports, 14
`(rnrs io simple (6))`: I/O: Simple, 15
`(rnrs lists (6))`: List utilities, 13
`(rnrs mutable-pairs (6))`: Mutable Pairs, 16
`(rnrs mutable-strings (6))`: Mutable Strings, 16
`(rnrs programs (6))`: Command-line Access and Exit Values, 15
`(rnrs r5rs (6))`: R5RS Compatibility, 17
`(rnrs records inspection (6))`: Records: Inspection, 14
`(rnrs records procedural (6))`: Records: Procedural, 14
`(rnrs records syntactic (6))`: Records: Syntactic, 14
`(rnrs sorting (6))`: Sorting, 13
`(rnrs syntax-case (6))`: Syntax-Case, 16
`(rnrs unicode (6))`: Unicode, 13
`*`, 13
`+`, 13
`++path`, 6
`-`, 13
`...`, 13
`...`, 16
`/`, 13
`<`, 13
`<=`, 13
`=`, 13
`=>`, 13
`=>`, 14
`>`, 13
`>=`, 13
`_`, 13
`_`, 16
`abs`, 13

[acos](#), 13
[and](#), 13
[angle](#), 13
[append](#), 13
[apply](#), 13
[asin](#), 13
[assert](#), 13
[assertion-violation](#), 13
[assertion-violation?](#), 14
[assoc](#), 13
[assp](#), 13
[assq](#), 13
[assv](#), 13
[atan](#), 13
[begin](#), 13
[binary-port?](#), 14
[bitwise-and](#), 15
[bitwise-arithmetic-shift](#), 15
[bitwise-arithmetic-shift-left](#), 15
[bitwise-arithmetic-shift-right](#), 15
[bitwise-bit-count](#), 15
[bitwise-bit-field](#), 15
[bitwise-bit-set?](#), 15
[bitwise-copy-bit](#), 15
[bitwise-copy-bit-field](#), 15
[bitwise-first-bit-set](#), 15
[bitwise-if](#), 15
[bitwise-ior](#), 15
[bitwise-length](#), 15
[bitwise-not](#), 15
[bitwise-reverse-bit-field](#), 15
[bitwise-rotate-bit-field](#), 15
[bitwise-xor](#), 15
[boolean=?](#), 13
[boolean?](#), 13
[bound-identifier=?](#), 16
[buffer-mode](#), 14
[buffer-mode?](#), 14
[bytevector->sint-list](#), 13
[bytevector->string](#), 14
[bytevector->u8-list](#), 13
[bytevector->uint-list](#), 13
[bytevector-copy](#), 13
[bytevector-copy!](#), 13
[bytevector-fill!](#), 13
[bytevector-ieee-double-native-ref](#), 13
[bytevector-ieee-double-native-set!](#), 13
[bytevector-ieee-double-ref](#), 13
[bytevector-ieee-single-native-ref](#), 13
[bytevector-ieee-single-native-set!](#), 13
[bytevector-ieee-single-ref](#), 13
[bytevector-length](#), 13
[bytevector-s16-native-ref](#), 13
[bytevector-s16-native-set!](#), 13
[bytevector-s16-ref](#), 13
[bytevector-s16-set!](#), 13
[bytevector-s32-native-ref](#), 13
[bytevector-s32-native-set!](#), 13
[bytevector-s32-ref](#), 13
[bytevector-s32-set!](#), 13
[bytevector-s64-native-ref](#), 13
[bytevector-s64-native-set!](#), 13
[bytevector-s64-ref](#), 13
[bytevector-s64-set!](#), 13
[bytevector-s8-ref](#), 13
[bytevector-s8-set!](#), 13
[bytevector-sint-ref](#), 13
[bytevector-sint-set!](#), 13
[bytevector-u16-native-ref](#), 13
[bytevector-u16-native-set!](#), 13
[bytevector-u16-ref](#), 13
[bytevector-u16-set!](#), 13
[bytevector-u32-native-ref](#), 13
[bytevector-u32-native-set!](#), 13
[bytevector-u32-ref](#), 13
[bytevector-u32-set!](#), 13
[bytevector-u64-native-ref](#), 13
[bytevector-u64-native-set!](#), 13
[bytevector-u64-ref](#), 13
[bytevector-u64-set!](#), 13

