```
;; comments
; text between ; and eol is skipped
#| this text is also skipped \#
#; skips next single s-expression

;; e = expression, p = pair
;; i, j = iter var; a,b,x = vars
;; l = list, v = vector,
;; t = hash table, k = key

;;quoting
'e      (quote e)
`e      (quasiquote e)
,e      (unquote e)
,@e     (unquote-splicing e)
;; data types
(boolean? e)        (pair? e)
(symbol? e)         (number? e)
(char? e)           (vector? e)
atom? fixnum?  negative?  zero?
procedure?  builtin?  bound?
positive?  even?  odd?  null? identity

;;equality
(eq? a b)
(eqv? a b)      ;number, string
(equal? a b)    ;list contents

;; operators
+  -  *  /  >  <
(quotient a b)  ;integer division
(= nums..)      ;numeric equality
(lognot a)      (logand a b)
(logior a b)    (logxor a b)
(ash a)  ; bit shift
mod mod0  div  abs  max  min

;; logic   #t   #f
(and e...)      ;short circuit
(or e..)        ;short circuit
(not e)         (compare? e e)
```

```
;;characters
#\a  #\1
#\newline  #\space

;;strings
"hello"

;;list/pair
(012)   ()
(cons h t)
(car p)   (cdr p)
(set-car! p i)
(set-cdr! p i)
(list? p)
(length p)
(list e...)
(append l...)
(reverse l)
(list-ref l i)

;;vector
#(012)  #()
(vector e...)
(vector.alloc n x)
(aref v)
(aset! v i x)
(vector->list v)
(list->vector l)

;;hash table

(table k x k x ...)
(put! t k x)
(get t k dval)
(has t k)
(del! t k)
(table.keys t)
(table.pairs t)
```

femtolisp

Programming Language Quick Reference Card

v0.5
Man code.google.com/p/femtolisp/wiki/Manual
API code.google.com/p/femtolisp/wiki/APIReference

```
(load filename-string)
(begin e... )    ;evaluate expr's
(prog1 e... )    ;sequential eval and
                   return 1st eval

;; variables
(set! sym e)
(define var e...)
(let ((var e)...) e...)
(let* ...)          ;in sequence
(letrec ...)        ;recursive procs

;; procedures
(define (proc args..) body..)
(lambda (args..) body..)

;; control flow
(if test true-expr
   false-expr)
(cond (test body..)..
      (else body..))
(case e
   ((x..) body..)..
   (else body..))
(do ((x init update)..)
    (testexit body..)
    body..))
(for h t (lambda (args) body))
(while test . body..)
```

```
;named let:
(let name ((v e)..) e..)

(yeild x) return a value in generator
(trycatch expr function)
(raise e)
(return e)

;;control functions
(with-delimited-continuations
  proc)
(map proc l...)
(for-each proc l...)

;;macros
(set-syntax! sym function)

;patterns
x               ;variable
x...            ;repetition
pat...          ;repeated pattern

;; other
append!, assoc, assv, assq,
member, memv, memq, every, any,
list-tail, list-ref, list*,
last-pair, lastcdr, length=,
length>, map!, mapcar, for-each,
filter, count, foldr, foldl,
reverse!, copy-list, copy-tree,
map-int, iota, revappend, nreconc,
delete-duplicates
```