JOD Best Practices Lab

January 29, 2019

1 JOD Best Practices Lab

1.0.1 Introduction

Software tools are like loaded guns: powerful weapons for slaying your coding demons but also dangerous when used improperly. Have you ever shot yourself in the foot? I know I'm missing a few toes and I bet you are as well.

This lab outlines a number of "best practices" or guidelines for using JOD. I've learned the hard way; if you take my advice to heart you *might* be spared!

```
In [1]: load 'general/jod'

        NB. use portable box drawing characters
        portchars ''
```

1.0.2 Test for JOD user defined folders

WARNING: THIS LAB REQUIRES A NUMBER OF USER DEFINED JOD FOLDERS.

This lab requires a number of JOD folders to run. The next step tests for the existence of these folders.

If (`TestJODDirectories`) lists any undefined JOD directories configure these directories before running this lab. Instructions on how to do this can be found in (`jod.pdf`). Install the joddocument addon to get (`jod.pdf`).

In [2]: TestJODDirectories_ijod_=:3 : 0

```
NB.*TestJODDirectories v- test user configured JOD directories.
NB.
NB. This  verb  checks  that  required  JOD  lab directories  are
NB. defined. "Defined" does not mean the  directories  exist only
NB. that (jpath) expands to something other than its default.
NB.
NB. monad:  clMsg =. TestJODDirectories uuIgnore

NB. when a relative directory does not exist (jpath) echoes its argument
jodudirs =. '~'&,&.> ;:'JOD JODDUMPS JODSOURCE JODTEST'
jodudefs =. jpath&.> jodudirs  NB. !(*)=. jpath
mask =. jodudirs = jodudefs
if. 1 e. mask do.
  > 'missing JOD folders - define before running JOD labs' ; mask # jodudirs
else.
  > 'all JOD folders are defined' ; jodudefs
end.
)

TestJODDirectories 0
```

```
all JOD folders are defined
c:/jod
c:/jod/joddumps
c:/jodtest/labtesting
c:/jodtest/test
```

1.0.3 Lab cleanup

WARNING: DICTIONARIES CREATED BY PRIOR RUNS OF LAB THIS WILL BE DELETED IN THE NEXT STEP.

When the Best Practices lab runs it creates a number of dictionaries in the (~JODSOURCE) directory. The next step will remove any of these dictionaries. **IF YOU CARE ABOUT THESE DICTIONARIES STOP NOW.**

```
In [3]: RemoveLabBestDictionaries_ijod_=: 3 : 0
        if. IFWIN do.
          shell 'rd /s /q "',(jpath '~JODSOURCE'),'"'
          smoutput 'Lab temporary best practices (win) dictionaries erased'
        elseif. IFUNIX do.
          NB. avoid blanks in Mac and Linux paths
          shell 'rm -rf ',jpath '~JODSOURCE'
          smoutput 'Lab temporary best practices (mac/linux) dictionaries erased'
        elseif.do.
          smoutput 'Erase any previous temporary best practices dictionaries manually.'
        end.
        )
```

Remove Best Practices lab dictionaries.

```
In [4]: NB. clear any previously opened dictionaries in master file
        dpset 'RESETME'

        NB. unregister dictionaries - IGNORE ERRORS
        3 regd&> ;:'bpcopy bptest' [ 3 od ''

        NB. delete dictionary files
        RemoveLabBestDictionaries 0
```

```
Lab temporary best practices (win) dictionaries erased
```

1.0.4 JOD does not belong in the J tree

My first and most important bit of advice is simply:
NEVER NEVER NEVER store your JOD dictionaries in J install directories!
I would also avoid any OS managed directories like

```
c:\Progam Files\..
```

Create a JOD master dictionary directory root that is completely independent of J. It's also a good idea to define a subdirectory structure that mirrors J's versions. JOD creates binary `jfiles`. These files are fairly stable but binaries can change when J changes.

```
In [5]: NB. create a master JOD directory root outside of J's directories.
        NB. This creation depends on a configured directory (~JODSOURCE).
        smoutput newd 'bptest';(jpath '~JODSOURCE/bptest');'best practices test dictionary'

        NB. This is an example - use another root for your dictionaries.

+-+--------------------+------+----------------------------+
|1|dictionary created ->|bptest|c:/jodtest/labtesting/bptest/|
+-+--------------------+------+----------------------------+
```

