JOD Introduction Lab

January 29, 2019

1 JOD Introduction Lab

1.0.1 What is JOD?

JOD is a word storage and retrieval system. It is mainly used to ***refactor*** and reuse J words.

The basic idea behind JOD is that J programming is best viewed as organizing collections of ***words*** to perform a task. Organized collections of words have a better name: ***dictionaries!***

JOD is a J addon. It is installed in the (`~addons\general\jod`) branch of the current J system folder by the J package manager.

The next lab step initializes the JOD system.

1.0.2 Start JOD

```
In [1]: NB. used by this lab
        require 'files dir task'

        NB. start jod - creates master file if necessary
        load 'general/jod'

        NB. use portable box drawing characters
        NB. simplifies rendering notebooks as (*.tex)
        portchars_ijod_=:[: 9!:7 '+++++++++|-'"_ [ ]
        portchars ''
```

1.0.3 Remove old lab dictionaries

JOD is installed without any dictionaries. To use JOD you must create some dictionaries. This lab uses four example dictionaries (`lab`), (`labdev`), (`toy`) and (`playpen`). JOD dictionaries are created with the (`newd`) "new dictionary" verb.

Before creating lab dictionaries remove any prior lab dictionaries. This step defines a utility that will erase dictionaries from default locations. It is run in the next step.

WARNING: IF THE TEMPORARY LAB DICTIONARIES CONTAIN INFORMATION YOU CARE ABOUT DO NOT EXECUTE THE NEXT LAB STEP!

```
In [2]: RemoveLabDictionaries_ijod_=: 3 : 0
        root=. jpath '~user'
        if. IFWIN do.
          shell 'rd /s /q "',root,'\joddicts\labdev"'
          shell 'rd /s /q "',root,'\joddicts\lab"'
          shell 'rd /s /q "',root,'\joddicts\toy"'
          shell 'rd /s /q "',root,'\joddicts\playpen"'
          smoutput 'Lab temporary (win) dictionaries erased'
        elseif. IFUNIX do.
          NB. avoid blanks in paths on Linux and Mac systems
          shell 'rm -rf ',root,'/joddicts/labdev'
          shell 'rm -rf ',root,'/joddicts/lab'
          shell 'rm -rf ',root,'/joddicts/toy'
          shell 'rm -rf ',root,'/joddicts/playpen'
```

```
  smoutput 'Lab temporary (mac/linux) dictionaries erased'
elseif.do.
  smoutput 'Erase any previous temporary lab dictionaries manually.'
end.
)
```

1.0.4 Remove any prior lab dictionaries

```
In [3]: NB. close any dictionaries - ignore errors
        0 0 $ 3 od ''

        NB. reset master file
        dpset 'RESETME'

        NB. unregister any lab dictionaries - ignore errors
        0 0 $ 3 regd&> ;:'labdev lab toy playpen'

        NB. remove dictionary directories and all contents - ignore errors
        RemoveLabDictionaries 0
```

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

1.0.5 This step creates the (`lab`) and (`labdev`) dictionaries

```
In [4]: NB. close any open dictionaries
        3 od ''

        NB. create (lab) and  (labdev) dictionaries
        smoutput newd 'lab'
        smoutput newd 'labdev'

        NB. list available dictionaries
        od ''
```

```
+-+---------------------+---+---------------------------------------+
|1|dictionary created ->|lab|c:/users/jbaker/j64-807-user/joddicts/lab/|
+-+---------------------+---+---------------------------------------+
+-+---------------------+------+------------------------------------------+
|1|dictionary created ->|labdev|c:/users/jbaker/j64-807-user/joddicts/labdev/|
+-+---------------------+------+------------------------------------------+
+-+------+------+----+----+---+------+----------+-------+---+------+----+---------+-----+---------+
|1|bpcopy|bptest|docs|imex|jod|joddev|joddevload|jodload|lab|labdev|play|smugpyter|utils|utilsload|
+-+------+------+----+----+---+------+----------+-------+---+------+----+---------+-----+---------+
```

1.0.6 Opening and closing dictionaries

The JOD verb for opening and closing dictionaries is (od) or (open dictionary). JOD verbs are short and easy to type.
(od) can open dictionaries READWRITE and READONLY. As you might expect READONLY dictionaries cannot be changed by JOD verbs.

```
In [5]: NB. open read/write
        smoutput od 'labdev'

        NB. open read/only
        smoutput 2 od 'lab'

        NB. close (labdev)
        3 od 'labdev'
```

```
+-+-------------+------+
|1|opened (rw) ->|labdev|
+-+-------------+------+
+-+-------------+---+
|1|opened (ro) ->|lab|
+-+-------------+---+
+-+---------+------+
|1|closed ->|labdev|
+-+---------+------+
```

