DYALOG

APL Germany e.V.

Solingen, November 5th 2026

# The Road Ahead

*Morten Kromberg, CTO*

DYALOG

APL Germany e.V.

Solingen, November 5th 2026

# The Road Ahead



*Morten Kromberg*
*CTO*

DYALOG

APL Germany e.V.

Solingen, November 5th 2026

# The Road Ahead




*Morten Kromberg*
*CTO*

# Large Language Models / AI



No presentation can be complete without mention of Artificial Intelligence

# Large Language Models / AI



Nobody can afford to ignore LLMs.

# Large Language Models / AI



We're taming the elephants ☺

# Recent Uses of AI at Dyalog

- More, better testing

**AT DYALOG '26**
**(videos published from mid-November)**

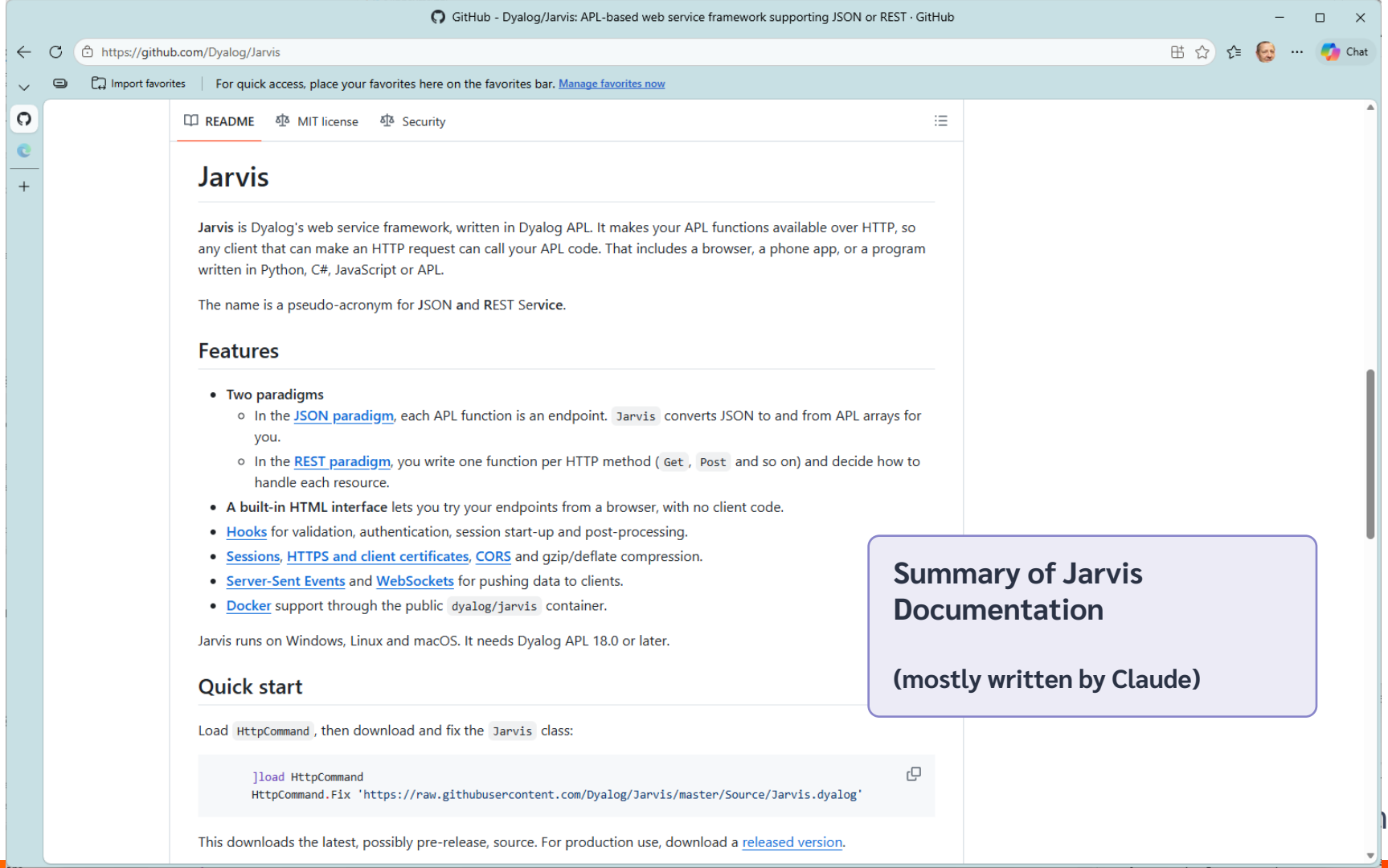
**D19: Testing the Untestable**
Aarush Bhat

**U04: Why CI is (Also) Important in the Age of D-AI-alog**
Lars Stampe Villadsen



# Recent Uses of AI at Dyalog

- More, better testing
- More, better documentation and samples




**Summary of Jarvis Documentation**

**(mostly written by Claude)**

# Recent Uses of AI at Dyalog

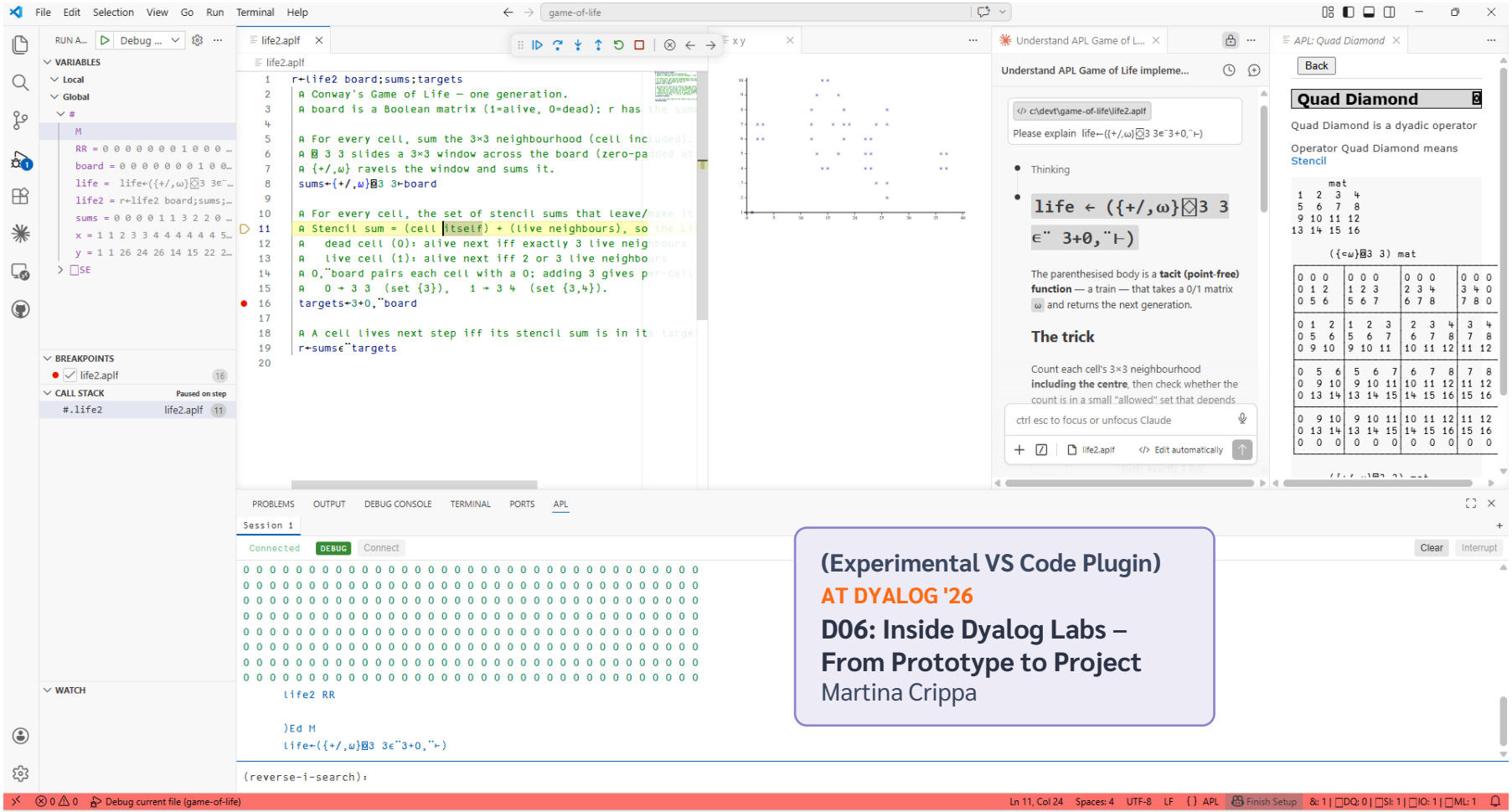
- More, better testing
- More, better documentation and samples
- Analysis of bugs in interpreter, RIDE, DFS – and proposed fixes

