

Error Handling

Adám Brudzewsky



Error Handling

Adám Brudzewsky



Overview



Overview

Event

Overview

Event

- Error

Overview

Event

- Error
- Interrupt

Overview

Event

- Error

Overview

Event

● Error

Event Number

Overview

Event

- Error

Event Number

Event Message

Overview

Error

Error Number

Error Message

Overview

Error

Error Number

Error Message

Trapping an Error

Overview

Error

Error Number

Error Message

Trapping an Error

• Tradfn

Overview

Error

Error Number

Error Message

Trapping an Error

- Tradfn

- Dfn

Error

normal execution cannot continue



Error

```
'yes'[4]
```

Error

'yes'[4]



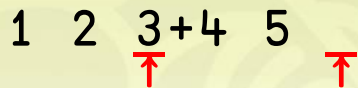
The image shows the text `'yes'[4]` with two red arrows pointing to the characters `'` and `[`. The background features two faint, stylized bird illustrations, one on the left and one on the right, facing each other.

Error

1 2 3+4 5

Error

1 2 3+4 5



The diagram illustrates a sequence of tokens: '1', '2', '3+4', and '5'. Below the token '3+4', a red arrow points to the '+' sign. Below the space between '4' and '5', another red arrow points to the space. This indicates that the parser encountered errors at these positions, likely due to unexpected characters or spacing in the input.

Error

$4 \div 0$

Error

$$\begin{array}{c} 4 \div 0 \\ \uparrow \uparrow \end{array}$$

Error Number

identity of a specific error

□EN

Error Number

`'yes'[4]`

Error Number

```
'yes'[4]
```

```
INDEX ERROR
```

```
'yes'[4]
```

```
^
```

Error Number

```
'yes'[4]  
INDEX ERROR  
      ^  
□EN
```


Error Number

```
      'yes'[4]  
INDEX ERROR  
      'yes'[4]  
      ^  
      □EN  
3
```

Error Number



Error Number

1 2 3+4 5

Error Number

1 2 3+4 5

LENGTH ERROR: Mismatched left and right argument shapes

1 2 3+4 5

^

Error Number

1 2 3+4 5

LENGTH ERROR: Mismatched left and right argument shapes

1 2 3+4 5

^

□EN

5

Error Number



Error Number

$4 \div 0$

DOMAIN ERROR: Divide by zero

$4 \div 0$

^

Error Number

$4 \div 0$

DOMAIN ERROR: Divide by zero

$4 \div 0$

^

□EN

11

Error Message

name for a specific error number

□EM

Error Message

```
'yes'[4]  
INDEX ERROR  
      'yes'[4]  
        ^  
□EN
```

3

Error Message

```
'yes'[4]
INDEX ERROR
'yes'[4]
      ^
□EN
3
□EM 3
```

Error Message

```
'yes'[4]
INDEX ERROR
'yes'[4]
  ^
□EN
3
□EM 3
INDEX ERROR
```

Error Message

'yes'[4]

INDEX ERROR

'yes'[4]

^

□EN

3

□EM 3

INDEX ERROR

Error Message

'yes'[4]

INDEX ERROR

'yes'[4]

^

EN

3

EM 3

INDEX ERROR

Error Message

4 3 5+1 2

LENGTH ERROR: Mismatched left and right argument shapes

4 3 5+1 2

^

□EN

5

Error Message

4 3 5+1 2

LENGTH ERROR: Mismatched left and right argument shapes

4 3 5+1 2

^

□EN

5

□EM 5

Error Message

4 3 5+1 2

LENGTH ERROR: Mismatched left and right argument shapes

4 3 5+1 2

^

□EN

5

□EM 5

LENGTH ERROR

Error Message

4 3 5+1 2

LENGTH ERROR: Mismatched left and right argument shapes

4 3 **5**+1 2 **1**

^

□EN

5

□EM 5

LENGTH ERROR

Error Message

4 3 5÷0

DOMAIN ERROR: Divide by zero

4 3 5÷0

^

□EN

11

Error Message

4 3 5÷0

DOMAIN ERROR: Divide by zero

4 3 5÷0

^

□EN

11

□EM 11

DOMAIN ERROR

Error Message

4 3 5÷0

DOMAIN ERROR: Divide by zero

4 3 5÷0

^

□EN

11

□EM 11

DOMAIN ERROR

Trapping an Error



Trapping an Error

Method	Where	Syntax
structure	tradfns	<code>:Trap</code>
error guard	dfns	<code>::</code>
global state	both	<code>□TRAP</code>