1.0.7 Some return code basics

All JOD verbs return boxed list results. The first item is a return code: (1) good (0) bad. Remaining items are messages and, usually, error related information. JOD verbs perform extensive argument checking. If you break a JOD verb please email me (`bakerjd99@gmail.com`) and tell me what you did.

```
In [6]: NB. bad open request
        smoutput od 'i am a missing dictionary'

        NB. bad types
        smoutput od 9

        NB. bad ranks
        od 3 3$'boo'
```

```
+-+------------------------------------------------+
|0|!JOD error: invalid or missing dictionary name(s)|
+-+------------------------------------------------+
+-+------------------------------------------------+
|0|!JOD error: invalid or missing dictionary name(s)|
+-+------------------------------------------------+
+-+------------------------------------------------+
|0|!JOD error: invalid or missing dictionary name(s)|
+-+------------------------------------------------+
```

1.0.8 Online JOD documentation

JOD has extensive (`pdf`) documentation. JOD documentation can be accessed with the (`jodhelp`) verb.

(`jodhelp`) spawns a PDF reader task. JOD uses J's configured PDF reader on Windows and Linux systems and the "open" shell command on Macs.

```
In [7]: NB. display JOD documentation
        jodhelp 0
```

```
+-+------------------+
|1|starting PDF reader|
+-+--------------------+
```

1.0.9 Dictionary paths

The open dictionaries of JOD define a search and fetch path. The (`did`) (dictionary identification) verb lists the path.

```
In [8]: NB. reopen only (labdev) and (lab)
        od ;:'labdev lab' [ 3 od ''

        NB. show path
        did 0
```

```
+-+------+---+
|1|labdev|lab|
+-+------+---+
```

The dyadic form of (`did`) returns details about the contents of each dictionary on the path.

```
In [9]: NB. did ~ 0   NB. handy idiom

        NB. dictionary details
        0 did 0
```

```
+-+-----------------------------------------------------+
|1|+------+--+-----+-----+-------+-------+------+-----+|
| ||      |--|Words|Tests|Groups*|Suites*|Macros|Path*||
| |+------+--+-----+-----+-------+-------+------+-----+|
| ||labdev|rw|0    |0    |0      |0      |0     |/    ||
| |+------+--+-----+-----+-------+-------+------+-----+|
| ||lab   |rw|0    |0    |0      |0      |0     |/    ||
| |+------+--+-----+-----+-------+-------+------+-----+|
+-+-----------------------------------------------------+
```

1.0.10 Some object orientation

The JOD system is a complete and detailed example of object oriented programming in J. The system consists of a number of classes (prefixed with `'ajod'`). When the system loads a variety of objects are created. The basic architecture is a main dictionary object that contains four subobjects. Each open dictionary is also associated with a directory object. Directory objects are created and destroyed as required. The following diagram shows JOD's class structure.

```
In [10]: NB. objects beginning with 'ajod' are the JOD classes.
         smoutput 80 list conl 0

         NB. JOD consists of six basic objects and as
         NB. many directory objects as there are path items.
         conl 1
```

```
ajod      ajoddob   ajodmake  ajodstore ajodtools ajodutil  base      ijod
j         jal       jcompare  jdefs     jdemo     jfif      jfile     jfiles
jfilesrc  jhs       jijs      jijx      jlab      jlogin    jpacman   jregex
jsocket   json      jsp       jtask     qjide     z
+-+-+-+-+-+-+-+-+
|0|1|2|3|4|5|8|9|
+-+-+-+-+-+-+-+-+
```

