DRAFT PROPOSAL

DYALOG

Bishnore 2023

Setting and Getting Variable Values

Adám Brudzewsky

I want to...

- get the values of variables using an array of variable names
- set variables using arrays containing names and values
- set a default left argument for an ambivalent tradfn
- base a new namespace on two source namespaces
- query data objects, but some have missing values
- construct a namespace from names and values
- populate class fields from name–value pairs
- convert between tables and namespaces
- check the value of an optional global

I want to...

Set

and

Get

Variable

Values

Ideally, safe and fast too...

Today: Set

`⍎name,'←val'`	Fails if name is `'val'`
`{⍎name,'←⍵'}val`	Doesn't handle multiple variables
`names{⍎⍺,'←⍵'}¨vals`	What if the target is another namespace?
`names{target⍎⍺,'←⍵'}¨vals`	Not a function
`names(target{⍺⍺⍎⍺,'←⍵'}¨)vals`	Cannot be mapped (¨) over namespaces
`target{⍺.{⍎⍺,'←⍵'}¨/⍵}names vals`	Might be nice to have name–value pairs
`target{⍺.{⍎⍺,'←⍵'}/¨⍵}('name1' val1)('name2' val2)`	Slow...

Today: Set performance

```
target{α.{⍺α,'←ω'}/¨ω}('name1' val1)('name2' val2)

target      ⎕WS      ('name1' val1)('name2' val2)


      nvs←100⍴⊂'Data' 42
      ]runtime -c "⎕SE{α.{⍺α,'←ω'}/¨ω}nvs" "⎕SE ⎕WS nvs"

  ⎕SE{α.{⍺α,'←ω'}/¨ω}nvs → 1.8E¯4 |   0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
* ⎕SE ⎕WS nvs            → 4.5E¯5 | -75% ⎕⎕⎕⎕⎕
```

Today: Set

```
{α.{⍺α,'←ω'}/¨ω}
```

Today: Get

`⍎name`	Doesn't handle multiple variables
`⍎¨names`	What if the target is another namespace?
`target⍎¨names`	No fallback value
`vals(target{6::⍺ ⋄ ⍺⍺⍎⍵}¨)names`	Cannot be mapped (¨) over namespaces
`target{⍺.{6::⍵ ⋄ ⍎⍺}¨/⍵}names vals`	Might be nice to have name–value pairs
`target{⍺.{6::⍵ ⋄ ⍎⍺}/¨⍵}('name1' val1)('name1' val1)`	Slow...

Today: Get performance

```
target{α.{6::ω ◇ ⍎α}/¨ω}('name1' val1)('name1' val1)
```

```
□WG
```

Today: Get

```
target{α.{6::ω ◇ ⍟α}/¨ω}('name1' val1)('name1' val1)

target        □WG          'name1'        'name1'
```

Today: Get

```
target{α      ⍤   ⍤⍤ω} 'name1'      'name1'

target      ⎕WG       'name1'      'name1'
```

```
      names←100⍴⊂'Data'
      ]runtime -c ⎕SE{α⍤⍤ω}names "⎕SE ⎕WG names    "

  ⎕SE{α⍤⍤ω}names    → 6.4E¯5 |   0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  ⎕SE ⎕WG names     → 1.7E¯5 | -73% ⎕⎕⎕⎕⎕⎕
```

Today: Get

```
target{α      ⍤   ⍤ω} 'name1'      'name1'

target      ⎕WG       'name1'      'name1'

target{α  .  ⎕OR  ⍤ω} 'name1'      'name1'
```

```
      names←100⍴⊂'Data'
      ]runtime -c ⎕SE{α⍤⍤ω}names "⎕SE ⎕WG names    " ⎕SE{α.⎕OR⍤⍤ω}names

  ⎕SE{α⍤⍤ω}names    → 6.4E¯5 |   0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  ⎕SE ⎕WG names     → 1.7E¯5 | -73% ⎕⎕⎕⎕⎕⎕
  ⎕SE{α.⎕OR⍤⍤ω}names → 1.6E¯5 | -76% ⎕⎕⎕⎕⎕
```

Today: Get

`⍎name`	Doesn't handle multiple vars
`⍎¨names`	Cannot reach another namespace
`target⍎¨names`	No fallback value
`vals(target{6::⍺ ⋄ ⍺⍺⍎⍵}¨)names`	Cannot be mapped over namespaces
`target{⍺.{6::⍵ ⋄ ⍎⍺}¨/⍵}names vals`	Slow
`target{⍺{6::⍵ ⋄ ⍺⍺.⎕OR'n'⊣n←⍵}¨/⍵}names vals`	Fails on namespace values
`target{⍺{6::⍵ ⋄ 2=⍺⍺.⎕NC ⍺:⍺⍺.⎕OR'n'⊣n←⍵ ⋄ ⍺⍺⍎⍺}¨/⍵}names vals`	

