

Link your code

Morten Kromberg
CXO, Dyalog Ltd.

Goals

- Tell a few stories about my journey towards building APL systems from source in text files
- Warm you up for Gil, so he can put everything in its proper place



Agenda

- Why text files?
- Converting Python to APL
 - Using the "PyCharm" IDE
- Look at some User commands:
 -]link: a namespace and a folder
 -]dbuild: bring several namespaces in
 -]dtest: is it still working?



Why text files?

- The idea of storing APL code in text files is hardly new...

▽ R←A RUA B

[1] →L2B×ιA>B

[2] ... spaghetti

[...]

[471] L2B:

[472] al ragu ..

▽

Source Code Mgt Demo

- All tools shown here downloaded from internet, none of them knew about APL in any way.

The screenshot displays a source code comparison tool with two panes. The left pane shows the original code, and the right pane shows a modified version. The code is in Dyalog APL and defines a namespace 'MyApp' with a function 'Main:Berlin'. The modified version includes a new line of code: 'U+##.U A Link to Utils'.

```
MyApp.dyalog-revBASE.svn001.tmp.dyalog + MyApp.dyalog - Compare It! 3.86
File Edit Merge View Options Tools Help
SecureConnection
C:\Users\vnkrom\AppData\Local\Temp\MyApp.dyalog-revBASE.svn001.tmp.dyalog
:Namespace MyApp
A This is a Very Simple Application done for APL Germany
U+##.U A Make a Link to Utils
Main:Berlin
  'Berlin'⎕WC'Form' 'Hello World!'('Size'(35 200))('Coord'
  'Berlin.OK'⎕WC'Button'(U.ucase'ok')(10 10)(θ 100)'Event'
  ⎕DQ'Berlin'
  A I think this is a mistake
  ⎕OFF
:EndNamespace
10-10-2008 11:04:01 WIN UTF8 Ln 4 Col 1

C:\myapp\MyApp.dyalog
:Namespace MyApp
A This is a Very Simple Application done for APL Germany
U+##.U A Link to Utils
Main:Berlin
  'Berlin'⎕WC'Form' 'Hello Berlin!'('Size'(35 200))('Coord'
  'Berlin.OK'⎕WC'Button'(U.ucase'ok')(10 10)(θ 100)'Event'
  ⎕DQ'Berlin'
  ⎕OFF
:EndNamespace
10-10-2008 10:21:59 WIN UTF8 Ln 4 Col 1
Source Only (1) Dest only Changed (2) EDIT OVR CAPS NUM
```

Source Code Mgt Demo

- All tools shown here downloaded from internet, none of them knew about APL in any way.

The screenshot shows a source code comparison tool with two panes. The left pane shows the original code, and the right pane shows the modified code. The modified code has several lines highlighted in green, indicating changes. A red box highlights the status bar at the bottom of the right pane, which shows 'Source Only (1) Dest only Changed (2)'. The status bar also includes 'EDIT', 'OVR', 'CAPS', and 'NUM' options.

```
:Namespace MyApp
A This is a Very Simple Application done for APL Germany

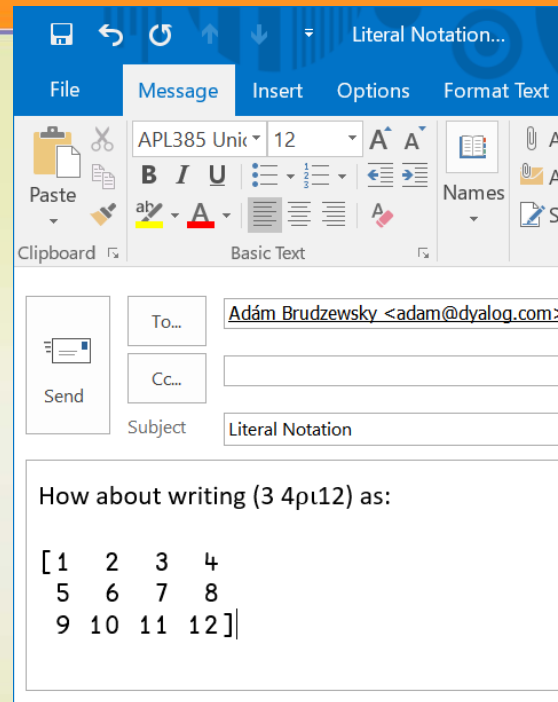
U+##.U A Make a Link to Utils

  ▽ Main;Berlin
  'Berlin'⎵WC'Form' 'Hello World!'('Size'(35 200))('Coord
  'Berlin.OK'⎵WC'Button'(U.ucase'ok')(10 10)(θ 100)'Event
  ⎵DQ'Berlin'
  A I think this is a mistake
  ⎵OFF
  ▽

:EndNamespace
```

