

R⁶RS: Standard Language

Version 4.0.2

July 4, 2008

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

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

Contents

| | | |
|----------|--|----------|
| 1 | Running Top-Level Programs | 4 |
| 2 | Installing Libraries | 5 |
| 3 | Libraries and Collections | 6 |
| 4 | Scheme Interoperability | 7 |
| 5 | R⁶RS Conformance | 8 |
| 6 | R⁶RS Libraries | 9 |
| 6.1 | <code>(rnrs base (6))</code> : Base | 9 |
| 6.2 | <code>(rnrs unicode (6))</code> : Unicode | 9 |
| 6.3 | <code>(rnrs bytevectors (6))</code> : Bytevectors | 9 |
| 6.4 | <code>(rnrs lists (6))</code> : List utilities | 9 |
| 6.5 | <code>(rnrs sorting (6))</code> : Sorting | 9 |
| 6.6 | <code>(rnrs control (6))</code> : Control Structures | 9 |
| 6.7 | <code>(rnrs records syntactic (6))</code> : Records: Syntactic | 10 |
| 6.8 | <code>(rnrs records procedural (6))</code> : Records: Procedural | 10 |
| 6.9 | <code>(rnrs records inspection (6))</code> : Records: Inspection | 10 |
| 6.10 | <code>(rnrs exceptions (6))</code> : Exceptions and Conditions | 10 |
| 6.11 | <code>(rnrs conditions (6))</code> : Exceptions and Conditions | 10 |
| 6.12 | <code>(rnrs io ports (6))</code> : I/O: Ports | 10 |
| 6.13 | <code>(rnrs io simple (6))</code> : I/O: Simple | 11 |
| 6.14 | <code>(rnrs files (6))</code> : File System | 11 |
| 6.15 | <code>(rnrs programs (6))</code> : Command-line Access and Exit Values | 11 |

| | | |
|------|--|----|
| 6.16 | <code>(rnrs arithmetic fixnums (6))</code> : Arithmetic: Fixnums | 11 |
| 6.17 | <code>(rnrs arithmetic flonums (6))</code> : Arithmetic: Flonums | 11 |
| 6.18 | <code>(rnrs arithmetic bitwise (6))</code> : Arithmetic: Bitwise | 11 |
| 6.19 | <code>(rnrs syntax-case (6))</code> : Syntax-Case | 12 |
| 6.20 | <code>(rnrs hashtables (6))</code> : Hashtables | 12 |
| 6.21 | <code>(rnrs enums (6))</code> : Enumerations | 12 |
| 6.22 | <code>(rnrs eval (6))</code> : Eval | 12 |
| 6.23 | <code>(rnrs mutable-pairs (6))</code> : Mutable Pairs | 12 |
| 6.24 | <code>(rnrs mutable-strings (6))</code> : Mutable Strings | 12 |
| 6.25 | <code>(rnrs r5rs (6))</code> : R5RS Compatibility | 13 |

| | | |
|--------------|--|-----------|
| Index | | 14 |
|--------------|--|-----------|

1 Running Top-Level Programs

To run a top-level program, either:

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

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

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

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

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

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

For example, if "hi.scm" contains

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

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

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

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

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

For example, if "hi.ss" contains

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

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

2 Installing Libraries

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

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

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

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

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

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

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

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

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

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

3 Libraries and Collections

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

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

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

Examples (assuming a typical PLT Scheme installation):

```
(rnrs io simple (6)) means (lib "rnrs/io/simple-6.ss")
(rnrs)                means (lib "rnrs/main-6.ss")
(rnrs (6))            means (lib "rnrs/main-6.ss")
(scheme base)        means (lib "scheme/base.ss")
```

4 Scheme Interoperability

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

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

5 R⁶RS Conformance

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

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

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

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

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

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

6 R⁶RS Libraries

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

`(require rnrs/base-6)`

Original specification: Base

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

`(require rnrs/unicode-6)`

Original specification: Unicode

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

`(require rnrs/bytevectors-6)`

Original specification: Bytevectors

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

`(require rnrs/lists-6)`

Original specification: List utilities

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

`(require rnrs/sorting-6)`

Original specification: Sorting

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

`(require rnrs/control-6)`

Original specification: Control Structures

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

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

Original specification: Records: Syntactic

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

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

Original specification: Records: Procedural

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

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

Original specification: Records: Inspection

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

`(require nrns/exceptions-6)`

Original specification: Exceptions and Conditions

See also §5 “R⁶RS Conformance”.