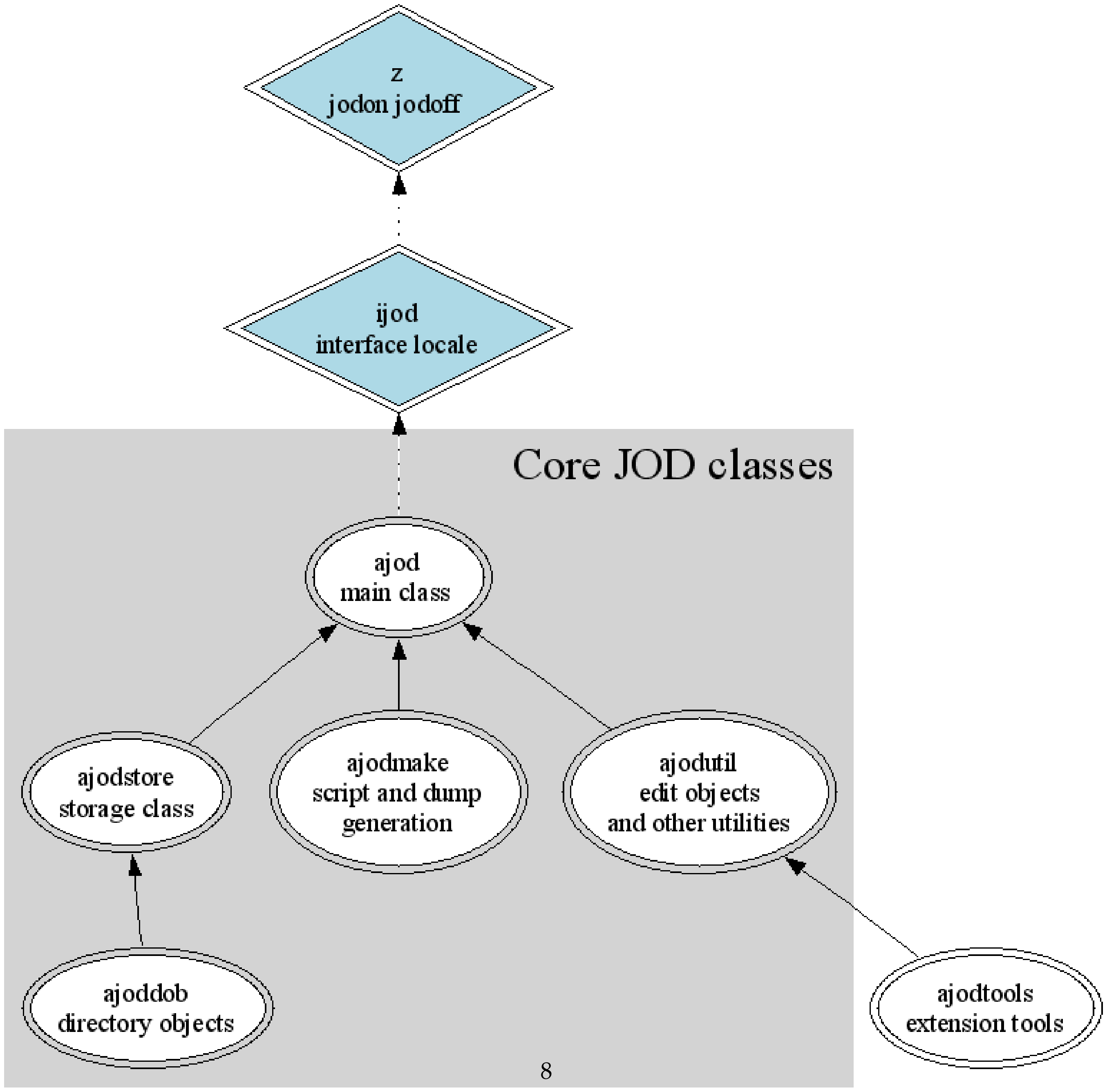
1.0.11 The put dictionary concept

The first dictionary on the path is *special*. It is the only dictionary that can be modified by JOD verbs. Because most dictionary modifications are put's I call this dictionary the "put" dictionary.

It's important to understand that you can use the contents of the other dictionaries on the path but you cannot change them in any way.

```
In [11]: NB. first path dictionary is the put dictionary
         did 0
```

```
+-+------+---+
|1|labdev|lab|
+-+------+---+
```


z
jodon jodoff
ijod
interface locale
Core JOD classes
ajod
main class
ajodstore
storage class
ajodmake
script and dump
generation
ajodutil
edit objects
and other utilities
ajoddob
directory objects
ajodtools
extension tools

1.0.12 Creating new dictionaries

Before modifying the contents of any dictionary let's create a new (`toy`) dictionary and make it the put dictionary.

```
In [12]: NB. close open dictionaries
         3 od ''

         NB. create (toy)
         newd 'toy'
```

```
+-+---------------------+---+----------------------------------------+
|1|dictionary created ->|toy|c:/users/jbaker/j64-807-user/joddicts/toy/|
+-+---------------------+---+----------------------------------------+
```

Make (`toy`) a put dictionary.

```
In [13]: NB. open toy, labdev and lab - toy is the put dictionary
         smoutput od ;:'toy labdev lab'

         NB. insure toy is read/write
         dpset 'READWRITE'
```

```
+-+--------------------+---+------+---+
|1|opened (rw/rw/rw) ->|toy|labdev|lab|
+-+--------------------+---+------+---+
+-+--------------------------------------------+---+
|1|put dictionary read/write status restored ->|toy|
+-+--------------------------------------------+---+
```

1.0.13 Getting and putting words

In the first section I said JOD is a word storage and retrieval system. Now we are ready to (`put`) and (`get`) some words.

First create some words to store.

```
In [14]: NB. create some words
         random=: ?10 10$100  NB. numeric noun
         text=: 'this is a test of the one pure thing'
         floats=: 2 + % 100#100
         symbols=: s: ' once more with feeling'
         boxed=: <"1 i. 2 3
         rationals=: 100 + % (>:i. 10x) ^ 50
         unicode=: u: 'this is now unicode'
         each=: &.>  NB. tacit adverb
         explicit=: 4 : 0
         NB. explicit verb
         x +. y
         )

         words=: ;:'random text floats symbols boxed rationals unicode each explicit'
```

(put) is the JOD command that stores words.
Save and erase the words. Take some time to convince yourself that the words have been erased before proceeding.

```
In [15]: NB. save words
         smoutput put words

         NB. erase definitions
         erase words

+-+-------------------+---+
|1|9 word(s) put in ->|toy|
+-+-------------------+---+
1 1 1 1 1 1 1 1 1
```

Now retrieve the stored words and check that they are properly restored.

```
In [16]: NB. get words
         get words
```

```
+-+------------------+
|1|9 word(s) defined|
+-+------------------+
```

1.0.14 Documentation 101

One of my pet peeves is undocumented code!

