



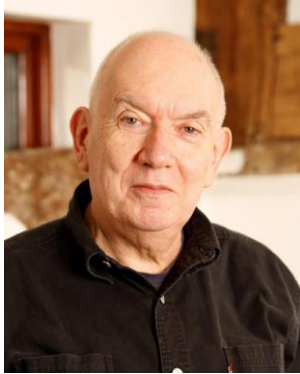
Mainz, April 30th 2024



News from Dyalog

Morten Kromberg, CTO

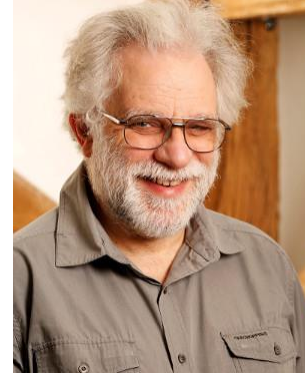
We Stand on the Shoulders of Giants



John Scholes (1948-2019)



Roger Hui (1953-2021)



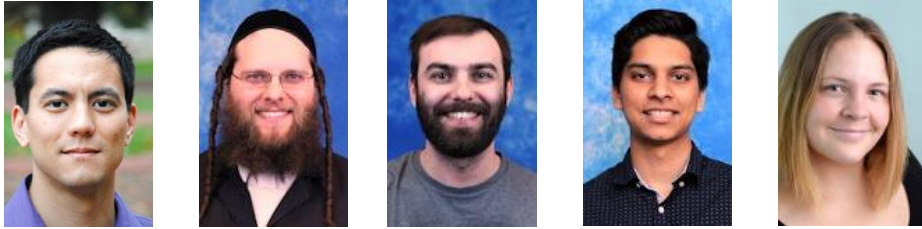
Geoff Streeter (retired 2023)

<https://rogerhui.rip>

<https://johnscholes.rip>

Dyalog – The Next Generation

2010-2021



2022



2023



2024



2023 Summer Interns



Our 2nd Employee to Retire



Gitte Christensen

The CEO has retired!

Under Gitte's Reign ...

- Headcount and Turnover ~~Quadrupled~~ Quintupled
 - Headcount now ~26 full time equivalents
 - Steadily increasing R&D budget
 - First generation of interpreter developers retired;
New Team installed
 - Largest sustained investment in APL technology in history

Under Gitte's Reign ...

- Trading Profit in every year since 2005. Sources:
 - Growing partnerships with major customers
 - Increasing share of APL market
 - A small number of completely new APL systems
- Constantly working towards positioning APL as a modern tool for prototyping and rapid application development

Financial Status

Owners	2008	2018	Present
SimCorp (Denmark)	32.33%	40%	24%
APL Italiana (Italy)	32.33%		
Management & Employees	35.33%	60%	76%

- Ownership
 - 24% owned by SimCorp (which acquired APL Italiana, and was then itself acquired by Deutsche Börse)
 - 76% owned by management and employees
- Revenue steadily increasing:

5-year period	Growth
2017-2023	35%
2012-2017	8%
2007-2012	57%

Our 2nd Employee to Retire



Gitte Christensen

The CEO has retired!



Kirstine Kromberg

Long Live the New CEO!

Spreading the Gospel



Aaron Hsu
co-dfns
compiler



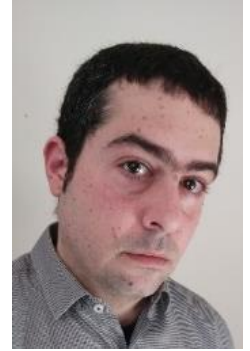
Adam Brudzewsky
Here, There and
Everywhere



Rich Park
Media, Training
and "Outreach"



Stefan Kruger
Technical
Writer



Jesús López
Metallurgist
(embedded
evangelist)

x (+/ x∘-1)∘1 2 ←y

Learning APL

- Introduction**
- It's arrays all the way down
- Indexing
- Glyphiary
- Direct functions and operators
- Iteration
- The Key operator: **⌈**
- The At operator: **@**
- The Rank/Atop operator: **⌊**
- The Stencil operator: **⌊**
- The Over operator: **⌊**



Introduction

A language that doesn't affect the way you think about programming is not worth knowing. —Alan Perlis



Stefan Kruger

Who is this for?

