mcode runtime library

```
// mcode runtime library for javascript
Δ.v r,ver='mcode runtime library version 0.08.04.2024'
'' □ ver ;  ⊙.test = {} // test artifacts namespace

r = `// mcode runtime library - generated file
mcode.logn('` + ver + ` loaded from cache');
/* jshint asi:true */
`

r += `
// regenerate mcode cache - called by the IDE regen button`

r += ≜.er `
∇ mcode.regen
    ∇.a sf : pr,r
        mcode.desktop →
            ⁝.done 'regen not currently available on desktop system'
            ⍛ 0
        // pr ← { ω.replace(/^.*\\?\\s/,'') }   // remove prompt
        // pr2 ← { ¯ 1 ↓ '? ' ⊃ ω }             // another way to remove prompt
        mcode.serverAuth ⇋ θ →
            // mcode.serverAuth = pr ( ⍁.c 'server write auth? ' )  // ask in session
            // ask outside of session for better security
            mcode.serverAuth = prompt('server write auth? ')
        // read and exec core source
        □ ≜ ⍁ 'core.mc.txt'
        // read and exec runtime library source
        □ ≜ ⍁ 'rtl.mc.txt'
        // write transpiled code to cache file
        r = mcode.cp.core + mcode.cp.lib
        'server rsp' □.nnl ⍁.'lib/mcode_cache.js' r
        ⁝.done 'regen done'                                    // tell IDE we're done
    □ 'regen'
    ⁝.busy 0    // tell IDE to wait for done promise
    ⁝.tmo sf    // call async 'sf' after busy flag is shown (otherwise prompt supercedes)
`

r += `
// Panel functions - in-browser floating movable windows`

r += ≜.er `
∇ mcode.panel
    ∇ init
```

```
        // one time css class definition
        ⍝.head \`
<!-- mcode.panel css -->
<style>
.mcode-panel {
    font-size: 13px;
    padding: 4px; border: 0; border-radius: 4px; box-shadow: 5px 10px 10px rgba(0,0,0,128);
    position:fixed; z-index:100;
    touch-action:manipulation;
    max-width: 800px; max-height: 800px;
    /* top:10%; left:10%; width:80%; height:80%; */
}
.mcode-panel_button {
    float: right; width: 32px; height: 32px;
    font-family: apl; font-size: 13px;
    border-radius: 4px; border: 0; margin: 2px; padding: 0; text-align: center;
}
.mcode-panel h3                 { font-family: Arial; }
.mcode-panel hr                 { border-color: black; margin-top: 5px; }
.mcode-panel div,pre,textarea   { font-family: apl; margin:0; }
</style>
\`

    init 0

    ∇ handleMoveButton : pid=⍵,svg,icon,moveAr={},moveID=0

        svg =
\`
VEQgU1ZHIDEuMS8vRU4iICJodHRwOi8vd3d3LnczLm9yZy9HcmFwaGljcy9TVkcvMS4xL0RURC9zdmcxMS5kdGQiPjxzdmcgeG1sbnM9Imh0dHA6Ly93d3cu
dzMub3JnLzIwMDAvc3ZnIiB4bWxuczp4bGluaz0iaHR0cDovL3d3dy53My5vcmcvMTk5OS94bGluayIgdmVyc2lvbj0iMS4xIiB3aWR0aD0iMjQiIGhlaWdo
dD0iMjQiIHZpZXdCb3g9IjAgMCAyNCAyNCI+PHBhdGggZD0iTTEzLDExSDE4TDE2LjUsOS41TDE3LjkxLDguMDhMMjEuODQsMTJMMTcuOTIsMTUuOTJMMTYu
NSwxNC41TDE4LDEzSDEzVjE4TDE0LjUsMTYuNUwxNS45MiwxNy45MkwxMi4xNiwyMS44NEw4LjA4LDE3LjkxTDkuNSwxNi41TDExLDE4VjEzSDZMNy41LDE0
LjVMNi4wOCwxNS45MkwyLjE2LDEyTDYuMDgsOC4wOEw3LjUsOS41TDYsMTFIMTFWNkw5LjUsNy41TDguMDgsNi4wOEwxMiwyLjE2TDE1LjkyLDYuMDhMMTQu
NSw3LjVMMTMsNlYxMVoiIC8+PC9zdmc+\`

        icon = \`<img src="\${svg}" draggable="false" style="width:18px;height:18px;vertical-align:bottom;">\`