How often have you had to face hundreds, maybe thousands, of lines of code with nary a comment in sight. Comments are not for wimps and girly-men. Telling comments are a hallmark of good code.

JOD provides a number of ways to document words. When a word is introduced it's a good idea to store a short one line description of the word.

```
In [17]: NB. store short word descriptions
         smoutput 0 8 put 'random';'random integer table'

         0 8 put 'each';'applies left argument to array items'
```

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

Of course you can view your stored descriptions with (get).

```
In [18]: NB. get short explanations
         0 8 get 'random';'each'
```

```
+-+-------------------------------------------+
|1|+------+------------------------------------+|
| ||random|random integer table                ||
| |+------+------------------------------------+|
| ||each  |applies left argument to array items||
| |+-------+------------------------------------+|
+-+---------------------------------------------+
```

More detailed documentation can be stored and retrieved. This step loads a realistic example of word documentation into the current put dictionary and then displays with (`disp`).

(`disp`) is a JOD utility. It is the only verb that returns a character list (when successful) instead of the usual boxed (`rc;value`)

```
In [19]: NB. loads (changestr) and (changestr) documentation into the current put dictionary
         script '~addons\general\jod\jodlabs\labexample001.ijs'
```

This steps displays the long document loaded in the previous step.

```
In [20]: NB. show long documentation
         0 9 disp 'changestr'
```

```
*changestr v-- replaces substrings within a string.

This algorithm was adapted from an APL algorithm. It requires
high speed boolean bit  manipulation and is not  as effective
in current  J systems as it  is in some  APL systems. Despite
J's non-optimal booleans this verb is still fast enough to be
fruitfully applied.  On  my  400MHZ NT  machine  you can make
20,000 length increasing replacements, (the worst case), in a
1  megabyte  string  in approximately  one  second.  For  100
kilobyte  strings typical operations complete is less than  a
tenth of second.

High speed substring replacement is difficult to achieve in J
and APL environments. This verb would be a good candidate for
an external compiled routine.

dyad:  clChanged =. clTargets changestr clStr

  '/change/becomes' changestr 'change me'
```

```
'/delete' changestr 'delete me'   NB. null replacement deletes

NB. first character is delimiter

'.remove..purge..wipe' changestr 'removepurgewipe'

'/' changestr 'nothing happens'

'' changestr 'nothing happens'

'/nothing/happens' changestr 'no matches to change'

NB. multiple replacements are made in left to right order

t =. 'once all things were many'

'/many/changes/all/at/once/ehh' changestr t

NB. even null subtring replacements are allowed

'//XX' changestr 'insert big x chars around us'

NB. finally all this applies in a clean elegant
NB. way to UNICODE strings as well

uchars=. u: 1033 + i. 500  NB. unicode string
datatype uchars            NB. (datatype) from j profile

usub0=. (100+i.11){uchars  NB. substrings
usub1=. (313+i.7){uchars
datatype usub0
datatype usub1
```

```
  NB. strings that will not occur in the original
  unew0=. u: 40027+i.33
  unew1=. u: 50217+i.7

  +./ unew0 E. uchars   NB. not in uchars
  +./ unew1 E. uchars

  ucharsnew=. ('/',usub0,'/',unew0,'/',usub1,'/',unew1) changestr uchars

  +./ unew0 E. ucharsnew  NB. now in string
  +./ unew1 E. ucharsnew
```

1.0.15 More putting and getting

(put) and (get) are quite flexible and can store entire locales. The locales can be named or numbered.

```
In [21]: NB. save the (ajodmake) locale/class in "toy"
         smoutput 'ajodmake' put zz=: nl_ajodmake_ i. 4

         NB. retrieve the (ajodmake) words into an 'xxx' locale
         'xxx' get zz
```

```
+-+--------------------+---+
|1|78 word(s) put in ->|toy|
+-+--------------------+---+
+-+------------------+
|1|78 word(s) defined|
+-+------------------+
```

1.0.16 Searching for words

Like most storage systems JOD provides facilities for searching the contents of its database.
The main search command is (`dnl`) (dictionary name lists).

```
In [22]: NB. list all the words on the path beginning with 'du'
         list }. dnl 'du'
```

```
dumpdictdoc  dumpdoc      dumpgs       dumpheader   dumpntstamps dumptext     dumptm       dumptrailer  dumpwords
```

(`dnl`) can search for words, tests, groups, suites and macros.
This step creates some groups and then lists all the groups on the path that begin with `'JOD'`.

```
In [23]: NB. create some groups
         grp 'strings';'changestr'
         grp 'loctest';nl_ajodmake_ i.4

         NB. groups beginning with 'loc'
         2 1 dnl 'loc'
```

```
+-+-------+
|1|loctest|
+-+-------+
```

