***Core Library*** ***Referencee***

***core* name space, version *0.0.2***

***type identifiers s***

%lambda *lambda*

%package *package*

%exception *exception*

%vector *vector*

%closure *closure*

:null (),:nil

:char *char*

:cons *cons*

:fixnum *fixnum, fix* 56 bit signed integer

:float *float*, *fl* 32 bit IEEE float

:func *function*, *fn* function

:keyword *keyword, key* symbol

:ns *namespace, ns* namespace

:stream *stream* file or string type

:struct*struct*typed vector

:symbol *symbol, sym* LISP-1symbol

:vector *vector*, *string, str*

:char :t :byte :fixnum :float

***Heap p***

**heap-info** *vector* heap information

#(:t *type* *pages* *pagesize*)

**heap-stat** *vector* heap allocations

#(:t :*type* *size* *total* *free ...*)

**heap-size** *T**fixnum* heap occupancy

***Frame e***

%**frame-stack** *list*active *frame*s

**%frame-pop** *fn fn* pop *function’s* top

frame binding

*frame* binding:(*fn* . #(:t *…*))

**%frame-push** *frame cons* push frame

**%frame-ref** *fn fix T function*, offset

***Symbol l***

**boundp** s*ymbol bool* is*symbol* bound?

**make-symbol** *string symbol* uninterned *symbol*

**makunbound** *string symbol* unbound *symbol*

**symbol-namespace** *symbol*

*key namespace*

**symbol-name** *symbol string* name binding

**symbol-value** *symbol T* value binding

***Special*** ***Forms s***

**:lambda** *list* . *List’* *function* anonymous *function*

***:*quote** *form* *list* quoted form

***:*if** *form* *T T’**T* conditional

***Core s***

**apply** *fn* *list* *T*apply *function* to *list*

**eval** *form* *T* evaluate *form*

**eq** *T*  *T’ bool**T* and *T’* identical?

**type-of** *T* *key* type keyword

**compile** *form* *T**mu* form compiler

**view** *form* *vector* vector of object

**internal-run-time** *fixnum* elapsed time *u*sec

**%if** *T T’ T’’* *key* **:if** implementation

**repr**  *type* *T* *T* tag representation

*type* :t :vector

if type is :vector, return 8 byte

byte vector of argument tag bits,

otherwise convert argument byte vector to tag.

**fix** *fn T*  *T* fixpoint of *function*

**gc** *bool* garbage collection

\***version\*** *string*versionstring

***Future s***

**defer** *fnlist* *struct* future application

**detach** *fnlist* *struct* future application

**force** *struct* *T* force completion

**poll** *struct* *bool* poll completion

***Fixnum m***

**1+** *fixnum* *fix* increment

**1-** *fixnum* *fix* decrement

**ceiling** *fix fix’ fix* ceiling

**floor** *fix fix’ fix* floor

**mod** *fix fix’ fix* mod

**rem** *fix fix’ fix* remainder

**round** *fix fix’ fix* round

**truncate** *fix fix’ fix* truncate

***Float t***

**fproduct** *fl* *fl’* *float* product

**fsum** *fl* *fl’* *float* sum

**fdifference** *fl* *fl’* *float* difference

**fless-than** *fl* *fl’* *bool fl* < *fl’?*

**fquotient** *fl* *fl’* *float* quotient

***Conses/Lists s***

**append** *list T* *list* append

**car** *list* *list* head of *list*

**cdr** *list* *T* tail of *list*

**cons** *T* *T’* *cons* (*form* . *form*’)

**length** *list* *fixnum* length of *list*

**nth** *fix* *list* *T n*th *car* of *list*

**nthcdr** *fix* *list* *T n*th *cdr* of *list*

***Vector s***

**make-vector** *key* *list* *vector* specialized vector

from list

**vector-size** *vector* *fixnum* length of *vector*

**vector-type** *vector* *key* type of *vector*

**svref** *vector* *fix* *T* *n*th element

***Reader/Printer s***

**read** *stream* *bool* *T* *T* read stream object

**write** *T* *bool stream T* write escaped object

***Struct t***

**make-struct** *key list struct* of type *key* from *list*

**struct-type** *struct* *key struct* type *keyword*

**struct-vec** *struct* *vector* of *struct* members

***Exception n***

**with-exception** *fn fn’* *T* catchexception

*fn*  - (:lambda (*obj**cond src*) ***.*** *body*)

*fn’* - (:lambda () **.** *body*)

**raise** *T* *keyword* raise exception

with condition:

:arity :eof :open :read

:syscall :write :error :syntax

:type :sigint :div0 :stream

:range :except :future :ns

:over :under :unbound :return

***Streams n***

**\*****standard*-*input\*** *stream* std input *stream*

**\*standard*-*output\*** *stream* std output *stream*

**\*error*-*output\*** *stream* std error *stream*

**open** *type* *dir string stream* open *stream*

*type* :file :string

*dir* :input :output :bidir

**close** *stream* *bool* close *stream*

**openp** *stream* *bool* is *stream* open?

**flush** s*tream* *bool* flush output *steam*

**get-string** *stream string* from *string* *stream*

**read-byte** *stream bool T*

*byte* read *byte* from

*stream,* error on

eof, *T:* eof value

**read-char** *stream bool T*

*char* read *char* from *stream,* error on

eof, *T:* eof value

**unread-char** *char* *stream*

*char* push *char* onto

*stream*

**write-byte** *byte* *stream* *byte* write *byte* to *stream*

**write*-*char** *char* *stream char* write *char* to *stream*

***Namespace .***

**make-namespace** *str* *ns* make *namespace*

**namespace-map** *list* list of mapped *namespaces*

**namespace-name** *ns* *string namespace* name

**unintern** *ns str symbol un*intern symbol

**intern** *ns str value symbol* intern bound symbol

**find-namespace** *str* *ns* map *string* to *namespace*

**find** *ns* *string* *symbol* map *string* to *symbol*

**namespace-symbols** *ns list namespace symbols*

***Features***  *I*

**[dependencies]**

**default = [ “nix”, "std", "sysinfo" ]**

**nix** uname

**std** command, exit

**sysinfo** sysinfo (disabled on macOS)

**ffi** Rust FFI

***Reader Syntax x***

; comment to end of line

#|...|# block comment

‘*form* quoted form

`*form* backquoted form

`(*...)* backquoted list (proper lists)

,*form* eval backquoted form

,@*form* eval-splice backquoted form

(…) constant *list*

() empty *list*, prints as :nil

(… . .) dotted *list*

“…” *string, char vector*

*\* single escape in strings

#\*... bit vector

#x... hexadecimal *fixnum*

#. read-time eval

#\. *char*

#(:type …) *vector*

#s(:type …) *struct*

#:symbol uninterned *symbol*

“`,; terminating macro char

# non-terminating macro char

!$%&\*+-. symbol constituents

<>=?@[]|

:^\_{}~/

A..Za..z

0..9

0x09 #\tab whitespace

0x0a #\linefeed

0x0c #\page

0x0d #\return

0x20 #\space