Trapping an Error

Method

Where

Syntax

structure

tradfns

:Trap

error guard

dfns

::

Trapping an Error

tradfn structure

: Trap

Trapping an Error: **:Trap** structure

```
▽ z←Foo y
  :If y
    z←'yes'
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                               Foo 1
  :If y
    z←'yes'
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

```
▽ z ← Foo y                                Foo 1
  :If y                                     yes
    z ← 'yes '                             done
  :Else
    z ← 'oh no '
  :End
  z ← ↑ z 'done '
```

▽

Trapping an Error: **:Trap** structure

```
▽ z ← Foo y                                Foo 1
  :If y                                     yes
    z ← 'yes'                               done
  :Else                                     Foo 0
    z ← 'oh no'
  :End
  z ← ↑ z 'done'
```

▽

Trapping an Error: **:Trap** structure

```
▽ z ← Foo y                                Foo 1
  :If y                                     yes
    z ← 'yes '                             done
  :Else                                     Foo 0
    z ← 'oh no '                           oh no
  :End                                     done
  z ← ↑ z 'done '
```

▽

Trapping an Error: **:Trap** structure

```
▽ z←Foo y
  :If y
    z←'yes'
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

▽ z←Foo y

:Trap 11

z←'yes'

:Else

z←'oh no'

:End

z←↑z 'done'

▽

Trapping an Error: **:Trap** structure

```
▽ z←Foo y
  :Trap 11
    z←4 3 5÷y
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                               Foo 1
  :Trap 11
    z←4 3 5÷y
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                                Foo 1
  :Trap 11                                4 3 5 0
    z←4 3 5÷y                             d o n e
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

▽ `z←Foo y`

`:Trap 11`

`z←4 3 5÷y`

`:Else`

`z←'oh no'`

`:End`

`z←↑z 'done'`

▽

Foo 1

4 3 5 0

d o n e

Foo 0

Trapping an Error: **:Trap** structure

```
▽ z←Foo y
  :Trap 11
    z←4 3 5÷y
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

```
Foo 1
4 3 5 0
d o n e
Foo 0
oh no
done
```

Trapping an Error: **:Trap** structure

```
▽ z←Foo y
  :Trap 11
    z←4 3 5÷y
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                               Foo 1 2
  :Trap 11
    z←4 3 5÷y
  :Else
    z←'oh no'
  :End
  z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

```

▽ z←Foo y
  :Trap 11
    z←4 3 5÷y
  :Else
    z←'oh no'
  :End
  z←↑z 'done'

```



```

Foo 1 2
LENGTH ERROR: Mismatched
Foo[2] □←4 3 5÷y
      ^

```


Trapping an Error: **:Trap** structure

```
▽ z←Foo y
  :Trap 11 5
    z←4 3 5÷y
  :Else
    z←'oh no'
  :End
z←↑z 'done'
```

▽

Trapping an Error: **:Trap** structure

▽ `z←Foo y` Foo 1 2
 :Trap 11 5
 `z←4 3 5÷y`
 :Else
 `z←'oh no'`
 :End
 `z←↑z 'done'`

▽

Trapping an Error: **:Trap** structure

▽ $z \leftarrow \text{Foo } y$

:Trap 11 5

$z \leftarrow 4 \ 3 \ 5 \div y$

:Else

$z \leftarrow \text{'oh no'}$

:End

$z \leftarrow \uparrow z \text{'done'}$

▽

Foo 1 2

oh no

done

Trapping an Error: **:Trap** structure

▽ z ← Foo y

:Trap 11 5

z ← 4 3 5 ÷ y

:Else

z ← 'oh no'

:End

z ← ↑ z 'done'

▽

Foo 1 2

oh no

done

Foo 0

Trapping an Error: **:Trap** structure

▽ `z←Foo y`

`:Trap 11 5`

`z←4 3 5÷y`

`:Else`

`z←'oh no'`

`:End`

`z←↑z 'done'`

▽

Foo 1 2

oh no

done

Foo 0

oh no

done

Trapping an Error: **:Trap** structure

▽ z ← Foo y

:If y

z ← 'yes'

:Else

z ← 'oh no'

:End

z ← ↑ z 'done'

▽

Foo 1

yes

done

Foo 0

oh no

done

Trapping an Error: **:Trap** structure

