

Mu Namespace

dyad mu version 0.0.8

Type keywords

T	type superclass
:t :nil	boolean
:char	char
:cons	cons
:fixnum	61 bit signed integer
:float	32 bit IEEE float
:func	function
:ns	symbol bindings
:stream	file, string, socket
:symbol	LISP-1 binding
:vector	:t :byte :char :fixnum :float

Heap

hp-info	heap values alist
hp-type	type of
	type occupancy:
	type: type keyword
	of: :alloc
	:in-use
	:free
	:size

Frame

:fr-ref	<i>fix fix'</i> ref frame variable
*fr-pop	func pop frame binding
*fr-push	list push frame binding

Functions

fn-prop	<i>prop fn</i> function property
	:lambda :frame :form

Symbols

boundp	<i>symbol</i> symbol bound?
keyp	<i>symbol</i> keyword predicate
keyword	<i>string</i> keyword from string
symbol	<i>string</i> uninterned symbol
sy-ns	<i>symbol</i> symbol ns binding
sy-name	<i>symbol</i> symbol name binding
sy-val	<i>symbol</i> symbol value binding

Special Forms

:lambda	<i>list . body</i> anonymous function
:quote	<i>T</i> quote form
:if	<i>T T' T''</i> conditional

Core

coerce	<i>T :type</i> coerce to type keyword
eval	<i>T</i> evaluate form
eq	<i>T T'</i> are <i>T</i> and <i>T'</i> identical?
type-of	<i>T</i> type keyword
apply	<i>fn list</i> apply function to arg list
compile	<i>T</i> library form compiler
*:context	active frame list
with-ex	<i>fn fn'</i> catch exception
raise	<i>keyword T</i> raise exception
tag-of	<i>T</i> object tag to <i>fixnum</i>
*gc	garbage collection
view	<i>T</i> view vector of object
fix	<i>fn T</i> fixpoint function
:if	<i>T fn fn'</i> :if implementation

Reader/Printer

read	<i>stream bool T</i> read object from stream
write	<i>T bool stream</i> print with escapes

Fixnums

fx-mul	<i>fix fix'</i> product of <i>fix</i> and <i>fix'</i>
fx-add	<i>fix fix'</i> sum of <i>fix</i> and <i>fix'</i>
fx-sub	<i>fix fix'</i> difference of <i>fix</i> and <i>fix'</i>
fx-lt	<i>fix fix'</i> is <i>fix</i> less than <i>fix'</i> ?
fx-div	<i>fix fix</i> <i>fix</i> divided by <i>fix'</i>
logand	<i>fix fix'</i> bitwise and of <i>fix</i> and <i>fix'</i>
logor	<i>fix fix'</i> bitwise or <i>fix</i> and <i>fix'</i>

Floats

fl-mul	<i>float float'</i> product of <i>float</i> and <i>float'</i>
fl-add	<i>float float'</i> sum of <i>float</i> and <i>float'</i>
fl-sub	<i>float float'</i> difference of <i>float</i> and <i>float'</i>
fl-lt	<i>float float'</i> is <i>float</i> less than <i>float'</i> ?
fl-div	<i>float float'</i> <i>float</i> divided by <i>float'</i>

Lists

car	<i>list</i> head of <i>list</i>
cdr	<i>list</i> tail of <i>list</i>
cons	<i>T T'</i> cons from <i>T</i> and <i>T'</i>
length	<i>list</i> length of <i>list</i>
nth	<i>fix list</i> nth car of <i>list</i>
nthcdr	<i>fix list</i> nth cdr of <i>list</i>

Vectors

vector	<i>type list</i> specialized vector from list
sv-len	<i>vector</i> <i>fixnum</i> length of <i>vector</i>
sv-ref	<i>vector fix</i> nth element
sv-type	<i>vector</i> type of <i>vector</i> elements

Condition Keywords

:arity	:eof
:open	:read
:write	:error
:syntax	:type
:unbound	:div0
:range	:stream

Streams

std-in standard input *stream symbol*
std-out standard output *stream symbol*
err-out standard error *stream symbol*

open *type dir string*
open *stream* from
type :file | :string
dir :input | :output

close stream close *stream*
openp stream is *stream* open?
eof stream is *stream* at end of file?

get-str stream
get *vector* from *stream*

rd-byte stream bool T
read *byte* from *stream*

un-byte byte stream push *byte* onto *stream*
wr-byte byte stream write *byte* to *stream*

rd-char stream bool T
read *char* from *stream*
un-char char stream push *char* onto *stream*
wr-char char stream write *char* to *stream*

Namespaces

make-ns string ns
make *namespace*
map-ns string map *string* to *namespace*

intern ns scope string value
intern bound symbol
scope :intern :extern

ns-map ns string
map *string* to *symbol*

ns-imp ns *namespace's* import
ns-name ns *namespace's* name
ns-int ns *namespace's* interns
ns-ext ns *namespace's* externs

Rust API

```
use crate::mu::core::mu::{
    Exception,
    Extern,
    Mu,
    MuCondition,
    Tag
},

<Mu as Extern>::new(config: String) -> Mu
    config: comma-separated
    list of name:value pairs:

    heap: npages
    gc: on|off

&'static str <Mu as Extern>::VERSION

pub trait Export for Mu {
    fn nil() -> Tag

    fn eq(tag: Tag, tag1: Tag) -> bool

    fn apply(&self, func: Tag, args) ->
        Exception::Result<Tag>

    fn compile(&self, expr: Tag) ->
        Exception::Result<Tag>

    fn eof(&self, stream: Tag) ->
        Exception::Result<Tag>

    fn eval(&self, expr: Tag) ->
        Exception::Result<Tag>

    fn read_stream(&self, stream: Tag,
        eof: Tag,
        eof_value: Tag) ->
        Exception::Result<Tag>

    fn read_string(&self, expr: String) ->
        Exception::Result<Tag>

    fn write(&self, expr: Tag,
        escape: bool,
        stream: Tag) ->
        Exception::Result<()>

    fn write_string(&self, string: String,
        stream: Tag) ->
        Exception::Result<()>
}
```

Reader Syntax

;
| . . . | # comment to end of line
block comment

(...)
() constant list
empty list, prints as :nil

`
"..." quoted form
string/char vector

*#x hexadecimal *fixnum*
#\ character

*#(:vector-type ...) vector
#:symbol uninterned *symbol*

\ single escape in strings

"` , ;
terminating macro char
non-terminating macro char

!\$%&*+- . symbol constituent:
<>=?@[|
: ^ _ { } ~ /
A . . Z a . . z
0 . . 9
backspace
rubout

0x09 tab whitespace:
0x0a linefeed
0x0c page
0x0d return
0x20 space

runtime

```
runtime: 0.0.8: [-h?psvcelq] [file...]
?: usage message
h: usage message
c: [name:value,...]
e: eval [form] and print result
l: load [path]
p: pipe mode
q: eval [form] quietly
s: script mode
v: print version and exit
```