I wrote this to be the book I would have wanted to read when I started to learn APL. **An introduction to APL for an experienced practitioner** from a different programming language or two. We all learn in different ways, and I prefer the fundamental concepts laid bare first, and then learn by example.

I came to APL after discovering a file of [solutions](#) to the [Advent of Code](#) 2015 challenge in [K](#), an APL derivative. That's around 100 lines of actual code, and whilst I didn't understand any of it, I kept looking at it, trying to

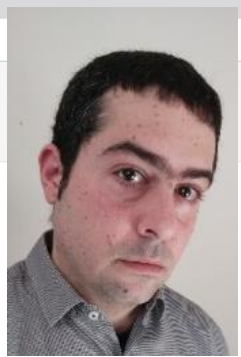
Contents

- Who is this for?
- What is APL?
- Why should I learn APL?
- ...but it's unreadable!
- Don't I need a lot of mathematics?
- A note on our APL subset
- Is terser better?
- Other resources
- Ok, I'm convinced, how do I get started?
- Our first tentative steps
- Valence



Jesus Galan

TU Delft
 Verified email at tudelft.nl - [Homepage](#)
[Materials Science](#) [Mechanical Engineering](#) [Computational Materials Sc...](#)



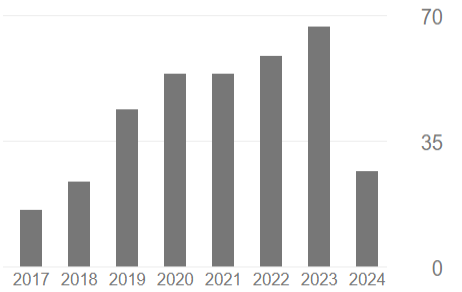
Jesús Galan López

[GET MY OWN PROFILE](#)

TITLE	CITED BY	YEAR
Advanced high strength steels for automotive industry J Galán, L Samek, P Verleysen, K Verbeken, Y Houbaert Revista de metalurgia 48 (2), 118	195	2012
Analysis of ESAFORM 2021 cup drawing benchmark of an Al alloy, critical factors for accuracy and efficiency of FE simulations AM Habraken, TA Aksent, JL Alves, RL Amaral, E Betaieb, N Chandola, ... International Journal of Material Forming 15 (5), 61	32	2022
Thermal effects during tensile deformation of Ti-6Al-4V at different strain rates J Galán, P Verleysen, J Degrieck Strain 49 (4), 354-365	26	2013
Effect of fatigue damage on static and dynamic tensile behaviour of electro-discharge machined Ti-6Al-4V JG Lopez, P Verleysen, J Degrieck Fatigue & Fracture of Engineering Materials & Structures 35 (12), 1120-1132	23	2012

Cited by [VIEW ALL](#)

	All	Since 2019
Citations	421	306
h-index	10	9
i10-index	10	8



Public access [VIEW ALL](#)

☰ README.md

quAPL [↗](#)

Santiago Núñez-Corrales, PhD (nunezco2@illinois.edu)

Marcos Frenkel (marcosf2@illinois.edu)

Bruno de Abreu (babreu@illinois.edu)

National Center for Supercomputing Applications

Description [↗](#)

quAPL is an APL implementation of a collection of primitives that can be used to simulate a quantum computer. The code is intended to be experimental, and will continue to be expanded upwards to contain composable primitives leading to the vision of an Instruction Set Architecture, as well as downwards to replace core entities with calls to existing quantum implementations, ideally using OpenQASM.

Why APL [↗](#)



- APL naturally captures the effect of operators over vectors in normed, finite-dimensional Hilbert spaces representing quantum states
- APL's array-based syntax allows quantum programmers to focus on the semantics of qubit operators while removing concerns about implementation details unrelated to quantum computing
- APL's ability to optimize operations across multiple data types
- APL's native handling of complex numbers

No releases published

Packages

No packages published

Contributors 2

-  **nunezco2** Santiago Nunez-Corrales
-  **marcosfrenkel**



Vertical sidebar with various icons: search, home, mail, calendar, and settings.



