IronSmalltalk

August 24th 2011

Todor Todorov

IronSmalltalk

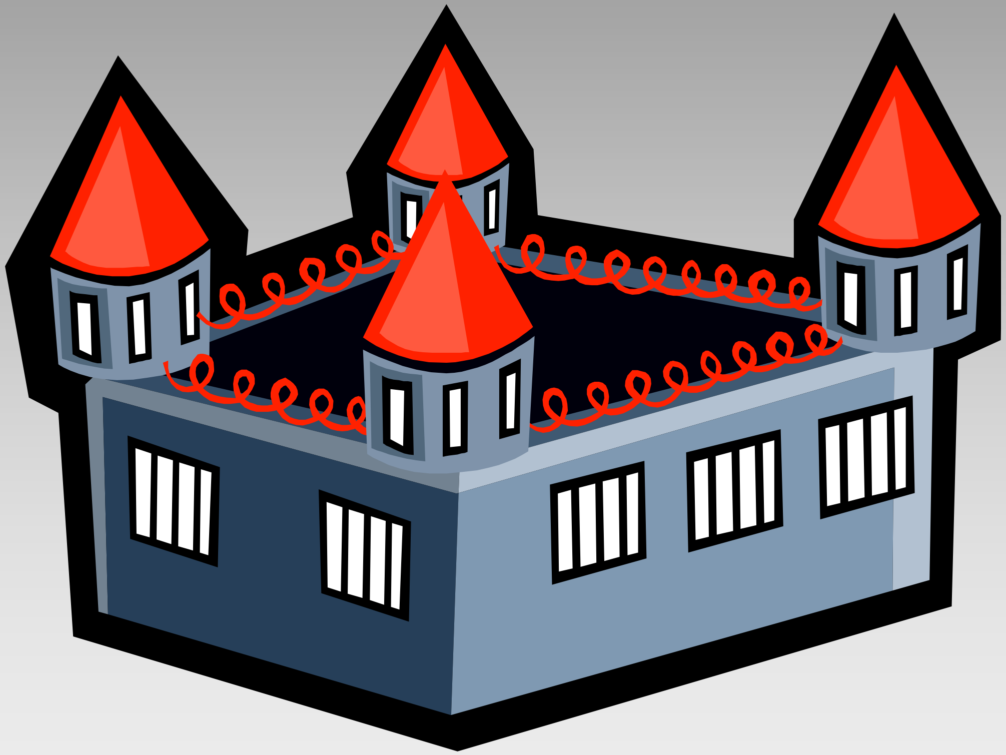
Smalltalk for the
Microsoft .Net DLR

Agenda

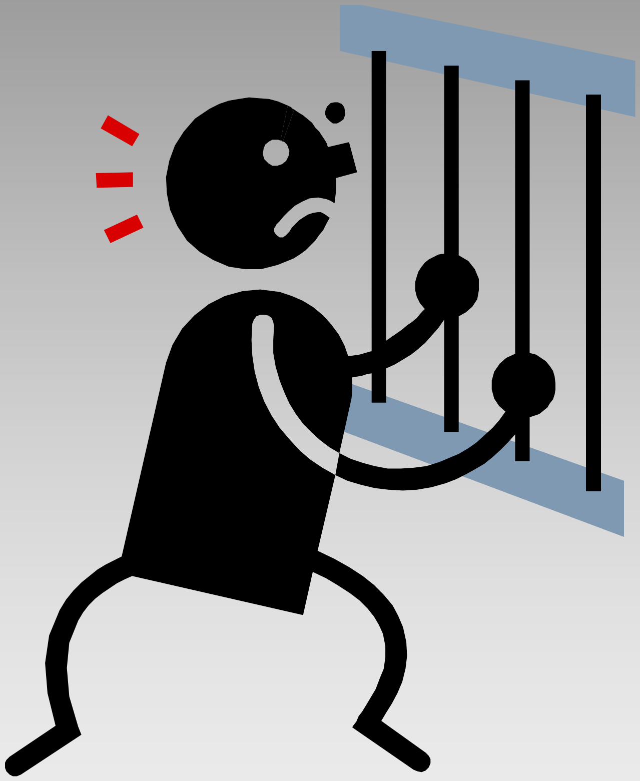
- History
- Motivation and Background
- Microsoft .Net DLR
- Message Sends and CallSites
- CallSite Binders
- Expression Trees
- Code Pipeline
- Polymorphic Inline Caching
- Conclusion
- Extras / Bonus

HISTORY

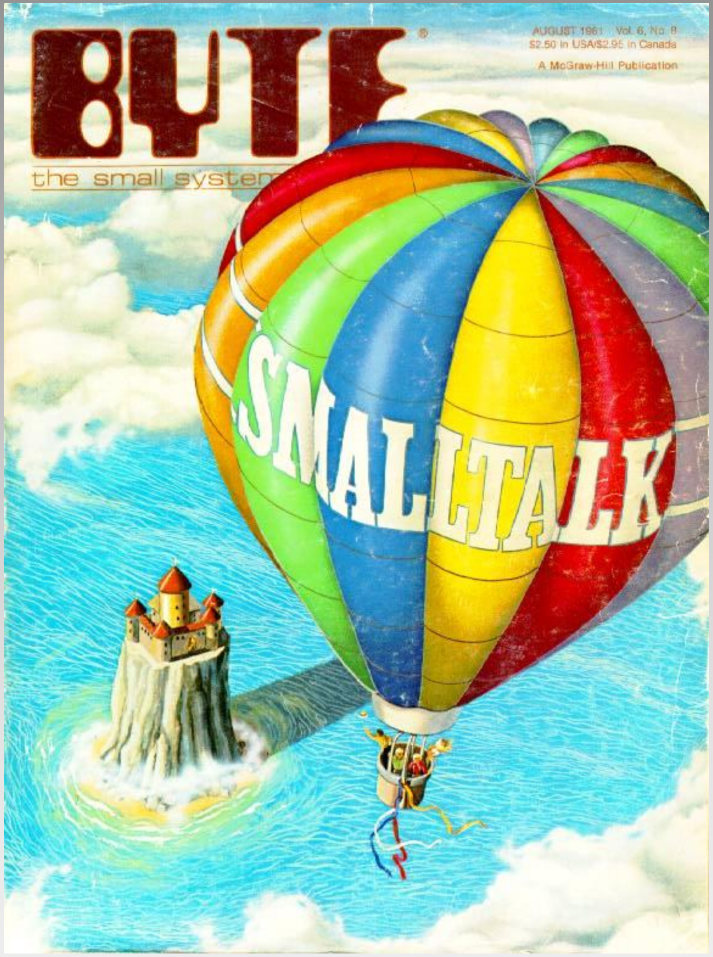
Smalltalk 1980



Smalltalk 1980



Smalltalk 1981


BYTE
the small system
AUGUST 1981 Vol. 6, No. 8
$2.50 in USA/$2.95 in Canada
A McGraw-Hill Publication
SMALLTALK

Smalltalk 2011

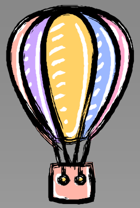
Edinburgh
ESUG 2011

Smalltalk 2005

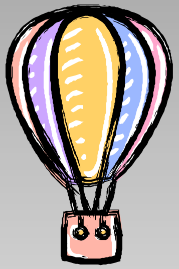
ESUG 2005
Brussels

The World Today

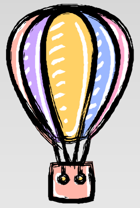
The World Today



Smalltalking since 1998



Todor Todorov



Business Applications

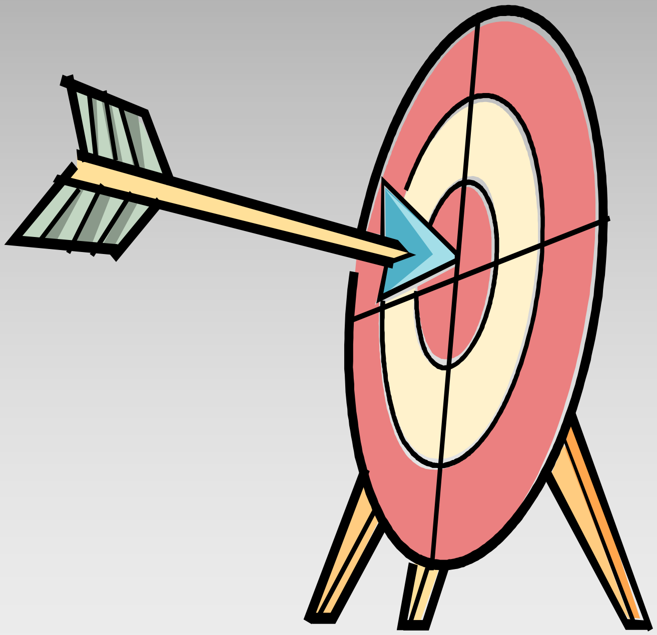
Microsoft .Net Platform

Costly vs. Free?

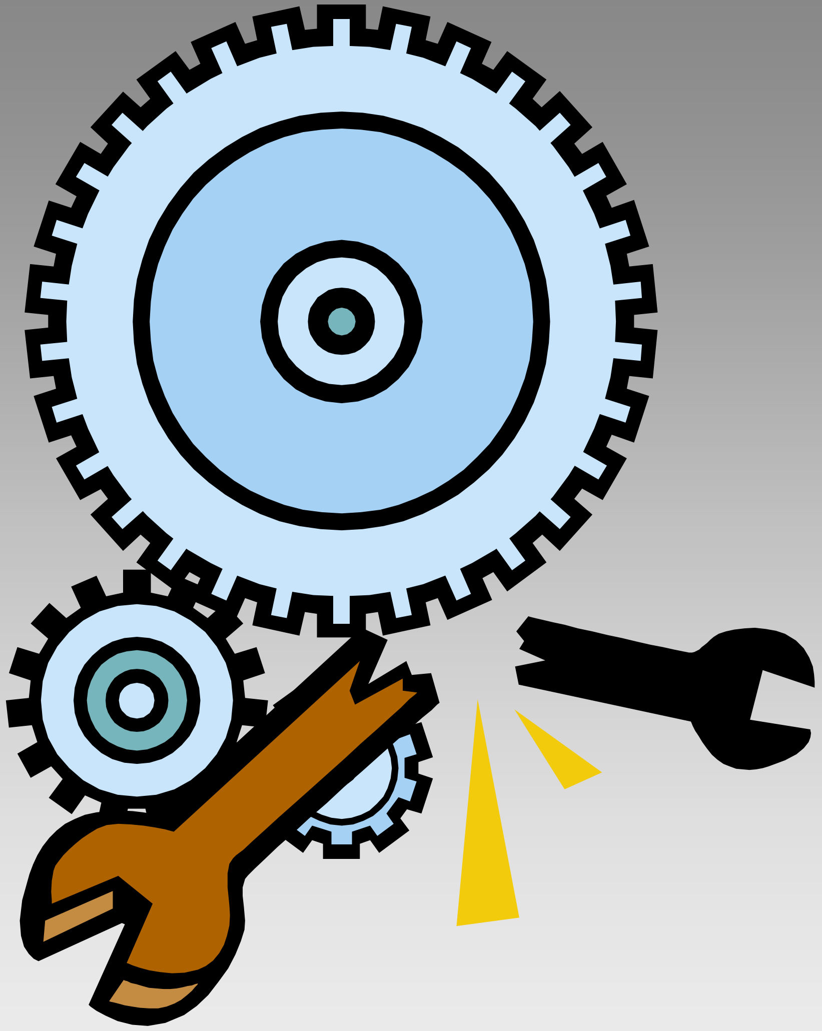
Commodity!