bytevector-u8-ref, 13	char<?, 13
bytevector-u8-set!, 13	char=?, 13
bytevector-uint-ref, 13	char>=?, 13
bytevector-uint-set!, 13	char>?, 13
bytevector=?, 13	char?, 13
bytevector?, 13	close-input-port, 15
caar, 13	close-output-port, 15
cadr, 13	close-port, 14
call-with-bytevector-output-port,	command-line, 15
14	complex?, 13
call-with-current-continuation, 13	cond, 13
call-with-input-file, 15	condition, 14
call-with-output-file, 15	condition-accessor, 14
call-with-port, 14	condition-irritants, 14
call-with-string-output-port, 14	condition-message, 14
call-with-values, 13	condition-predicate, 14
call/cc, 13	condition-who, 14
car, 13	condition?, 14
case, 13	cons, 13
case-lambda, 13	cons*, 13
cdddar, 13	cos, 13
cddddr, 13	current-error-port, 14
cdr, 13	current-input-port, 14
ceiling, 13	current-output-port, 14
char->integer, 13	datum->syntax, 16
char-alphabetic?, 13	define, 13
char-ci<=?, 13	define-condition-type, 14
char-ci<?, 13	define-enumeration, 16
char-ci=?, 13	define-record-type, 14
char-ci>=?, 13	define-syntax, 13
char-ci>?, 13	delay, 17
char-downcase, 13	delete-file, 15
char-foldcase, 13	denominator, 13
char-general-category, 13	display, 15
char-lower-case?, 13	div, 13
char-numeric?, 13	div-and-mod, 13
char-title-case?, 13	div0, 13
char-titlecase, 13	div0-and-mod0, 13
char-upcase, 13	do, 13
char-upper-case?, 13	dynamic-wind, 13
char-whitespace?, 13	else, 13
char<=?, 13	else, 14

- endianness, 13
- enum-set->list, 16
- enum-set-complement, 16
- enum-set-constructor, 16
- enum-set-difference, 16
- enum-set-indexer, 16
- enum-set-intersection, 16
- enum-set-member?, 16
- enum-set-projection, 16
- enum-set-subset?, 16
- enum-set-union, 16
- enum-set-universe, 16
- enum-set=?, 16
- environment, 16
- eof-object, 14
- eof-object?, 14
- eol-style, 14
- eq?, 13
- equal-hash, 16
- equal?, 13
- eqv?, 13
- error, 13
- error-handling-mode, 14
- error?, 14
- eval, 16
- even?, 13
- exact, 13
- exact->inexact, 17
- exact-integer-sqrt, 13
- exact?, 13
- exists, 13
- exit, 15
- exp, 13
- expt, 13
- fields, 14
- file-exists?, 15
- file-options, 14
- filter, 13
- find, 13
- finite?, 13
- fixnum->flonum, 15
- fixnum-width, 15
- fixnum?, 15
- fl*, 15
- fl+, 15
- fl-, 15
- fl/, 15
- fl<=?, 15
- fl<?, 15
- fl=?, 15
- fl>=?, 15
- fl>?, 15
- flabs, 15
- flacos, 15
- flasin, 15
- flatan, 15
- flceiling, 15
- flcos, 15
- fldenominator, 15
- fldiv, 15
- fldiv-and-mod, 15
- fldiv0, 15
- fldiv0-and-mod0, 15
- fileven?, 15
- flexp, 15
- flexpt, 15
- flfinite?, 15
- flfloor, 15
- flinfinite?, 15
- flinteger?, 15
- fllog, 15
- flmax, 15
- flmin, 15
- flmod, 15
- flmod0, 15
- flnan?, 15
- flnegative?, 15
- flnumerator, 15
- flodd?, 15
- flonum?, 15
- floor, 13
- flpositive?, 15
- flround, 15
- flsin, 15