1.0.17 What are these funny argument numbers?

By now you have probably noticed that many JOD verbs take integer arguments. JOD argument codes are of basically three types, object codes, option codes and qualifiers.
The objects JOD stores and retrieves all have object codes. The next table displays JOD object codes.

```
In [24]: NB. JOD object codes
         (<"0 i. 6) ,. ;:'WORD TEST GROUP SUITE MACRO DICTIONARY'
```

```
+-+----------+
|0|WORD      |
+-+----------+
|1|TEST      |
+-+----------+
|2|GROUP     |
+-+----------+
|3|SUITE     |
+-+----------+
|4|MACRO     |
+-+----------+
|5|DICTIONARY|
+-+----------+
```

Option and qualifier codes select and modify options. They are all integers. For more information about argument codes read JOD's documentation.

Now look at some more (`dnl`) commands.

```
In [25]: NB. (group, option 1 - match prefix) - case matters
         smoutput 2 1 dnl 'l'

         NB. (macro, option 2 - name contains string) - no macros yet
         smoutput 4 2 dnl 'ar'

         NB. make a macro and search again
         smoutput 4 put 'arrgh';JSCRIPT_ajod_;'NB. my do nothing J macro'

         4 2 dnl 'ar'
```

```
+-+-------+
|1|loctest|
+-+-------+
+-++
|1||
+-++
+-+--------------------+---+
|1|1 macro(s) put in ->|toy|
+-+--------------------+---+
+-+-----+
|1|arrgh|
+-+-----+
```

1.0.18 Groups and suites

JOD provides a simple way to group words and tests. A group is a collection of J words. A suite is a collection of J test scripts. You create and modify groups and suites with the (grp) verb.

```
In [26]: NB. create a group of words with names beginning with 'ch'
         grp 'testgroup' ; }. dnl 'ch'

         NB. create a test
         1 put 'helloworld';'1 [ ''JOD tests are J scripts that return 1s'''

         NB. create a test suite - note left argument code
         3 grp 'testsuite' ; }. 1 dnl ''
```

```
+-+---------------------------+---+
|1|suite <testsuite> put in ->|toy|
+-+---------------------------+---+
```

1.0.19 You can list the contents of groups or suites with (grp)

```
In [27]: NB. list contents of testgroup group
         smoutput grp 'testgroup'

         NB. contents of testsuite, note suite code left argument
         3 grp 'testsuite'
```

```
+-+---------+
|1|changestr|
+-+---------+
+-+----------+
|1|helloworld|
+-+----------+
```

1.0.20 Making groups and suites

One of the main advantages of storing J code in JOD vs. a plain script is that you can maintain a *single* version of a word, test, group or suite and then generate many J load scripts that use dictionary objects. Database designers call this "one version of the truth."

The following inserts a single word in a (`toy`) group and then generates scripts.

```
In [28]: NB. left justify table verb
         ljust=:' '&$: :([ |."_1~ i."1&0@(] e. [))

         NB. store in put dictionary
         put 'ljust'

         NB. insert in all put dictionary groups
         (}. 2 revo '') addgrp&> <'ljust'

         NB. lookup (revo) in jod.pdf with (jodhelp)

         NB. generate all put dictionary groups
         smoutput 0 mls&> }. 2 revo''

         NB. if the left argument is elided the groups are made into (load) scripts
         NB. mls&> }. 2 revo''
```

```
+-+-------------+----------------------------------------------------------+
|1|file saved ->|c:/users/jbaker/j64-807-user/joddicts/toy/script/strings.ijs  |
+-+-------------+----------------------------------------------------------+
|1|file saved ->|c:/users/jbaker/j64-807-user/joddicts/toy/script/loctest.ijs  |
+-+-------------+-----------------------------------------------------------+
|1|file saved ->|c:/users/jbaker/j64-807-user/joddicts/toy/script/testgroup.ijs|
+-+-------------+-----------------------------------------------------------+
```

1.0.21 Macros

Tasks, like updating generated scripts, can be simplified with JOD macros. A JOD macro is an arbitrary J script that can be fetched and executed with (`rm`).

```
In [29]: NB. macro that generates all put dictionary groups
         jodmacro=: 'NB. generate all put dictionary groups',LF,'0 mls&> }. 2 revo'''' '

         NB. store macro - code (JSCRIPT_ajod_) tells JOD this is a J script
         4 put 'makeputgrps';JSCRIPT_ajod_;jodmacro
```

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

Running a JOD macro is a simple matter of opening the appropriate dictionaries and using (`rm`) - run macro.

```
In [30]: NB. fetch and execute silently - will only display explicit code output
         NB. 1 rm 'makeputgrps'

         NB. fetch and execute
         rm 'makeputgrps'
```

```
   NB. generate all put dictionary groups
   0 mls&> }. 2 revo''
+-+-------------+-----------------------------------------------------------+
|1|file saved ->|c:/users/jbaker/j64-807-user/joddicts/toy/script/strings.ijs  |
+-+-------------+-----------------------------------------------------------+
|1|file saved ->|c:/users/jbaker/j64-807-user/joddicts/toy/script/loctest.ijs  |
+-+-------------+-----------------------------------------------------------+
|1|file saved ->|c:/users/jbaker/j64-807-user/joddicts/toy/script/testgroup.ijs|
+-+-------------+-----------------------------------------------------------+
```