Goal

- X3J20 Compliant Smalltalk
- First Class Member of the DLR Family
- Easy interop. with .Net
- Decent Performance



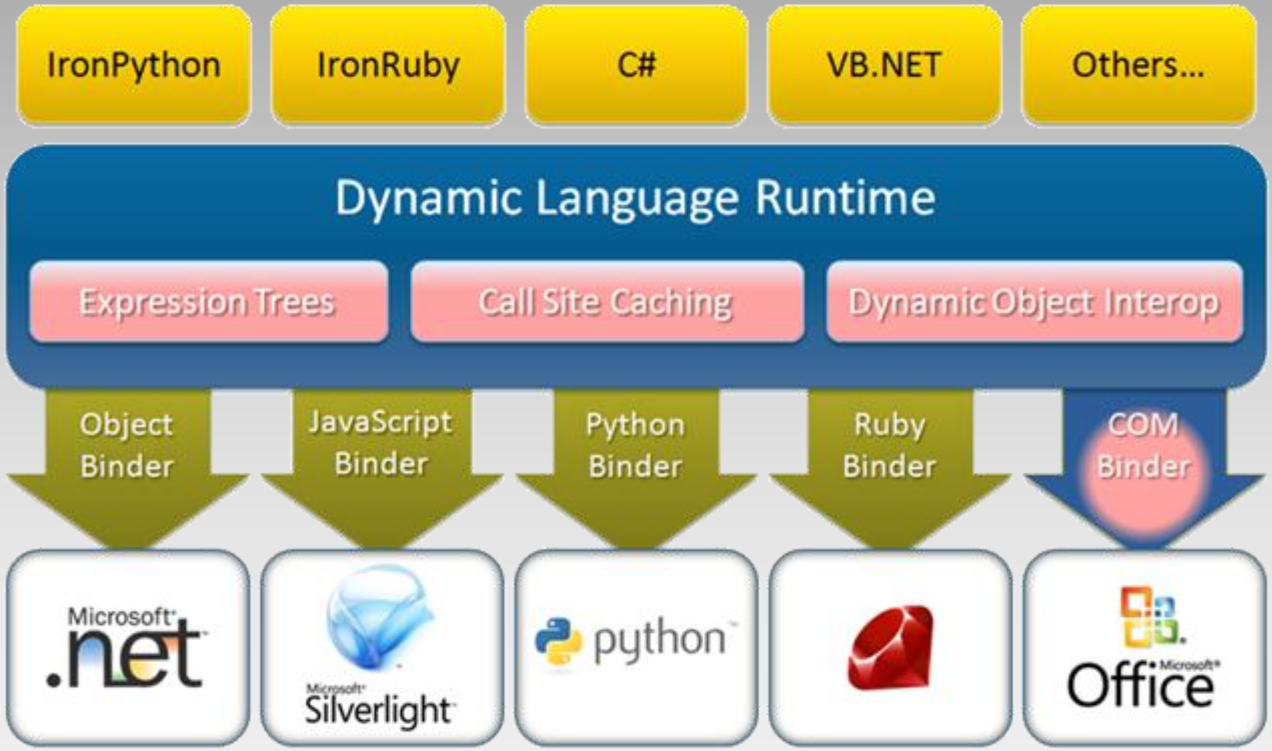
Personal Challenge



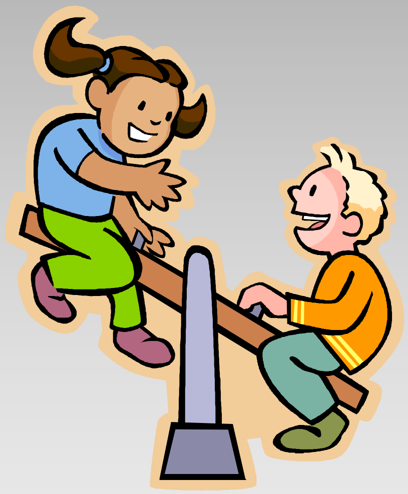
Microsoft®

.net™

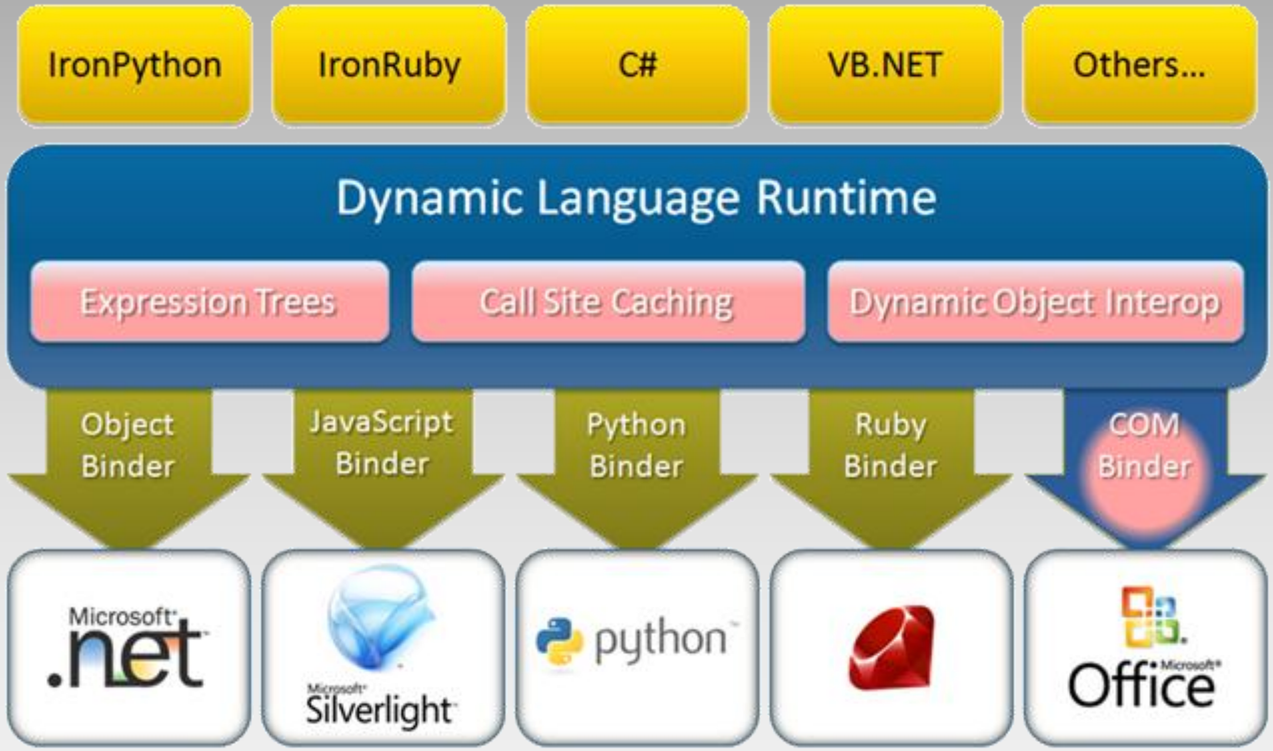
.NET DLR


IronPython
IronRuby
C#
VB.NET
Others…
Dynamic Language Runtime
Expression Trees
Call Site Caching
Dynamic Object Interop
Object Binder
JavaScript Binder
Python Binder
Ruby Binder
COM Binder
Microsoft .net
Microsoft Silverlight
python
Microsoft Office

.NET DLR

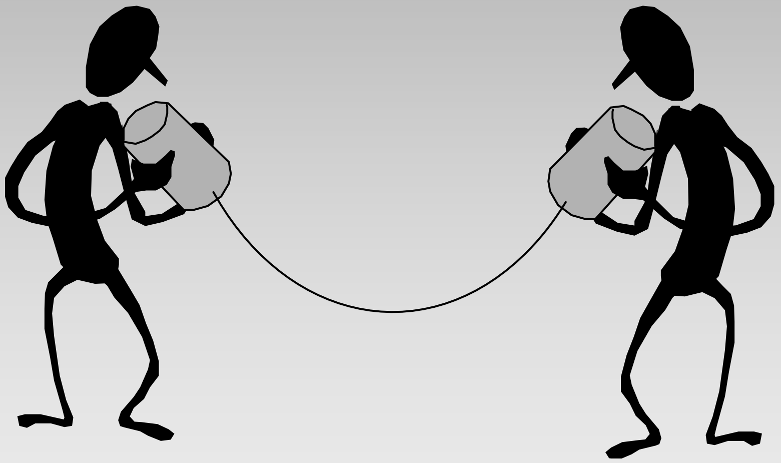


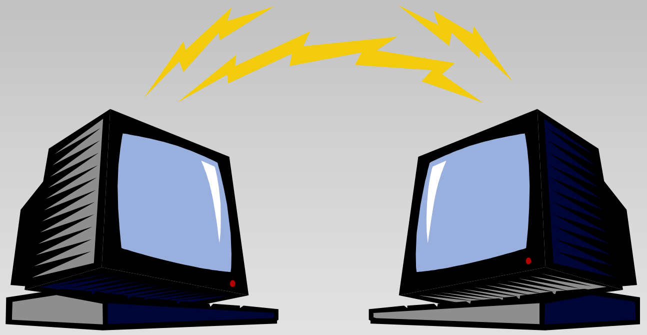
.NET DLR


IronPython
IronRuby
C#
VB.NET
Others…
Dynamic Language Runtime
Expression Trees
Call Site Caching
Dynamic Object Interop
Object Binder
JavaScript Binder
Python Binder
Ruby Binder
COM Binder
Microsoft .net
Microsoft Silverlight
python
Microsoft Office

Message Send

Message Send Communication





Message Send

```
Transcript show:
    'Hello, Edinburgh!'.
```

Message Send

```
Transcript show:  'Hello, Edinburgh!'.

TranscriptStream >> show: anObject

self
        nextPutAll: anObject asString;
        endEntry.
```

Method Lookup

```
Transcript show:  'Hello, Edinburgh!'.

TranscriptStream >> show: anObject

self
        nextPutAll: anObject asString;
        endEntry.
```

Call Site

```
Transcript show:  'Hello, Edinburgh!'.

call1 := Message receiver: Transcript
        selector: #show:
        arguments: #('Hello, Edinb… ').
call1 evaluate.
```

Call Site

```
Transcript show:  'Hello, Edinburgh!'.

call1 := CallSite receiver: Transcript
        selector: #show:
        arguments: #('Hello, Edinb... ').
call1 evaluate.
```

Call Site

```
Transcript show:  'Hello, Edinburgh!'.

call1 := CallSite selector: #show:.

call1
    evaluateWithReceiver: Transcript
    andArgs: #('Hello, Edinburgh').
```