1.0.5 Load lab dictionary from dump script

To illustrate key features of JOD we need a nonempty dictionary. This next step loads the (`bptest`) dictionary from a dump script distributed with JOD.

```
In [6]: NB. open new dictionary
        od 'bptest' [ 3 od ''

        NB. load from example dump script
        0!:0 <jpath '~addons/general/jod/jodlabs/labdump.ijs'
```

```
        NB. regenerate all references - show last three messages
        _3 {. 0 globs&> }. revo''
```

```
+-+------------------+------+
|1|1 word(s) put in ->|bptest|
+-+------------------+------+
+-+--------------------+------+
|1|35 word(s) put in ->|bptest|
+-+--------------------+------+
+-+--------------------------------+------+
|1|36 word explanation(s) put in ->|bptest|
+-+--------------------------------+------+
+-+----------------------------+------+
|1|2 word document(s) put in ->|bptest|
+-+----------------------------+------+
+-+--------------------------+------+
|1|group <bstats> put in -> |bptest|
+-+--------------------------+------+
|1|group <sunmoon> put in ->|bptest|
+-+--------------------------+------+
NB. end-of-JOD-dump-file regenerate cross references with:  0 globs&> }. revo ''
+-+-----------------------------------+------+
|1|<var> references put in ->          |bptest|
+-+-----------------------------------+------+
|1|<yeardates> references put in ->    |bptest|
+-+-----------------------------------+------+
|1|<NORISESET> is a noun - no references|      |
+-+-----------------------------------+------+
```

1.0.6 Backup backup backup

A wise man once said, *"You're either backed up or f&%ked up!"*

Care to go over the options again?

It is literally a snap to make a backup with JOD. So backup often!

```
In [7]: NB. open the best practice dictionary
        od 'bptest' [ 3 od ''

        NB. back it up
        smoutput packd 'bptest'

        NB. restd recovers the most current backup
        restd 'bptest'
```

```
+-+--------------------+------+-+
|1|dictionary packed ->|bptest|0|
+-+--------------------+------+-+
+-+----------------------+------+-+
|1|dictionary restored ->|bptest|0|
+-+----------------------+------+-+
```

1.0.7 Take a script dump

(packd) backs up binary `jfiles` but it's also a good idea to "dump" your dictionaries as plain text. JOD can dump all open dictionaries as a single J script. Script dumps are the most stable way to store J dictionaries. The `jodsource` addon distributes all JOD source code in this form.

```
In [8]: NB. dump only (bptest)
        od 'bptest' [ 3 od ''

        NB. (make) creates a dictionary dump in the dump subdirectory
        bptestdump=: showpass_ajod_ make ''
```

```
+-+--------------------------+-----------------------------------------------+
|1|object(s) on path dumped ->|c:/jodtest/labtesting/bptest/dump/bptest.ijs|
+-+--------------------------+-----------------------------------------------+
```

1.0.8 Some uses of dump scripts

JOD dump scripts can be used to reload, copy and merge dictionaries.

```
In [9]: NB. reload bptest
        od 'bptest' [ 3 od ''
        0!:0 {:bptestdump

+-+------------------+------+
|1|1 word(s) put in ->|bptest|
+-+------------------+------+
+-+--------------------+------+
|1|35 word(s) put in ->|bptest|
+-+--------------------+------+
+-+--------------------------------+------+
|1|36 word explanation(s) put in ->|bptest|
+-+--------------------------------+------+
+-+----------------------------+------+
|1|2 word document(s) put in ->|bptest|
+-+----------------------------+------+
+-+-------------------------+------+
|1|group <bstats> put in -> |bptest|
+-+-------------------------+------+
|1|group <sunmoon> put in ->|bptest|
+-+-------------------------+------+
+-+------------------------------+------+
|1|dictionary document updated ->|bptest|
+-+------------------------------+------+
NB. end-of-JOD-dump-file regenerate cross references with:  0 globs&> }. revo ''
```

