FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

Array Notation and Language Vision

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

APL Array Notation and Language Vision

Adám Brudzewsky

Head of Language Design, Dyalog Ltd.

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

APL
Array Notation and
Language Visi

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

APLAN: Why?

- Avoiding complex expressions when constructing arrays
  Might not fit comfortably on a single line

- Using array definitions with source code management
  These tend to handle changes on a line-by-line basis

- Arrays in text form
  Edit with any editor, email, transfer, create with 3rd party tools…

APLAN: What?

- Medium sized arrays
  Empty and trivial arrays are better done as expressions

- Higher rank arrays
  We have good notations for vectors and small vectors of vectors

- Depth deeper than 2

APLAN: What?

- Medium sized arrays
  Empty and trivial arrays are better done as expressions

- Higher rank arrays
  We have good notations for vectors and small vectors of vectors

- Depth deeper than 2

APLAN and Link

```
      ]Link.Create # C:\tmp\myproj\src
Linked: # ←→ C:\tmp\myproj\src [directory was created]

      ⎕←var←⍳2 3
┌───┬───┬───┐
│1 1│1 2│1 3│
├───┼───┼───┤
│2 1│2 2│2 3│
└───┴───┴───┘

      ]add var
Added: #.var

      ]view ⊃⎕NGET'C:\tmp\myproj\src\var.apla'
```

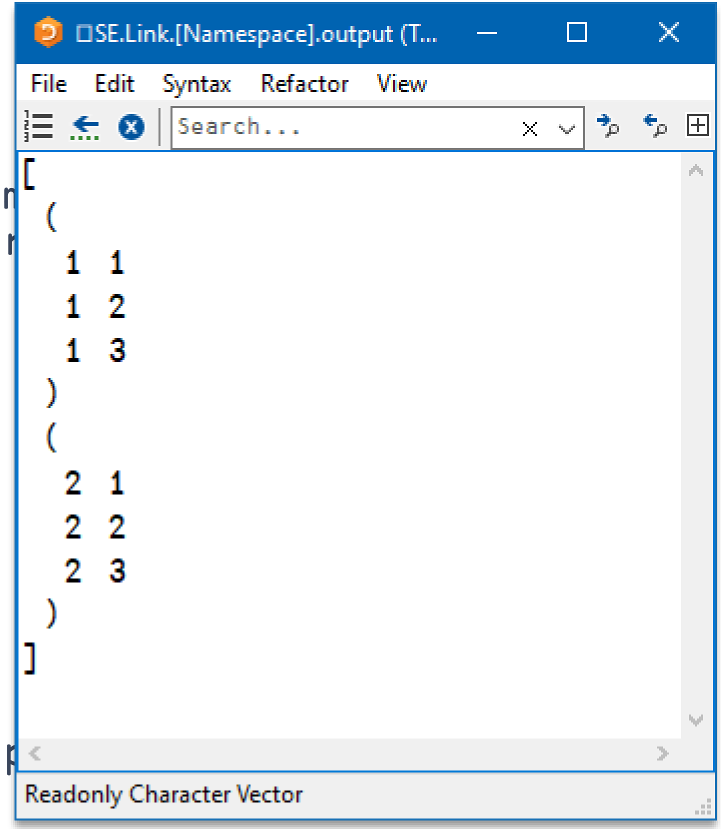
APLAN and Link

```
      ]Link.Create # C:\tmp\m
Linked: # ↔ C:\tmp\myproj\s

      ⎕←var←⍳2 3
┌───┬───┬───┐
│1 1│1 2│1 3│
├───┼───┼───┤
│2 1│2 2│2 3│
└───┴───┴───┘

      ]add var
Added: #.var

      ]view ⊃⎕NGET'C:\tmp\my
```



```
[
 (
  1 1
  1 2
  1 3
 )
 (
  2 1
  2 2
  2 3
 )
]
```

Readonly Character Vector

APLAN arrays of rank 2 and up

```
Multi-line              Inline                     Expression

[1 2
 3 4        ⟺     [1 2 ⋄ 3 4 ⋄ 5 6]     ⟺     3 2⍴1 2 3 4 5 6
 5 6]

[1
 2          ⟺       [1 ⋄ 2 ⋄ 3]         ⟺       3 1⍴1 2 3
 3]
```