Call Site Binders

```
Transcript show:  'Hello, Edinburgh!'.

call1 := CallSite onBinder:
        (Binder selector: #show:).

call1
      evaluateWithReceiver: Transcript
      andArgs: #('Hello, Edinburgh').
```

Call Site Binders

```
Binder >> bindFor: aReceiver
      arguments: anArray

^BindingRule for:
      (aReceiver compiledMethodAt:
            #show).
```

Dynamic Objects

Reflective Objects

Call Site Binders

```
Binder >> bindFor: aReceiver
      arguments: anArray

^BindingRule for:
      (aReceiver compiledMethodAt:
            #show).
```

Call Site Binders

```
Binder >> bindFor: aReceiver
       arguments: anArray

^aReceiver bindEvaluateFor: self
       arguments: anArray.
```

Call Site Binders

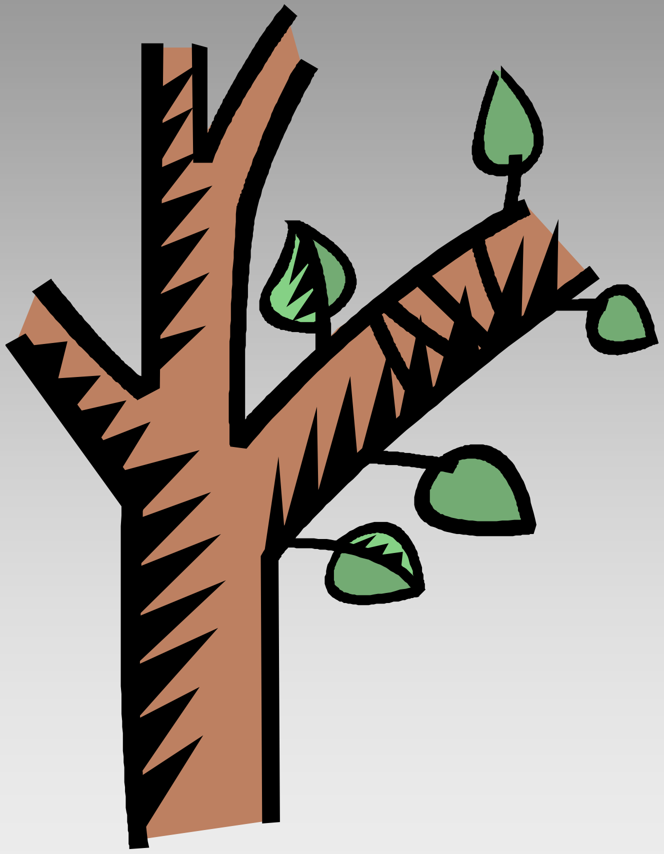
```
Object >> bindEvaluateFor: aBinder
        arguments: anArray

^BindingRule for:
        (self compiledMethodAt:
                aBinder selector).

        " or return whatever it wants! "
```

Call Site Binders

… receiver can’t help us, then …

```
fallbackBindInvokeFor: aReceiver
      arguments: anArray

^BindingRule for:
      … some compiled method …
```

Expression Trees



Expression Trees

Abstract Semantic Tree

Expression Trees

Abstract Semantic Tree

Abstract Syntax Tree

Expression Trees

Abstract Semantic Tree

Abstract Syntax Tree

Expression trees **model** code

Expression Trees

Abstract Semantic Tree

Abstract Syntax Tree

Expression trees **model** code

Expressions are used to represent the **implementation** of certain code or logic

Expression Trees

```
signString

    ^self < 0
        ifTrue: [ 'Negative' ]
        ifFalse: [ 'Positive' ].
```

Expression Trees

```
signString

^Expression
    condition: (Expression
        lowerThan: self value: 0)
    true: 'Negative'
    false: 'Positive'.
```

Expression Trees

```
signString

selfArg := Expression parameter: 'self'.

^Expression
        condition: (Expression
                lowerThan: selfArg
                value: (Expression constant: 0))
        true: (Expression constant: 'Negative')
        false: (Expression constant: 'Positive').
```

Expression Trees

```
TranscriptStream>>show: anObject

self
	nextPutAll: anObject.
```

Expression Trees

```
self
        nextPutAll: anObject.

selfArg := Expression parameter: 'self'.
arg1 := Expression parameter: 'anObject'.

^Expression
        dynamic: (Binder selector: #nextPutAll:)
        receiver: selfArg
        arguments: (Array with: arg1).
```

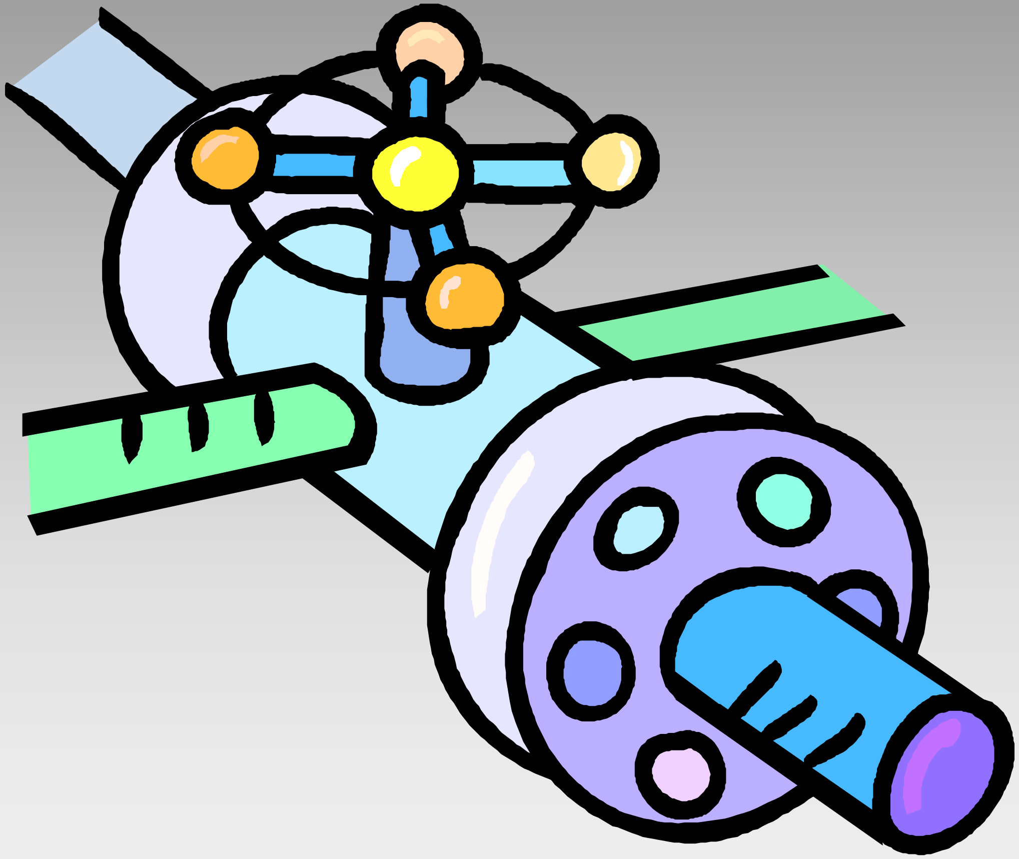
Expression Trees

```
Binder >> bindFor: aReceiver
      arguments: anArray

^BindingRule for:
      (aReceiver compiledMethodAt:
            #nextPutAll).
```

Expression Trees

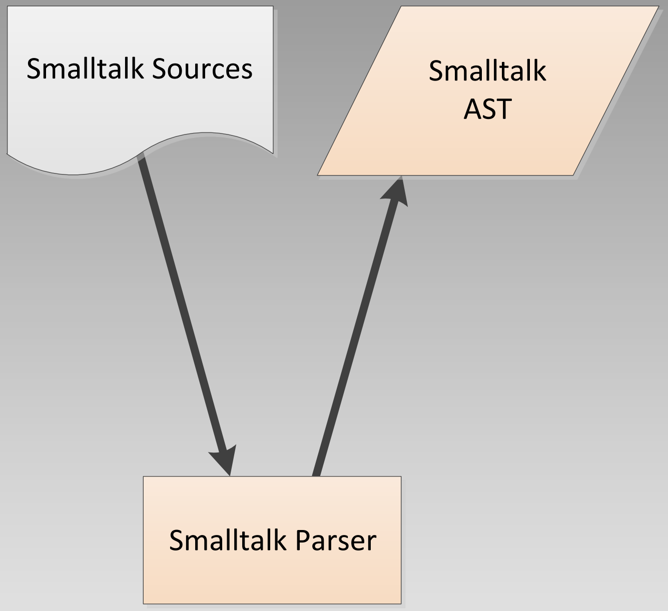
```
Binder >> bindFor: aReceiver
        arguments: anArray

^BindingRule expression: (Expression
        dynamic: (Binder selector: #nextPutAll:)
        receiver: aReceiver expression
        arguments: (Array
                with: anArray first expression).
```

