

## 10. A bit of Smalltalk



# Roadmap

- > The origins of Smalltalk
- > What is Smalltalk?
- > Syntax in a nutshell
- > Seaside — web development with Smalltalk

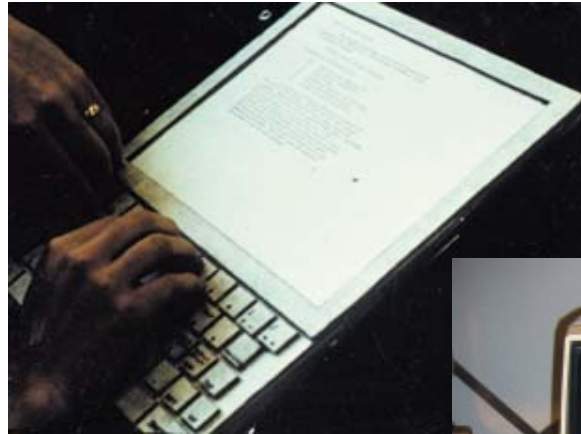


# Roadmap

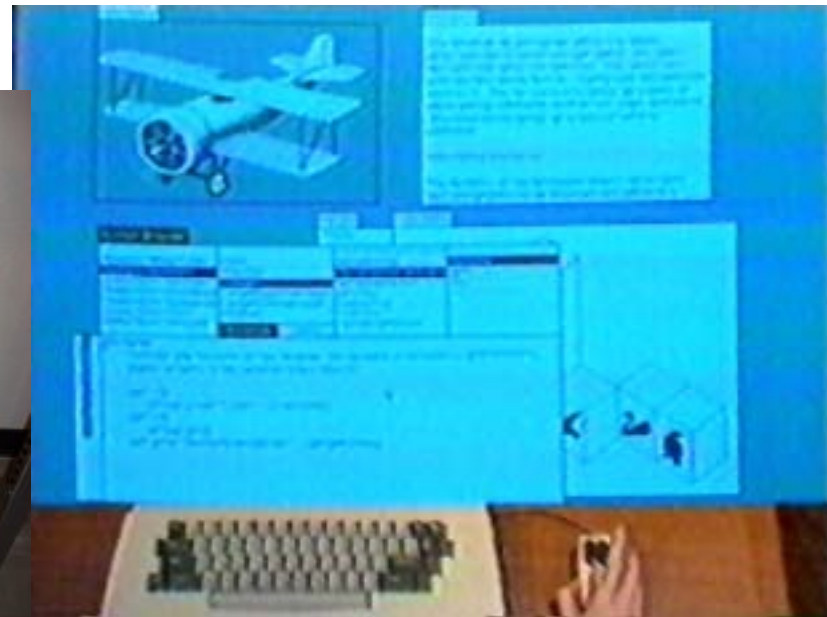
- > **The origins of Smalltalk**
- > What is Smalltalk?
- > Syntax in a nutshell
- > Seaside — web development with Smalltalk



# The origins of Smalltalk



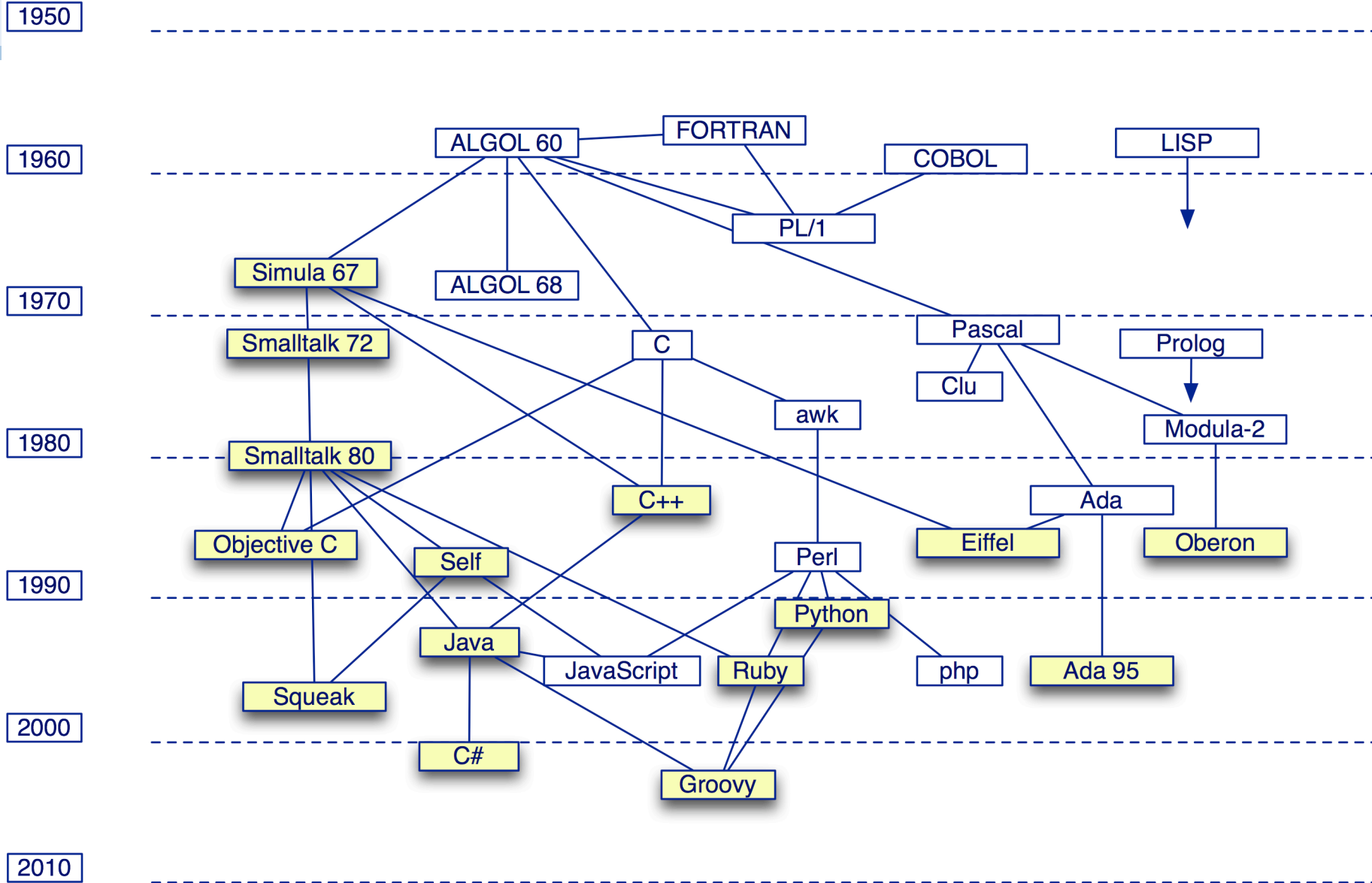
Alan Kay's Dynabook project (1968)



