Языки программирования ruby и crystal

Выполнил студент ФИТУ 4-6
Гадзюк Дмитрий

- **Ruby** (англ. *ruby* — рубин, произносится ['ru:bɪ] — *ру́би*)- динамический, рефлективный, интерпретируемый высокоуровневый язык программирования. Язык обладает независимой от операционной системы реализацией многопоточности, сильной динамической типизацией, сборщиком мусора и многими другими возможностями. По особенностям синтаксиса он близок к языкам Perl и Eiffel, по объектно-ориентированному подходу — к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и Клу.
- Кроссплатформенная реализация интерпретатора языка является полностью свободной.

История создания и развития

- Создатель Ruby — Юкихиро Мацумото (Matz) — интересовался языками программирования, ещё будучи студентом, но идея о разработке нового языка появилась позже. Ruby начал разрабатываться 23 февраля 1993 года и вышел в свет в 1995 году.
- Название навеяно языком Perl, многие особенности синтаксиса и семантики из которого заимствованы в Ruby: англ. *pearl* — «жемчужина», *ruby* — «рубин».
- Одним из источников вдохновения для Мацумото для разработки Ruby был научно-фантастический роман «Вавилон-17», основанный на гипотезе Сепира — Уорфа.
- Целью разработки было создание «настоящего объектно-ориентированного», лёгкого в разработке, интерпретируемого языка программирования. Из письма автора:

- Ruby родился 24 февраля 1993 года. В тот день я беседовал со своим коллегой о возможности существования объектно-ориентированного сценарного языка. Я знал Perl (Perl4, а не Perl5), но он мне не нравился — был в нём некий привкус игрушечного языка (да и поныне есть). А объектно-ориентированный интерпретируемый язык казался многообещающим. В то время я знал Python. Но он мне не нравился потому, что я не считал его настоящим объектно-ориентированным языком. Его ОО-свойства казались надстройкой над языком. Мне, как языковому маньяку и фанату объектно-ориентированного программирования с пятнадцатилетним стажем, очень, очень хотелось, чтобы был истинно объектно-ориентированный, простой в использовании язык. Я пытался найти такой язык, но его не было.
- Тогда я решил его создать. Прошло несколько месяцев, прежде чем интерпретатор заработал. Я добавил в свой язык то, что мне хотелось — итераторы, обработку исключений, автоматическую сборку мусора. Затем я переорганизовал свойства Perl и реализовал их как библиотеку классов. В декабре 1995 года я опубликовал Ruby 0.95 в японских новостных группах. С тех пор появились сайты, списки рассылок. В списках рассылок идут жаркие обсуждения. Самый старый список сейчас содержит 14 789 писем.

- В Японии Ruby стал популярным с момента появления первой общедоступной версии в 1995 году, однако наличие документации только на японском языке сдерживало его дальнейшее распространение. Лишь в 1997 году появилось описание Ruby на английском языке, а в 1998 году открылся форум «ruby-talk». Это положило начало росту известности языка в остальном мире. В начале 2000-х вышло несколько книг на английском языке, что способствовало росту популярности Ruby в Западной Европе и Америке. В 2003 году была выпущена версия Ruby 1.8.0, а в 2005 году появился веб-фреймворк Ruby on Rails, написанный на Ruby и сразу завоевавший признание благодаря лёгкости построения на нём типичных веб-приложений. Ruby в нём является не только языком реализации самого фреймворка, но и языком описания решений (в частности, используются HTML-шаблоны с встроенным кодом на Ruby).

Философия

- Мацумото, фанат объектно-ориентированного программирования, мечтал о языке, более мощном, чем Perl, и более объектно-ориентированном, чем Python. Основное назначение Ruby — создание простых и в то же время понятных программ для решения задач, в которых время разработки, понятность и простота важнее, чем скорость работы.
- Принципы устройства Ruby и программирования на нём иногда выделяются в термин «Путь Ruby» (англ. *Ruby Way*). В целом «путь Ruby» не имеет точной формулировки, иногда этот термин используется для критики. В относительно сжатом виде его положения изложены в книгах «Программирование на языке Ruby» Хэла Фултона и «Путь Ruby» Хэла Фултона и Андре Арке.

