in production

Рычков Н.Н

Малютин М.С.
malyutin.mikhail@gmail.com
@MikhailMalyuti2

A long time ago, in organisation far, far away....


CEYLON
WARS

Episode I

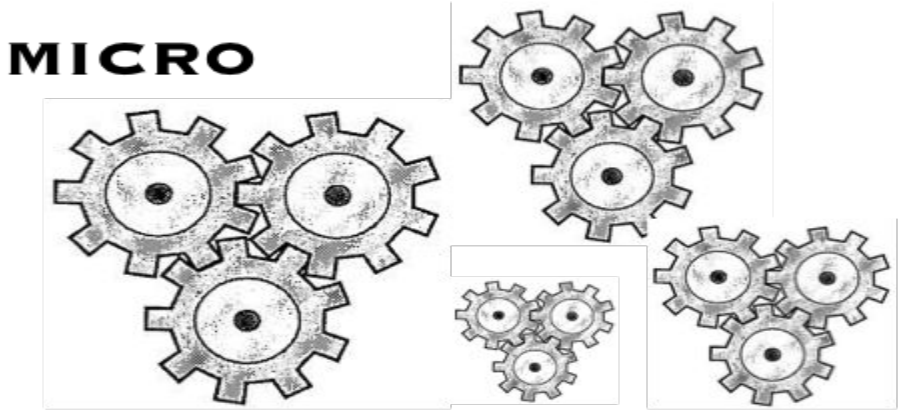
СКРЫТАЯ УГРОЗА

Episode I

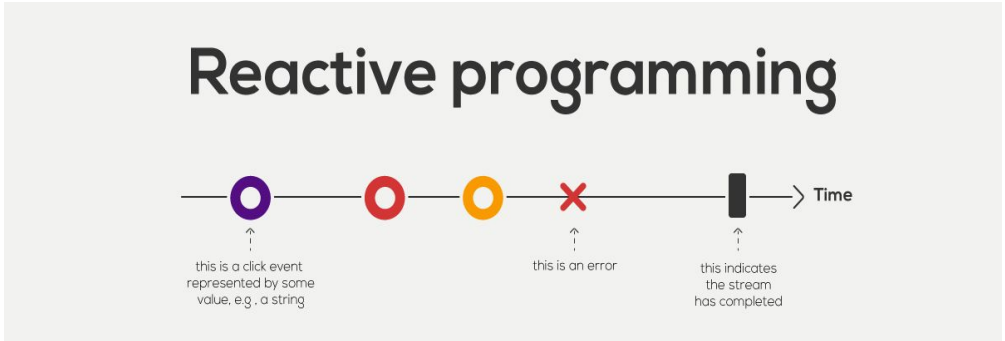
СКРЫТАЯ УГРОЗА

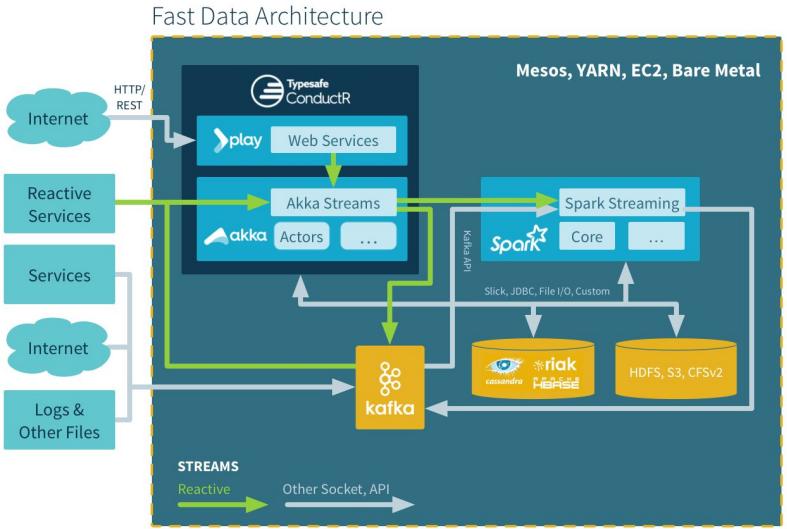
С течением времени стало понятно, что у Java есть недостатки. Многословность, медленное развитие языка, мутабельность по умолчанию, множество скрытых проблем в дизайне привело к потребности в языках, в которых эти недостатки исправлены. Одним из них стал Ceylon.

Новые департаменты - новые технологии...