Alto — Xerox PARC (1973)

[gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html](http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html)

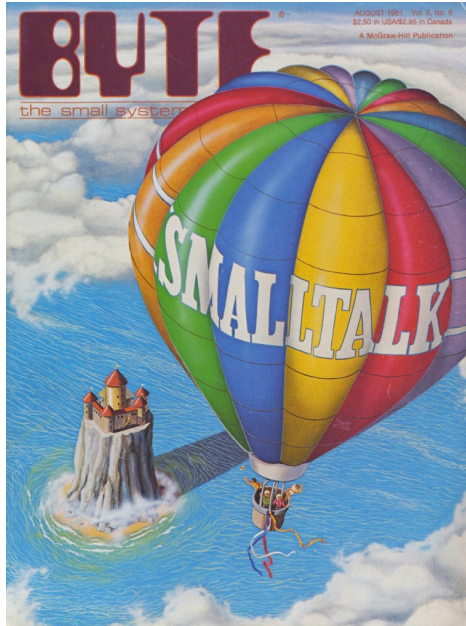
# Object-oriented language genealogy



# Smalltalk vs. C++ vs. Java

	<b><i>Smalltalk</i></b>	<b><i>C++</i></b>	<b><i>Java</i></b>
<i>Object model</i>	Pure	Hybrid	Hybrid
<i>Garbage collection</i>	Automatic	Manual	Automatic
<i>Inheritance</i>	Single	Multiple	Single
<i>Types</i>	Dynamic	Static	Static
<i>Reflection</i>	Fully reflective	Introspection	Introspection
<i>Concurrency</i>	Semaphores	Some libraries	Monitors
<i>Modules</i>	Categories, namespaces	Namespaces	Packages

# Smalltalk-80 and Squeak



- Everything is an object
- Everything is there, all the time
- First windowing system with mouse
- First graphical IDE

Squeak: a modern, portable, fast, open-source Smalltalk



# Squeak resources



[www.squeak.org](http://www.squeak.org)

*Downloads and links*

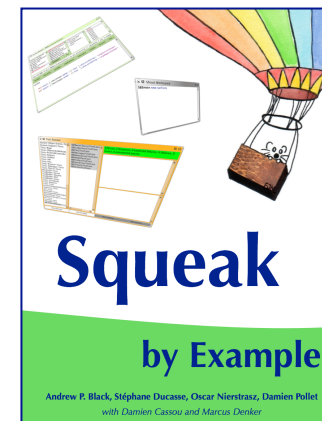


[www.seaside.st](http://www.seaside.st)

*One-click image*

[SqueakByExample.org](http://SqueakByExample.org)

*Free download — Print-on-demand*





# Don't panic!

*New Smalltalkers often think they need to understand all the details of a thing before they can use it.*

**Try to answer the question**

***“How does this work?”***

**with**

***“I don't care”.***

*—Alan Knight. Smalltalk Guru*

# Roadmap

- > The origins of Smalltalk
- > **What is Smalltalk?**
- > Syntax in a nutshell
- > Seaside — web development with Smalltalk