Copy a dictionary.

```
In [10]: NB. copy/merge bptest dictionary
         smoutput newd 'bpcopy';jpath '~JODSOURCE/bpcopy'
         od 'bpcopy' [ 3 od ''
```

```
0!:0 {:bptestdump


NB. clear path
dpset 'CLEARPATH'
```

```
+-+--------------------+------+----------------------------+
|1|dictionary created ->|bpcopy|c:/jodtest/labtesting/bpcopy/|
+-+--------------------+------+----------------------------+
+-+-------------------+------+
|1|1 word(s) put in ->|bpcopy|
+-+-------------------+------+
+-+--------------------+------+
|1|35 word(s) put in ->|bpcopy|
+-+--------------------+------+
+-+--------------------------------+------+
|1|36 word explanation(s) put in ->|bpcopy|
+-+--------------------------------+------+
+-+----------------------------+------+
|1|2 word document(s) put in ->|bpcopy|
+-+----------------------------+------+
+-+-------------------------+------+
|1|group <bstats> put in -> |bpcopy|
+-+-------------------------+------+
|1|group <sunmoon> put in ->|bpcopy|
+-+-------------------------+------+
+-+-----------------------------+------+
|1|dictionary document updated ->|bpcopy|
+-+-----------------------------+------+
NB. end-of-JOD-dump-file regenerate cross references with:  0 globs&> }. revo ''
+-+---------------+------+
|1|path cleared ->|bpcopy|
+-+---------------+------+
```

1.0.9 Dump scripts are the best way to share and version control dictionaries

Dump scripts can be used to share and version control dictionaries. See this Git repository for examples.

Example JOD Dump Script Repository

1.0.10 Make a master re-register script

JOD only sees the dictionaries registered in the `jmaster.ijf` file so maintaining a list of registered dictionaries is a good idea. JOD can generate a re-register script that you can edit.

Generate a re-register script and put it in your main JOD dictionary directory root.

```
In [11]: NB. generate re-register script
         rereg=: showpass_ajod_ ;{: 5 od ''

         NB. save it in the master root
         rereg write_ajod_ jpath '~JODSOURCE/jodregister.ijs'
```

```
NB. JOD registrations: 29 Jan 2019 18:47:07
NB. require 'general/jod'
3 regd&> }. od'' [ 3 od ''
regd 'utils';'c:/jod/utils/'
regd 'jod';'c:/jod/jod/'
regd 'joddev';'c:/jod/joddev/'
regd 'play';'c:/jod/play/'
regd 'imex';'c:/jod/imex/'
regd 'smugpyter';'c:/jod/smugpyter/'
regd 'rdi';'c:/jod/rdi/'
regd 'smugdev';'c:/jod/smugdev/'
regd 'image';'c:/jod/image/'
regd 'smug';'c:/jod/smug/'
regd 'docs';'c:/jod/docs/'
regd 'simplot';'c:/jod/simplot/'
regd 'rdidev';'c:/jod/rdidev/'
regd 'simplotdev';'c:/jod/simplotdev/'
regd 'utilsload';'c:/jodtest/labtesting/utilsload/'
```

```
regd 'jodload';'c:/jodtest/labtesting/jodload/'
regd 'joddevload';'c:/jodtest/labtesting/joddevload/'
regd 'bptest';'c:/jodtest/labtesting/bptest/'
regd 'bpcopy';'c:/jodtest/labtesting/bpcopy/'
```

1.0.11 Set library dictionaries to READONLY

Open JOD dictionaries define a search path. The first dictionary on the path is the only dictionary that can be changed. It is called the "put" dictionary. Even though nonput dictionaries cannot be changed by JOD it's a good idea to set them READONLY because:

READONLY dictionaries can be accessed by any number of JOD tasks. READWRITE dictionaries can only be accessed by one task. Keeping libraries READONLY prevents accidental put's as you open and close dictionaries.

```
In [12]: NB. make bptest READONLY
         od 'bptest' [ 3 od ''
         dpset 'READONLY'

         NB. bpcopy is now the put dictionary
         od ;:'bpcopy bptest' [ 3 od ''

         NB. first group/suite sets path
         grp 'agroup'; ;:'datecheck yeardates today sunriseset0'

         NB. note dictionary path
         did ~ 0
```

```
+-+-------------------------------------------------------------+
|1|+------+--+-----+-----+-------+-------+------+--------------+|
| ||      |--|Words|Tests|Groups*|Suites*|Macros|Path*         ||
| |+------+--+-----+-----+-------+-------+------+--------------+|
| ||bpcopy|rw|36   |0    |3      |0      |0     |/bpcopy/bptest||
| |+------+--+-----+-----+-------+-------+------+--------------+|
| ||bptest|ro|36   |0    |2      |0      |0     |/bptest       ||
| |+------+--+-----+-----+-------+-------+------+--------------+|
+-+-------------------------------------------------------------+
```

