ok.js Source Code

```
////////////////////////////////////
//
//   A small(ish) implementation
//   of the K programming language.
//
//   John Earnest
//
////////////////////////////////////

"use strict";

var TN = [
        "number"    , //  0 : value
        "char"      , //  1 : value
        "symbol"    , //  2 : value
        "list"      , //  3 : array -> k
        "dictionary", //  4 : values, k(keys)
        "function"  , //  5 : body, args, curry, env
        "view"      , //  6 : value, r, cache, depends->val
        "nameref"   , //  7 : name, l(index?), r(assignment), global?
        "verb"      , //  8 : name, l(?), r, curry?
        "adverb"    , //  9 : name, l(?), verb, r
        "return"    , // 10 : return (deprecated)
        "nil"       , // 11 :
        "cond"      , // 12 : body (list of expressions)
        "quote"     , // 13 : value (for quoting verbs/etc as a value)
];

var NIL = ks("");
var k0 = k(0, 0);
var k1 = k(0, 1);
var EC = [["\\","\\\\"],["\"","\\\""],["\n","\\n"],["\t","\\t"]];
var kt = [-9, -10, -11, 0, 99, 102, NaN, NaN, 107, 105, NaN, NaN, NaN];
var SP = k(1, " ".charCodeAt(0));
var NA = k(0, NaN);

function k     (t, v)    { return { 't':t, 'v':v }; }
function md    (x, y)    { return { t:4, k:sl(x,y), v:y }; }
function ks    (x)       { return k(2, x); }
function asVerb(x, y, z) { return { t:8, v:x, l:y, r:z }; }
function kl    (x)       { return x.length==1 ? x[0] : k(3,x); }
function kf    (x)       { return match(k(3,[]), x).v || match(k0, x).v; }
function kb    (x)       { return x ? k1 : k0; }
function s     (x)       { return x.t == 3 && x.v.every(function(c) { return c.t == 1; }); }
function kmod  (x, y)    { return x-y*Math.floor(x/y); }
function len   (x)       { return l(x).v.length; }
function krange(x, f)    { var r=[]; for(var z=0;z<x;z++) { r.push(f(z)); } return k(3,r); }
function h2    (x)       { return (x.v+0x100).toString(16).substr(-2); }
function lget  (x, y)    { if(y<0||y>=len(x)) { throw new Error("length error."); } return x.v[y]; }
function dget  (x, y)    { var i=find(x.k, y); return (i.v==len(x.k)) ? NA : atx(x.v, i); }
function lset  (x, y, z) { if (len(x) <= p(y)) { throw new Error("index error."); } x.v[y.v]=z; }
function dset  (x, y, z) { var i=find(x.k, y).v; if(i==len(x.k)) { x.k.v.push(y); } x.v.v[i]=z; }
function lower (x)       { return k(1, String.fromCharCode(x.v).toLowerCase().charCodeAt(0)); }
function kmap  (x, f)    { return k(3, l(x).v.map(f)); }
function kzip  (x, y, f) { return kmap(sl(x,y), function(z, i) { return f(z, y.v[i]); }); }
function sl    (x, y)    { if (len(x) != len(y)) { throw new Error("length error."); } return x; }
function n     (x)       { return (x.t==0||x.t==1) ? x : ct(x, 0); }
function l     (x)       { return ct(x, 3); }
function d     (x)       { return ct(x, 4); }
function a     (x)       { if (x.t > 2) { throw new Error("domain error."); } return x; }
function na    (x)       { return x.t === 0 && isNaN(x.v); }

function stok(x) { return kl(krange(x.length, function(z) { return k(1,x.charCodeAt(z)); }).v); }
function c(x)    { return (x.t==3) ? k(x.t, x.v.slice(0)) : (x.t==4) ? md(c(x.k), c(x.v)) : x; }
function ct(n,t) { if (n.t!=t) throw new Error(TN[t]+" expected, found "+TN[n.t]+"."); return n; }
function p(x) { if (n(x).v<0||x.v%1!=0) { throw new Error("positive int expected."); } return x.v; }
function ktos(x, esc) {
        if (x.t != 3) { x = enlist(x); }
        var h = x.v.some(function(v){ return (v.v<32||v.v>127)&v.v!=9&v.v!=10; });
        if (h) { return "0x"+x.v.map(h2).join(""); }
        var r = x.v.map(function(k) { return String.fromCharCode(k.v); }).join("");
        return esc ? '"'+EC.reduce(function(r,p) { return r.split(p[0]).join(p[1]); }, r)+'"' : r;
}

////////////////////////////////////
//
//   Primitive Verbs
//
////////////////////////////////////

function plus  (x, y) { return k(0, n(x).v + n(y).v); }
function minus (x, y) { return k(0, n(x).v - n(y).v); }
function times (x, y) { return k(0, n(x).v * n(y).v); }
function divide(x, y) { return k(0, n(x).v / n(y).v); }
function mod   (x, y) { return k(0, n(x).v>0 ? kmod(n(y).v, x.v) : Math.floor(n(y).v / -x.v)); }
function max   (x, y) { return na(x)?y:na(y)?x:k(0, Math.max(n(x).v, n(y).v)); }
function min   (x, y) { return                 k(0, Math.min(n(x).v, n(y).v)); }
function less  (x, y) { return kb(a(x).v < a(y).v); }
function more  (x, y) { return kb(a(x).v > a(y).v); }
function equal (x, y) { return kb((x.v == y.v) || (na(x) && na(y))); }
function join  (x, y) { return l(y).v.reduce(function(z, y) { return cat(z, cat(x, y)); }); }
function ident    (x) { return x; }
function negate   (x) { return k(0, -n(x).v); }
function first    (x) { return (x.t == 4) ? first(x.v) : (x.t != 3) ? x : len(x) ? x.v[0]:k(3,[]); }
function sqrt     (x) { return k(0, Math.sqrt(n(x).v)); }
function keys     (x) { return c(d(x).k); }
function rev      (x) { return x.t==4?md(rev(x.k),rev(x.v)):x.t==3?k(3,c(l(x)).v.reverse()):x; }
function asc      (x) { return grade(-1, x); }
function desc     (x) { return grade(1, x); }
function not      (x) { return equal(n(x), k0); }