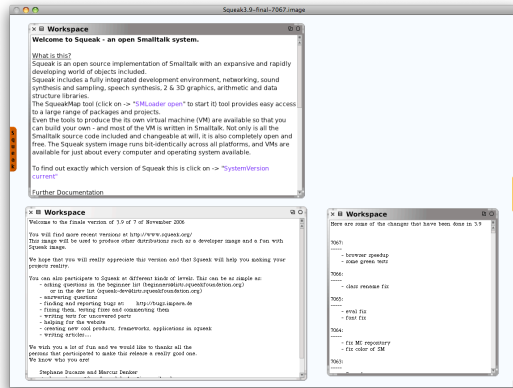
# ***Two rules to remember***

**Everything is an object**

**Everything happens by  
sending messages**

# What is Smalltalk?

Image

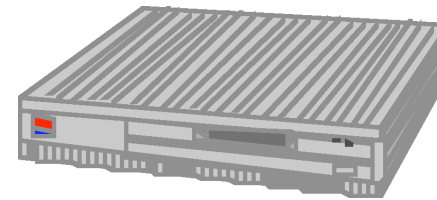


+

Changes



Virtual machine

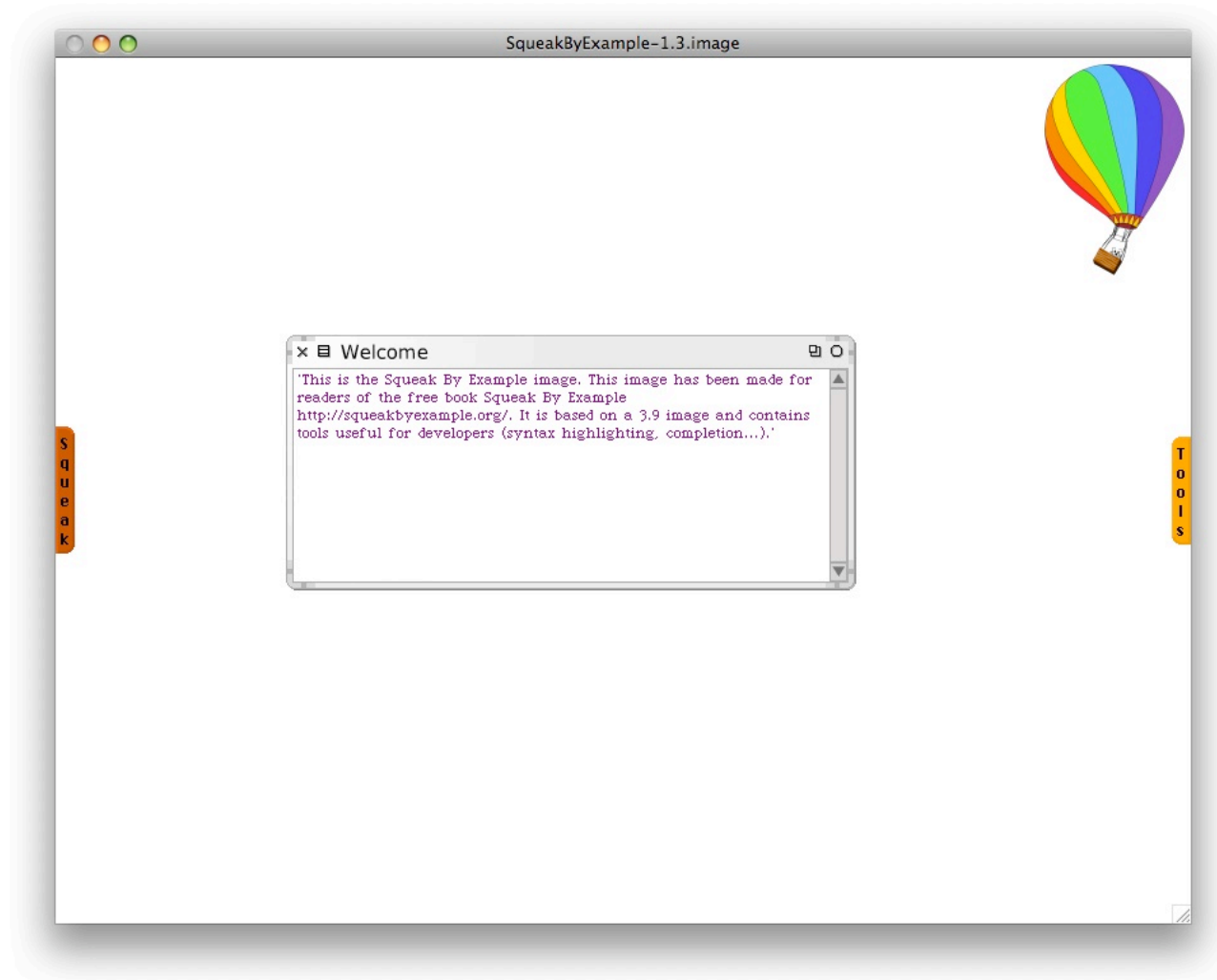


+

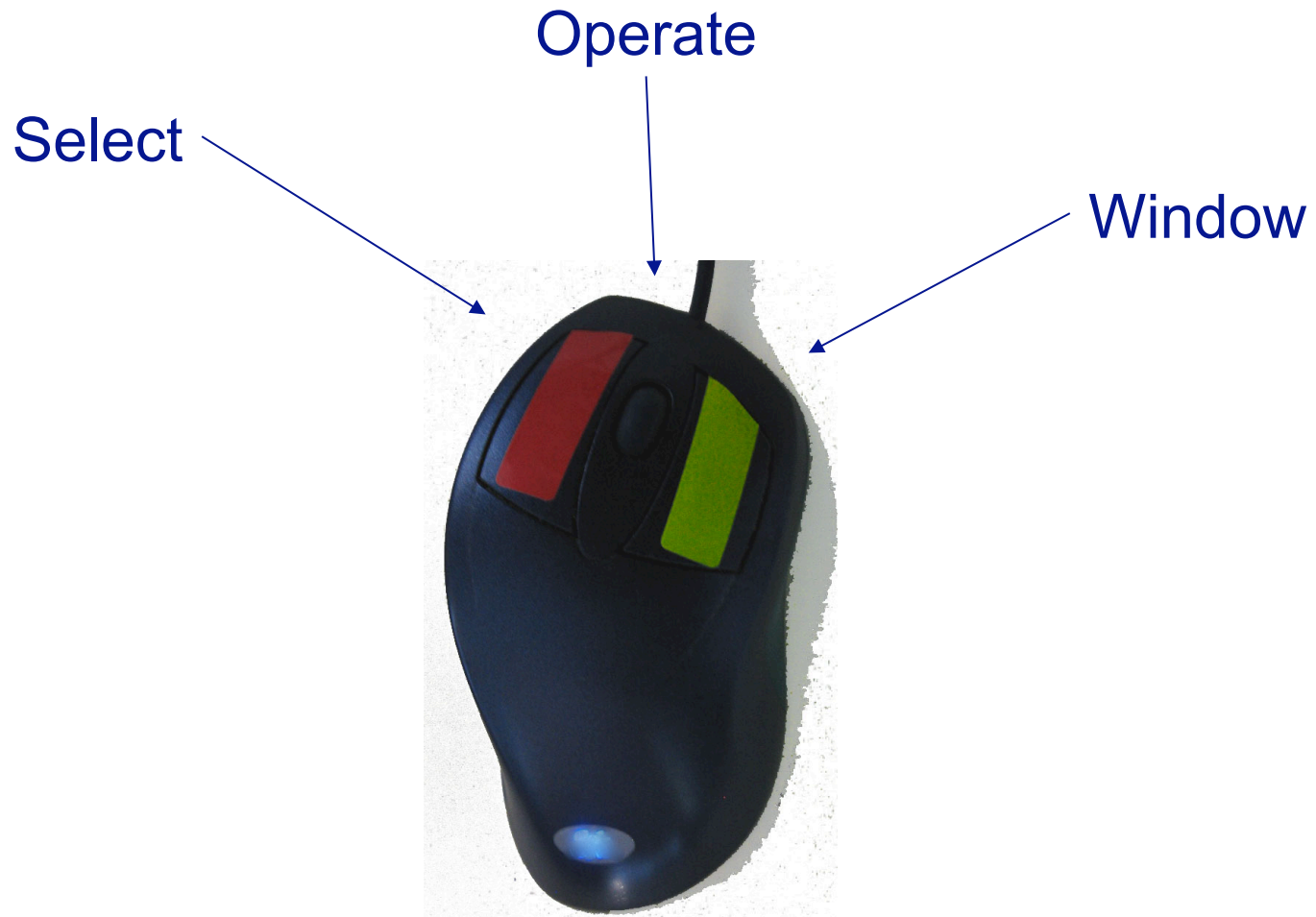
Sources