- Язык для человека, а не для компьютера.
- Приоритетом является удобство и минимизация затрат труда программиста при разработке программы, освобождение программиста от рутинной работы, которую компьютер может выполнять быстрее и качественнее. Особое внимание, в частности, уделено будничным рутинным занятиям (обработка текстов, администрирование), и для них язык настроен особенно хорошо. В противовес машинно-ориентированным языкам, работающим быстрее, Ruby — язык, наиболее близкий к человеку. Любая работа с компьютером выполняется людьми и для людей, и необходимо заботиться в первую очередь о затрачиваемых усилиях людей.
- Просто, но не слишком просто.
- Упрощение является благом, но оно не должно переходить некие границы, за которыми превращается в самоцель и вредит конечному результату.
- Принцип наименьшей неожиданности

- Не быть рабом производительности.
- Если производительность для конкретного случая недопустимо низка, то это требует исправления, а если заранее известно, что она будет существенна — необходимо изначально проектировать программу с учётом этого. Но следует предпочитать элегантность эффективности в тех случаях, когда эффективность не слишком критична.

- Следовать простым и строгим правилам, но не доходить до педантизма.
- «В Ruby мы видим не „педантичную непротиворечивость“, а строгое следование набору простых правил». Правила и соглашения (в частности, соглашения об именовании, принятые в языке) нужны для того, чтобы сделать понимание программы проще. Если отступление от правила в конкретном случае логично и понятно — оно оправданно.

- «Не нужно с этим бороться».
- Ruby таков, каким он придуман. Программисту не следует ждать, что Ruby будет вести себя так же, как другой привычный ему язык. Программист может следовать своим представлениям и привычкам, сложившимся под влиянием других языков (см. «Принцип наименьшей неожиданности»[3]), но если ожидания оказываются неверны, это нужно просто принять и использовать.

Возможности Ruby

- Имеет лаконичный и простой синтаксис, частично разработанный под влиянием Ада, Eiffel и Python.
- Позволяет обрабатывать исключения в стиле Java и Python.
- Позволяет переопределять операторы, которые на самом деле являются методами.
- Полностью объектно-ориентированный язык программирования. Все данные в Ruby являются объектами в понимании Smalltalk. Например, число «1» — это экземпляр класса Integer. Единственное исключение — управляющие конструкции, которые в Ruby, в отличие от Smalltalk, не являются объектами. Также поддерживается добавление методов в класс и даже в конкретный экземпляр во время выполнения программы.
- Не поддерживает множественное наследование, но вместо него может использоваться концепция «примесей», основанная в данном языке на механизме модулей.
- Содержит автоматический сборщик мусора. Он работает для всех объектов Ruby, в том числе для внешних библиотек.
- Создавать расширения для Ruby на Си очень просто частично из-за сборщика мусора, частично из-за несложного и удобного API.
- Поддерживает замыкания с полной привязкой к переменным.

Алфавит , ключевые слова

- Ruby — регистро-зависимый язык, прописные и строчные буквы в идентификаторах являются различными. Все ключевые слова языка, за двумя исключениями, пишутся в нижнем регистре.
- До версии 2.0 язык использовал множество символов 7-битной кодировки ASCII. Начиная с версии 2.0 поддерживается Unicode, по умолчанию файлы исходного кода используют кодировку UTF-8. Все буквенные символы Unicode допускается использовать в идентификаторах наравне с английскими буквами. Полностью поддерживаются Unicode-строки.
- Список ключевых слов Ruby:

```
alias     and     BEGIN   begin   break   case    class   def
defined?  do      else    elsif   END     end     ensure  false
for       if      in      module  next    nil     not     or
redo      rescue  retry   return  self    super   then    true
undef     unless  until   when    while   yield
```