LambdaConf speakers

[Apply to be a speaker! ↗](#)



Aaron Hsu



Adam Fraser



Adam McCullough



Afsal Thaj



Ziverge





Joseph Guadagno



Josh David



Jubin Thomas



Julia Belyakova



Liam Fitzgerald



Luke Champine



Madhu Patel



Martin Förtsch



Matthew Fuchs



Max Demoulin



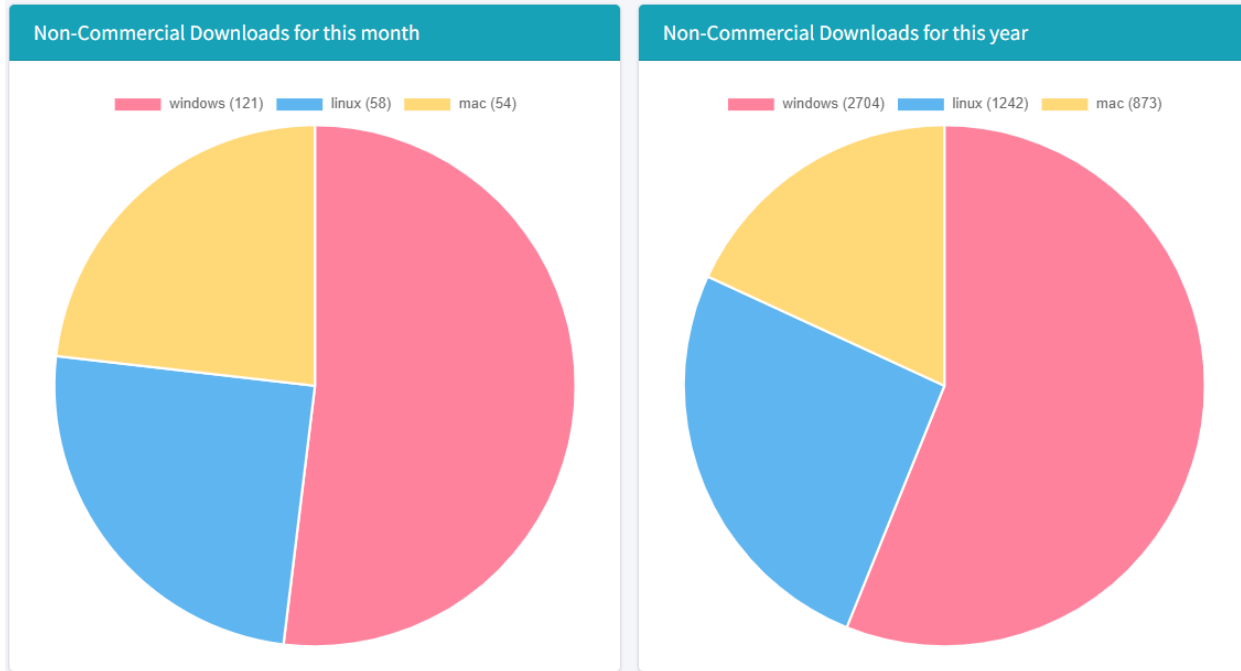
Max Gfeller



Max Sun

Non-Commercial Downloads (snapshot September 18th, 2023)

= ~7,000 / year



Sketch of Version 20.0 (Q2/2025)

Ongoing from v19.0

- Resume Optimisation Work
- .NET Bridge enhancements
 - Support "Generic" methods & classes
 - Important for many NuGet packages
- More HTMLRenderer improvements
- Health Monitor
- Script Engine Support

Next Set of Projects

- Relax Interpreter Limits
- Open-Source HTMLRenderer & Conga
- Set and Get values w/out Execute
- Array Notation
- HTMLRenderer Enhancements
- Token-by-token Debugging
- New "Shell" System Command
- JavaScript emulation of `WC`