function enlist   (x) { return k(3, [x]); }
function isnull   (x) { return max(match(x, NIL),match(x,k(11))); }
function count    (x) { return k(0, x.t == 4 ? len(x.v) : x.t == 3 ? len(x) : 1); }
function floor    (x) { return x.t == 1 ? lower(x) : k(0, Math.floor(n(x).v)); }
function type     (x) { return k(0, kt[x.t]); }
function kfmt     (x) { var r=stok(format(x, 0, 1)); return r.t == 3 ? r : enlist(r); }
function real     (x) { return krange(n(x).v, function() { return k(0, Math.random()); }); }

function iota(x) {
        if (x.t == 4) { return keys(x); }
        var i = krange(Math.abs(n(x).v), k.bind(null, 0)); return x.v>=0 ? i : ar(plus)(x, i);
}

function cat(x, y) {
        if (x.t==4&&y.t==4) { x=c(x); kmap(y.k, function(v) { dset(x,v,dget(y,v)); }); return x; };
        return k(3, (x.t==3?x.v:[x]).concat(y.t==3?y.v:[y]));
}

function keval(x, env) {
        if (x.t == 5) { return x.env.d; }
        return x.t == 4 ? c(x.v) : x.t == 2 ? env.lookup(x, true) : run(parse(ktos(x)), env);
}

function dfmt(x, y) {
        if ( x.t == 3           && y.t == 3) { return kzip(x, y, dfmt); }
        if ( x.t == 3           && y.t != 3) { return kmap(x, function(z) { return dfmt(z, y); }); }
        if ((x.t == 2 || !s(y)) && y.t == 3) { return kmap(y, function(z) { return dfmt(x, z); }); }
        if (x.t == 2) { return {b: k(0,y.v&1), i: k(0,y.v|0), f: k(0,y.v), c: k(1,y.v)}[x.v]; }
        if (y.t == 1) { return y; } var r=c(y); var d=Math.abs(x.v);
        while(len(r) < d) { x.v>0 ? r.v.push(SP) : r.v.unshift(SP); }
        while(len(r) > d) { x.v>0 ? r.v.pop()    : r.v.shift();     }
        return r;
}

function except(x, y) {
        y = y.t == 3 ? y : enlist(y);
        return k(3, (x.t == 3 ? x : iota(x)).v.filter(function(z) { return na(pfind(y, z)); }));
}

function ddrop(x, y) { var k = except(d(y).k, x); return md(k, atx(y, k)); }
function drop(x, y) {
        if (y.t == 4) { return md(drop(x, y.k), drop(x, y.v)); }
        return (y.t != 3 || len(y) < 1) ? y : k(3, n(x).v<0 ? y.v.slice(0,x.v) : y.v.slice(x.v));
}

function take(x, y, env) {
        if (x.t == 5 || x.t == 8 || x.t == 9) {
                var k = where(each(x, y, env), env); var v = atx(y, k);
                return y.t == 4 ? md(k, v) : v;
        }
        if (y.t == 4) { return md(take(x, y.k, env), take(x, y.v, env)); }
        if (y.t != 3 || len(y) == 0) { y = enlist(y); }
        var s=n(x).v<0?kmod(x.v, len(y)):0;
        return krange(Math.abs(x.v), function(x) { return y.v[kmod(x+s, len(y))]; });
}

function reshape(x, y) {
        if (y.t == 4) { return md(x, atx(y, x)); }
        if (y.t != 3) { y = enlist(y); }
        var a = first(x); var b = x.v[len(x)-1]; var c = 0;
        function rshr(x, y, i) {
                return krange(x.v[i].v, function(z) {
                        return i==len(x)-1 ? y.v[kmod(c++, len(y))] : rshr(x, y, i+1);
                });
        }
        return na(a) ? (!len(y) ? y : cut(krange(len(y)/b.v, function(z) { return k(0, z*b.v); }), y)) :
               na(b) ? cut(krange(a.v, function(z) { return k(0, Math.floor(z*len(y)/a.v)); }), y) :
               rshr(l(x), len(y) ? y : enlist(y), 0);
}

function match(x, y) {
        if (x.t != y.t) { return k0; }
        if (x.t == 4) { return min(match(x.k, y.k), match(x.v, y.v)); }
        if (x.t != 3) { return equal(x, y); }
        if (len(x) != len(y)) { return k0; }
        return kb(x.v.every(function(x,i) { return match(x, y.v[i]).v; }));
}

function find(x, y) { y=x.v.findIndex(function(z){return match(z,y).v}); return k(0,y>=0?y:len(x)) }
function pfind(x, y) { y=x.v.findIndex(function(z){return equal(z,y).v}); return y>=0?k(0,y):NA }
function pisnull(x) { return kb(match(x, NIL).v || match(x, k(11)).v || na(x)); }

function cut(x, y) {
        return kzip(x, cat(drop(k1,x),count(y)), function(a, b) { // {x{x@y+!z-y}[y]'1_x,#y} ?
                var r=[]; for(var z=p(a);z<p(b);z++) { r.push(lget(y,z)); } return k(3,r);
        });
}

function rnd(x, y, env) {
        if (x.t == 4) { return atx(x.k, ar(pfind)(x.v,y), env); }
        if (y.t == 1) { return dfmt(k(2,"c"),rnd(x,ar(plus)(y,iota(k(0,26))))); }
        if (y.t == 3) { return atx(y, rnd(x, count(y))); } p(y);
        if (n(x).v<0) { if(-x.v>y.v) throw new Error("length error.");return take(x,asc(real(y)),env); }
        return kmap(iota(x), function(x){ return k(0,Math.floor(Math.random()*y.v)); });
}

function flip(x, env) {
        x=eachright(k(8,"#"), over(k(8,"|"), each(k(8,"#"), x, env), env), x, env);
        return krange(len(first(x)), function(z){
                return krange(len(x), function(t){ return x.v[t].v[z]; });
        });
}

function grade(dir, x) {
        return x.t == 4 ? atx(x.k, grade(dir, x.v)) : k(3, iota(count(x)).v.sort(function(a, b) {
                var f = function(i) { var v = x.v[i.v]; return s(v) ? ks(ktos(v)) : v; }
                var av = f(a), bv = f(b); return less(av,bv).v ? dir : more(av,bv).v ? -dir : a.v - b.v;
        }));
}

function where(x, env) {
        if (x.t == 4) { return atx(x.k, where(x.v, env)); } // {,/(0|x)#'!#x}...