Структура программы

- Программа на Ruby представляет собой текстовый файл, содержащий последовательность инструкций — команд и описаний. При запуске программного файла на исполнение интерпретатор последовательно читает файл и выполняет инструкции. В Ruby не требуется организовывать тело главной программы в виде специального программного модуля (подобно функции main() в языке Си), составляющие его команды просто записываются непосредственно в тексте файла программы. Поскольку программный файл обрабатывается интерпретатором последовательно, любые функции, методы, описания должны предшествовать в тексте программы их первому использованию.
- Программа может быть разделена на несколько файлов. В этом случае главный файл программы должен загрузить остальные файлы с помощью инструкции require или require_relative:

```
require 'module1'      # загрузка программного файла с именем 'module1.rb' либо библиотеки с именем 'module1'
require 'pkgs/package2' # загрузка программного файла 'package2' из подкаталога pkgs
```

Реализация

- Для Ruby существуют несколько реализаций: официальный интерпретатор, написанный на Си, JRuby — реализация для Java, интерпретатор для платформы .NET IronRuby, Rubinius — написанная в основном на Ruby и базирующаяся на идеях Smalltalk-80 VM, MagLev — другая базирующаяся на Smalltalk разработка от компании Gemstone, Blue Ruby — реализация Ruby для виртуальной машины ABAP, MacRuby — реализация для Mac OS с фокусом на максимальную интеграцию с возможностями операционной системы, mruby — реализация для встраивания в программы.
- Официальный интерпретатор портирован под большинство платформ, включая Unix, Microsoft Windows (в том числе Windows CE), DOS, Mac OS X, OS/2, Amiga, BeOS, Syllable, Acorn RISC OS и другие. Для Windows существует специализированный установщик RubyInstaller и есть возможность запуска под Cygwin для большей совместимости с Unix.

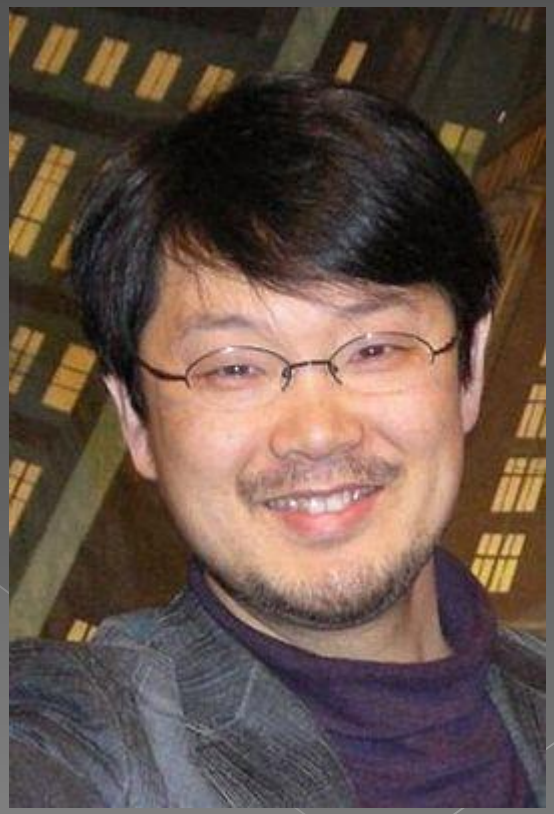
Использование

- Ruby используется в NASA, NOAA (национальная администрация по океану и атмосфере), Motorola и других крупных организациях[24]. Следующие программы используют Ruby как скриптовый язык для расширения возможностей программы или написаны на нём (частично или полностью).
- RPG Maker (RPG Maker XP) — RGSS (Ruby Game Scripting System).
- Amarok — аудиоплеер для среды KDE.
- SketchUp — система трёхмерного моделирования и визуализации.
- Inkscape — скрипты для обработки векторных изображений.
- Metasploit — среда для поиска и тестирования уязвимостей компьютерных систем.
- Chef, Puppet — системы управления конфигурациями.
- Redmine — система управления проектами, включающая багтрекер и вики-движок.
- XChat — кроссплатформенный IRC-клиент.
- Для KOffice разрабатывается проект Kross — механизм для поддержки скриптов, который включает Ruby.