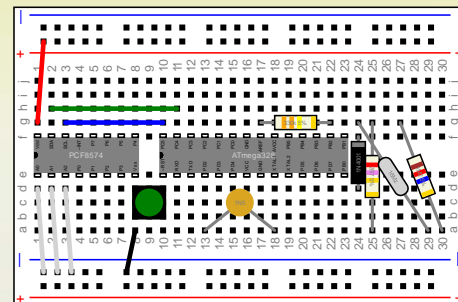
Benefits of Text Source

- Use mainstream source code management systems
 - ❖ SVN, GitHub, Mercurial, CVS, ...
 - ❖ File "diff" tools etc
- Easily share code between APL versions
- Read, collaboratively write and exchange APL code without installing an APL IDE
 - ❖ Most operating systems display readable APL without installing APL fonts
 - ❖ Recent Linuxes even come with APL keyboard support built in (thanks to Geoff!)
- ❖ New users expect it
 - ❖ Not just Software Engineers; anyone who collaborated on software will have used Git[Hub] etc

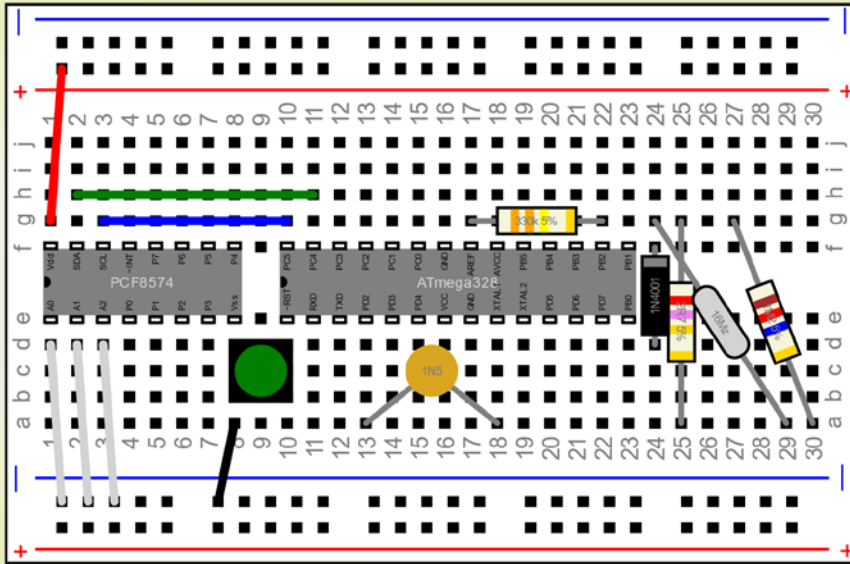


Project to convert Python Project to APL

- Learn a little Python
 - Know your "enemy"
- Learn about the Python IDE
 - Any ideas worth st... er, borrowing?
- Thanks to Romilly Cocking for providing the example and doing some teaching!



<https://github.com/romilly/breadboarder>



- Romilly is an electronics geek
- He is developing a tool to create "breadboard" designs
- In Python, on GitHub



Working with Python

- PyCharm from JetBrains is a very popular Python IDE
- I installed the "Community Edition"
- Will probably get the "Professional Edition" any day now ...