APLAN vectors & nested arrays

Multi-line		Inline		Expression

`(1 2`
` 3 4`
` 5 6)`
	⇔	`(1 2 ⋄ 3 4 ⋄ 5 6)`	⇔	`(1 2)(3 4)(5 6)`

`(1`
` 2`
` 3)`
	⇔	`(1 ⋄ 2 ⋄ 3)`	⇔	`1 2 3`

APLAN namespaces

Multi-line

```
(
   a:'APL'
   b:,⍨1 2
)
```

⇔

Inline

```
(a:'APL' ⋄ b:,⍨1 2)
```

⇔

Expression

```
{
   ⍺←⎕NS⍬
   ⍺.a←'APL'
   ⍺.b←{
      ,⍨1 2
   }⍬
   ⍺
}⍬
```

```
(
 )
```

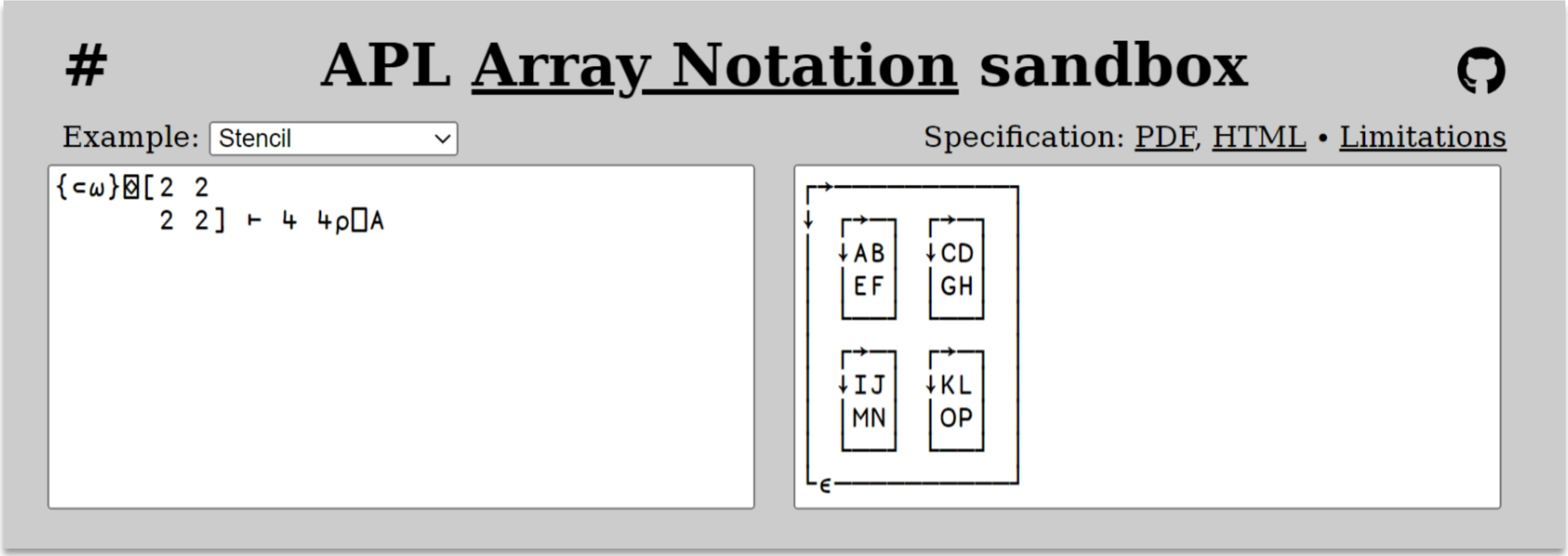
⇔

```
()
```

⇔

```
⎕NS⍬
```

APLAN sandbox: *is.gd/APLAN*



APL Array Notation sandbox

Example: Stencil

Specification: PDF, HTML • Limitations

```
{⊂⍵}⌺[2 2
      2 2] ⊢ 4 4⍴⎕A
```

```
┌→──────────┐
↓ ┌→─┐ ┌→─┐ │
│ ↓AB│ ↓CD│ │
│ │EF│ │GH│ │
│ └──┘ └──┘ │
│ ┌→─┐ ┌→─┐ │
│ ↓IJ│ ↓KL│ │
│ │MN│ │OP│ │
│ └──┘ └──┘ │
└∊──────────┘
```