See Separate Presentations

- ◆ Set and Get values without Execute
- ◆ Array Notation

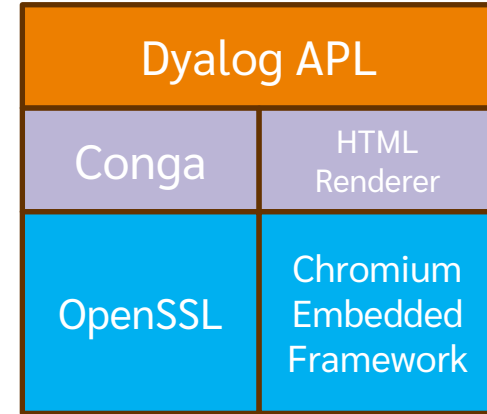
Relax Limits in the Interpreter

We are going to spend the time relaxing limitations in the interpreter, like

- Max Rank (15)
- # of lines in a function (9,999)
- # tokens in a line (32,676)
- Number of token types (needed to add more language structure)
 - The primitives in Adam Brudzewsky's presentation "Filling the Core Language Gaps" at Dyalog'22 will have to wait a year

An Open [Source] Plugin Architecture

- A modern replacement for Auxiliary Processors
 - Uses Direct Workspace Access (DWA) for high performance
 - Accessible via `WC` / `NEW`
- Make parts of the interpreter open source, initially
 - HTMLRenderer
 - Conga (our TCP platform)
 - Cryptographic Library
- These make heavy use of open source components
 - Open sourcing makes it simpler to comply with open source licences



Plugin Architecture Benefits

- ◆ Allow users to move faster and prototype enhancements rather than wait for Dyalog
- ◆ Make the Dyalog community more inclusive by allowing users to contribute
- ◆ Users can develop completely new extensions and easily share them with others
- ◆ Simplify use of open source libraries

Health Monitor

Version 19.0 contains a prototype. Ideas for v20.0 include:

- ◆ Add feature to find last known location of a "hanging" interpreter
- ◆ Sending signals to interrupt or terminate tasks
- ◆ Discoverability: allow APL process to broadcast services that it provides
- ◆ Switch `□PROFILE` on and off; collect data
- ◆ Possibly add OpenTelemetry data feed



Token-by-token Debugging

Debugger

Tools View

<no value>

Search

```
tbt;r;group;shift;part;count
(+≠)10
(+≠)10 10p100
(+≠)10 10p100
(+≠)10 10p100
{(+/\ ' '=ω)ω}' Dyalog A ▶

{ω=1:1+ω ◊ 1-ω}1
{ω=1:1+ω ◊ 1-ω}2

r←8 4 12 3≡6(+,-,×,÷)2 ◊ (t; ▶

group←t>“,oεt/“ A Group pai ▶
group ◊←(5 3)(5 6)(7 5)(4 7) ▶

shift←t,0t~
3 shift10

part←(t~1,2≠/+)
part'aaabbccc'
```

Function Pos: 4/24,0

Right Argument

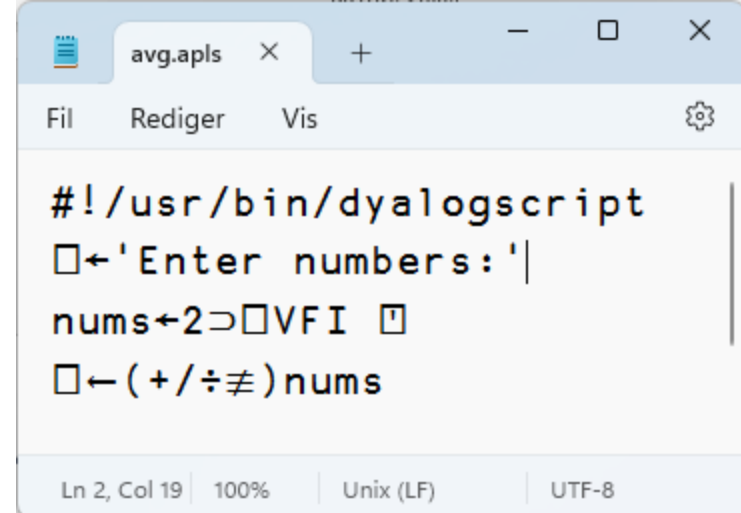
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Current Function

Sistack (Tid: Tid:0)