Today: Get

Slow

```
target{α{6::ω ◇ 2=αα.⍁NC α:αα.⍁OR'n'⍛n←ω ◇ αα≜α}¨/ω}names vals
```

Today: Get

```
target{α{6::ω ◇ 2=αα.⎕NC α:αα.⎕OR'n'⍝n←ω ◇ αα≠α}¨⍵}names vals
```

```
target{α{0=nc←αα.⎕NC α:ω ◇ 2=nc:αα.⎕OR'n'⍝n←ω ◇ αα≠α}¨⍵}names vals
```

Today: Get

```
      GetTrap←{⍺{6::⍵ ⋄ 2=⍺⍺.⎕NC ⍺:⍺⍺.⎕OR'n'⊣n←⍵ ⋄ ⍺⍺⍟⍺}¨/⍵}

      GetNC←{⍺{0=nc←⍺⍺.⎕NC ⍺:⍵ ⋄ 2=nc:⍺⍺.⎕OR'n'⊣n←⍵ ⋄ ⍺⍺⍟⍺}¨/⍵}


      names←100⍴⊂'Missing'
      vals←100⍴42
      ]runtime -c "⎕SE GetTrap names vals" "⎕SE GetNC names vals"

⎕SE GetTrap names vals → 4.7E¯4 |   0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
⎕SE GetNC   names vals → 7.6E¯5 | -84% ⎕⎕⎕
```

Today: Get

```
{α{0=nc←αα.⎕NC α:ω ⋄ 2=nc:αα.⎕OR'n'⊣n←ω ⋄ αα⍎α}¨/ω}
```

Today: Name–Values

Two separate lists:

```
(names vals)←target{α(⊢,ö⊂⍤¨)α.⍎NL ⍵}ncs
```

Name–value pairs:

```
pairs←target{↑α(⊢,ö⊂⍤)¨α.⍎NL ⍵}ncs
```

Today: Name–Values

Two separate lists:

```
(names vals)←target{α(⊢,ö⊂⍏⍤⍨)α.⌷NL ⍵}ncs
```

Name–value pairs:

```
pairs←target{↑α(⊢,ö⊂⍏)⍤⍨α.⌷NL ⍵}ncs
```

Today: Name–Values

Two separate lists:

```
(names vals)←target{α(⊢,ö⊂{2=α.⎕NC ω:α.⎕OR'n'⍪n←ω ◇ α⍪ω}¨)α.⎕NL ω}ncs
```

Name–value pairs:

```
pairs←target{↑α(⊢,ö⊂{2=α.⎕NC ω:α.⎕OR'n'⍪n←ω ◇ α⍪ω})¨α.⎕NL ω}ncs
```

Today: Name–Values

Two separate lists:

```
(names vals)←target{α(⊢,ö⊂{2=α.⎕NC ω:α.⎕OR'n'⍪n←ω ⋄ α⍪ω}¨)α.⎕NL ω}ncs
```

Name–value pairs:

```
pairs←target{↑α(⊢,ö⊂{2=α.⎕NC ω:α.⎕OR'n'⍪n←ω ⋄ α⍪ω})¨α.⎕NL ω}ncs
```

OK, so what do we do about all this?

□NS: Name Set

```
ref←target □NS ('name1' val1) ('name2' val2)
```

Add vars with values

```
ref←target □NS 'name1' 'name2'
```

Pick values from here

⎕NS: Name Set

`ref←target ⎕NS ('name1' val1) ('name2' val2)`	Add vars with values
`ref←target source.⎕NS 'name1' 'name2'`	Pick values from there
`ref←target ⎕NS ⊂'name1' 'val1'`	Set single variable
`ref←target ⎕NS ref`	Copy vars from ref
`ref←⎕NS …`	New namespace
`… ⎕NS (↑'name1' 'name2') (val1 val2)`	Two separate lists

⎕NG: Name Get

```
vals←source ⎕NG ('name1' val1) ('name2' val2)
```
Values w/ fallbacks

```
vals←source ⎕NG 'name1' 'name2'
```
Values w/o fallbacks

```
vals←source ⎕NG ⊂'name1' val1
```
Single name w/ fallback

```
 ref←source ⎕NG ref
```
Clone w/ PCopy

```
          …←⎕NG …
```
Read ⎕THIS

```
          … ⎕NG (↑'name1' 'name2') (val1 val2)
```
Two separate lists

⎕NV: Name–Values

`(name1 val1)(name2 val2)←source ⎕NV ¯2`	Name–value pairs
`(nameMatrix valueVector)←source ⎕NV 2`	Two separate lists
`…←⎕NV …`	Read ⎕THIS

source & target: flexibility

`ref`	Namespace reference
`ref1 ref2 …`	Several references
`'name'`	Namespace name
`'name1' 'name2' …`	Several names
`ref1 'name1' ref2 …`	Any mixture of the above

everything on right to each on left

result structure from left argument

Let’s see that in context!