1.0.12 Keep references updated

JOD stores word references. References enable many useful operations. References allow (getrx) to load words that call other words in new contexts.

```
In [13]: NB. only put dictionary references need updating - show last five messages
         _5 {. 0 globs&> }. revo ''

+-+--------------------------------+------+
|1|<tabit> references put in ->    |bpcopy|
+-+--------------------------------+------+
|1|<tan> references put in ->      |bpcopy|
+-+--------------------------------+------+
|1|<today> references put in ->    |bpcopy|
+-+--------------------------------+------+
|1|<var> references put in ->      |bpcopy|
+-+--------------------------------+------+
|1|<yeardates> references put in ->|bpcopy|
+-+--------------------------------+------+
```

1.0.13 Document dictionary objects

Documentation is a long standing sore point for programmers. Most of them hate it. Some claim it's unnecessary and even distracting. Others put in half baked efforts. In my opinion this "isn't even wrong!" Good documentation elevates code. In Knuth's opinion it separates "literate programming" from the alternative - "illiterate programming."

JOD provides a number of easy ways to document code. You can enter a single sentence or a large dissertation. I would recommend the former. See JOD documentation for more documentation options.

```
In [14]: NB. for new words try a single line of documentation.
         afterlaststr=:] }.~ #@[ + 1&(i:~)@([ E. ])
         put 'afterlaststr'

         NB. insert sentence
         0 8 put 'afterlaststr';'retains string (y) after last occurrence of (x)'
```

```
+-+------------------------------+------+
|1|1 word explanation(s) put in ->|bpcopy|
+-+------------------------------+------+
```

JOD uses documentation in new contexts.

```
In [15]: NB. for tacits the docment sentence is fetched
         smoutput disp 'afterlaststr'

         NB. briefly describe sunmoon group  - the payback for entering those sentences
         hlpnl }. grp 'sunmoon'

NB. retains string (y) after last occurrence of (x)
afterlaststr=:] }.~ #@[ + 1&(i:~)@([ E. ])
+----------+-------------------------------------------------------------------------+
|NORISESET |indicates sun never rises or sets in (sunriseset0) and (sunriseset1) results|
|arctan    |arc tangent                                                              |
|calmoons  |calendar dates of new and full moons                                     |
|cos       |cosine radians                                                           |
|datecheck |checks dates in YYYY MM DD format                                        |
|fromjulian|converts Julian day numbers to dates, converse (tojulian)                |
|moons     |times of new and full moons for n calendar years                         |
|round     |round y to nearest x (e.g. 1000 round 12345)                             |
|sin       |sine radians                                                             |
|sunriseset0|computes sun rise and set times - see long documentation                 |
|sunriseset1|computes sun rise and set times - see long documentation                 |
|tabit     |promotes only atoms and lists to tables                                  |
|tan       |tan radians                                                              |
|today     |returns todays date                                                      |
|yeardates |returns all valid dates for n calendar years                             |
+----------+-------------------------------------------------------------------------+
```

An example of long documentation.

```
In [16]: NB. long document
         0 9 disp 'sunriseset0'

*sunriseset0 v-- sunrise and sunset times.

This  verb has been adapted from a BASIC program submitted by
Robin  G.  Stuart  Sky  &  Telescope's  shortest  sunrise/set
program  contest. Winning  entries were listed  in the  March
1995 Astronomical Computing column.

The  J version of this algorithm has been vectorized.  It can
compute any number of sunrise and sunset times in one call.

NB. verbatim:

The (y) argument is a 5*n floating point table where:

 0{ is latitude in degrees with northern latitudes positive.
 1{ is longitude in degrees with western longitudes negative.
 2{ is western time zones expressed as positive whole hours.
 3{ is the month number.
 4{ is the day number.

The result is a numeric table with four rows.  To handle the cases
when the sun never rises or sets the first two elements of the
corresponding result columns n are:

 0{ is NORISESET an invalid hour indicating no rise or set
 1{ is 0 when the sun never rises
 1{ is 1 when the sun never sets

Warning: this algorithm breaks for latitudes close
to the South pole.

The original BASIC code has been slightly modified
```

to use control structures in place of GOTO's and line numbers.