        var s = kmap(x.t==3 ?x:enlist(x), function(v,i) { return take(k(0,p(v)), k(0,i), env); });
        return over(asVerb(","), s, env);
}

function group(x) {
        var r={t:4, k:unique(x)}; r.v=kmap(r.k, function(){ return k(3,[]); });
        for(var z=0;z<len(x);z++) { dget(r, x.v[z]).v.push(k(0, z)); } return r;
}

function unique(x) {
        var r=[]; for(var z=0;z<len(x);z++) {
                if (!r.some(function(e) { return match(x.v[z], e).v })) { r.push(x.v[z]); }
        } return k(3,r);
}

function bin(x, y) {
        var a=0; var b=len(x); if (b<1 || less(y, first(x)).v) { return k(0,-1); }
        while(b - a > 1) { var i=a+Math.floor((b-a)/2); if (more(x.v[i], y).v) { b=i; } else { a=i; } }
        return k(0, a);
}

function split  (x, y) { return (x.t != 1) ? unpack(x, y) : call(splitimpl, k(3, [x,y])); }
function unpack (x, y) { return call(unpackimpl, k(3, [x,y])); }
function pack   (x, y) { return (x.t == 1) ? join(x, y) : call(packimpl, k(3, [x,y])); }
function kwindow(x, y) { return call(winimpl, k(3, [x,y])); }
function splice(xyz)   { return call(spliceimpl, k(3, xyz)); }
function imat(x)       { var i = iota(x); return kmap(i, function(z) { return ar(equal)(z, i); }); }
function odometer(x)   { return call(odoimpl, enlist(x)); }

////////////////////////////////////
//
//   Primitive Adverbs
//
////////////////////////////////////

function each(monad, x, env) {
        if (x.t == 4) { return md(x.k, each(monad, x.v, env)); }
        return kmap(x, function(x) { return applym(monad, x, env); });
}

function eachd(dyad, left, right, env) {
        if (!env) { return kmap(left, function(x) { return applyd(dyad, x, null, right); }); }
        if (left.t==4&&right.t==4) { return md(left.k,eachd(dyad,left.v,atx(right,left.k),env)); }
        if (left.t!=3) { return eachright(dyad, left, right, env); }
        if (right.t!=3) { return eachleft(dyad, left, right, env); }
        return kzip(left, right, function(x, y) { return applyd(dyad, x, y, env); });
}

function eachright(dyad, left, list, env) {
        return kmap(list, function(x) { return applyd(dyad, left, x, env); });
}

function eachleft(dyad, list, right, env) {
        return kmap(list, function(x) { return applyd(dyad, x, right, env); });
}

function eachprior(dyad, x, env) {
        var specials = {"+":k0, "*":k1, "-":k0, "&":first(x), ",":k(3,[]), "%":k1};
        return eachpc(dyad, (dyad.v in specials) ? specials[dyad.v] : NA, x, env);
}

function eachpc(dyad, x, y, env) {
        return kmap(y, function(v) { var t=x; x=v; return applyd(dyad, v, t, env); });
}

function over(dyad, x, env) {
        var specials = {"+":k0, "*":k1, "|":k(0,-1/0), "&":k(0,1/0)};
        if (x.t == 3 && len(x) < 1 && dyad.v in specials) { return specials[dyad.v]; }
        if (x.t == 3 && len(x) == 1 && dyad.v == ",") { return first(x).t != 3 ? x : first(x); }
        if (x.t != 3 || len(x) < 1) { return x; }
        return overd(dyad, first(x), drop(k1,x), env);
}

function overd(dyad, x, y, env) {
        return y.v.reduce(function(x, y) { return applyd(dyad, x, y, env); }, x);
}

function eacha(func, args, env) {
        var x = args[0]; var y = flip(k(3, args.slice(1)), env);
        if (x.t != 3) { return kmap(y, function(y) { return call(func, cat(x, y), env); }); }
        return kzip(x, y, function(x, y) { return call(func, cat(x, y), env); });
}
function overa(func, args, env) {
        var x = args[0]; var y = flip(k(3, args.slice(1)), env);
        return y.v.reduce(function(x, y) { return call(func, cat(enlist(x), y), env); }, x);
}
function scana(func, args, env) {
        var x = args[0]; var y = flip(k(3, args.slice(1)), env);
        return cat(x, kmap(y, function(y) { return x = call(func, cat(enlist(x), y), env); }));
}

function fixed(monad, x, env) {
        var r=x, p=x;
        do { r=applym(monad, p=r, env); } while(!match(p, r).v && !match(r, x).v); return p;
}

function fixedwhile(monad, x, y, env) {
        if (x.t == 0) { for(var z=0;z<x.v;z++) { y = applym(monad, y, env); } }
        else { do { y = applym(monad, y, env); } while (applym(x, y, env).v); } return y;
}

function scan(dyad, x, env) {
        if (x.t != 3 || len(x) <= 1) { return x; }
        var i = first(x); var r = enlist(i);
        kmap(drop(k1,x), function(z) { r.v.push(i = applyd(dyad, i, z, env)); }); return r;
}

function scand(dyad, x, y, env) {
        return kmap(y, function(v) { return x = applyd(dyad, x, v, env); });
}

function scanfixed(monad, x, env) {
        var r=[x]; while(1) {
                var p = r[r.length-1]; var n = applym(monad, p, env);
                if (match(p, n).v || match(n, x).v) { break; } r.push(n);
        } return k(3,r);
}

function scanwhile(monad, x, y, env) {
        var r=[y]; if (x.t == 0) { for(var z=0;z<x.v;z++) { r.push(y = applym(monad, y, env)); } }
        else { do { y = applym(monad, y, env); r.push(y); } while (applym(x, y, env).v != 0); }
        return k(3, r);
}

////////////////////////////////////
//
//   Interpreter
//
////////////////////////////////////

function am(f) { // create an atomic monad
        return function recur(x, env) {
                return x.t == 4 ? md(x.k, recur(x.v, env)) :
                       x.t == 3 ? kmap(x, function(x) { return recur(x, env); }) : f(x, env);
        };
}
function ar(f) { // create a right atomic dyad
        return function recur(x, y, env) {
