SRFIs: Libraries

Version 5.93

January 29, 2014

The Scheme Requests for Implementation (a.k.a. *SRFI*) process allows individual members of the Scheme community to propose libraries and extensions to be supported by multiple Scheme implementations.

Racket is distributed with implementations of many SRFIs, most of which can be implemented as libraries. To import the bindings of SRFI *n*, use

```
(require srfi/n)
```

This document lists the SRFIs that are supported by Racket and provides a link to the original SRFI specification (which is also distributed as part of Racket's documentation).

Contents

Index **54**

SRFI 1: List Library

```
(require srfi/1)
```
package: `srfi-lite-lib`

Original specification: SRFI 1

This SRFI works with pairs and lists as in `racket`, which are immutable, so it does not export `set-car!` and `set-cdr!`. The other provided bindings that end in `!` are equivalent to the corresponding bindings without `!`. Functions that are documented in the SRFI in bold (but not bold italic) correspond to `racket` functions, while the others are distinct from same-named `racket` functions.

SRFI 2: AND-LET*: an AND with local bindings...

```
(require srfi/2)
```
package: `srfi-lib`

Original specification: SRFI 2

SRFI 4: Homogeneous numeric vector datatypes

`(require srfi/4)` package: `srfi-lib`

Original specification: SRFI 4

This SRFI's reader and printer syntax is not supported. The bindings are also available from `scheme/foreign`.

SRFI 5: A compatible let form with signatures and rest arguments

```
(require srfi/5)
```
package: `srfi-lib`

Original specification: SRFI 5

SRFI 6: Basic String Ports

```
(require srfi/6)
```
package: srfi-lib

Original specification: SRFI 6

This SRFI's bindings are also available in `racket/base`.

SRFI 7: Feature-based program configuration language

`(require srfi/7)` package: `srfi-lib`

Original specification: SRFI 7

SRFI 8: RECEIVE: Binding to multiple values

```
(require srfi/8)
```
package: `srfi-lite-lib`

Original specification: SRFI 8

SRFI 9: Defining Record Types

`(require srfi/9)` package: `srfi-lib`

Original specification: SRFI 9

SRFI 11: Syntax for receiving multiple values

```
(require srfi/11)
```
package: srfi-lib

Original specification: SRFI 11

This SRFI's bindings are also available in `racket/base`, but without support for dotted "rest" bindings.

SRFI 13: String Libraries

`(require srfi/13)` package: `srfi-lite-lib`

Original specification: SRFI 13

SRFI 14: Character-set Library

```
(require srfi/14)
```
package: `srfi-lite-lib`

Original specification: SRFI 14

SRFI 16: Syntax for procedures of variable arity

```
(require srfi/16)
```
package: srfi-lib

Original specification: SRFI 16

This SRFI's bindings are also available in racket/base.

SRFI 17: Generalized set!

```
(require srfi/17)
```
package: `srfi-lib`

Original specification: SRFI 17

SRFI 19: Time Data Types and Procedures

```
(require srfi/19)
```
package: srfi-lite-lib

Original specification: SRFI 19

The date structure produced by this SRFI library is identical to the one provided by `racket/base` in most cases (see `date`).

For backwards compatibility, when an invalid date field value is provided to the SRFI constructor, the constructor will produce a lax date structure. A lax date structure is *not* compatible with functions from `racket/base` or `racket/date`. SRFI functions such as `string->date` may return a lax date structure depending on the format string.

```
(lax-date? v) → boolean?
  v : any/c
```

Returns `#t` if *v* is a lax date structure. Otherwise returns `#f`.

Examples:

```
> (lax-date? (make-date 0 19 10 10 14 "bogus" "bogus" 0))
#t
> (lax-date? (make-date 0 19 10 10 14 1 2013 0))
#f
> (lax-date? (string->date "10:21:00" "~H:~M:~S"))
#t
```

SRFI 23: Error reporting mechanism

```
(require srfi/23)
```
package: srfi-lib

Original specification: SRFI 23

This SRFI's bindings are also available in racket/base.

SRFI 25: Multi-dimensional Array Primitives

`(require srfi/25)` package: `srfi-lib`

Original specification: SRFI 25

SRFI 26: Notation for Specializing Parameters without Currying

`(require srfi/26)` package: `srfi-lib`

Original specification: SRFI 26

SRFI 27: Sources of Random Bits

```
(require srfi/27)
```
package: `srfi-lib`

Original specification: SRFI 27

SRFI 28: Basic Format Strings

```
(require srfi/28)
```
package: srfi-lib

Original specification: SRFI 28

This SRFI's bindings are also available in racket/base.