Examples of issues found by Claude



# Recent Uses of AI at Dyalog

- More, better testing
- More, better documentation and samples
- Analysis of bugs in interpreter, RIDE, DFS – and proposed fixes
- Prototyping new language and tool features


(Experimental VS Code Plugin)
AT DYALOG '26
D06: Inside Dyalog Labs – From Prototype to Project
Martina Crippa

# Recent Uses of AI at Dyalog

- More, better testing
- More, better documentation and samples
- Analysis of bugs in interpreter, RIDE, DFS – and proposed fixes
- Prototyping new language and tool features
- Work on billable consulting projects

Editor - [FsInvoke in C:/devt/metsim/dyalog/FsInvoke.aplf]

File Edit Syntax Window Refactor View

Search...

```
[0]     r←name FsInvoke args;i;tbl;names;sigs;err;dll;rc;msg;z;len;buf
[1]   ⍝ Invoke a METSIM_link64.dll entry point by name.
[2]   ⍝ name is a C export name from docs/METSIM_link.h. Sections 1-5 (the
[3]   ⍝ supported call sequence) are listed first; legacy/diagnostic (§7) and
[4]   ⍝ removed (§8) entries follow, each commented with its status - see the
[5]   ⍝ per-line comments below before using one of those. The JSON entries (§6)
[6]   ⍝ are excluded entirely: the header says outright not to build against them.
[7]   ⍝ args is passed as the right argument to the ⎕NA'd function: a scalar for
[8]   ⍝ a single-parameter entry point, or a strand of values in parameter order
[9]   ⍝ for the rest, matching that entry point's C signature.
[10]  ⍝ Returns whatever the DLL function returns (0 = success, non-zero = error).
[11]
[12]    :If 3≠⎕NC name
[13]        tbl←['METSIM_INIT_SIMULATION'        '=I4 <0T1'
[14]             'METSIM_TERM_SIMULATION'        '=I4'
[15]             'METSIM_GET_DLL_VERSION'        '=I4 =I4 =I4'
[16]             'METSIM_SET_DEBUG_LEVEL'        'I4'
[17]             'METSIM_SET_UNITS'              '<0T1 <0T1 <0T1 <0T1 <0T1'
[18]             'METSIM_SET_INFO'               'I4 I4 I4'
[19]             'METSIM_SET_COMPONENTS'         '<0T1 <0T1 <0T1 <I4[] <F8[] <F8[] <F8[] <F8[] <F8[] <F8[]'
[20]             'METSIM_GET_INFO'               '=I4 =I4 =I4'
[21]             'METSIM_GET_COMPONENTS'         '=0T1 =0T1 =I4[] =I4[] =I4[] =F8[] =F8[]'
[22]             'METSIM_ADD_REACTOR'            'I4 <0T1'
[23]             'METSIM_DELETE_REACTOR'         'I4'
[24]             'METSIM_SET_REACTOR_CONDITIONS' 'I4 <F8[]'
[25]             'METSIM_GET_NUM_REACTORS'       '=I4'
[26]             'METSIM_GET_REACTOR_NUMS'       '=I4[]'
[27]             'METSIM_SET_HEAT_BALANCE'       '<I4[] <F8[] <F8[]'
```

# Recent Uses of AI at Dyalog

- More, better testing
- More, better documentation and samples
- Analysis of bugs in interpreter, RIDE, DFS – and proposed fixes
- Prototyping new language and tool features
- Work on external consulting projects
- Drafting policies

# Dyalog Practices and Procedures

## Use of AI and LLMs

## Version 0.5 - 13 April 2026

[P

NOTE: This process is security r

Do NOT edit this wiki page; write

**Contents** [hide]


1 Purpose and Scope
2 Core Principles
  2.1 Personal Accountability
  2.2 Content Verification
  2.3 Legal Compliance
  2.4 Images, Audio and Video
  2.5 Security Considerations
  2.6 Transparency
3 Permitted Inputs
  3.1 Dyalog Proprietary Code
  3.2 Public Information


## Purpose and Scope [edit | | edit source]

This policy establishes guidelines for the responsible use of AI tools and Large Language Models (LLMs) at Dyalog Ltd. It applies to all employees and contractors using AI tools for any work-related purpose.

**NB:** The available tools and the legal aspects surrounding them are in a state of flux. As a result, the policy may change at short notice. If you are considering a fundamentally new use of AI, discuss it with ai@dyalog.com.

AI tools are a valuable part of our working toolkit. We encourage their use for drafting documents, restructuring materials, writing internal scripts, exploring ideas, code, and improving day-to-day productivity. We have official team plans available for approved AI services - contact ai@dyalog.com to get set up.

Building practical AI literacy across the company is important: our customers are already using AI and looking to us for informed guidance. The more experience we have, the better we can advise them.

The use of AI or LLMs to write source code that ends up in products or services that we make available to clients is more controlled and **requires** a discussion with security@dyalog.com before proceeding.

All questions, concerns, or worries that AI is being misused should be directed to the same group.

## Core Principles [edit | | edit source]

### Personal Accountability [edit | | edit source]

- You are personally responsible for all AI-generated content you use, publish, or incorporate into any work product. You must verify and approve all AI-generated outputs before usage, especially in outward facing contexts.

# Dyalog Internal AI Policy

We conduct regular internal workshops on safe use of LLMs in accordance with the policy. Key points:

- No personal or customer data may be exposed to an AI without explicit permission
- Free AI tools without protections are prohibited, except for experimentation with formal approval
- Anyone wishing to publish or make long-term use of LLM generated materials must take FULL responsibility for the content
  - Even before sending it to a colleague to read



# Thank You Mike!

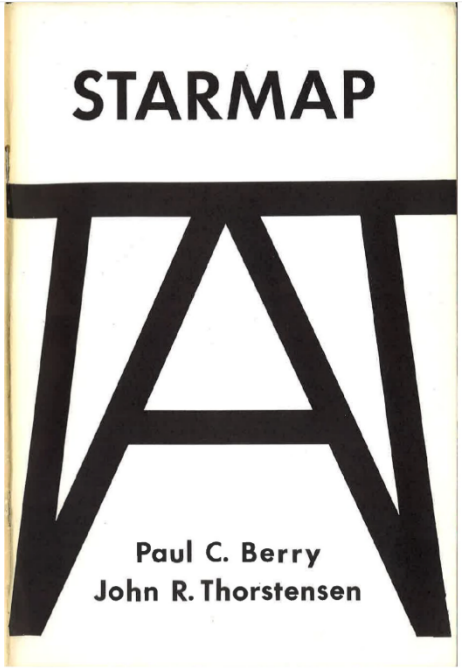





Mike Mingard

# Recent Uses of AI at Dyalog

- More, better testing
- More, better documentation and samples
- Analysis of bugs in interpreter, RIDE, DFS – and proposed fixes
- Prototyping new language and tool features
- Work on external consulting projects
- Drafting policies
- "Recovery projects"




- Descriptions of the calculations
- The relevant APL
- The result of plotting

## Calculating the Positions of the Planetary Bodies

The function *CALCULATEPLANETS* finds *PLANETS*, a table of the positions of the sun, moon, and planets at the desired date. When first calculated by the function *PLANETPOS*, these positions are stated in 3-dimensional heliocentric Cartesian coordinates. But the function *EARTHVIEW* converts them to geocentric polar coordinates (right ascension in hours, declination in degrees, and distance in astronomical units), locating the planets with respect to the center of the Earth.

In order to plot the sky above a particular place, the function *SKYPOS* (see p. 29) is used to calculate *AA*, a table of altiude and azimuth with respect to given time and location on the Earth's surface. The function *VISIBLE* is used to select from *P* those members that are above the horizon, and saves them in the table *AAP*. Finally, the *PROJECTION* of these coordinates onto a flat surface is calculated, and translated to the Cartesian form expected by the plotting function; the function *IF* is simply a compression of the left argument by the right, along the first axis, defined by the APL symbol /.

```
∇ CALCULATEPLANETS; AA; MOON; SUN; KOHOUTEK
  PLANETCOORD←AAP←PLANETS←VP←⍳0
  PLANETS←DATE EARTHVIEW DATE PLANETPOS (3≠⍳9)/PLANETS
  SUN←DATE EARTHVIEW 0 0 0
  K←100≥|DATE-JNU 12 28 1973
  KOHOUTEK←DATE EARTHVIEW (DATE IF K) COMETPOS KOHOUTEK
  MOON←MOONPOS DATE
  PHASE←MOON[1;1] MOONPHASE SUN[1;1]
  PLANETS←MOON,[1] SUN,[1] PLANETS,[1] KOHOUTEK
  MOON←MOON[;3] PARALLAXADJUST (LAT,DATE,TIME) SKYPOS MOON
  AA←MOON,[1] (LAT,DATE,TIME) SKYPOS 1 0+PLANETS
  PLANETCOORD←MAPCARTESIAN PROJECTION AAP←AA IF VP←VISIBLE AA
```

Execution of *CALCULATEPLANETS* causes new values to be assigned to four global variables. (These are initially set to ⍳0 in the first statement, mainly to draw attention to a list of the gobal variables which will be reset as a consequence of executing this function.) The four are:

*PLANETS* — The right ascension and declination of the moon, sun, planets, and Kohoutek.

