

# R6RS: Standard Language

Version 4.2.5

April 2, 2010

The Revised<sup>6</sup> Report on the Algorithmic Language Scheme defines a dialect of Scheme.  
We use *R<sup>6</sup>RS* to refer to both the standard and the language defined by the standard.

*R<sup>6</sup>RS* defines both *libraries* and *top-level programs*. Both correspond to PLT Scheme *modules* (see §6 “Modules”). That is, although *R<sup>6</sup>RS* 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 *R<sup>6</sup>RS* top-level program cannot export any bindings to other modules.

See §21 “Dialects of Scheme” for general information about different dialects of Scheme within PLT Scheme.

# Contents

<b>1</b>	<b>Running Top-Level Programs</b>	<b>4</b>
<b>2</b>	<b>Installing Libraries</b>	<b>5</b>
<b>3</b>	<b>R<sup>6</sup>RS Module Language</b>	<b>6</b>
<b>4</b>	<b>Libraries and Collections</b>	<b>7</b>
<b>5</b>	<b>Scheme Interoperability</b>	<b>8</b>
<b>6</b>	<b>R<sup>6</sup>RS Conformance</b>	<b>9</b>
<b>7</b>	<b>R<sup>6</sup>RS Libraries</b>	<b>11</b>
7.1	(rnrs base (6)): Base . . . . .	11
7.2	(rnrs unicode (6)): Unicode . . . . .	11
7.3	(rnrs bytewectors (6)): Bytewectors . . . . .	11
7.4	(rnrs lists (6)): List utilities . . . . .	11
7.5	(rnrs sorting (6)): Sorting . . . . .	11
7.6	(rnrs control (6)): Control Structures . . . . .	11
7.7	(rnrs records syntactic (6)): Records: Syntactic . . . . .	12
7.8	(rnrs records procedural (6)): Records: Procedural . . . . .	12
7.9	(rnrs records inspection (6)): Records: Inspection . . . . .	12
7.10	(rnrs exceptions (6)): Exceptions . . . . .	12
7.11	(rnrs conditions (6)): Conditions . . . . .	12
7.12	(rnrs io ports (6)): I/O: Ports . . . . .	12
7.13	(rnrs io simple (6)): I/O: Simple . . . . .	13
7.14	(rnrs files (6)): File System . . . . .	13

7.15 ( <code>rnrs programs (6)</code> ): Command-line Access and Exit Values . . . . .	13
7.16 ( <code>rnrs arithmetic fixnums (6)</code> ): Arithmetic: Fixnums . . . . .	13
7.17 ( <code>rnrs arithmetic flonums (6)</code> ): Arithmetic: Flonums . . . . .	13
7.18 ( <code>rnrs arithmetic bitwise (6)</code> ): Arithmetic: Bitwise . . . . .	13
7.19 ( <code>rnrs syntax-case (6)</code> ): Syntax-Case . . . . .	14
7.20 ( <code>rnrs hashtables (6)</code> ): Hashtables . . . . .	14
7.21 ( <code>rnrs enums (6)</code> ): Enumerations . . . . .	14
7.22 ( <code>rnrs eval (6)</code> ): Eval . . . . .	14
7.23 ( <code>rnrs mutable-pairs (6)</code> ): Mutable Pairs . . . . .	14
7.24 ( <code>rnrs mutable-strings (6)</code> ): Mutable Strings . . . . .	14
7.25 ( <code>rnrs r5rs (6)</code> ): R5RS Compatibility . . . . .	15
<b>Index</b>	<b>16</b>

# 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 R<sup>6</sup>RS 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 R<sup>6</sup>RS 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 R<sup>6</sup>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.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 §4 “Libraries and Collections” for information on how R<sup>6</sup>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-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<sup>6</sup>RS library prefixed with `#!r6rs`, then `plt-r6rs ++path <dir>` directs the R<sup>6</sup>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 §4 “Libraries and Collections”. Multiple paths can be provided with multiple uses of `++path`; the paths are searched in order, and before the installation’s collections.