Demo 1: Breadboarding



Nice Things About Python

```
:Class Text
```

```

▽ make args
:Access Public
:Implements Constructor
(text start color anchor size)+5args,(#args)+args,('(0 0)'black' 'middle' 8
▽

▽ r+element;props
:Access Public

props+!'x' 'y',;start
props,+c'style' ('fill:',color,';font-size: ',(▼size),'pt;text-anchor:',anchor)
:If angle≠0
  props,+c('R -') 'transform' ('rotate(',(▼angle),(c','"▼"[start),'))
:EndIf

r-NEW #.Element((c'text'),props)
r.text+text
▽

```

```
class Text(Drawable):
```

```

def __init__(self, text, start, color='black', anchor='start', size=8, **attributes):
    Drawable.__init__(self, start)
    self.text = text
    self.color = color
    self.anchor = anchor
    self.size = size
    self._attributes = attributes
    self.angle = 0

```

```
def element(self):
```

```

text = Element('text', x=str(self.start.x), y=str(self.start.y),
               style= 'fill:%s;text-anchor:%s;font-size: %dpt' % (self.color, self.anchor, self.size))
text.text = self.text
if self.angle != 0:
    text.set('transform','rotate(%d,%d,%d)' % (self.angle, self.start.x, self.start.y))
return text

```



Nice Things About Python

:Class Text

```

▽ make args
:Access Public
:Implements Constructor
(text start color anchor size)+5args,(#args)+args,'(0 0)'black' 'middle' 8
▽

▽ r+element;props
:Access Public

props+! 'x' 'y',;start
props,+c 'style' ('fill:',color,';font-size: ',(▼size),'pt;text-anchor:',anchor)
:If angle≠0
  props,+c('R -') 'transform' ('rotate(',(▼angle),(c','"▼"[start),')')
:EndIf

r-NEW #.Element((c'text'),props)
r.text+text
▽

```

class Text(Drawable):

```
def __init__(self, text, start, color='black', anchor='start', size=8, **attributes):
```

```
    Drawable.__init__(self, start)
```

```
    self.text = text
```

```
    self.color = color
```

```
    self.anchor = anchor
```

```
    self.size = size
```

```
    self._attributes = attributes
```

```
    self.angle = 0
```

```
def element(self):
```

```
    text = Element('text', x=str(self.start.x), y=str(self.start.y),
```

```
        style= 'fill:%s;text-anchor:%s;font-size: %dpt' % (self.color, self.anchor, self.size))
```

```
    text.text = self.text
```

```
    if self.angle != 0:
```

```
        text.set('transform','rotate(%d,%d,%d)' % (self.angle, self.start.x, self.start.y))
```

```
    return text
```



Nice Things About Python

```
:Class Text
  ▽ make args
    :Access Public
    :Implements Constructor
    (text start color anchor size)+5args,(#args)+args,('(0 0)'black' 'middle' 8)
  ▽
  ▽ r+element;props
    :Access Public

    props+!'x' 'y',;start
    props,+c'style' ('fill:',color,';font-size: ',(▼size),'pt;text-anchor:',anchor)
    :If angle≠0
      props,+c('R -') 'transform' ('rotate(',(▼angle),(ε','"▼[start],)')')
    :EndIf
    r-NEW #.Element((c'text'),props)
    r.text+text
  ▽
```

```
class Text(Drawable):
    def __init__(self, text, start, color='black', anchor='start', size=8, **attributes):
        Drawable.__init__(self, start)
        self.text = text
        self.color = color
        self.anchor = anchor
        self.size = size
        self._attributes = attributes
        self.angle = 0

    def element(self):
        text = Element('text', x=str(self.start.x), y=str(self.start.y),
            style= 'fill:%s;text-anchor:%s;font-size: %dpt' % (self.color, self.anchor, self.size))
        text.text = self.text
        if self.angle != 0:
            text.set('transform','rotate(%d,%d,%d)' % (self.angle, self.start.x, self.start.y))
        return text
```



Nice things about APL

```
:Class Rectangle  
  
▽ r←set_center point  
:Access Public  
topleft←point-0.5×extent  
r←□THIS  
▽  
  
▽ r←center  
:Access Public  
r←topleft+0.5×extent  
▽
```

```
class Rectangle(Drawable):  
    def set_center(self, x, y):  
        self.move_to(Point(x-0.5*self.width, y-0.5*self.height))  
        return self  
  
    def center(self):  
        return self.start + Point(self.width, self.height).scale(0.5)
```