Get the values of variables using an array of variable names

```
vals←namespace ⎕NG names
```

Get the values of variables using an array of variable names

```
vals←namespace ⎕NG ↑names
```

Set variables using arrays containing names and values

```
namespace ␣NS names vals
```

Set variables using arrays containing names and values

```
namespace □NS(↑names)vals
```

Set a default left argument for an ambivalent tradfn

```
∇ r←{x} Foo y
  x←⎕NG⊂'x' 42
∇
```

Base a new namespace on two source namespaces

`new←input ⎕NG defaults`

- All names from both namespaces included
- `input`'s values prevail

`new←⎕NG/namespaces`

- All names from all namespaces included
- Leftmost values prevail

Query data objects, but some have missing values

```
myFamily ⎕NG ⊂'kidAges'⍬
families ⎕NG ⊂'kidAges'⍬

myFamily ⎕NG ('kidAges'⍬)('kidNames'(0⍴⊂''))
families ⎕NG ('kidAges'⍬)('kidNames'(0⍴⊂''))
```

Construct a namespace from names and values

```
myns←⎕NS (↑names) values
```

```
myns←⎕NS↓⍉↑names   values
```

Populate class fields from name–value pairs

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

Convert table to namespace

```
(data header)←⎕CSV path 0 4 1
                        (↓⍉data)
```

Convert table to namespace

```
(data header)←⎕CSV path 0 4 1
              (↑header) (↓⍉data)
```

Convert table to namespace

```
(data header)←⎕CSV path ⍬ 4 1
namespace←⎕NS (↑header) (↓⍉⍉data)

(data header)←⎕CSV(⎕OPT'Invert' 2) path ⍬ 4 1
namespace←⎕NS (↑header) data
```

Convert table to namespace

```
(data header)←⎕CSV path ⍬ 4 1

namespace←⎕NS (↑header) (↓⍉↑data)
```

```
(data header)←⎕CSV(⎕OPT'Invert' 3) path ⍬ 4 1

namespace←⎕NS   header  data
```

Convert table to namespace

```
(data header)←⎕CSV path ⍬ 4 1
namespace←⎕NS (↑0(7162⌶)¨header) (↓⍉data)


(data header)←⎕CSV(⎕OPT'Invert' 2) path ⍬ 4 1
namespace←⎕NS (↑0(7162⌶)¨header) data
```

Convert namespace to table

```
(header data)←namespace ⎕NV 2

data (↓header) ⎕CSV path


pairs←namespace ⎕NV ¯2

(header data)←↓⍉↑pairs

]disp header⍪⍉↑data
```

Convert namespace to table

```
(header data)←namespace ⎕NV 2
data (1(7162⌶)⍤↓header) ⎕CSV path


pairs←namespace ⎕NV ¯2
(header data)←↓⍉↑pairs
]disp (1(7162⌶)⍤header),⍉↑data
```

Convert namespace to table

```
ns←0⎕JSON'{"NUM":[1,2,3,4],
           "DA" :["En","To","Tre","Fire"],
           "EN" :["One","Two","Three","Fou
ns ⎕NV ¯2
```

Convert namespace to table

```
ns←0⎕JSON'{"NUM":[1,2,3,4],
           "DA" :["En","To","Tre","Fire"],
           "EN" :["One","Two","Three","Fou
⊃{⍺ ⍵}/ns ⎕NV ¯2
```

Convert namespace to table

```
ns←0⎕JSON'{"NUM":[1,2,3,4],
           "DA" :["En","To","Tre","Fire"],
           "EN" :["One","Two","Three","Fou
⊃{α,↑ω}/ns ⎕NV ¯2
```

DA	En	To	Tre	Fire
EN	One	Two	Three	Four
NUM	1	2	3	4

Convert namespace to table

```
ns←0⎕JSON'{"NUM":[1,2,3,4],
           "DA" :["En","To","Tre","Fire"],
           "EN" :["One","Two","Three","Fou
⊃{α,⍪⍉↑ω}/ns ⎕NV ¯2
```

DA	EN	NUM
En	One	1
To	Two	2
Tre	Three	3

Check the value of an optional global

```
⎕NG⊂'DEBUG' 0

:Trap (⎕NG⊂'DEBUG' 0)↓0

:Trap ⎕NG⊂'DEBUG' 0
```

Setting and Getting Variable Values

DRAFT PROPOSAL

Name Set	Name Get	Name–Value
⎕NS name name…	⎕NG name name…	⎕NV type type…
⎕NS (name val)…	⎕NG (name val)…	
⎕NS names vals	⎕NG names vals	
⎕NS ref	⎕NG ref	⎕NV ref
'target'⎕NS …	'target'⎕NG …	'target'⎕NV …
ref ref… ⎕NS …	ref ref… ⎕NG …	ref ref… ⎕NV …