News from Dyalog – Spring 2022

Morten Kromberg
Agenda

- Building the Team
- The next Big Thing
- The next, next Big Thing
- Version 18.2 recap
- Version 19.0 and beyond
[Re]Building the Team

- We stand upon the shoulders of Giants

John Scholes (1948-2019)  
Roger Hui (1953-2021)
Still Going Strong…

- Geoff Streeter wrote Dyalog APL v1.0 in 1981-1983 together with John Scholes
- Geoff is in good health
  - (Now working 3-4 days / week)
  - And still volunteering at night...
Still Going Strong...

- Geoff Streeter wrote Dyalog APL v1.0 in 1981-1983 together with John Scholes
- Geoff is in good health
  - (Now working 3-4 days / week)
  - And still volunteering at night...
- BUT: He has announced that he intends to retire in April'23
Dyalog ... The Next Generation

Adam
APL
Rich
Education
Josh
Consultant
Rodrigo
Education
Stine
Accounting
Shuhao
Interpreter
Karta
Interpreter
Dyalog … The Next Generation

Adam
APL
Rich
Education
Josh
Consultant
Rodrigo
Education
Stine
Accounting
Karta
Interpreter

News from Dyalog, Spring 2022
Next Generation, Continued...

- We planned to recruit a tester and one more interpreter developer in 2022
Now, we plan to recruit a tester and **two** more interpreter developers in 2022.
Some quotes from recent applicants...
Passionate about APL, able to create non-trivial applications with it. Improving my tacit, leading-axis, and array-oriented skills daily.

Active on Stack Overflow, the APL Orchard, and in the APLFarm, I enjoy teaching, helping, and problem solving with others when given the opportunity.
I’ve recently discovered APL and other various array languages and projects around them.

Would be amazing to work for Dyalog, implementing and improving core feature set.

I’m aware don’t have great deal experience with large codebases but have enjoyed exploring Plan 9’s source code and understanding how sections fit together.
The area in computer science that interests me the most is programming language design and implementation.

I have written a few interpreters in my spare time for Prolog and for APL.

I am always interested in learning new topics in computer science and learning new ways to apply my knowledge. Last year I participated in the Dyalog APL Problem Solving Competition and won a phase 1 prize, and I am also participating this year again.
All these applicants (our two most recent hires before that) approached us to inquire about work at Dyalog.
Dyalog ... The Next Generation
The Next Big Thing

No, not more product features...

- MUCH more documentation, samples and tutorials

- (there will also be lots of features too, of course)
Training Materials

- mastering.dyalog.com
  Rodrigo Girão Serrao

- course.dyalog.com
  Rich Park

- tutorial.dyalog.com
  Gary Bergquist + Andrew Sengul

- xpqz.github.io/learnapl
  Stefan Kruger (IBM)