*VP* — A logical vector indicating which planets are visible from the place, date, and time requested.

*AAP* — The altitude and azimuth of the visible planets.

*PLANETCOORD* — The Cartesian coordinates used to plot the projection of the visible planets.



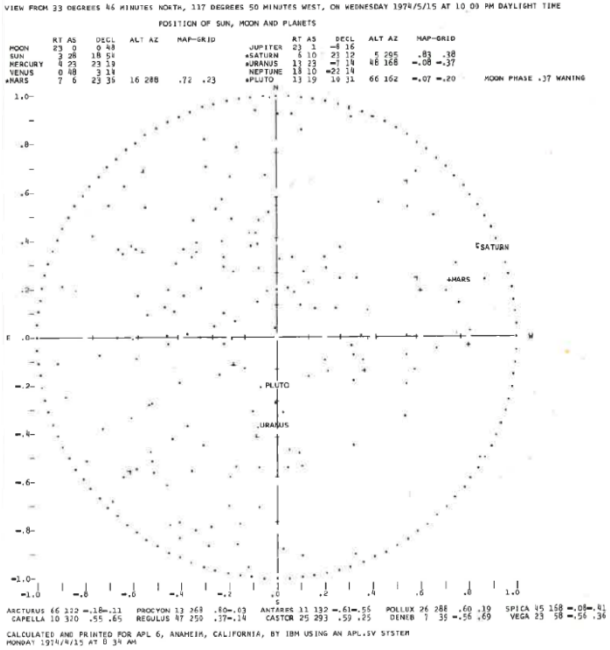



Fig. 2 Sample chart produced by the STARMAP workspace

# STARMAP animation generator

Start date

15/05/1974

End date

12/06/1975

UTC time (hours, decimal)

5

e.g. 5 = 05:00 UTC = 10 PM PDT

Frames per second

15

Latitude

52.526

Longitude (east +, west −)

+0.388

Large display

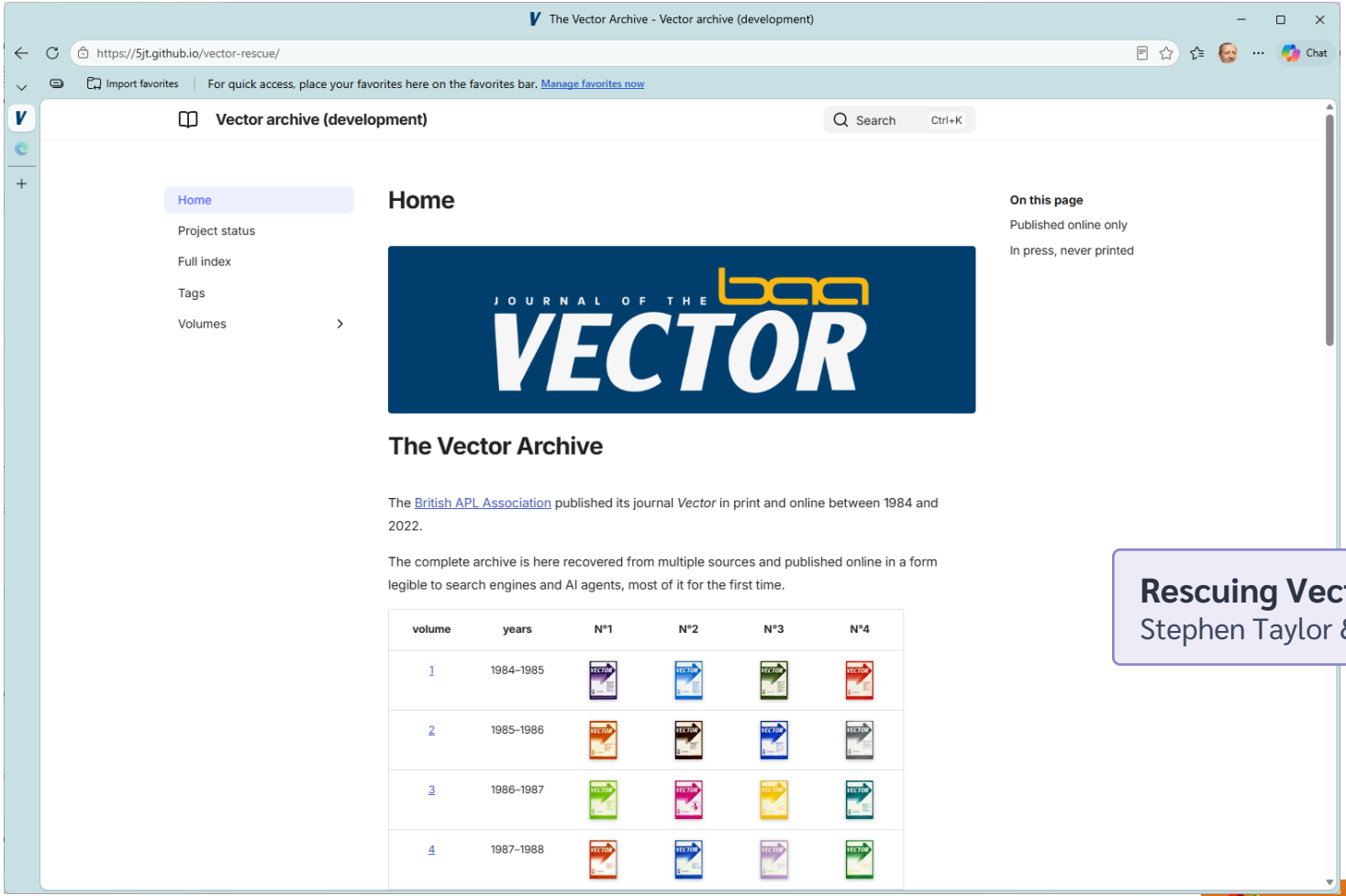
Reduce size (every other day)

Generate SVG

Generate .mp4 (slow)

Neil Kirsopp's presentation on Starmap can be found at:

https://www.dyalog.com/uploads/files/presentations/aplgermany_2026_04_AWebInterfaceForSTARMAP.pdf




**Rescuing Vector**
Stephen Taylor & Claude

Home
Project status
Full index
Tags
Volumes
Comparison with the archive

# Report: Dyalog APL Users Meeting, Elsinore, October 2005

*reported by Adrian Smith (adrian@apl385.com)*

Vector 22:1, page 25

conference reports APL community

## Location

Architecture and modern art, sun, sea, woodlands, knife-throwing, 9-pin bowling ... need I go on? This was Denmark at its best, and almost everyone there really enjoyed themselves. LO-skolen was built by the Danish Trades Union movement in the 60s, as a showpiece of all that is most characteristically Danish. The whole place is plain, simple, beautifully crafted out of good materials and just feels 'right'.



On this page
Location
Conference Highlights
Day-1 Wednesday October 19th
Welcome and overview of Version 11 *Gitte and Morten*
Hologram – Intelligent Business *John Miller*
Version 11 language extensions *John Scholes*
Version 11 object-oriented features *John Daintree*
A keyboard viewer for Dyalog APL *Kai Jäger*
Is that an APL system in your pocket? *Richard Procter*
From Dyalog to pure .Net runtime *Adrian Smith*
The PX-Edit statistical table editor *Veli-Matti Jantunen*
After dinner entertainment – Lists *John Scholes*
Homage to Ken Iverson
Day-2 Thursday October 20th
Tutorial – introducing OO to APLers *Morten Kromberg*
Using APL-based extensions in ASP.Net *Pertti*




- Significant Productivity Gains
- Quality Improvements
- Projects we would never have attempted
- We're just getting started!

