學習

smalltalk

#smalltalk

1: smalltalk

smalltalk。

smalltalk。 Smalltalk。

Examples

SmalltalkANSI SmalltalkSmalltalk 80。　。

。

FOSS

Pharo。　Windows / Linux / Mac OSX。　1

Windows / Linux / Mac OSX

GNU Smalltalk Windows / Linux / Mac OSX

Dolphin Smalltalk。　Windows

Cuis Smalltalk Squeak fork。

VisualWorks / Cincom Smalltalk。

VisualAge SmalltalkIBMInstatiations。

Smalltalk / x

GemStone / s。

Smalltalk

Amber SmalltalkSmalltalk。

JVMRedline Smalltalk Smalltalk。

world.stSmalltalk

SmalltalkHello World

```
Transcript show: 'Hello World!'.
```

`Hello World!` **SmalltalkTranscript**。 `Transcriptshow:` **Transcript**。 。 **Smalltalk**。

2: Smalltalk

Examples

```
"Comments are enclosed in double quotes. BEWARE: This is NOT a string!"
"They can span
 multiple lines."
```

```
'Strings are enclosed in sigle quotes.'
'Single quotes are escaped with a single quote, like this: ''.'
''''  "<--This string contains one single quote"

'Strings too can span
multiple lines'

''   "<--An empty string."
```

```
#thiIsaSymbol  "Symbols are interned strings, used for method and variable names,
                and as values with fast equality checking."
#'hello world' "A symbol with a space in it"
#''            "An empty symbol, not very useful"
#+
#1             "Not the integer 1"
```

```
$a     "Characters are not strings. They are preceded by a $."
$A     "An uppercase character"
$      "The spacecharacter!"
$→     "An unicode character"
$1     "Not to be confused with the number 1"
```

- 10

  -1

  0

  1000000000000000000000000000000000000000000

- 16rAB1F

  16r0

  -16rFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

- ScaledDecimal

  17s0

  3.14159265s8

- 8r7731""

  2r1001""

  10r99987""

```
3.14 1.2e3     "2 floating-point numbers"
```

。

```
4/3            "The fraction 4/3"
355/113        "A rational approximation to pi"
```

———

```
#(#abc 123)    "A literal array with the symbol #abc and the number 123"
```

———

```
#[1 2 3 4]       "separators are blank"
#[]              "empty ByteArray"
#[0 0 0 0 255]   "length is arbitrary"
```

———

。 。

```
{self foo. 3 + 2. i * 3}   "A dynamic array built from 3 expressions"
```

———

```
[ :p | p asString ] "A code block with a parameter p.
                    Blocks are the same as lambdas in other languages"
```

———

```
#(256 16rAB1F 3.14s2 2r1001 $A #this)
"is the same as:"
#(256
  16rAB1F
  3.14s2
  2r1001
  $A #this)
```

```
#[255 16rFF 8r377 2r11111111]        (four times 255)
```

```
#(#[1 2 3] #('string' #symbol))     (arrays of arrays)
```

""

```
#(symbol) = #(#symbol)             (missing # => symbol)

#('string' ($a 'a'))               (missing # => array)
```

```
#([1 2 3]) ~= #(#[1 2 3])          (missing # => misinterpreted)
```

```
#(true nil false)                  (pseudo variables ok)
```

```
#(self) = #(#self)                 (missing # => symbol)
```

- true falsenilselfsuper。

- #(#(#[ByteArray。

Smalltalk。

```
#(1 2 3) size
"This sends the #size message to the #(1 2 3) array.
 #size is a unary message, because it takes no arguments."
```

```
1 + 2
"This sends the #+ message and 2 as an argument to the object 1.
 #+ is a binary message because it takes one argument (2)
 and it's composed of one or two symbol characters"
```

```
'Smalltalk' allButFirst: 5.
"This sends #allButFirst: with argument 5 to the string 'Smalltalk',
 resulting in the new string 'talk'"

3 to: 10 by: 2.
"This one sends the single message #to:by:, which takes two parameters (10 and 2)
 to the number 3.
 The result is a collection with 3, 5, 7, and 9."
```

```
unary > binary > keyword
```

```
1 + 2 * 3 " equals 9, because it evaluates left to right"
1 + (2 * 3) " but you can use parenthesis"

1 to:  #(a b c d) size  by:  5 - 4
 "is the same as:"
1 to: ( #(a b c d) size ) by: ( 5 - 4 )
```

;

```
OrderedCollection new
  add: #abc;
  add: #def;
  add: #ghi;
  yourself.
```

"#newOrderedCollection#new。 OrderedCollection。 #add。 "

Smalltalk IDE。

```
XMLTokenizer subclass: #XMLParser
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'XML-Parser'
```

◦ #subclass:instanceVariableNames:classVariableNames:poolDictionaries:category: ,。

XMLTokenizer#XMLParser。

◦

```
aKeywordMethodWith: firstArgument and: secondArgument
    "Do something with an argument and return the result."

    ^firstArgument doSomethingWith: secondArgument
```

^ 。

```
** anInteger
    "Raise me to anInteger"
    | temp1 temp2 |

    temp1 := 1.
    temp2 := 1.
    1 to: anInteger do: [ :i | temp1 := temp1 * self + temp2 – i ].
    ^temp1
```

temporaries *temp1temp2 i* 。

Smalltalk

```
Ordered CollectionOrderedCollection。
newOrderedCollection add:4add:
```

```
anOrderedCollection := OrderedCollection new.
anOrderedCollection add: 1; add: 2; add: 3; add: 4.
```

◦

---

1. `do:`

   。

   ```
   anOrderedCollection do:[:each | Transcript show: each]. "Prints --> 1234"
   ```

   `:each do:Transcript`。

2. `collect:`

   `collect:`
   `2collect:`

   ```
   evenCollection := anOrderedCollection collect:[:each | each*2]. "#(2 4 6 8)"
   ```

3. `select:`

   `select:`。 `select:` message

   ```
   oddCollection := anOrderedCollection select:[:each | each odd].
   ```

   `each odd`**true**`oddCollection oddCollection#(1 3)`。

4. `reject:`

   `select:select:`**Boolean**`true`。 `false`。 `oddCollection`。 `reject:`

   ```
   oddCollection := anOrderedCollection reject:[:each | each even].
   ```

   `oddCollection#(1 3)`。

Smalltalk。 `Collections`。

Smalltalk https://riptutorial.com/zh-TW/smalltalk/topic/5422/smalltalk

S. No		Contributors
1	smalltalk	Bananaweizen, Community, fede s., Leandro Caniglia, Stephan Eggermont, ybce
2	Smalltalk	aka.nice, fede s., ybce