        ∇ reset : pid=⍵,t
            t = ⍝ pid ; !t → ⎕.↑ 'mcode.panel: no panel element ' + pid
            moveAr[pid] = {tgt:t,x:0,y:0}

        ∇ doMove : m=⍺,e=⍵,dx,dy
            dx = m.x - e.clientX ; dy = m.y - e.clientY;
            m.x = e.clientX ; m.y = e.clientY;
            m.tgt.style.left = ( m.tgt.offsetLeft - dx ) + 'px'
```

```
        m.tgt.style.top  = ( m.tgt.offsetTop - dy ) + 'px'

    ∇ tmove : e=α,m,f           // e is touchmove event with touch move list
        !moveID → ⍒ 0
        m = moveAr[moveID]
        f ⌺ e.touches : m doMove f

    ∇ mmove : e=α,m             // e is mousemove event
        !moveID → ⍒ 0
        m = moveAr[moveID] ; m doMove e

    ∇ pend
        moveID = 0

    ∇ pstart : e=α,id,m         // α is pointerdown event from DOM
        id = e.currentTarget.getAttribute('move-btn')
        moveID = id
        m = moveAr[id] ; m.x = e.clientX ; m.y = e.clientY
        ⍥.ael [ 'touchmove' tmove 0 ] ; ⍥.ael [ 'touchend' pend 0 ]
        ⍥.ael [ 'mousemove' mmove 0 ] ; ⍥.ael [ 'mouseup'  pend 0 ]

    ∇ initMoveButton : pid=ω,id,el
        // pid is panel el id to be moved (using top and left)
        id = pid+'_move'                        // button user presses to cause move
        el = ⍥ id ; !el → ⍞.↑ 'mcode.panel: no move element ' + id
        reset pid

        el ⍥.attr [ 'move-btn' pid ]            // for pstart moveID
        el.style.cursor = 'move'
        el.style.touchAction = 'none'           // nb. stops browser from scrolling while button is touched
        el ⍥ icon
        el ⍥.ael [ 'pointerdown' pstart 1 ]

    initMoveButton pid

∇ setColors : id=ω,el
    ⍙.v [bg,btn,clr] = α
    el = ⍥ id ; !el → ⍒ 0
    el ⍥.clr clr ; el ⍥.bg bg
    id+'_hr'    ⍥.brd clr
    id+'_hdg'   ⍥.clr clr
    id+'_move'  ⍥.bg  btn
    id+'_close' ⍥.bg  btn

∇ show : id=ω,el
    α ⍪ θ → α = ['80%']
```

```
    Δ.v [w=θ,h='400px',x='100px',y='100px'] = α
    el = ∘ id ; !el → ∇ 0
    el ∘.dsp 'block'
    w ↭ θ → ∇ el // show only
    el ∘.top y  ; el ∘.left x ; el ∘.w w
    h = h Δ # ; el ∘.h h
    id+'_body' ∘.h h - 45
    ∇ id

∇ close : id=ω,el
    !id → id = α      // call from mcode or DOM
    id ∘.dsp 'none' ; ∇ id

∇ remove : id=ω,el
    !id → id = α      // call from mcode or DOM
    id ∘.rm 0 ; ∇ 0

∇ create : id=ω
    α ↭ θ → ⁚.↑ 'mcode.panel.create: heading and body required'
    Δ.v [hdg='',body='',bg='888',btn='777',clr='000',template=''] = α
    template ↭ '' →
        // useful unicode:  help 10067
```

```
mcode.panel 0
`

⌽ `
∇ ⊙.test.panel : p,d
    ⍞ 'test.panel'
    p = 'testPanel1'
    d = \`
Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
\`
    [ 'test panel 1' d 375 486 'ccc' ] mcode.panel.create p
    mcode.panel.show p
// ⊙.test.panel 0
`

r += '// mcode Language Syntax Coloring'
r += ⌽.er `
∇ mcode.syntaxColor

    ∇ init
        ⍜.head \`
<!-- mcode.syntaxColor css -->
<style>
.mcode-lang-stmts               { color: #f94; }
.mcode-lang-prims               { color: #fc4; }
.mcode-lang-vars                { color: #0ec; }
.mcode-lang-fns                 { color: #ff4; }