Script-Engine Support

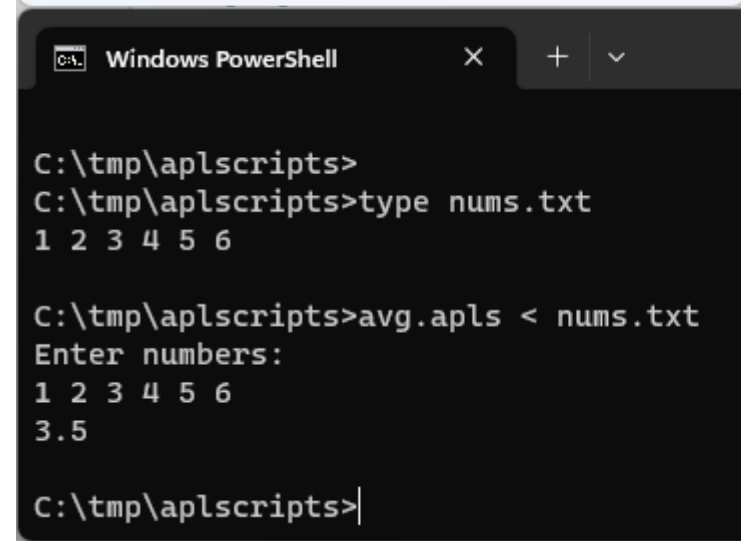
- #! (hash bang) scripting
- We think the script engine is critical for attracting new users
- Still a bit of a prototype
 - Will be hardened for v20.0
 - Need to be able to debug scripts via RIDE



```
avg.apls x + - □ ×
Fil Rediger Vis ⚙️

#!/usr/bin/dyalogscript
□←'Enter numbers: '|
nums←2⊃□VFI □
□←(+/÷≠)nums

Ln 2, Col 19 | 100% | Unix (LF) | UTF-8
```



```
Windows PowerShell x + v
C:\tmp\aplscripts>
C:\tmp\aplscripts>type nums.txt
1 2 3 4 5 6

C:\tmp\aplscripts>avg.apls < nums.txt
Enter numbers:
1 2 3 4 5 6
3.5

C:\tmp\aplscripts>|
```

□SHELL to replace existing □SH

Invoke OS commands from APL

- ◆ Interruptible
- ◆ Optionally return data as an asynchronous Stream
- ◆ Manage stdin, stdout & stderr independently
- ◆ Handle variety of data encodings

Static Analysis of APL Code

- ◆ Static Analysis of application code is seen as a required "best practice" by some corporations
- ◆ We are building a prototype of a tool which will
 - ◆ Detect vulnerabilities
 - ◆ "Lint" APL Code
 - ◆ Compute readability and other metrics
- ◆ This tool will initially be licensed separately
- ◆ A free "community edition" may follow