                return y.t == 3 ? kmap(y, function(z) { return recur(x, z, env); }) : f(x, y, env);
        };
}
function ad(f) { // create an atomic dyad
        return function recur(x, y, env) {
                if (x.t == 4 && y.t == 4) {
                        var r=md(k(3,[]),k(3,[])); kmap(unique(cat(x.k,y.k)), function(k) {
                                var a=dget(x,k), b=dget(y,k); dset(r,k,a==NA?b:b==NA?a:recur(a,b,env));
                        }); return r;
                }
                return x.t == 3 && y.t == 3 ? kzip(x, y, function(a,b) { return recur(a, b, env); }) :
                       x.t == 4 ? md(x.k, recur(x.v, y, env)) :
                       y.t == 4 ? md(y.k, recur(x, y.v, env)) :
                       x.t == 3 ? kmap(x, function(z) { return recur(z, y, env); }) :
                       y.t == 3 ? kmap(y, function(z) { return recur(x, z, env); }) : f(x, y, env);
        };
}

function applym(verb, x, env) {
        if (verb.t == 5) { return call(verb, enlist(x), env); }
        if (verb.t == 3) { return atx(verb, x, env); }
        if (verb.t == 9 & verb.r == null) { verb.r=x; var r=run(verb, env); verb.r=null; return r; }
        if (verb.sticky) {
                var s=verb.sticky; s.r=x; verb.sticky=null;
                var r=run(verb, env); verb.sticky=s; s.r=null; return r;
        }
        return applyverb(verb, [x], env);
}

function applyd(verb, x, y, env) {
        if (verb.t == 5) { return call(verb, k(3,[x,y]), env); }
        if (verb.sticky && verb.sticky != verb) {
                var s=verb.sticky; s.l=x; s.r=y; verb.sticky=null;
                var r=run(verb, env); verb.sticky=s; s.r=null; s.l=null; return r;
        }
        return applyverb(verb, [x, y], env);
}

var verbs = {
        //     a          l           a-l         l-a         a-l         l-l         triad    tetrad
        "+" : [ident,     flip,       ad(plus),   ad(plus),   ad(plus),   ad(plus),   null,    null  ],
        "-" : [am(negate),am(negate), ad(minus),  ad(minus),  ad(minus),  ad(minus),  null,    null  ],
        "*" : [first,     first,      ad(times),  ad(times),  ad(times),  ad(times),  null,    null  ],
        "%" : [sqrt,      am(sqrt),   ad(divide), ad(divide), ad(divide), ad(divide), null,    null  ],
        "!" : [iota,      odometer,   mod,        null,       ar(mod),    md,         null,    null  ],
        "&" : [where,     where,      ad(min),    ad(min),    ad(min),    ad(min),    null,    null  ],
        "|" : [rev,       rev,        ad(max),    ad(max),    ad(max),    ad(max),    null,    null  ],
        "<" : [asc,       asc,        ad(less),   ad(less),   ad(less),   ad(less),   null,    null  ],
        ">" : [desc,      desc,       ad(more),   ad(more),   ad(more),   ad(more),   null,    null  ],
        "=" : [imat,      group,      ad(equal),  ad(equal),  ad(equal),  ad(equal),  null,    null  ],
        "~" : [am(not),   am(not),    match,      match,      match,      match,      null,    null  ],
        "," : [enlist,    enlist,     cat,        cat,        cat,        cat,        null,    null  ],
        "^" : [pisnull,   am(pisnull),except,     except,     except,     except,     null,    null  ],
        "#" : [count,     count,      take,       reshape,    take,       reshape,    null,    null  ],
        "_" : [am(floor), am(floor),  drop,       ddrop,      drop,       cut,        null,    null  ],
        "$" : [kfmt,      am(kfmt),   dfmt,       dfmt,       dfmt,       dfmt,       null,    null  ],
        "?" : [real,      unique,     rnd,        pfind,      rnd,        ar(pfind),  splice,  null  ],
        "@" : [type,      type,       atx,        atx,        atx,        atx,        amend4,  amend4],
        "." : [keval,     keval,      call,       call,       call,       call,       dmend4,  dmend4],
        "'" : [null,      null,       null,       bin,        null,       ar(bin),    null,    null  ],
        "/" : [null,      null,       null,       null,       pack,       pack,       null,    null  ],
        "\\": [null,      null,       null,       unpack,     split,      null,       null,    null  ],
        "':": [null,      null,       null,       null,       kwindow,    null,       null,    null  ],
};

function applyverb(node, args, env) {
        if (node.curry) {
                var a=[]; var i=0; for(var z=0;z<node.curry.length;z++) {
                        if (!isnull(node.curry[z]).v) { a[z]=run(node.curry[z], env); continue; }
                        while(i<args.length && !args[i]) { i++; } if (!args[i]) { return node; }
                        a[z]=args[i++];
                } args = a;
        }
        if (node.t == 9) { return applyadverb(node, node.verb, args, env); }
        var left  = args.length == 2 ? args[0] : node.l ? run(node.l, env) : null;
        var right = args.length == 2 ? args[1] : args[0];
        if (!right) { return { t:node.t, v:node.v, curry:[left,k(11)] }; }
        var r = null; var v = verbs[node.forcemonad ? node.v[0] : node.v];
        if (!v) {}
        else if (args.length == 3)            { r = v[6]; }
        else if (args.length == 4)            { r = v[7]; }
        else if (!left       && right.t != 3) { r = v[0]; }
        else if (!left       && right.t == 3) { r = v[1]; }
        else if (left.t != 3 && right.t != 3) { r = v[2]; }
        else if (left.t == 3 && right.t != 3) { r = v[3]; }
        else if (left.t != 3 && right.t == 3) { r = v[4]; }