## 3 R<sup>6</sup>RS Module Language

```
#lang r6rs
```

The `r6rs` language is usually used in the form `#!r6rs`, which is equivalent to `#lang r6rs` and is also valid R<sup>6</sup>RS syntax.

The `r6rs` module language provides only a `#%module-begin` binding, which is used to process the entire module body (see [module](#)). It allows the body of a module to use the syntax of either a R<sup>6</sup>RS library or a R<sup>6</sup>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<sup>6</sup>RS library, while a module body that starts with an `import` form defined an R<sup>6</sup>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.

## 4 Libraries and Collections

An R<sup>6</sup>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 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 R<sup>6</sup>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<sup>6</sup>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<sup>6</sup>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"`, and `".sls"`. When resolving version constraints, these extensions are all tried when looking for matches.

To ensure that all R<sup>6</sup>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<sup>6</sup>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 PLT Scheme installation):

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

## 5 Scheme Interoperability

Using the conversion rules in §4 “Libraries and Collections”, and R<sup>6</sup>RS 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 R<sup>6</sup>RS.

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

## 6 R<sup>6</sup>RS Conformance

PLT Scheme's R<sup>6</sup>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-raised without restoring the continuation to the one that raised the exception.

This difference can be made visible using `dynamic-wind`. According to R<sup>6</sup>RS, the following program should print “in” and “out” twice, but each prints once using PLT Scheme:

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

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

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

- Inexact numbers are printed without a precision indicator, and precision indicators are ignored on input (e.g., `0.5|7` is read the same as `0.5`).
- Word boundaries for `string-downcase`, `string-upcase`, and `string-titlecase` are not determined as specified by Unicode Standard Annex #29.
- When an identifier bound by `letrec` or `letrec*` is referenced before it is bound, an exception is not raised; instead, the reference produces `#<undefined>`.
- A custom textual port must represent positions using integers, and the positions must correspond to bytes in a UTF-8 encoding of the port’s data. For custom ports (byte or character) that support both input and output, beware that buffered input can create a mismatch between the position implemented by the custom procedures and the port’s current position; the result from a custom position procedure is automatically adjusted to account for buffering, and setting the port’s position flushes all buffered bytes, but writing after a read does *not* automatically reset the port’s position to counteract the effects of buffering.
- The bindings in a namespace produced by `null-environment` or `scheme-report-environment` correspond to R<sup>5</sup>RS bindings instead of R<sup>6</sup>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.

## 7 R<sup>6</sup>S Libraries

### 7.1 (rnrs base (6)): Base

(require rnrs/base-6)

Original specification: Base

### 7.2 (rnrs unicode (6)): Unicode

(require rnrs/unicode-6)

Original specification: Unicode

### 7.3 (rnrs bytevectors (6)): Bytevectors

(require rnrs/bytevectors-6)

Original specification: Bytevectors

### 7.4 (rnrs lists (6)): List utilities

(require rnrs/lists-6)

Original specification: List utilities

### 7.5 (rnrs sorting (6)): Sorting

(require rnrs/sorting-6)

Original specification: Sorting

### 7.6 (rnrs control (6)): Control Structures

(require rnrs/control-6)

Original specification: Control Structures

## 7.7 (rnrs records syntactic (6)): Records: Syntactic

(require rnrs/records/syntactic-6)

Original specification: Records: Syntactic

## 7.8 (rnrs records procedural (6)): Records: Procedural

(require rnrs/records/procedural-6)

Original specification: Records: Procedural

## 7.9 (rnrs records inspection (6)): Records: Inspection

(require rnrs/records/inspection-6)

Original specification: Records: Inspection

## 7.10 (rnrs exceptions (6)): Exceptions

(require rnrs/exceptions-6)

Original specification: Exceptions

See also §6 “R<sup>6</sup>RS Conformance”.

