Бесплатная электронная книга

учусь

smalltalk

Free unaffiliated eBook created from
Stack Overflow contributors.

#smalltalk

1
1: smalltalk 2
2
Examples 2
2
FOSS : 2
Smalltalks : 2
Smalltalk 3
Smalltalk 3
2: Smalltalk 4
Examples 4
4
4
4
4
4
4
: 5
5
5
6
- 6
6
6
6
: 6
7
8
8
9
Smalltalk 9

12

Около

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

глава 1: Начало работы с smalltalk

замечания

В этом разделе представлен обзор того, что такое smalltalk, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в smalltalk и ссылаться на связанные темы. Поскольку Documentation for Smalltalk является новым, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Установка или настройка

Название Smalltalk обычно относится к ANSI Smalltalk или Smalltalk 80 (из которых первый основан на). Хотя большинство реализаций близки к стандарту, они различаются в разных аспектах (обычно называемых *диалектами*).

Каждая реализация имеет собственный метод установки.

Известными реализациями FOSS являются:

Фаро начинал как вилка для скрипа. (Windows / Linux / Mac OSX). У Pharo есть своя документация в документации Stackoverflow, поэтому, пожалуйста, посмотрите там

Squeak (Windows / Linux / Mac OSX)

GNU Smalltalk (Windows / Linux / Mac OSX)

Dolphin Smalltalk Первоначально коммерческий, теперь бесплатный открытый исходный код. (Только для Windows)

Cuis Smalltalk Вилка Squeak с акцентом на снижение сложности системы.

Коммерческие Smalltalks включают:

Доступна бесплатная пробная версия VisualWorks / Cincom Smalltalk .

VisualAge Smalltalk Первоначально IBM, теперь Instatiations. Бесплатная пробная версия

Smalltalk / x (бесплатно для личного пользования?)

GemStone / s Бесплатная версия сообщества доступна.

Другие диалекты Smalltalk

Янтарный Smalltalk Smalltalk, который живет в браузере.

Redline Smalltalk Smalltalk для JVM.

Список реализаций Smalltalk на world.st

Привет мир в Smalltalk

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

Это напечатает `Hello World!` в окно Transcript в Smalltalk. `Transcript` - это класс, который позволяет печатать в окне Transcript, отправляя сообщение `show:` этому объекту. Двоеточие указывает, что для этого сообщения требуется параметр, который в этом случае является строкой. Строки представлены одинарными кавычками и одинарными кавычками только потому, что двойные кавычки зарезервированы для комментариев в Smalltalk.

Прочитайте Начало работы с smalltalk онлайн: https://riptutorial.com/ru/smalltalk/topic/5316/начало-работы-с-smalltalk

глава 2: Синтаксис Smalltalk

Examples

Литералы и комментарии

коментарии

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

Струны

```
'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."
```

Символы

```
#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"
```

Персонажи

```
$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"
```

чисел

Числа входят во все разновидности:

Целые:

- Десятичный

  10

  -1

  0

  1000000000000000000000000000000000000000000000

- шестнадцатеричный

  16rAB1F

  16r0

  -16rFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

- ScaledDecimal

  17s0

  3.14159265s8

- Другой

  8r7731 "восьмеричный"

  2r1001 "двоичный"

  10r99987 «снова десятичный!»

Плавающая запятая

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

Фракции

Фракции не совпадают с числами с плавающей запятой, они являются точными числами (без ошибок округления).

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

Массивы

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

Байт-массивы

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

Динамические массивы

Динамические массивы построены из выражений. Каждое выражение внутри фигурных скобок вычисляется в другом значении в построенном массиве.

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

Блоки

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

Некоторые примечания:

Обратите внимание, что литеральные массивы используют любые типы и количество пробелов в качестве разделителей

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

Заметим также, что вы можете составлять литералы

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

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

Существует некоторая «толерантность» к расслабленной нотации

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

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

Но не здесь:

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

тем не мение

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

Но не здесь!

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

Как видите, есть несколько несоответствий:

- Хотя псевдо переменные `true` , `false` и `nil` принимаются как литералы внутри массивов, псевдопеременные `self` и `super` интерпретируются как **символы** (используя более общее правило для неквалифицированных строк).
- Хотя не обязательно писать `#(` для запуска вложенного массива в массиве и достаточной скобки, обязательно писать `#[` для запуска вложенного `ByteArray` .

