Pharo: Syntax in a Nutshell

S. Ducasse

http://www.pharo.org

Less is More

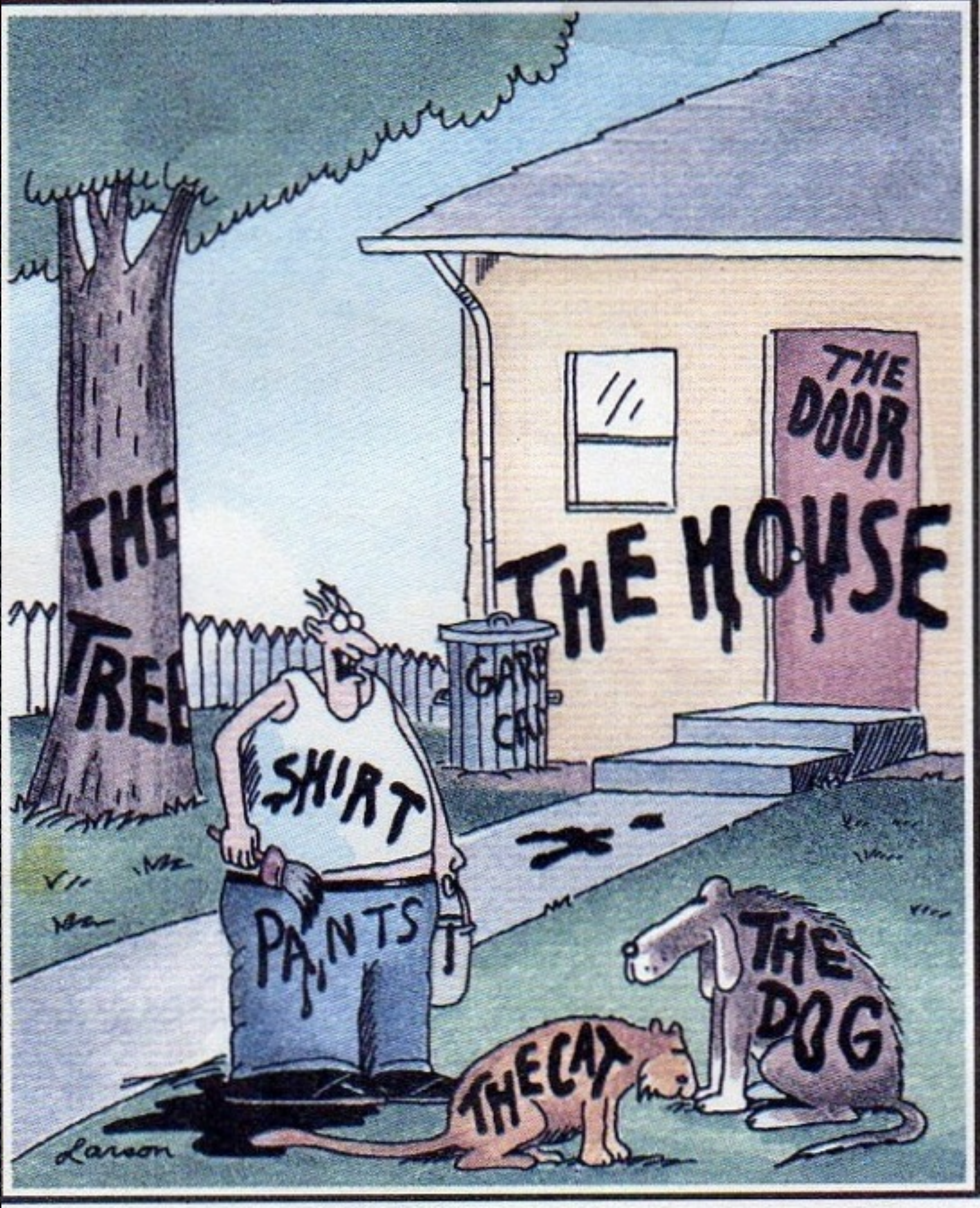
- No constructors
- No types declaration
- No interfaces
- No packages/private/protected
- No parametrized types
- No boxing/unboxing
- **And really powerful**

```
ArrayList<String> strings
                = new ArrayList<String>();
```

strings := ArrayList new.

```
Thread regThread = new Thread(
       new Runnable() {
         public void run() {
              this.doSomething();} });
regThread.start();
```

[self doSomething] fork.



"Now! ... *That* should clear up a few things around here!"

A Pure OO World

Only objects!

mouse, booleans, arrays, numbers, strings, windows, scrollbars, canvas, files, trees, compilers, sound, url, socket, fonts, text, collections, stack, shortcut, streams, …

3 kinds of messages

Unary messages

```
5 factorial
Transcript cr
```

Binary messages

```
3 + 4
```

Keywords messages

```
2 between: 0 and: 5

Transcript show: 'hello world'
```

```
postman.send(mail,recipient);
postman.send(mail,recipient);
```

postman send mail recipient

postman send mail to recipient

postman send: mail to: recipient

() > Unary > Binary > Keywords

```
1 class maxVal + 1

(1 class maxVal + 1) class

1 class maxVal raised: 3+ 2
```

Typical Expression

```
ZnEasy client
    url: 'http://bugs.pharo.org/issues/name/', text asString;
    get;
    response.
```