Разработка мобильных приложений

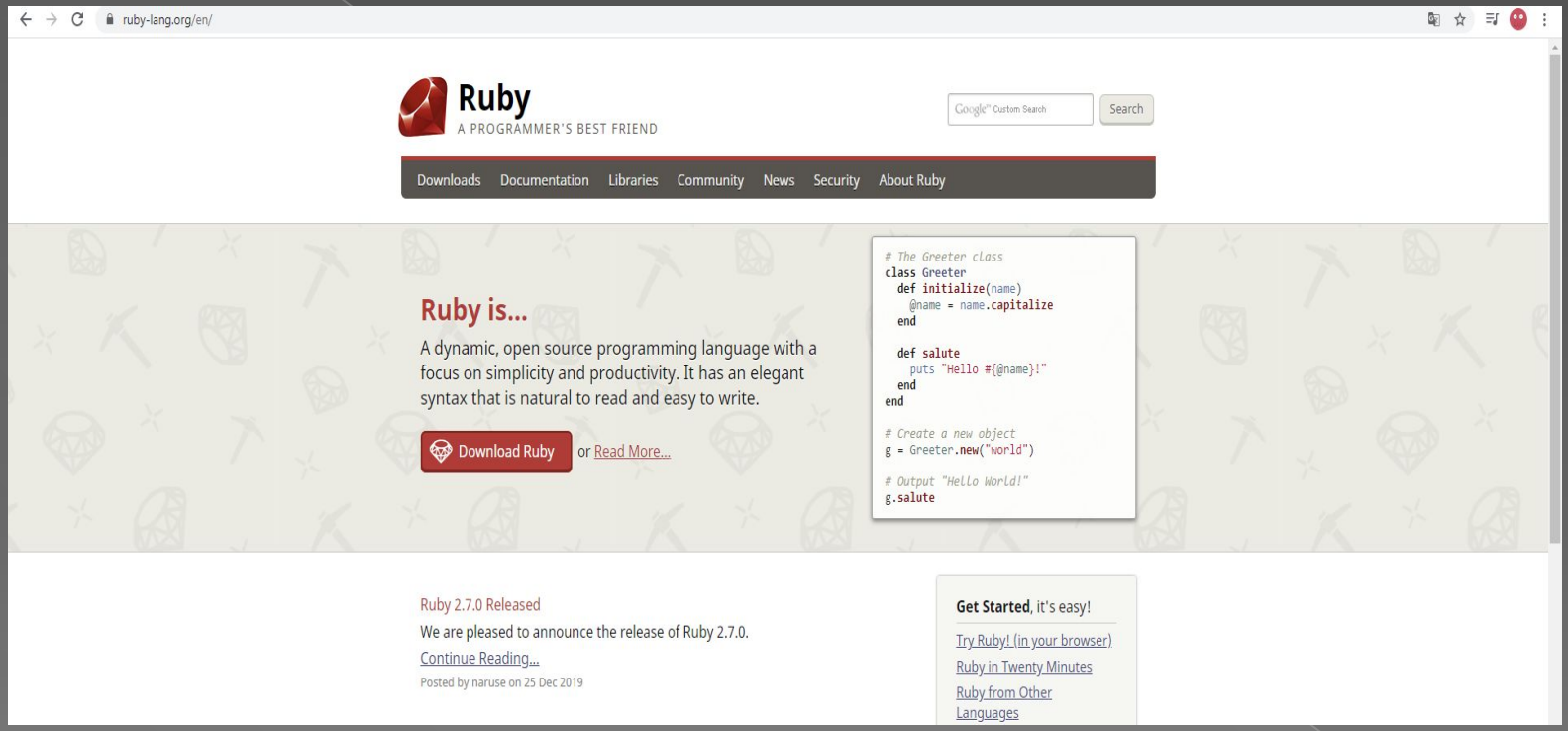
- Titanium Studio — среда разработки мобильных приложений на HTML5, CSS3, Javascript, Ruby, Rails, Python, PHP
- Ruboto — среда разработки Android приложений на Ruby
- RubyMotion — среда разработки iOS приложений на Ruby
- MobiRuby — инструмент разработки Android и iOS приложений на Ruby
- Rhodes — фреймворк для разработки Enteprise Mobility приложений для смартфонов и устройств Motorola

