무료 전자 책

배우기

smalltalk

Free unaffiliated eBook created from
Stack Overflow contributors.

#smalltalk

1
1: smalltalk 2
2
Examples 2
2
OSS . 2
. 2
2
Hello World 2
2: 4
Examples 4
4
4
4
4
4
4
: 4
5
5
5
5
5
: 5
6
7
7
8
8

10

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: smalltalk

It is an unofficial and free smalltalk ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official smalltalk.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: smalltalk

smalltalk .

. Smalltalk .

Examples

Smalltalk ANSI Smalltalk Smalltalk 80 ( ) . ( ) .

.

OSS .

. (Windows / Linux / Mac OSX). Pharo ,

(Windows / Linux / Mac OSX)

GNU (Windows / Linux / Mac OSX)

(Dolphin Smalltalk) . (Windows)

Cuis Smalltalk .

.

VisualWorks / Cincom Smalltalk .

VisualAge Smalltalk IBM , Instatiations.

/ x (?)

GemStone / s .

(Amber Smalltalk) .

JVM .

world.st

Hello World

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

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

smalltalk : https://riptutorial.com/ko/smalltalk/topic/5316/smalltalk-

2:

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

  10000000000000000000000000000000000000000

- 16

  16rAB1F

  16r0

  -16rFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

- ScaledDecimal

  17s0

  3.14159265s8

- 8r7731 "8 "

  2r1001 ""

  10r99987 "decimal again!"

```
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` , `false` `nil` `self` `super` .
- `#(` `#[` `ByteArray` .

.

http://stackoverflow.com/a/37823203/4309858

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.
```

" #new OrderedCollection . (.) OrderedCollection #add ( ) . "

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
```

exponentiation ( ) (, *temp1* *temp2*) (*i*) .

Ordered Collection OrderedCollection .
new OrderedCollection add: 4 add:

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

.

1. do:
   . .

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

   . :each This do: Transcript .

2. collect:
   collect:
   , 2 collect: .

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

3. select:

`select:` . , `select:` :

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

`each odd` `#(1 3)` `oddCollection` .

4. `reject:`
   `select:` `true` `true` . , Boolean `false` . `oddCollection` `reject:` :

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

`oddCollection` `#(1 3)` .

. `Collections` .

: https://riptutorial.com/ko/smalltalk/topic/5422/--

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