        else if (left.t == 3 && right.t == 3) { r = v[5]; }
        if (!r) { throw new Error("invalid arguments to "+node.v); }
        return (args.length > 2) ? r(args, env) : left ? r(left, right, env) : r(right, env)
}

function valence(node, env) {
        if (node.t == 5) {
                return (node.curry||[]).reduce(function(x,v) { return x-!isnull(v).v; }, node.args.length);
        }
```

```
	if (node.t == 7) { return valence(env.lookup(ks(node.v))); }
	if (node.t == 9 && node.v == "'") { return valence(node.verb, env); }
	if (node.t == 9)       { return 1; }
	if (node.t != 8)       { return 0; }
	if (node.forcemonad)   { return 1; }
	if (node.v in natives) { return 1; }
	return (node.sticky && (node.sticky.t==9 || node.sticky.forcemonad || node.sticky.l)) ? 1 : 2;
}

var adverbs = {
	//       mv/nv       dv          l-mv         l-dv       3+v
	"':"  : [null,       eachprior,  null,        eachpc,    null ],
	"'"   : [each,       eachd,      eachd,       eachd,     eacha],
	"/:"  : [null,       null,       eachright,   eachright, null ],
	"\\:" : [null,       null,       eachleft,    eachleft,  null ],
	"/"   : [fixed,      over,       fixedwhile,  overd,     overa],
	"\\"  : [scanfixed,  scan,       scanwhile,   scand,     scana],
};

function applyadverb(node, verb, args, env) {
	if (verb.t == 7) { verb = run(verb, env); }
	var r = null; var v = valence(verb, env);
	if (v > 2)                 { return adverbs[node.v][4](verb, args, env); }
	if (v == 0 && verb.t != 5) { return applyverb(k(8,node.v), [verb, args[1]], env); }
	if (v == 0 && verb.t == 5) { v = 1; }
	if (v == 2 && !args[1])    { args = [null, args[0]]; }
	if (v == 1 && !args[0])    { r = adverbs[node.v][0]; }
	if (v == 2 && !args[0])    { r = adverbs[node.v][1]; }
	if (v == 1 &&  args[0])    { r = adverbs[node.v][2]; }
	if (v == 2 &&  args[0])    { r = adverbs[node.v][3]; }
	if (!r) { throw new Error("invalid arguments to "+node.v+" ["+
		(args[0]?format(args[0])+" ":"")+" "+format(verb)+" (valence "+v+"), "+format(args[1])+"]");
	}
	return args[0] ? r(verb, args[0], args[1], env) : r(verb, args[1], env);
}

function Environment(pred) {
	this.p = pred; this.d = md(k(3,[]), k(3,[]));
	this.put = function(n, g, v) {
		if (typeof n == "string") { n = ks(n); }
		if (g && this.p) { this.p.put(n, g, v); } else { dset(this.d, n, v); }
	};
	this.contains = function(x) { return find(this.d.k, x).v != len(this.d.k); }
	this.lookup = function(n, g) {
		if (g && this.p) { return this.p.lookup(n, g); }
		if (!this.contains(n)) {
			if (!this.p) { throw new Error("the name '"+n.v+"' has not been defined."); }
			return this.p.lookup(n);
		}
		var view = dget(this.d, n);
		if (view.t == 6) {
			var dirty = view.cache == 0, env = this;
			Object.keys(view.depends).forEach(function(z) {
				var n = (z == view.v) ? view.cache : env.lookup(ks(z)), o = view.depends[z];
				if (!o || !match(n,o).v) { dirty=1; view.depends[z]=n; }
			})
			if (dirty) { view.cache = run(view.r, this); } return view.cache;
		}
		return view;
	};
}

function atx(x, y, env) {
	return x.t == 2 ? atx(env.lookup(x), y, env) : y.t == 11 ? x :
		x.t == 3 && y.t == 4 ? md(y.k, atx(x, y.v, env)) :
		x.t == 8 || x.t == 9 ? applym(x, y, env) :
		(x.t == 3 || x.t == 4) && y.t == 3 ? kmap(y, function(z) { return atx(x, z); }) :
		x.t == 3 ? (y.t > 1 || y.v < 0 || y.v >= len(x) || y.v%1 != 0) ? NA : x.v[y.v] :
		x.t == 4 ? dget(x, y) : call(x, enlist(y), env)
}

function atdepth(x, y, i, env) {
	if (i >= len(y)) { return x; }; var c = y.v[i]; var k = atx(x, c, env);
	return (c.t != 11 && c.t != 3) ?    atdepth(k, y, i+1, env) :
		kmap(k, function(t) { return atdepth(t, y, i+1, env) })
}

function call(x, y, env) {
	if (x.sticky) { return (valence(x.sticky, env)==1?applym:applyd)(x, y.v[0], y.v[1], env); }
	if (x.t == 2) { return call(env.lookup(x), y, env); }
	if (x.t == 3 || x.t == 4) { return y.t == 3 ? atdepth(x, y, 0, env) : atx(x, y, env); }
	if (x.t == 8) { return applyverb(x, y.t == 3 ? y.v : [y], env); }
	if (x.t == 9) { return applyadverb(x, run(x.verb, env), y.v, env); }
	if (x.t != 5) { throw new Error("function or list expected, found " + TN[x.t]+'.'); }
	if (y.t == 4) { var e=new Environment(null); e.d=y; x.env=e; return x; }
	if (y.t != 3) { y = enlist(y); }
	var environment = new Environment(x.env); var curry = x.curry?x.curry.concat([]):[];
	if (x.args.length != 0 || len(y) != 1 || !isnull(y.v[0]).v) {
		var all=true; var i=0; for(var z=0;z<x.args.length;z++) {
			if (curry[z] && !isnull(curry[z]).v) { continue; }
```

```
                        if (i >= len(y)) { all=false; break; }
                        if (y.v[i] == null || isnull(y.v[i]).v) { all=false; }
                        curry[z]=y.v[i++];
                }
                if (!all) { return { t:5, v:x.v, args:x.args, env:x.env, curry:curry }; }
                if (i < len(y) && x.args.length != 0) { throw new Error("valence error."); }
                for(var z=0;z<x.args.length;z++) { environment.put(ks(x.args[z]), false, curry[z]); }
        }
        environment.put(ks("o"), false, x); return run(x.v, environment);
}

function run(node, env) {
        if (node instanceof Array) { return node.reduce(function(_,x) { return run(x, env); }, null); }
        if (node.sticky) { return node; }
        if (node.t == 3) { return rev(kmap(rev(node), function(v) { return run(v, env); })); }
