

# R5RS: Legacy Scheme

Version 9.2

May 27, 2026

The The Revised<sup>5</sup> Report on the Algorithmic Language Scheme defines a dialect of Scheme. We use *R<sup>5</sup>RS* to refer to both the standard and the language defined by the standard.

The default dialect of Lisp provided by `racket` and other Racket tools differs from R<sup>5</sup>RS in many ways, but Racket includes tools and libraries for running R<sup>5</sup>RS programs.

See §23 “Dialects of Racket and Scheme” for general information about different dialects of Scheme within Racket.

# Contents

<b>1</b>	<b>Running R<sup>5</sup>RS Programs</b>	<b>3</b>
<b>2</b>	<code>plt-r5rs</code>	<b>4</b>
<b>3</b>	<b>R<sup>5</sup>RS Module Language</b>	<b>5</b>
3.1	Non-R <sup>5</sup> RS Bindings from <code>r5rs</code> . . . . .	5
3.2	Notes on R <sup>5</sup> RS Functions . . . . .	6
<b>4</b>	<b>R<sup>5</sup>RS Initialization Library</b>	<b>7</b>

# 1 Running R<sup>5</sup>RS Programs

Racket provides several layers of support for programs written according to R<sup>5</sup>RS:

- DrRacket provides an R<sup>5</sup>RS language, which can be selected via the Language|Choose Language... menu item. See Choose Language... in the DrRacket documentation for more information.
- The `plt-r5rs` executable runs an R<sup>5</sup>RS program or provides a read-eval-print loop for evaluating R<sup>5</sup>RS expressions and definitions. See §2 “`plt-r5rs`” (later in this manual) for more information.
- The `r5rs` library implemented R<sup>5</sup>RS procedures and syntactic forms. It can also be used with `#lang` to create a module whose body is implemented in an R<sup>5</sup>RS-like language. See §3 “R<sup>5</sup>RS Module Language” (later in this manual) for more information.
- The `r5rs/init` library extends `r5rs` to set parameters (such as case-insensitive symbol reading) for R<sup>5</sup>RS loading or an R<sup>5</sup>RS read-eval-print loop. See §4 “R<sup>5</sup>RS Initialization Library” (later in this manual) for more information.

## 2 plt-r5rs

The `plt-r5rs` executable runs an R<sup>5</sup>RS program from a file that is supplied on the command line. If no program file is provided as a command-line argument, then a read-eval-print loop is started.

Before starting a read-eval-print loop, an initialization file is loaded, if it exists. The file is the same as the file reported by `(find-system-path 'init-file)`, but with the characters `racket` in the filename replaced by `pltr5rs`. For example, on Unix, the file is `"~/pltr5rsrc"`.

By default, `plt-r5rs` departs from R<sup>5</sup>RS conformance in one crucial way: the names of pre-defined functions cannot be redefined at the top level. This restriction enables better run-time performance. Use the `--no-prim` command-line flag—before a file to load, if any—to obtain the standard behavior for primitive bindings (at the cost of performance).

### 3 R<sup>5</sup>RS Module Language

```
#lang r5rs      package: r5rs-lib
```

As a library, `r5rs` provides the syntactic forms and procedures defined by R<sup>5</sup>RS. When used as a language via `#lang`, the program is read with the following parameterizations:

```
(read-case-sensitive #f)
(read-accept-infix-dot #f)
(read-curly-brace-as-paren #f)
(read-square-bracket-as-paren #f)
```

The `r5rs` bindings can be imported into a top-level environment, and then evaluation in that top-level environment corresponds to R<sup>5</sup>RS. Use `(namespace-require/copy 'r5rs)` with an empty namespace to maximize conformance with R<sup>5</sup>RS; Using `(namespace-require 'r5rs)`, in contrast, creates primitive bindings as imports, which is the same as using `plt-r5rs` without the `--no-prim` flag. More simply, use `(scheme-report-environment 5)`. See also `r5rs/init`, which sets reader and printer parameters to increase conformance.

Using `r5rs` via `#lang` creates a module whose body is implemented with an R<sup>5</sup>RS-like language. The main difference from R<sup>5</sup>RS is that, as a module language, `r5rs` does not allow redefinition of top-level bindings, and expressions evaluated through `load` and `eval` cannot automatically access bindings defined within the module.

Changed in version 6.0.1.4 of package `r5rs-lib`: When an identifier bound by `letrec` is referenced before it is initialized, an exception is raised, instead of producing `#<undefined>`.

#### 3.1 Non-R<sup>5</sup>RS Bindings from `r5rs`

In addition to the bindings defined by R<sup>5</sup>RS, the `r5rs` library provides the following bindings from `racket/base` (which are not legal identifiers in R<sup>5</sup>RS syntax, so there is no danger of collisions in R<sup>5</sup>RS programs):

```
#!/app #/datum #/top #/top-interaction #/require #/provide
```

It also provides a `#!/module-begin` binding as defined below.

Note that `#!/require` can be used to import Racket libraries into an otherwise R<sup>5</sup>RS program, and `#!/provide` can be used to export from a module that is implemented in an R<sup>5</sup>RS-like language.

Changed in version 1.1 of package `r5rs-lib`: Added an R<sup>5</sup>RS-specific `#!/module-begin`, instead of reexporting `racket's #/plain-module-begin`.

| (`#:module-begin form ...`)

Besides allowing definitions and other forms like `racket`'s `#:plain-module-begin`, defines a `configure-runtime` submodule (see §18.1.5 “Language Run-Time Configuration”) that runs `r5rs/init`.

## 3.2 Notes on R<sup>5</sup>RS Functions

The `cons` of `r5rs` corresponds to `racket/base`'s `mcons`. Similarly, `cdr` is `mcdr`, and `map` is `compatibility/mlist`'s `mmap`, and so on.

An R<sup>5</sup>RS *environment* is implemented as a `racket/base namespace`. Also, relative to `racket/base`, the `expr` passed to `eval` is constructed using mutable pairs.

The `scheme-report-environment` function returns a namespace containing the bindings of `r5rs`. Procedure values are installed into the namespace using `namespace-require/copy`, so that they can be redefined.

The `null-environment` function returns a namespace containing the syntactic forms of `r5rs`, not including `#:module-begin` (which is not useful outside of a module).

## 4 R<sup>5</sup>RS Initialization Library

```
(require r5rs/init)      package: r5rs-lib
```

The `r5rs/init` module re-exports `r5rs`, and also sets parameters as follows:

```
(read-case-sensitive #f)
(read-accept-infix-dot #f)
(read-curly-brace-as-paren #f)
(read-square-bracket-as-paren #f)
(print-mpair-curly-braces #f)
```

The side-effect of setting these parameters is useful when the module is `required` before loading an R<sup>5</sup>RS program, so that the reader and printer behave more as specified in R<sup>5</sup>RS. In particular, the `plt-r5rs` executable initializes by importing `r5rs/init`.