Adapted from:

```
' Sunrise/set by R. G. Stuart,  Mexico City, Mexico
PI = 3.14159265#: DR = PI / 180: RD = 1 / DR
INPUT "Lat, Long (deg)"; B5, L5
INPUT "Time zone (hrs)"; H
B5 = DR * B5
INPUT "Month, day"; M, D
N = INT(275 * M / 9) - 2 * INT((M + 9) / 12) + D - 30
L0 = 4.8771 + .0172 * (N + .5 - L5 / 360)
C = .03342 * SIN(L0 + 1.345)
C2 = RD * (ATN(TAN(L0 + C)) - ATN(.9175 * TAN(L0 + C)) - C)
SD = .3978 * SIN(L0 + C): CD = SQR(1 - SD * SD)
SC = (SD * SIN(B5) + .0145) / (COS(B5) * CD)
IF ABS(SC) <= 1 THEN
  C3 = RD * ATN(SC / SQR(1 - SC * SC))
  R1 = 6 - H - (L5 + C2 + C3) / 15
  HR = INT(R1): MR = INT((R1 - HR) * 60)
  PRINT USING "Sunrise at ##:##"; HR; MR
  S1 = 18 - H - (L5 + C2 - C3) / 15
  HS = INT(S1): MS = INT((S1 - HS) * 60)
  PRINT USING "Sunset at ##:##"; HS; MS
ELSEIF SC > 1 THEN
  PRINT "Sun up all day"
ELSEIF SC < -1 THEN
  PRINT "Sun down all day"
END IF
END
```

```
monad: ntRiseset =. sunriseset0 flBLHMD
```

```
NB. rise and set times at Dog Lake today (daylight savings)
td=. (44 + 19%60),(- 76 + 21%60), 4 , }. today 0
sunriseset0 td

NB. rise and set times on June 30 on Greenwich meridian
t0=.   0 0 0 6 30   NB. equator
t1=.  49 0 0 6 30   NB. north - lat of western US/Canada border
t2=. _47 0 0 6 30   NB. south - southern Chile and Argentina
t3=.  75 0 0 6 30   NB. far north (sun always up)
t4=. _75 0 0 6 30   NB. far south (sun always down)

sunriseset0 t0

sunriseset0 t0 , t1 , t2 , t3 ,: t4

NB. times on equator for March 21 for all 1 hour time zones
sunriseset0  0 0 ,"1 (,.i. 24) ,"1 ] 3 21

NB. times for calendar year 1995 on the Greenwich meridian
md95=.  47 0 0 ,"1 }."1 yeardates 1995
rs095=. sunriseset0 md95
```

1.0.14 Define your own JOD shortcuts

JOD words can be used within arbitrary J programs. If you don't find a JOD primitive that meets your needs do a little programming. There are many examples of JOD programming in JOD's source code. Install the `jodsource` addon to get JOD source code.

```
In [17]: NB. examples of using JOD words to define new facilities
```

```
        NB. describe a JOD group
        hg_ijod_=: [: hlpnl [: }. grp

        NB. re-reference put dictionary show any errors
        reref_ijod_=: 3 : '(n,.s) #~ -.;0{"1 s=.0 globs&>n=.}.revo'''' [ y'

        NB. show words referenced by words in a group that are not in the group
        jodg_ijod_=: 'agroup'
        nx_ijod_=: 3 : '(allrefs  }. gn) -. gn=. grp jodg'

        NB. missing from (agroup)
        nx ''
```

```
+---------+------+---+---+-----+---+
|NORISESET|arctan|cos|sin|tabit|tan|
+---------+------+---+---+-----+---+
```

1.0.15 Customize JOD edit facilities

The main JOD edit words (nw) and (ed) can be customized by defining a DOCUMENTCOMMAND script.

Note: Verbs that spawn J editors may not work in Jupyter labs. The following (nw) call opens a JQT or JHS editor in standard J front ends but does nothing here. This is because the J kernal is essentially a barebones `jconsole.exe` process that is is not running JQT.

```
In [18]: NB. define document command script - {~N~} is word name placeholder
         DOCUMENTCOMMAND_ijod_=: 0 : 0
         smoutput pr '{~N~}'
         )

         NB. create a new word - opens an edit window in JQT and JHS
         NB. nw 'bpword'
```

Find the (bpword) edit window and note how DOCUMENTCOMMAND text has been modified and insert. When the script window is saved with CRTL-W DOCUMENTCOMMAND runs.