```
▽ z ← Foo y                                Foo 1
  :Select y                                yes
  :Case 1                                done
    z ← 'yes'                                Foo 0
  :Case 0                                oh no
    z ← 'oh no'                                done
  :End
  z ← ↑ z 'done'
```



Trapping an Error: **:Trap** structure

```
▽ z←Foo y                                Foo 1 2
  :Trap 11 5                             count!
    z←4 3 5÷y                             done
  :Case 11                               Foo 0
    z←'bad!'                             bad!
  :Case 5                               done
    z←'count!'
  :End
z←↑z 'done'
```


Trapping an Error: **:Trap** structure

▽ z←Foo y

:Trap 0

z←4 3 5÷y

:Case 11

z←'bad!'

:Case 5

z←'count!'

:End

z←↑z 'done'

Foo 1 2

count!

done

Foo 0

bad!

done

Trapping an Error: **:Trap** structure

```
▽ z←Foo y
  :Trap 0
    z←4 3 5÷y
  :Case 11
    z←'bad!'
  :Case 5
    z←'count!'
  :Else
    z←'oh no'
```

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                               Foo"(1 2) 0
  :Trap 0
    z←4 3 5÷y
  :Case 11
    z←'bad!'
  :Case 5
    z←'count!'
  :Else
    z←'oh no'
```

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                                Foo"(1 2) 0
  :Trap 0                                count! bad!
    z←4 3 5÷y                            done done
  :Case 11
    z←'bad!'
  :Case 5
    z←'count!'
  :Else
    z←'oh no'
```

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                                Foo"(1 2) 0
  :Trap 0                                count! bad!
    z←4 3 5÷y                            done done
  :Case 11                               Foo 1 2p3
    z←'bad!'
  :Case 5
    z←'count!'
  :Else
    z←'oh no'
```

Trapping an Error: **:Trap** structure

```
▽ z←Foo y                                Foo"(1 2) 0
  :Trap 0                                count! bad!
    z←4 3 5÷y                            done done
  :Case 11                               Foo 1 2p3
    z←'bad!'                             oh no
  :Case 5                                done
    z←'count!'
  :Else
    z←'oh no'
```

Trapping an Error: **:Trap** structure

```

▽ z←Foo y
  :Trap 0
    z←4 3 5÷y
  :Case 11
    z←'bad!'
  :Case 5
    z←'count!'
  :Else
    z←'oh no'

```

Foo (1 2) 0
 bad!
 done
 Foo 1 2p3
 oh no
 done

RANK ERROR (4)

Messages instead of Numbers

"□EM*-1"

Messages instead of Numbers

- Simple variables
- Errors' namespace
- EM inverse function

Messages instead of Numbers: variables

(INDEX RANK LENGTH VALUE) ← 3 4 5 6

Messages instead of Numbers: variables

(INDEX RANK LENGTH VALUE) ← 3 4 5 6

:Trap RANK LENGTH

Messages instead of Numbers: namespace

ERROR ← ⊞ NS 0

Messages instead of Numbers: namespace

ERROR ← NS0

names ← ' ERROR ' '\W+' 'R' ' ' _ ' EM 199

Messages instead of Numbers: namespace

```
ERROR ← NS0
```

```
names ← ' ERROR ' ' \W+' R ' ' _ ' EM 199
```

```
ERROR 10 ' ( ' , names , ' ) ← ' , 199
```

Messages instead of Numbers: namespace

```
ERROR ← NS0
```

```
names ← ' ERROR ' ' \W+' R ' ' _ ' EM 199
```

```
ERROR 100 ' ( ' , names , ' ) ← ' , 199
```

```
:Trap ERROR.DOMAIN
```

Messages instead of Numbers: namespace

```
ERROR ← NS0
```

```
names ← ' ERROR ' ' \W+' R ' ' _ ' EM 99
```

```
ERROR 99 ' ( ' , names , ' ) ← ' , 99
```

```
:Trap ERROR.DOMAIN
```

```
:Trap ERROR.(RANK LENGTH)
```