Nice things about APL

```
:Class Rectangle  
  
▽ r←set_center point  
:Access Public  
topleft←point-0.5×extent  
r←□THIS  
▽  
  
▽ r←center  
:Access Public  
r←topleft+0.5×extent  
▽
```

```
class Rectangle(Drawable):  
    def set_center(self, x, y):  
        self.move_to(Point(x-0.5*self.width, y-0.5*self.height))  
        return self  
  
    def center(self):  
        return self.start + Point(self.width, self.height).scale(0.5)
```


Nice things about APL

```
:Class Rectangle  
  
▽ r←set_center point  
:Access Public  
topleft←point-0.5×extent  
r←□THIS  
▽  
  
▽ r←center  
:Access Public  
r←topleft+0.5×extent  
▽
```

```
class Rectangle(Drawable):  
    def set_center(self, x, y):  
        self.move_to(Point(x-0.5*self.width, y-0.5*self.height))  
        return self  
  
    def center(self):  
        return self.start + Point(self.width, self.height).scale(0.5)
```

Really Nice thing about APL

```
sortstyles←{
  w←ω[⊆ω;]
  0≠#styles←⊆' ; 'ε''w[;2]: ω
  w[styles;2]←{1↓ε' ; ' , ''{ω[⊆ω]}((~m)×1++\m←ω=' ; ' )≤ω}''w[styles;2]
  w
}
```

```
fixtrailingzeros←{
  0≠#i←⊆(t←⊆''⊆VFI''ω[;2])ε<,1: ω
  w←ω
  w←w[i;2]←⊆''t[i]
}
```

```
project←⊆NEW Project
(breadboard←⊆NEW Breadboard).move_to 20 20
project.Add breadboard
```

```
svg←project.svg
pysvg←⊆NGET'c:\devt\breadboarder\svg\bb-apl.svg'
```

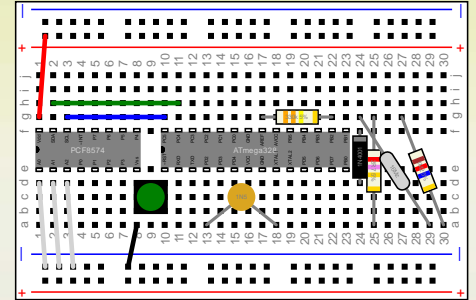
```
(xml pyxml)←⊆XML'' svg pysvg
xml[;4]←fixtrailingzeros''sortstyles''xml[;4]
pyxml[;4]←fixtrailingzeros''sortstyles''pyxml[;4]
```

```
r←{(≡/ω),ω} xml pyxml
```



Breadboarder Conclusions

- Being able to share a single project and data files on GitHub has been a HUGE benefit for this project
- The APL code is out there for Romilly's fans to see if they are curious
- The `]link` user command allowed me to
 - Work interactively with APL in my customary fashion
 - Have all changes immediately reflected in the code repository
 - If Romilly made any changes to the APL code, I could do a "Git Pull" while talking to him, and immediately see his code in my workspace



```
]link ns dir
```



]link ns dir

- "links" a namespace structure to a folder structure:
 - Each function, operator, class or scripted namespace corresponds to an external file



Demo 2:]link



How it works: Changes in the Workspace



How it works: Changes in the Workspace

- Link attaches itself to the EditorFix callback so that it can react to changes made by the user in the editor.



How it works: Changes in the Workspace

- Link attaches itself to the EditorFix callback so that it can react to changes made by the user in the editor.
- The exact same code can be called under program control:
 - `ns name [oldname] SE.Link.Fix src`



How it works: Changes in the Workspace

- Link attaches itself to the EditorFix callback so that it can react to changes made by the user in the editor.
- The exact same code can be called under program control:
 - `ns name [oldname] □SE.Link.Fix src`
- This allows tools that search and replace or make other changes to source code to report the changes to the link system



How it works: Changes in the Workspace

- Link attaches itself to the EditorFix callback so that it can react to changes made by the user in the editor.
- The exact same code can be called under program control:
 - `ns name [oldname] □SE.Link.Fix src`
- This allows tools that search and replace or make other changes to source code to report the changes to the link system
- `src` can either be source to use, or empty to ask link to retrieve the source from the workspace itself