Code Pipeline

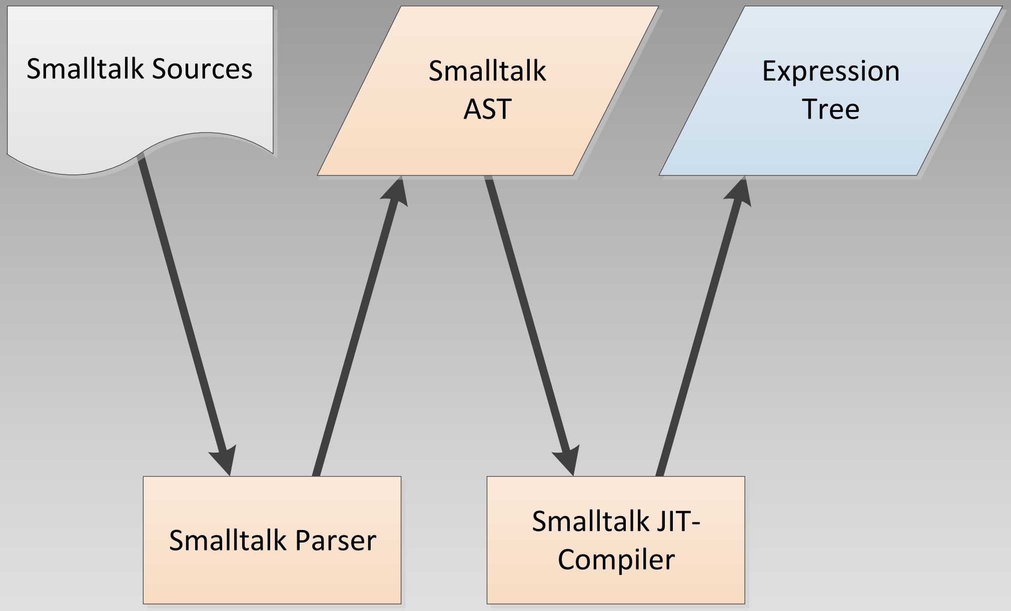


Code Pipeline

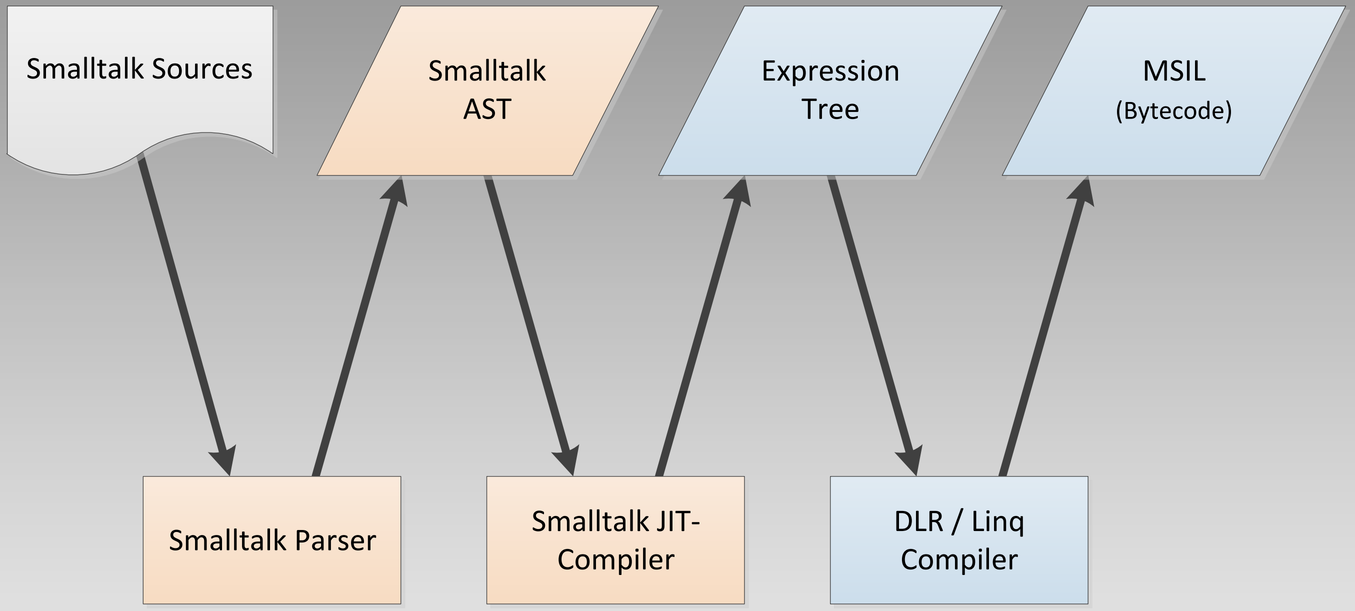
Code Pipeline



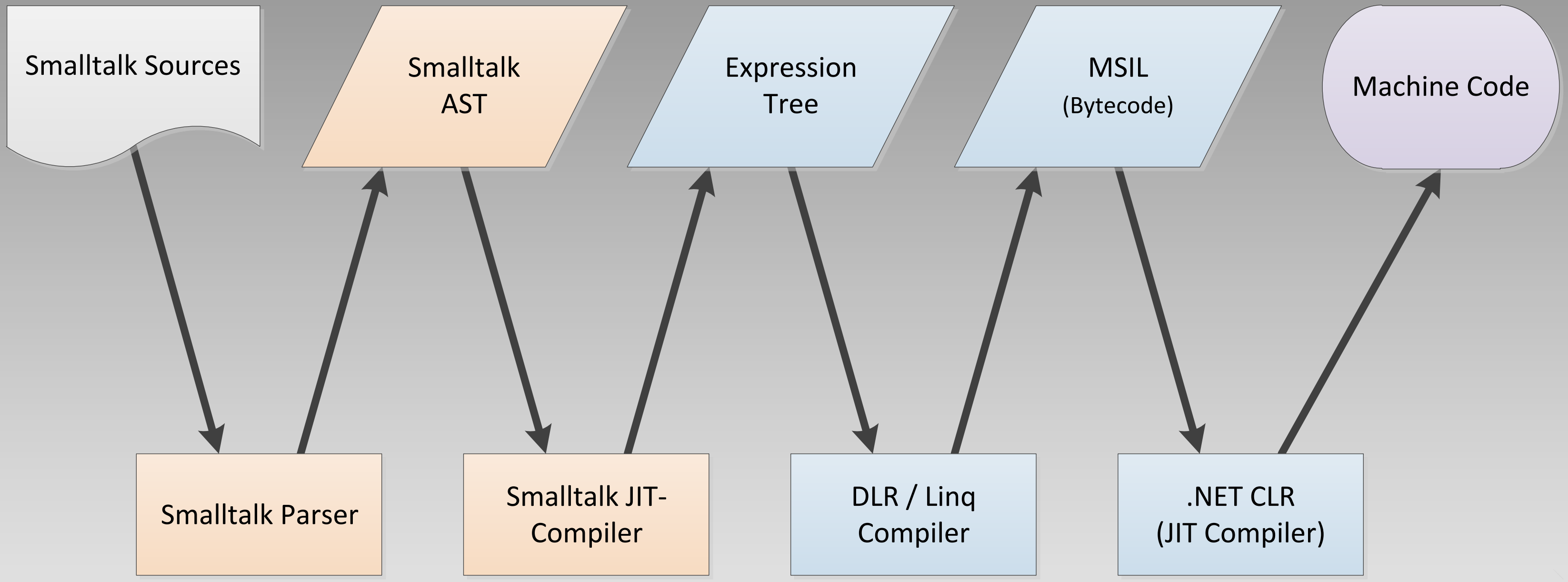
Code Pipeline



Code Pipeline



Code Pipeline



Polymorphic Inline Caching

Caching

```
Transcript show:  'Hello, Edinburgh!'.

call1 := CallSite onBinder:
      (Binder selector: #show:).

call1
    evaluateWithReceiver: Transcript
    andArgs: #('Hello, Edinburgh').
```

Caching

```
Binder >> bindFor: aReceiver
          arguments: anArray

method := aReceiver value compiledMethodAt: #show:.

" … magic … to generate expression from method … "

^BindingRule expression: (Expression
          dynamic: (Binder selector: #nextPutAll:)
          receiver: aReceiver expression
          arguments: (Array
                    with: anArray first expression).
```

Caching

```
Transcript show:  'Hello, Edinburgh!'.

call1 := CallSite onBinder:
      (Binder selector: #show:).


call1
    evaluateWithReceiver: Transcript
    andArgs: #('Hello, Edinburgh').
```

Caching

```
Binder >> bindFor: aReceiver
          arguments: anArray

method := aReceiver value compiledMethodAt: #show:.

" … magic … to generate expression from method … "

^BindingRule expression: " … expression … "
          restriction: [ :obj |
                    obj class = TranscriptStream ].
```

Caching

```
Transcript show:  'Hello, Edinburgh!'.

call1 := CallSite onBinder:
      (Binder selector: #show:).


call1
    evaluateWithReceiver: Transcript
    andArgs: #('Hello, Edinburgh').
```