# Running Squeak

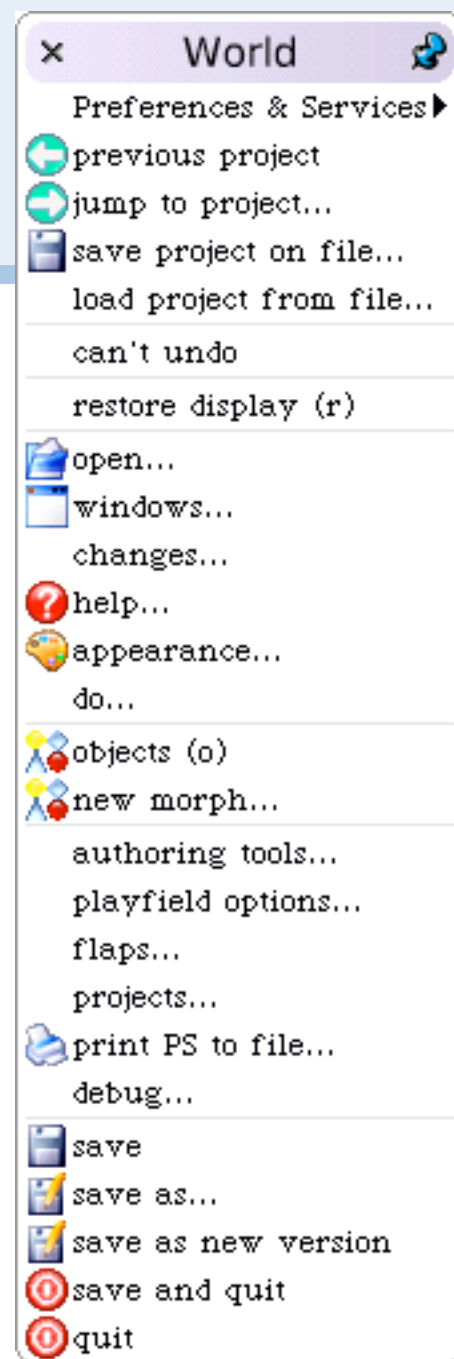
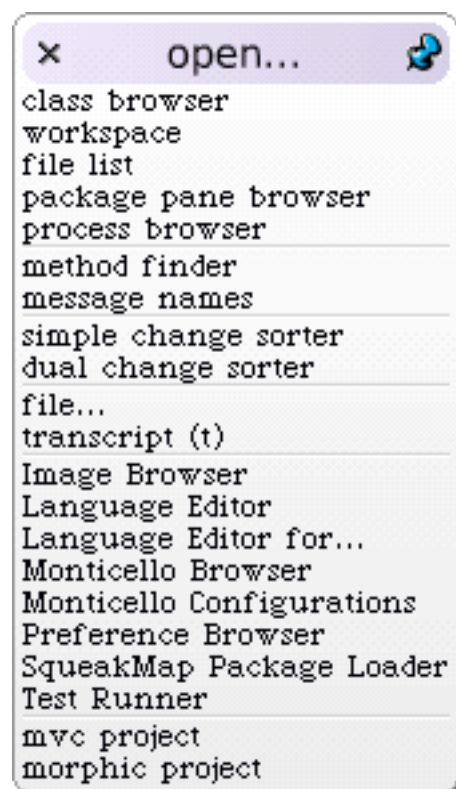


# Three-button mouse

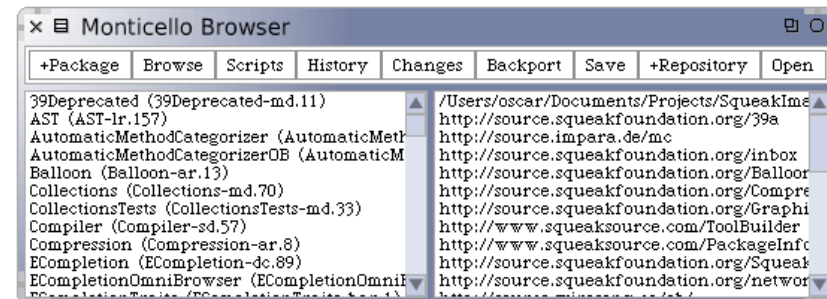
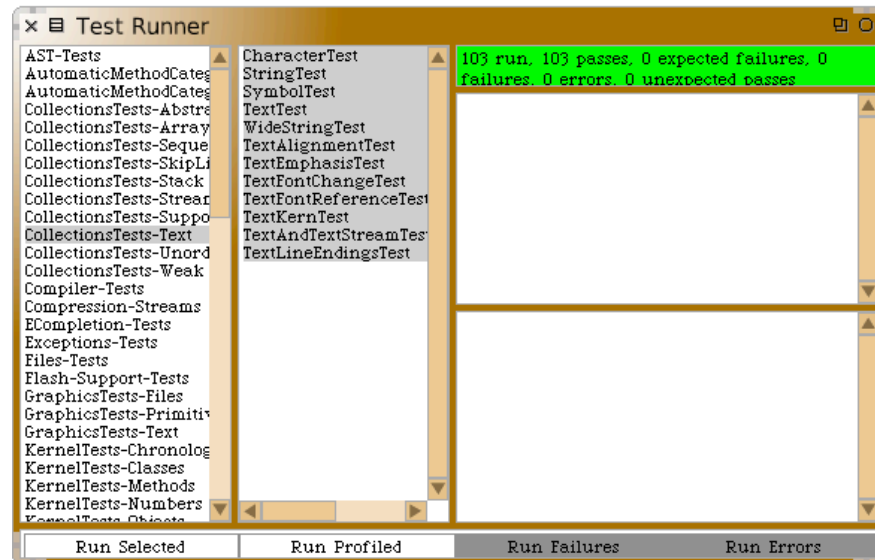
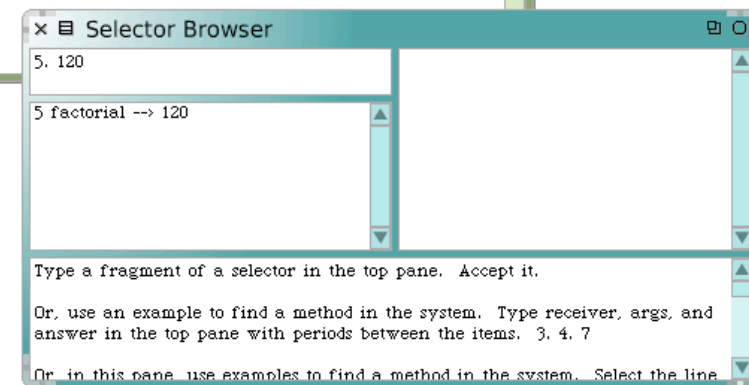
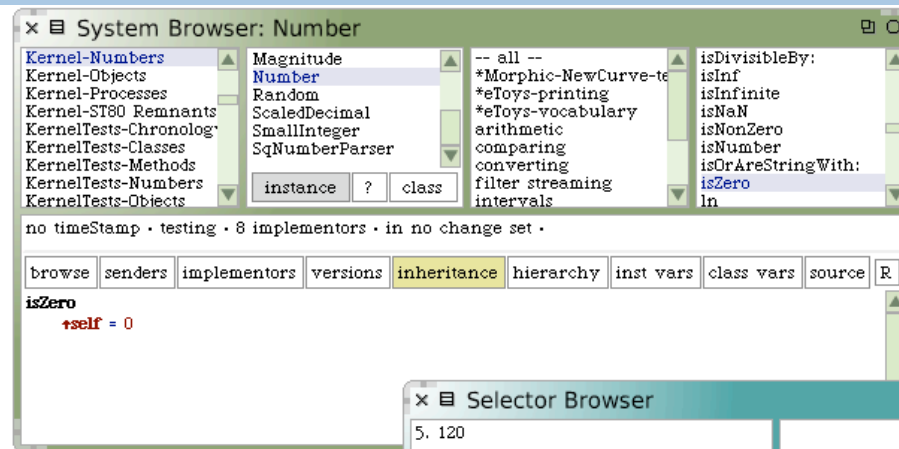
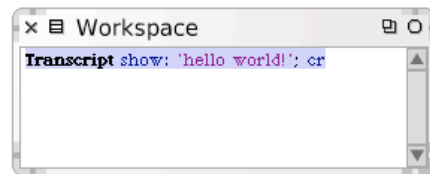
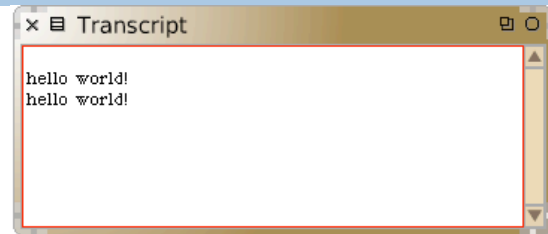