VERT.X

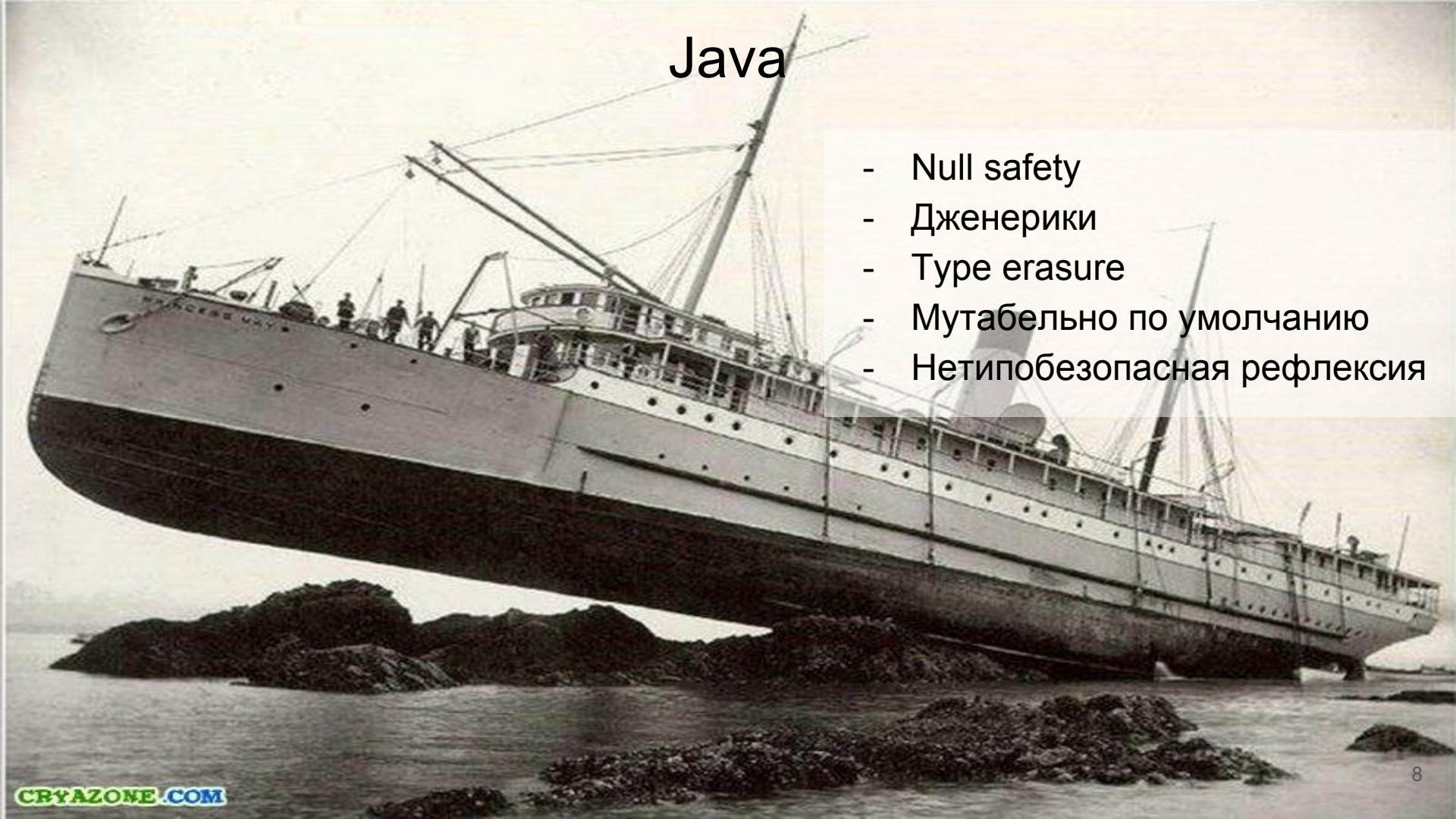




Java

- Громоздкий синтаксис
- Подводные камни
- Выкрутасы с DSL

Java



- Null safety
- Дженерики
- Type erasure
- Мутабельно по умолчанию
- Нетипобезопасная рефлексия

previous post New Road Map
next post Intersections and varian

Ceylon 1.0.0 is now available

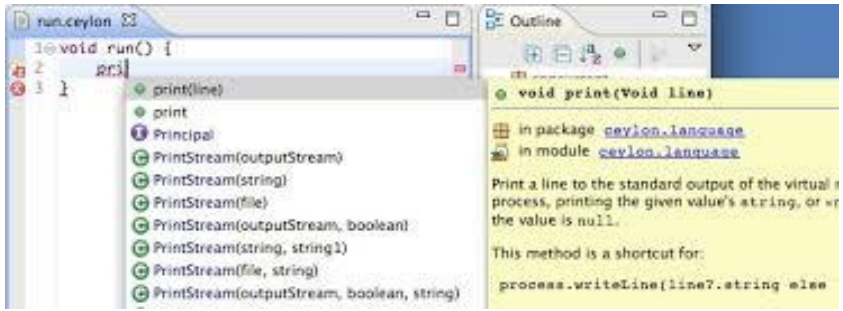
by Gavin King 12 Nov 2013 release news progress 1.0.0

