Constant Blocks Pharo11

Marcus Denker

Inria Evref

The Problem

Morph>>#minHeight

```
minHeight
    "answer the receiver's minHeight"
    ^ self
        valueOfProperty: #minHeight
        ifAbsent: [2]
```

The Problem

Morph>>#minHeight

```
minHeight

"answer the receiver's minHeight"

^ self

valueOfProperty: #minHeight

ifAbsent: [2]
```

The Problem

all Objects understand #value, so we can write

```
minHeight
    "answer the receiver's minHeight"
    ^ self
        valueOfProperty: #minHeight
        ifAbsent: 2
```

Compare Speed

```
m := Morph new.
[m minHeight] bench
```

~3 times faster

Compare Speed

What is going on?

Block: creation

- Compiler adds a CompiledBlock to the literal frame
- Block creation bytecode creates instance of FullBlockClosure on the Stack

```
49 <4C> self
50 <20> pushConstant: #minHeight
51 <F9 01 00> fullClosure:[ 2 ] NumCopied: 0
54 <A2> send: valueOfProperty:ifAbsent:
55 <5C> returnTop
```

Blocks: execution

- Here is #value of BlockClosure
- Primitive that tells the VM to execute the CompiledBlock

value

```
"Activate the receiver, creating a closure activation (MethodContext)
whose closure is the receiver and whose caller is the sender of this
message. Supply the copied values to the activation as its copied
temps. Primitive. Essential."
<primitive: 207>
numArgs ~= 0 ifTrue:
    [self numArgsError: 0].
^self primitiveFailed
```

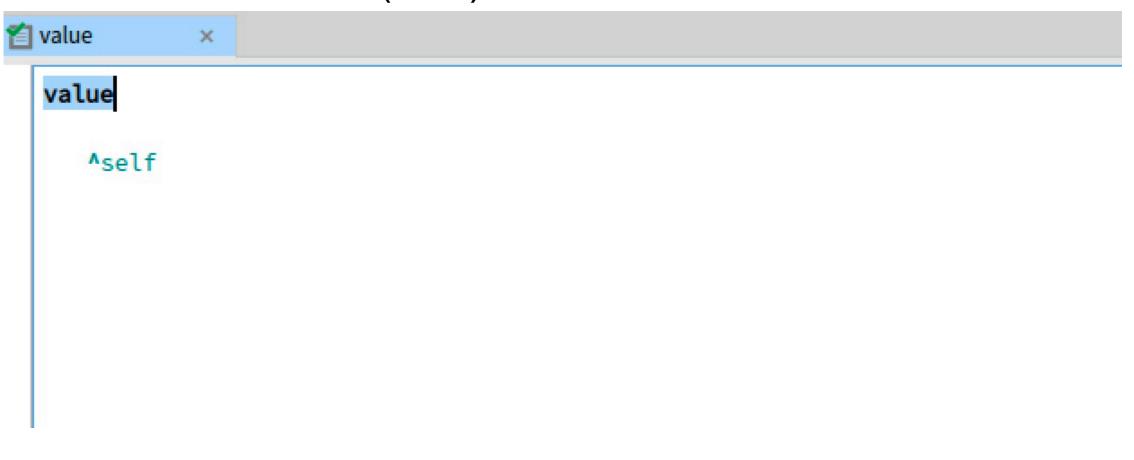
Constant: "creation"

- Compiler adds the constant to the literal frame
- pushLiteral: bytecode pushed it on the stack

```
49 <4C> self
50 <20> pushConstant: #minHeight
51 <21> pushConstant: 2
52 <A2> send: valueOfProperty:ifAbsent:
53 <5C> returnTop
```

Constant: execution

- Here is #value of Object
- Just a return self (fast!)