Run and close the (bpword) edit window.

Edit a word in the dictionary. ***JOD is always editing copies.*** Originals can only be changed with explicit (put) operations.

```
In [19]: NB. load dictionary word into edit window - requires JQT or JHS front ends
         ed 'sunriseset0'

         NB. find the (sunriseset0) edit window and note DOCUMENTCOMMAND

         NB. close edit window
```

```
$$$edit$$$c:/users/jbaker/j64-807-user/temp/sunriseset0.ijs
```

1.0.16 Define JOD project macros

When programming with JOD you typically open a set of dictionaries. Load system scripts and define some handy nouns. This can be done in a project macro script.

```
In [20]: NB. define a project macro - I use the prefix prj for such scripts
         prjsunmoon=: 0 : 0

         NB. standard j scripts
         require 'debug task'

         NB. local script nouns
         jodg_ijod_=: 'sunmoon'
         jods_ijod_=: 'sunmoontests'

         NB. put/xref
         DOCUMENTCOMMAND_z_ =: 'smoutput pr ''{~N~}'''
         )

         NB. store macro
         4 put 'prjsunmoon';JSCRIPT_ajod_;prjsunmoon
```

```
+-+--------------------+------+
|1|1 macro(s) put in ->|bpcopy|
+-+--------------------+------+
```

Once project macros are defined it's easy to configure your J session. Open the required dictionaries and run the macor with (`rm`).

```
In [21]: NB. setup project
         rm 'prjsunmoon'  [ od ;:'bpcopy bptest' [ 3 od ''
```

```
NB. standard j scripts
require 'debug task'
NB. local script nouns
jodg_ijod_=: 'sunmoon'
jods_ijod_=: 'sunmoontests'
NB. put/xref
DOCUMENTCOMMAND_z_ =: 'smoutput pr ''{~N~}'''
```

1.0.17 Edit your `jodprofile.ijs`

When JOD loads the script (`~addons/general/jod/jodprofile.ijs`) is executed. This script can be used to set up JOD the way you want. Note how you can execute project macros when JOD loads with sentences like:

```
rm 'prjsunmoon' [ od ;:'bpcopy bptest'
```