# Impact of AI on Road Map



# Impact of AI on Road Map

New "opportunities":

- Fixing new-found vulnerabilities
- Projects we previously thought were "too hard"
- More power to the younger team members



Focus on Making Dyalog APL more LLM-friendly:

- Documentation in machine-friendly formats
- Making the interpreter easy to call
  - From the command line / via a socket / using scripts
- Integration with VS Code or other IDEs



# Road Map Focus Areas

1. Deployment / Operations
2. Developer Productivity
3. APL Language
4. Migration Tools
5. Data Science – Driven Work



# 1. Deployment / Operations

- Calling APL as a service
  - Jarvis with SSE & WebSocket Support
  - Stark: Auto-Generate OpenAPI Specs
  - Designing "APL as a Socket Server" 

AT DYALOG '26

**D08: Web Services Overview**
Brian Becker

**D09: Making Web Services More Accessible**
Holden Hoover

# Deployment / Operations

- Calling APL as a service 
- Thread Scheduling & Async: Scalability
- Static Analysis of APL Code
- Continuously Increasing Testing
- Encrypting workspaces and component files?

AT DYALOG '26

**D04: Improving APL Threads**
Peter Mikkelsen

**D11: Dyalog Static Analysis Tool – In Practice and the Future**
Brandon Wilson

**D19: Testing the Untestable**
Aarush Bhat

# 2. Developer Productivity



# 2. Developer Productivity

- Calling APL as a service
- Rolling out Array Notation



# Array Notation (v20.0)

```
    RC←[0 'OK'
        1 'WS FULL'
        2 'SYNTAX ERROR'
        3 'INDEX ERROR'
        4 'RANK ERROR']

    ⍴RC
5 2

    person←(name:'Morten' ⋄ height:175)

    person.(name height)
 Morten  175
```

Creates a **5×2 nested matrix**

Creates a **namespace** containing two variables

# Taking Full Advantage of Array Notation (v22.0)

- ⎕REPR
  - "Format" an array using Array Notation
- ⎕EVAL
  - Safely Execute Array Notation (only define constants)
- Literal Namespaces as alternatives to Name/Value Pairs

Replace current APL models

# Replacing Name + Value Pairs

```
⎕WC  'Timer' ('Active' 0)('Interval' 500)
```

...could be written using literal namespaces...

```
⎕WC  'Timer' (Active:0 ⋄ Interval:500)
```

Similarly:

```
(U:0 ⋄ J:1) ⎕FPROPS tn
⎕SIGNAL(EN:11 ⋄ EM:'USER ERROR' ⋄ Message:'Oops!')
```

# Namespaces as Arguments

```
    ∇ r←Range params;from;step;to
[1]   (from step to)←3↑params
[2]   r←from,from+step×⍳step÷⍨to-from
    ∇

     Range 0 6 42
0 6 12 18 24 30 36 42
```

# Namespaces as Arguments

```
    ∇ r←Range params;from;step;to
[1]   (from step to)←params ⎕VGET ('from' 0)('step' 1)'to'
[2]   r←from,from+step×⍳step÷⍨to-from
    ∇

     Range(to:42 ⋄ step:6)
0 6 12 18 24 30 36 42
```

# Namespaces as Arguments

```
    ∇ r←Range params;from;step;to
[1]   (from step to)←params ⎕VGET ('from' 0)('step' 1)'to'
[2]   r←from,from+step×⍳step÷⍨to-from
    ∇

     Range(to:42 ⋄ step:6)
0 6 12 18 24 30 36 42
```

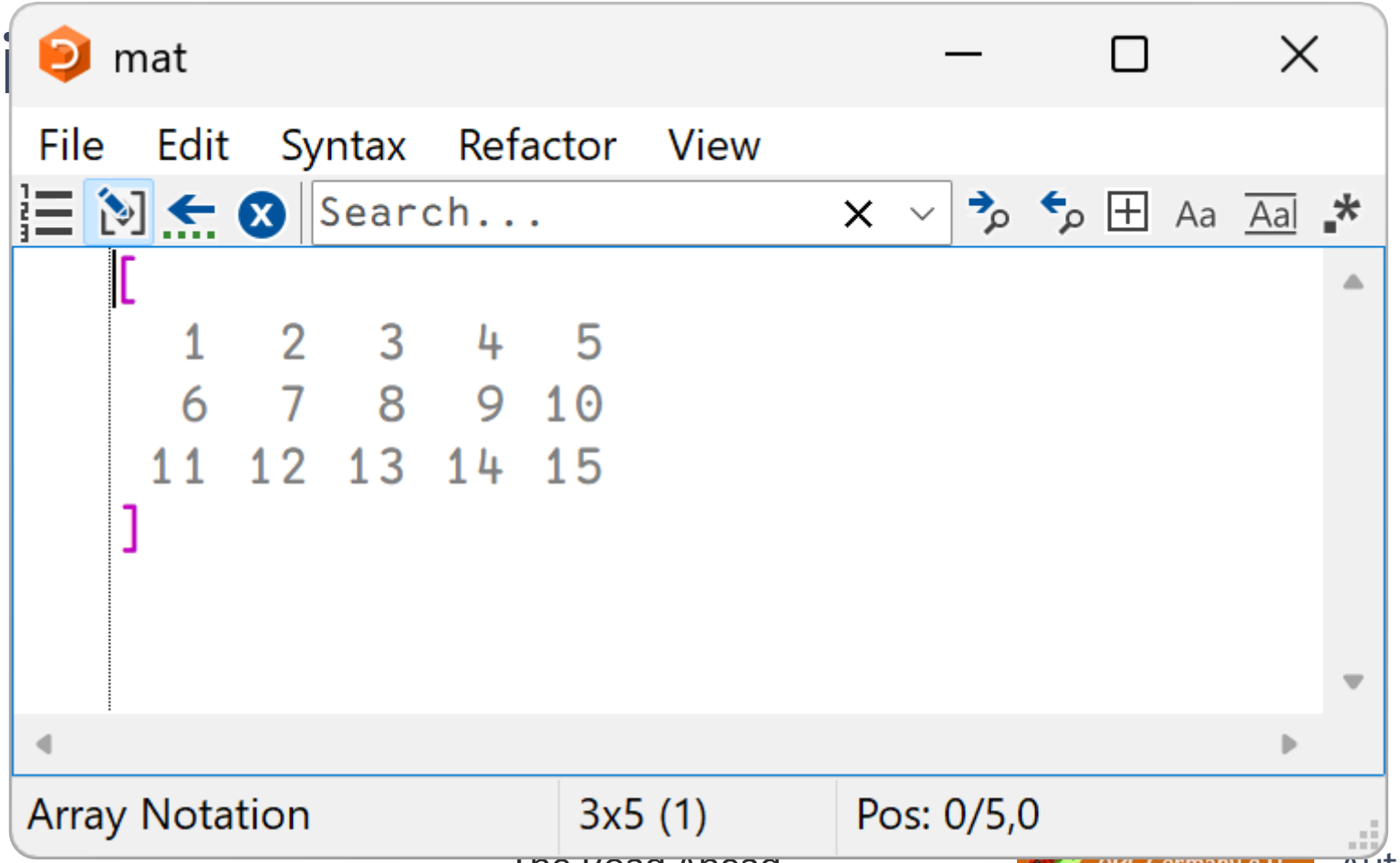
⎕VGET extracts names from a namespace, with optional defaults

Namespace notation combined with ⎕VGET makes named arguments much more attractive