So know we know why

- Block creation is slow as it creates an object
 - vs just a pushLiteral
- #value is slow because it executes two method
 - #value
 - the CompiledBlock

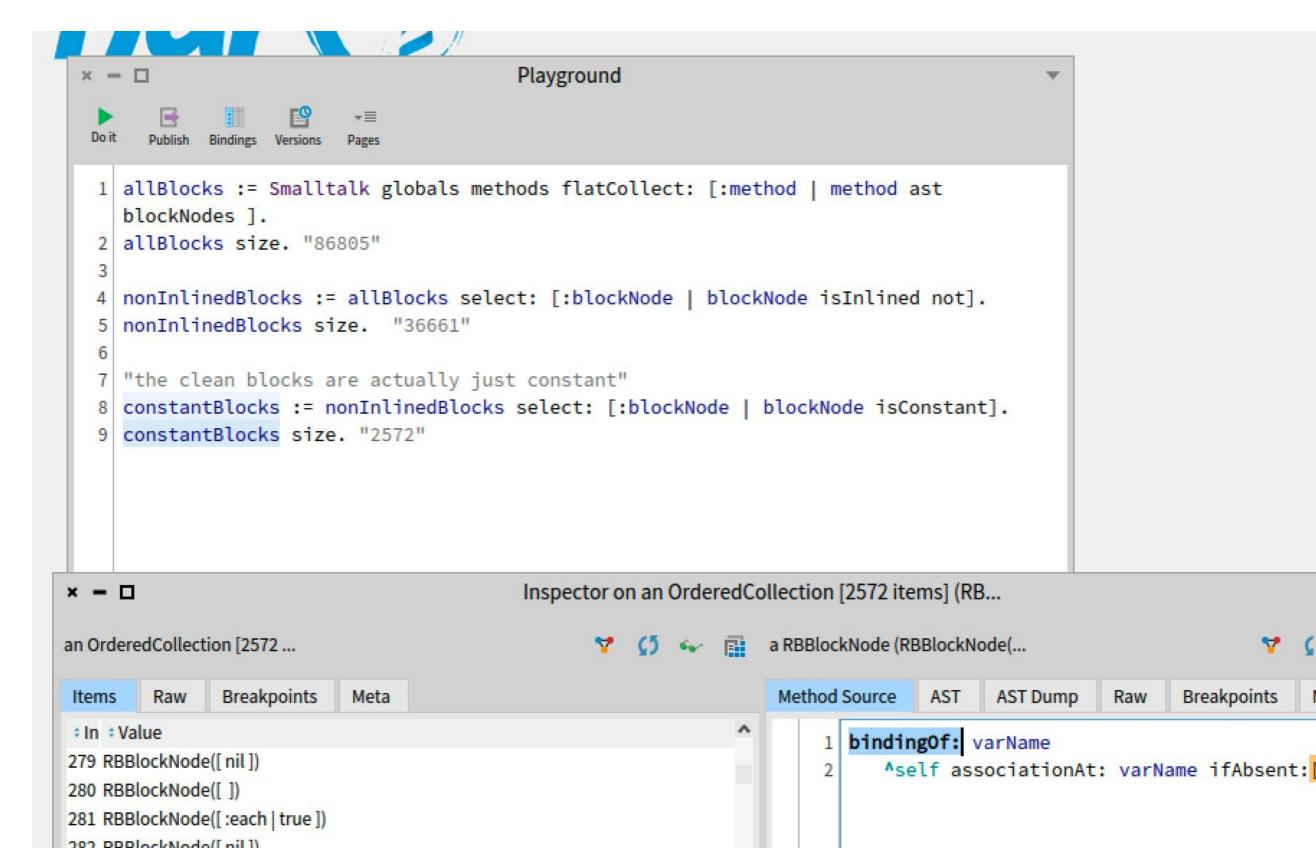
How often do we use them?

```
allBlocks := Smalltalk globals methods
    flatCollect: [:method | method ast blockNodes].
allBlocks size.

nonInlinedBlocks := allBlocks select: [:blockNode |
    blockNode isInlined not].
nonInlinedBlocks size.

"the blocks are actually just constant"
constantBlocks := nonInlinedBlocks select: [:blockNode |
    blockNode isConstant].
constantBlocks size.
```

How often do we use them?



What do we do?

- We started to rewrite code like that in some cases to use the #literal instead of [#literal]
 - But it looks ugly!
 - What about [:arg | #literal]?
 - There is no #value: implemented in Object!

Why not just compile constants?

- Why not just change the compiler to compile a literal instead of a constant block?
 - What about more than 1 arg?
 - Mapping source <-> bc ?

Better Solution: Wrap it

- We can create an object with the interface of BlockClosure
- It stores the constant as an ivar
- Created by the compiler, stored in the literal frame
- #value is a quick return of the ivar

But senders-of?