        if (node.t == 4) { return md(node.k, kmap(node.v, function(x) { return run(x, env); })); }
        if (node.t == 5) {
                if (node.r) { return atx(node, run(node.r, env), env); }
                if (!node.env) { return { t:5, v:node.v, args:node.args, curry:node.curry, env:env }; }
        }
        if (node.t == 6) { env.put(ks(node.v), false, node); return node; }
        if (node.t == 7) {
                if (node.r) { env.put(ks(node.v), node.global, run(node.r, env)); }
                return env.lookup(ks(node.v));
        }
        if (node.t == 8 && node.curry && !node.r) { return applyverb(node, [], env); }
        if (node.t == 8 && node.r) {
                var right = run(node.r, env);
                var left  = node.l ? run(node.l, env) : null;
                return applyverb(node, [left, right], env);
        }
        if (node.t == 9 && node.r) {
                var right = run(node.r, env);
                var verb  = run(node.verb, env);
                var left  = node.l ? run(node.l, env) : null;
                return applyadverb(node, verb, [left, right], env);
        }
        if (node.t == 12) {
                for(var z=0;z<node.v.length-1;z+=2) {
                        if (!kf(run(node.v[z], env))) { return run(node.v[z+1], env); }
                } return run(node.v[node.v.length-1], env);
        }
        if (node.t == 13) { return run(node.v, env); }
        return node;
}

function amend4(args, env) { return mend(args, env, amendm, amendd); }
function dmend4(args, env) { return mend(args, env, dmend, dmend); }

function mend(args, env, monadic, dyadic) {
        var ds = args[0], i = args[1], f = args[2], y = args[3];
        (y?dyadic:monadic)(ds.t == 2 ? env.lookup(ds,true) : ds, i, y, f, env); return ds;
}

function amendm(d, i, y, monad, env) {
        if (monad.t == 0) { monad = { t:5,args:["x"],v:[{ t:0, v:monad.v }] }; }
        if (i.t != 3) { lset(d, i, applym(monad, atx(d, i, env), env)); }
        else { kmap(i, function(v) { amendm(d, v, y, monad, env); }); }
}

function amendd(d, i, y, dyad, env) {
        if (i.t == 3) { kmap(i, function(iv, z) { amendd(d, iv, y.t == 3 ? y.v[z] : y, dyad, env) }); }
        else { (d.t == 4 ? dset : lset)(d, i, applyd(dyad, atx(d, i, env), y, env)); }
}

function dmend(d, i, y, f, env) {
        if (i.t != 3) { (y?amendd:amendm)(d, i, y, f, env); return; }
        if (len(i) == 1) { dmend(d, i.v[0], y, f, env); return; }
        var rest = drop(k1,i); if (len(i)<1) { return; } if (i.v[0].t == 3) {
                if (y && y.t == 3) { kzip(i, y, function(a, b) { amendd(d, a, b, f, env); }); return; }
                kmap(i.v[0],function(x) { dmend(atx(d,x,env), rest, y, f, env); });
        }
        else if (isnull(i.v[0]).v) { kmap(d,function(x,i) { dmend(atx(d,k(0,i),env),rest,y,f,env); }); }
        else if (d.v[0].t != 3) { (y?amendd:amendm)(d, i, y, f, env); }
        else { dmend(atx(d, first(i), env), rest, y, f, env); }
}

////////////////////////////////////
//
//   Tokenizer
//
////////////////////////////////////

var NUMBER  = /^(-?0w|0N|-?\d+\.\d*|-?\d*\.?\d+)/;
var HEXLIT  = /^0x[a-zA-Z\d]+/;
var BOOL    = /^[01]+b/;
var NAME    = /^[a-z][a-z\d]*/i;
var SYMBOL  = /^`([a-z.][a-z0-9.]*)?/i;
var STRING  = /^"(\\.|[^"\\\r\n])*"/;
var VERB    = /^[+\-*%!&|<>=~,^#_$?@.]/;
var ASSIGN  = /^[+\-*%!&|<>=~,^#_$?@.]:/;
var IOVERB  = /^\d:/;
var ADVERB  = /^['\\\/]:?/;
var SEMI    = /^;/;
var COLON   = /^:/;
var VIEW    = /^::/;
var COND    = /^\$\[/;
var DICT    = /^\[[a-z]+:/i;
var OPEN_B  = /^\[/;
var OPEN_P  = /^\(/;
var OPEN_C  = /^{/;
var CLOSE_B = /^\]/;
var CLOSE_P = /^\)/;
var CLOSE_C = /^}/;

var des = {};
des[NUMBER ]="number";des[NAME   ]="name"   ;des[SYMBOL ]="symbol";des[STRING]="string";
des[VERB   ]="verb"  ;des[IOVERB ]="IO verb";des[ADVERB ]="adverb";des[SEMI  ]="';'";
des[COLON  ]="':'"   ;des[VIEW   ]="view"   ;des[COND   ]="'$['"  ;
des[OPEN_B ]="'['"   ;des[OPEN_P ]="'('"    ;des[OPEN_C ]="'{'"   ;des[ASSIGN]="assignment";
des[CLOSE_B]="']'"   ;des[CLOSE_P]="')'"    ;des[CLOSE_C]="'}'";

var text = "";
var funcdepth = 0;
function begin(str) {
        str = str.replace(/("(?:[^"\\\n]|\\.)*")|(\s\/.*)|([a-z\d\]\)]-\.?\d)/gi, function(_, x, y, z) {
                // preserve a string (x), remove a comment (y), disambiguate a minus sign (z)
                return x ? x : y ? "" : z.replace('-', '- ')
        })
        text = str.trim().replace(/\n/g, ";"); funcdepth = 0;
}
function done()         { return text.length < 1; }
function at(regex)      { return regex.test(text); }
function matches(regex) { return at(regex) ? expect(regex) : false; }
function expect(regex) {
        var found = regex.exec(text);
        if (regex == OPEN_C) { funcdepth++; } if (regex == CLOSE_C) { funcdepth--; }
        if (found == null) { throw new Error("parse error. "+des[regex]+" expected."); }
        text = text.substring(found[0].length).trim(); return found[0];
}

////////////////////////////////////
//
//   Parser
//
////////////////////////////////////

function findNames(node, names) {
        if (node == null)          { return names; }
        if (node instanceof Array) { node.forEach(function(v) { findNames(v, names); }); return names; }
        if (node.t == 7)           { names[node.v] = 0; }
        if (node.t != 5)           { findNames(node.v, names); }
        return findNames([node.l, node.r, node.verb, node.curry], names);
}