.mcode-lang-builtins            { color: #0fa; }
.mcode-lang-oper                { color: #ee4; }
</style>
\`
    init 0

    printCss = \`
<!-- mcode.syntaxColor printCss -->
<style>
.mcode-lang-stmts               { color: #d70; }
.mcode-lang-prims               { color: #c0c; }
.mcode-lang-vars                { color: #0c9; }
.mcode-lang-fns                 { color: #880; }
.mcode-lang-builtins            { color: #0c6; }
.mcode-lang-oper                { color: #a06; }
</style>
\`
```

```
∇ colorize : e0,r                          // colorize mcode in ω with HTML

    // ⎕ 'mcode.syntaxColor.colorize'
    // '  mcode' ⎕.j ω
    mcode.setLang()
    // '  mcode.lang' ⎕ mcode.lang

    e0 = '' ⊃ '?+*!^~|&'                   // escapes needed for regexp

    ∇ escapes                              // escape HTML in regexs and source
        ⍢ />/g ≠.'>' /</g ≠.'<' /\\//g ≠.'&sol;' ω
    // 'escapes' ⎕.j escapes '1 < 2/3 > 5'  // test

    ∇ spans : s={}
        s.z0 = '<span class="mcode-lang-' ; s.z1 = '">' ; s.z2 = '</span>'
        ⍢ '' ≠.s.z2 '">' ≠.s.z1 '≠.s.z0 ω

    ∇ apply : c,e,s={}                     // apply lang regexp to mcode

        ∇ makeRe : b,d,m
            m = mcode.lang[ω]
            b ⍥ e0                         // process each regexp  nb. 2x \\ escapes
                s.b = '\\\\'+b ; m = b ≠.s.b m
            d = /\\s/g ≠.'|' m             // insert |
            ω ↭ 'stmts' → d += '|//'       // add stmt comment to regexp
            ⍢ '('+d+')' ⍙.g '/'            // convert to regexp

        c = '$1'
        s.x = α ; s.t = 'X' ≠.s.x c        // create HTML template
        e = makeRe α
        ⍢ e ≠.s.t ω                        // insert template at matches to f

    r = ω
    r = 'stmts'     apply r
    r = 'prims'     apply r
    r = 'vars'      apply r
    r = 'fns'       apply r
    r = 'oprs'      apply r
    r = 'builtins'  apply r
    r = spans 0 escapes r
    ⍢ r

mcode.syntaxColor.colorize = colorize
mcode.syntaxColor.printCss = printCss
```

```
mcode.syntaxColor 0
`

⍝ `
∇ ⊙.test.syntaxColor : c,s,p
    ⎕ '⊙.test.syntaxColor'
    c = \`∇ foo     // test
        ⍝ 1 < 0
    \`
    s = mcode.syntaxColor.colorize c
    s = '<pre>'+s+'</pre>'
    p = 'testPanel2'
    [ 'test panel 2' s 357 468 'ccc' ] mcode.panel.create p
    mcode.panel.show p
// ⊙.test.syntaxColor 0
`

r += ⍝.er `
∇ mcode.showGuide : r,p='mcodeGuide'
    mcode.panel.show p → ⍝ 0
    r = mcode.guide 0
    r = mcode.syntaxColor.colorize r
    r = '<pre>'+r+'</pre>'
    [ 'mcode guide' r '357' '579' 'eee' ] mcode.panel.create p
    mcode.panel.show p
`

r += ⍝.er `
∇ mcode.printDoc : p
    p = \`
<html lang="en">
<head>
<title>mcode guide</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Content-Language" content="en">
<style>
    body {
        background-color: #fff; color: #000;
        font-family:Arial, Helvetica, sans-serif;
        max-width: 1000px;
        max-height: 1000px;
        /*overflow: hidden;*/
    }
    @font-face { font-family:'apl'; src: local('apl386 Unicode'),
        url('apl386.woff') format('woff'); }
    pre {
```

```
        font-family: apl, monospace;