- System scans the CompiledBlock for many queries (e.g. senders-of)
- Solution: we compile a CompiledBlock even though we never execute it
- This way CompiledBlock has really the same API as a CleanBlock
- It can even be a subclass!

Multiple Arguments

- Object just implements #value
- But our Wrapper can implement #value, #value:value: and so on

We need error handling

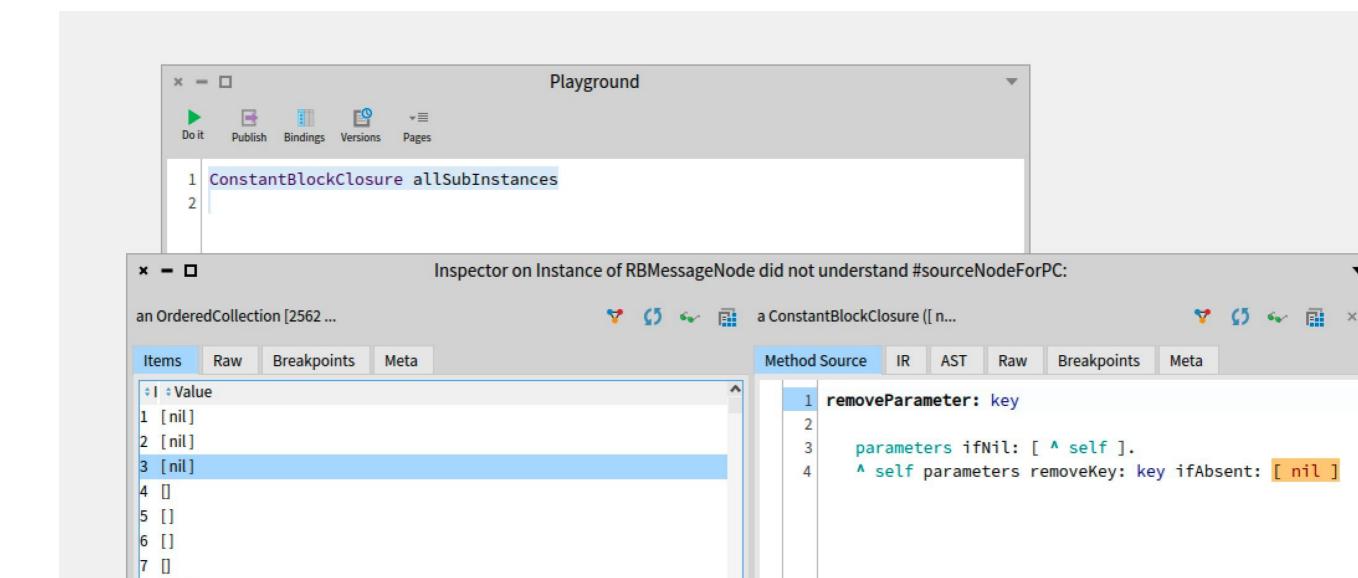
- When using value: on a 0 arg block, we need to raise errors
- Therefore: subclasses
- Let's look at the Hierarchy

The Compiler

 We just test for isConstant and then call #visitConstantBlockNode:

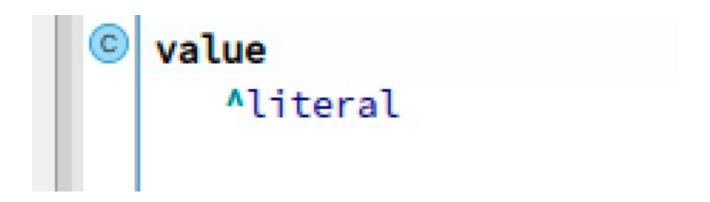
Mapping works

Let's inspect all instances of CleanBlockClosure



It is called a lot

Add counter to #value of ConstantBlockClosure



What about CleanBlocks?

- I did not talk about CleanBlocks... as this is another talk
- ConstantBlocks are a subset of all clean blocks
- Same creation speed (created at compile time)
- But: Constant block *execution* is faster than that of a clean block

We learned...

- Blocks are slow
- For Constant blocks we can easily do better
 - Both Creation and Evaluation

Constant Blocks are active by default in Pharo11

Questions?

- Blocks are slow
- For Constant blocks we can easily do better
 - Both Creation and Evaluation

Constant Blocks are active by default in Pharo11