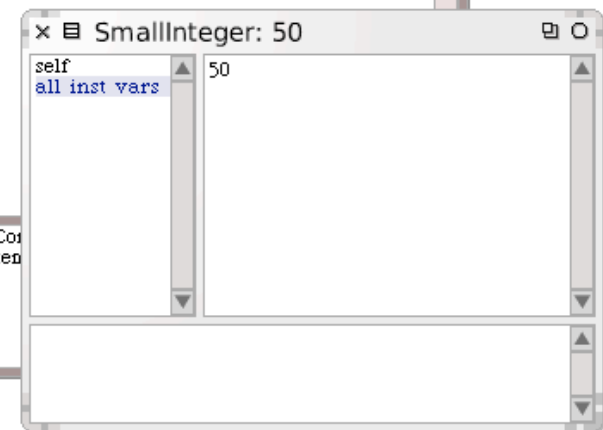
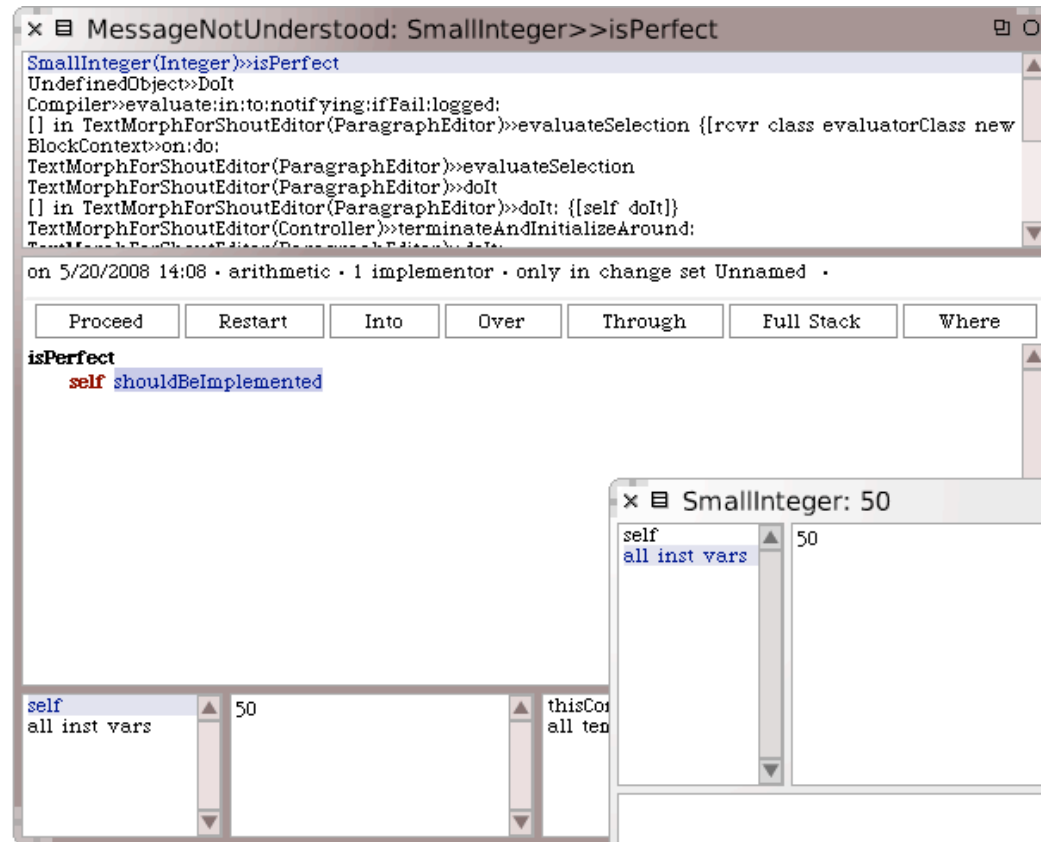
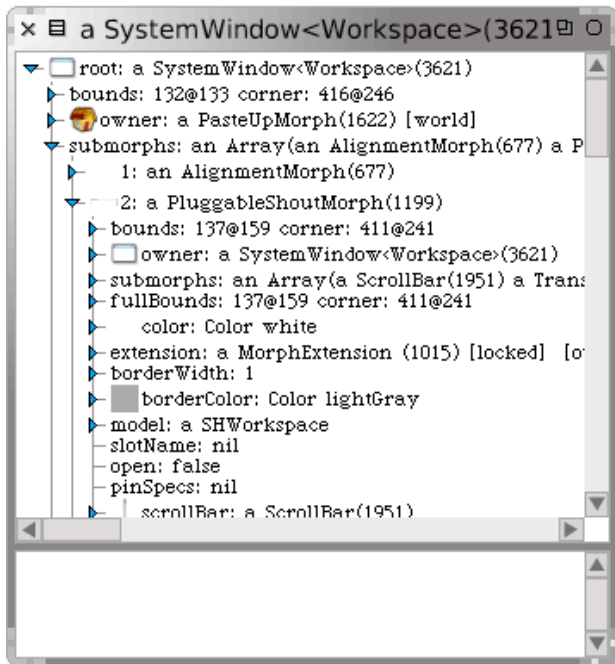
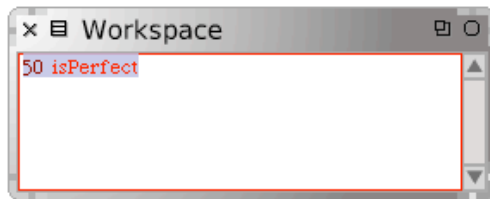
# World Menu and Open Menu