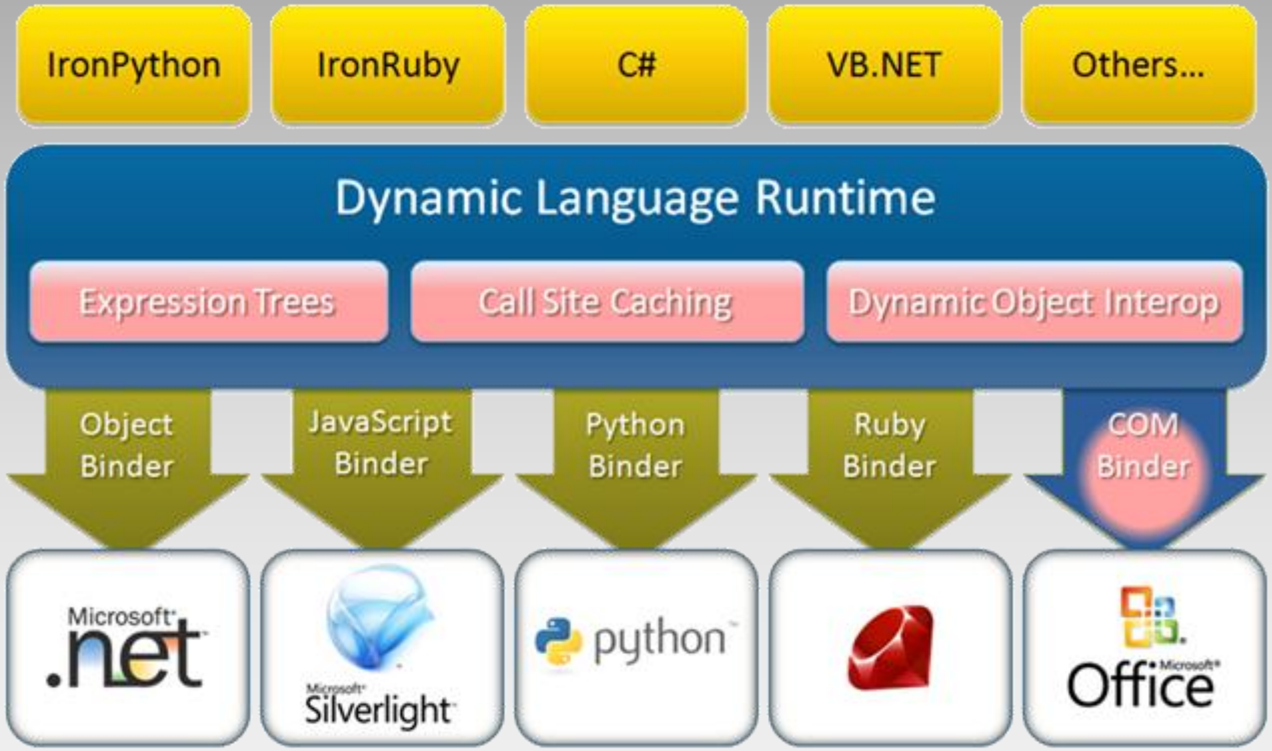
Restrictions

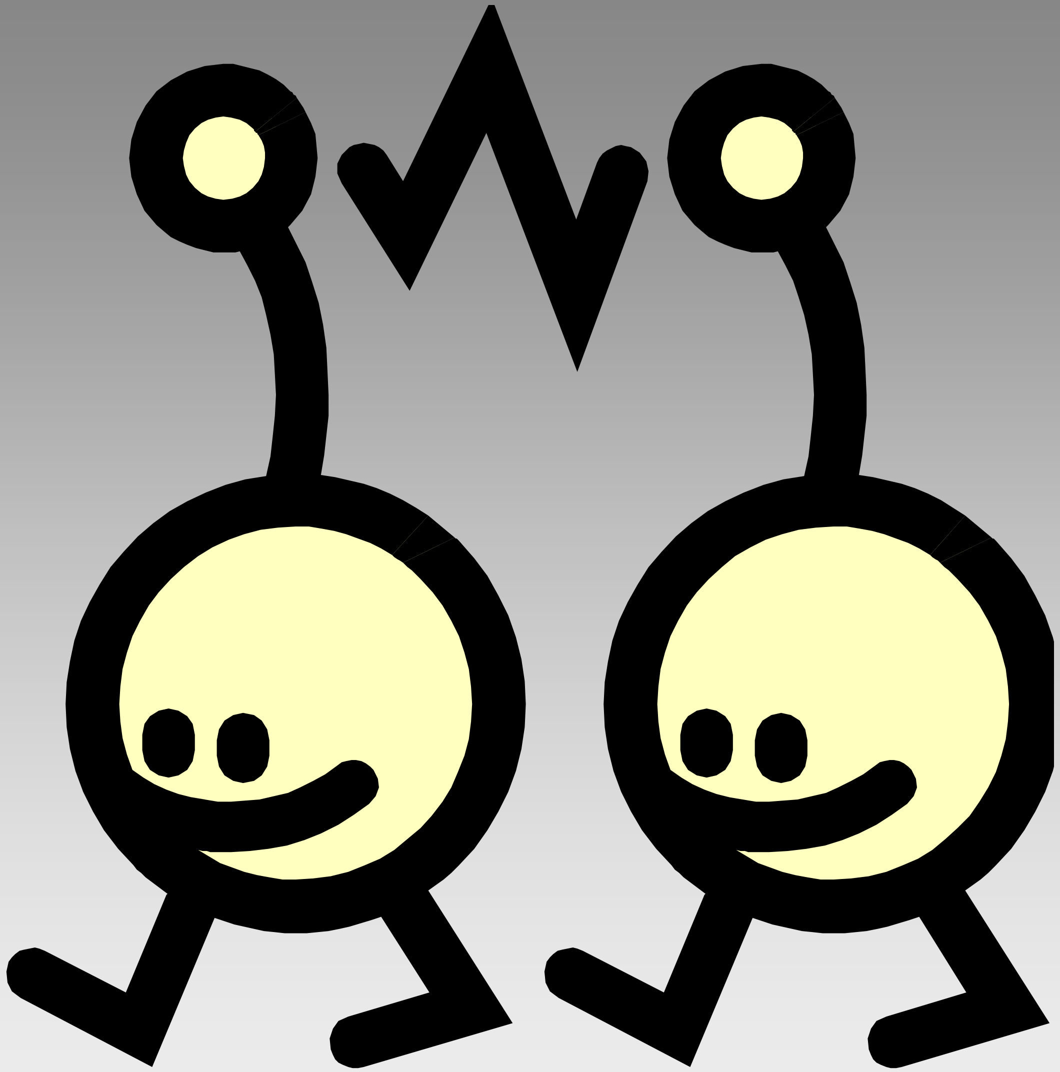
- Restrictions can be more complex
  - For example, instance specific or object version specific
- Restrictions determine the polymorphism of objects
- Restrictions are the key to polymorphic inline caching!

The Cache

- 3 Levels of Caching:
- Level 0 : Rule for last call
- Level 1 : Last 10 rules
- Level 2 : 100 rules, shares across similar call sites

.NET DLR

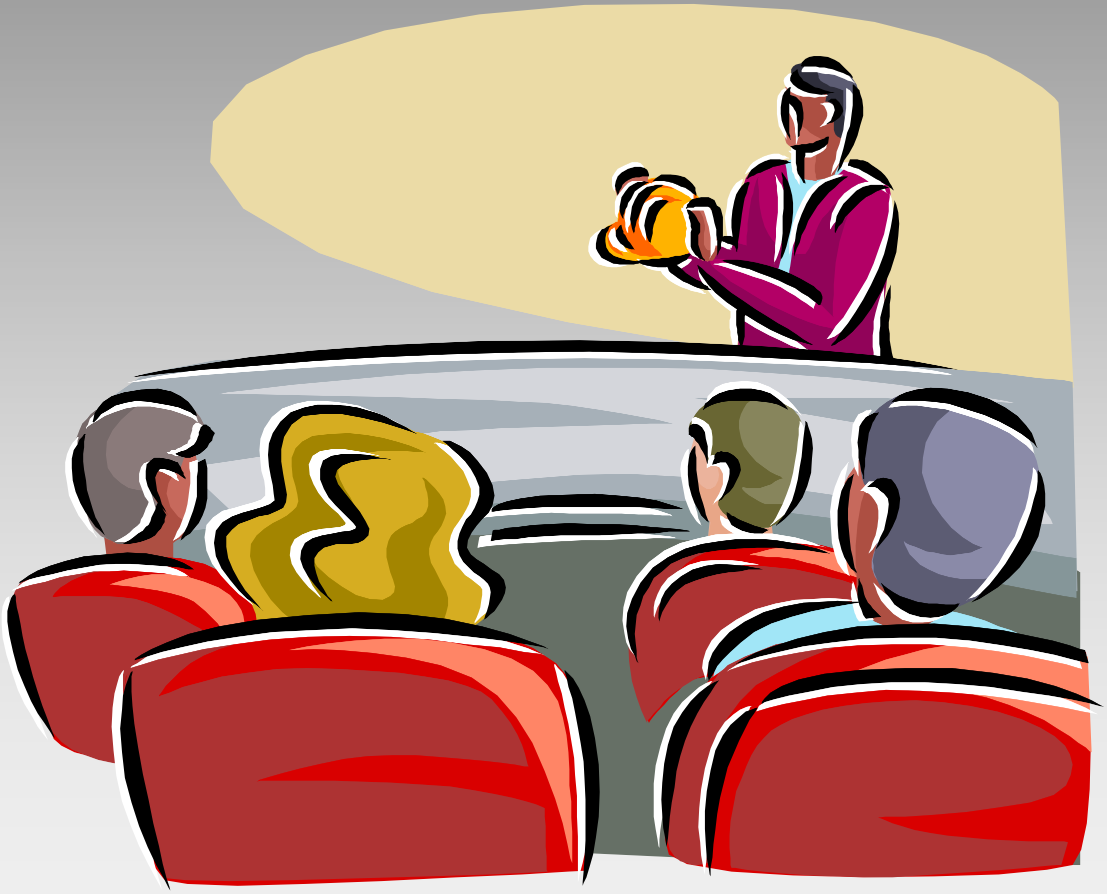

IronPython
IronRuby
C#
VB.NET
Others…
Dynamic Language Runtime
Expression Trees
Call Site Caching
Dynamic Object Interop
Object Binder
JavaScript Binder
Python Binder
Ruby Binder
COM Binder
Microsoft .net
Microsoft Silverlight
python
Microsoft Office



Common Vocabulary

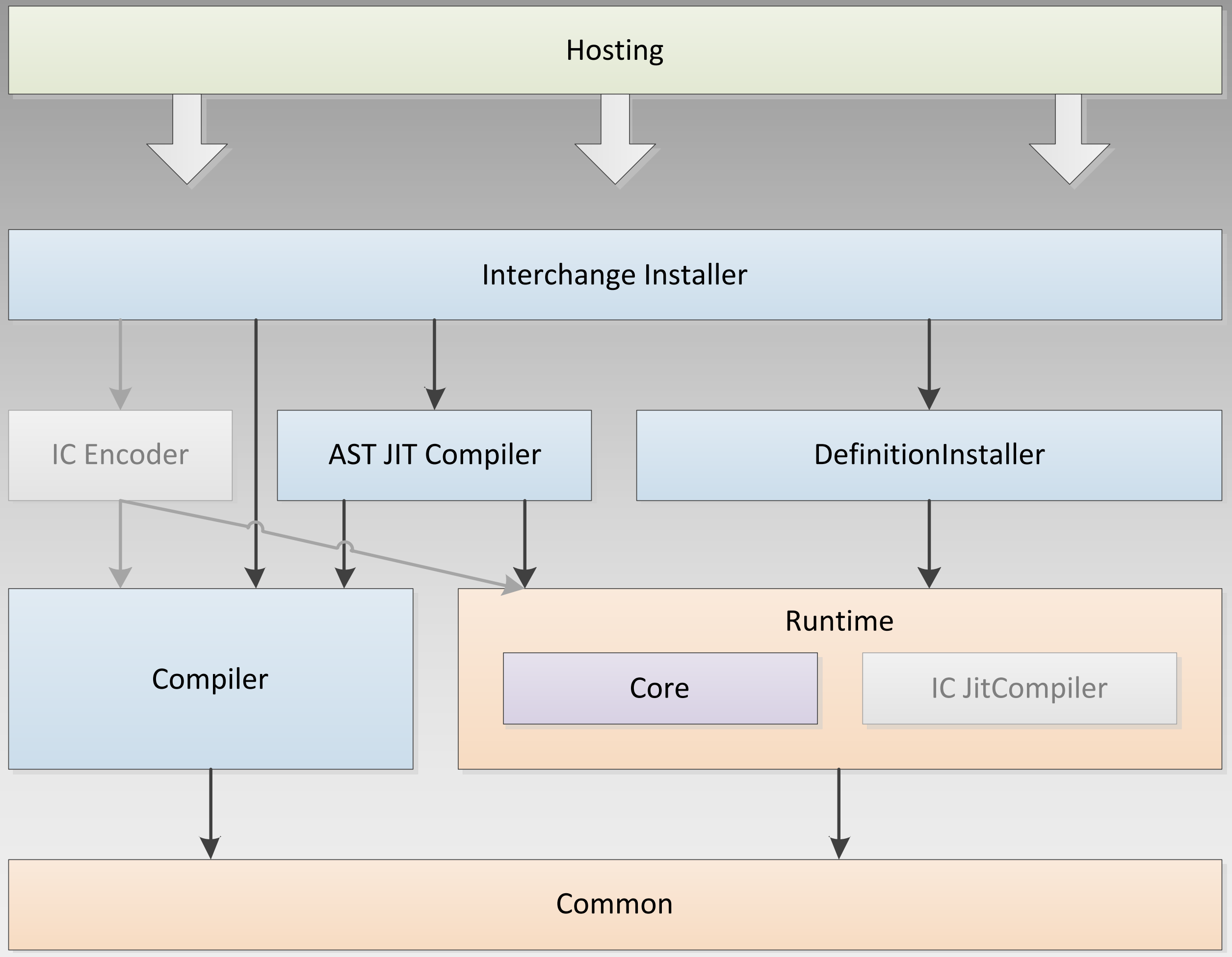
- None Defined
- De-Facto, the .Net class library
- We aim at reusing the common classes
  - Example: System.Char => Character
- Not everything maps easily, but most do
  - Example: System.String ~= String