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

`(require nrns/conditions-6)`

Original specification: Exceptions and Conditions

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

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

Original specification: I/O: Ports

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

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

Original specification: I/O: Simple

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

`(require rnrs/files-6)`

Original specification: File System

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

`(require rnrs/programs-6)`

Original specification: Command-line Access and Exit Values

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

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

Original specification: Arithmetic: Fixnums

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

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

Original specification: Arithmetic: Flonums

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

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

Original specification: Arithmetic: Bitwise

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

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

Original specification: Syntax-Case

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

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

Original specification: Hashtables

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

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

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

Original specification: Enumerations

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

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

Original specification: Eval

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

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

Original specification: Mutable Pairs

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

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

Original specification: Mutable Strings

6.25 `(nrns r5rs (6))`: **R5RS Compatibility**

`(require nrns/r5rs-6)`

Original specification: R5RS Compatibility

See also §5 “R⁶RS Conformance”.

Index

`&assertion`, 10
`&condition`, 10
`&error`, 10
`&i/o`, 10
`&i/o-decoding`, 10
`&i/o-encoding`, 10
`&i/o-file-already-exists`, 10
`&i/o-file-does-not-exist`, 10
`&i/o-file-is-read-only`, 10
`&i/o-file-protection`, 10
`&i/o-filename`, 10
`&i/o-invalid-position`, 10
`&i/o-port`, 10
`&i/o-read`, 10
`&i/o-write`, 10
`&implementation-restriction`, 10
`&irritants`, 10
`&lexical`, 10
`&message`, 10
`&no-infinities`, 11
`&no-nans`, 11
`&non-continuable`, 10
`&serious`, 10
`&syntax`, 10
`&undefined`, 10
`&violation`, 10
`&warning`, 10
`&who`, 10
`(rnrs arithmetic bitwise (6))`:
 Arithmetic: Bitwise, 11
`(rnrs arithmetic fixnums (6))`:
 Arithmetic: Fixnums, 11
`(rnrs arithmetic flonums (6))`:
 Arithmetic: Flonums, 11
`(rnrs base (6))`: Base, 9
`(rnrs bytevectors (6))`: Bytevectors, 9
`(rnrs conditions (6))`: Exceptions and
 Conditions, 10
`(rnrs control (6))`: Control Structures,
 9
`(rnrs enums (6))`: Enumerations, 12
`(rnrs eval (6))`: Eval, 12
`(rnrs exceptions (6))`: Exceptions and
 Conditions, 10
`(rnrs files (6))`: File System, 11
`(rnrs hashtables (6))`: Hashtables, 12
`(rnrs io ports (6))`: I/O: Ports, 10
`(rnrs io simple (6))`: I/O: Simple, 11
`(rnrs lists (6))`: List utilities, 9
`(rnrs mutable-pairs (6))`: Mutable
 Pairs, 12
`(rnrs mutable-strings (6))`: Mutable
 Strings, 12
`(rnrs programs (6))`: Command-line
 Access and Exit Values, 11
`(rnrs r5rs (6))`: R5RS Compatibility,
 13
`(rnrs records inspection (6))`:
 Records: Inspection, 10
`(rnrs records procedural (6))`:
 Records: Procedural, 10
`(rnrs records syntactic (6))`:
 Records: Syntactic, 10
`(rnrs sorting (6))`: Sorting, 9
`(rnrs syntax-case (6))`: Syntax-Case,
 12
`(rnrs unicode (6))`: Unicode, 9
`*`, 9
`+`, 9
`++path`, 5
`-`, 9
`...`, 9
`...`, 12
`/`, 9
`<`, 9
`<=`, 9
`=`, 9
`=>`, 9
`=>`, 10
`>`, 9
`>=`, 9
`->`, 9
`->`, 12
`abs`