WARNING: store your profile in one of your dictionaries. This file is reset when JAL updates the JOD addon.

```
In [22]: NB. display jodprofile.ijs
         smoutput read_ajod_ jpath'~addons/general/jod/jodprofile.ijs'
```

```
NB.*jodprofile s-- JOD dictionary profile.
NB.
NB. An example JOD profile script. Save this script in
NB.
NB. ~addons/general/jod/
NB.
NB. with the name jodprofile.ijs
NB.
NB. This script  is  executed  after all dictionary  objects have
```

```
NB. been created. It  can  be used  to  set up  your default  JOD
NB. working environment.
NB.
NB. WARNING: Do not dpset 'RESETME' if more than one  JOD task is
NB. active. If only one task is active RESETME's prevent annoying
NB. already open  messages that frequently result from forgetting
NB. to close dictionaries upon exiting J.
NB.
NB. Note to  J developers. A shutdown sentence (a line  of J  the
NB. interpretor  executes  before  terminating)  would  be  very
NB. useful.
NB.
NB. author: John D. Baker
NB. email:  bakerjd99@gmail.com

NB. set white space preservation on
9!:41 [ 1

NB. minimum print precision to show yyyymmdd dates (see jodage)
9!:11 [ 8

NB. set jqt windows console size - automatic for linux/mac/ios
Cwh_j_=: 160 24

NB. do not reset if you are running more than one JOD instance
NB. dpset 'RESETME'

NB. JOD interface locale - (ijod) is a good place for ad hoc JOD addons
coclass 'ijod'

NB. override EDCONSOLE - set (jconsole.exe) editor - use full path
NB. NIMP: currently EDCONSOLE is windows only - see verb (et)
EDCONSOLE_ajodutil_=: '"c:\Program Files\Microsoft VS Code\code.exe"'
NB. EDCONSOLE_ajodutil_=: '"c:\Program Files (x86)\notepad++\notepad++.exe"'
```

```
NB. used by some macros: WHEREAMI=: ;0 { ;:'home work test'
NB. WHEREAMI=: 'home'

NB. (ijod) error/ message text
ERRIJOD00=: 'current group name (jodg_ijod_) not set'
ERRIJOD01=: 'current suite name (jods_ijod_) not set'
OKIJOD00=:  'no matches'

NB. add delete objects from current group or current suite
ag=: 3 : 0
if. wex_ajod_ <'jodg' do. jodg addgrp y else. jderr_ajod_ ERRIJOD00 end.
)
as=: 3 : 0
if. wex_ajod_ <'jods' do. (jods;3) addgrp y else. jderr_ajod_ ERRIJOD01 end.
)
dg=: 3 : 0
if. wex_ajod_ <'jodg' do. jodg delgrp y else. jderr_ajod_ ERRIJOD00 end.
)
ds=: 3 : 0
if. wex_ajod_ <'jods' do. (jods;3) delgrp y else. jderr_ajod_ ERRIJOD01 end.
)

NB. referenced words not in current group
nx=: 3 : 0
if. -.wex_ajod_ <'jodg' do. jderr_ajod_ ERRIJOD00 return. end.
if. badrc_ajod_ gn=. grp jodg do. gn return. end.
(allrefs  }. gn) -. gn
)

NB. words in current group using a word
ug=: 3 : 0
if. -.wex_ajod_ <'jodg' do. jderr_ajod_ ERRIJOD00 return. end.
if. badrc_ajod_ gn=. grp jodg do. gn return. end.
```

```
y usedby }. gn
)

NB. generate current group and save load script
sg=: 3 : 0
if. wex_ajod_ <'jodg' do. mls jodg else. jderr_ajod_ ERRIJOD01 end.
)

NB. top (put dictionary) words, groups, suites in revision order
tw=: revo
tg=: 2&revo
ts=: 3&revo

NB. run tautology as plaintest - does not stop on nonzero results
rt=: 2&rtt

NB. run macro silently - will show explict smoutput
rs=: 1&rm

NB. short help for groups/suites
hg=: [: hlpnl [: }. grp
hs=: 1 hlpnl [: }. 3 grp ]

NB. short help on put objects in revised order from code:
NB.     hr 4  NB. macro
NB.     hr 2  NB. groups
NB.  10 hr 0  NB. last ten words
hr=: 3 : 0
if. badrc_ajod_ w=. y revo '' do. w return. end.
y hlpnl }. w
:
if. badrc_ajod_ w=. y revo '' do. w return. end.
y hlpnl x {. }. w
)
```

```
NB. search short help for string and list matching words
NB.     hss 'see long'   NB. search word short text
NB.  2  hss 'see long'   NB. search group short text
NB.  4  hss 'post'       NB. search macro short text
hss=: 3 : 0
0 hss y
:
if. badrc_ajod_ w=. x dnl ''   do. w return. end.
d=. x hlpnl }. w
if. 0=#w=. 1 >@{ d             do. ok_ajod_ OKIJOD00 return. end.
if. 0=#s=. I. (,:y) +./"1@E. w do. ok_ajod_ OKIJOD00 return. end.
s&{ &.> d
)

NB. single line explanation
NB.    slex 'word'         NB. word
NB.  4 slex 'jodregister'  NB. macro
NB.  1 slex 'thistest'     NB. test
slex=: 3 : 0
0 slex y
:
if. badcl_ajod_ sl=. x disp y do. sl return. end.
(x,8) put y;firstcomment__JODtools sl
)

NB. regenerate put dictionary word cross references
reref=: 3 : 0
if. badrc_ajod_ n=. revo '' do. n return. end.
(n,.s) #~ -.;0{"1 s=.0 globs&> n=.}.n
)

NB. handy cl doc helpers
docscr=: 3 : 'ctl_ajod_ (61;0;0;''NB.'') docct2__UT__JODobj ];._1 LF,y-.CR'
```

```
doctxt=: 3 : 'ctl_ajod_ (61;0;0;'''') docct2__UT__JODobj ];._1 LF,y-.CR'

NB. display noun on screen and return noun value
showpass=:] [ 1!:2&2

NB. portable box drawing characters
portchars=:[: 9!:7 '+++++++++|-'"_ [ ]

NB. windows lucida console box drawing characters
winlcchars=:[: 9!:7 (a.{~16+i.11)"_ [ ]

NB. edit command
DOCUMENTCOMMAND=: 'showpass pr ''{~N~}'''

NB. read & write files
read=:1!:1&(]`<@.(32&>@(3!:0)))
write=:1!:2 ]`<@.(32&>@(3!:0))
readnoun=:3!:2@(1!:1&(]`<@.(32&>@(3!:0))))
writenoun=:([: 3!:1 []) (1!:2 ]`<@.(32&>@(3!:0))) ]

NB. read TAB delimited table files - faster than (readtd) - see long document in (utils)
readtd2=:[: <;._2&> (9{a.) ,&.>~ [: <;._2 [: (] , ((10{a.)"_ = {:) }. (10{a.)"_) (13{a.) -.~ 1!:1&(]`<@.(32&>@(3!:0)))

NB. writes tables as TAB delimited LF terminated text - see long document in (utils)
writetd2=:] (1!:2 ]`<@.(32&>@(3!:0)))~ [: ([: (] , ((10{a.)"_ = {:) }. (10{a.)"_) [: }.@(,@(1&(,"1)@(-.@(*./\."1@(=&' '@]))))

NB. fetch edit text/macros
tt=:] ; gt
mt=:] ; 25"_ ; gt   NB. *.txt
mj=:] ; 21"_ ; gt   NB. *.ijs
md=:] ; 27"_ ; gt   NB. *.markdown
mx=:] ; 22"_ ; gt   NB. *.tex

NB. ~user/temp object text - default j script
```

```
os=: 'ijs'&$: : ([: jpath '~user/temp/' , (' ' -.~ ]) , '.' , ' ' -.~ [)

NB. read text from j user temp directory
jt=:[: read os

NB.  load j script from j user temp
jl=: (0!:0)@jt

NB. number of objects - used by various (utils) macros (sizeput, ageput, ...) if present
NOBS=: 10

NB. where am I used by some macros values 'work' or 'home'
NB. WHEREAMI=: 'work'

NB. dump drive - used by (utils) macro (dumpput) if present
NB. DUMPWINDRV=: 'h:'

NB. dump paths - used by (utils) macro (dumpput) if present
NB. DUMPPATH=: '/jod/joddumps/'
NB. DUMPPRVPATH=: '/jod/jodprvdumps/'

NB. clear dictionaries - used by (utils) macro (clearput) if present
NB. CLEARJDICS=: ;:''

NB. JOD verbs typically run from the base locale
cocurrent 'base'

NB. examples of JOD session start ups - shows
NB. how to open dictionaries and invoke project macros

NB. set up current project (1 suppress IO, 0 or elided display)
NB. 1 rm 'prjsmughacking' [ smoutput od ;:'smugdev smug utils' [ 3 od ''
NB. 1 rm 'prjmep' [ od 'mep' [ 3 od ''
```