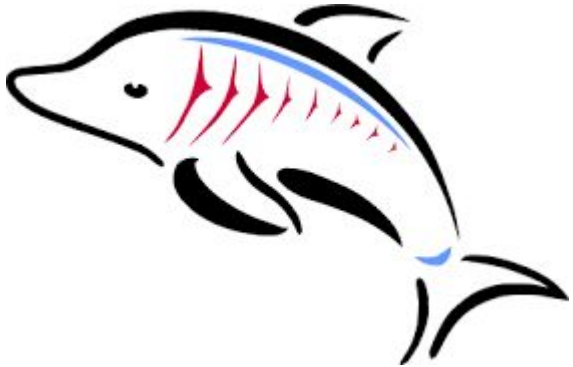
Today, we're proud to announce the first production release of the Ceylon language specification, compiler, and IDE. Ceylon 1.0 is a modern, modular, statically typed programming language for the Java and JavaScript virtual machines.

Ceylon enables the development of cross-platform modules that execute portably in both virtual machine environments. Alternatively, a Ceylon module may target one or the other platform, in which case it may interoperate with native code written for the platform.

In the box

Оптимистические факты Ceylon 1.0

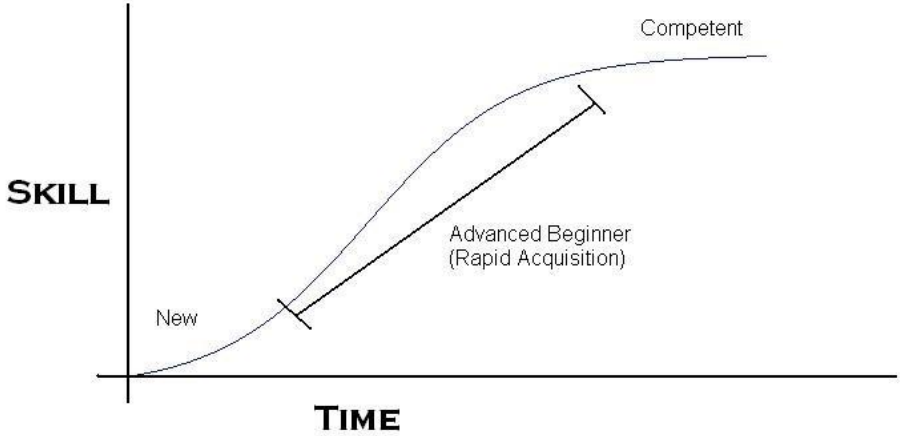




Оптимистические факты Ceylon 1.0

Оптимистические факты Ceylon 1.0

Ceylon бизнесу



Episode II

АТАКА КЛОНОВ

Episode II

АТАКА КЛОНОВ

Одни и те же задачи различным сотрудникам было удобно писать на совершенно разных языках. В конце концов эти клоны функциональности захотелось как то упорядочить и структурировать ...

Вавилонская башня языков

MATLAB REPL (Read-Eval-Print-Loop)

```
                    < M A T L A B (R) >
          Copyright 1984-2017 The MathWorks, Inc.
           R2017a (9.2.0.538062) 64-bit (glnxa64)
                      February 23, 2017


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> A = rand(3, 2)

A =

    0.8147    0.9134
    0.9058    0.6324
    0.1270    0.0975

>> A'

ans =

    0.8147    0.9058    0.1270
    0.9134    0.6324    0.0975

>>
```


Gavin King
Released into the Wild
ceylon

Created by Nikolay Rychkov 22 Aug 2011 03:09 Updated by Andrey Breslav 04 Sep 2011 10:54

~~KT-230~~ Merge with Ceylon

I propose to double forces to make one great language and not to devide comunity.

Comments 0 History Linked Issues Similar Issues VCS Changes 0

Issue has no comments

Project	Kotlin
Priority	Normal N
Type	Task
Target versions	No Target versions
State	As Designed
Assignee	Andrey Breslav
Subsystems	Language design
Affected versions	No Affected versions
Verified (Kotlin)	No