function atNoun() {
        return !done()&&at(NUMBER)||at(NAME)||at(SYMBOL)||at(STRING)||at(COND)||at(OPEN_P)||at(OPEN_C);
}

function indexedassign(node, indexer) {
        var op = { t:5, args:["x","y"], v:[{ t:7, v:"y" }] }; // {y}
        var gl = matches(COLON);
        var ex = parseEx(parseNoun());
        //t[x]::z  ->  ..[`t;x;{y};z]   t[x]:z  ->  t:.[t;x;{y};z]
        if (!gl) { node.r = { t:8, v:".", curry:[ k(7,node.v), kl(indexer), op, ex] }; return node; }
        return { t:8, v:".", r:{ t:8, v:".", curry:[ks(node.v), kl(indexer), op, ex] }};
}

function compoundassign(node, indexer) {
        if (!at(ASSIGN)) { return node; }
        var op = expect(ASSIGN).slice(0,1); var gl = matches(COLON); var ex = parseEx(parseNoun());
        if (!indexer) {
                // t+::z  -> t::(.`t)+z
                var v = gl ? asVerb(".", null, ks(node.v)) : node;
                return { t:node.t, v:node.v, global:gl, r:asVerb(op, v, ex) };
        }
        // t[x]+::z -> ..[`t;x;+:;z]   t[x]+:z -> t:.[t;x;{y};z]
        if (!gl) { node.r={ t:8, v:".", curry:[ k(7,node.v),kl(indexer),asVerb(op),ex] }; return node; }
        return asVerb(".", null, { t:8, v:".", curry:[ks(node.v), indexer, asVerb(op), ex] });
}

function applycallright(node) {
        while (matches(OPEN_B)) {
                var args = parseList(CLOSE_B); node = asVerb(".", node, k(3, args.length ? args : [NIL]));
        } return node;
}

function applyindexright(node) {
        if (node.sticky && at(VERB)) {
                var x = parseNoun(); x.l = node; x.r = parseEx(parseNoun()); return x;
```

```
	}
	while (matches(OPEN_B)) { node = asVerb(".", node, k(3, parseList(CLOSE_B))); }
	return node;
}

function findSticky(root) {
	var n = root; if (n == null || (n.t == 9 && n.r == null)) { return; }
	while(n.t == 8 && !n.curry || n.t == 9) {
		if (n.r == null) { root.sticky = n; return; } n = n.r;
	}
}

function parseList(terminal, cull) {
	var r=[]; do {
		if (terminal && at(terminal)) { break; }
		while(matches(SEMI)) { if (!cull) { r.push(k(11)); } }
		var e = parseEx(parseNoun()); findSticky(e);
		if (e != null) { r.push(e); }
		else if (!cull) { r.push(k(11)); }
	} while(matches(SEMI)); if (terminal) { expect(terminal); } return r;
}

function parseNoun() {
	if (matches(COLON)) { return { t:5, args:["x","y"], v:[{ t:7, v:"y" }] }; } // {y}
	if (at(IOVERB)) { return k(8, expect(IOVERB)); }
	if (at(BOOL)) {
		var n = expect(BOOL); var r=[];
		for(var z=0;z<n.length-1;z++) { r.push(k(0, parseInt(n[z]))); }
		return applyindexright(k(3, r));
	}
	if (at(HEXLIT)) {
		var h=expect(HEXLIT); if (h.length%2) { throw new Error("malformed byte string."); }
		var r=krange(h.length/2-1, function(z) { return k(1,parseInt(h.slice(2*z+2,2*z+4),16)); });
		return (r.v.length == 1) ? first(r) : r;
	}
	if (at(NUMBER)) {
		var r=[]; while(at(NUMBER)) {
			var n=expect(NUMBER); r.push(k(0, n=="0w"?1/0:n=="-0w"?-1/0:n=="0N"?NaN:parseFloat(n)));
		} return applyindexright(kl(r));
	}
	if (at(SYMBOL)) {
		var r=[]; while(at(SYMBOL)) { r.push(k(2, expect(SYMBOL).slice(1))); }
		return applyindexright(kl(r));
	}
	if (at(STRING)) {
		var str = expect(STRING); str = str.substring(1, str.length-1);
		for(var z=0;z<EC.length;z++) { str=str.split(EC[z][1]).join(EC[z][0]); }
		return applyindexright(stok(str));
	}
	if (matches(OPEN_B)) {
		var m=md(k(3,[]), k(3,[])); if (!matches(CLOSE_B)) { do {
			var key = ks(expect(NAME)); expect(COLON);
			dset(m, key, matches(COLON) ? dget(m, ks(expect(NAME))) : parseEx(parseNoun()));
		} while(matches(SEMI)); expect(CLOSE_B); } return applyindexright(m);
	}
	if (matches(OPEN_C)) {
		var args=[]; if (matches(OPEN_B)) {
			do { args.push(expect(NAME)); } while(matches(SEMI)); expect(CLOSE_B);
		}
		var r = k(5, parseList(CLOSE_C, true));
		if (args.length == 0) {
			var names = findNames(r.v, {});
			if      ("z" in names) { args = ["x","y","z"]; }
			else if ("y" in names) { args = ["x","y"]; }
			else if ("x" in names) { args = ["x"]; }
		}
		r.args = args; return applycallright(r);
	}
	if (matches(OPEN_P)) { return applyindexright(kl(parseList(CLOSE_P))); }
	if (matches(COND))   { return k(12, parseList(CLOSE_B, true)); }
	if (at(VERB)) {
		var r = k(8, expect(VERB));
		if (matches(COLON)) { r.v += ":"; r.forcemonad = true; }
		if (at(OPEN_B) && !at(DICT)) {
			expect(OPEN_B); r.curry = parseList(CLOSE_B, false);
			if (r.curry.length < 2 && !r.forcemonad) { r.curry.push(k(11)); }
		} return r;
	}
	if (at(NAME)) {
		var n = k(7, expect(NAME));
		if (n.v in natives) { return applycallright(k(8, n.v)); }
		if (funcdepth == 0 && matches(VIEW)) {
			var r = k(6, n.v);
			r.r = parseEx(parseNoun());
			r.depends = findNames(r.r, {});
			r.cache = k(11);
			return r;
		}
		if (matches(COLON)) {
			n.global = matches(COLON); n.r = parseEx(parseNoun());
```

```
                        if (n.r == null) { throw new Error("noun expected following ':'."); }
                        findSticky(n.r); if (n.r == n.r.sticky) { n.r.sticky = null; }
                        return n;
                }
                if (matches(OPEN_B)) {
                        var index = parseList(CLOSE_B);
                        if (at(ASSIGN)) { return compoundassign(n, index); }
                        if (matches(COLON)) { return indexedassign(n, index); }
                        if (index.length == 0) { index = [NIL]; }
                        n = asVerb(".", n, k(3, index));