Demo

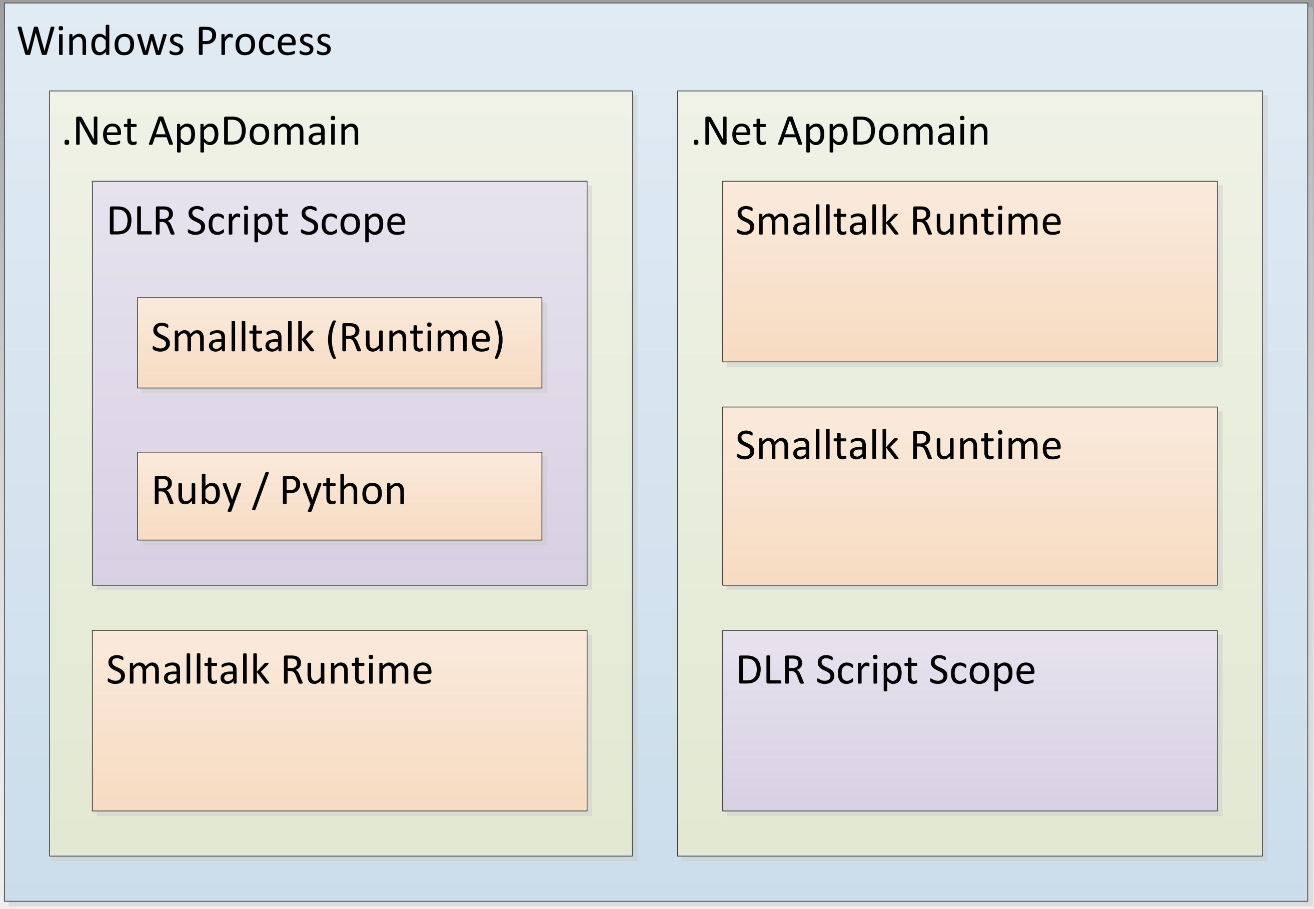


Summary

Components


Hosting
Interchange Installer
IC Encoder
AST JIT Compiler
DefinitionInstaller
Compiler
Runtime
Core
IC JitCompiler
Common

Hosting


Windows Process
.Net AppDomain
DLR Script Scope
Smalltalk (Runtime)
Ruby / Python
Smalltalk Runtime
.Net AppDomain
Smalltalk Runtime
Smalltalk Runtime
DLR Script Scope

Specifics

Symbols

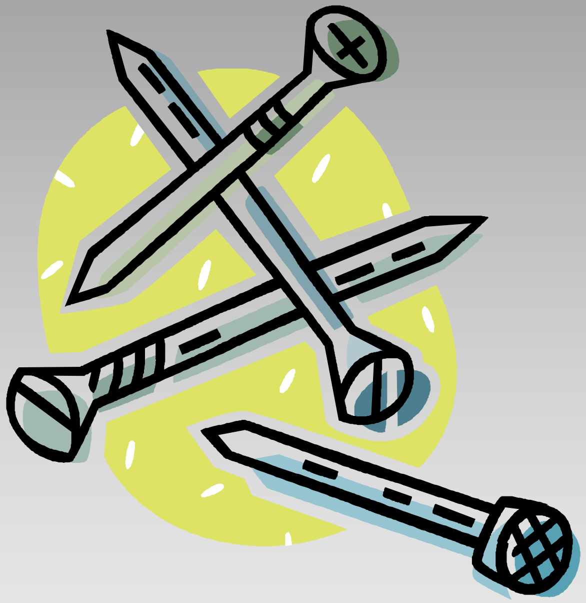
Strings

Smalltalk Process

#sender

#respondsTo:

Events



Object Space

No Object Space Reflection

No #allInstances

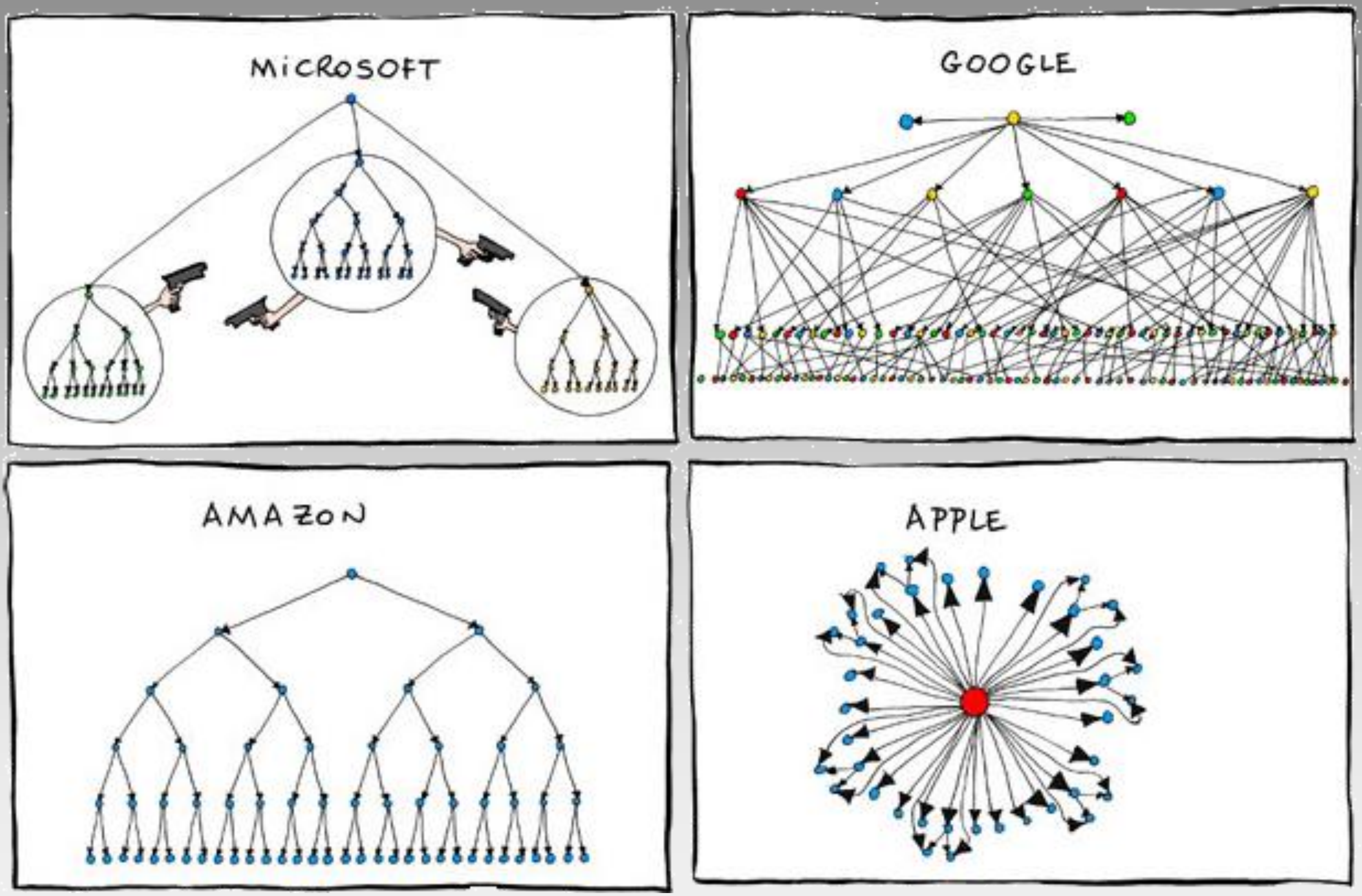
No #allReferences

No #become:

But #behaveLike: is possible!

Debugger inspection is possible.

Major Tech Companies



Good News

Unicode
Multi Threaded
Large Class Library
Web-Hostable
MIT License

Near Future

Generics

Class Library Importer

(or auto-integration)

Finish Version 1.0

Write Tests (…)

Future

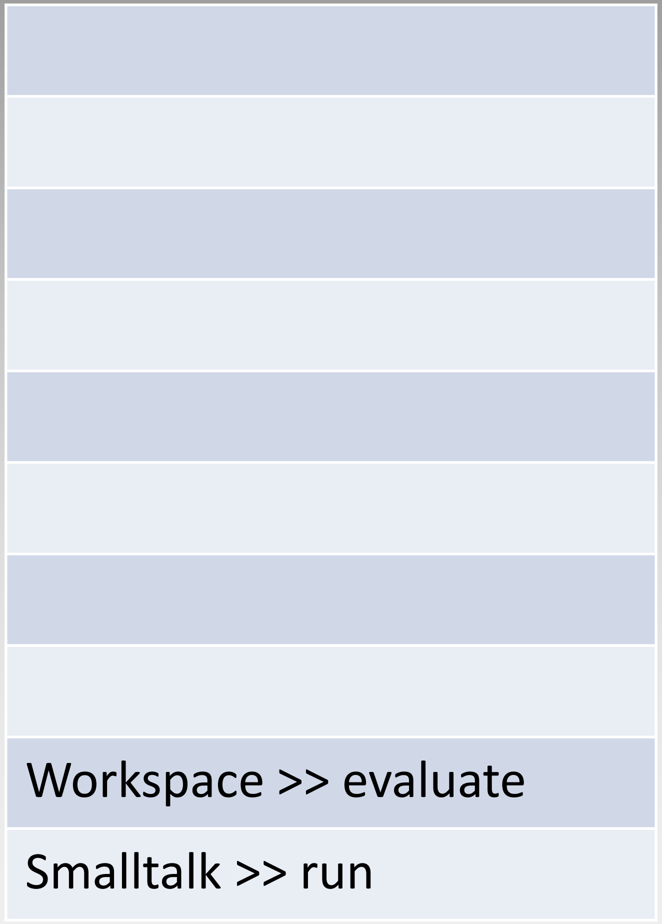
GUI Integration (WinForms + WPF)
SubSystems
(Interactive) Debugger
Visual Studio Integration
Mixins
SubSystems / Namespaces
Instance Specific Behavior