Boards 1

Episode III

МЕСТЬ БЕСОВ

Episode III

МЕСТЬ БЕСОВ

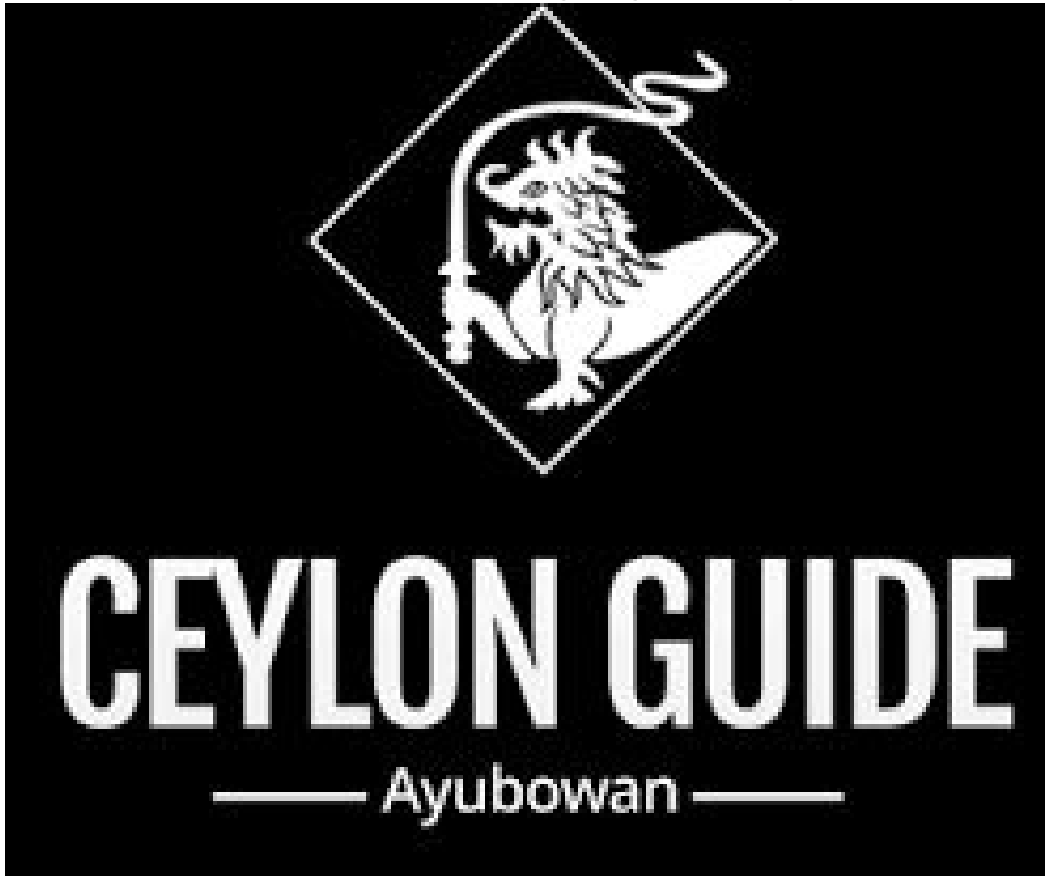
Благодаря достигнутым результатам, удалось получить карт-бланш на использование Ceylon.

Однако часть руководства не понимало новых веяний, возникли некоторые трудности …

Любовь математиков к фортрановскому стилю

C ← FOR COMMENT	STATEMENT NUMBER (1–5)	CONTINUATION (6)	FORTRAN STATEMENT (7–72)	IDENTIFICATION (73–80)
C			PROGRAM FOR FINDING THE LARGEST VALUE	
C		X	ATTAINED BY A SET OF NUMBERS	
			DIMENSION A(999)	
			FREQUENCY 30(2,1,10), 5(100)	
			READ 1, N,-(A(I), I = 1,N)	
	1		FORMAT (I3/(12F6.2))	
			BIGA = A(1)	
	5		DO 20 I = 2,N	
	30		IF (BIGA-A(I)) 10,20,20	
	10		BIGA = A(I)	
	20		CONTINUE	
			PRINT 2, N, BIGA	
	2		FORMAT (22H1THE LARGEST OF THESE I3, 12H NUMBERS IS F7.2)	
			STOP 77777	

Документация по Ceylon нацелена на продвинутую аудиторию



Трудности в нахождении взаимопонимании с программистами 1С

Саботаж начальника отдела разработки

Стеснение задавать вопросы в сообществе

Episode IV

НОВАЯ НАДЕЖДА

Episode IV

НОВАЯ НАДЕЖДА

Кроме административных аспектов, были еще и технические аспекты. С технической точки зрения Ceylon был весьма и весьма многообещающим языком, весьма выделяющимся на фоне конкурентов.

Преимущества

- умеет почти все, что умеет Kotlin (пока нет Data Class, async await, Extension Functions)

Дополнительно:

- модульность
- типизация
- кортежи
- comprehensions
- синтаксис

Union Types

```
function parse(String str) {
  return
    if(is Integer i=Integer.parse(str)) then i
    else if(is Float f=Float.parse(str)) then f
    else if(is Boolean b=Boolean.parse(str)) then b
    else if(str == "null") then null
    else str;
}
```

Union Types

```
function parse(String str) {
  return
    if(is Integer i=Integer.parse(str)) then i
    else if(is Float f=Float.parse(str)) then f
    else if(is Boolean b=Boolean.parse(str)) then b
    else if(str == "null") then null
    else str;
}
```

Union Types

```
function parse(String str) {
  return
    if(is Integer i=Integer.parse(str)) then i
    else if(is Float f=Float.parse(str)) then f
    else if(is Boolean b=Boolean.parse(str)) then b
    else if(str == "null") then null
    else str;
}
```