APLAN in dfns.dws: cal

```
Q1←'January' 'February' 'March    '~¨' '    ⍝ 1st quarter month names.
Q2←'April  ' 'May     ' 'June     '~¨' '    ⍝ 2nd   ..      ..    ..
Q3←'July   ' 'August  ' 'September'~¨' '    ⍝ 3rd   ..      ..    ..
Q4←'October' 'November' 'December '~¨' '    ⍝ 4th   ..      ..    ..
months←Q1,Q2,Q3,Q4                         ⍝ month names for year.
```

APLAN in dfns.dws: cal

```
months←(                                    ⍝ month names for year.
   'January'⋄'February'⋄'March'             ⍝ 1st quarter month names.
   'April'  ⋄'May'     ⋄'June'              ⍝ 2nd   ..      ..    ..
   'July'   ⋄'August'  ⋄'September'         ⍝ 3rd   ..      ..    ..
   'October'⋄'November'⋄'December'          ⍝ 4th   ..      ..    ..
)
```

APLAN in math.dws: Eigen

```
⌽{⍵,⊂' <C1   ' 'V'}{              ⍝ JOBZ
  ⍵,⊂' <C1   ' 'L'}{              ⍝ UPLO
  ⍵,⊂' <I4   'n}{                 ⍝ N
  ⍵,⊂' =F8[] '(∊⍉mat)}{           ⍝ A
  ⍵,⊂' <I4   'n}{                 ⍝ LDA
  ⍵,⊂' >F8[] 'n}{                 ⍝ W
  ⍵,⊂' >F8[] '(¯2+4×n)}{          ⍝ WORK
  ⍵,⊂' <I4   '(¯1+2×n)}{          ⍝ LWORK
  ⍵,⊂' >F8[] '(¯2+3×n)}{          ⍝ RWORK
  ⍵,⊂' >I4   '0}⍬                 ⍝ INFO
```

APLAN in math.dws: Eigen

```
( ' <C1 ' 'V'                   ⍝ JOBZ
  ' <C1 ' 'L'                   ⍝ UPLO
  ' <I4 '  n                    ⍝ N
  ' =F8[] '(∊⍉mat)              ⍝ A
  ' <I4 '  n                    ⍝ LDA
  ' >F8[] 'n                    ⍝ W
  ' >F8[] '(¯2+4×n)             ⍝ WORK
  ' <I4 '  (¯1+2×n)             ⍝ LWORK
  ' >F8[] '(¯2+3×n)             ⍝ RWORK
  ' >I4 '  0        )           ⍝ INFO
```

APLAN in Profile ucmd: DBMenuCB

```
poss←1 2⍴'fns'((0 1)(0.7 0)(0.7 0)×size)
poss⍪←'fnd'((0 1)(0 0)(0 0)×size)
poss⍪←'lines'((0 0)(0.7 0)(0.7 0)×size)
poss⍪←'lnd'((0 0)(0 0)(0 0)×size)
```

APLAN in Profile ucmd: DBMenuCB

```
poss←['fns'  ((0.0 1 ⋄ 0.7 0 ⋄ 0.7 0)×size)
      'fnd'  ((0.0 1 ⋄ 0.0 0 ⋄ 0.0 0)×size)
      'lines'((0.0 0 ⋄ 0.7 0 ⋄ 0.7 0)×size)
      'lnd'  ((0.0 0 ⋄ 0.0 0 ⋄ 0.0 0)×size)]
```

APLAN in Link: DefaultOpts*

```
(
  codeExtensions:   ( 'aplf'
                      'aplo'
                      'apln'
                      'aplc' )
  flatten:          0
  source:           'dir'
  typeExtensions:   [ 2   'apla'
                      3   'aplf'
                      4   'aplo'
                      9.1 'apln'
                      9.4 'aplc'
                      9.5 'apli' ]
  watch:            'ns'
)
```

* Abbreviated slightly.