Questions

http://ironsmalltalk.codeplex.com/

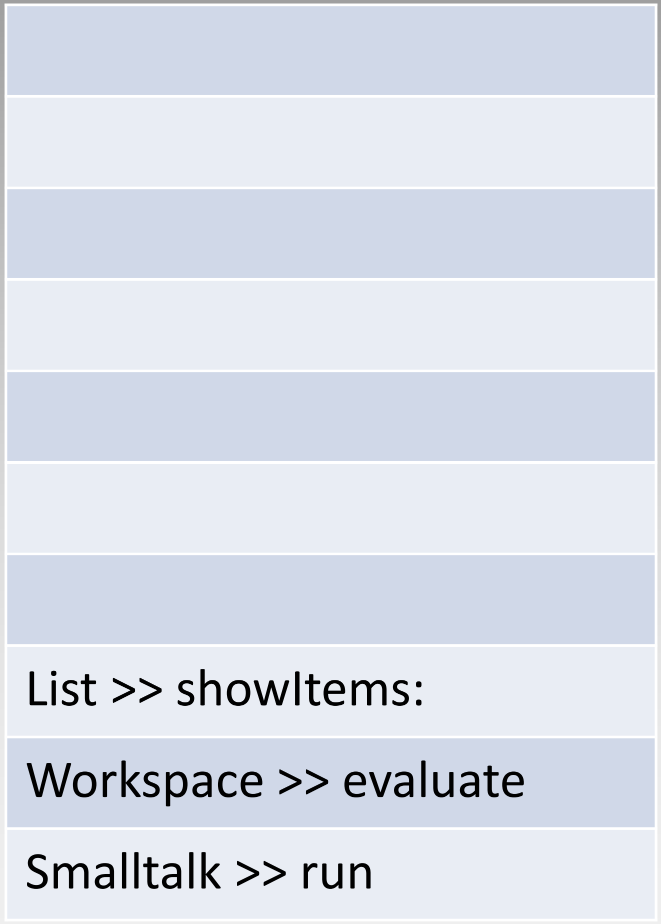
mailto:todor@scientia.dk

Callbacks



```
List showItems:
     #( 'TT' 'ST' ).
```

Callbacks

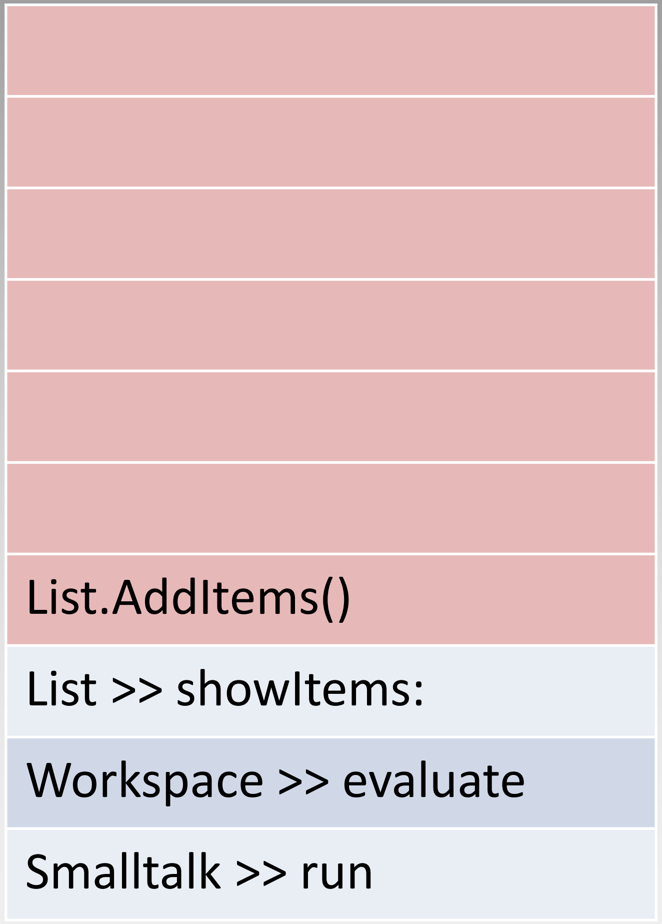


List >> showItems: anArray

" AddItems(anArray, self) "

<Windows API>

Callbacks



```
List.AddItems(string[] keys,
        dynamic stList)
{
        for(i=0; i<…;i++)
                this.AddItem(
                        keys[i],
                        stList);
}
```

Callbacks

List.AddItem()
List.AddItems()
List >> showItems:
Workspace >> evaluate
Smalltalk >> run

```
List.AddItem(string[] keys,
        dynamic stList)
{
        string text = stList.
                .GetItemText(key);

        " ...somehow add it... "
}
```

Callbacks

List >> getItemText:
List.AddItem()
List.AddItems()
List >> showItems:
Workspace >> evaluate
Smalltalk >> run

```
List >> getItemText:
        aString

        ^self itemTexts
              at: aString.
```

Callbacks

List >> getItemText:
List.AddItem()
List.AddItems()
List >> showItems:
Workspace >> evaluate
Smalltalk >> run

.Net 4.0 has the **dynamic** keyword used for invoking method on dynamic objects.

IronSmalltalk objects can transparently be used by any DLR aware language.

Callbacks

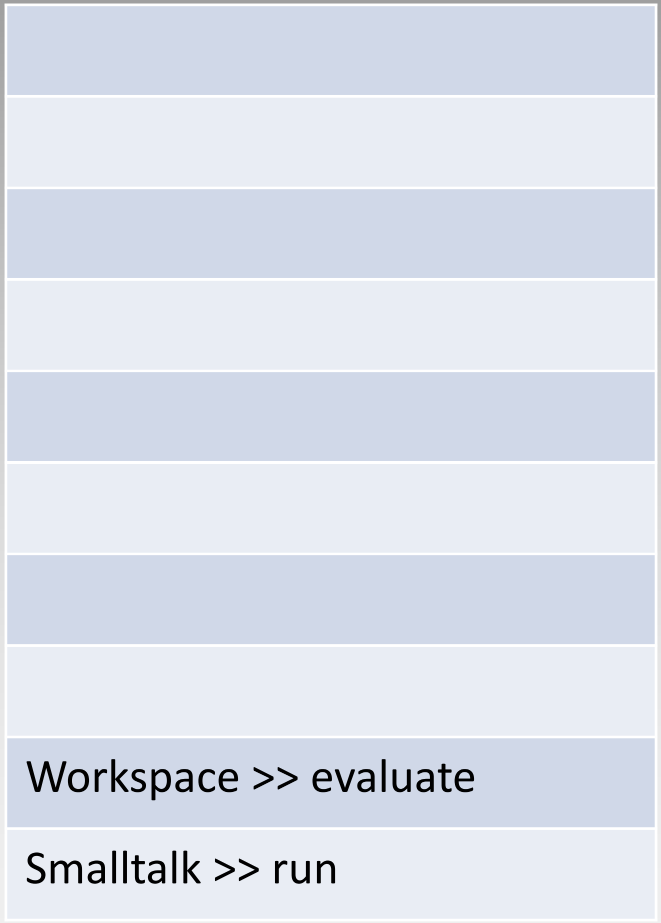
List >> getItemText:
List.AddItem()
List.AddItems()
List >> showItems:
Workspace >> evaluate
Smalltalk >> run

Native Name: GetItemText

List >> getItemText:
    aString

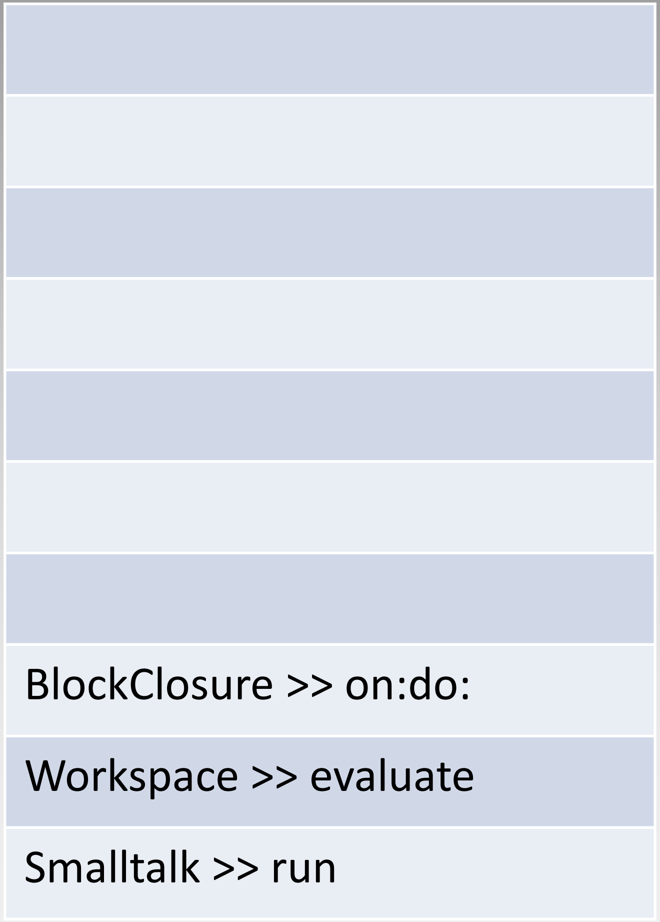
The native name is the method name (selector) that is exposed to the other DLR languages.

Exceptions



```
[ Transcript show:
        'Hello, Edinburgh!' ]
        on: Error do: [ … ].
```

Exceptions



```
BlockClosure >>
        on: exception
        do: handlerAction

        " Some magic … "

        ^self value.
```

Exceptions

BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

BlockClosure >> value

< Primitive >

Exceptions

TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
    show: anObject

self
    nextPutAll:
        anObject asString;
        endEntry.
```

Exceptions

String >> asString
TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
String >> asString

    ^self.
```

Exceptions

TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
    show: anObject

self
    nextPutAll:
        anObject asString;
        endEntry.
```

Exceptions

TranscriptStream >>nextPutAll:
TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
    nextPutAll: aString

" … some code here … "
```

Exceptions

TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
    show: anObject

self
    nextPutAll:
        anObject asString;
        endEntry.
```

Exceptions

TranscriptStream >> endEntry
TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
      endEntry

" … some code here … "
```