Union Types

```
function parse(String str) {
  return
    if(is Integer i=Integer.parse(str)) then i
    else if(is Float f=Float.parse(str)) then f
    else if(is Boolean b=Boolean.parse(str)) then b
    else if(str == "null") then null
    else str;
}
```

Union Types

```
function parse(String str) {
  return
    if(is Integer i=Integer.parse(str)) then i
    else if(is Float f=Float.parse(str)) then f
    else if(is Boolean b=Boolean.parse(str)) then b
    else if(str == "null") then null
    else str;
}
```

Union Types

```
function parse(String str) {
  return
    if(is Integer i=Integer.parse(str)) then i
    else if(is Float f=Float.parse(str)) then f
    else if(is Boolean b=Boolean.parse(str)) then b
    else if(str == "null") then null
    else str;
}
```

Union Types

```
Integer|Float|Boolean|String|Null parse(String str) {
  return
    if(is Integer i=Integer.parse(str)) then i
    else if(is Float f=Float.parse(str)) then f
    else if(is Boolean b=Boolean.parse(str)) then b
    else if(str == "null") then null
    else str;
}
```

Intersection Types

```
interface CanSwim {
   shared void swim() => print("I am swimming");
}
interface CanFly {
   shared void fly() => print("I am flying");
}
```

Intersection Types

```
interface CanSwim {
   shared void swim() => print("I am swimming");
}
interface CanFly {
   shared void fly() => print("I am flying");
}
```

Intersection Types

```
interface CanSwim {
    shared void swim() => print("I am swimming");
}
interface CanFly {
    shared void fly() => print("I am flying");
}
```

Intersection Types

```
class Duck() satisfies CanSwim & CanFly {}

class Fish() satisfies CanSwim {}

void f(CanFly & CanSwim arg) {
   arg.fly();
   arg.swim();
}
f(Duck()); //OK Duck can swim and fly
f(Fish());//ERROR = fish can swim only
```

Intersection Types

```
class Duck() satisfies CanSwim & CanFly {}

class Fish() satisfies CanSwim {}

void f(CanFly & CanSwim arg) {
   arg.fly();
   arg.swim();
}
f(Duck()); //OK Duck can swim and fly
f(Fish());//ERROR = fish can swim only
```

Intersection Types

```
class Duck() satisfies CanSwim & CanFly {}

class Fish() satisfies CanSwim {}

void f(CanFly & CanSwim arg) {
   arg.fly();
   arg.swim();
}
f(Duck()); //OK Duck can swim and fly
f(Fish());//ERROR = fish can swim only
```

Intersection Types

```
class Duck() satisfies CanSwim & CanFly {}

class Fish() satisfies CanSwim {}

void f(CanFly & CanSwim arg) {
   arg.fly();
   arg.swim();
}
f(Duck()); //OK Duck can swim and fly
f(Fish());//ERROR = fish can swim only
```

Intersection Types

```
class Duck() satisfies CanSwim & CanFly {}

class Fish() satisfies CanSwim {}

void f(CanFly & CanSwim arg) {
   arg.fly();
   arg.swim();
}
f(Duck()); //OK Duck can swim and fly
f(Fish());//ERROR = fish can swim only
```

Intersection Types

```
class Duck() satisfies CanSwim & CanFly {}

class Fish() satisfies CanSwim {}

void f(CanFly & CanSwim arg) {
   arg.fly();
   arg.swim();
}
f(Duck()); //OK Duck can swim and fly
f(Fish());//ERROR = fish can swim only
```

Enum Types

```
abstract class Point()
       of Polar | Cartesian {}

class Cartesian(shared Float x, shared Float y)
    extends Point() {}
class Polar(shared Float radius, shared Float angle)
    extends Point() {}
```

Enum Types

```
abstract class Point()
        of Polar | Cartesian {}

class Cartesian(shared Float x, shared Float y)
    extends Point() {}
class Polar(shared Float radius, shared Float angle)
    extends Point() {}
```

Enum Types

```
abstract class Point()
        of Polar | Cartesian {}

class Cartesian(shared Float x, shared Float y)
    extends Point() {}
class Polar(shared Float radius, shared Float angle)
    extends Point() {}
```

Enum Types

```
abstract class Point()
        of Polar | Cartesian {}

class Cartesian(shared Float x, shared Float y)
    extends Point() {}
class Polar(shared Float radius, shared Float angle)
    extends Point() {}
```

Enum Types

```
void printPoint(Point point) {
   switch (point)
   case (is Polar) {
       print("r = " + point.radius.string);
       print("theta = " + point.angle.string);
   }
   case (is Cartesian) {
       print("x = " + point.x.string);
       print("y = " + point.y.string);
   }
}
```

Enum Types

```
void printPoint(Point point) {
    switch (point)
    case (is Polar) {
        print("r = " + point.radius.string);
        print("theta = " + point.angle.string);
    }
    case (is Cartesian) {
        print("x = " + point.x.string);
        print("y = " + point.y.string);
    }
}
```