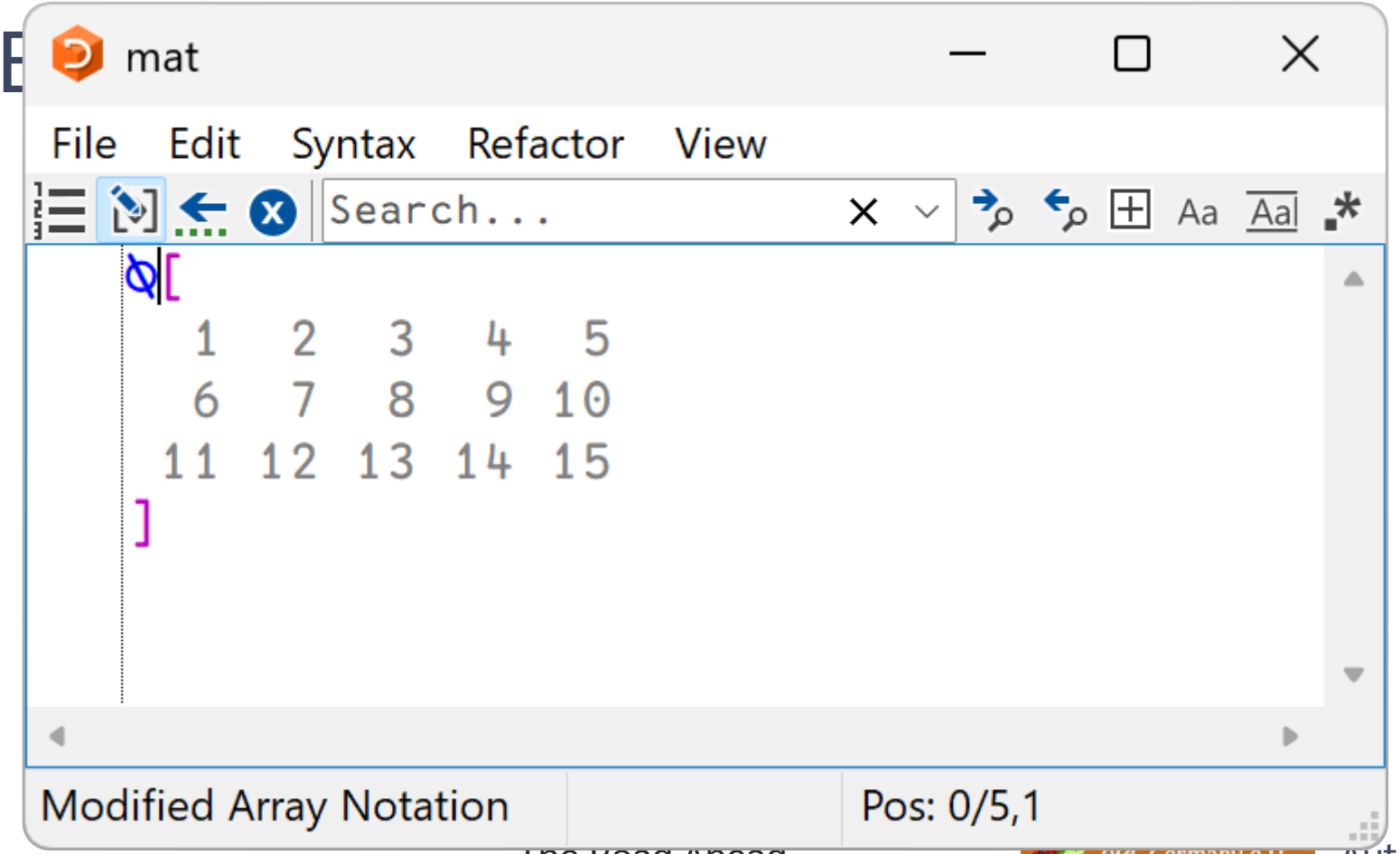
# Editing Array Notation

```
      mat←3 5⍴⍳15
      '⋄'⎕ED'mat'
```

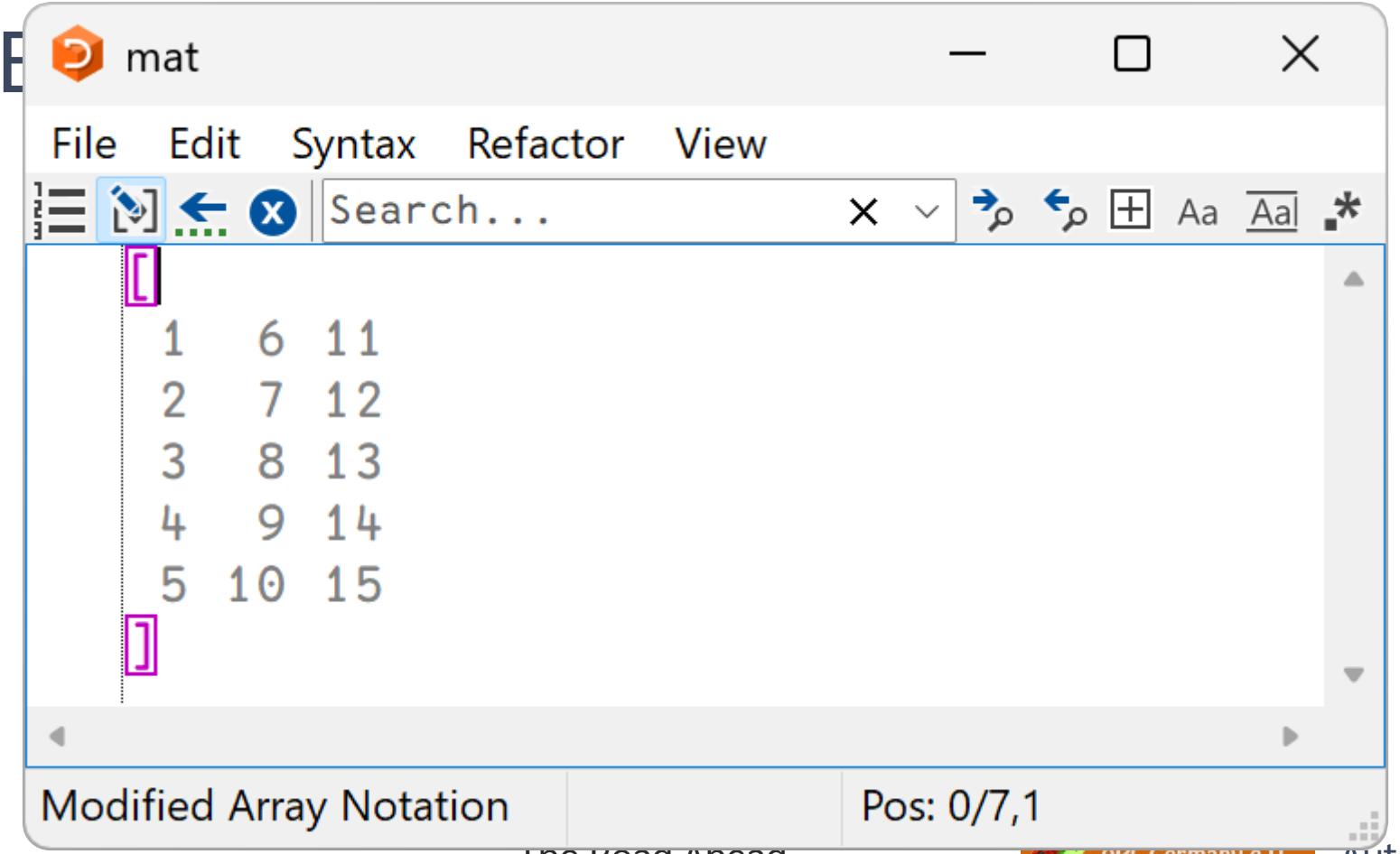
# Edi


mat
File Edit Syntax Refactor View
Search...
Aa
Aa|
[
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
]
Array Notation
3x5 (1)
Pos: 0/5,0

# IDE


mat
File
Edit
Syntax
Refactor
View
Search...
Aa
Aa|
⍳[
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
]
Modified Array Notation
Pos: 0/5,1

# IDE


mat
File
Edit
Syntax
Refactor
View
Search...
Aa
Aa|
[
1 6 11
2 7 12
3 8 13
4 9 14
5 10 15
]
Modified Array Notation
Pos: 0/7,1

# Developer Productivity

- Calling APL as a service 👍
- Rolling out Array Notation
- Modules



# Language support for Modules

- There are many, many ways to load code into the active workspace
  - `⎕FX∘⎕FREAD`, `⎕FIX`, `⎕CY`, `⎕LOAD`, `∆pdef∘⎕FREAD`
  - `HttpCommand.Fix`, `]get`, `]Link.Create`, `⎕SE.SALT.Load`
- Most big APL teams defined their own framework
  - Sharing between APL teams was impractical
  - Original "flat" APL workspaces made things even harder
- We intend to define a new mechanism that future module systems can build upon
  - (Oh no, yet another way … ☺)

Tatin
https://tatin.dev
Chat
Import favorites | For quick access, place your favorites here on the favorites bar. Manage favorites now

# Tatin Registry



The Tatin Registry holds packages for Dyalog APL — code together with everything needed to load it: which versions of Dyalog it runs under, what it depends on and under what licence it may be used.

Tatin installs a package into a folder of your choosing and fetches whatever it depends on at the same time, so using a package is just a single command.

**74** packages

**612** releases

**12** groups

## Recently published

All packages →

2026-10-08 **aplteam-APLGit2** `1.3.1`
Git interface from Dyalog APL via Git Bash

2026-10-03 **dyalog-WebSocketClient** `1.0.1`
Utility to execute WebSocket requests

2026-10-02 **aplteam-HandleError** `5.0.3`
Allows to catch errors on an application level; saves information useful to analyze the error

2026-09-30 **aplteam-Cider** `0.54.1`
A project manager for Dyalog APL that cooperates with Tatin

2026-09-29 **dyalog-HttpCommand** `5.11.2`
Utility to execute HTTP requests

2026-09-29 **dyalog_labs-Stark** `0.2.4`

## Using this Registry

1 Check with `]TATIN.ListRegistries` whether you have this Registry already; if not add it to your user settings:

```
https://tatin.dev
```

2 Install packages with

```
]TATIN.InstallPackages [alias]PkgName /path/
```

where *alias* is the name this Registry goes by on your machine. Every package page carries that command ready to be copied.