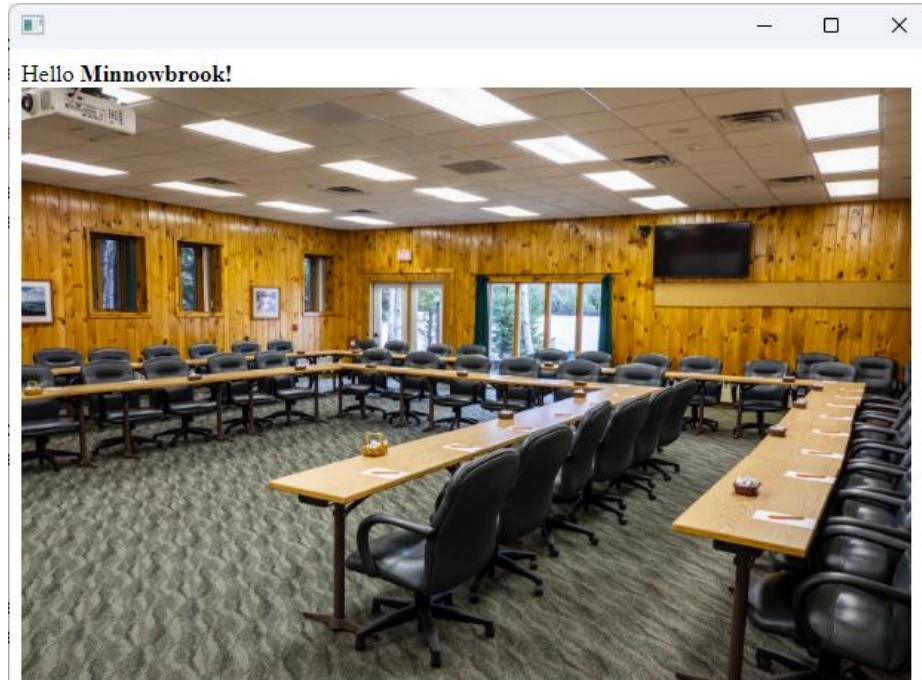
HTMLRenderer Enhancements

Suggested:

- File Upload
- Modal HTMLRenderer Windows
- More control over "Chrome"
- Other changes driven by JSWC project

HTMLRenderer – what's that?

```
pic←'https://minnowbrook.org/wp-content/uploads/2022/04/  
Screen-Shot-2022-04-18-at-2.24.30-PM.png'  
'MyForm' WC 'HTMLRenderer' ('Hello <b>Minnowbrook!</b><br/>  
')
```



Chromium
Embedded
Framework
(CEF)

HTMLRenderer Enhancements

Suggested:

- File Upload
- Modal HTMLRenderer Windows
- More control over "Chrome"
- Other changes driven by EWC project

Easy Web Creator - EWC

- ◆ A JavaScript emulation of our Win32 layer (□WC, □WG, □WS ...)
- ◆ Still mostly referred to as JSWC (JavaScript Window Create)

Easy Web Creator – EWC (JavaScript implementation of WC)

The screenshot shows the EWC desktop application window titled "Function Table". It features a menu bar with "File" and "Colours", a data entry form, a tree view, a calculator, and a grid.

Data Entry Form:

Name	Gender	Score	Expert
Amir	Male	12	<input type="checkbox"/>
Fatima	Female	13	<input checked="" type="checkbox"/>

Average Score: 12.5

Tree View:

- Q1
 - Q2
 - Apr
 - May
 - Jun

Calculator: 10 × [dropdown] [Calc] [display: *****]

Grid:

	A	B	C	D	E	F	G
1	1	2	3	4	5	6	
2	2	4	6	8	10	12	
3	3	6	9	12	15	18	
4	4	8	12	16	20	24	
5	5	10	15	20	25	30	
6	6	12	18	24	30	36	
7	7	14	21	28	35	42	
8	8	16	24	32	40	48	
9	9	18	27	36	45	54	
10	10	20	30	40	50	60	

The screenshot shows the EWC web browser interface at localhost:22322. It includes a browser window with a title bar, address bar, and a toolbar. The main content area displays the same application as the desktop version, but with an "Initialise" button at the top.

Browser Window: JSWC | localhost:22322

Initialise [dropdown]

Data Entry Form:

Name	Gender	Score	Expert
Amir	Male	12	<input type="checkbox"/>
Fatima	Female	13	<input checked="" type="checkbox"/>

Average Score: 12.5

Tree View:

- Q1
 - Q2
 - Apr
 - May
 - Jun

Calculator: 10 × [dropdown] [Calc] [display: *****]

Grid:

	A	B	C	D	E	F	G
1	1	2	3	4	5	6	
2	2	4	6	8	10	12	
3	3	6	9	12	15	18	
4	4	8	12	16	20	24	
5	5	10	15	20	25	30	
6	6	12	18	24	30	36	
7	7	14	21	28	35	42	
8	8	16	24	32	40	48	
9	9	18	27	36	45	54	
10	10	20	30	40	50	60	