- **Юкихиро Мацумо́то** (яп. 松本行弘, чаще яп. まつもとゆきひろ, также известный как **Matz**, род. 14 апреля 1965) — японский разработчик свободного ПО, создатель языка программирования Ruby.
- В интервью «Japan Inc.» он говорил, что сам учился программировать ещё до окончания школы[2]. Он окончил университет города Цукуба, где он занимался исследованиями языков программирования и компиляторов. С 2006 года возглавляет отдел исследований и разработок Network Applied Communication Laboratory, японский системный интегратор свободного ПО.



- Класс языка объектно-ориентированный язык программирования
- Появился в1995
- Автор Мацумото, Юкихиро
- Расширение файлов.rb или .rbw
- Выпуск 2.7.0 (25 декабря 2019)
- Система типовстрогая, динамическая
- Основные реализацииRuby MRI (англ.), JRuby,Rubinius
- Испытал влияние Ада, Dylan, Perl, Python, Smalltalk, C++, Клу, Eiffel, Лисп, Бейсик и Lua
- ЛицензияRuby License[d], GNU GPL 2 и 2-пунктная лицензия BSD[d]
- Сайтruby-lang.org (англ.)
- ОС кроссплатформенность

Внешний вид сайта



Индекс TIOBE

Apr 2020	Apr 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.73%	+1.69%
2	2		C	16.72%	+2.64%
3	4	^	Python	9.31%	+1.15%
4	3	v	C++	6.78%	-2.06%
5	6	^	C#	4.74%	+1.23%
6	5	v	Visual Basic	4.72%	-1.07%
7	7		JavaScript	2.38%	-0.12%
8	9	^	PHP	2.37%	+0.13%
9	8	v	SQL	2.17%	-0.10%
10	16	^^	R	1.54%	+0.35%
11	19	^^	Swift	1.52%	+0.54%
12	18	^^	Go	1.36%	+0.35%
13	13		Ruby	1.25%	-0.02%

- **Crystal** — это объектно-ориентированный язык программирования общего назначения, спроектированный и разработанный Ary Borenszweig, Juan Wajnerman, Brian Cardiff и более 300 разработчиками. Имея Ruby-подобный синтаксис Crystal является компилируемым и статически типизированным языком, однако объявление типов переменных либо аргументов методов не является обязательным, так как компилятор применяет специальный алгоритм вывода типов. Язык находится в активной стадии разработки и распространяется как свободное и открытое программное обеспечение под лицензией Apache версии 2.0.

История

- Работа над новым языком программирования была начата в июне 2011 года в компании Manas. Разработчики поставили перед собой цель создать язык с элегантностью и продуктивностью Ruby и скоростью, эффективностью и безопасностью типов, присущих компилируемым языкам программирования. Первоначально разработка получила название Joy, однако позже была переименована в Crystal.
- Первый официальный релиз языка состоялся в июне 2014 года. Изначально компилятор языка был написан на Ruby, пока в 2013 году не был переписан на Crystal. В июле 2016 года Crystal вошёл в список индекса TIOBE.