## 7.11 (rnrs conditions (6)): Conditions

(require rnrs/conditions-6)

Original specification: Conditions

## 7.12 (rnrs io ports (6)): I/O: Ports

(require rnrs/io/ports-6)

Original specification: I/O: Ports

### **7.13 (rnrs io simple (6)): I/O: Simple**

(require rnrs/io/simple-6)

Original specification: I/O: Simple

### **7.14 (rnrs files (6)): File System**

(require rnrs/files-6)

Original specification: File System

### **7.15 (rnrs programs (6)): Command-line Access and Exit Values**

(require rnrs/programs-6)

Original specification: Command-line Access and Exit Values

### **7.16 (rnrs arithmetic fixnums (6)): Arithmetic: Fixnums**

(require rnrs/arithmetic/fixnums-6)

Original specification: Arithmetic: Fixnums

### **7.17 (rnrs arithmetic flonums (6)): Arithmetic: Flonums**

(require rnrs/arithmetic/flonums-6)

Original specification: Arithmetic: Flonums

### **7.18 (rnrs arithmetic bitwise (6)): Arithmetic: Bitwise**

(require rnrs/arithmetic/bitwise-6)

Original specification: Arithmetic: Bitwise

## 7.19 (rnrs syntax-case (6)): Syntax-Case

(require rnrs/syntax-case-6)

Original specification: Syntax-Case

## 7.20 (rnrs hashtables (6)): Hashtables

(require rnrs/hashtables-6)

Original specification: Hashtables

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

## 7.21 (rnrs enums (6)): Enumerations

(require rnrs/enums-6)

Original specification: Enumerations

## 7.22 (rnrs eval (6)): Eval

(require rnrs/eval-6)

Original specification: Eval

## 7.23 (rnrs mutable-pairs (6)): Mutable Pairs

(require rnrs/mutable-pairs-6)

Original specification: Mutable Pairs

## 7.24 (rnrs mutable-strings (6)): Mutable Strings

(require rnrs/mutable-strings-6)

Original specification: Mutable Strings

## **7.25 (rnrs r5rs (6)): R5RS Compatibility**

(require rnrs/r5rs-6)

Original specification: R5RS Compatibility

See also §6 “R<sup>6</sup>RS Conformance”.

## Index

#%module-begin, 6  
&assertion, 12  
&condition, 12  
&error, 12  
&i/o, 12  
&i/o-decoding, 12  
&i/o-encoding, 12  
&i/o-file-already-exists, 12  
&i/o-file-does-not-exist, 12  
&i/o-file-is-read-only, 12  
&i/o-file-protection, 12  
&i/o-filename, 12  
&i/o-invalid-position, 12  
&i/o-port, 12  
&i/o-read, 12  
&i/o-write, 12  
&implementation-restriction, 12  
&irritants, 12  
&lexical, 12  
&message, 12  
&no-infinities, 13  
&no-nans, 13  
&non-continuable, 12  
&serious, 12  
&syntax, 12  
&undefined, 12  
&violation, 12  
&warning, 12  
&who, 12  
(rnrs arithmetic bitwise (6)):  
    Arithmetic: Bitwise, 13  
(rnrs arithmetic fixnums (6)):  
    Arithmetic: Fixnums, 13  
(rnrs arithmetic flonums (6)):  
    Arithmetic: Flonums, 13  
(rnrs base (6)): Base, 11  
(rnrs bytewectors (6)): Bytewectors,  
    11  
(rnrs conditions (6)): Conditions, 12  
(rnrs control (6)): Control Structures,  
    11  
  