SRFI 29: Localization

`(require srfi/29)` package: `srfi-lite-lib`

Original specification: SRFI 29

SRFI 30: Nested Multi-line Comments

```
(require srfi/30)
```
package: srfi-lib

Original specification: SRFI 30

This SRFI's syntax is part of Racket's default reader.

SRFI 31: A special form rec for recursive evaluation

```
(require srfi/31)
```
package: srfi-lib

Original specification: SRFI 31

SRFI 34: Exception Handling for Programs

`(require srfi/34)` package: `srfi-lib`

Original specification: SRFI 34

SRFI 35: Conditions

```
(require srfi/35)
```
package: `srfi-lib`

Original specification: SRFI 35

SRFI 38: External Representation for Data With Shared Structure

```
(require srfi/38)
```
package: srfi-lib

Original specification: SRFI 38

This SRFI's syntax is part of Racket's default reader and printer.

SRFI 39: Parameter objects

```
(require srfi/39)
```
package: srfi-lib

Original specification: SRFI 39

This SRFI's bindings are also available in racket/base.

SRFI 40: A Library of Streams

`(require srfi/40)` package: `srfi-lib`

Original specification: SRFI 40

Superceded by `srfi/41`.

SRFI 41: Streams

```
(require srfi/41)
```
package: srfi-lib

Original specification: SRFI 41

The `stream-cons` operation from `srfi/41` is the same as from `racket/stream`.

SRFI 42: Eager Comprehensions

(require srfi/42) package: srfi-lib

Original specification: SRFI 42

Forms that syntactically detect `if` recognize both `if` from `scheme/base` and `if` from `mzscheme`.

SRFI 43: Vector Library

```
(require srfi/43)
```
package: `srfi-lib`

Original specification: SRFI 43

SRFI 45: Primitives for Expressing Iterative Lazy Algorithms

```
(require srfi/45)
```
package: srfi-lib

Original specification: SRFI 45

Additional binding:

```
(promise? v) → boolean?
  v : any/c
```

Returns `#t` if `v` is a promise, `#f` otherwise.

SRFI 48: Intermediate Format Strings

```
(require srfi/48)
```
package: srfi-lib

Original specification: SRFI 48

SRFI 54: Formatting

```
(require srfi/54)
```
package: `srfi-lib`

Original specification: SRFI 54

SRFI 57: Records

```
(require srfi/57)
```
package: `srfi-lib`

Original specification: SRFI 57

SRFI 59: Vicinity

```
(require srfi/59)
```
package: `srfi-lib`

Original specification: SRFI 59

SRFI 60: Integers as Bits

```
(require srfi/60)
```
package: `srfi-lib`

Original specification: SRFI 60

SRFI 61: A more general cond clause

```
(require srfi/61)
```
package: `srfi-lib`

Original specification: SRFI 61

SRFI 62: S-expression comments

Original specification: SRFI 62

This SRFI's syntax is part of Racket's default reader (no `require` is needed).

SRFI 63: Homogeneous and Heterogeneous Arrays

```
(require srfi/63)
```
package: `srfi-lib`

Original specification: SRFI 63

SRFI 64: A Scheme API for test suites

`(require srfi/64)` package: `srfi-lib`

Original specification: SRFI 64

SRFI 66: Octet Vectors

```
(require srfi/66)
```
package: `srfi-lib`

Original specification: SRFI 66

SRFI 67: Compare Procedures

`(require srfi/67)` package: `srfi-lib`

Original specification: SRFI 67

SRFI 69: Basic hash tables

```
(require srfi/69)
```
package: `srfi-lib`

Original specification: SRFI 69

SRFI 71: Extended LET-syntax for multiple values

```
(require srfi/71)
```
package: srfi-lib

Original specification: SRFI 71

SRFI 74: Octet-Addressed Binary Blocks

```
(require srfi/74)
```
package: `srfi-lib`

Original specification: SRFI 74

SRFI 78: Lightweight testing

```
(require srfi/78)
```
package: `srfi-lib`

Original specification: SRFI 78

SRFI 86: MU & NU simulating VALUES & CALL-WITH-VALUES...

`(require srfi/86)` package: `srfi-lib`

Original specification: SRFI 86

SRFI 87: => in case clauses

```
(require srfi/87)
```
package: srfi-lib

Original specification: SRFI 87

SRFI 98: An interface to access environment variables

```
(require srfi/98)
```
package: `srfi-lib`

Original specification: SRFI 98

Index