Code environments using the listings package

Sample code environment Use the script, method, classdef and example environments. Each takes a name and an optional label. With labels we can have cross references to `\egref{history}` exemple **??**, `\scrref{helloworld}` script **??**, `\clsref{myclass}` classe **??**, and `\mthref{doit}` méthode **??**.

```
\begin{example}[history]{The first thing that Smalltalk could do}{}
3 + 4 --> 7
\end{example}
```

Exemple 1 – *The first thing that Smalltalk could do*

```
3 + 4  ⟶  7
```

```
\begin{script}[helloworld]{The first thing you should try}
Transcript show: 'hello world'
\end{script}
```

Script 2 – *The first thing you should try*

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

```
\begin{classdef}[myclass]{MyClass is defined}
Object subclass: #MyClass
instancevariables: '...'
...
\end{classdef}
```

Classe 3 – *MyClass is defined*

```
Object subclass: #MyClass
   instancevariables: '...'
   ...
```

```
\begin{method}[doit]{A doit method}
MyClass>>>doit
<tab>^ super doit
\end{method}
```

Méthode 4 – *A doit method*

```
MyClass»doit
   ↑ super doit
```

The plain code environment has no title, label or numbering :

```
\begin{code}{}
just some plain code
\end{code}
```

```
just some plain code
```

Listings environments and macros The code environments

```
\begin{code}{}
    ...
\end{code}
```

take plain, verbatim code, and translate some special characters like ∧ to ↑. Even tabs are handled, (which is not true for verbatim).

```
"All handled correctly: ↑ $ ' % \\ << >> ← { }"
"NB: If you really want an exclamation mark you must spell it !"
"
If you really want a caret to look like this: ^ instead of this: ↑, you must spell it out."
| y |
true & false not & (nil isNil) ifFalse: [self halt].
y ← self size + super size.
#($a #a 'a' 1 1.0)
   do: [:each | Transcript
         show: (each class name);
               show: ' ';
               show: (each printString).
{ 1 + 2 . 3 \\ 4 . 1 << 3. 2 >> 5 . 1 % 2 }.
↑ x < y
```

QWERTY layout :

```
! @ # $ % ↑ & * ( ) _ +
1 2 3 4 5 6 7 8 9 0 - =
 Q W E R T Y U I O P { }
 q w e r t y u i o p [ ]
  A S D F G H J K L : "" |        (twice " to turn off italics")
  a s d f g h j k l ; ' \
   Z X C V B N M < > ?
   z x c v b n m , . /
```

LaTeX escape :

```
\begin{code}{}
plain code and !\textbf{bolded text}!
\end{code}
```

```
plain code and bolded text
```

In-line code with `\ct` is typed like this `\ct{1 + 2 --> 3}` and looks like this : 1 + 2 ⟶ 3, text can follow immediately. The "brackets" around `\ct` can be any matching pair of characters, useful if you want { and } in the code.

Special chars with \ct

```
   ↑ ~ # $ ' % \\ << >> ← { } ! --  ⟶
\ct=^ ~ # $ ’ % \\ << >> _ {  } ! -- --> =
```

Special conventions

`\ct{Class>>>method}` prints as Class»method.
`\ct{3 + 4 - 5 --> 2}` prints as 3 + 4 - 5 ⟶ 2.

Use `@TEST` to include the code in the automatic tests and use `-->` to represent the expected result

```
\begin{code}{@TEST}
true        --> true
3@4      --> 3@4
$a          --> $a
#(1 2 3) --> #(1 2 3)
\end{code}
```

Other macros

url SqueakByExample.org

names SUnit xUnit Smalltalk Squeak Pharo

editorial À CORRIGER! please rephrase this please insert this text ~~delete this~~ and ~~change this~~→to this Andrew ►...◄

abbreviation *c-à-d. par ex.*, etc

Smalltalk macros sep : »

scat and prot System category *Kernel-Objects* and protocol *accessing*.

menu World menu ▷ open ...

button Create

do this `\dothis` ⟶  *Download and install Pharo.*

Keyboard shortcut `\short{d}` ⟶ CMD-d

Footnote citations

There is a great book on Squeak by Ducasse[1].

1. ?, .

Important stuff

This is really important

Bibliographie

Stéphane Ducasse: Squeak : Learn Programming with Robots. APress, 2005, ISBN : 1-59059-491-6