(rnrs enums (6)): Enumerations, 14  
(rnrs eval (6)): Eval, 14  
(rnrs exceptions (6)): Exceptions, 12  
(rnrs files (6)): File System, 13  
(rnrs hashtables (6)): Hashtables, 14  
(rnrs io ports (6)): I/O: Ports, 12  
(rnrs io simple (6)): I/O: Simple, 13  
(rnrs lists (6)): List utilities, 11  
(rnrs mutable-pairs (6)): Mutable  
    Pairs, 14  
(rnrs mutable-strings (6)): Mutable  
    Strings, 14  
(rnrs programs (6)): Command-line  
    Access and Exit Values, 13  
(rnrs r5rs (6)): R5RS Compatibility,  
    15  
(rnrs records inspection (6)):  
    Records: Inspection, 12  
(rnrs records procedural (6)):  
    Records: Procedural, 12  
(rnrs records syntactic (6)):  
    Records: Syntactic, 12  
(rnrs sorting (6)): Sorting, 11  
(rnrs syntax-case (6)): Syntax-Case,  
    14  
(rnrs unicode (6)): Unicode, 11  
\*, 11  
+, 11  
++path, 5  
-, 11  
..., 11  
..., 14  
/, 11  
<, 11

```

acos, 11
and, 11
angle, 11
append, 11
apply, 11
asin, 11
assert, 11
assertion-violation, 11
assertion-violation?, 12
assoc, 11
assp, 11
assq, 11
assv, 11
atan, 11
begin, 11
binary-port?, 12
bitwise-and, 13
bitwise-arithmetic-shift, 13
bitwise-arithmetic-shift-left, 13
bitwise-arithmetic-shift-right, 13
bitwise-bit-count, 13
bitwise-bit-field, 13
bitwise-bit-set?, 13
bitwise-copy-bit, 13
bitwise-copy-bit-field, 13
bitwise-first-bit-set, 13
bitwise-if, 13
bitwise-ior, 13
bitwise-length, 13
bitwise-not, 13
bitwise-reverse-bit-field, 13
bitwise-rotate-bit-field, 13
bitwise-xor, 13
boolean=?, 11
boolean?, 11
bound-identifier=?, 14
buffer-mode, 12
buffer-mode?, 12
bytevector->sint-list, 11
bytevector->string, 12
bytevector->u8-list, 11
bytevector->uint-list, 11
bytevector-copy, 11
bytevector-copy!, 11
bytevector-fill!, 11
bytevector-ieee-double-native-ref,
    11
bytevector-ieee-double-native-
    set!, 11
bytevector-ieee-double-ref, 11
bytevector-ieee-single-native-ref,
    11
bytevector-ieee-single-native-
    set!, 11
bytevector-ieee-single-ref, 11
bytevector-length, 11
bytevector-s16-native-ref, 11
bytevector-s16-native-set!, 11
bytevector-s16-ref, 11
bytevector-s16-set!, 11
bytevector-s32-native-ref, 11
bytevector-s32-native-set!, 11
bytevector-s32-ref, 11
bytevector-s32-set!, 11
bytevector-s64-native-ref, 11
bytevector-s64-native-set!, 11
bytevector-s64-ref, 11
bytevector-s64-set!, 11
bytevector-s8-ref, 11
bytevector-s8-set!, 11
bytevector-sint-ref, 11
bytevector-sint-set!, 11
bytevector-u16-native-ref, 11
bytevector-u16-native-set!, 11
bytevector-u16-ref, 11
bytevector-u16-set!, 11
bytevector-u32-native-ref, 11
bytevector-u32-native-set!, 11
bytevector-u32-ref, 11
bytevector-u32-set!, 11
bytevector-u64-native-ref, 11
bytevector-u64-native-set!, 11
bytevector-u64-ref, 11
bytevector-u64-set!, 11

