***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***

**::append** *list* *list'*  append two lists

**::list** *T* *T’* list from two objects

**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

***Functions***

::clone fn ... clone function

::closure fn create closure

::fn-arity fn function accessor

::fn-call fn list call function

::fn-call-closure fn list call function closure

::fn-form fn function accessor

::fn-frame-id fn function accessor

::fn-lambda fn function accessor

::fn-macrop fn function accessor

::fn-name fn function accessor

::fn-nreqs fn function accessor

::fn-restp fn function accessor

::fn-unclosedp function accessor

::frame-descriptor frame descriptor

::frame-fn frame descriptor accessor

::frame-id frame descriptor fn id

::frame-symbols frame symbol bindings

::lambda-closure vector lambda descriptor accessor

::lambda-descriptor vector lambda descriptor accessor

::lambda-env vector lambda descriptor accessor

::lambda-macrop vector lambda descriptor accessor

::lambda-nreqs vector lambda descriptor accessor

::lambda-reqs vector lambda descriptor accessor

::lambda-rest vector lambda descriptor accessor

::lambda-syms vector lambda descriptor accessor

closure create closure

***Compiler***

::compile-activation list list'

compile activation form in environment

::compile-add-env fn list add function to environment

::compile-closure fn list compile closure in environment

::compile-lambda list list' compile lambda form in environment

::compile-lambda-body list list'

compile lambda body in environment

::compile-macro form list compile macro in environment

::compile-symbol form list compile symbol in environment

::core-lambda form list compile lambda definition

::core-macro form list compile macro definition

::must-funcall list core:funcall this activation?

::symbol-frame symbol list resolve symbol in environment

compile T compile form

***Quasiquote***

::quasi-comma , syntax

::quasi-comma-at ,@ syntax

::quasi-list `list syntax

::quasiquote `form syntax

***Reader***

::read reader implementation

::read-atom reader implementation

::read-char-syntax reader implementation

::read-comment reader implementation

::read-consume-ws reader implementation

::read-dispatch reader implementation

::read-dispatch-table reader implementation

::read-fixnum reader implementation

::read-fixnump reader implementation

::read-list reader implementation

::read-list-eol reader implementation

::read-macro reader implementation

::read-macro-table reader implementation

::read-namespaces reader implementation

::read-quote reader implementation

::read-resolve-symbol reader implementation

::read-sharp reader implementation

::read-sharp-char reader implementation

::read-sharp-comment reader implementation

::read-sharp-symbol reader implementation

::read-sharp-table reader implementation

::read-sharp-vector reader implementation

::read-string reader implementation

::read-symbol-externp reader implementation

::read-symbol-keywordp reader implementation

::read-symbol-name reader implementation

::read-symbol-ns reader implementation

::read-table reader implementation

::reader-stream reader implementation

read stream-designator eof read form from stream

***Predicates***

**consp** *T* cons type

**charp** *T* char type

**doublep** *T* double float type

**exceptionp** T exception type

**fixnump** T fixnum type

**floatp** T single 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 start end substring