Некоторые из них были взяты из:

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

Передача сообщений

В Smalltalk почти все, что вы делаете, *отправляет сообщения* на объекты (называемые вызовами на других языках). Существует три типа сообщений:

Унарные сообщения:

```
#(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."
```

Бинарные сообщения:

```
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"
```

Сообщения по ключевому слову:

```
'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."
```

Несколько сообщений в инструкции оцениваются порядком приоритета

```
unary > binary > keyword
```

и слева направо.

```
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 )
```

Если вы хотите отправить много сообщений на один и тот же объект, вы можете использовать каскадный оператор ; (точка с запятой):

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

«Это сначала отправляет сообщение #new классу OrderedCollection (#new - это просто сообщение, а не оператор), в результате получается новый OrderedCollection, который затем отправляет новую коллекцию в три раза сообщение #add (с разными аргументами) и само сообщение.

Классы и методы

Классы и методы обычно определяются в среде Smalltalk IDE.

Классы

Определение класса в браузере выглядит примерно так:

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

На самом деле это сообщение, которое браузер отправит вам для создания нового класса в системе. (В этом случае это `#subclass:instanceVariableNames:classVariableNames:poolDictionaries:category:` но есть и другие, которые также создают новые классы).

В первой строке показан класс, который вы подклассифицируете (в данном случае это XMLTokenizer) и имя нового подкласса (#XMLParser).

Следующие три строки используются для определения переменных, которые будут иметь класс и его экземпляры.

методы

Методы в браузере выглядят так:

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

    ^firstArgument doSomethingWith: secondArgument
```

Оператор `^` (карет) является оператором возврата.

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

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

это неправильный способ сделать возведение в степень, но он показывает определение *двоичного сообщения* (они определены как любое другое сообщение) и некоторые *временные переменные* метода (или временные *методы temp* , *temp1* и *temp2*) плюс блок-аргумент (*i*).

Петли в Smalltalk

В этом примере `Ordered Collection` будет использоваться для отображения различных сообщений, которые могут быть отправлены объекту `OrderedCollection` для циклического

`OrderedCollection` по элементам.
В приведенном ниже коде создается экземпляр пустого `OrderedCollection` с использованием `new` сообщения, а затем заполняется его четырьмя номерами с помощью сообщения `add:`

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

Все эти сообщения будут принимать блок как параметр, который будет оцениваться для каждого из элементов внутри коллекции.

1. `do:`
   Это основное сообщение перечисления. Например, если мы хотим напечатать каждый элемент в коллекции, мы можем добиться этого как такового:

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

   Каждый из элементов внутри коллекции будет определяться как пользователь хочет использовать этот синтаксис `:each` This `do:` loop будет печатать каждый элемент в коллекции в окне `Transcript` .

2. `collect:`
   Сообщение `collect:` позволяет сделать что-то для каждого элемента в коллекции и помещает результат вашего действия в новую коллекцию
   Например, если мы хотели бы умножить каждый элемент в нашей коллекции на 2 и добавить его в новую коллекцию, мы можем использовать сообщение `collect:` как таковое:

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

3. `select:`
   Сообщение `select:` позволяет создать подсерию, в которой элементы из исходной коллекции выбраны на основе некоторого условия, являющегося для них истинным.
   Например, если мы хотим создать новую коллекцию нечетных чисел из нашей коллекции, мы можем использовать сообщение `select:` как таковое:

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

   Поскольку `each odd` возвращает Boolean, в `oddCollection` будут добавлены только те элементы, которые делают логическое возвращение true, и будут иметь `#(1 3)` .

4. `reject:`

Это сообщение работает напротив `select:` и отклоняет любые элементы, которые возвращают Boolean `true` . Или, другими словами, он выберет любые элементы, которые возвращают логическое значение `false` . Например, если мы хотим создать тот же `oddCollection` как в предыдущем примере. Мы можем использовать `reject:` как таковой:

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

`oddCollection` снова будет содержать `#(1 3)` .

Это четыре основных метода перечисления в Smalltalk. Однако не стесняйтесь просматривать класс `Collections` для получения большего количества сообщений, которые могут быть реализованы.

Прочитайте Синтаксис Smalltalk онлайн: https://riptutorial.com/ru/smalltalk/topic/5422/синтаксис-smalltalk

кредиты

S. No	Главы	Contributors
1	Начало работы с smalltalk	Bananeweizen, Community, fede s., Leandro Caniglia, Stephan Eggermont, ybce
2	Синтаксис Smalltalk	aka.nice, fede s., ybce