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

Array Notation and Language Vision

Idea
⌄
Design
⌄
Models
⌄
I-beams ☜
⌄
Editing
⌄
Native
⌄
Tracing

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

Array Notation and

Language Vision

☞ **System Functions**

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

Setting & Getting Variable Values

⎕NS	Name Set*
⎕NG	Name Get
⎕NV	Name–Values

* Consistent extension of existing ⎕NS

Setting & Getting Variable Values

```
myInstance ⎕NS('name1' val1)('name2' val2)
```

```
(data header)←⎕CSV(⎕OPT'Invert' 2) path ⍬ 4 1
table←⎕NS (↑header) data
table.age    ⍝ one of the columns is "age"
```

Setting & Getting Variable Values

```
      jsondata
[{"first":"Kelju","mid":"K.","last":"Kojootti"},
 {"first":"Mikki","last":"Hiiri"}]
      persons←0 ⎕JSON jsondata
      (⊃persons).mid
K.
      persons.mid
VALUE ERROR: Undefined name: mid
      persons.mid
              ∧
```

Setting & Getting Variable Values

```
      jsondata
[{"first":"Kelju","mid":"K.","last":"Kojootti"},
 {"first":"Mikki","last":"Hiiri"}]
      persons←0 ⎕JSON jsondata
      (⊃persons).mid
K.
      persons ⎕NG ⊂'mid' ''
┌──┬┐
│K.││
└──┴┘
```

Setting & Getting Variable Values

```
:If 900⌶0 ⋄ leftArg←42 ⋄ :EndIf
leftArg←⎕NG⊂'leftArg' 42

:Trap (⎕NG⊂'DEBUG' 0)↓0
:Trap ⎕NG⊂'TRAP' 0
```

```
effective ← specified ⎕NG defaults
effective ← ⎕NG/settings
```

Setting & Getting Variable Values

```
:If 900⌶0 ⋄ leftArg←42 ⋄ :EndIf

leftArg←⎕NG⊂'leftArg' 42


:Trap (⎕NG⊂'DEBUG' 0)↓0
:Trap ⎕NG⊂'TRAP' 0

effective ← specified ⎕NG defaults
effective ← ⎕NG/settings
```

Setting & Getting Variable Values

```
(header data)←namespace ⎕NV 2
data (↓header) ⎕CSV path

↑⎕DMX ⎕NV ¯2
┌────────────────┬──────────────────────────────
│Category        │General
├────────────────┼──────────────────────────────
│DM              │
│                │ ┌────────────┬────────┬──────
│                │ │DOMAIN ERROR│      ÷0│     ^
│                │ └────────────┴────────┴──────
├────────────────┼──────────────────────────────
│EM              │DOMAIN ERROR
├────────────────┼──────────────────────────────
│EN              │11
```

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

Array Notation and

Language Vision

☞ **Primitives**

Adám Brudzewsky

Head of Language Design, Dyalog Ltd.

Function Application

Depth

f⍤k

Function Application

```
      Fp←!
      Fp 4 (5 6)
┌──┬───────┐
│24│120 720│
└──┴───────┘
```

Scalar function!

Function Application

```
      Fd←{×/⍳⍵}
      Fd 4 (5 6)
DOMAIN ERROR
Fd[0] Fd←{×/⍳⍵}
            ∧
      Fs←{×/⍳⍵}⍤0
      Fs 4 (5 6)
┌──┬───────┐
│24│120 720│
└──┴───────┘
```

Scalar function!

Function Application

```
      Fd←{×/⍳⍵}
      Fd 4 (5 6)
DOMAIN ERROR
Fd[0] Fd←{×/⍳⍵}
            ∧
      Fs←{×/⍳⍵}⍤0
      Fs 4 (5 6)
┌──┬───────┐
│24│120 720│
└──┴───────┘
```

Scalar function!