# Standard development tools

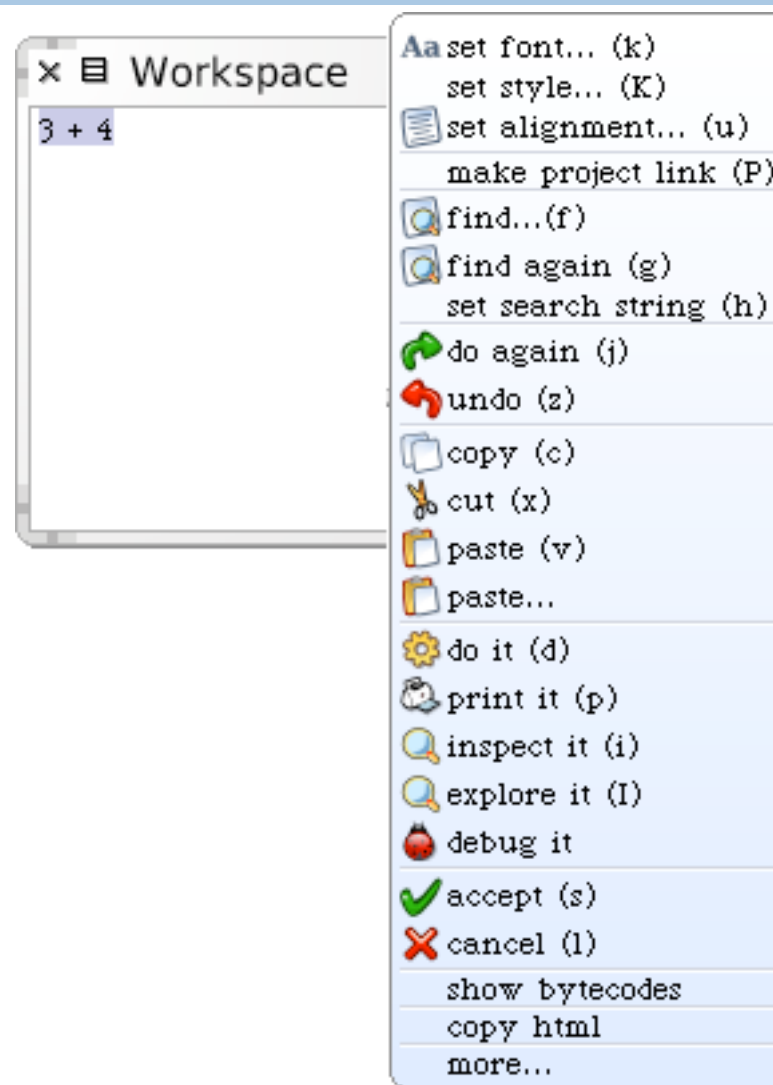


# Debuggers, Inspectors, Explorers



# Do it, Print it, ...

*You can evaluate any expression anywhere in Smalltalk.*



# Roadmap

- > The origins of Smalltalk
- > What is Smalltalk?
- > **Syntax in a nutshell**
- > Seaside — web development with Smalltalk



## Three kinds of messages

> *Unary messages*

```
5 factorial  
Transcript cr
```

> *Binary messages*

```
3 + 4
```

> *Keyword messages*

```
3 raisedTo: 10 modulo: 5
```

```
Transcript show: 'hello world'
```

# Precedence

*First unary, then binary, then keyword:*

```
2 raisedTo: 1 + 3 factorial
```

```
128
```

*Same as:*

```
2 raisedTo: (1 + (3 factorial))
```

*Use parentheses to force order:*

```
1 + 2 * 3
```

```
1 + (2 * 3)
```

```
9 (!)
```

```
7
```

# A typical method in the class Point

Method name

Argument

Comment

```
<= aPoint
```

```
"Answer whether the receiver is neither
below nor to the right of aPoint."
```

```
^ x <= aPoint x and: [y <= aPoint y]
```