```

bytevector-u8-ref, 11	char<?, 11
bytevector-u8-set!, 11	char=?, 11
bytevector-uint-ref, 11	char>=?, 11
bytevector-uint-set!, 11	char>?, 11
bytevector=? , 11	char?, 11
bytevector?, 11	close-input-port, 13
caa , 11	close-output-port, 13
cadr, 11	close-port, 12
call-with-bytevector-output-port,	command-line, 13
12	complex?, 11
call-with-current-continuation, 11	cond, 11
call-with-input-file, 13	condition, 12
call-with-output-file, 13	condition-accessor, 12
call-with-port, 12	condition-irritants, 12
call-with-string-output-port, 12	condition-message, 12
call-with-values, 11	condition-predicate, 12
call/cc, 11	condition-who, 12
car, 11	condition?, 12
case, 11	cons, 11
case-lambda, 11	cons*, 11
cdddar, 11	cos, 11
cddddr, 11	current-error-port, 12
cdr, 11	current-input-port, 12
ceiling, 11	current-output-port, 12
char->integer, 11	datum->syntax, 14
char-alphabetic?, 11	define, 11
char-ci<=?, 11	define-condition-type, 12
char-ci< ? , 11	define-enumeration, 14
char-ci=? , 11	define-record-type, 12
char-ci>=?, 11	define-syntax, 11
char-ci>?, 11	delay, 15
char-downcase, 11	delete-file, 13
char-foldcase, 11	denominator, 11
char-general-category, 11	display, 13
char-lower-case?, 11	div, 11
char-numeric?, 11	div-and-mod, 11
char-title-case?, 11	div0, 11
char-titlecase, 11	div0-and-mod0, 11
char-upcase, 11	do, 11
char-upper-case?, 11	dynamic-wind, 11
char-whitespace?, 11	else, 11
char< ?, 11	else, 12

endianness, 11	fixnum?, 13
enum-set->list, 14	fl*, 13
enum-set-complement, 14	fl+, 13
enum-set-constructor, 14	fl-, 13
enum-set-difference, 14	fl/, 13
enum-set-indexer, 14	fl<?, 13
enum-set-intersection, 14	fl<?, 13
enum-set-member?, 14	fl=?, 13
enum-set-projection, 14	fl>?, 13
enum-set-subset?, 14	fl>?, 13
enum-set-union, 14	flabs, 13
enum-set-universe, 14	flacos, 13
enum-set=?, 14	flasin, 13
environment, 14	flatan, 13
eof-object, 12	flceiling, 13
eof-object?, 12	flcos, 13
eol-style, 12	fldenominator, 13
eq?, 11	fldiv, 13
equal-hash, 14	fldiv-and-mod, 13
equal?, 11	fldivo, 13
eqv?, 11	fldivo-and-mod0, 13
error, 11	fleven?, 13
error-handling-mode, 12	flexp, 13
error?, 12	flexpt, 13
eval, 14	flfinite?, 13
even?, 11	flfloor, 13
exact, 11	flinfinite?, 13
exact->inexact, 15	flinteger?, 13
exact-integer-sqrt, 11	fllog, 13
exact?, 11	flmax, 13
exists, 11	flmin, 13
exit, 13	flmod, 13
exp, 11	flmodo, 13
expt, 11	flnan?, 13
fields, 12	flnegative?, 13
file-exists?, 13	flnumerator, 13
file-options, 12	flodd?, 13
filter, 11	flonum?, 13
find, 11	floor, 11
finite?, 11	flpositive?, 13
fixnum->flonum, 13	flround, 13
fixnum-width, 13	flsin, 13