Adam Brudzewsky (here, there and everywhere)
The Next, Next Big Thing

- Package Management

Tatin
## Tatin Registry

### List of all packages (aggregated)

<table>
<thead>
<tr>
<th>Package name</th>
<th>Description</th>
<th>Major versions</th>
<th>Link to project</th>
</tr>
</thead>
<tbody>
<tr>
<td>allteam-APLGit</td>
<td>Git interface from Dyalog APL via Git Bash</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-APLProcess</td>
<td>Start an APL process from within Dyalog APL</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-APLTreeUtils2</td>
<td>General utilities required by most members of the APLTree library</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-APLCoverage</td>
<td>Monitors which parts of an application got actually executed</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-Compare</td>
<td>Allows comparing and merging objects in the workspace with a file or a file with another file</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-CompareSimple</td>
<td>Allows comparing objects in the workspace with a file or a file with another file</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-DateAndTime</td>
<td>Utilities related to Date and Time, including doing math</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-DocToPL</td>
<td>Zipping and unzipping with .NET Core on all major platforms</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-EVENTCodes</td>
<td>Constants with meaningful names for Dyalog error codes</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-Execute</td>
<td>Start a process from within APL</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-FitnGym</td>
<td>Utilities for doing gymnastics with files and directories</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-GitHubRespository</td>
<td>Utilities for dealing with GitHub repositories</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-HandleError</td>
<td>Allows to catch errors on an application level; saves information that allow analyzing the error</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-InItFiles</td>
<td>Allows instantiating good old ITI files in APL; comes with extended syntax supporting APL-like data structures</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-Inspect</td>
<td>Managing and displaying help pages based on markdown files</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-Logger</td>
<td>Allows writing to LOG files; (almost) guaranteed to never break the application</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-MarkToHTML</td>
<td>Converts Markdown to HTML</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-OS</td>
<td>OS-related tools for all major platforms</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-SMTP</td>
<td>SMTP client for sending emails from within Dyalog APL</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-ServicesState</td>
<td>Interface between the Windows SCM (Service Control Manager) and applications running as a service</td>
<td>2</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-SevenZip</td>
<td>Zip files with the Open Source tool 7zip</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-ShowChmFiles</td>
<td>Display CHM files under program control (Windows only)</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-Socket</td>
<td>Dyalog APL test framework</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-Vending</td>
<td>Tools for dealing with the Windows Registry</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-WinRegFlags</td>
<td>Limited set of tools for dealing with the Windows Registry</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-WinVista</td>
<td>Windows-only OS calls and system info</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>allteam-WindowsEventLog</td>
<td>Tools to read from and write to the Windows Event Log</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>dateTime</td>
<td>Easy calculations with dates</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>edit-in-fileplug</td>
<td>Extend component files to use named components</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>edit-in-SQFS</td>
<td>Easily create text SQL commands for use with any SQL program interface</td>
<td>1</td>
<td>github.com</td>
</tr>
<tr>
<td>edit-in-Tester</td>
<td>Simplified function-level testing of programs</td>
<td>1</td>
<td>github.com</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group name</th>
<th>S</th>
<th>apitam</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>davlin</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dyalog</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tiamatica</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bluedolphin</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Number of published packages of that group

Created by Tatin version 1.4.4-GIT from 2022-06-04 under Linux-64 1.6.4275.3.5 Runtime -- Bugs, questions, probate, tatin.dev@yruel.com
## Tatin Test Registry

### List of all packages (aggregated)

You may publish any packages to the Tatin Test Registry with the API key "Tatin-Test-API-Key" except for the groups "dylag" and "apitam". However, note that the data of the Test Registry is occasionally reset.

<table>
<thead>
<tr>
<th>Package name</th>
<th>Description</th>
<th>Major versions</th>
<th>Link to project</th>
</tr>
</thead>
<tbody>
<tr>
<td>aplteam-MultipleObjectsNoAPI</td>
<td>Multiple objects of all types, no API • must be a folder • Exposes everything in the folder</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>aplteam-versions</td>
<td>This and that</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td><strong>dylag-HttpCommand</strong></td>
<td>Utility to execute HTTP requests</td>
<td>1</td>
<td><a href="https://dylag.github.io">dylag.github.io</a></td>
</tr>
<tr>
<td><strong>dylag-Jarvis</strong></td>
<td>JSON and REST Web Service Framework</td>
<td>1</td>
<td><a href="https://dylag.github.io">dylag.github.io</a></td>
</tr>
<tr>
<td>example-Foo</td>
<td>Has three dependencies, one lives on a different server</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>example-Goo</td>
<td>Has NO dependencies but a file apl-dependencies.txt (empty)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>example-Moo</td>
<td>Has NO dependencies and NO file apl-dependencies.txt</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>example-loo</td>
<td>Has NO dependencies and NO file apl-dependencies.txt</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>example-versions</td>
<td>This and that</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

Created by Tatin version 0.64.0+944 from 2022-06-21 under Linux-64 18.0.40273.0 S RunTime — Bugs, questions, problems: [tatin.dev@gmail.com](mailto:tatin.dev@gmail.com)
The other Big Thing

- Package Management

- Demo tomorrow
Recap of v18.2  (March 2022)

- Windows Patches
- Basic Licence
- Launching from Text Files:
  - Shell Scripts
  - Text Source
  - LOAD Parameter
  - Windows Explorer Integration
- Improved Docker Support
- RIDE v4.4
- Developer Productivity:
  - JSON tables/datasets
  - ⎕ATX metadata one-stop-shop
  - Generating random numbers
  - New user commands
Windows Patches

Updates delivered as Microsoft Patch files (MSP) under Windows

- Allows patching ALL components of Dyalog installation including APL code & docs
- All components were already patched by re-installing on non-Windows platforms
Welcome to MyDyalog

Welcome to MyDyalog. Please note that the top of the page to access the part of the site that you want to view. Notices will be posted to this page when there are major site changes, such as a new release, or for an add-on package (for example, SQAPL/Conga).