Return

Binary message

Block

Instance variable

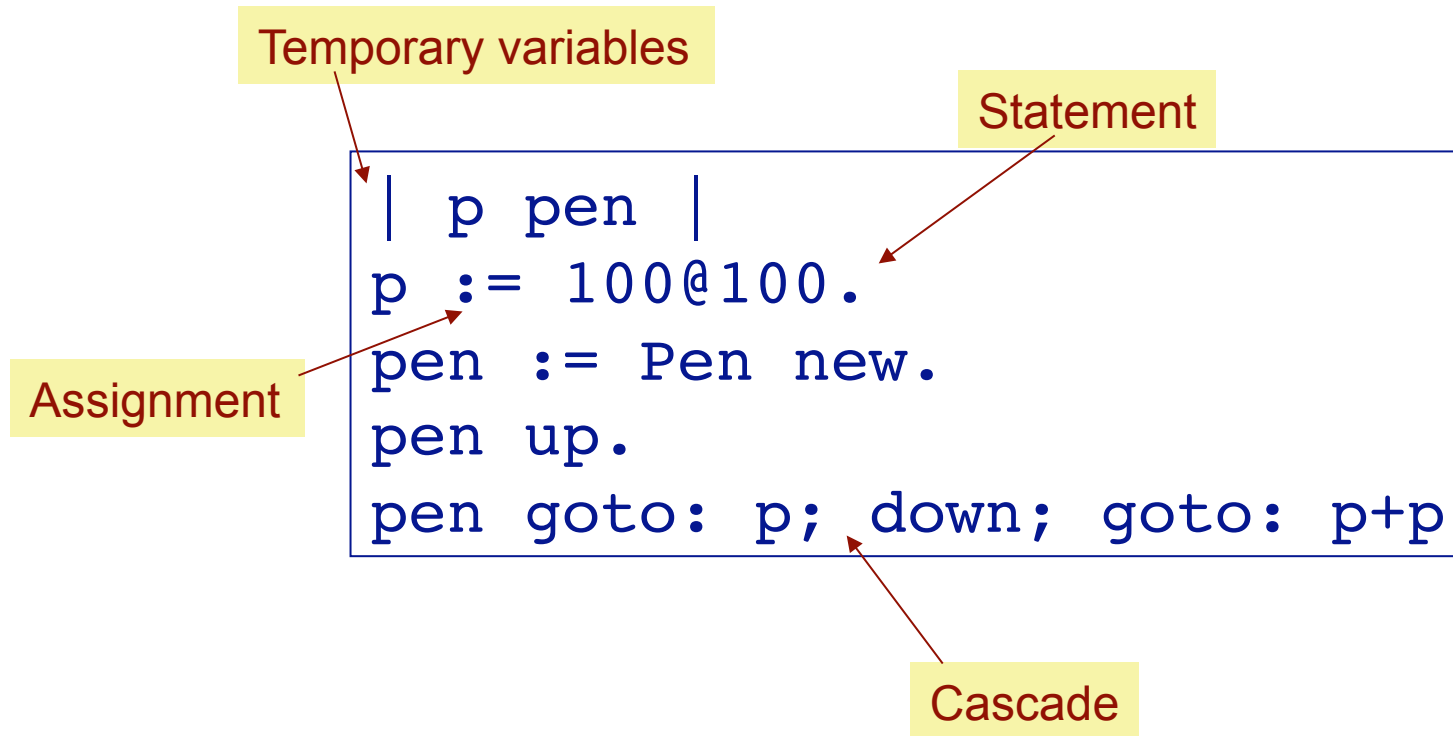
Keyword message

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

```
true
```



# Statements and cascades



# Literals and constants

<b>Strings &amp; Characters</b>	'hello' \$a
<b>Numbers</b>	1 3.14159
<b>Symbols</b>	#yadayada
<b>Arrays</b>	#(1 2 3)
<b>Pseudo-variables</b>	self super
<b>Constants</b>	true false

# Method Return

- > Use a *caret* to return a value from a method or a block

```
max: aNumber  
  ^ self < aNumber  
    ifTrue: [aNumber]  
    ifFalse: [self]
```

```
1 max: 2
```

```
2
```

- > By default, methods return `self`

# Blocks

- > Use *square brackets* to delay evaluation of expressions

```
^ 1 < 2 ifTrue: ['smaller'] ifFalse: ['bigger']
```

```
'smaller'
```

# Variables

- > *Local variables* are delimited by `| var |`  
*Block variables* by `: var |`

```
OrderedCollection>>collect: aBlock
  "Evaluate aBlock with each of my elements as the argument."
  | newCollection |
  newCollection := self species new: self size.
  firstIndex to: lastIndex do:
    [ :index |
      newCollection addLast: (aBlock value: (array at: index))].
  ^ newCollection
```

```
(OrderedCollection with: 10 with: 5) collect: [:each| each factorial ]
```

```
an OrderedCollection(3628800 120)
```

# Control Structures

- > *Every control structure is realized by message sends*

```
max: aNumber  
  ^ self < aNumber  
    ifTrue: [aNumber]  
    ifFalse: [self]
```

```
4 timesRepeat: [Beeper beep]
```

# Creating objects

## > *Class methods*

```
OrderedCollection new  
Array with: 1 with: 2
```

## > *Factory methods*

```
1@2
```

```
1/2
```

```
a Point
```

```
a Fraction
```

# Creating classes

- > Send a message to a class (!)

```
Number subclass: #Complex
  instanceVariableNames: 'real imaginary'
  classVariableNames: ''
  poolDictionaries: ''
  category: 'ComplexNumbers'
```

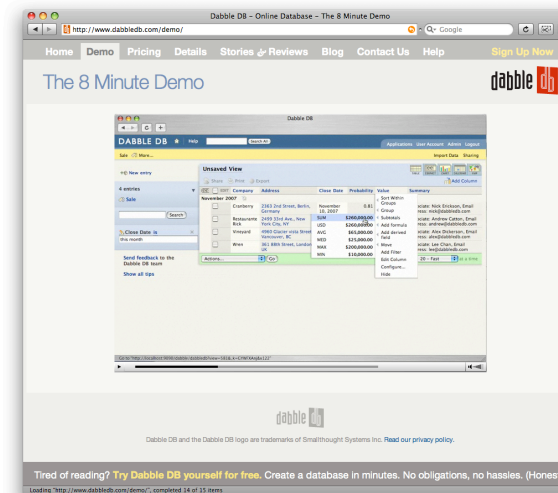
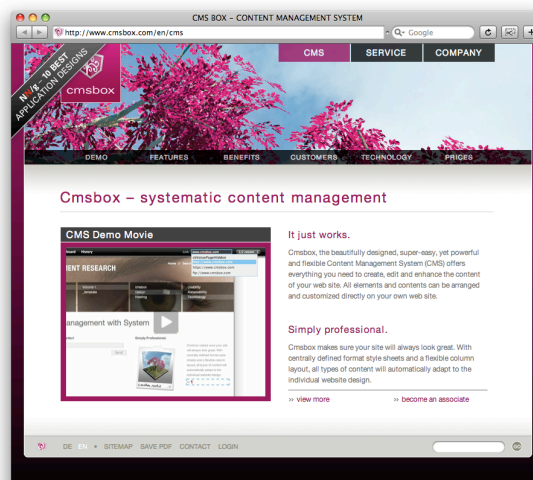
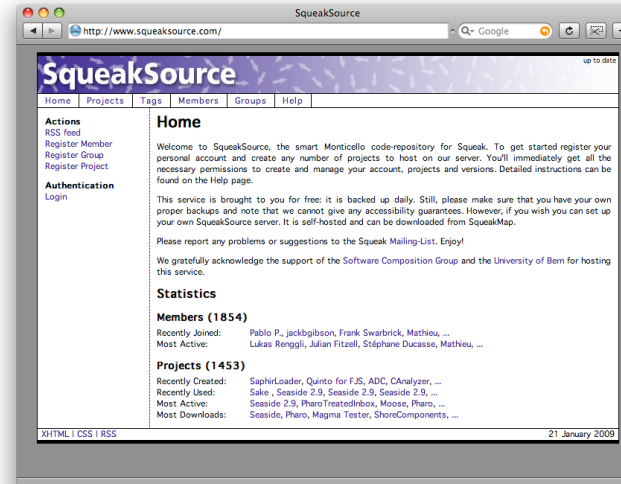