This server operates a "delete none" policy: once published, packages cannot be deleted.

# Tatin Registry



## Packages 3 of 73

process

Include tags in the filter

L Linux M macOS W Windows ✓ user command

Sort by: Group | Name | Major versions

### aplteam
45 packages

**APLProcess** — Start an APL process from within Dyalog APL
process
L M W — 1 major

**Execute** — Start a process from within APL
process run-applications
L M W — 2 majors

### dyalog
6 packages

**APLProcess** — Utility to start APL processes
pi
L M W — 1 major



**Kai Jaeger**

APL Toolsmith Extraordinaire

Author of dozens of free and open source libraries

Including the Tatin Package Manager

# Groups 12

A group is the part of a package ID in front of the first hyphen: `aplteam-Cider` belongs to `aplteam`. A group name leads to the group's home page if it has one, otherwise to its packages.

## abrudz
2 packages

ColSchemes sort

## aplteam
45 packages

ADOC APLGit2 APLGUI APLProcess APLTreeUtils2 Cider CodeBrowser CodeCoverage CommTools Compare CompareFiles CompareGUI CompareSimple DateAndTime DotNetZip Dropbox EventCodes Execute FilesAndDirs GitHub GitHubAPIv3 HandleError HashPasswords IniFiles Inno Laguntza Latest ListObjects Logger MakeHelpers MarkAPL OS PresentAPL ServiceState SevenZip ShowChmHelp SMTP Snippets Tester2 WindowsEventLog WinReg WinRegSimple WinSCP_NET WinSys ZipArchive

## bkaw
4 packages

APLPage DotEnvAPL JWTAPL Stark

## bpbecker
1 package

XMLUtils

## BrianED
1 package

raylibAPL

## davin
4 packages

DateTime FilePlus SQLFns Tester

## dyalog
6 packages

APLProcess HttpCommand Jarvis NuGet OpenAI WebSocketClient

## dyalog_labs
1 package

Stark

## rikedyp
2 packages

HTMLTables TinyTest

## sjt
3 packages

text tinytest translate

## tiamatica
2 packages

Mutsu Notela

## yaml
2 packages

yamlscript yamlstar

2 packages

## aplteam

45 packages

ADOC APLGit2 APLGUI APLProcess
APLTreeUtils2 Cider CodeBrowser
CodeCoverage CommTools Compare
CompareFiles CompareGUI CompareSimple
DateAndTime DotNetZip Dropbox EventCodes
Execute FilesAndDirs GitHub GitHubAPIv3
HandleError HashPasswords IniFiles Inno
Laguntza Latest ListObjects Logger
MakeHelpers MarkAPL OS PresentAPL
ServiceState SevenZip ShowChmHelp SMTP
Snippets Tester2 WindowsEventLog WinReg
WinRegSimple WinSCP_NET WinSys
ZipArchive



1 package

## BrianED

1 package

raylibAPL

6 packages

s NuGet

## dyalog_labs

1 package

Stark

## rikedyp

2 packages

HTMLTables TinyTest

# Language support for "Modules"

| Before | With modules |
|---|---|
| `(dfns←()).⎕CY 'dfns'` | `dfns←⎕IMPORT 'dfns'` |
| `'cmpx' ⎕CY 'dfns'` | `'cmpx' ⎕IMPORT 'dfns'` |
| `2 ⎕FIX '/src/Utils.apln'` | `Utils←⎕IMPORT 'Utils'` |
| `⎕SE.Link.Import '#.MyApp' '/src/MyApp'` | `MyApp←⎕IMPORT 'MyApp'` |

Goal:
- Make it easy to share code.
- Simplify the implementation of Tatin and similar tools.

Planning to complete and publish a design and a model in early 2027

# Developer Productivity

- Calling APL as a service 
- Rolling out Array Notation
- Modules
- Scripts
  - Still not "first class" despite several rounds of improvements



# Command Line Interface



## Idea: run scripts directly via #!

```
$ cat hello
#!/usr/bin/env dyalog
⎕←'Hi ',⍪⎕SYSTEM.CommandLine.ScriptArgs
$ ./hello Ada
Hi Ada
```

## Idea: one-liners with -e

```
$ dyalog -e '+/⍳100'
5050
$ dyalog -e 'x←⍳5' -e 'x*2'
1 4 9 16 25
```

## Idea: programs on stdin

```
$ echo "⎕←×/⍳10" | dyalog -
3628800
$ dyalog -e '⍳3' | wc -w
3
```

## Idea: an evaluation server

```
$ dyalog --serve localhost:11612 app.dws &
$ echo 'x←⍳5' | nc localhost 11612
$ echo '+/x' | nc localhost 11612
15
```

# Developer Productivity

- Calling APL as a service 
- Rolling out Array Notation
- Modules
- Scripts 

- Dyalog APL Plugins
- VS Code Plugin for APL 



# Plugin Architecture

- Extensions written in C++ exposing classes usable via ⎕WC / ⎕NEW
- Allows us to extract interface code from interpreter and open-source it
- Allow users to
  - Contribute to "Dyalog Plugins"
  - Write their own extensions & interfaces



# Dyalog Plugins

- Will be Open Source
- HTMLRenderer (CEF wrapper)
  - Experimental with v21.0
  - Default in v22.0
- "Conga Replacement"
  - TCP/UDP Communications Library
- Cryptographic Library
  - (cross platform)



# More on Open Source

- Interpreter remains Closed
- Plugins allow open-sourcing interface code
- Tools writen **in** APL are open
- dyalog-labs repository



dyalog-labs

Overview | Repositories 15 | Projects 1 | Packages | Teams | People 13

## Dyalog Labs

10 followers | United Kingdom

AT DYALOG '26

**D06: Inside Dyalog Labs – From Prototype to Project**

**Examples and demos!**
Martina Crippa

- New initiative for sharing experimental and supporting work from Dyalog Ltd
- GitHub organization – already 12 public projects



# Developer Productivity

- Calling APL as a service 
- Rolling out Array Notation
- Modules
- Scripts 

- Dyalog APL Plugins
- VS Code Plugin for APL 



# 3. APL Language Work

**Already Discussed**

- Array Notation
- Language Support for Modules

**Designed, awaiting approval**

- Key with Vocabulary
- Adám's seven new monadic forms



**Design Ongoing**

- "Finally" constructs / More control over Destructors
- Grand Unified Theory of Async via Futures

# Key with Vocabulary

```
      dna←'AAC TTC ACA TCT AAT'
      {⍺ ⍵}⌸dna
A   1 2 9 11 17 18
C   3 7 10 14
    4 8 12 16
T   5 6 13 15 19

      'ACGT'⌸dna
```




```
      ≠⌸'ACGT'⌸dna
6 4 0 5
```