How it works: Changes in the Workspace

- Link attaches itself to the EditorFix callback so that it can react to changes made by the user in the editor.
- The exact same code can be called under program control:
 - `ns name [oldname] []SE.Link.Fix src`
- This allows tools that search and replace or make other changes to source code to report the changes to the link system
- `src` can either be source to use, or empty to ask link to retrieve the source from the workspace itself
- Functions to remove objects from the workspace, and to register dfns/tacit functions, will be added



How it works: Changes to Files



How it works: Changes to Files

- Link uses a Microsoft.Net FileSystemWatcher to monitor linked folders
 - This is only available under Microsoft Windows; we hope to add cross-platform support in Dyalog 18.0



How it works: Changes to Files

- Link uses a Microsoft.Net FileSystemWatcher to monitor linked folders
 - This is only available under Microsoft Windows; we hope to add cross-platform support in Dyalog 18.0
- Under program control, the link system can be made aware of changes, again by calling exactly the same code that handles FSW callbacks:
 - `SE.Link.Notify type path [oldpath]`
 - `type` is one of `created|changed|deleted|renamed`
 - `path` and `oldpath` are file names (the latter only provided if `type` is "renamed")



The -onRead and -onWrite Hooks



The -onRead and -onWrite Hooks

- You can specify the names of two functions which will be called *before* link does it's own processing



The -onRead and -onWrite Hooks

- You can specify the names of two functions which will be called **before** link does it's own processing
- This allows you to
 - Support file formats not supported natively by link, such as XML, JSON, or acre .charmat and .charvec formats
 - Perform additional processing in the workspace if configuration changes
 - ...etc...



The -onRead and -onWrite Hooks

- You can specify the names of two functions which will be called *before* link does it's own processing
- This allows you to
 - Support file formats not supported natively by link, such as XML, JSON, or acre .charmat and .charvec formats
 - Perform additional processing in the workspace if configuration changes
 - ...etc...
- The hooks can return 1 to allow link to perform default processing, or 0 if the hook has done everything that needs to be done



] link modifiers: Summary

]link ns directory

-source = { ns|dir|both}

Whether to consider the ns or dir as the source (both will synchronise)

Defaults to "both" except when linking #, when it must be specified

-watch = {none|ns|dir|both} (after initial copying, default=both)

-extn = File extension considered to be APL source code (default=.dialog)

-flatten Ignore dir hierarchy, loads everything into ns (default=off)

-prompt Prompts user to verify all synchronisation (default=off, not recommended)

-reset Removes an existing link (directory argument not required)

-onRead, -onWrite: hooks to handle files in other formats (xml, json, custom)



Purpose of the Link mechanism



Purpose of the Link mechanism

- Keep source in the workspace synchronised with files which have a similar structure



Purpose of the Link mechanism

- Keep source in the workspace synchronised with files which have a similar structure
- Changes on either side are immediately replicated on the other
 - Edit a function → file updates
 - Add a new function → new file created
 - Rename a folder → namespace renamed (except at the top level)
- Direction of synchronisation is optional
 - outbound, inbound, or bi-directional
 - inbound only available with .NET framework in v17.0



Purpose of the Link mechanism



Purpose of the Link mechanism

- NOT a source code management or project system.



Purpose of the Link mechanism

- NOT a source code management or project system.
- Expects:
 - A SCM system "below" to manage the source files (SVN, GitHub, etc)
 - A project management "on top" to manage dependencies, building, testing etc



Purpose of the Link mechanism

- NOT a source code management or project system.
- Expects:
 - A SCM system "below" to manage the source files (SVN, GitHub, etc)
 - A project management "on top" to manage dependencies, building, testing etc
- Will be available as a proper API, not just a UCMD – to be called by project mgt systems



```
]dbuild and ]dtest
```



]dbuild and]dtest

- Dyalog has used these two user command internally since v16.0



]dbuild and]dtest

- Dyalog has used these two user command internally since v16.0
- Used to build and test
 - conga.dws (in v16.0)
 - isolate.dws (from v17.0)



]dbuild and]dtest

- Dyalog has used these two user command internally since v16.0
- Used to build and test
 - conga.dws (in v16.0)
 - isolate.dws (from v17.0)
- They do not yet use]link, are still based on SALT]load
 - Outward links only
 - No automatic support for adding new functions



