Language Features of version 18.0 in Depth

Adám Brudzewsky

Language Features of version 18.0 in Depth

Adám Brudzewsky

New

`⎕C`	Case convert
`f⍤⍥g`	Over
`f⍥⍤g`	Atop
`≠Y`	Unique mask
`A⍨⍨`	Constant
`⎕DT`	Date-time
`1200⌶`	Format date-time

Improved

```
⎕JSON⍠'HighRank'
⎕JSON⍠'Dialect'
⎕R/⎕S⍠'Regex'
⎕NPUT⍠'NEOL'
⍳Y
X⊂Y
↑[k]Y
```

New

⎕C	Case convert
f⍤⍥g	Over
f⍥⍤g	Atop
≠Y	Unique mask
A⍨	Constant
⎕DT	Date-time
1200⌶	Format date-time

Improved

```
⎕JSON⍠'HighRank'
⎕JSON⍠'Dialect'
⎕R/⎕S⍠'Regex'
⎕NPUT⍠'NEOL'
⍳Y
X⊂Y
↑[k]Y
```

Case Convert

makes `819⌶` obsolete

`⎕C`

Wait, what?

Uppercase: `1(819⌶)'Y'` ⇨ `1 ⎕C 'Y'`

Lowercase: `819⌶'Y'` ⇨ `⎕C 'Y'`

Wait, what?

Uppercase:	`1(819⌶)Y`	⇨	`1 ⎕C Y`
Lowercase:	`819⌶Y`	⇨	`⎕C Y`
Lowercase:	`0(819⌶)Y`	⇨	`⎕C Y`

Wait, what?

Uppercase: `1(819⌶)Y` ⇨ `1 ⎕C Y`

Lowercase: `819⌶Y` ⇨ `⎕C Y`

Lowercase: `0(819⌶)Y` ⇨ `⎕C Y`

```
Big←{⍺←0
    ⍺:1 ⎕C ⍵
    ⎕C ⍵}
```

Pain without gain?

Pain without gain?

```
      819⌶'Hi'#(3J14 'PI')
DOMAIN ERROR: Invalid right argument
      819⌶'Hi'#(3J14 'PI')
          ^
```

Pain without gain?

```
      819⌶'Hi'#(3J14 'PI')
DOMAIN ERROR: Invalid right argument
      819⌶'Hi'#(3J14 'PI')
          ^
      ⎕C'Hi'#(3J14 'PI')
 hi   #   3J14   pi
```

Pain without gain?

```
      + 'Hi'#(3J14 'PI')
Hi   #   3J¯14   PI


      ⎕C'Hi'#(3J14 'PI')
hello   #   3J14   pi
```

Case Convert

Monadic ⎕C: Case **Fold**

normalisation
for machine comparison

Dyadic ⎕C: Case **Map**

display form
for human readers

Case Folding: ⍁C Y

Case Folding: ⍁C ⍸

Case Mapping: X ⍁C Y

1 : Upper		Origin		¯1 : Lower
A	⇦	A|a	⇨	a
B	⇦	B|b	⇨	b
C	⇦	C|c	⇨	c
D	⇦	D|c	⇨	d
E	⇦	E|e	⇨	e
F	⇦	F|f	⇨	f
G	⇦	G|g	⇨	g

Folding vs Mapping


```
      'Μωυσής'
Μωυσής
      ⍺⍺C'Μωυσής'     ⍝ fold
μωυσήσ
      ¯1⍺⍺C'Μωυσής'   ⍝ map
μωυσής
```

Folding vs Mapping

```
      ⎕C 'Μωυσής' 'ΜΩΥΣΉΣ'      ⍝ fold
μωυσήσ  μωυσήσ
      ¯1 ⎕C 'Μωυσής' 'ΜΩΥΣΉΣ'  ⍝ map
μωυσής  μωυσήσ
```

Folding vs Mapping

```
      ⎕C 'Μωυσής' 'ΜΩΥΣΎΣ'      ⍝ fold
μωυσήσ  μωυσήσ
      1 ⎕C 'Μωυσής' 'ΜΩΥΣΎΣ'    ⍝ map
ΜΩΥΣΎΣ  ΜΩΥΣΎΣ
```

Folding vs Mapping


```
      ⍺⍺C 'Straße' 'STRAßE'      ⍝ fold
straße  straße
      1 ⍺⍺C 'Straße' 'STRAßE'    ⍝ map
STRAßE  STRAßE
```

Folding vs Mapping


```
      ⍷C 'Straße' 'STRAßE'      ⍝ fold
straße  straße
      1 ⍷C 'Straße' 'STRAßE'    ⍝ map
STRAßE  STRAßE


STRASSE  STRASSE
```

Would you ever map?