Result in order of appearance

Proposed extension:
Array operand specifies order
Result is indices

# APL Language Work

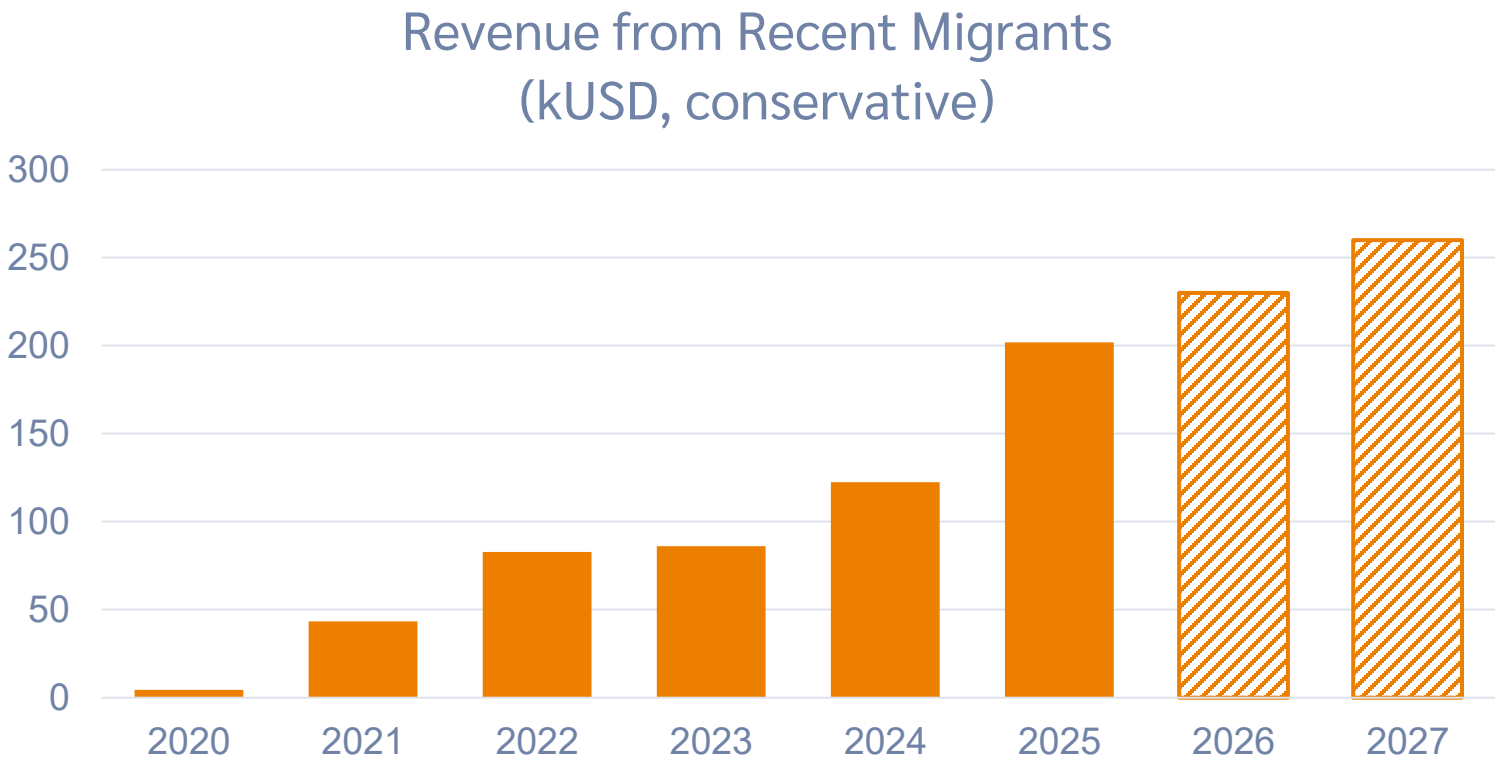
## Monadic Extensions proposed by Adám

⍋⍒ sort
⍤⍣ promote and demote rank
⊤⊥ encode and decode binary
⊇ pick last

AT DYALOG '26
**D18: One Extension and Seven New Primitives**
Adám Brudzewsky

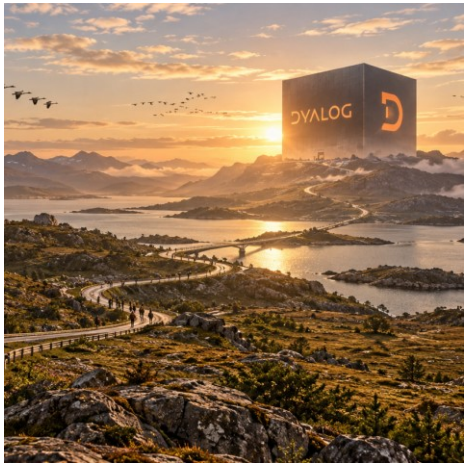
# 4. Migrations




Revenue from Recent Migrants
(kUSD, conservative)
300
250
200
150
100
50
0
2020
2021
2022
2023
2024
2025
2026
2027

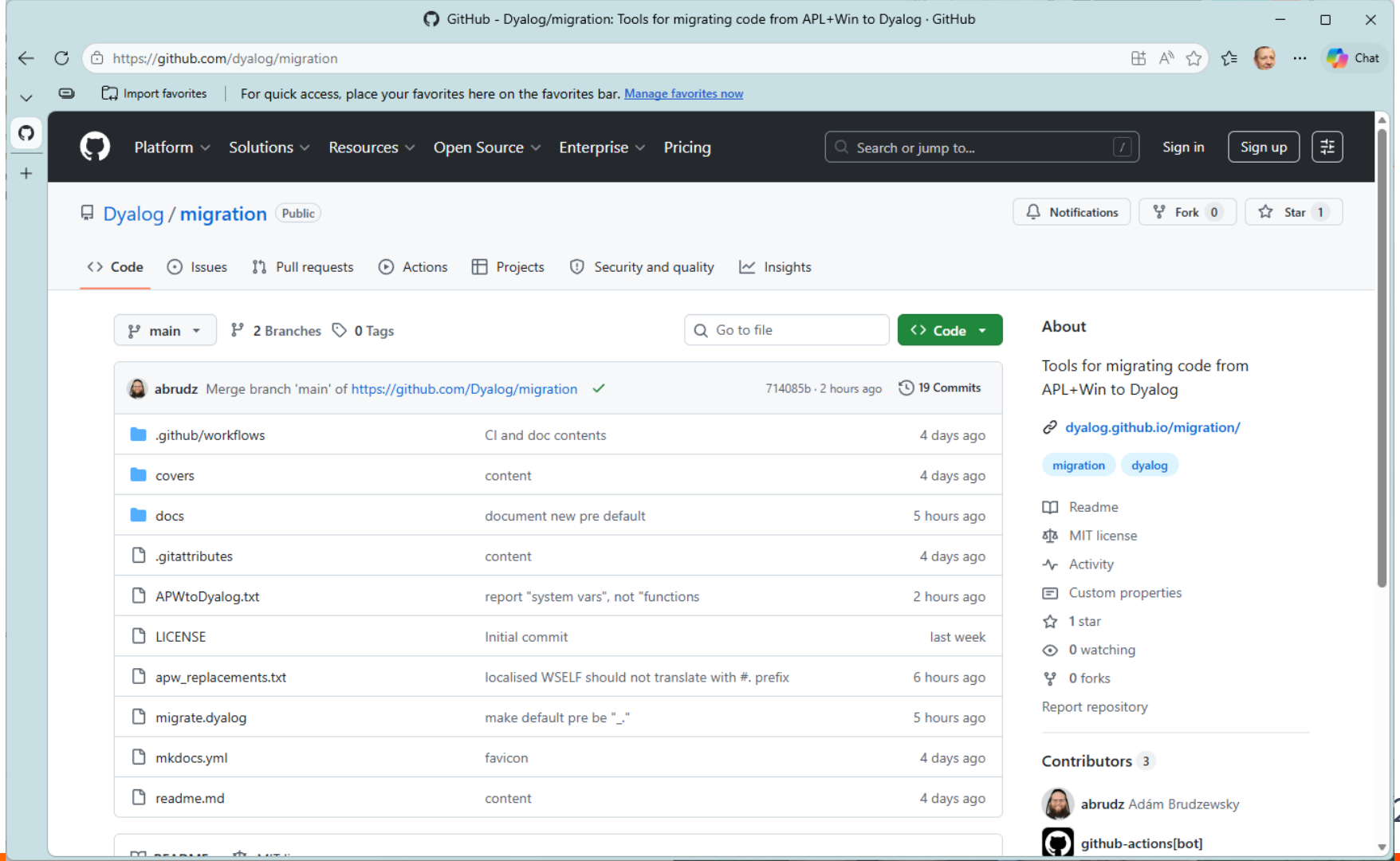
# Building Migration Tools

- Tools to import & convert source code and data from APL+Win and APL2
- Emulation of non-Dyalog language features – in particular GUI



# Graphical User Interfaces

- We have an emulator for GDDM / AP126
  - Generates HTML/svg
- And a "`⎕WI`" emulator for APL+Win GUI
  1. First version in production in the METSIM® system
  2. Version 2.0 rewrite in progress by Davin Church.


