無料電子ブック

学習

smalltalk

Free unaffiliated eBook created from
Stack Overflow contributors.

#smalltalk

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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をいめる

このセクションでは、smalltalkのと、がそれをいたいについてします。

また、さなのにあるきなテーマについてもし、するトピックにリンクするがあります。 Smalltalk のドキュメンテーションはしいものなので、それらのトピックのバージョンをするがあります。

Examples

インストールまたはセットアップ

Smalltalkというは、、ANSI SmalltalkまたはSmalltalk 80そのうちの1つがづいていますをします 。ほとんどのはにいものの、さまざまなはとばれます によってなります。

にはのインストールがあります。

よくられているFOSSのはのとおりです。

ファロはフォークからまります。 Windows / Linux / Mac OSX。ファロはStackOverflowのドキュメントで、のドキュメントエントリをっているので、てくださいが

スクィーク Windows / Linux / Mac OSX

GNU Smalltalk Windows / Linux / Mac OSX

Dolphin Smalltalkで、はのオープンソースです。 Windowsのみ

キッチンスモールトーク システムのさをらすことにをてたスクィークフォーク。

Smalltalkにはがまれます

VisualWorks / Cincom Smalltalkがです。

VisualAge Smalltalk IBMの々、Instatiations。

Smalltalk / x のは

GemStone / sコミュニティがです。

そののスモールトークの

Amber Smalltalkブラウザにするスモールトーク。

JVMのためのRedline Smalltalk Smalltalk

world.stでのSmalltalkのリスト

スモールトークのHello World

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

これは、 `Hello World!`をします`Hello World!` SmalltalkのTranscriptウィンドウにします。 `Transcript`は、そのオブジェクトにメッセージ`show:`をすることで、Transcriptウィンドウにするためのクラスです。コロンは、このメッセージがこのはであるパラメータをとすることをします。はとでのみされます。はSmalltalkのコメントにされているからです。

オンラインでsmalltalkをいめるをむ https://riptutorial.com/ja/smalltalk/topic/5316/smalltalkをいめる

2: スモールトークシンタックス

Examples

リテラルとコメント

コメント

```
"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."
```

シンボル

```
#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"
```

キャラクター

```
$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"
```

はすべてのにります

- 10

-1

0

100000000000000000000000000000000000000000

- 16

  16rAB1F

  16r0

  -16rFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

- ScaledDecimal

  17s0

  3.14159265s8

- その

  8r7731 "8"

  2r1001 "バイナリ"

  10r99987 "decimal again"

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

はとじではなく、なですめはありません。

```
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"
```

バイト

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

ダイナミックアレイ

はからされます。のは、されたのなるにされます。

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

ブロック

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

いくつかのメモ

リテラルはのとのをセパレータとしてすることにしてください

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

リテラルをすることもできます

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

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

リラックスしたにはいくつかの「」があります

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

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

しかし、ここではない

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

しかしながら

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

しかし、ここではない

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

ごのとおり、いくつかのがあります

- ながらtrue 、 falseとnilリテラルとしてけれられ、selfとsuper されていないのためのよりなルーールをして。 シンボルとしてされます
- #(をすることはではありませんが#(にネストされたをするにはでですが、ネストされたByteArrayをするには#[をくことがです。

このうちのいくつかは、からされました

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

メッセージ

Smalltalkでは、ほとんどすべてのことがオブジェクトにメッセージをしますののびしメソッドとばれます。メッセージにはの3があります。

メッセージ

```
#(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."
```

バイナリメッセージ

```
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"
```

キーワードメッセージ

```
'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."
```

ステートメントののメッセージは、のにされます

```
unary > binary > keyword
```

からへ。

```
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 )
```

じオブジェクトにくのメッセ―ジをするは、カスケ―ドをできます; セミコロン

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

"これはにメッセ―ジ#newをクラスOrderedCollectionにりますnewはではなくなるメッセ―ジです。しいOrderedCollectionがされ、#addがなるというメッセ―ジの3しいコレクションがされます。そしてあなたのメッセ―ジ。

クラスとメソッド

クラスとメソッドは、、Smalltalk IDEでされます。

クラス

クラスはブラウザでのようになります。

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

これは、には、ブラウザでシステムにしいクラスをするためにするメッセ―ジです。 このは`#subclass:instanceVariableNames:classVariableNames:poolDictionaries:category:`がしいクラスをするのクラスもあります。

のは、サブクラスするクラスこのはXMLTokenizerとしいサブクラスがつ#XMLParserをしています。

の3は、クラスとそのインスタンスがつをするためにされます。

メソッド

メソッドはブラウザでのようになります。

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

```
    ^firstArgument doSomethingWith: secondArgument
```

^ キャレットはリターンです。

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

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

これはをうしいではありませんが、 バイナリメッセージのメッセージのようにされていますといくつかのメソッドの またはメソッド、 *temp1*と*temp2* とブロック *i* をしています。

スモールトークのループ

このでは、 `Ordered Collection`をして、 `OrderedCollection`オブジェクトにしてをループするさまざまなメッセージをします。
のコードは、メッセージ`new`をしての`OrderedCollection`をインスタンスし、メッセージ`add:`をして4つのをします`add:`

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

これらのすべてのメッセージは、コレクションのについてされるパラメータとしてブロックをります。

1. `do:`
   これはなメッセージです。たとえば、コレクションのをするは、そのようにすることができます。

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

   コレクションのは、のをしてユーザーがむものとしてされます。 `:each`この`do:`ループは、コレクションのすべてのを`Transcript`ウィンドウにします。

2. `collect:`
   `collect:`メッセージでは、コレクションのアイテムについてかをうことができ、アクションのをしいコレクションにれます
   たとえば、コレクションのに2をけてしいコレクションにするは、 `collect:`メッセージをのようにできます。

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

3. `select:`
   `select:`メッセージをすると、のコレクションのアイテムが、それらにしてのについてされるサブコレクションをできます。たとえば、コレクションからしいのコレクションをするは、 `select:`メッセージをのようにできます。

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

   `each odd`はブールをすので、ブールをすだけが`#(1 3)`をつ`oddCollection`にされます。

4. `reject:`
   このメッセージは、を`select:`とにし、Booleanを`true`すをし`true`。 つまり、Booleanを`false`すをします。えば、ののようにじ`oddCollection`をしたいなどです。 `reject:`をうことができます

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

   `oddCollection`は、そのとしてび`#(1 3)`をちます。

Smalltalkの4つのなです。ただし、 `Collections`クラスをして、なそののメッセージをにできます。

オンラインでスモールトークシンタックスをむ https://riptutorial.com/ja/smalltalk/topic/5422/スモールトークシンタックス

クレジット

S. No		Contributors
1	smalltalkをいめる	Bananeweizen, Community, fede s., Leandro Caniglia, Stephan Eggermont, ybce
2	スモールトークシンタックス	aka.nice, fede s., ybce