acos, 9
and, 9
angle, 9
append, 9
apply, 9
asin, 9
assert, 9
assertion-violation, 9
assertion-violation?, 10
assoc, 9
assp, 9
assq, 9
assv, 9
atan, 9
begin
binary-port?, 10
bitwise-and, 11
bitwise-arithmetic-shift, 11
bitwise-arithmetic-shift-left, 11
bitwise-arithmetic-shift-right, 11
bitwise-bit-count, 11
bitwise-bit-field, 11
bitwise-bit-set?, 11
bitwise-copy-bit, 11
bitwise-copy-bit-field, 11
bitwise-first-bit-set, 11
bitwise-if, 11
bitwise-ior, 11
bitwise-length, 11
bitwise-not, 11
bitwise-reverse-bit-field, 11
bitwise-rotate-bit-field, 11
bitwise-xor, 11
boolean=?, 9
boolean?, 9
bound-identifier=?, 12
buffer-mode, 10
buffer-mode?, 10
bytevector->sint-list, 9
bytevector->string, 10
bytevector->u8-list, 9
bytevector->uint-list, 9
bytevector-copy, 9
bytevector-copy!, 9
bytevector-fill!, 9
bytevector-ieee-double-native-ref,
9
bytevector-ieee-double-native-
set!, 9
bytevector-ieee-double-ref, 9
bytevector-ieee-single-native-ref,
9
bytevector-ieee-single-native-
set!, 9
bytevector-ieee-single-ref, 9
bytevector-length, 9
bytevector-s16-native-ref, 9
bytevector-s16-native-set!, 9
bytevector-s16-ref, 9
bytevector-s16-set!, 9
bytevector-s32-native-ref, 9
bytevector-s32-native-set!, 9
bytevector-s32-ref, 9
bytevector-s32-set!, 9
bytevector-s64-native-ref, 9
bytevector-s64-native-set!, 9
bytevector-s64-ref, 9
bytevector-s64-set!, 9
bytevector-s8-ref, 9
bytevector-s8-set!, 9
bytevector-sint-ref, 9
bytevector-sint-set!, 9
bytevector-u16-native-ref, 9
bytevector-u16-native-set!, 9
bytevector-u16-ref, 9
bytevector-u16-set!, 9
bytevector-u32-native-ref, 9
bytevector-u32-native-set!, 9
bytevector-u32-ref, 9
bytevector-u32-set!, 9
bytevector-u64-native-ref, 9
bytevector-u64-native-set!, 9
bytevector-u64-ref, 9
bytevector-u64-set!, 9

bytevector-u8-ref, 9
 bytevector-u8-set!, 9
 bytevector-uint-ref, 9
 bytevector-uint-set!, 9
 bytevector=?, 9
 bytevector?, 9
 caar
 cadr, 9
 call-with-bytevector-output-port,
 10
 call-with-current-continuation, 9
 call-with-input-file, 11
 call-with-output-file, 11
 call-with-port, 10
 call-with-string-output-port, 10
 call-with-values, 9
 call/cc, 9
 car, 9
 case, 9
 case-lambda, 9
 cdddar, 9
 cddddr, 9
 cdr, 9
 ceiling, 9
 char->integer, 9
 char-alphabetic?, 9
 char-ci<=?, 9
 char-ci<?, 9
 char-ci=?, 9
 char-ci>=?, 9
 char-ci>?, 9
 char-downcase, 9
 char-foldcase, 9
 char-general-category, 9
 char-lower-case?, 9
 char-numeric?, 9
 char-title-case?, 9
 char-titlecase, 9
 char-upcase, 9
 char-upper-case?, 9
 char-whitespace?, 9
 char<=?, 9
 char<?, 9
 char=?, 9
 char>=?, 9
 char>?, 9
 char?, 9
 close-input-port, 11
 close-output-port, 11
 close-port, 10
 command-line, 11
 complex?, 9
 cond, 9
 condition, 10
 condition-accessor, 10
 condition-irritants, 10
 condition-message, 10
 condition-predicate, 10
 condition-who, 10
 condition?, 10
 cons, 9
 cons*, 9
 cos, 9
 current-error-port, 10
 current-input-port, 10
 current-output-port, 10
 datum->syntax
 define, 9
 define-condition-type, 10
 define-enumeration, 12
 define-record-type, 10
 define-syntax, 9
 delay, 13
 delete-file, 11
 denominator, 9
 display, 11
 div, 9
 div-and-mod, 9
 div0, 9
 div0-and-mod0, 9
 do, 9
 dynamic-wind, 9
 else
 else, 10