Function Application

```
(⌽⍥1) 'FinnAPL' ('APL' 'yhdistyksen')
┌───────┬─────────────────┐
│LPAnniF│┌───┬───────────┐│
│       ││LPA│neskytsidhy││
│       │└───┴───────────┘│
└───────┴─────────────────┘
(⌽⍥2) 'FinnAPL' ('APL' 'yhdistyksen')
┌───────┬─────────────────┐
│LPAnniF│┌───────────┬───┐│
│       ││yhdistyksen│APL││
│       │└───────────┴───┘│
└───────┴─────────────────┘
```

Function Application

```
      (⌽⍤¯2) 'FinnAPL' ('APL' 'yhdistyksen')
┌───────┬─────────────────────┐
│FinnAPL│                     │
│       │ ┌───┬─────────────┐ │
│       │ │LPA│neskytsidhy  │ │
│       │ └───┴─────────────┘ │
└───────┴─────────────────────┘
      (⌽¨¨) 'FinnAPL' ('APL' 'yhdistyksen')
┌───────┬─────────────────────┐
│FinnAPL│                     │
│       │ ┌───┬─────────────┐ │
│       │ │LPA│neskytsidhy  │ │
│       │ └───┴─────────────┘ │
└───────┴─────────────────────┘
```

Data Transformation

Select

$$X \supseteq Y$$

Data Transformation

X⊇Y Select/Permute

```
⬢ Sort    ← {(⍋⍵)⊇⍵}
⬢ Sorts   ← {(⍋⍺)⊇⍵}      ⍝ "sort Y by X"
⬢ Shuffle ← {(?⍨≢⍵)⊇⍵}
⬢ Grade   ← {(bounds⍸⍵)⊇grades}
```

Data Transformation

```
      ⎕←t←3 8⍴⎕A
┌→───────┐
↓ABCDEFGH│
│IJKLMNOP│
│QRSTUVWX│
└────────┘
```

```
      t[(1 8)(2 7)]
HO

      (1 8)(2 7)⊇t
HO
```

Data Transformation

```
      ⎕←s←2 2⍴'Dad'3'Mum'5
┌→───────────┐
↓ ┌→──┐      │
│ │Dad│  3   │
│ └───┘      │
│ ┌→──┐      │
│ │Mum│  5   │
│ └───┘      │
└∊───────────┘
```

```
      s[((2 1)3)((1 1)2)]
ma
      ((2 1)3)((1 1)2)⊇s
ma
```

Function Composition

Behind

f⍛g

Function Composition

Behind

```
f⍛g
```

```
X(f⍛g)Y
```

Function Composition

Behind

```
f⍛g

X(f    )g Y
```

Function Composition

Behind

```
(f X)g Y
  f⍛g
```

```
f⍛g
```

```
(f X)g Y
```

Function Composition

```
(f X)g Y
  f⍛g
(f Y)g Y
```

Behind

```
f⍛g
```

```
(f X)g Y
```

Data Transformation

X⊇Y Select/Permute

```
(f X)g Y
  f⍛g
(f Y)g Y
```

- `Sort    ← {(⍋⍵)⊇⍵}`
- `Sorts   ← {(⍋⍺)⊇⍵}      ⍝ "sort Y by X"`
- `Shuffle ← {(?⍨≢⍵)⊇⍵}`
- `Grade   ← {(bounds⍸⍵)⊇grades}`

Data Transformation

X⊇Y Select/Permute

```
(f X)g Y
  f⍛g
(f Y)g Y
```

- `Sort    ← {(⍋⍵)⊇⍵}`
- `Sorts   ← {(⍋⍺)⊇⍵}      ⍝ "sort Y by X"`
- `Shuffle ← {(?⍨≢⍵)⊇⍵}`
- `Grade   ← {(bounds⍸⍵)⊇grades}`

Data Transformation

X⊇Y Select/Permute

```
(f X)g Y
  f⍛g
(f Y)g Y
```

```
Sort    ← {⍵⍋⍛⊇⍵}
Sorts   ← {⍵⍋⍛⊇⍵}      ⍝ "sort Y by X"
Shuffle ← {⍵(?⍨≢)⍛⊇⍵}
Grade   ← {⍵(bounds∘⍸)⍛⊇grades}
```

Data Transformation

X⊇Y Select/Permute

```
Sort    ← ⍋⍛⊇
Sorts   ← ⍋⍛⊇           ⍝ "sort Y by X"
Shuffle ← (?⍨≢)⍛⊇
Grade   ← (bounds∘⍸)⍛⊇∘grades
```

```
(f X)g Y
  f⍛g
(f Y)g Y
```