flsqrt, 15	fxmod, 15
fltan, 15	fxmod0, 15
fltruncate, 15	fxnegative?, 15
flush-output-port, 14	fxnot, 15
flzero?, 15	fxodd?, 15
fold-left, 13	fxpositive?, 15
fold-right, 13	fxreverse-bit-field, 15
for-all, 13	fxrotate-bit-field, 15
for-each, 13	fxxor, 15
force, 17	fxzero?, 15
free-identifier=?, 16	gcd, 13
fx*, 15	generate-temporaries, 16
fx*/carry, 15	get-bytevector-all, 14
fx+, 15	get-bytevector-n, 14
fx+/carry, 15	get-bytevector-n!, 14
fx-, 15	get-bytevector-some, 14
fx-/carry, 15	get-char, 14
fx<=?, 15	get-datum, 14
fx<?, 15	get-line, 14
fx=?, 15	get-string-all, 14
fx>=?, 15	get-string-n, 14
fx>?, 15	get-string-n!, 14
fxand, 15	get-u8, 14
fxarithmetic-shift, 15	greatest-fixnum, 15
fxarithmetic-shift-left, 15	guard, 14
fxarithmetic-shift-right, 15	hashtable-clear!, 16
fxbit-count, 15	hashtable-contains?, 16
fxbit-field, 15	hashtable-copy, 16
fxbit-set?, 15	hashtable-delete!, 16
fxcopy-bit, 15	hashtable-entries, 16
fxcopy-bit-field, 15	hashtable-equivalence-function, 16
fxdiv, 15	hashtable-hash-function, 16
fxdiv-and-mod, 15	hashtable-keys, 16
fxdiv0, 15	hashtable-mutable?, 16
fxdiv0-and-mod0, 15	hashtable-ref, 16
fxeven?, 15	hashtable-set!, 16
fxfirst-bit-set, 15	hashtable-size, 16
fxif, 15	hashtable-update!, 16
fxior, 15	hashtable?, 16
fxlength, 15	i/o-decoding-error?, 14
fxmax, 15	i/o-encoding-error-char, 14
fxmin, 15	i/o-encoding-error?, 14

i/o-error-filename , 14	letrec* , 13
i/o-error-port , 14	letrec-syntax , 13
i/o-error-position , 14	lexical-violation? , 14
i/o-error? , 14	Libraries and Collections, 9
i/o-file-already-exists-error? , 14	list , 13
i/o-file-does-not-exist-error? , 14	list->string , 13
i/o-file-is-read-only-error? , 14	list->vector , 13
i/o-file-protection-error? , 14	list-ref , 13
i/o-filename-error? , 14	list-sort , 13
i/o-invalid-position-error? , 14	list-tail , 13
i/o-port-error? , 14	list? , 13
i/o-read-error? , 14	log , 13
i/o-write-error? , 14	lookahead-char , 14
identifier-syntax , 13	lookahead-u8 , 14
identifier? , 16	magnitude , 13
if , 13	make-assertion-violation , 14
imag-part , 13	make-bytevector , 13
immutable , 14	make-custom-binary-input-port , 14
implementation-restriction-violation? , 14	make-custom-binary-input/output-port , 14
inexact , 13	make-custom-binary-output-port , 14
inexact->exact , 17	make-custom-textual-input-port , 14
inexact? , 13	make-custom-textual-input/output-port , 14
infinite? , 13	make-custom-textual-output-port , 14
input-port? , 14	make-enumeration , 16
Installing Libraries, 6	make-eq-hashtable , 16
integer->char , 13	make-eqv-hashtable , 16
integer-valued? , 13	make-error , 14
integer? , 13	make-hashtable , 16
irritants-condition? , 14	make-i/o-decoding-error , 14
lambda , 13	make-i/o-encoding-error , 14
Language Interoperability, 10	make-i/o-error , 14
latin-1-codec , 14	make-i/o-file-already-exists-error , 14
lcm , 13	make-i/o-file-does-not-exist-error , 14
least-fixnum , 15	make-i/o-file-is-read-only-error , 14
length , 13	make-i/o-file-protection-error , 14
let , 13	make-i/o-filename-error , 14
let* , 13	make-i/o-invalid-position-error , 14
let*-values , 13	
let-syntax , 13	
let-values , 13	
letrec , 13	