DFS (Dyalog File System) cannot currently be downloaded using MyDyalog; if you have a DFS licence and want to download it, please contact sales@dyalog.com

Latest News

18.2 is released to all users

23rd March 2022

Dyalog APL Version 18.2 is now available for DSS users. It is available for all platforms.

The installation images have been updated to remove a problem with the LOAD= parameter; more details appear in the DSS email. The Windows patch files include the fix for the issue, so if you already have 18.2 installed you can simply download them and apply the patch. The patch wizard files do not include the entire fix, so we strongly recommend using the patch files.

Note that for 18.2 onwards on Windows we will be issuing patches for all the files that are part of Dyalog APL, as well as still supported the old patch wizard. These are available from the Update Existing Windows Installations page.
18.2 is released to all users

23rd March 2022

Dyalog APL Version 18.2 is now available for DSS users. It is available for all platforms.

The installation images have been updated to remove a problem with the LOAD= parameter; more details appear in the DSS email. The Windows patch files include the fix for the issue, so if you already have 18.2 installed you can simply download them and apply the patch. The patch wizard files do not include the entire fix, so we strongly recommend using the patch files.

Note that for 18.2 onwards on Windows we will be issuing patches for all the files that are part of Dyalog APL, as well as still supported the old patch wizard. These are available from the Update Existing Windows Installations page.
Latest News

18.2 is released to all users

23rd March 2022

Dylog APL Version 18.2 is now available.

The installation images have been updated to remove a problem with the LOAD= parameter, more details appear in the DSS email. The Windows patch files include the fix for the issue, so if you already have 18.2 installed you can simply download them and apply the patch. The patch wizard files do not include the entire fix, so we strongly recommend using the patch files.

Note that for 18.2 onwards on Windows we will be issuing patches for all the files that are part of Dylog APL, as well as still supported the old patch wizard. These are available from the Update Existing Windows Installations page.

https://my.dylog.com/#DownloadPatches
Basic Licence

Possibly the most important v18.2 "feature"!

Intended for:
- non-commercial use
- education
- personal projects
- experiments
- sharing your experience
- proof of concepts / trials
- participating in programming competitions for cash prizes
- fun
Basic Licence

- Replaces *non-commercial licence*

- Allows distribution of Dyalog along with your work, under the default royalty licence
  - Fee is 2% of gross APL-based revenue
  - No fees due if revenue < GBP 5,000 in a calendar year
  - Multiple alternative commercial licence schemes are available

- For GBP 150 per year, you can subscribe to the Dyalog Support Service (DSS)
Launching from Text Files

Another important 18.2 theme ...

- Shell scripts
- LOAD= parameter supports starting the interpreter directly from text source
- Link properly settling down with v3.0
- Windows Explorer Integration
- Improved Docker support

... demo tomorrow ...
RIDE v4.4

- Debug multi-threaded applications
- Language bar and Help adapts to Dyalog version
- Save responses to prompts
- ... and many more tweaks and fixes, see https://github.com/Dyalog/ride/milestone/4?closed=1
A Convert "Inverted" Matrix to JSON

```
fields ← 'Item' 'Price' 'Qty'
items ← 'Knife' 'Fork'
price ← 3 4
qty ← 23 45

fields⍪⍉↑items price qty
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knife</td>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>Fork</td>
<td>4</td>
<td>45</td>
</tr>
</tbody>
</table>

inverted ← 4 A Data Format Code
data ← items price qty
table ← ⊂ inverted (data fields)
(⎕JSON⍠'Compact'0)table

```
[{
  "Item": "Knife",
  "Price": 3,
  "Qty": 23
},
{
  "Item": "Fork",
  "Price": 4,
  "Qty": 45
}
]```
Developer Productivity:

- Unifies:
  - \( \square SIZE \) Size of Object
  - \( \square NC \) Name Classification
  - \( \square NR \) Nested Representation
  - \( \square SRC \) Source
  - \( \square AT \) Attributes
  - \( 5179\square \) Loaded Object Info

- Additional metadata:
  - Last edit UTC
  - Source "as typed"
  - Normalised source
  - Best available source
  - More to come...
### Extended Attributes

This function provides information about a name in a workspace, including its usage, history, restrictions, role and origin.