Macros are not restricted to J scripts. You can also store HTML, LaTeX, XML, TEXT, BTYE, MARKDOWN, UTF8, SQL, PYTHON and JSON scripts in JOD dictionaries. Only J scripts can be run however.

```
In [31]: NB. store LaTeX (22) and HTML (23) texts
         4 put 'latex';22;'... LaTeX code ...'

         4 put 'html';23;' ... HTML code ...'

         NB. store XML and arbitrary TEXT (bytes).
         4 put 'xml';XML_ajod_;'<test>this is lame xml</test>'

         NB. BYTE is uninterpreted bytes and can store binaries - not recommended for large files.
         4 put 'BIN';26;read_ajod_ jpath '~addons\general\jod\jmaster.ijf'

         NB. byte size of macro
         smoutput 4 15 get 'BIN'

         NB. macro text types are contants in the main JOD class
         JSCRIPT_ajod_, LATEX_ajod_, HTML_ajod_, XML_ajod_, TEXT_ajod_, BYTE_ajod_, MARKDOWN_ajod_, UTF8_ajod_, PYTHON_ajod_,
```

```
+-+-----+
|1|82048|
+-+-----+
21 22 23 24 25 26 27 28 29 30 31
```

1.0.22 Loading dictionary dump scripts

To demonstrate other JOD features we need some words in our dictionary. The next step loads (`labdump.ijs`).

```
In [32]: NB. insure correct path
         od ;:'toy labdev lab' [ 3 od ''
         NB. load dump script
         0!:0 <jpath '~addons/general/jod/jodlabs/labdump.ijs'

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

Dump scripts do not store word references. They must be generated.

```
In [33]: NB. update word references - show first 5 messages
         5 {. 0 globs&> }. revo''

+-+--------------------------------+---++++++++++++
|1|<antimode> references put in -> |toy|||||||||||
+-+--------------------------------+---++++++++++++
|1|<arctan> references put in ->   |toy|||||||||||
+-+--------------------------------+---++++++++++++
|1|<calmoons> references put in -> |toy|||||||||||
+-+--------------------------------+---++++++++++++
|1|<cos> references put in ->      |toy|||||||||||||
+-+--------------------------------+---++++++++++++
|1|<datecheck> references put in ->|toy|||||||||||||
+-+--------------------------------+---++++++++++++
```

1.0.23 Global references

JOD has facilities for carrying out static name analysis on J words and tests.
The (`globs`) and (`uses`) verbs analyze and stored name references.

```
In [34]: NB. analyze names
         get 'dstat'

         NB. classify name use in base locale word
         11 globs 'dstat'
```

```
+-+------------------------------------------------------------------------+
|1|+------+---------------------------------------------------------------+|
| ||Global|+--------+--------+----+------+-----+--+--+--------+------+||
| ||      ||antimode|kurtosis|mean|median|mode2|q1|q3|skewness|stddev|||
| ||      |+--------+--------+----+------+-----+--+--+--------+------+||
| |+------+---------------------------------------------------------------+|
| ||Local |+---+---+-+-+                                                  ||
| ||      ||max|min|t|v|                                                  ||
| ||      |+---+---+-+-+                                                  ||
| |+------+---------------------------------------------------------------+|
| ||(*)=: |                                                               ||
| |+------+---------------------------------------------------------------+|
| ||(*)=. |                                                               ||
| |+------+---------------------------------------------------------------+|
| ||for.  |                                                               ||
| |+------+---------------------------------------------------------------+|
+-+------------------------------------------------------------------------+
```

You can update global word references.

```
In [35]: NB. 0 is the word code - stores global references
         0 globs 'dstat'
```

```
+-+----------------------------+---+
|1|<dstat> references put in ->|toy|
+-+----------------------------+---+
```

(uses) retrieves stored references.

```
In [36]: NB. global references for (dstat)
         uses 'dstat'
```

```
+-+-----------------------------------------------------------------+
|1|+-----+---------------------------------------------------------++|
| ||dstat|+--------+--------+----+------+-----+--+--+--------+------+||
| ||     ||antimode|kurtosis|mean|median|mode2|q1|q3|skewness|stddev|||
| ||     |+--------+--------+----+------+-----+--+--+--------+------+||
| |+-----+---------------------------------------------------------++|
+-+-----------------------------------------------------------------+
```

(uses) becomes very "useful" when all words have stored references.

```
In [37]: NB. insure toy is the put dictionary
         od ;:'toy labdev lab' [ 3 od ''
```

```
+-+--------------------+---+------+---+
|1|opened (rw/rw/rw) ->|toy|labdev|lab|
+-+--------------------+---+------+---+
```

(uses) can return many reference lists at once. The same path search mechanism is used for retrieving references.