- All-lowercase to generate URLs or hash-tags

```
'Ο Μωυσής Ζει' ⇨ '#ομωυσήςζει'
```

- All-caps for display purposes

```
'Sale' ⇨ 'S A L E'
```

- Title-case a heading…

```
'Would you ever map?' ⇨
  'Would You Ever Map?'
```

```
t←'Would you ever map?'
```

```
      t←'Would you ever map?'
      ¯1*0,¯1↓' '≠t
1 ¯1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1
```

```
      t←'Would you ever map?'
      ¯1*0,¯1↓' '≠t
1 ¯1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1
      (¯1*0,¯1↓' '≠t)⌷⍨C t
```

```
      t←'Would you ever map?'
      ¯1*0,¯1↓' '≠t
1 ¯1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1
      (¯1*0,¯1↓' '≠t)⌷C t
DOMAIN ERROR: Invalid left argument
      (¯1*0,¯1↓' '≠t)⌷C t
                     ^
```

```
      t←'Would you ever map?'
      ¯1*0,¯1↓' '≠t
1 ¯1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1 ¯1 1 ¯1 ¯1 ¯1
      (¯1*0,¯1↓' '≠t)⌷C t
DOMAIN ERROR: Invalid left argument
      (¯1*0,¯1↓' '≠t)⌷C t
                     ^
      (¯1*0,¯1↓' '≠t)⌷C¨t
Would You Ever Map?
```

Compositions

New

⍠C	Case convert
f⍤⍥g	Over
f⍤⍤g	Atop
≠Y	Unique mask
A⍨	Constant
⎕DT	Date-time
1200⌶	Format date-time

Improved

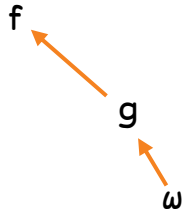
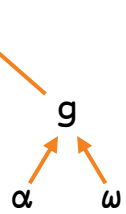
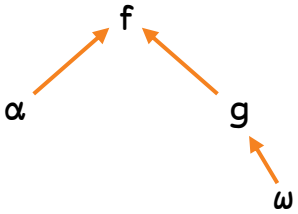
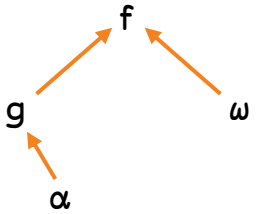
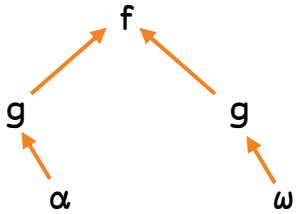
⎕JSON⍠'HighRank'	
⎕JSON⍠'Dialect'	
⎕R⎕S⍠'Regex'	
⎕NPUT⍠'NEOL'	
⍳Y	
X⊂Y	
↑[k]Y	

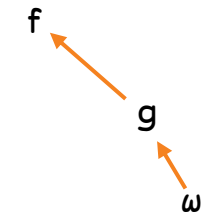
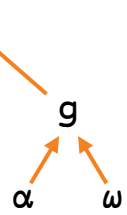
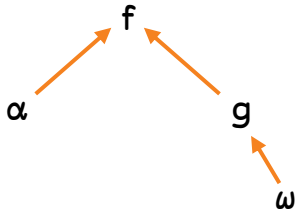
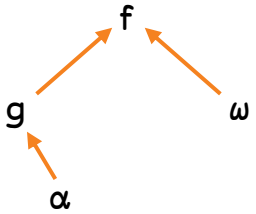
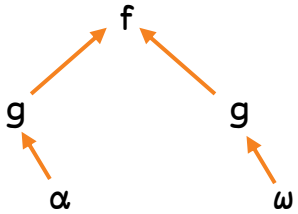
We have a few compositions...