Алиасы типов

```
interface People => Set<Person>;

alias Num => Float|Integer;

alias ListOrMap<Element> =>
    List<Element>|Map<Integer,Element>;

class People({Person*} people) => ArrayList<Person>(people);
```

Алиасы типов

```
interface People => Set<Person>;

alias Num => Float|Integer;

alias ListOrMap<Element> =>
    List<Element>|Map<Integer,Element>;

class People({Person*} people) => ArrayList<Person>(people);
```

Алиасы типов

```
interface People => Set<Person>;

alias Num => Float|Integer;

alias ListOrMap<Element> =>
    List<Element>|Map<Integer,Element>;

class People({Person*} people) => ArrayList<Person>(people);
```

Алиасы типов

```
interface People => Set<Person>;

alias Num => Float|Integer;

alias ListOrMap<Element> =>
    List<Element>|Map<Integer,Element>;

class People({Person*} people) => ArrayList<Person>(people);
```

Алиасы типов

```
interface People => Set<Person>;

alias Num => Float|Integer;

alias ListOrMap<Element> =>
    List<Element>|Map<Integer,Element>;

class People({Person*} people) => ArrayList<Person>(people);
```

Кортежи

```
value t = ["Str", 1, 2.3]; // [String, Integer, Float]

value [str, intVar, floatType] = t;

value [first, *rest] = t;

value [i, f] = rest;
```

Кортежи

```
value t = ["Str", 1, 2.3]; // [String, Integer, Float]

value [str, intVar, floatType] = t;

value [first, *rest] = t;

value [i, f] = rest;
```

Кортежи

```
value t = ["Str", 1, 2.3]; // [String, Integer, Float]

value [str, intVar, floatType] = t;

value [first, *rest] = t;

value [i, f] = rest;
```

Кортежи

```
value t = ["Str", 1, 2.3]; // [String, Integer, Float]

value [str, intVar, floatType] = t;

value [first, *rest] = t;

value [i, f] = rest;
```

Кортежи

```
value t = ["Str", 1, 2.3]; // [String, Integer, Float]

value [str, intVar, floatType] = t;

value [first, *rest] = t;

value [i, f] = rest;
```

Конструирование коллекций

$$\{\, x^2 \mid x \in \{1 \ldots 25\},\ \neg(x \vdots 3),\ x = 1 + 2k,\ k \in \mathbb{N} \,\}$$

```
val res = (1..25 step 2).filter{x -> x % 3 != 0 }.map{x -> x*x}

value res = [for(x in (1:25).by(2) ) if(x % 3 != 0) x*x];
value res = {for(x in (1:25).by(2) ) if(x % 3 != 0) x*x};

value m = HashMap { for (i in 1..10) i -> i + 1 };
```

Конструирование коллекций

$$\{\, x^2 \mid x \in \{1 \ldots 25\},\ \neg(x \vdots 3),\ x = 1 + 2k,\ k \in \mathbb{N} \,\}$$

```
val res = (1..25 step 2).filter{x -> x % 3 != 0 }.map{x -> x*x}

value res = [for(x in (1:25).by(2) ) if(x % 3 != 0) x*x];
value res = {for(x in (1:25).by(2) ) if(x % 3 != 0) x*x};

value m = HashMap { for (i in 1..10) i -> i + 1 };
```

Конструирование коллекций

$$\{\, x^2 \mid x \in \{1 \ldots 25\},\ \neg(x \vdots 3),\ x = 1 + 2k,\ k \in \mathbb{N} \,\}$$

```
val res = (1..25 step 2).filter{x -> x % 3 != 0 }.map{x -> x*x}

value res = [for(x in (1:25).by(2) ) if(x % 3 != 0) x*x];
value res = {for(x in (1:25).by(2) ) if(x % 3 != 0) x*x};

value m = HashMap { for (i in 1..10) i -> i + 1 };
```

Конструирование коллекций

$$\{\, x^2 \mid x \in \{1 \ldots 25\},\ \neg(x \vdots 3),\ x = 1 + 2k,\ k \in \mathbb{N} \,\}$$

```
val res = (1..25 step 2).filter{x -> x % 3 != 0 }.map{x -> x*x}

value res = [for(x in (1:25).by(2) ) if(x % 3 != 0) x*x];
value res = {for(x in (1:25).by(2) ) if(x % 3 != 0) x*x};

value m = HashMap { for (i in 1..10) i -> i + 1 };
```

Конструирование коллекций

$$\{\, x^2 \mid x \in \{1 \ldots 25\},\ \neg(x \vdots 3),\ x = 1 + 2k,\ k \in \mathbb{N} \,\}$$

```
val res = (1..25 step 2).filter{x -> x % 3 != 0 }.map{x -> x*x}

value res = [for(x in (1:25).by(2) ) if(x % 3 != 0) x*x];
value res = {for(x in (1:25).by(2) ) if(x % 3 != 0) x*x};

value m = HashMap { for (i in 1..10) i -> i + 1 };
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Непустые стримы