```
In [38]: NB. global references of words beginning with 'm'
         NB. uses }. dnl 'm'

         NB. global references of words ending with 's'
         uses }. 0 3 dnl's'
```

```
+-+--------------------------------------+
|1|+------------+---------------------++|
| ||calmoons    |+----------+-----+    |||
| ||            ||fromjulian|moons|    |||
| ||            |+----------+-----+    |||
| |+------------+---------------------++|
| ||cos         |                      |||
| |+------------+---------------------++|
| ||dumpgs      |                      |||
| |+------------+---------------------++|
| ||dumpntstamps|                      |||
| |+------------+---------------------++|
| ||dumpwords   |                      |||
| |+------------+---------------------++|
| ||extscopes   |                      |||
| |+------------+---------------------++|
| ||floats      |                      |||
| |+------------+---------------------++|
| ||fuserows    |                      |||
| |+------------+---------------------++|
| ||getallts    |                      |||
| |+------------+---------------------++|
| ||halfbits    |                      |||
| |+------------+---------------------++|
| ||jscriptdefs |                      |||
| |+------------+---------------------++|
| ||kurtosis    |+---+-----+           |||
| ||            ||dev|ssdev|           |||
| ||            |+---+-----+           |||
| |+------------+---------------------++|
| ||makegs      |                     |||
| |+------------+---------------------++|
| ||moons       |+---+                |||
| ||            ||sin|                |||
| ||            |+---+                |||
| |+------------+---------------------++|
| ||namecats    |                     |||
| |+------------+---------------------++|
| ||opaqnames   |                     |||
| |+------------+---------------------++|
| ||putallts    |                     |||
| |+------------+---------------------++|
| ||rationals   |                     |||
| |+------------+---------------------++|
| ||skewness    |+---+-----+          |||
| ||            ||dev|ssdev|          |||
| ||            |+---+-----+          |||
| |+------------+---------------------++|
| ||symbols     |                     |||
| |+------------+---------------------++|
| ||wrdglobals  |                     |||
| |+------------+---------------------++|
| ||writeijs    |                     |||
| |+------------+---------------------++|
| ||yeardates   |+---------+---------+|||
| ||            ||datecheck|yeardates||||
| ||            |+---------+---------+|||
| |+------------+---------------------++|
+-+-------------------------------------+
```

1.0.24 The `uses` union

Option 31 of (`uses`) returns the ***uses-union*** of a word. The uses-union is basically a unique list of all the words on the call tree of a word.

```
In [39]: NB. uses union of two words
         31 uses ;:'calmoons sunriseset0'

+-+--------------------------------------------------+
|1|+-----------+-------------------------------------++|
| ||calmoons   |+----------+-----+---+              |||
| ||           ||fromjulian|moons|sin|              |||
| ||           |+----------+-----+---+              |||
| |+-----------+-------------------------------------++|
| ||sunriseset0|+---------+------+---+---+-----+---+|||
| ||           ||NORISESET|arctan|cos|sin|tabit|tan||||
| ||           |+---------+------+---+---+-----+---+|||
| |+-----------+-------------------------------------++|
+-+--------------------------------------------------+
```

1.0.25 Generating load scripts

JOD can generate J load scripts from dictionary groups. The generated scripts are written to the put dictionary's script subdirectory.

```
In [40]: NB. generate load script
         mls 'sunmoon'   NB. sun/moon rise set

+-+--------------------+-------------------------------------------------------------+
|1|load script saved ->|c:/users/jbaker/j64-807-user/joddicts/toy/script/sunmoon.ijs|
+-+--------------------+-------------------------------------------------------------+
```

(`mls`) appends generated scripts to the current user's `startup.ijs` file so they can be loaded independently of JOD.
Note: mls scripts are added to `PUBLIC_j_` or `Public_j_` for the current user.

```
In [41]: NB. load generated script
         load 'sunmoon'

         calmoons 2019  NB. full (1)  and new (0) moons in 2019
```

```
0 2019  1  5
1 2019  1 20
0 2019  2  4
1 2019  2 19
0 2019  3  6
1 2019  3 20
0 2019  4  4
1 2019  4 18
0 2019  5  4
1 2019  5 18
0 2019  6  2
1 2019  6 16
0 2019  7  2
1 2019  7 16
0 2019  7 31
1 2019  8 15
0 2019  8 29
1 2019  9 13
0 2019  9 28
1 2019 10 13
0 2019 10 27
1 2019 11 12
0 2019 11 26
1 2019 12 11
0 2019 12 25
```

1.0.26 Generating scripts on demand

JOD can also generate and load scripts without creating load scripts.

```
In [42]: NB. load basic statistics group
         lg 'bstats'
```

```
+-+-------------------+
|1|bstats group loaded|
+-+-------------------+
```

(getrx) loads all the words called by a given word.