GitHub - Dyalog/migration: Tools for migrating code from APL+Win to Dyalog · GitHub
https://github.com/dyalog/migration
Import favorites
For quick access, place your favorites here on the favorites bar. Manage favorites now
Chat
Platform
Solutions
Resources
Open Source
Enterprise
Pricing
Search or jump to...
Sign in
Sign up
Dyalog / migration
Public
Notifications
Fork 0
Star 1
Code
Issues
Pull requests
Actions
Projects
Security and quality
Insights
main
2 Branches
0 Tags
Go to file
Code
abrudz Merge branch 'main' of https://github.com/Dyalog/migration
714085b · 2 hours ago
19 Commits
.github/workflows CI and doc contents 4 days ago
covers content 4 days ago
docs document new pre default 5 hours ago
.gitattributes content 4 days ago
APWtoDyalog.txt report "system vars", not "functions 2 hours ago
LICENSE Initial commit last week
apw_replacements.txt localised WSELF should not translate with #. prefix 6 hours ago
migrate.dyalog make default pre be "_." 5 hours ago
mkdocs.yml favicon 4 days ago
readme.md content 4 days ago
About
Tools for migrating code from APL+Win to Dyalog
dyalog.github.io/migration/
migration
dyalog
Readme
MIT license
Activity
Custom properties
1 star
0 watching
0 forks
Report repository
Contributors 3
abrudz Adám Brudzewsky
github-actions[bot]

# DeltaWI Progress

23 classes supported:

| | |
|---|---|
| System (#) | Option |
| Button | Page |
| Check | Picture |
| Combo | Printer |
| Edit | Progress |
| Form | RichEdit |
| Frame | Selector |
| Label | Scroll |
| List | Spinner |
| ListView | Timer |
| MDIForm | Tree |
| Menu | |

So far:

358 Methods
708 Properties
296 Events
26 Draw Ops

1388 Total

Next Up

Trackbar
Status
DateTime
ImageList
CommandBar
CommandButton
Media
Zip

*(Some uncommonly used features are not yet implemented)*

# DeltaWI Implementation

- Source code is currently on GitHub at:
  https://github.com/DavinChurch/DeltaWI
- Free and open source, but supported

- For more information
  - Alexander Krämer (HBA) will present DeltaWI at tomorrow
  - Davin Church will be at Spring (May 10) and Fall (September 16-17) DYNA in New York

# Does AI make Obsolete...



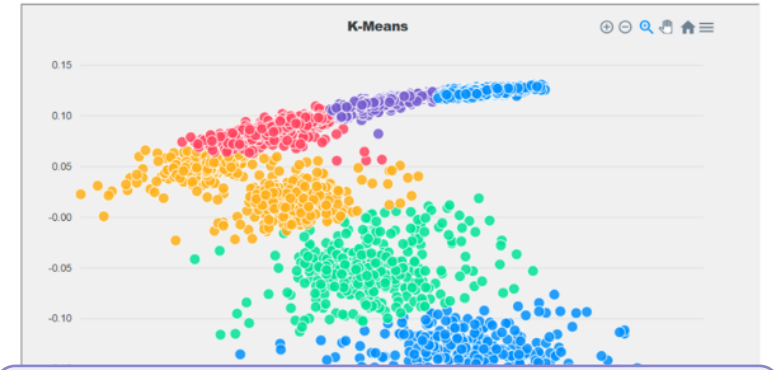
# 5. "Data Science – Driven" Work

- We ran an Internal experiment to duplicate a known Pandas+Numpy solution using Dyalog APL

# Data Science Findings (2025)

We need to work on:

- Faster Data Loading **v22.0**
- Faster Matrix Multiplication **v22.0**
- Easy access to LLMs ✓ **see D08**
- New Visualisation Tools ✓ **see D20**
- Arrow & Parquet Data Formats
- Python Integration
  - Use Python packages from APL
  - Use APL as a package from Python




AT DYALOG '26

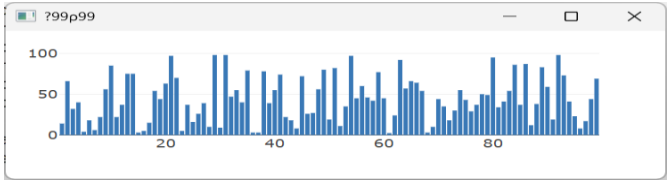
**D03: Slowly Getting Faster**
Asher Harvey-Smith

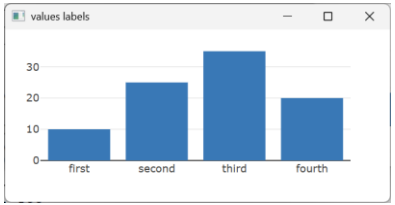
**D08: Web Services Overview**
Brian Becker

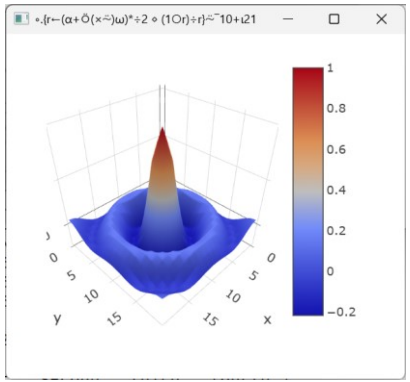
**D20: Plots and Tables: A Modern Approach**
Jesús Galán López

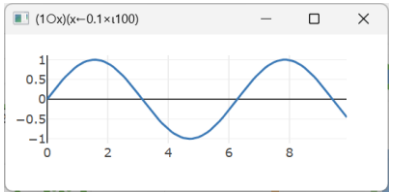
plotly.js

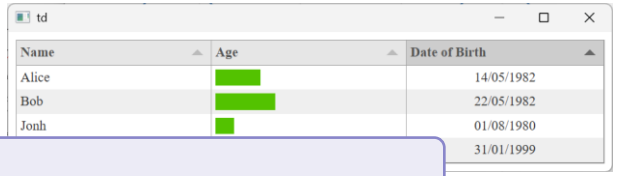
tabulator

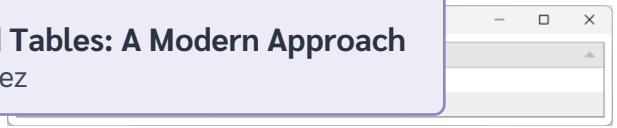


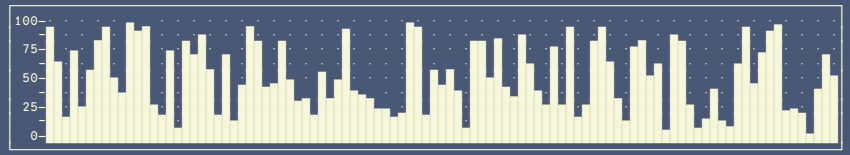



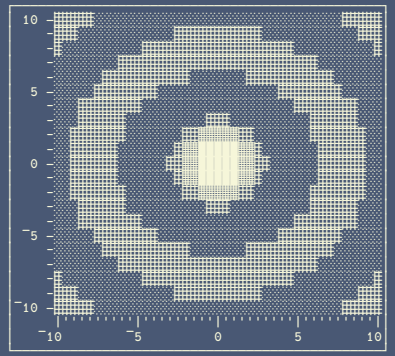
**AT DYALOG '26**

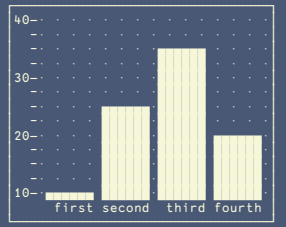
**D20: Plots and Tables: A Modern Approach**

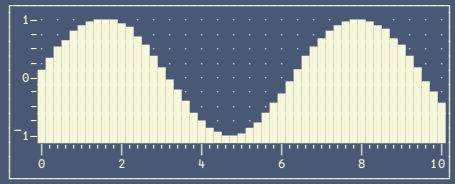
Jesús Galán López

textdraw




| Name | Age | Date of Birth |
|---|---|---|
| Alice | | 14/05/1982 |
| Bob | | 22/05/1982 |
| Jonh | | 01/08/1980 |
| Sarah | | 31/01/1999 |

| A | B | C |
|---|---|---|
| 1 | 3 | 5 |
| 2 | 4 | 6 |

DYALOG

Eastbourne 2026

# Version 21.0 Highlights

- Software Bill of Materials
- Dark APL
- ⎕DT – format and parse dates
- Language support for .NET Generics
- ⎕SYSTEM – "Platform Features



https://docs.dyalog.com/21.0/release-notes/new-enhanced/

# Building the Team

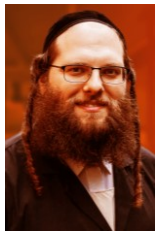
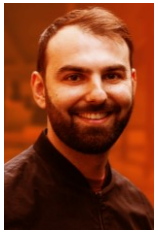





The Road Ahead

# Dyalog – The Next Generation

2010-2021



2022











2023

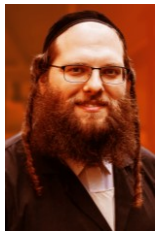
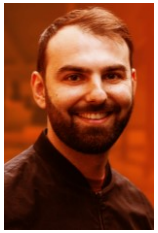




2024

# Dyalog – The Next Generation

2010-2021



2022





2023

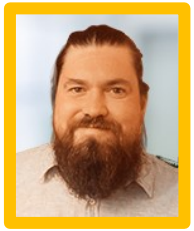






2024





# 2025 & 2026

Developers:



Asher (Interpreter)



Holden (Intern)

?

? (Testing Engineer)

Business support:



Martin (IT)



Jasmin (Accounting)

# And in 2027

Tiamatica AB (Malmö) joins the "Dyalog Group"



Gilgamesh



Sandra



Karl