Demo 3:]dbuild and]dtest




```
conga-apl.dyalogbuild - Notepad
File Edit Format View Help
DyalogBuild: 0.1
ID           : CONGA, Version=3.0
Description: Conga workspace for Dyalog v16.0
Defaults    : IO+ML+1
TARGET      : Distribution/ws/conga.dws

EXEC  : EX '#.DRC' '#.Conga'

NS    : v2/DRC/*.dyalog, Target=DRC
NS    : v2/HTTPUtils/*.dyalog, Target=HTTPUtils
NS    : v2/Samples/*.dyalog, Target=Samples

NS    : v3/Conga/*.dyalog, Target=Conga
NS    : v3/IWA/*.dyalog, Target=Conga.IWA
Class : v3/LIB.dyalog, Target=Conga
Class : v3/Client.dyalog, Target=Conga
Class : v3/Connection.dyalog, Target=Conga
Class : v3/Server.dyalog, Target=Conga

Class : Common/X509Cert.dyalog, Target=DRC
Class : Common/X509Cert.dyalog, Target=Conga
CSV   : Common/ErrorTable.csv, ColTypes=2 1 1, Target=Conga.ErrorTable
CSV   : Common/ErrorTable.csv, ColTypes=2 1 1, Target=DRC.ErrorTable

LX    : +'This is the Conga v3.0 workspace.'
```

isolate

```
isolate.dyalogbuild - Notepad
File Edit Format View Help
DyalogBuild: 0.1
ID           : isolate, Version=1.1
Description: isolate workspace for Dyalog v17.0
Defaults    : (␣IO ␣ML ␣WX)+1 1 3
TARGET      : isolate.dws
LX          : #.isolate.ynys.isoStart 0

EXEC        : ␣EX '#.ll' '#.isolate'

NS          : Source/root/*.dyalog, Target=#
NS          : Source/isolate/*.dyalog, Target=#.isolate
NS          : Source/ynys/*.dyalog, Target=#.isolate.ynys

NS          : Build/*.dyalog, Target=#
EXEC        : #.Build
EXEC        : ␣EX 'Build'
```

```

|+Build;file;ver;src;t;version;rev;path;root;buildver
A As part of running isolate.dbuild, tweak the workspace a bit:
A   Build cover-functions in # and #.isolate (see function "BuildCovers")
A   Insert isolate.Version to include GIT version numbers

r+'
version+'1.1' A base version
root+ϕ{((L/ω1'/\')↓ω)ϕWSID

buildver+'16.0' A Use v16.0 or later to build
:If buildver≠(#buildver)↑2>'.'⊠WG'APLVersion'
  ⊠+'*** WARNING - Production builds should be run using Dyalog ',buildver,' ***'
:EndIf

:If 0=ρver+ϕ{0::'' ⊠ CMD'git -C "',ω,'" rev-list HEAD --count'}root
:OrIf (,1)≠1>⊠VFI ver
  ⊠+'NB: Unable to get GIT revision information!'
  ver+'0'
:Else
  ver+ver~' '
:EndIf

ver+version,','ϕ2>⊠VFIϕver A Join base version and git push count
⊠+'isolate.Version set to:'
⊠+'isolate.Version+'Version ',ver,' built at ',, 'ZI4,<->,ZI2,<->,ZI2,< >,ZI2,<:>,ZI2,<:>,ZI2'⊠FMT 1 6ρ⊠TS

```

]dbuild "commands"

Command	Use
Id, Description	Metadata
Copy (path, target)	Copy a file or folder
Run	Not yet implemented: Call another build file
NS, Class, APL	Load one or more .dyalog files
Lib	Bring in a standard Dyalog tool
CSV	Load a .CSV file into a matrix
LX, Defaults	Set latent expression, or system variable defaults
EXEC, PROD	Execute expressions (PROD = production builds only)