Exceptions

TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
    show: anObject

self
    nextPutAll:
        anObject asString;
        endEntry.
```

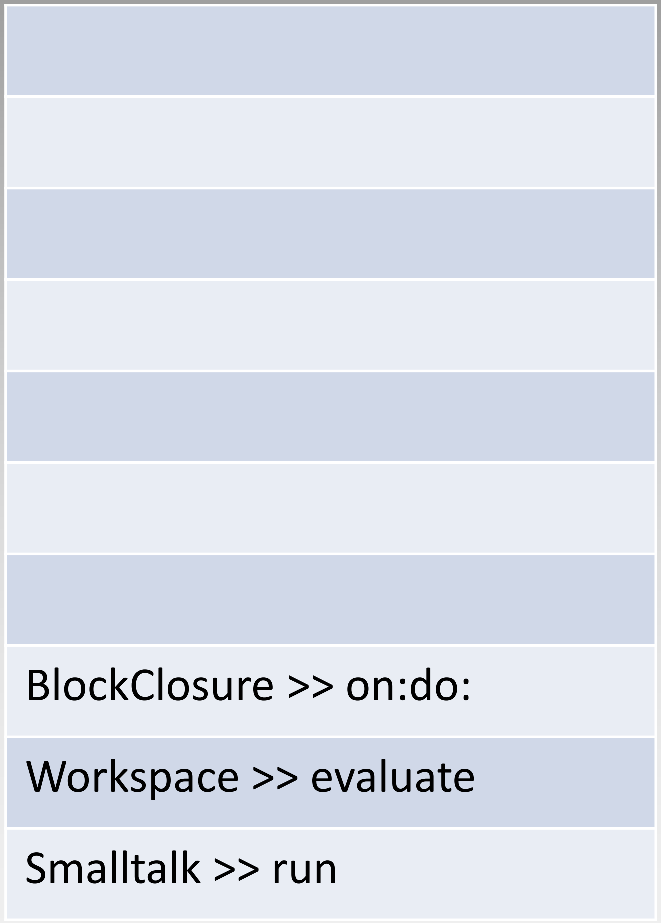
Exceptions

BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

BlockClosure >> value

< Primitive >

Exceptions



```
BlockClosure >>
          on: exception
          do: handlerAction

          " Some magic … "

          ^self value.
```

Exceptions

Workspace >> evaluate
Smalltalk >> run

```
Transcript show:
        'Hello, Edinburgh!'.
```

Exceptions

TranscriptStream >>nextPutAll:
TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
      nextPutAll: aString

" Transcript Window cannot
hold more text! "
```

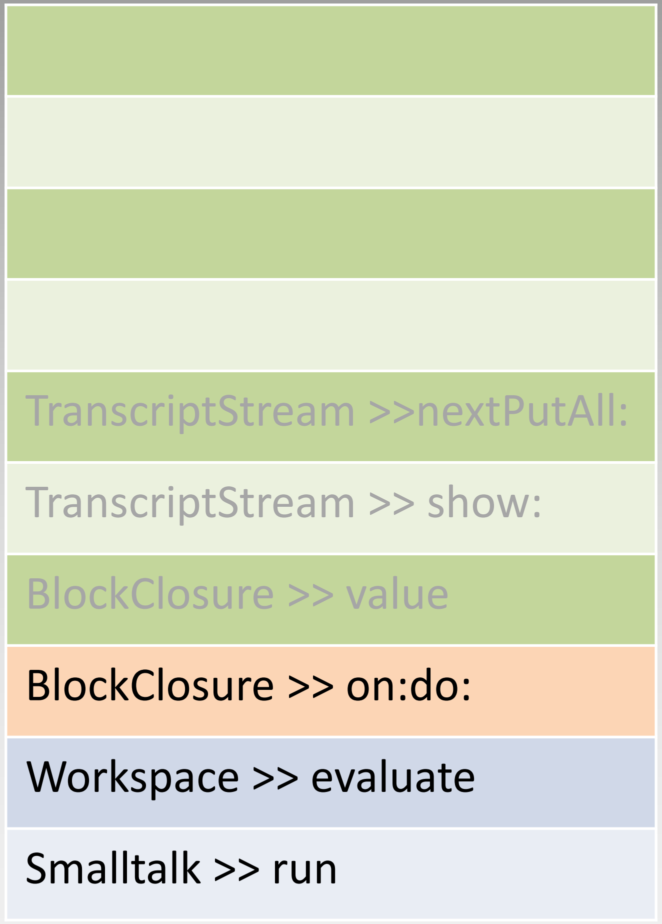
Exceptions

TranscriptStream >>nextPutAll:
TranscriptStream >> show:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
TranscriptStream >>
      nextPutAll: aString

" Transcript Window cannot
hold more text! "
```

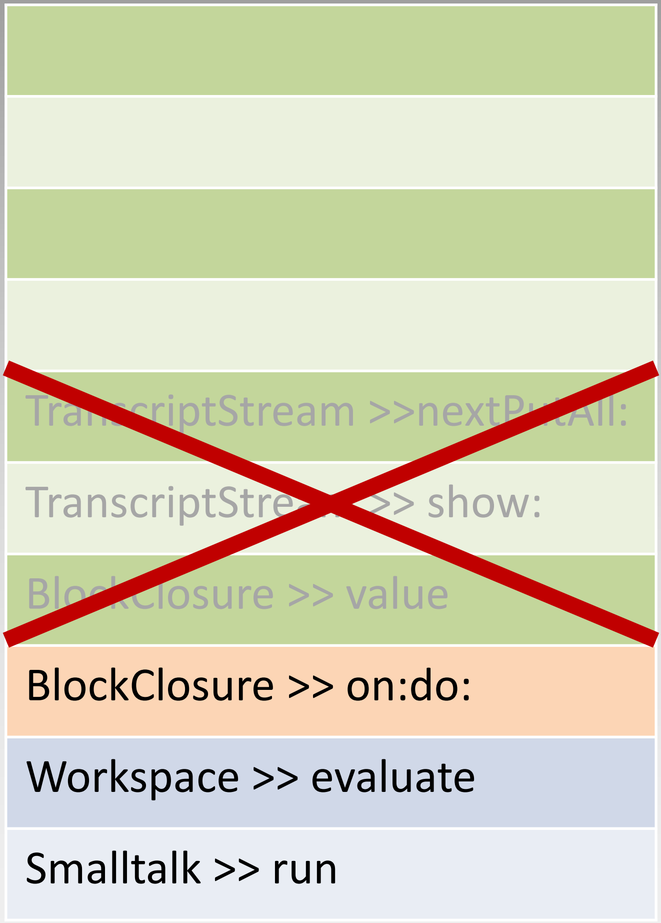
FAIL !

Exceptions



.Net Exceptions unwind the stack **when thrown**!

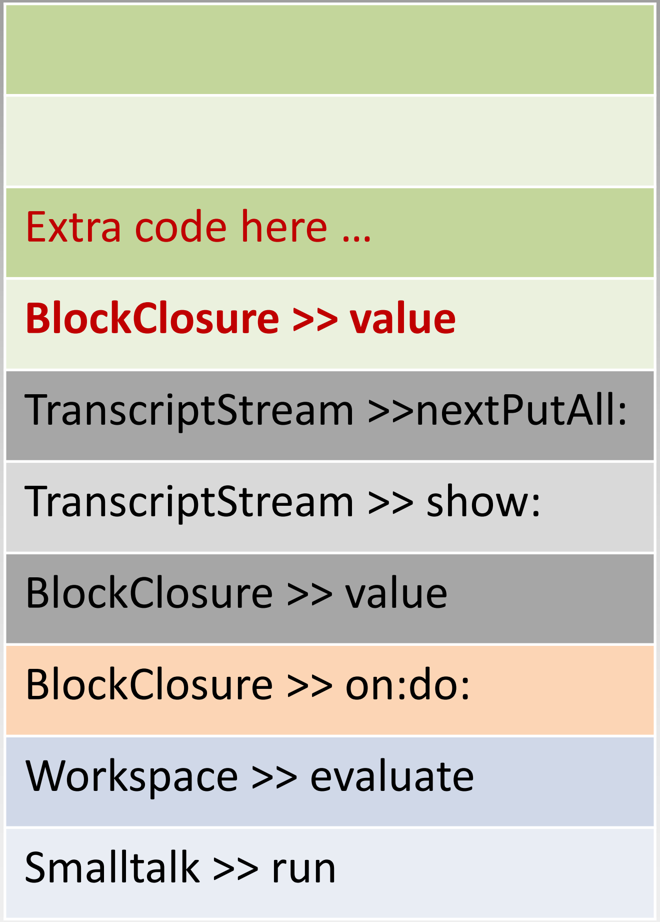
Exceptions



Smalltalk expects **resumable exceptions**!

Stack **is gone**!

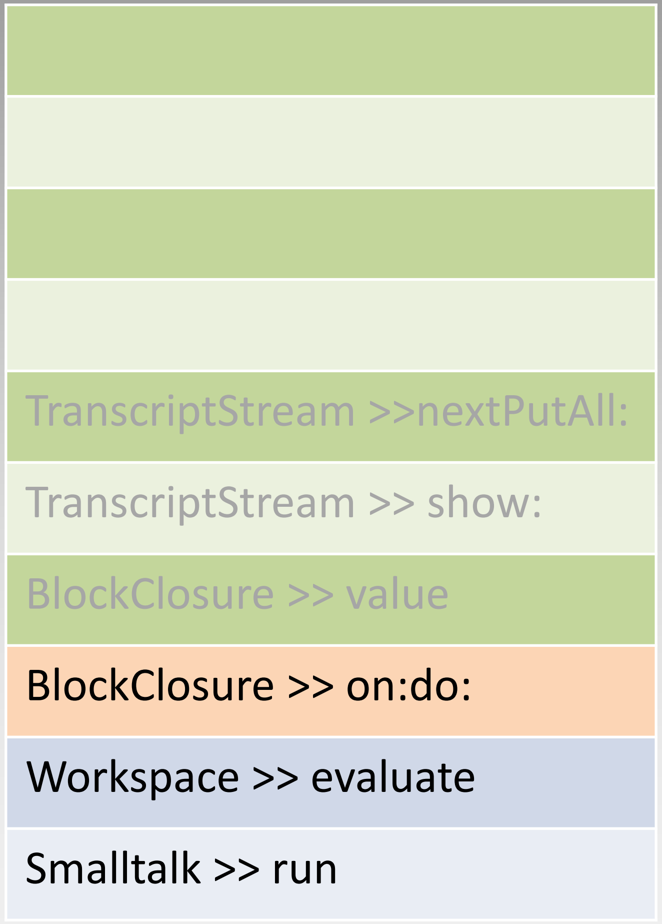
Exceptions



Must **keep** stack!

Code executes as usually.

Exceptions



Unwinding is executes as usually.

Exceptions

List >> getItemText:
List.AddItem()
List.AddItems()
List >> showItems:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
List >> getItemText:
        aString

        ^self itemTexts
               at: aString.
```

Exceptions

List >> getItemText:
List.AddItem()
List.AddItems()
List >> showItems:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
List >> getItemText:
         aString

         ^self itemTexts
                  at: aString.
```

KEY IS MISSING

Exceptions

List >> getItemText:
List.AddItem()
List.AddItems()
List >> showItems:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
List >> getItemText:
        aString

        ^self itemTexts
              at: aString.
```

KEY IS MISSING

Exceptions

List >> getItemText:
List.AddItem()
List.AddItems()
List >> showItems:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

```
List >> getItemText:
        aString

        ^self itemTexts
              at: aString.
```

KEY IS MISSING

Exceptions

List >> getItemText:
List.AddItem()
try { … } catch { … }
List.AddItems()
List >> showItems:
BlockClosure >> value
BlockClosure >> on:do:
Workspace >> evaluate
Smalltalk >> run

Exceptions **must** integrate with the rest of the .Net!

Exceptions

Non-Resumable exceptions can be mapped to native .Net exceptions

Exceptions

Non-Resumable exceptions can be mapped to native .Net exceptions

Resumable exceptions cannot be mapped to native .Net exceptions and will need a IronSmalltalk special implementation.