**Note:** To retrieve this information for an unnamed value, wrap `DATX` in a dfn and use the name `w`, for example `{60DATX'w'}`

Y can be a simple character scalar, a simple or enclosed character vector, or a vector of character scalars and vectors (as least one must be a character vector) of the name(s) for which information is required.

X can be a scalar or a vector indicating the information required:

<table>
<thead>
<tr>
<th>Group</th>
<th>X</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identity</strong></td>
<td>0</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td>10</td>
<td>Function result (0: none or not a function, 1: explicit, 2: shy)</td>
<td>0</td>
</tr>
<tr>
<td>Syntax</td>
<td>11</td>
<td>Function valence (0: niladic, 1: monadic, 2: dyadic, 3: ambivalent)</td>
<td>0</td>
</tr>
<tr>
<td>Syntax</td>
<td>12</td>
<td>Operator valence: (0: not an operator, 1: monadic, 2: dyadic)</td>
<td>0</td>
</tr>
<tr>
<td>Last edit</td>
<td>20</td>
<td>Author of last edit</td>
<td>**</td>
</tr>
<tr>
<td>Last edit</td>
<td>21</td>
<td>Number of days passed between 1899-12-31 at 00:00 UTC and last edit (includes fractional days)</td>
<td>0</td>
</tr>
<tr>
<td>Last edit</td>
<td>22</td>
<td>Local timestamp at last edit format is the 7-item vector described by DTS. See Timestamp.</td>
<td>9</td>
</tr>
<tr>
<td>Last edit</td>
<td>23</td>
<td>Number of bytes required for storage without sharing</td>
<td>0</td>
</tr>
<tr>
<td>Restrictions</td>
<td>30</td>
<td>Source can be displayed</td>
<td>°°°</td>
</tr>
<tr>
<td>Restrictions</td>
<td>31</td>
<td>Execution can be suspended mid-execution</td>
<td>°°°</td>
</tr>
<tr>
<td>Restrictions</td>
<td>32</td>
<td>Responds to weak interrupt</td>
<td>°°°</td>
</tr>
<tr>
<td>Class</td>
<td>40</td>
<td>Syntactic supra-class (1: invalid name, 0: undefined, 1: label, 2: variable, 3: function, 4: operator, 5: event, 6: object)</td>
<td>°°°</td>
</tr>
<tr>
<td>Class</td>
<td>41</td>
<td>Syntactic sub-class (0: none, 1: traditional/plain, 2: field/dynamic/instance, 3: property/derived/primitive, 4: class, 5: interface, 6: external, 7: external interface)</td>
<td>0</td>
</tr>
<tr>
<td>Class</td>
<td>42</td>
<td>Full syntactic class (sum of supra- and sub-class)</td>
<td>°°°</td>
</tr>
<tr>
<td>Source</td>
<td>50</td>
<td>File name</td>
<td>°°°</td>
</tr>
<tr>
<td>Source</td>
<td>51</td>
<td>File encoding</td>
<td>°°°</td>
</tr>
<tr>
<td>Source</td>
<td>52</td>
<td>File checksum</td>
<td>°°°</td>
</tr>
<tr>
<td>Source</td>
<td>54</td>
<td>Definition's offset from top</td>
<td>0</td>
</tr>
<tr>
<td>Source</td>
<td>55</td>
<td>Number of lines in definition</td>
<td>0</td>
</tr>
<tr>
<td>Definition</td>
<td>60</td>
<td>Verbatim source (as typed)</td>
<td>0pe+°°</td>
</tr>
<tr>
<td>Definition</td>
<td>61</td>
<td>Normalised source (with AUTOFORMAT=1 and TAB_STOPS=4)</td>
<td>0pe+°°</td>
</tr>
<tr>
<td>Definition</td>
<td>62</td>
<td>Most precise available source (verbatim with fallback to normalised)</td>
<td>0pe+°°</td>
</tr>
</tbody>
</table>
Developer Productivity:
Generating random numbers

```
Dist ← 16808⍠
val ← 2 5 Dist 'Beta' 1e5
res ← 100
bin ← ⌽res×val
x ← ¯1+1res
y ← {≢⍵}⌸x,bin
]plot y x
```
New Win32 GUI (⎕WC) features!!!