endianness, 9
enum-set->list, 12
enum-set-complement, 12
enum-set-constructor, 12
enum-set-difference, 12
enum-set-indexer, 12
enum-set-intersection, 12
enum-set-member?, 12
enum-set-projection, 12
enum-set-subset?, 12
enum-set-union, 12
enum-set-universe, 12
enum-set=?, 12
environment, 12
eof-object, 10
eof-object?, 10
eol-style, 10
eq?, 9
equal-hash, 12
equal?, 9
eqv?, 9
error, 9
error-handling-mode, 10
error?, 10
eval, 12
even?, 9
exact, 9
exact->inexact, 13
exact-integer-sqrt, 9
exact?, 9
exists, 9
exit, 11
exp, 9
expt, 9
fields
file-exists?, 11
file-options, 10
filter, 9
find, 9
finite?, 9
fixnum->flonum, 11
fixnum-width, 11
fixnum?, 11
fl*, 11
fl+, 11
fl-, 11
fl/, 11
fl<=?, 11
fl<?, 11
fl=?, 11
fl>=?, 11
fl>?, 11
flabs, 11
flacos, 11
flasin, 11
flatan, 11
flceiling, 11
flcos, 11
fldenominator, 11
fldiv, 11
fldiv-and-mod, 11
fldiv0, 11
fldiv0-and-mod0, 11
fleven?, 11
flexp, 11
flexpt, 11
flfinite?, 11
flfloor, 11
flinfinite?, 11
flinteger?, 11
fllog, 11
flmax, 11
flmin, 11
flmod, 11
flmod0, 11
flnan?, 11
flnegative?, 11
flnumerator, 11
flodd?, 11
flonum?, 11
floor, 9
flpositive?, 11
flround, 11
flsin, 11

flsqrt, 11
 fltan, 11
 fltruncate, 11
 flush-output-port, 10
 flzero?, 11
 fold-left, 9
 fold-right, 9
 for-all, 9
 for-each, 9
 force, 13
 free-identifier=?, 12
 fx*, 11
 fx*/carry, 11
 fx+, 11
 fx+/carry, 11
 fx-, 11
 fx-/carry, 11
 fx<=?, 11
 fx<?, 11
 fx=?, 11
 fx>=?, 11
 fx>?, 11
 fxand, 11
 fxarithmetic-shift, 11
 fxarithmetic-shift-left, 11
 fxarithmetic-shift-right, 11
 fxbit-count, 11
 fxbit-field, 11
 fxbit-set?, 11
 fxcopy-bit, 11
 fxcopy-bit-field, 11
 fxdiv, 11
 fxdiv-and-mod, 11
 fxdiv0, 11
 fxdiv0-and-mod0, 11
 fxeven?, 11
 fxfirst-bit-set, 11
 fxif, 11
 fxior, 11
 fxlength, 11
 fxmax, 11
 fxmin, 11
 fxmod, 11
 fxmod0, 11
 fxnegative?, 11
 fxnot, 11
 fxodd?, 11
 fxpositive?, 11
 fxreverse-bit-field, 11
 fxrotate-bit-field, 11
 fxxor, 11
 fxzero?, 11
 gcd
 generate-temporaries, 12
 get-bytevector-all, 10
 get-bytevector-n, 10
 get-bytevector-n!, 10
 get-bytevector-some, 10
 get-char, 10
 get-datum, 10
 get-line, 10
 get-string-all, 10
 get-string-n, 10
 get-string-n!, 10
 get-u8, 10
 greatest-fixnum, 11
 guard, 10
 hashtable-clear!
 hashtable-contains?, 12
 hashtable-copy, 12
 hashtable-delete!, 12
 hashtable-entries, 12
 hashtable-equivalence-function, 12
 hashtable-hash-function, 12
 hashtable-keys, 12
 hashtable-mutable?, 12
 hashtable-ref, 12
 hashtable-set!, 12
 hashtable-size, 12
 hashtable-update!, 12
 hashtable?, 12
 i/o-decoding-error?
 i/o-encoding-error-char, 10
 i/o-encoding-error?, 10