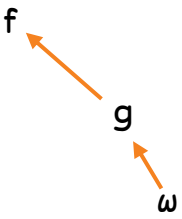
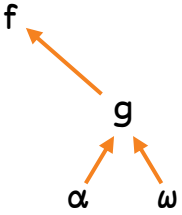
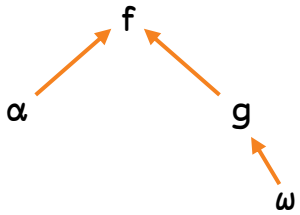
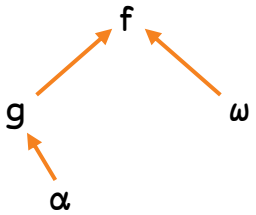
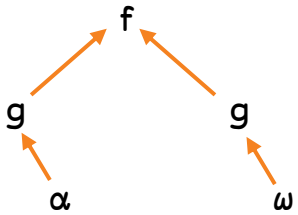
```
      f∘g ⍵     ⇔     f   g ⍵
    (f  g)⍵     ⇔     f   g ⍵
  ⍺ f∘g ⍵     ⇔   ⍺ f   g ⍵
  ⍺(f  g)⍵     ⇔     f ⍺ g ⍵
```

We have a few compositions... but is there a system to these?

```
      f∘g ⍵     ⇔     f   g ⍵
     (f  g)⍵    ⇔     f   g ⍵
   ⍺ f∘g ⍵     ⇔   ⍺ f   g ⍵
   ⍺(f  g)⍵    ⇔     f ⍺ g ⍵
```

	f g	g f	g f g
f Y g Y			
f Y X g Y			
X f Y g Y			

	f g	g f	g f g
fY gY	f∘g 		
fY XgY			
XfY gY	f∘g 	f̈∘g̈ 	

	f g	g f	g f g
fY gY	f⨟g f∘g f⨾g 		
fY XgY	f⨟g 		
XfY gY	f∘g 	f⨟̃∘g⨟̃ 	f⨾g 

Over

on your keyboard as APL+Shift+Over (Shift+⍥)

⍥

Over (⍤) avoids the need to specify g twice

Over (⍤) avoids the need to specify g twice

```
{M←+⌿÷≢ ⋄ (M ⍺) + (M ⍵)}
```

Over (⍤) avoids the need to specify g twice

```
{M←+⌿÷≢ ⋄ (M ⍺) + (M ⍵)}

((+⌿÷≢)⊣) + ((+⌿÷≢)⊢)
```

Over (⍤) avoids the need to specify g twice

```
{M←+⌿÷≢ ⋄ (M ⍺) + (M ⍵)}

((+⌿÷≢)⊣) + ((+⌿÷≢)⊢)

(M⊣) +⍤(M←+⌿÷≢)
```

Over (⍤) avoids the need to specify g twice

```
{M←+⌿÷≢ ⋄ (M α) + (M ω)}

((+⌿÷≢)⊣) + ((+⌿÷≢)⊢)

(M⊣) +∘(M←+⌿÷≢)

+⍤(+⌿÷≢)
```

Over (⍤) represents a very common construct

Sum of squares	`+⍤(*∘2)`
Caseless function application	`≡⍤⎕C ⍳⍤⎕C ∊⍤⎕C` etc.
Anagram?	`≡⍤{⍵[⍋⍵]~' '}`
Juxtaposition (`{⍺ ⍵}`)	`,⍤⊂`
Laminate vectors (`,[1.5]`)	`,⍤⍪`

Over (⍥) is closely related to Beside (∘)

```
    f∘g ⍵   ⇔       f (g ⍵)
    f⍥g ⍵   ⇔       f (g ⍵)
⍺   f∘g ⍵   ⇔     ⍺ f (g ⍵)
⍺   f⍥g ⍵   ⇔ (g ⍺) f (g ⍵)
```

New name!