<code>fsqrt</code> , 13	<code>fxmod</code> , 13
<code>ftan</code> , 13	<code>fxmod0</code> , 13
<code>truncate</code> , 13	<code>fxnegative?</code> , 13
<code>flush-output-port</code> , 12	<code>fxnot</code> , 13
<code>flzero?</code> , 13	<code>fxodd?</code> , 13
<code>fold-left</code> , 11	<code>fxpositive?</code> , 13
<code>fold-right</code> , 11	<code>fxreverse-bit-field</code> , 13
<code>for-all</code> , 11	<code>fxrotate-bit-field</code> , 13
<code>for-each</code> , 11	<code>fxxor</code> , 13
<code>force</code> , 15	<code>fxzero?</code> , 13
<code>free-identifier=?</code> , 14	<code>gcd</code> , 11
<code>fx*</code> , 13	<code>generate-temporaries</code> , 14
<code>fx*/carry</code> , 13	<code>get-bytevector-all</code> , 12
<code>fx+</code> , 13	<code>get-bytevector-n</code> , 12
<code>fx+/carry</code> , 13	<code>get-bytevector-n!</code> , 12
<code>fx-</code> , 13	<code>get-bytevector-some</code> , 12
<code>fx-/carry</code> , 13	<code>get-char</code> , 12
<code>fx&lt;=?</code> , 13	<code>get-datum</code> , 12
<code>fx&lt;?</code> , 13	<code>get-line</code> , 12
<code>fx=?</code> , 13	<code>get-string-all</code> , 12
<code>fx&gt;=?</code> , 13	<code>get-string-n</code> , 12
<code>fx&gt;?</code> , 13	<code>get-string-n!</code> , 12
<code>fxand</code> , 13	<code>get-u8</code> , 12
<code>fxarithmetic-shift</code> , 13	<code>greatest-fixnum</code> , 13
<code>fxarithmetic-shift-left</code> , 13	<code>guard</code> , 12
<code>fxarithmetic-shift-right</code> , 13	<code>hashtable-clear!</code> , 14
<code>fxbit-count</code> , 13	<code>hashtable-contains?</code> , 14
<code>fxbit-field</code> , 13	<code>hashtable-copy</code> , 14
<code>fxbit-set?</code> , 13	<code>hashtable-delete!</code> , 14
<code>fxcopy-bit</code> , 13	<code>hashtable-entries</code> , 14
<code>fxcopy-bit-field</code> , 13	<code>hashtable-equivalence-function</code> , 14
<code>fxdiv</code> , 13	<code>hashtable-hash-function</code> , 14
<code>fxdiv-and-mod</code> , 13	<code>hashtable-keys</code> , 14
<code>fxdiv0</code> , 13	<code>hashtable-mutable?</code> , 14
<code>fxdiv0-and-mod0</code> , 13	<code>hashtable-ref</code> , 14
<code>fxeven?</code> , 13	<code>hashtable-set!</code> , 14
<code>fxfirst-bit-set</code> , 13	<code>hashtable-size</code> , 14
<code>fxif</code> , 13	<code>hashtable-update!</code> , 14
<code>fxior</code> , 13	<code>hashtable?</code> , 14
<code>fxlength</code> , 13	<code>i/o-decoding-error?</code> , 12
<code>fxmax</code> , 13	<code>i/o-encoding-error-char</code> , 12
<code>fxmin</code> , 13	<code>i/o-encoding-error?</code> , 12