[i/o-error-filename](#), 10
[i/o-error-port](#), 10
[i/o-error-position](#), 10
[i/o-error?](#), 10
[i/o-file-already-exists-error?](#), 10
[i/o-file-does-not-exist-error?](#), 10
[i/o-file-is-read-only-error?](#), 10
[i/o-file-protection-error?](#), 10
[i/o-filename-error?](#), 10
[i/o-invalid-position-error?](#), 10
[i/o-port-error?](#), 10
[i/o-read-error?](#), 10
[i/o-write-error?](#), 10
[identifier-syntax](#), 9
[identifier?](#), 12
[if](#), 9
[imag-part](#), 9
[immutable](#), 10
[implementation-restriction-violation?](#), 10
[inexact](#), 9
[inexact->exact](#), 13
[inexact?](#), 9
[infinite?](#), 9
[input-port?](#), 10
[Installing Libraries](#), 5
[integer->char](#), 9
[integer-valued?](#), 9
[integer?](#), 9
[irritants-condition?](#), 10
[lambda](#)
[latin-1-codec](#), 10
[lcm](#), 9
[least-fixnum](#), 11
[length](#), 9
[let](#), 9
[let*](#), 9
[let*-values](#), 9
[let-syntax](#), 9
[let-values](#), 9
[letrec](#), 9
[letrec*](#), 9
[letrec-syntax](#), 9
[lexical-violation?](#), 10
[Libraries and Collections](#), 6
[list](#), 9
[list->string](#), 9
[list->vector](#), 9
[list-ref](#), 9
[list-sort](#), 9
[list-tail](#), 9
[list?](#), 9
[log](#), 9
[lookahead-char](#), 10
[lookahead-u8](#), 10
[magnitude](#)
[make-assertion-violation](#), 10
[make-bytevector](#), 9
[make-custom-binary-input-port](#), 10
[make-custom-binary-input/output-port](#), 10
[make-custom-binary-output-port](#), 10
[make-custom-textual-input-port](#), 10
[make-custom-textual-input/output-port](#), 10
[make-custom-textual-output-port](#), 10
[make-enumeration](#), 12
[make-eq-hashtable](#), 12
[make-eqv-hashtable](#), 12
[make-error](#), 10
[make-hashtable](#), 12
[make-i/o-decoding-error](#), 10
[make-i/o-encoding-error](#), 10
[make-i/o-error](#), 10
[make-i/o-file-already-exists-error](#), 10
[make-i/o-file-does-not-exist-error](#), 10
[make-i/o-file-is-read-only-error](#), 10
[make-i/o-file-protection-error](#), 10
[make-i/o-filename-error](#), 10
[make-i/o-invalid-position-error](#), 10
[make-i/o-port-error](#), 10

make-i/o-read-error, 10
 make-i/o-write-error, 10
 make-implementation-restriction-violation, 10
 make-irritants-condition, 10
 make-lexical-violation, 10
 make-message-condition, 10
 make-no-infinities-violation, 11
 make-no-nans-violation, 11
 make-non-continuable-violation, 10
 make-polar, 9
 make-record-constructor-descriptor, 10
 make-record-type-descriptor, 10
 make-rectangular, 9
 make-serious-condition, 10
 make-string, 9
 make-syntax-violation, 10
 make-transcoder, 10
 make-undefined-violation, 10
 make-variable-transformer, 12
 make-vector, 9
 make-violation, 10
 make-warning, 10
 make-who-condition, 10
 map, 9
 max, 9
 member, 9
 memq, 9
 memq, 9
 memv, 9
 message-condition?, 10
 min, 9
 mod, 9
 mod0, 9
 modulo, 13
 mutable, 10
 nan?
 native-endianness, 9
 native-eol-style, 10
 native-transcoder, 10
 negative?, 9
 newline, 11
 no-infinities-violation?, 11
 no-nans-violation?, 11
 non-continuable-violation?, 10
 nongenerative, 10
 not, 9
 null-environment, 13
 null?, 9
 number->string, 9
 number?, 9
 numerator, 9
 odd?
 opaque, 10
 open-bytevector-input-port, 10
 open-bytevector-output-port, 10
 open-file-input-port, 10
 open-file-input/output-port, 10
 open-file-output-port, 10
 open-input-file, 11
 open-output-file, 11
 open-string-input-port, 10
 open-string-output-port, 10
 or, 9
 output-port-buffer-mode, 10
 output-port?, 10
 pair?
 parent, 10
 parent-rtd, 10
 partition, 9
 peek-char, 11
 port-eof?, 10
 port-has-port-position?, 10
 port-has-set-port-position!?, 10
 port-position, 10
 port-transcoder, 10
 port?, 10
 positive?, 9
 procedure?, 9
 protocol, 10
 put-bytevector, 10
 put-char, 10
 put-datum, 10