Описание

- Несмотря на схожесть синтаксиса, Crystal намного эффективнее, чем Ruby, компилируется в машинный код, используя LLVM, жертвуя при этом динамическими аспектами языка. По результатам тестов Crystal показывает схожую с языком C производительность. Язык использует Boehm garbage collector, обладает системой макросов, поддерживает дженерики, а также перегрузку методов и операторов.
- В Crystal реализован интерфейс вызова функций из двоичных библиотек на языках C, C++ и пр., при этом синтаксис взаимодействия с такими библиотеками максимально упрощён, что позволяет легко создавать библиотеки-обёртки, а не писать весь код с нуля. Также Crystal поддерживает ассемблерные вставки и прямое обращение по указателям - это считается unsafe, но не запрещено, поскольку необходимо, в частности, и для взаимодействия с функциями из внешних библиотек.

Примеры

- **Hello, world!**
- Простейший вариант написания:
- puts "Hello World!"
- В объектно-ориентированном стиле:

```ruby
class Greeter
  def initialize(@name)
  end

  def salute
    "Hello #{@name}!"
  end
end

g = Greeter.new("world")
puts g.salute
```

HTTP Server

```
require "http/server"

server = HTTP::Server.new do |context|
  context.response.content_type = "text/plain"
  context.response.print "Hello world!"
end

server.bind_tcp 8080
puts "Listening on http://127.0.0.1:8080"
server.listen
```

Вывод типов и типы объединения

- Следующий код объявляет коллекцию (массив), состоящую из элементов различных типов данных. Crystal в данном случае автоматически создаёт тип объединения с индивидуальными типами данных элементов массива.

```
desired_things = [:unicorns, "butterflies", 1_000_000]
p typeof(desired_things.first) # typeof возвращает типы данных на момент компиляции, в данном случае (Int32 | String | Symbol)
p desired_things.first.class   # метод класса возвращает тип данных времени исполнения, в данном случае Symbol
```

Конкурентная модель

- Каналы (Channel) используются для коммуникации между файберами, которые инстанциируются при помощи команды spawn:

```
channel = Channel(Int32).new

spawn do
  puts "Перед первой отправки сообщения"
  channel.send(1)
  puts "После второй отправки сообщения"
  channel.send(2)
end

puts "Перед первым получением"
value = channel.receive
puts value # => 1

puts "Перед вторым получением"
value = channel.receive
puts value # => 2
```