- `CornerTitleBCol` on Grid
- `FireOnce` on Timer
Developer Productivity:

New user commands
Overview

]OUTPUT.View
]WS.Names
]TOOLS.APLCart
]OUTPUT.Repr
]EXPERIMENTAL.Get
Service Orientation

A rapidly increasing share of new APL code (and some old code) is exposed as services:

- **Jarvis** wraps APL code as HTTP/JSON or RESTful services on any platform
  - https://github.com/dyalog/jarvis
- Version 18.2 is easier to deploy and debug as a “headless” service
  - Allows improved docker containers
Summary – Version 18.2

- Windows Patches
- Basic Licence
- Launching from Text Files:
  - Shell Scripts
  - Text Source
  - LOAD Parameter
  - Windows Explorer Integration
- Improved Docker Support
- RIDE v4.4
- Developer Productivity:
  - JSON tables/datasets
  - ⎕ATX metadata one-stop-shop
  - Generating random numbers
  - New user commands
Version 19.0 and Beyond

Everything mentioned in the following is a statement of intent...

- Version 19.0 release target = 1 year after 18.2, or March 2023
Planned Features of v19.0

- .NET 6 bridge
- I/O re-implementation
- Improve Installers
- Keyboarding on all platforms
- Arm64 (M1 Mac & Pi)
- Monitoring / Heath Checks
- Optimisation work
- Portable File Functions
- HTMLRenderer improvements
- Session initialisation
- Configuration
.NET 6 Bridge

- Export APL code as .NET assemblies
- Create executables under
  - Linux: Amd/Intel x64 and Pi on Arm64
  - macOS (x64 and M1)
  - Windows (x64 – maybe Arm64)
  - (maybe Android on Arm64 soon)

V19.0 should also support .NET 3.1, 5 and 7
.NET 6: Asynchronous Features

- Asynchronous function calls have appeared in many languages
  - C# / .NET use is widespread
- V19.0 work on the .NET bridge will hopefully include Async support
- "futures" may be a natural solution
  - Already used by isolates for asynch processing
I/O re-implementation

(completion of support for shell scripts)

- Allow control over whether I/O streams are connected to RIDE or redirected files
- (allows RIDE debugging of APL shell scripts)
Installation Issues

- Linux-style installation for macOS
  - (most Mac APL users are really using the Mac as a very cool UNIX workstation)

- Remove "termcap" dependency
  - (Varies from one Linux to the next)

- Make HTMLRenderer easily upgrade-able
  - (new version of Chromium without reinstalling APL)
Quite simply: We need to review how APL symbols are entered into APL - and other applications

- The way Windows handles keyboards is changing – IME no longer works with all Windows 10 apps
- "ctrl" as the APL key is more and more problematic
- We need to provide more easily configureable options, and much better documentation
64-bit ARM chips are appearing in important places that Dyalog should support:

- New "M1" Macs
- Raspberry Pi – 64 Bit

We will also look at, but are unlikely to add official support in v19.0

- ARM based Windows
- Android
Health Checks

Tools to monitor the state of APL processes
- CPU consumption
- Memory usage, Compaction counts, etc
- Are any threads suspended?
- )SI and Error information
- Available via API and or protocols like SNMP

Important as APL becomes more widely used to implement multi-process services
Optimisation work

- Identify and carefully implement most valuable (disabled) v18.0 optimisations
- Continue work on adding new idioms, special cases of rank and other operators

(resume steady performance work, following new process)

PLEASE send us your application benchmarks!
[Portable] File Functions

- ⎕NCOPY to allow a regular callback, so you can cancel long-running ops
- ⎕NATTRIBUTES to set e.g. timestamps, read-only/hidden and other attributes
- ⎕NINFO recursion depth option
HTMLRenderer improvements

- Find a way to easily upgrade the version of the Chromium Embedded Framework
  - Possibly turn the HTMLRenderer into an Open Source project
- Multiple Modal Instances
  - (only allow access to one window if required)
- Thread switching in OnHTTPReceive and OnWebSocketReceive to allow threaded GUI apps
Session initialisation mechanism