# Roadmap

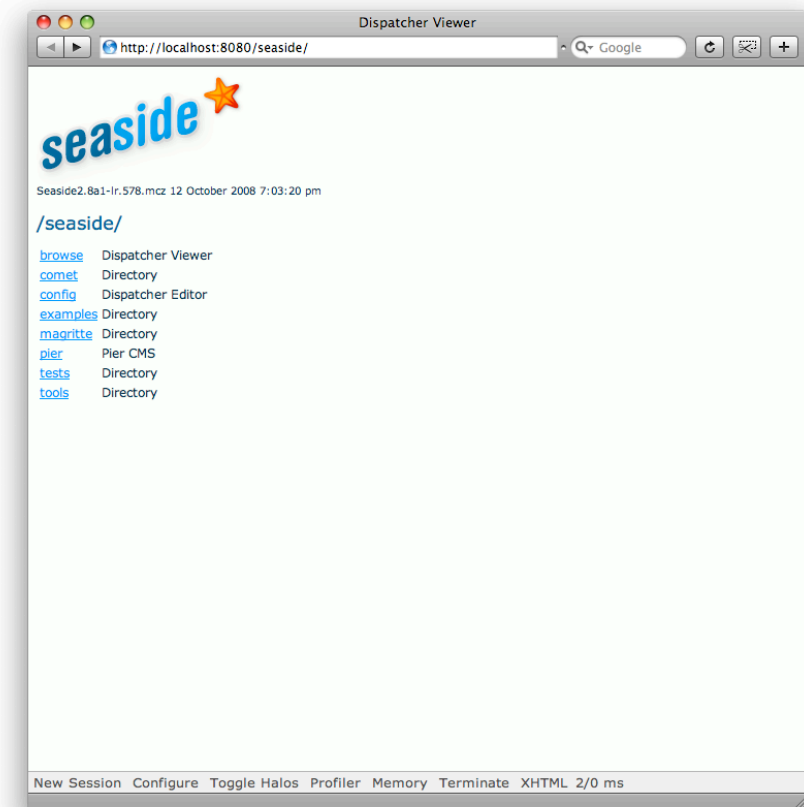
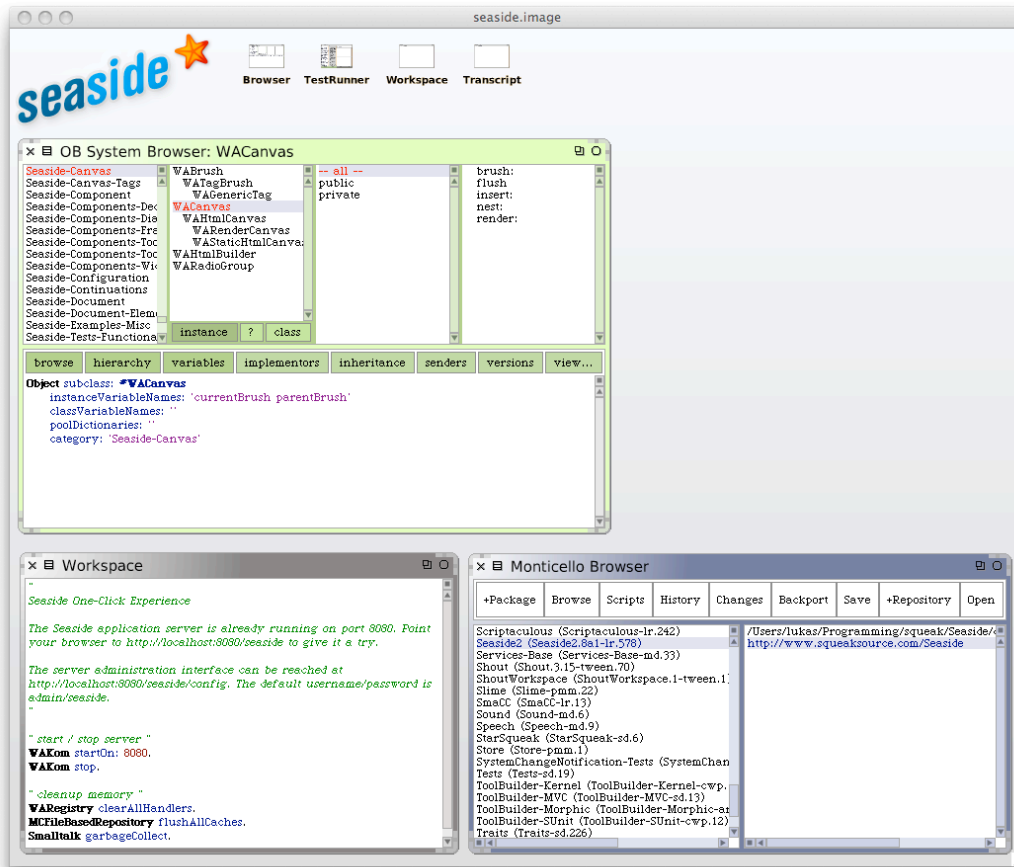
- > The origins of Smalltalk
- > What is Smalltalk?
- > Syntax in a nutshell
- > **Seaside — web development with Smalltalk**












# Seaside — a Smalltalk web development platform











# Seaside demo



## ***What you should know!***

-  *What are the key differences between Smalltalk, C++ and Java?*
-  *What is at the root of the Smalltalk class hierarchy?*
-  *What kinds of messages can one send to objects?*
-  *What is a cascade?*
-  *Why does  $1+2 / 3 = 1$  in Smalltalk?*
-  *How are control structures realized?*
-  *How is a new class created?*
-  *What are categories for?*
-  *What are Factory methods? When are they useful?*

## *Can you answer these questions?*

-  *Which is faster, a program written in Smalltalk, C++ or Java?*
-  *Which is faster to develop & debug, a program written in Smalltalk, C++ or Java?*
-  *How are Booleans implemented?*
-  *Is a comment an Object? How would you check this?*
-  *What is the equivalent of a static method in Smalltalk?*
-  *How do you make methods private in Smalltalk?*
-  *What is the difference between = and ==?*
-  *If classes are objects too, what classes are they instances of?*

# License

<http://creativecommons.org/licenses/by-sa/2.5/>



**You are free:**

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

**Under the following conditions:**



**Attribution.** You must attribute the work in the manner specified by the author or licensor.



**Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

**Your fair use and other rights are in no way affected by the above.**