Over (⍥) is closely related to Beside (∘)

```
      f∘g ⍵   ⇔       f (g ⍵)
      f⍥g ⍵   ⇔       f (g ⍵)
⍺     f∘g ⍵   ⇔     ⍺ f (g ⍵)
⍺     f⍥g ⍵   ⇔ (g ⍺) f (g ⍵)
```

Enough, give it to me already!

```
Over ← {(⍹⍹ ⍺) ⍺⍺ (⍹⍹ ⍵)}
```

Atop

shares symbol with Rank

Atop (⍥) is closely related to Beside (∘)

```
      f ∘ g  ⍵     ⇔      f  g  ⍵
      f ⍥ g  ⍵     ⇔      f  g  ⍵
  ⍺   f ∘ g  ⍵     ⇔   ⍺  f  g  ⍵
  ⍺   f ⍥ g  ⍵     ⇔   f  ⍺  g  ⍵
```

Wait a minute! — We already have atop ( )

```
(f g)ω      ⇔      f g ω
f⍤g ω       ⇔      f g ω
α(f g)ω     ⇔    f α g ω
α f⍤g ω     ⇔    f α g ω
```

- mitigates function/operator duality by forcing function-ness

- mitigates function/operator duality by forcing function-ness

```
{  /  /  ⎕VFI ⍵}   ⇔   {  (/  /) ⎕VFI ⍵}
```

- **mitigates function/operator duality by forcing function-ness**

```
{    /   /  ⎕VFI ⍵}   ⇔   {   (/   /) ⎕VFI ⍵}
{⊃   /   /  ⎕VFI ⍵}   ⇔   {(⊃ /) /   ⎕VFI ⍵}
```

- ⍤ mitigates function/operator duality by forcing function-ness

```
{    /   /  ⎕VFI ⍵}   ⇔   {   (/   /) ⎕VFI ⍵}
{⊃   /   /  ⎕VFI ⍵}   ⇔   {(⊃ /) /   ⎕VFI ⍵}
{⊃ ⊢⍤/   /  ⎕VFI ⍵}   ⇔   { ⊃(/   /) ⎕VFI ⍵}
```

- ⍤ mitigates function/operator duality by forcing function-ness

```
{    /   /  ⎕VFI ⍵}   ⇔   {   (/   /) ⎕VFI ⍵}
{⊃   /   /  ⎕VFI ⍵}   ⇔   {(⊃ /) /   ⎕VFI ⍵}
{⊃ ⊢⍤/ /   ⎕VFI ⍵}   ⇔   { ⊃(/   /) ⎕VFI ⍵}
```

```
{(5<⍵)   /   ⍵} ⇔ { (5<⍵) /   ⍵}
```

⍤ mitigates function/operator duality by forcing function-ness

```
{    /   /  ⎕VFI ⍵}   ⇔   {   (/   /) ⎕VFI ⍵}
{⊃   /   /  ⎕VFI ⍵}   ⇔   {(⊃ /) /   ⎕VFI ⍵}
{⊃ ⊢⍤/   /  ⎕VFI ⍵}   ⇔   { ⊃(/   /) ⎕VFI ⍵}
```

```
{(5<⍵)   /   ⍵} ⇔ { (5<⍵) /   ⍵}
((5<⊢)   /   ⊢) ⇔ {({5<⍵} /) ⍵}
```

- ⍤ mitigates function/operator duality by forcing function-ness

```
{    /   /  ⎕VFI ⍵}   ⇔   {   (/   /) ⎕VFI ⍵}
{⊃   /   /  ⎕VFI ⍵}   ⇔   {(⊃ /) /   ⎕VFI ⍵}
{⊃ ⊢⍤/   /  ⎕VFI ⍵}   ⇔   { ⊃(/   /) ⎕VFI ⍵}

{(5<⍵)   /   ⍵} ⇔ { (5<⍵) /   ⍵}
((5<⊢)   /   ⊢) ⇔ {({5<⍵} /) ⍵}
((5<⊢) ⊢⍤/   ⊢) ⇔ { (5<⍵) /   ⍵}
```

... and make the structure of tacit functions clearer

```
{(Pea Que α Are ω)Ess(Tee You α Vie ω)}
```

... and make the structure of tacit functions clearer

```
{(Pea Que α Are ω)Ess(Tee You α Vie ω)}
((Pea Que)Are)Ess((Tee You)Vie)
```