                }
                return applycallright(compoundassign(n, null));
        }
        return null;
}

function parseAdverb(left, verb) {
        var a = expect(ADVERB);
        while(at(ADVERB)) { var b = expect(ADVERB); verb = { t:9, v:a, verb:verb }; a = b; }
        if (at(OPEN_B)) { return applycallright({ t:9, v:a, verb:verb, l:left }); }
        return { t:9, v:a, verb:verb, l:left, r:parseEx(parseNoun()) };
}

function parseEx(node) {
        if (node == null) { return null; }
        if (at(ADVERB)) { return parseAdverb(null, node); }
        if (node.t == 8 && !node.r) {
                var p = at(OPEN_P); var x = parseNoun();
                node.r = parseEx((p && x.t == 8) ? k(13, x) : x); node.sticky = null;
        }
        if (atNoun() && !at(IOVERB)) {
                var x = parseNoun();
                if (x.t == 7 && x.v in infix) { return asVerb(".", x, k(3, [node, parseEx(parseNoun())])); }
                if (at(ADVERB)) { return parseAdverb(node, x); }
                return asVerb("@", node, parseEx(x));
        }
        if (at(VERB) || at(IOVERB)) {
                var x = parseNoun();
                if (x.forcemonad) { node.r = parseEx(x); return node; }
                if (at(ADVERB)) { return parseAdverb(node, x); }
                x.l = node; x.r = parseEx(parseNoun()); node = x;
        }
        return node;
}

function parse(str) {
        begin(" "+str); var r = parseList(null, false); if (done()) { return r; }
        throw new Error("unexpected character '"+text[0]+"'");
}

////////////////////////////////////
//
//   Prettyprinter
//
////////////////////////////////////

function format(k, indent, symbol) {
        if (typeof indent == "number") { indent = ""; } if (k == null) { return ""; }
        function indented(k) { return format(k, indent+" "); };
        if (k instanceof Array) { return k.map(format).join(";"); }
        if (k.sticky) { var s=k.sticky; k.sticky=null; var r=format(k); k.sticky=s; return "("+r+")"; }
        if (k.t == 0) {
                return k.v==1/0?"0w":k.v==-1/0?"-0w":na(k)?"0N":
                ""+(k.v % 1 === 0 ? k.v : Math.round(k.v * 10000) / 10000);
        }
        if (k.t == 1) { return ktos(k,true); }
        if (k.t == 2) { return (symbol==1?"":"`")+k.v; }
        if (k.t == 3) {
                if (len(k) <  1) { return "()"; }
                if (len(k) == 1) { return ","+format(k.v[0]); }
                var same = true; var sublist = false; indent = indent || "";
                for(var z=0;z<len(k);z++) { same &= k.v[z].t == k.v[0].t; sublist |= k.v[z].t == 3; }
                if (sublist) { return "("+k.v.map(indented).join("\n "+indent)+")"; }
                if (same & k.v[0].t == 1) { return ktos(k, true); }
                if (same & k.v[0].t <  3) { return k.v.map(format).join(k.v[0].t == 2 ? "" : " "); }
                return "("+k.v.map(format).join(";")+")" ;
        }
        if (k.t == 4) {
                if (len(k.k)<1 || k.k.v[0].t != 2)
                { var t=format(k.k); if (len(k.k)==1) { t="("+t+")"; } return t+"!"+format(k.v); }
                return "["+kzip(k.k,k.v,function(x,y){return x.v+":"+format(y);}).v.join(";")+"]";
        }
        if (k.t == 5) {
                return "{"+(k.args.length?"["+k.args.join(";")+"]":"")+format(k.v)+"}" +
                                (k.curry ? "["+format(k.args.map(function(x,i) { return k.curry[i]; }))+"]" : "");
        }
        if (k.t ==  6) { return k.v+"::"+format(k.r); }
        if (k.t ==  7) { return k.v+(k.r?(k.global?"::":":")+format(k.r):""); }
        if (k.t ==  8) {
                if (k.curry) { return k.v+"["+format(k.curry)+"]"+format(k.r); }
                var left = (k.l?format(k.l):""); if (k.l && k.l.l) { left = "("+left+")"; }
                return left+k.v+(k.r?format(k.r):"");
        }
        if (k.t ==  9) { return (k.l?format(k.l)+" ":"")+format(k.verb)+k.v+format(k.r); }
        if (k.t == 11) { return ""; }
        if (k.t == 12) { return "$["+format(k.v)+"]"; }
        if (k.t == 13) { return "("+format(k.v)+")"; }
}

// js natives and k natives:
var natives = {"log":0,"exp":0,"cos":0,"sin":0};
var infix   = {"o":0,"in":0};
function nmonad(n, f) { verbs[n]=[f, am(f), null,null,null,null,null,null]; }
function baseEnv() {
        var env = new Environment(null);
        nmonad("log", function(x) { return k(0, Math.log(n(x).v)) });
        nmonad("exp", function(x) { return k(0, Math.exp(n(x).v)) });
        nmonad("cos", function(x) { return k(0, Math.cos(n(x).v)) });
        nmonad("sin", function(x) { return k(0, Math.sin(n(x).v)) });
        run(parse("prm:{{$[x;,/x,''o'x^/:x;,x]}@$[-8>@x;!x;x]}"), env);
        run(parse("in:{~^y?x}"), env);
        return env;
}

var packimpl   = parse("{+/y*|*\\1,|1_(#y)#x}")[0];
var unpackimpl = parse("{(1_r,,y)-x*r:|y(_%)\\|x}")[0];
var spliceimpl = parse("{,/(*x;$[99<@z;z x 1;z];*|x:(0,y)_x)}")[0];
var winimpl    = parse("{$[0>x;3':0,y,0;y(!0|1+(#y)-x)+\\:!x]}")[0];
var odoimpl    = parse("{+x\\'!*/x}")[0];
var splitimpl  = parse("{1_'(&x=y)_y:x,y}")[0];

// export the public interface:
function setIO(symbol, slot, func) {
        if (!(symbol in verbs)) { verbs[symbol]=[null,null,null,null,null,null]; }
        verbs[symbol][slot] = func;
}

this.version = "0.1";
this.parse = parse;
this.format = format;
this.run = run;
this.Environment = Environment;
this.baseEnv = baseEnv;
this.setIO = setIO;
```