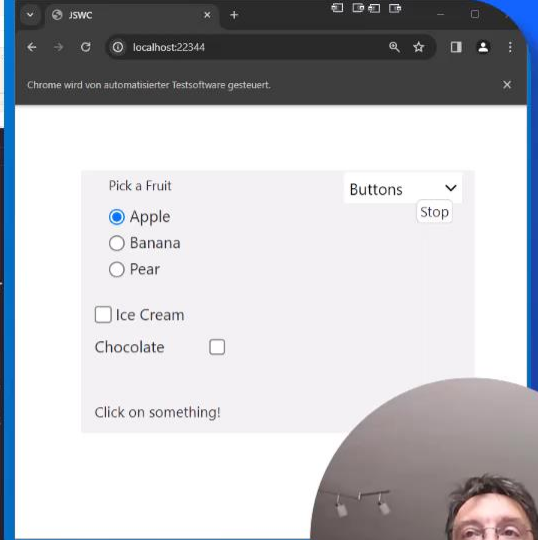
Demo of EWC



Automated Testing w/ Selenium

- One important benefit of HTML rendering is that Selenium can be used for automated testing


```
log 19.0u64
WS Bearbeiten Anzeigen Fenster Session-Objekt Objekteinstellungen-Objekt Werkzeuge Threads Hilfe
Language Bar
Debugger
r-TestButtons sink;TRAP;fruit;addon;t;z;f_ctl;lbl;f_status;fruit_name;c;exp;addon_ids;addons;addon_names;sel_a
r+
:If 0=f_status+#.S.Find'F1.STATUS'
r+Cannot find F1.STATUS (#136)
-end A not much we can sensibly test w/o the ability to control the feedback!
:EndIf
addon_ids+'F1.ICE' 'F1.CHOKO'
A label follows or preceds control:
addon_names+('{{XPath'#.S.Find'//input[@id="'',ω,']'/following-sibling::div | //input[@id="'',ω,']'/preceding-sibling::div'}).T
:For fruit :In ('F1.FRUIT.C'),"t"
{6:;0 ω c+#.S.Find ω c.Selected:c.Click}"addon_ids A make sure all options are unselected
:If fruit#''
z+#.S.Click fruit
DL 1 A bad and ugly hack - we need to give eWS some time to update the status field, otherwise the test will
f_ctl+#.S.Find fruit
lbl+'XPath'#.S.Find'//input[@id="'',fruit,']'/following-sibling::div' A the div that immediately follows the input
fruit_name+lbl.Text A read label's text
t+f_status.Text
exp+'You selected ',fruit_name
:If exp Check t
r,+c'Click on ',fruit,' did not generate expected message but "',t,'"
:EndIf
:EndIf
:For addon :In ~1+i(#addon_ids)*2
{6:;0 ω c+#.S.Find ω c.Selected:c.Click}"addon_ids A make sure all options are unselected
sel_a+2 2addon A bit flags for selected addons
A get names of addons: the div that immediately follows the input has the label
:For c :In sel_a/addon_ids
:If 1 Check #.S.Click c A were we able to click the addon?
r,+c'could not select addon "',c,'"
:EndIf
DL 1 A bad and ugly hack - but we need to give eWS some time to update the status field, otherwise the test will fail!
:EndFor
t+f_status.Text
exp+'You selected ',fruit_name,(0<+/sel_a)'/ with ',~5↓sel_a/addon_names,'"c' and '
:If exp Check t
r,+c'Status ≠ "',exp,'" - found value "',t,'"
:EndIf
:EndFor
:EndFor
end:
:If 0<#r
r+~1+e(εr),"UCS 10
:EndIf
Function
Last saved by: mbass: Montag, 8. April 2024 09:00
Pos: 36/45,53
Editor
```



EWC also Open Source

- Both the server-side APL code and the client-side JavaScript will be GitHub projects
- Users will be able to create their own GUI classes
- We are planning a VegaLite widget
- We are experimenting with responsive design, allowing positioning using e.g. Flex rather than Posn & Size

Sketch of Version 20.0 (Q2/2025)

Ongoing from v19.0

- Resume Optimisation Work
- .NET Bridge enhancements
 - Support "Generic" methods & classes
- More HTMLRenderer improvements
- Health Monitor
- Script Engine Support

Next Set of Projects

- Relax Interpreter Limits
- Open-Source HTMLRenderer & Conga
- Set and Get values w/out Execute
- Array Notation
- HTMLRenderer Enhancements
- Token-by-token Debugging
- New "Shell" System Command

- JavaScript emulation of `□WC`