... and make the structure of tacit functions clearer

```
{(Pea Que α Are ω)Ess(Tee You α Vie ω)}
((Pea Que)Are)Ess((Tee You)Vie)
(Pea∘Que Are)Ess(Tee∘You Vie)
```

... and make the structure of tacit functions clearer

```
{(Pea Que α Are ω)Ess(Tee You α Vie ω)}
((Pea Que)Are)Ess((Tee You)Vie)
(Pea∘Que Are)Ess(Tee∘You Vie)
(Pea∘Que Are)Ess∘Tee∘You Vie
```

... and make the structure of tacit functions clearer

```
{(Pea Que α Are ω)Ess(Tee You α Vie ω)}
((Pea Que)Are)Ess((Tee You)Vie)
(Pea∘Que Are)Ess(Tee∘You Vie)
(Pea∘Que Are)Ess∘Tee∘You Vie
Pea⍤Que⍤Are Ess Tee⍤You⍤Vie
```

Enough, give it to me already!

```
Atop ← {α⍺⍵ ⋄ ⍺⍺ α ⍵⍵ ⍵}
```

Unique Mask

(a.k.a. nub-sieve)

≠⍸

```
      u'Mississippi'
Misp
```

```
      ∪'Mississippi'
Misp
      ≠'Mississippi'
1 1 1 0 0 0 0 0 1 0 0
```

```
      ∪'Mississippi'
Misp
      ≠'Mississippi'
1 1 1 0 0 0 0 0 1 0 0
      'Mississippi'  ↑⍤⍥,⍤⊂  ≠'Mississippi'
```

```
      ∪'Mississippi'
Misp
      ≠'Mississippi'
1 1 1 0 0 0 0 0 1 0 0
      'Mississippi'  ↑⍤⍥,⍤⊂  ≠'Mississippi'
M i s s i s s i p p i
1 1 1 0 0 0 0 0 1 0 0
```

```
      ∪'Mississippi'
Misp
      ≠'Mississippi'
1 1 1 0 0 0 0 0 1 0 0
      'Mississippi' ↑⍤⊢,⍤⊂ ≠'Mississippi'
M i s s i s s i p p i
1 1 1 0 0 0 0 0 1 0 0
      (≠ ⊢⍤⊢/ ⊢) 'Mississippi'
Misp
```

Why, though?

```
≠Y    is to    ∪Y
      as
⍋Y    is to    Sort Y
```

⍋Y vs Sort Y

```
      Sort 3 1 4 1 5
1 1 3 4 5
      ⍋ 3 1 4 1 5
2 4 1 3 5
      'Moses'[2 4 1 3 5]
oeMss
```

≠Y vs ∪Y

```
      ∪ 3 1 4 1 5
3 1 4 5
      ≠ 3 1 4 1 5
1 1 1 0 1
      1 1 1 0 1 / 'Moses'
Moss
```

Non-unique

```
      ∪ 'Mississippi'
Misp
      ≠ 'Mississippi'
1 1 1 0 0 0 0 0 1 0 0
      ~≠ 'Mississippi'
0 0 0 1 1 1 1 1 0 1 1
      (~⍤≠ ⊢⍤≠ ⊢) 'Mississippi'
sissipi
```

Enough, give it to me already!

```
Umask ← ⍳⍨ = ⍳∘≠
```

New

⎕C	Case convert
f⍤⍥g	Over
f⍥⍤g	Atop
≠Y	Unique mask
A⍨	Constant
⎕DT	Date-time
1200⌶	Format date-time

Improved

⎕JSON⍠'HighRank'	
⎕JSON⍠'Dialect'	
⎕R/⎕S⍠'Regex'	
⎕NPUT⍠'NEOL'	
⍳Y	
X⊂Y	
↑[k]Y	

Questions?

New

`⎕C`	Case convert
`f⍤⍥g`	Over
`f⍥⍤g`	Atop
`≠Y`	Unique mask
`A⍨⍨`	Constant
`⎕DT`	Date-time
`1200⌶`	Format date-time

Improved

```
⎕JSON⍠'HighRank'
⎕JSON⍠'Dialect'
⎕R/⎕S⍠'Regex'
⎕NPUT⍠'NEOL'
⍳Y
X⊂Y
↑[k]Y
```

APL and Microsoft Excel

Richard Park

May 27, 2020