**i/o-error-filename**, 12  
**i/o-error-port**, 12  
**i/o-error-position**, 12  
**i/o-error?**, 12  
**i/o-file-already-exists-error?**, 12  
**i/o-file-does-not-exist-error?**, 12  
**i/o-file-is-read-only-error?**, 12  
**i/o-file-protection-error?**, 12  
**i/o-filename-error?**, 12  
**i/o-invalid-position-error?**, 12  
**i/o-port-error?**, 12  
**i/o-read-error?**, 12  
**i/o-write-error?**, 12  
**identifier-syntax**, 11  
**identifier?**, 14  
**if**, 11  
**imag-part**, 11  
**immutable**, 12  
**implementation-restriction-violation?**, 12  
**inexact**, 11  
**inexact->exact**, 15  
**inexact?**, 11  
**infinite?**, 11  
**input-port?**, 12  
 Installing Libraries, 5  
**integer->char**, 11  
**integer-valued?**, 11  
**integer?**, 11  
**irritants-condition?**, 12  
**lambda**, 11  
**latin-1-codec**, 12  
**lcm**, 11  
**least-fixnum**, 13  
**length**, 11  
**let**, 11  
**let\*, 11**  
**let\*-values**, 11  
**let-syntax**, 11  
**let-values**, 11  
**letrec**, 11  
**letrec\***, 11  
**letrec-syntax**, 11  
**lexical-violation?**, 12  
 Libraries and Collections, 7  
**list**, 11  
**list->string**, 11  
**list->vector**, 11  
**list-ref**, 11  
**list-sort**, 11  
**list-tail**, 11  
**list?**, 11  
**log**, 11  
**lookahead-char**, 12  
**lookahead-u8**, 12  
**magnitude**, 11  
**make-assertion-violation**, 12  
**make-bytevector**, 11  
**make-custom-binary-input-port**, 12  
**make-custom-binary-input/output-port**, 12  
**make-custom-binary-output-port**, 12  
**make-custom-textual-input-port**, 12  
**make-custom-textual-input/output-port**, 12  
**make-custom-textual-output-port**, 12  
**make-enumeration**, 14  
**make-eq-hashtable**, 14  
**make-eqv-hashtable**, 14  
**make-error**, 12  
**make-hashtable**, 14  
**make-i/o-decoding-error**, 12  
**make-i/o-encoding-error**, 12  
**make-i/o-error**, 12  
**make-i/o-file-already-exists-error**, 12  
**make-i/o-file-does-not-exist-error**, 12  
**make-i/o-file-is-read-only-error**, 12  
**make-i/o-file-protection-error**, 12  
**make-i/o-filename-error**, 12  
**make-i/o-invalid-position-error**, 12  
**make-i/o-port-error**, 12

```

make-i/o-read-error, 12
make-i/o-write-error, 12
make-implementation-restriction-
    violation, 12
make-irritants-condition, 12
make-lexical-violation, 12
make-message-condition, 12
make-no-infinities-violation, 13
make-no-nans-violation, 13
make-non-continuable-violation, 12
make-polar, 11
make-record-constructor-
    descriptor, 12
make-record-type-descriptor, 12
make-rectangular, 11
make-serious-condition, 12
make-string, 11
make-syntax-violation, 12
make-transcoder, 12
make-undefined-violation, 12
make-variable-transformer, 14
make-vector, 11
make-violation, 12
make-warning, 12
make-who-condition, 12
map, 11
max, 11
member, 11
memp, 11
memq, 11
memv, 11
message-condition?, 12
min, 11
mod, 11
mod0, 11
modulo, 15
mutable, 12
nan?, 11
native-endianness, 11
native-eol-style, 12
native-transcoder, 12
negative?, 11
newline, 13
no-infinities-violation?, 13
no-nans-violation?, 13
non-continuable-violation?, 12
nongenerative, 12
not, 11
null-environment, 15
null?, 11
number->string, 11
number?, 11
numerator, 11
odd?, 11
opaque, 12
open-bytevector-input-port, 12
open-bytevector-output-port, 12
open-file-input-port, 12
open-file-input/output-port, 12
open-file-output-port, 12
open-input-file, 13
open-output-file, 13
open-string-input-port, 12
open-string-output-port, 12
or, 11
output-port-buffer-mode, 12
output-port?, 12
pair?, 11
parent, 12
parent-rtd, 12
partition, 11
peek-char, 13
port-eof?, 12
port-has-port-position?, 12
port-has-set-port-position!?, 12
port-position, 12
port-transcoder, 12
port?, 12
positive?, 11
procedure?, 11
protocol, 12
put-bytevector, 12
put-char, 12
put-datum, 12