]dbuild modifiers

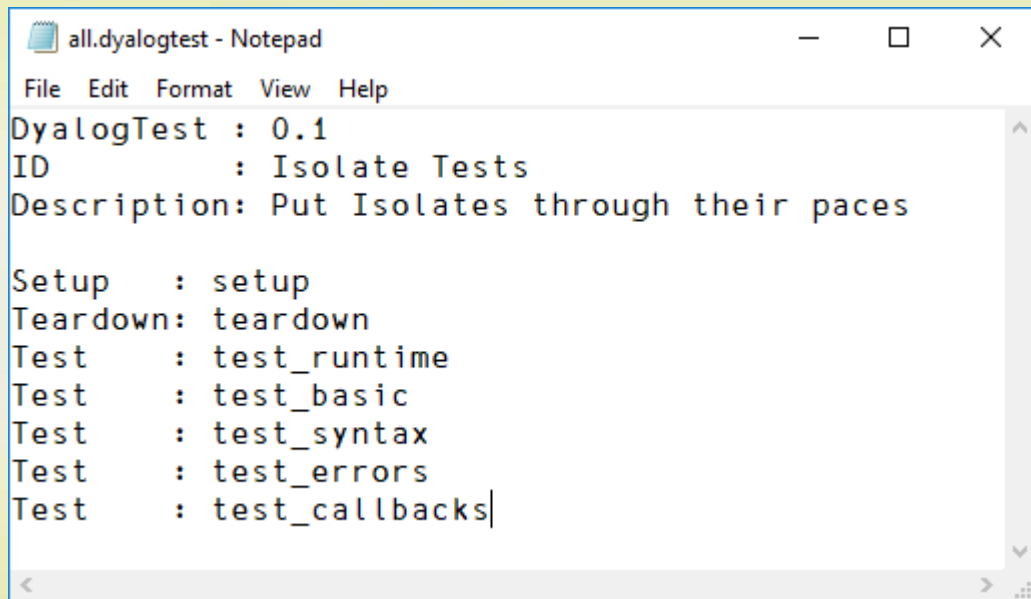
- clear[=NCs] expunge all objects,
(optionally of specified name classes only)
- production remove links to source files
- quiet only output actual errors



DTest Examples



DTest Examples



```
all.dyalogtest - Notepad
File Edit Format View Help
DyalogTest : 0.1
ID          : Isolate Tests
Description: Put Isolates through their paces

Setup      : setup
Teardown   : teardown
Test       : test_runtime
Test       : test_basic
Test       : test_syntax
Test       : test_errors
Test       : test_callbacks|
```



DTest Examples

all.dyalogtest - Notepad

File Edit Format View Help

DyalogTest : 0.1

ID : Isolate Tests

Description: Put Isolates through their p

Setup : setup

Teardown: teardown

Test : test_runtime

Test : test_basic

Test : test_syntax

Test : test_errors

Test : test_callbacks

```
C:\Dev\isolate\Tests\setup.dyalog
File Edit View Help
Search...
|+setup dummy
A Setup for isolate tests - reset any settings to defaults

:If 0=NC 'Fail' A Running v16.0 or earlier
    FX 'msg Fail value' 'msg [SIGNAL (1+value)/777]'
:EndIf

{}#.isolate.Config 'runtime' 1
{}#.isolate.Config 'onerror' 'signal'
{}#.isolate.Config 'processors' 4

r+''
Function Pos: 0/12,1
```




```

|+test_basic dummy;time;delta;is;ns;test;dfns;double
A Take futures and isolates for a little spin

{}#.isolate.Config'listen' 0
{}#.isolate.Config'processors' 4

{}#.isolate.Reset 0

test+'Basic IÏ test'
double+(w+w)#.IÏi4

[]DL 0.5
:If 2 4 6 8#double
:AndIf 2 4 6 8=="double
    Log 'Still not fixed: http://mantis.dyalog.com/view.php?id=15672'
:EndIf

double+>"double
test Fail 2 4 6 8 Check double A Remove the above :If clause when the bug is fixed

time+3=[]AI ♦ z+[]DL #.IÏ 4p1
:If 100<delta+(3=[]AI)-time A Getting futures back should take <100ms
    Log'*** WARNING: Futures took ',(wdelta),' ms to materialise.'
:EndIf
z++/z A will block
'[]DL IÏ 4p1 ran in less than 1 second'Fail 100>delta+(3=[]AI)-time

A Check that passing argument to defined functions does not block...
time+3=[]AI ♦ z+(w w)[]DL #.IÏ 1
:If 100<delta+(3=[]AI)-time A Getting futures back should take <100ms
    Log'*** WARNING: Futures took ',(wdelta),' ms to materialise.'
:EndIf

```



]dtest "commands"

Command	Use
Setup (fnlist)	Listof setups to run (all tests will be repeated for each setup)
Test (fn)	Name of a test function to run
Teardown	Function which performs any final checks and "tears down" what was set up.