make-i/o-port-error, 14
 make-i/o-read-error, 14
 make-i/o-write-error, 14
 make-implementation-restriction-violation, 14
 make-irritants-condition, 14
 make-lexical-violation, 14
 make-message-condition, 14
 make-no-infinities-violation, 15
 make-no-nans-violation, 15
 make-non-continuable-violation, 14
 make-polar, 13
 make-record-constructor-descriptor, 14
 make-record-type-descriptor, 14
 make-rectangular, 13
 make-serious-condition, 14
 make-string, 13
 make-syntax-violation, 14
 make-transcoder, 14
 make-undefined-violation, 14
 make-variable-transformer, 16
 make-vector, 13
 make-violation, 14
 make-warning, 14
 make-who-condition, 14
 map, 13
 max, 13
 member, 13
 mexp, 13
 memq, 13
 memv, 13
 message-condition?, 14
 min, 13
 mod, 13
 mod0, 13
 modulo, 17
 mutable, 14
 nan?, 13
 native-endianness, 13
 native-eol-style, 14
 native-transcoder, 14
 negative?, 13
 newline, 15
 no-infinities-violation?, 15
 no-nans-violation?, 15
 non-continuable-violation?, 14
 nongenerative, 14
 not, 13
 null-environment, 17
 null?, 13
 number->string, 13
 number?, 13
 numerator, 13
 odd?, 13
 opaque, 14
 open-bytevector-input-port, 14
 open-bytevector-output-port, 14
 open-file-input-port, 14
 open-file-input/output-port, 14
 open-file-output-port, 14
 open-input-file, 15
 open-output-file, 15
 open-string-input-port, 14
 open-string-output-port, 14
 or, 13
 output-port-buffer-mode, 14
 output-port?, 14
 pair?, 13
 parent, 14
 parent-rtd, 14
 partition, 13
 peek-char, 15
 port-eof?, 14
 port-has-port-position?, 14
 port-has-set-port-position!?, 14
 port-position, 14
 port-transcoder, 14
 port?, 14
 positive?, 13
 procedure?, 13
 protocol, 14
 put-bytevector, 14
 put-char, 14

- [put-datum](#), 14
- [put-string](#), 14
- [put-u8](#), 14
- [quasiquote](#), 13
- [quasisyntax](#), 16
- [quote](#), 13
- [quotient](#), 17
- [r6rs](#), 8
- [R⁶RS Conformance](#), 11
- [R⁶RS Libraries](#), 13
- [R⁶RS Module Language](#), 8
- [R6RS: Scheme](#), 1
- [raise](#), 14
- [raise-continuable](#), 14
- [rational-valued?](#), 13
- [rational?](#), 13
- [rationalize](#), 13
- [read](#), 15
- [read-char](#), 15
- [real->flonum](#), 15
- [real-part](#), 13
- [real-valued?](#), 13
- [real?](#), 13
- [record-accessor](#), 14
- [record-constructor](#), 14
- [record-constructor-descriptor](#), 14
- [record-field-mutable?](#), 14
- [record-mutator](#), 14
- [record-predicate](#), 14
- [record-rtd](#), 14
- [record-type-descriptor](#), 14
- [record-type-descriptor?](#), 14
- [record-type-field-names](#), 14
- [record-type-generative?](#), 14
- [record-type-name](#), 14
- [record-type-opaque?](#), 14
- [record-type-parent](#), 14
- [record-type-sealed?](#), 14
- [record-type-uid](#), 14
- [record?](#), 14
- [remainder](#), 17
- [remove](#), 13
- [remp](#), 13
- [remq](#), 13
- [remv](#), 13
- [reverse](#), 13
- [rnrs/arithmetic/bitwise-6](#), 15
- [rnrs/arithmetic/fixnums-6](#), 15
- [rnrs/arithmetic/flonums-6](#), 15
- [rnrs/base-6](#), 13
- [rnrs/bytevectors-6](#), 13
- [rnrs/conditions-6](#), 14
- [rnrs/control-6](#), 13
- [rnrs/enums-6](#), 16
- [rnrs/eval-6](#), 16
- [rnrs/exceptions-6](#), 14
- [rnrs/files-6](#), 15
- [rnrs/hashtables-6](#), 16
- [rnrs/io/ports-6](#), 14
- [rnrs/io/simple-6](#), 15
- [rnrs/lists-6](#), 13
- [rnrs/mutable-pairs-6](#), 16
- [rnrs/mutable-strings-6](#), 16
- [rnrs/programs-6](#), 15
- [rnrs/r5rs-6](#), 17
- [rnrs/records/inspection-6](#), 14
- [rnrs/records/procedural-6](#), 14
- [rnrs/records/syntactic-6](#), 14
- [rnrs/sorting-6](#), 13
- [rnrs/syntax-case-6](#), 16
- [rnrs/unicode-6](#), 13
- [round](#), 13
- [Running Top-Level Programs](#), 5
- [scheme-report-environment](#), 17
- [sealed](#), 14
- [serious-condition?](#), 14
- [set!](#), 13
- [set-car!](#), 16
- [set-cdr!](#), 16
- [set-port-position!](#), 14
- [simple-conditions](#), 14
- [sin](#), 13
- [sint-list->bytevector](#), 13
- [sqrt](#), 13