```
In [43]: NB. load into arbitrary locales
         NB. 'statloc' getrx 'dstat'
         NB. '99' getrx 'dstat'

         NB. load all words needed to run (dstat)
         getrx 'dstat'
```

```
+-+------------------------------+
|1|(14) words loaded into -> base|
+-+------------------------------+
```

1.0.27 Backing up and restoring dictionaries

JOD is database for J words, scripts and other precious program texts. Most database systems have means for backing up and restoring databases and JOD does as well. The (packd) verb backups up a database.

```
In [44]: NB. save a backup of the current put dictionary
         packd 'toy'
```

```
+-+--------------------+---+-+
|1|dictionary packed ->|toy|0|
+-+--------------------+---+-+
```

(packd) copies the current dictionary files to the backup subdirectory and prefixes all the files with a unique ever increasing backup number.

```
In [45]: NB. list put dictionary backup files
         BDIR=: {:{.  DPATH__ST__JODobj   NB. put directory
         dir BAK__BDIR,'*.ijf'            NB. backup files
```

```
0jgroups.ijf       22400 28-Jan-19 11:21:27
0jmacros.ijf      142208 28-Jan-19 11:21:27
0jsuites.ijf        6272 28-Jan-19 11:21:26
0jtests.ijf         6016 28-Jan-19 11:21:26
0juses.ijf         21120 28-Jan-19 11:21:27
0jwords.ijf       202560 28-Jan-19 11:21:28
```

(`restd`) restores the last backup by selecting backup files with the highest prefix.

```
In [46]: NB. restore last backup
         restd 'toy'
```

```
+-+----------------------+---+-+
|1|dictionary restored ->|toy|0|
+-+----------------------+---+-+
```

(`packd`) creates binary backups. You can also backup dictionaries as dump scripts. Dump scripts are single J scripts that can be used to backup, copy and merge dictionaries.

```
In [47]: NB. dump all the words on the path as a single dump script.
         toydump=: showpass_ajod_ make ''
```

```
+-+---------------------------+-------------------------------------------------------+
|1|object(s) on path dumped ->|c:/users/jbaker/j64-807-user/joddicts/toy/dump/toy.ijs|
+-+---------------------------+-------------------------------------------------------+
```

When we load (`toydump`) into a new dictionary observe how the path is changed. The dictionaries have been merged.

```
In [48]: NB. new dictionary
         newd 'playpen' [ 3 od ''

         NB. open
         od 'playpen'
```

```
    NB. load (toydump)
    0!:0 < ;{: toydump

    NB. dictionary information
    did~ 0

+-+--------------------+-------+
|1|42 word(s) put in ->|playpen|
+-+--------------------+-------+
+-+--------------------+-------+
|1|50 word(s) put in ->|playpen|
+-+--------------------+-------+
+-+--------------------+-------+
|1|32 word(s) put in ->|playpen|
+-+--------------------+-------+
+-+--------------------------------+-------+
|1|38 word explanation(s) put in ->|playpen|
+-+--------------------------------+-------+
+-+----------------------------+-------+
|1|1 word document(s) put in ->|playpen|
+-+----------------------------+-------+
+-+----------------------------+-------+
|1|2 word document(s) put in ->|playpen|
+-+----------------------------+-------+
+-+-------------------+-------+
|1|1 test(s) put in ->|playpen|
+-+-------------------+-------+
+-+--------------------+-------+
|1|6 macro(s) put in ->|playpen|
+-+--------------------+-------+
+-+---------------------------+-------+
|1|group <bstats> put in ->   |playpen|
+-+---------------------------+-------+
|1|group <loctest> put in ->  |playpen|
+-+--------------------------+-------+
|1|group <strings> put in ->  |playpen|
+-+--------------------------+-------+
|1|group <sunmoon> put in ->  |playpen|
+-+--------------------------+-------+
|1|group <testgroup> put in ->|playpen|
+-+--------------------------+-------+
+-+--------------------------+-------+
|1|suite <testsuite> put in ->|playpen|
+-+--------------------------+-------+
NB. end-of-JOD-dump-file regenerate cross references with:  0 globs&> }. revo ''
+-+----------------------------------------------------------+
|1|+-------+--+-----+-----+-------+-------+------+--------+|
| ||       |--|Words|Tests|Groups*|Suites*|Macros|Path*   ||
| |+-------+--+-----+-----+-------+-------+------+--------+|
| ||playpen|rw|124  |1    |5      |1      |6     |/playpen||
| |+-------+--+-----+-----+-------+-------+------+--------+|
+-+----------------------------------------------------------+
```

1.0.28 Final words

You now have some idea of what JOD is all about. To learn more read JOD's documentation and run the other JOD labs. If you have any problems, questions or complaints please email me at bakerjd99@gmail.com

```
 John Baker
 bakerjd99@gmail.com
 January 2019
```

```
In [49]: NB. close any open dictionaries
         3 od ''
```

```
+-+---------+-------+
|1|closed ->|playpen|
+-+---------+-------+
In [ ]:
```