- Re-implement the way that the session object (⎕SE) is populated at startup
- Completely separate Dyalog content from user content with no risk of conflict
  - (and no need to reinstall own content when APL is upgraded)
- Design a declarative way to add user-defined menu[items] and widgets
- User Command implementation will also be reviewed
Configuration

Move towards portable configuration

- Remove need for a startup script under 'NIXes
  - Good defaults for all settings
  - All settings configurable via text files
- Remove need for the Windows Registry
  - (Windows APL already has good defaults)
  - Move all "interpreter settings" to config files
  - Leave IDE settings in the Registry (as RIDE settings are in a JSON repository)

This might not make 19.0, but remains an important short term goal.
Recap - Planned Features of v19.0

- .NET 6 bridge
- I/O re-implementation
- Improve Installers
- Keyboarding on all platforms
- Arm64 (M1 Mac & Pi)
- Monitoring / Heath Checks

- Optimisation work
- Portable File Functions
- HTMLRenderer improvements
- Session initialisation
- Configuration
Improving "Cross Platform" capability & similarity

- .NET 6 bridge
- I/O re-implementation
- Improve Installers
- Keyboarding on all platforms
- Arm64 (M1 Mac & Pi)
- Monitoring / Heath Checks

- Optimisation work
- Portable File Functions
- HTMLRenderer improvements
- Session initialisation
- Configuration
Beyond 19.0

- Array Notation
- Multiple numeric towers
- Under ¯
- Simple cross-platform user interfaces
Array Notation

\[
\begin{bmatrix}
\text{'Jan'} & \text{'Feb'} & \text{'Mar'} \\
(101, 102, 103, 104) & (201, 202, 203, 204) & (301, 302, 303, 304)
\end{bmatrix}
\]
Under ⍺

Computational Under: Mathematical definition

\[ \alpha f \circ g \omega \leftrightarrow (g^{-1}) (g \alpha) f (g \omega) \]

2 3 4 +⍥ 2
4 6 8

Structural Under: Less strict, but "more useful"

1 ⎕C⍥(⊃¨) 'adam' 'karta' 'nathaniel'
Adam Karta Nathaniel

Also to come...

Obverse: FFT⍢IFT
(declared inverse)

Depth: f ⍺n
(f on depth-n sub-arrays)
Enhancing APL: Multiple Towers

Goals:
- Better control of DECIMAL data type
- Accommodate INT64 and RATIONAL types

Model: Three Towers, selectable **per array**
- Fast: BOOL, INT 8-16-32, DOUBLE, COMPLEX
- Decimal: BOOL, INT 8-16-32-64, DECF
- Unlimited: RATIONAL

Significant design remains to be done on notation, rules for comparison and promotion. What is:
\[(\nu 0.1D, (1 \div 10), 1R10)\]
Quest for the Holy Grail

Lucasfilm, Ltd.
Simple Cross-Platform User Interfaces

'hr' ⍴WC 'HTMLRenderer' 'Hello <b>World</b>''

WE’VE ALREADY GOT ONE!

OH, YES. IT’S VERY NICE!
Simple Cross-Platform User Interfaces

'hr' ⊂WC 'HTMLRenderer' 'Hello <b>World</b>'

- The challenge is generating the HTML/JS...
- Our "DUI" has not yet caught on
- Perhaps the community can help?
GitHub: the-carlisle-group / Abacus

- Project to generate HTML from simple APL code

```apl
Run←{
   THIS.H←##.Main
   p←NS''
   p.Caption←'Sample Form'
   p.Title←'A Sample Dialog Box'
   p.Subtitle←'This is a resizable, modal dialog box.'
   p.OnOK←'OnOK'
   p.OnCancel←'OnCancel'
   p.Content←##.Grid.New GridDataLarge ø
   ##.DialogBox.Run p
}
```
Web Assembly (WASM)

We could compile Dyalog APL to run in the browser.
(some features would need to be disabled)

WebAssembly (abbreviated Wasm) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.
Conclusion

- Although improving documentation, samples and training materials is at least as important
- ... and Packages ...
- ... we are not about to run out of ideas for ways to improve Dyalog APL and it's tooling