Micro quick syntax reference

A quick syntax reference for the Micro programming language

Generic syntax

Script : Any sequence of expressions, separated by `;`, last semicolon optionnal

Comments : inline with `##` or multiline enclosed in `#' ... '#`

Literals

Base building block of every Micro script

Are however almost never explicitely written (but implicitely generated by special syntactic forms)

Syntax : `'lit` or ``'`lit with spaces or special chars`` ``

Operators

Arity : sets how many operands it takes

Precedence : sets who gets the parentheses

Syntax :

- any combination of `&|~^@=+-*%!$/:.,?!<>`
- `#alphanumeric_chars_with_no_spaces`

Special operators :

- `#call` : `f(...)`
- `#index` : `x[...]`
- `#list` : `[a;b;...]`
- `#tuple` : `(a;b;...)`

Primitive operators :

- `#number`
  - `12` is `[#number '`12`]`
  - `1.2` is `[#number '`1.2`]`
  - `0b011` is `[#number '`011` 'b]`
- `#string`
  - `"s"` is `[#string 's]`
  - `"s0 {arg1} ... {argn} sn"` is `[#string 's0 arg1 ... argn 'sn]`
- `#name`
  - `identifier` is `[#name 'identifier]`

Macros

Are valid expressions

Three styles : block, inline, declarative

Block :

- `block (args) { stmts } limb { stmts }`
- `block { stmts } limb { stmts }`
- `block (args) stmt limb stmt`
- `block stmt limb stmt`

Inline :

- `inline (args) limb expr`
- `inline arg limb expr`
- `inline`

Declarative :

- `decl name(args) limb expr { stmts }`
- `decl name limb expr { stmts }`

where `stmts` and `args` are (possibly empty) lists of expressions seperated by semicolons (last one being optionnal)