1.0.18 Use JOD help and documentation

Install the (general/joddocument) addon to use JOD PDF help.

```
In [23]: NB. location of jod.pdf - install addon (general/joddocument) with JAL
         smoutput jpath '~addons/general/joddocument/pdfdoc/jod.pdf'

         NB. opens jod.pdf if a pdf reader is available on your system
         jodhelp 0
```

```
c:/j64/j64-807/addons/general/joddocument/pdfdoc/jod.pdf
+-+------------------+
|1|starting PDF reader|
+-+------------------+
```

1.0.19 Summary and final remarks

1. JOD DOES NOT BELONG IN THE J TREE
2. BACKUP BACKUP BACKUP
3. TAKE A SCRIPT DUMP
4. MAKE A MASTER RE-REGISTER SCRIPT
5. SET LIBRARY DICTIONARIES TO READONLY
6. KEEP REFERENCES UPDATED
7. DOCUMENT DICTIONARY OBJECTS
8. DEFINE YOUR OWN JOD SHORTCUTS
9. CUSTOMIZE JOD EDIT FACILITIES
10. DEFINE JOD PROJECT MACROS

11. EDIT YOUR `jodprofile.ijs`

12. USE JOD HELP AND DOCUMENTATION

These are just some of the JOD practices I have found useful. As you use JOD you will probably find your own methods. If you find an a useful method please let me know. I can be reached at:

John Baker
January 2019
bakerjd99@gmail.com

In [ ]: