***Core* Namespace**

for core version*0.0.2*

***Sequences***

**findl-if** *fn**sequence* sequence search left

**findr-if** *fn**sequence* sequence search right

**foldl** fn *T* *sequence* sequence fold left

**foldr** fn *T* *sequence* sequence fold right

**length** *sequence* length of sequence

**sv-list** *sequence* coerce vector to list

**positionl** *T**sequence* sequence position left

**positionr** *T**sequence* sequence position right

***Exceptions***

**::break** *T* *exception* break loop commands

**assert** *fn* *T* *string* raise exception or return T

**break** *exception* enter break loop

**check-type** *type* *T* *string*

raise exception or return T

**error** *T* *string* raise exception

**print-except** *exception* *stream-designator escape*

print exception

**warn** *T* *string* print warning, return T

***Macros***

**::macroexpand-1** *T* expand macro form once

**macro-function** *fn* macro expander function

**macroexpand**expand macro completely

***Special Forms***

**defconst** *symbol* *T* define constant symbol

**defmacro** *symbol* *list* . *body*

define macro expander

**defun** symbol *list* . *body*

define function

**if** *T T' [T'']* conditional,

optional third argument

**capture-env** *T*capture lexical env

arg ignored

**lambda** *list* . *body* lambda definition

***Streams***

**::stream-designator** *T*

map designator to stream

**load** *string* *bool* *bool* load file *verbose* *print*

***Lists***

**assoc** *T* *list* association list lookup

**copy** *list* copy list

**reverse** *list* reverse list

**dropl** *list* *fixnum* drop from left

**dropr** *list* *fixnum* drop from right

**mapc** *fn* *list* apply fn to list cars

**mapcar** *fn* *list* new list from list cars

**mapl** *fn* *list* apply fn to list cdrs

**maplist** *fn* *list* list from list cdrs

***Core***

**version** version string

**eval** *T* evaluate form

**funcall** *fn* *list* apply list to fn

**identity** *T* return T

**1+** *fixnum fixnum +* 1

**1-** *fixnum fixnum* - 1

***Compiler***

compile *T* compile form

***Reader***

read stream-designator eof

read form from stream

***Predicates***

**consp** *T* cons type

**charp** *T* char type

**exceptionp** *T* exception type

**fixnump** *T* fixnum type

**floatp** *T* float type

**functionp** *T* function type

**listp** *T* cons or nil

**namespacep** *T* namespace type

**null** *T* is nil

**sequencep** *T* list or vector

**streamp** *T* stream type

**stringp** *T* string type

**symbolp** *T* symbol type

**vectorp** *T* vector type

**zerop** *T* zero fixnum

***Strings***

**schar** *vector fixnum* char from string at index

**string** *char | symbol* convert char or symbol

name to string

**string-append** s*tring string*'

append string and string'

**string-length** *string* length of string

**string=** *string* *string*' strings *eql*

**substr** *string* *fx* *fx*’ *start, end* substring