Dfns
Past Present Future

[whence what whither]

[quibus quid quo]

[kwo- … kwi]

John Scholes / Dyalog

Dfns

Characterised by:

- Anonymous functions and operators
- Symbolic reference to arguments ⍺ ⍵ ∇ ⍺⍺ ...
- Local names by default
- Lexical scope rules
- Guards
- Recursion instead of iteration
- Optimised Tail calls

Past

September 1989 FP!

1930s Lambda Calculus (Church)
1966 ISWIM (Landin)
1972 SASL; 1981 KRC; 1985 Miranda (Turner)
1989 JBCS “Lazy Functional Languages”
1990 Haskell 1.0
1996 Dyalog Dfns

α/ω in common usage as a notation
KEI direct definitions

Past

Implementation

- Classic (pre-unicode) ⎕AV → α αα
- Built on Control Structs (GRS)
- Built on Dyalog's Dynamic Shadowing mechanism

[Direct/Dynamic] "Dynamic Functions"

Present

dfns.dws
- Utility library
)copy dfns easter
- Coding examples: Google[dyalog easter]
- QA for the interpreter

Lexical scope V18.0 re-implementation [JF+JS]
- Fixes *** bug 17 ***
- Allows operand fns to mutate their environment
- Simpler & quicker

Present

```
Regrets: I have a few ...

abuse←{
     data←⍵
     data←data+1          ⍝ name re-use
     ok⍥←1                ⍝ global assign
     sink←⎕fx ⎕cr'foo'    ⍝ side-effects
     foo data
     surplus lines
}
```

Future

Guarded guards

```
      p: expr
      p: q: expr
      p: q: r: ... expr

      sqrt←{
          2|⎕dr ⍵: 0∧.≤∈⍵: ⍵*÷2
          'Eh?'
      }
```

Future

Declarative thinking: where clauses

```
avg←{
      sum÷num
    ; sum ← +⌿ω
    ; num ← ≢ω
}
```

Future

Declarative thinking: where clauses

```
qsort←{          ⍵ (RH)
     1≥≢⍵:⍵
     ⍵ q p ⍵
     ;    q←(∇<S),=S,(∇>S)
     ;    ;    S←{(⍺ ⍺⍺ ⍵)/⍺}
     ;    p←?∘≢ ⊃ ⊢
}
```

Future

Declarative thinking: where clauses

Tacit form:

```
avg ← sum ÷ num ;
    sum ← +⌿ ;
    num ← ≢
```

Future

Optional type specification:

```
dfs←{    ⍺ depth-first search operator
     ⊃∇⍨/⍵(⊂⍺ ⍺⍺ ⍵),⍺ ⍵⍵ ⍵
   } :: a←a(a←a ∇ t)∇∇([t]←a ∇ t)t
```

Type variables a b c ... z

Future

Optional type specification:

primitive types:

- ~ num
- ' char
- # ref

constructors

- ← result
- ∇ function
- ∇∇ operator
- [] vector

Future

Optional type specification:

```
    {α+ω} ::                          ⍝ type inference
~ ← ~ ∇ ~
```

```
    {2|⎕dr ω:ω ◇ 0≡≡ω:ω } ::   ⍝ multiple types
~ ◇ a ◇
```

Future

Optional type specification:

```
    <  ::                    ⍝ primitive type
~ ← ~ ∇ ~
```

Type definitions:

```
    Date := ~ ~ ~            ⍝ yyyy mm dd
    days :: ~ ← ∇ Date
```

Future

Optional type specification:

This kind of type system wouldn’t go faster

Hard to get: simple ^ flexible ^ rigorous
with a nice default progression

Needs a bigger (more radical) idea

Dfns
Past Present Future

[whence what whither]

[quibus quid quo]

[kwo- … kwi]

John Scholes / Dyalog

Can't fix, won't fix ...

```
      ⎕io←1
      ⍳{
          ⎕io←0
          ⍺⍺ ⍵
      }3
0 1 2
```

Can’t fix, won’t fix ...

```
      ⎕io←1
      {⍳⍵}{
          ⎕io←0
          ⍺⍺ ⍵
      }3
1 2 3
```