    }
</style>
\`
    p += mcode.syntaxColor.printCss

    ∇ comp
        p += '</head><body><h3>' + ⍺ + '</h3>\\n'
        ⍵ = mcode.syntaxColor.colorize ⍵
        p += '<pre>' + ⍵ + '</pre></body></html>'

        ⍥.docwr p

    ∇.a rf : d
        ⍵ ⇆ 'guide' → d = mcode.guide 0     // guide from transpiler tables
        ◇ d = ⍁ ⍵                           // document file from server
        ⍺ comp d

    // 'mcode guide' rf 'guide'
    // 'mcode primer' rf 'primer.mc.txt'
    // 'mcode core' rf 'core.mc.txt'
    'mcode runtime library' rf 'rtl.mc.txt'

    // NIU - use cut/paste into MS Word instead
    /*
    ∇.a wf
        mcode.serverAuth ⇆ ⍬ →
            mcode.serverAuth = prompt('server write auth? ')
        'server rsp' ⍞.nnl ⍁.'downloads/guide.html' p
        ⍠.done 'file write done'
        ⍛ 0
    ⍠.busy 0
    ⍠.tmo wf
    */
\`

// mcode keyboard map
⍙.v vik={}
vik.kbd = `
~ ⍀|! ⌶|@ ⍒|# ⍋|$ ⍏|% ⌽|^ ⍉|& ⊖|* ⍟|( ⍱|) ⍷|_ ⌸|+ ⌹
\` ◇|1 ¨|2 ¯|3 ⍁|4 ≤|5 ≈|6 ≥|7 ⍂|8 ≠|9 ⇆|0 ⍬|- ×|= ÷
--
Q ⍠|W ⍵|E ⍷|R ⍴|T ⍨|Y ⍋|U ⍒|I ⍳|O ⍤|P ⍣|{ ⍇|} ⍬|| ⍲
q ⍀|w ⍵|e ⍷|r ⍴|t ~|y ↑|u ↓|i ⍳|o ○|p ⍱|[ ←|] →|\\ ⊢
--
A ⍺|S ⍒|D ⍫|F ⍁|G ⍛|H ⍙|J ⍤|K ⍈|L ⍞|: ≡|" ≠
```

```
a ⍺|s ⌈|d ⌊|f ⍥|g ∇|h ∆|j ∘|k ⊙|l ⍞|; ⍫|' ⍣
--
Z ⊆|X ⊇|C «|V »|B ⊥|N ⍑|M ‖|< ⍪|> ⍂|? ⍰
z ⊂|x ⊃|c ∩|v ∪|b ⊥|n ⍑|m ⍭|, ⍋|. ⍒|/ ≠
`

∇ vik.generate : a,b,c,d,e,f,k={},l=[],s={}
    // 'kbd' ⍞ vik.kbd
    ∇ ins                                           // insert fixed keys
        ⍺ ⇜ 'a' → ⍛ ⍵ ↓ 'tab'
        ⍺ ⇜ 'z' → ⍛ ⍵ ↓ 'sft'
    ∇ app                                           // append fixed keys
        ⍺ ⇜ "'" → ⍛ ⍵ ↓ 'rtn'
        ⍺ ⇜ '/' → ⍛ ( ⍵ ↓ 'bks' ) ↓ 'del'
    a = /\s/g ≠.'' ¨ '\n' ⊃ ¨ '--' ⊃ vik.kbd        // split kbd rows and remove lf and spaces
    b ⌺ a                                           // each kybd row
        c = '|' ⊃ ¨ [ 1 2 ] ⌷ b                     // remove non-data
        i = 0 ; e = []                              // reset
        d ⌺ c[1]                                    // each key
            f = d[0]                                // base key
            k[f] = c[0][i++][0]                     // set shifted key
            f ins e ; e ↓ f ; f app e               // set layout array
        d ⌺ ⍪ c : s[d[0]] = d[1]                    // set each key symbol
        l ↓ e                                       // push layout row
    l ↓ [ 'sym' 'spc' 'pup' 'pdn' 'mov' 'cls' ]     // bottom row layout
    // 'shift map' ⍞ k ; 'layout arr' ⍞.j l ; 'mcode map' ⍞ s
    mcode.vik.shift   = ⍣.j k
    mcode.vik.layout  = ⍣.j l
    mcode.vik.symbols = ⍣.j s
    ⍛ 0
mcode.vik = {}
vik.generate 0

r += `
// Virtual Input Keyboard (VIK) - generated data
    mcode.vik = {}
    mcode.vik.shift   = ` + mcode.vik.shift   + `
    mcode.vik.layout  = ` + mcode.vik.layout  + `
    mcode.vik.symbols = ` + mcode.vik.symbols + `
`

r += `// end library

`

⍙.v core = mcode.cp.core            // save generated core
```

```
Δ.v lib  = r                        // save generated lib
mcode.cp = {}                       // clear context
mcode.cp.core = core
mcode.cp.lib = lib

▽ '  library loaded'
```