Data Transformation

X⊇Y Select/Permute

- Sort ← ⍋⍛⊇
- Sorts ← ⍋⍛⊇ ⍝ "sort Y by X"
- Shuffle ← ?⍨⍤≢⍛⊇
- Grade ← bounds∘⍸⍛⊇∘grades

```
(f X)g Y
  f⍛g
(f Y)g Y
```

Function Composition

X≡Y Match

- `SameAsFirst    ← ⊃⍛≡`
- `HasDuplicates ← ∪⍛≡`
- `Palindrome     ← ⌽⍛≡`
- `IsPermutation ← ⍋⍤⍋⍛≡`

```
   f⍛g
(f  Y)g  Y
```

Function Composition

f⍛g Behind

```
(f X)g Y
  f⍛g
```

```
Whence ← ⍳⍛∊        ⍝ {(⍳⍺)∊⍵}
InPoly ← ⍪⍛⊥        ⍝ {(⍪⍺)⊥⍵}
Shapes ← ⍴⍛⍴        ⍝ {(⍴⍺)⍴⍵}
ToFile ← ⊂⍛⎕NPUT    ⍝ {(⊂⍺)⎕NPUT ⍵}
```

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

Array Notation and

Language Vision

⎕NS
⎕NG
⎕NV
f⍥k
f⍛g
X⊇Y

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

SPECULATIVE

Array Notation and

Language Vision
☞ Primitives

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

21+
ADDITIONS

Speculative Language Vision

⊇¨pairs

Last?

⊇Y

(⊃F⊇)

⊃⌽,Y

⊇⎕VFI

⊃ is First

Speculative Language Vision

Sort?

∧/2≤/ ≤Y

≤Y

∧/2≥/ ≥Y

≥Y

X ≡⍥≤ Y

Speculative Language Vision

Promote and Demote?

```
row1⍪⍤∧row2
```

```
∧Y
```

```
mat←∧row1
mat⍪←row2
```

```
∨Y
```

```
⍪⍤2∨1 3 2 4⍉Y
```

```
,[÷2]Y
```

```
,[⍳2]Y
```

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

SPECULATIVE

Array Notation and

Language Vision

☞ Primitives

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

21+

ADDITIONS

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

Array Notation and

SPECULATIVE
Language Vision
☞ Primitives

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

Speculative Language Vision

High-Rank Set Functions

∪Y ✓   X∪Y   X∩Y   X~Y

Speculative Language Vision

Auto-Sizing Reshape

```
⍰ 2⍴Y
(⌊(×/⍴Y)÷2)2⍴Y
```

Speculative Language Vision

Auto-Sizing Reshape

```
⍰ 2⍴Y
```

```
((×/⍴Y)÷2)2⍴Y
```

Speculative Language Vision

Auto-Sizing Reshape

```
⍰ 2⍴Y
(⌈(×/⍴Y)÷2)2⍴Y
```

Speculative Language Vision

Auto-Sizing Reshape

```
⍰ 2⍴Y
```

```
(⌈(×/⍴Y)÷2)2⍴Y⍪Y
```

Speculative Language Vision

Auto-Sizing Reshape

```
⍰ 2⍴Y
```

```
(⌈(×/⍴Y)÷2)2⍴Y⍪2↑0⍴Y
```

Speculative Language Vision

Auto-Sizing Reshape

```
¯1 2⍴Y
```

Speculative Language Vision

Auto-Sizing Reshape

```
'R'   2⍴Y
```

Speculative Language Vision

Auto-Sizing Reshape

```
0.5 2⍴Y
```

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

SPECULATIVE

Array Notation and

Language Vision

☞ Primitives

Adám Brudzewsky

Head of Language Design, Dyalog Ltd.

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

SPECULATIVE

Array Notation and

Language Vision

☞ Dfns Syntax

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

Speculative Language Vision

Guarded Guards

```
{cond1:cond2:res ⋄ else}
```

```
{⎕NEXISTS ⍵:2=1 ⎕NINFO ⍵:⊃⎕NGET ⍵ ⋄ 0⍴⊂''}
```

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

SPECULATIVE

Array Notation and

Language Vision

21+

EXTENSIONS

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

DYALOG

FinnAPL Autumn Meeting
Suomen APL-yhdistyksen syyskokous

Array Notation and Language Vision

Adám Brudzewsky
Head of Language Design, Dyalog Ltd.

adam@dyalog.com