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
Object-oriented language genealogy
# Smalltalk vs. C++ vs. Java

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

Downloads and links

Free download — Print-on-demand

One-click image
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?
Running Squeak
Three-button mouse

Select

Operate

Window
World Menu and Open Menu

open...

- class browser
- workspace
- file list
- package pane browser
- process browser
- method finder
- message names
- simple change sorter
- dual change sorter
- file...
- transcript (t)
- Image Browser
- Language Editor
- Language Editor for...
- Monticello Browser
- Monticello Configurations
- Preference Browser
- SqueakMap Package Loader
- Test Runner
- mvc project
- morphic project

World

- Preferences & Services
  - previous project
  - jump to project...
  - save project on file...
  - load project from file...
  - can’t undo
  - restore display (r)
- open...
- windows...
  - changes...
- help...
- appearance...
  - do...
- objects (o)
- new morph...
  - authoring tools...
  - playfield options...
  - flaps...
  - projects...
- print PS to file...
  - debug...
- save
- save as...
- save as new version
- save and quit
- quit
Standard development tools

- Transcript
- Workspace
- Test Runner
- System Browser: Number
- Selector Browser
- Monticello Browser
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**

- $5 \text{ factorial}$
- Transcript cr

> **Binary messages**

- $3 + 4$

> **Keyword messages**

- $3 \text{ raisedTo: 10 \ modulo: 5}$
- Transcript show: 'hello world'
Precedence

First unary, then binary, then keyword:

\[ 2 \text{ raisedTo: } 1 + 3 \text{ factorial} \]

128

Same as:

\[ 2 \text{ raisedTo: } (1 + (3 \text{ factorial})) \]

Use parentheses to force order:

\[ 1 + 2 \times 3 \]

9 (!)

\[ 1 + (2 \times 3) \]

7
A typical method in the class Point

\[
\leq \ aPoint
\]

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

\[
^x \leq \ aPoint \ x \ \text{and:} \ [y \leq \ aPoint \ y]
\]

\[
(2@3) \leq (5@6) \quad \text{true}
\]
Statements and cascades

```
p | pen |
p := 100@100.
pen := Pen new.
pen up.
pen goto: p; down; goto: p+p
```
### Literals and constants

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strings &amp; Characters</td>
<td>'hello'</td>
</tr>
<tr>
<td></td>
<td>$a</td>
</tr>
<tr>
<td>Numbers</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3.14159</td>
</tr>
<tr>
<td>Symbols</td>
<td>#yadayada</td>
</tr>
<tr>
<td>Arrays</td>
<td>#(1 2 3)</td>
</tr>
<tr>
<td>Pseudo-variables</td>
<td>self</td>
</tr>
<tr>
<td></td>
<td>super</td>
</tr>
<tr>
<td>Constants</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>
> Use a *caret* to return a value from a method or a block

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

> 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 |

```smalltalk
OrderedCollection>>collect: aBlock
  "Evaluate aBlock with each of my elements as the argument."
  | newCollection |
  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)
> *Every control structure is realized by message sends*

```smalltalk
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  a Point
1/2  a Fraction
Creating classes

> Send a message to a class (!)

```plaintext
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
Smalltalk

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.