[put-string](#), 10
[put-u8](#), 10
[quasiquote](#)
[quasisyntax](#), 12
[quote](#), 9
[quotient](#), 13
R⁶RS Conformance
R⁶RS Libraries, 9
R⁶RS: Standard Language, 1
[raise](#), 10
[raise-continuable](#), 10
[rational-valued?](#), 9
[rational?](#), 9
[rationalize](#), 9
[read](#), 11
[read-char](#), 11
[real->flonum](#), 11
[real-part](#), 9
[real-valued?](#), 9
[real?](#), 9
[record-accessor](#), 10
[record-constructor](#), 10
[record-constructor-descriptor](#), 10
[record-field-mutable?](#), 10
[record-mutator](#), 10
[record-predicate](#), 10
[record-rtd](#), 10
[record-type-descriptor](#), 10
[record-type-descriptor?](#), 10
[record-type-field-names](#), 10
[record-type-generative?](#), 10
[record-type-name](#), 10
[record-type-opaque?](#), 10
[record-type-parent](#), 10
[record-type-sealed?](#), 10
[record-type-uid](#), 10
[record?](#), 10
[remainder](#), 13
[remove](#), 9
[remp](#), 9
[remq](#), 9
[remv](#), 9
[reverse](#), 9
[rnrs/arithmetic/bitwise-6](#), 11
[rnrs/arithmetic/fixnums-6](#), 11
[rnrs/arithmetic/flonums-6](#), 11
[rnrs/base-6](#), 9
[rnrs/bytevectors-6](#), 9
[rnrs/conditions-6](#), 10
[rnrs/control-6](#), 9
[rnrs/enums-6](#), 12
[rnrs/eval-6](#), 12
[rnrs/exceptions-6](#), 10
[rnrs/files-6](#), 11
[rnrs/hashtables-6](#), 12
[rnrs/io/ports-6](#), 10
[rnrs/io/simple-6](#), 11
[rnrs/lists-6](#), 9
[rnrs/mutable-pairs-6](#), 12
[rnrs/mutable-strings-6](#), 12
[rnrs/programs-6](#), 11
[rnrs/r5rs-6](#), 13
[rnrs/records/inspection-6](#), 10
[rnrs/records/procedural-6](#), 10
[rnrs/records/syntactic-6](#), 10
[rnrs/sorting-6](#), 9
[rnrs/syntax-case-6](#), 12
[rnrs/unicode-6](#), 9
[round](#), 9
Running Top-Level Programs, 4
Scheme Interoperability
[scheme-report-environment](#), 13
[sealed](#), 10
[serious-condition?](#), 10
[set!](#), 9
[set-car!](#), 12
[set-cdr!](#), 12
[set-port-position!](#), 10
[simple-conditions](#), 10
[sin](#), 9
[sint-list->bytevector](#), 9
[sqrt](#), 9
[standard-error-port](#), 10
[standard-input-port](#), 10

standard-output-port, 10
 string, 9
 string->bytevector, 10
 string->list, 9
 string->number, 9
 string->symbol, 9
 string->utf16, 9
 string->utf32, 9
 string->utf8, 9
 string-append, 9
 string-ci-hash, 12
 string-ci<=?, 9
 string-ci<?, 9
 string-ci=?, 9
 string-ci>=?, 9
 string-ci>?, 9
 string-copy, 9
 string-downcase, 9
 string-fill!, 12
 string-foldcase, 9
 string-for-each, 9
 string-hash, 12
 string-length, 9
 string-normalize-nfc, 9
 string-normalize-nfd, 9
 string-normalize-nfkc, 9
 string-normalize-nfkd, 9
 string-ref, 9
 string-set!, 12
 string-titlecase, 9
 string-upcase, 9
 string<=?, 9
 string<?, 9
 string=?, 9
 string>=?, 9
 string>?, 9
 string?, 9
 substring, 9
 symbol->string, 9
 symbol-hash, 12
 symbol=?, 9
 symbol?, 9
 syntax, 12
 syntax->datum, 12
 syntax-case, 12
 syntax-rules, 9
 syntax-violation, 12
 syntax-violation-form, 10
 syntax-violation-subform, 10
 syntax-violation?, 10
 tan
 textual-port?, 10
 transcoded-port, 10
 transcoder-codec, 10
 transcoder-eol-style, 10
 transcoder-error-handling-mode, 10
 truncate, 9
 u8-list->bytevector
 uint-list->bytevector, 9
 undefined-violation?, 10
 unless, 9
 unquote, 9
 unquote-splicing, 9
 unsyntax, 12
 unsyntax-splicing, 12
 utf-16-codec, 10
 utf-8-codec, 10
 utf16->string, 9
 utf32->string, 9
 utf8->string, 9
 values
 vector, 9
 vector->list, 9
 vector-fill!, 9
 vector-for-each, 9
 vector-length, 9
 vector-map, 9
 vector-ref, 9
 vector-set!, 9
 vector-sort, 9
 vector-sort!, 9
 vector?, 9
 violation?, 10
 warning?

when, 9
who-condition?, 10
with-exception-handler, 10
with-input-from-file, 11
with-output-to-file, 11
with-syntax, 12
write, 11
write-char, 11
zero?