- standard-error-port, 14
- standard-input-port, 14
- standard-output-port, 14
- string, 13
- string->bytevector, 14
- string->list, 13
- string->number, 13
- string->symbol, 13
- string->utf16, 13
- string->utf32, 13
- string->utf8, 13
- string-append, 13
- string-ci-hash, 16
- string-ci<=?, 13
- string-ci<?, 13
- string-ci=?, 13
- string-ci>=?, 13
- string-ci>?, 13
- string-copy, 13
- string-downcase, 13
- string-fill!, 16
- string-foldcase, 13
- string-for-each, 13
- string-hash, 16
- string-length, 13
- string-normalize-nfc, 13
- string-normalize-nfd, 13
- string-normalize-nfkc, 13
- string-normalize-nfkd, 13
- string-ref, 13
- string-set!, 16
- string-titlecase, 13
- string-upcase, 13
- string<=?, 13
- string<?, 13
- string=?, 13
- string>=?, 13
- string>?, 13
- string?, 13
- substring, 13
- symbol->string, 13
- symbol-hash, 16
- symbol=?, 13
- symbol?, 13
- syntax, 16
- syntax->datum, 16
- syntax-case, 16
- syntax-rules, 13
- syntax-violation, 16
- syntax-violation-form, 14
- syntax-violation-subform, 14
- syntax-violation?, 14
- tan, 13
- textual-port?, 14
- The Implementation of R⁶RS, 8
- transcoded-port, 14
- transcoder-codec, 14
- transcoder-eol-style, 14
- transcoder-error-handling-mode, 14
- truncate, 13
- u8-list->bytevector, 13
- uint-list->bytevector, 13
- undefined-violation?, 14
- unless, 13
- unquote, 13
- unquote-splicing, 13
- unsyntax, 16
- unsyntax-splicing, 16
- Using R⁶RS, 8
- Using R⁶RS with DrRacket, 4
- utf-16-codec, 14
- utf-8-codec, 14
- utf16->string, 13
- utf32->string, 13
- utf8->string, 13
- values, 13
- vector, 13
- vector->list, 13
- vector-fill!, 13
- vector-for-each, 13
- vector-length, 13
- vector-map, 13
- vector-ref, 13
- vector-set!, 13

- [vector-sort](#), 13
- [vector-sort!](#), 13
- [vector?](#), 13
- [violation?](#), 14
- [warning?](#), 14
- [when](#), 13
- [who-condition?](#), 14
- [with-exception-handler](#), 14
- [with-input-from-file](#), 15
- [with-output-to-file](#), 15
- [with-syntax](#), 16
- [write](#), 15
- [write-char](#), 15
- [zero?](#), 13