Some basic objects

- String: 'a string'
- Symbole (unique String): #aSymbol
- Character: $A
- Array: #(1 2 3)   { 1 . 2 . 3}
- OrderedCollection: OrderedCollection new add: 35; add: 45; yourself
- Set: Set new add: 1; add: 2; yourself

yourself?

```
Set new add: 1; add: 2; yourself
```

- is equivalent to

```
| s |
s := Set new.
s add: 1; add: 2.
s
```

Block: Lambda Expressions

```
[ :x | x + 2 ] value: 5

    -> 7
```

- anonymous method
- [ ]
- :x is the block arguments

Blocks can be stored

```
| b |
b := [ :x | x + 2 ].
b value: 5
    -> 7
b value: 33
    -> 35
```

Conditionals: ifTrue:ifFalse:

- Booleans are objects
- Conditional are messages sent to booleans or block

```
initialAnswer := fullName isEmptyOrNil
    ifTrue: ['FirstnameLastname' translated]
    ifFalse: [fullName].
```

ifTrue:

```
forceItalicOrOblique
        self slantValue = 0 ifTrue: [ slantValue := 1 ]
```

ifFalse:ifTrue:

```
index = 0
    ifFalse: [values at: index]
    ifTrue: [self privateAt: key put: aBlock value]
```

ifEmpty:

```
(myProtocol
    ifEmpty: ['As yet unclassified’]

self listItems
        ifNotEmpty: [ :aList | aList at: currentIndex  ]
```

(...) vs. [ ]

- Use [ ] when you do not know how many time it is executed.
- (x isNil) ifTrue: [....]

Some loops

- 4 timesRepeat: [self doSomething]
- 0 to: 100 [:i | ...]
- 0 to: 100 by: 3 [:i | ...]
- aCol do: [:each | ...]

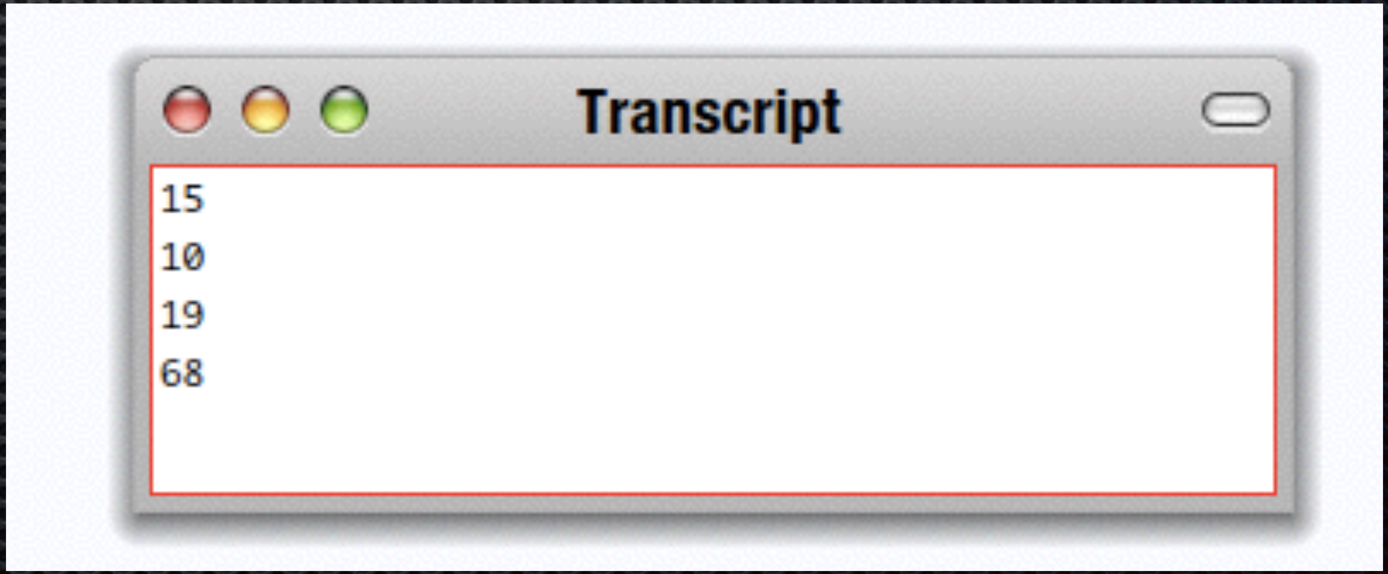
Some loops: to:by:do:

```
| sum |
0 to: 100 by: 3 [ :i | sum= sum + i ]
```

```
#(15 10 19 68) do:

    [:i | Transcript show: i ; cr ]
```

```
#(15 10 19 68) do:
    [:i | Transcript show: i ; cr ]
```



```
15
10
19
68
```

```
#(2 -3 4 -35 4) collect: [ :each| each abs]
```

```
#(2 -3 4 -35 4) collect: [ :each| each abs]

> #(2 3 4 35 4)
```

Defining a method

- (2@3) <= (5@6)

```
<= aPoint
	"Answer whether the receiver is neither below nor to the right of aPoint."
	^ x <= aPoint x and: [y <= aPoint y]
```

Class definition

```
Object subclass: #Point
    instanceVariableNames: 'x y'
    classVariableNames: ''
    category: 'Graphics-Primitives'
```

Conclusion

- Messages (unary, binary, keywords)
- Blocks
- Methods are named blocks