Цели языка

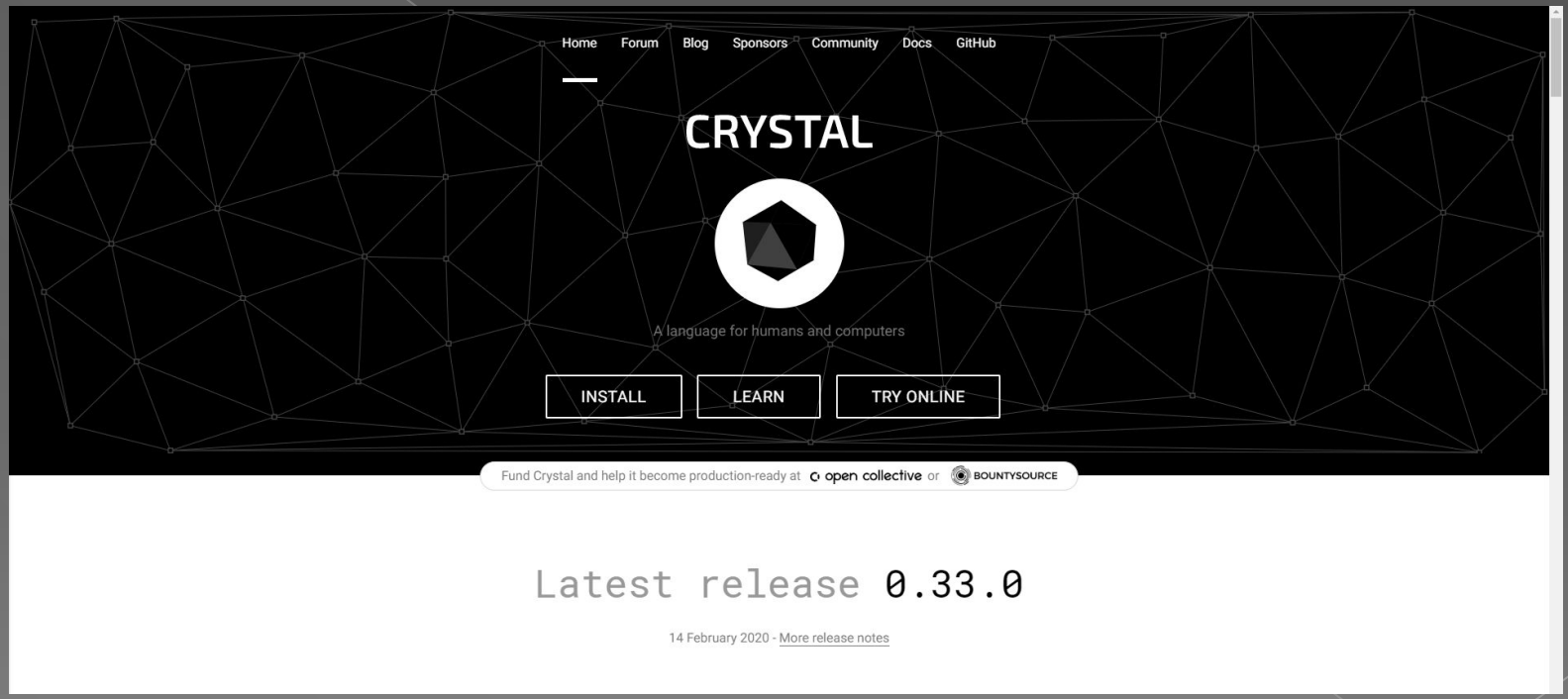
- Ruby-подобный синтаксис.
- Статическая типизация, но без указания типа переменных или аргументов метода.
- Возможность вызывать код на C путем написания биндингов внутри Crystal.
- Есть оценка времени компиляции и генерации кода, дабы избежать шаблонности кода.
- Эффективная компиляция в машинный код.

Арье Боренвейг

- Создавая язык, авторы задавались следующими целями:
  иметь максимально похожий на Ruby синтаксис
- иметь вывод типов
- вызывать код на C с помощью написания байндингов
- иметь возможность выполнения кода и кодогенерации на стадии компиляции
- компилировать все это в нативный код

- Семантика-мультипарадигмальный: объектно-ориентированный
- Класс языка-язык программирования и объектно-ориентированный язык программирования
- Появился 18 июня 2014
- Расширение файлов.cr
- Выпуск-0.31.1 (30 сентября 2019)
- Система типов Статическая
- Испытал влияниеRuby, Go, Rust, C#, Python
- Лицензия Apache License 2.0
- Сайт crystal-lang.orgПлатформаIA-32 (i386), x86-64
- ОС Linux, macOS (Homebrew)

Внешний вид сайта



Индекс TIOBE

50	Julia	0.17%

The Next 50 Programming Languages

The following list of languages denotes #51 to #100. Since the differences are relatively small, the programming languages are only listed (in alphabetical order).

- (Visual) FoxPro, ABC, ActionScript, Alice, Arc, ATLAS, Awk, bc, Bourne shell, C shell, CL (OS/400), Clojure, Common Lisp, Crystal, cT, Elixir, Forth, Hack, Icon, Inform, Io, J, Korn shell, Ladder Logic, LiveCode, Maple, Mercury, MQL4, NATURAL, Object Pascal, OCaml, OpenCL, OpenEdge ABL, Oz, PL/I, PostScript, Programming Without Coding Technology, Pure Data, Q, Red, Ring, S, Smalltalk, Solidity, SPARK, Tcl, Vala/Genie, Verilog, VHDL, Whitespace

Спасибо за внимание!