```
Integer max({Integer+} values) {
    return if (is {Integer+} rest = values.rest)
           then Math.max(values.first, max(rest))
           else values.first;
}

max({5}); //OK, 5

max({5, 7}); //OK, 7

max({});//COMPILE ERROR
```

Метамодель

```
shared interface DataCollector {}

service(`interface DataCollector`)
shared class DataCollectorUserV1()
    satisfies DataCollector {}

shared void example() {
   {DataCollector*} allDataCollectorsImpls =
       `module`.findServiceProviders(`DataCollector`);
}
```

Метамодель

```
shared interface DataCollector {}

service(`interface DataCollector`)
shared class DataCollectorUserV1()
    satisfies DataCollector {}

shared void example() {
   {DataCollector*} allDataCollectorsImpls =
       `module`.findServiceProviders(`DataCollector`);
}
```

Метамодель

```
shared interface DataCollector {}

service(`interface DataCollector`)
shared class DataCollectorUserV1()
    satisfies DataCollector {}

shared void example() {
   {DataCollector*} allDataCollectorsImpls =
       `module`.findServiceProviders(`DataCollector`);
}
```

Метамодель

```
shared interface DataCollector {}

service(`interface DataCollector`)
shared class DataCollectorUserV1()
    satisfies DataCollector {}

shared void example() {
   {DataCollector*} allDataCollectorsImpls =
       `module`.findServiceProviders(`DataCollector`);
}
```

Общий дизайн языка

```
[] unit = [];
[Integer] singleton = [1];
[Float,Float] pair = [1.0, 2.0];
[Float,Float,String] triple = [0.0, 0.0, "origin"];
[Integer*] cubes = [ for (x in 1..100) x^3 ];

val unit: Unit = ()
val singleton: Tuple1[Long] = new Tuple1(1)
val pair: (Double,Double) = (1.0, 2.0)
val triple: (Double,Double,String) = (0.0, 0.0, "origin")
val cubes: List[Integer] = ...
```

Общий дизайн языка

```
[] unit = [];
[Integer] singleton = [1];
[Float,Float] pair = [1.0, 2.0];
[Float,Float,String] triple = [0.0, 0.0, "origin"];
[Integer*] cubes = [ for (x in 1..100) x^3 ];

val unit: Unit = ()
val singleton: Tuple1[Long] = new Tuple1(1)
val pair: (Double,Double) = (1.0, 2.0)
val triple: (Double,Double,String) = (0.0, 0.0, "origin")
val cubes: List[Integer] = ...
```

Общий дизайн языка

```
[] unit = [];
[Integer] singleton = [1];
[Float,Float] pair = [1.0, 2.0];
[Float,Float,String] triple = [0.0, 0.0, "origin"];
[Integer*] cubes = [ for (x in 1..100) x^3 ];

val unit: Unit = ()
val singleton: Tuple1[Long] = new Tuple1(1)
val pair: (Double,Double) = (1.0, 2.0)
val triple: (Double,Double,String) = (0.0, 0.0, "origin")
val cubes: List[Integer] = ...
```

Общий дизайн языка

```
[] unit = [];
[Integer] singleton = [1];
[Float,Float] pair = [1.0, 2.0];
[Float,Float,String] triple = [0.0, 0.0, "origin"];
[Integer*] cubes = [ for (x in 1..100) x^3 ];

val unit: Unit = ()
val singleton: Tuple1[Long] = new Tuple1(1)
val pair: (Double,Double) = (1.0, 2.0)
val triple: (Double,Double,String) = (0.0, 0.0, "origin")
val cubes: List[Integer] = ...
```

Общий дизайн языка

```
[] unit = [];
[Integer] singleton = [1];
[Float,Float] pair = [1.0, 2.0];
[Float,Float,String] triple = [0.0, 0.0, "origin"];
[Integer*] cubes = [ for (x in 1..100) x^3 ];

val unit: Unit = ()
val singleton: Tuple1[Long] = new Tuple1(1)
val pair: (Double,Double) = (1.0, 2.0)
val triple: (Double,Double,String) = (0.0, 0.0, "origin")
val cubes: List[Integer] = ...
```

Общий дизайн языка

```
[] unit = [];
[Integer] singleton = [1];
[Float,Float] pair = [1.0, 2.0];
[Float,Float,String] triple = [0.0, 0.0, "origin"];
[Integer*] cubes = [ for (x in 1..100) x^3 ];

val unit: Unit = ()
val singleton: Tuple1[Long] = new Tuple1(1)
val pair: (Double,Double) = (1.0, 2.0)
val triple: (Double,Double,String) = (0.0, 0.0, "origin")
val cubes: List[Integer] = ...
```

Общий дизайн языка