]dtest modifiers

```
]DTest {ns|file|path} [modifiers]
```

```
ns      namespace in the current workspace  
file    .dyalog file containing a namespace  
path    path to directory containing functions in .dyalog files
```

Optional modifiers are:

```
-setup=fn      run the function fn before any tests  
-teardown=fn   run the function fn after all tests  
-tests=        comma-separated list of tests to run  
-filter=string only run functions where string is found in the leading @Test: comment  
-repeat=n      repeat test n times  
  
-halt          halt on error rather than log and continue  
-trace         set stop on line 1 of each test function  
  
-quiet         qa mode: only output actual errors  
-verbose       display more status messages while running
```



]dbuild /]dtest Summary

- Implemented fast to solve an urgent problem
- Very effective:
 - allowing Dyalog to use GitHub to build parts of the distribution
- Bit of a mess, refactoring will happen



Selected Dyalog Projects on GitHub

- library-core, library-conga, samples-conga (shipped folders)
- conga-apl (conga.dws)
- isolate (isolate.dws)
- link (user command)
- aplssh (Run SSH from APL)
- pynapl (Python Bridge)
- MiServer, MiSites
- JSONServer, SAWS
- WC2, Selenium
- vecdb, MatLab
- aplx (conversion tools)



GitHub



In v17.0 (on track for May/June release)

-] l i n k will be included
- □ F I X can handle functions as well as namespaces and classes
- Locals outside of line [0] to allow easier merging
- Classes will be fix'able with missing dependencies:

```
:Class MyClass : BaseClass  
:Include SomeNamespace  
:..  
:EndClass
```



The coming year...

- Work with Gil and Acre developers/users to agree on shared guidelines for using text files and packages or "modules"
- All Dyalog users **must** be able to share source files, and as many tools as possible, in the 2nd 50 years!



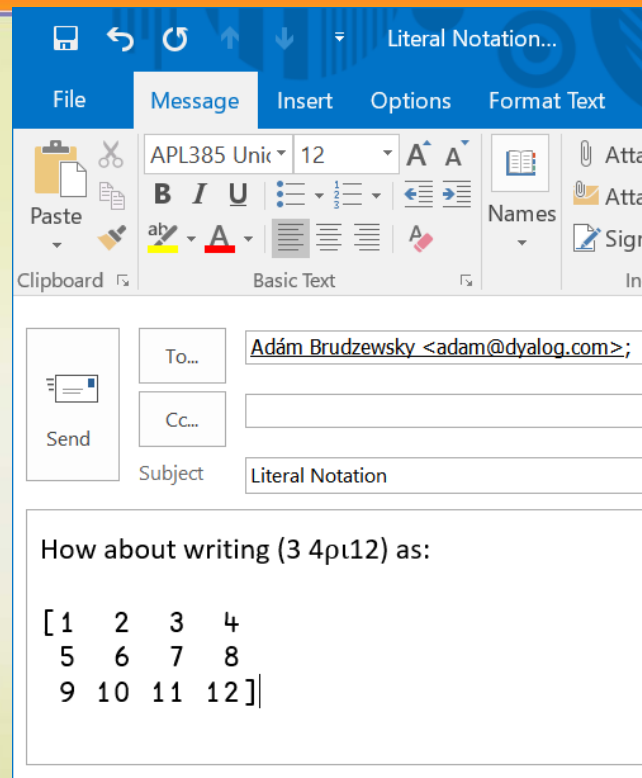
v18.0 Targets

- FileSystemWatcher on all platforms
- Implement a format for array constants in version 18.0
- APL Syntax Colouring "plugins" for GitHub, VS Code, Notepad++(?)



v18.0 Targets

- FileSystemWatcher on all platforms
- Implement a format for array constants in version 18.0
- APL Syntax Colouring "plugins" for GitHub, VS Code, Notepad++(?)



Conclusion

- It is time for the community to adopt common platforms for
 - How to represent source code
 - How to manage dependencies
- Not necessarily
 - How to manage projects
 - How to do testing



Open Source is Key

- All the tools that we develop as a community should be open source

- Over to Gil