```

```

put-string, 12
put-u8, 12
quasiquote, 11
quasisyntax, 14
quote, 11
quotient, 15
r6rs, 6
R6RS Conformance, 9
R6RS Libraries, 11
R6RS Module Language, 6
R6RS: Standard Language, 1
raise, 12
raise-continuable, 12
rational-valued?, 11
rational?, 11
rationalize, 11
read, 13
read-char, 13
real->flonum, 13
real-part, 11
real-valued?, 11
real?, 11
record-accessor, 12
record-constructor, 12
record-constructor-descriptor, 12
record-field-mutable?, 12
record-mutator, 12
record-predicate, 12
record-rtd, 12
record-type-descriptor, 12
record-type-descriptor?, 12
record-type-field-names, 12
record-type-generative?, 12
record-type-name, 12
record-type-opaque?, 12
record-type-parent, 12
record-type-sealed?, 12
record-type-uid, 12
record?, 12
remainder, 15
remove, 11
remp, 11
remq, 11
remv, 11
reverse, 11
rnrs/arithmetic/bitwise-6, 13
rnrs/arithmetic/fixnums-6, 13
rnrs/arithmetic/flonums-6, 13
rnrs/base-6, 11
rnrs/bytewectors-6, 11
rnrs/conditions-6, 12
rnrs/control-6, 11
rnrs/enums-6, 14
rnrs/eval-6, 14
rnrs/exceptions-6, 12
rnrs/files-6, 13
rnrs/hashtables-6, 14
rnrs/io/ports-6, 12
rnrs/io/simple-6, 13
rnrs/lists-6, 11
rnrs/mutable-pairs-6, 14
rnrs/mutable-strings-6, 14
rnrs/programs-6, 13
rnrs/r5rs-6, 15
rnrs/records/inspection-6, 12
rnrs/records/procedural-6, 12
rnrs/records/syntactic-6, 12
rnrs/sorting-6, 11
rnrs/syntax-case-6, 14
rnrs/unicode-6, 11
round, 11
Running Top-Level Programs, 4
Scheme Interoperability, 8
scheme-report-environment, 15
sealed, 12
serious-condition?, 12
set!, 11
set-car!, 14
set-cdr!, 14
set-port-position!, 12
simple-conditions, 12
sin, 11
sint-list->bytewector, 11
sqrt, 11

```

standard-error-port	, 12
standard-input-port	, 12
standard-output-port	, 12
string	, 11
string->bytevector	, 12
string->list	, 11
string->number	, 11
string->symbol	, 11
string->utf16	, 11
string->utf32	, 11
string->utf8	, 11
string-append	, 11
string-ci-hash	, 14
string-ci<=?	, 11
string-ci<?	, 11
string-ci=?	, 11
string-ci>=?	, 11
string-ci>?	, 11
string-copy	, 11
string-downcase	, 11
string-fill!	, 14
string-foldcase	, 11
string-for-each	, 11
string-hash	, 14
string-length	, 11
string-normalize-nfc	, 11
string-normalize-nfd	, 11
string-normalize-nfkc	, 11
string-normalize-nfkd	, 11
string-ref	, 11
string-set!	, 14
string-titlecase	, 11
string-upcase	, 11
string<=?	, 11
string<?	, 11
string=?	, 11
string>=?	, 11
string>?	, 11
string?	, 11
substring	, 11
symbol->string	, 11
symbol-hash	, 14
symbol=?	, 11
symbol?	, 11
syntax	, 14
syntax->datum	, 14
syntax-case	, 14
syntax-rules	, 11
syntax-violation	, 14
syntax-violation-form	, 12
syntax-violation-subform	, 12
syntax-violation?	, 12
tan	, 11
textual-port?	, 12
transcoded-port	, 12
transcoder-codec	, 12
transcoder-eol-style	, 12
transcoder-error-handling-mode	, 12
truncate	, 11
u8-list->bytevector	, 11
uint-list->bytevector	, 11
undefined-violation?	, 12
unless	, 11
unquote	, 11
unquote-splicing	, 11
unsyntax	, 14
unsyntax-splicing	, 14
utf-16-codec	, 12
utf-8-codec	, 12
utf16->string	, 11
utf32->string	, 11
utf8->string	, 11
values	, 11
vector	, 11
vector->list	, 11
vector-fill!	, 11
vector-for-each	, 11
vector-length	, 11
vector-map	, 11
vector-ref	, 11
vector-set!	, 11
vector-sort	, 11
vector-sort!	, 11
vector?	, 11

```
violation?, 12
warning?, 12
when, 11
who-condition?, 12
with-exception-handler, 12
with-input-from-file, 13
with-output-to-file, 13
with-syntax, 14
write, 13
write-char, 13
zero?, 11
```