Messages instead of Numbers: function

`ERROR ← ( ' ERROR ' ⍲R ' ' ⍲EM 99 ) ⍲⊆`

Messages instead of Numbers: function

```
ERROR ← ( ' ERROR ' ⍲R ' ' ⍲EM 99 ) ⍲⊆
```

```
:Trap ERROR 'RANK' 'LENGTH'
```

Trapping an Error

dfn error guard

::

Trapping an Error: :: error guard

```
Foo ← {  
  'yes' [ω]  
}
```

Trapping an Error: :: error guard

```
Foo ← {  
  'yes' [ω]  
}
```

```
Foo 4  
INDEX ERROR  
Foo[1] 'yes' [ω]  
      ^
```

Trapping an Error: :: error guard

```
Foo ← {  
  3 :: 'oh no'  
  'yes' [ω]  
}
```

Trapping an Error: :: error guard

```
Foo ← {  
  3 :: 'oh no'  
  'yes' [ω]  
}
```

```
                Foo 3  
2  
                Foo 4  
oh no
```

Trapping an Error: :: error guard

```
Foo ← {  
  3 11 :: 'oh no'  
  'yes' [ω]  
}
```


Trapping an Error: :: error guard

```
Foo ← {  
  3 11 :: 'oh no'  
  'yes' [ω]  
}
```

```
Foo '4'  
oh no  
Foo 4  
oh no
```

Trapping an Error: :: error guard

```
Foo ← {  
  0 :: 'oh no'  
  'yes' [ω]  
}
```

```
Foo '4'  
oh no  
Foo 4  
oh no
```

Trapping an Error: :: error guard

```
Foo←{
```

```
  1 1 :: 'bad!'
```

```
  5 :: 'count!'
```

```
  4 3 5÷ω
```

```
}
```

```
Foo 1 2
```

```
count!
```

```
Foo 0
```

```
bad!
```

Trapping an Error: :: error guard

```

Foo←{
  1 1 :: z←'bad!'
  5 :: z←'count!'
  z←4 3 5÷ω
  ↑z 'done'
}

```

```

Foo 1 2
count!
done
Foo 0
bad!
done

```

Trapping an Error: :: error guard

```
Foo ← {
```

```
  11 :: z ← 'bad!'
```

```
  5 :: z ← 'count!'
```

```
  z ← 4 3 5 ÷ ω
```

```
  ↑ z 'done'
```

```
}
```

```
Foo 1 2
```

```
count!
```

```
done
```

```
Foo 0
```

```
bad!
```

```
done
```

Trapping an Error: :: error guard

```

Foo ← {
  z ← {
    1 1 :: 'bad!'
    5 :: 'count!'
    4 3 5 ÷ ω
  } ω
  ↑ z 'done'
}

```

Foo 1 2
 count!
 done
 Foo 0
 bad!
 done

Trapping an Error: :: error guard

```

Foo←{
  Bar←{
    1 1 :: 'bad!'
    5 :: 'count!'
    4 3 5÷ω
  }
  ↑(Bar ω) 'done'
}

```

```

Foo 1 2
count!
done
Foo 0
bad!
done

```

Error Handling – Part 1

Tradfn structure `:Trap errno` (0 means all)

Dfn error guard `errno::result_expression`

Error Number of last error `□EN`

Error Message for number `□EM`

Error Handling – Part 1

Number for given message
aplcart.info?q=:: message

Tradfn structure :Trap errno (0 means all)

Dfn error guard errno::result_expression

Error Number of last error □EN

Error Message for number □EM

Error Handling – Part 1

Number for given message
 aplcart.info?q=:: message

Tradfn structure :Trap errno (0 means all)

Dfn error guard errno::result_expression

Error N for

□EN

□EM

Press  for full docs!

Upcoming webinars

Feb	25	BAA	open session
Mar	4	---	
Mar	11	BAA	open session
Mar	18	Dyalog	Boolean Scans and Reductions
Mar	25	BAA	open session
Apr	1	---	
Apr	8	BAA	open session
Apr	15	Dyalog	Error Handling – Part 2

Upcoming webinars

Feb	25	BAA	open session
Mar	4	---	
Mar	11	BAA	open session
Mar	18	Dyalog	Boolean Scans and Reductions
Mar	25	BAA	open session
Apr	1	---	
Apr	8	BAA	open session
Apr	15	Dyalog	Error Handling – Part 2

britishaplassociation.org