```
[] unit = [];
[Integer] singleton = [1];
[Float,Float] pair = [1.0, 2.0];
[Float,Float,String] triple = [0.0, 0.0, "origin"];
[Integer*] cubes = [ for (x in 1..100) x^3 ];
```

```
val unit: Unit = ()
val singleton: Tuple1[Long] = new Tuple1(1)
val pair: (Double,Double) = (1.0, 2.0)
val triple: (Double,Double,String) = (0.0, 0.0, "origin")
val cubes: List[Integer] = ...
```

Мультипарадигменность

```
"All lines of text in the given file."
shared String[] lines(File file) {
   try (reader = file.Reader()) {
       return { reader.readLine() }
           .cycled
           .takeWhile((line) => line exists)
           .coalesced
           .sequence();
   }
}
```

Мультипарадигменность

```
"All lines of text in the given file."
shared String[] lines(File file) {
   try (reader = file.Reader()) {
       return { reader.readLine() }
           .cycled
           .takeWhile((line) => line exists)
           .coalesced
           .sequence();
   }
}
```

Мультипарадигменность

```
"All lines of text in the given file."
shared String[] lines(File file) {
   try (reader = file.Reader()) {
       return { reader.readLine() }
           .cycled
           .takeWhile((line) => line exists)
           .coalesced
           .sequence();
   }
}
```

Мультипарадигменность

```
"All lines of text in the given file."
shared String[] lines(File file) {
   try (reader = file.Reader()) {
       return { reader.readLine() }
           .cycled
           .takeWhile((line) => line exists)
           .coalesced
           .sequence();
   }
}
```

Мультипарадигменность

```
"All lines of text in the given file."
shared String[] lines(File file) {
   try (reader = file.Reader()) {
       return { reader.readLine() }
           .cycled
           .takeWhile((line) => line exists)
           .coalesced
           .sequence();
   }
}
```

Мультипарадигменность

```
"All lines of text in the given file."
shared String[] lines(File file) {
   try (reader = file.Reader()) {
       return { reader.readLine() }
           .cycled
           .takeWhile((line) => line exists)
           .coalesced
           .sequence();
   }
}
```

Мультипарадигменность

```
"All lines of text in the given file."
shared String[] lines(File file) {
   try (reader = file.Reader()) {
       return { reader.readLine() }
           .cycled
           .takeWhile((line) => line exists)
           .coalesced
           .sequence();
   }
}
```

Episode V

БАГИ НАНОСЯТ ОТВЕТНЫЙ УДАР

Episode V

БАГИ НАНОСЯТ ОТВЕТНЫЙ УДАР

Но в случае с Ceylon не все было безоблачно. К сожалению, баги не дремлют, и, кроме достоинств, пришлось столкнуться и с некоторыми проблемами, о которых тоже следует рассказать.


- HashMap HashSet 1.2.0
- json parse 1.3.0
- group

HashMap (Ceylon 1.2.0)

```
value map = HashMap<String, Integer>();
map.put("Hello", 2);
//..
value found = map.get("Hello"); //found = null
value set = HashSet<Integer>();
set.add(5);
//..
value foundInSet = set.contains(5); //false
```

HashMap (Ceylon 1.2.0)

```
value map = HashMap<String, Integer>();
map.put("Hello", 2);
//..
value found = map.get("Hello"); //found = null
value set = HashSet<Integer>();
set.add(5);
//..
value foundInSet = set.contains(5); //false
```

HashMap (Ceylon 1.2.0)

```
value map = HashMap<String, Integer>();
map.put("Hello", 2);
//..
value found = map.get("Hello"); //found = null
value set = HashSet<Integer>();
set.add(5);
//..
value foundInSet = set.contains(5); //false
```

HashMap (Ceylon 1.2.0)

```
value map = HashMap<String, Integer>();
map.put("Hello", 2);
//..
value found = map.get("Hello"); //found = null
value set = HashSet<Integer>();
set.add(5);
//..
value foundInSet = set.contains(5); //false
```

HashMap (Ceylon 1.2.0)

```
value map = HashMap<String, Integer>();
map.put("Hello", 2);
//..
value found = map.get("Hello"); //found = null
value set = HashSet<Integer>();
set.add(5);
//..
value foundInSet = set.contains(5); //false
```

Group (1.2.2)

```
value bigArray = [for (i in 1..1000000) i];
value afterGroup =
    bigArray.group( (element) => "gr" ).get("gr");
assert (exists afterGroup);
variable i = 0;
afterGroup.each( (item) {
   if (i % 100 == 0) {
       print("``i`` ``now()``");
   }
   ++i;
});
```

Group (1.2.2)

```
value bigArray = [for (i in 1..1000000) i];
value afterGroup =
    bigArray.group( (element) => "gr" ).get("gr");
assert (exists afterGroup);
variable i = 0;
afterGroup.each( (item) {
   if (i % 100 == 0) {
       print("``i`` ``now()``");
   }
   ++i;
});
```

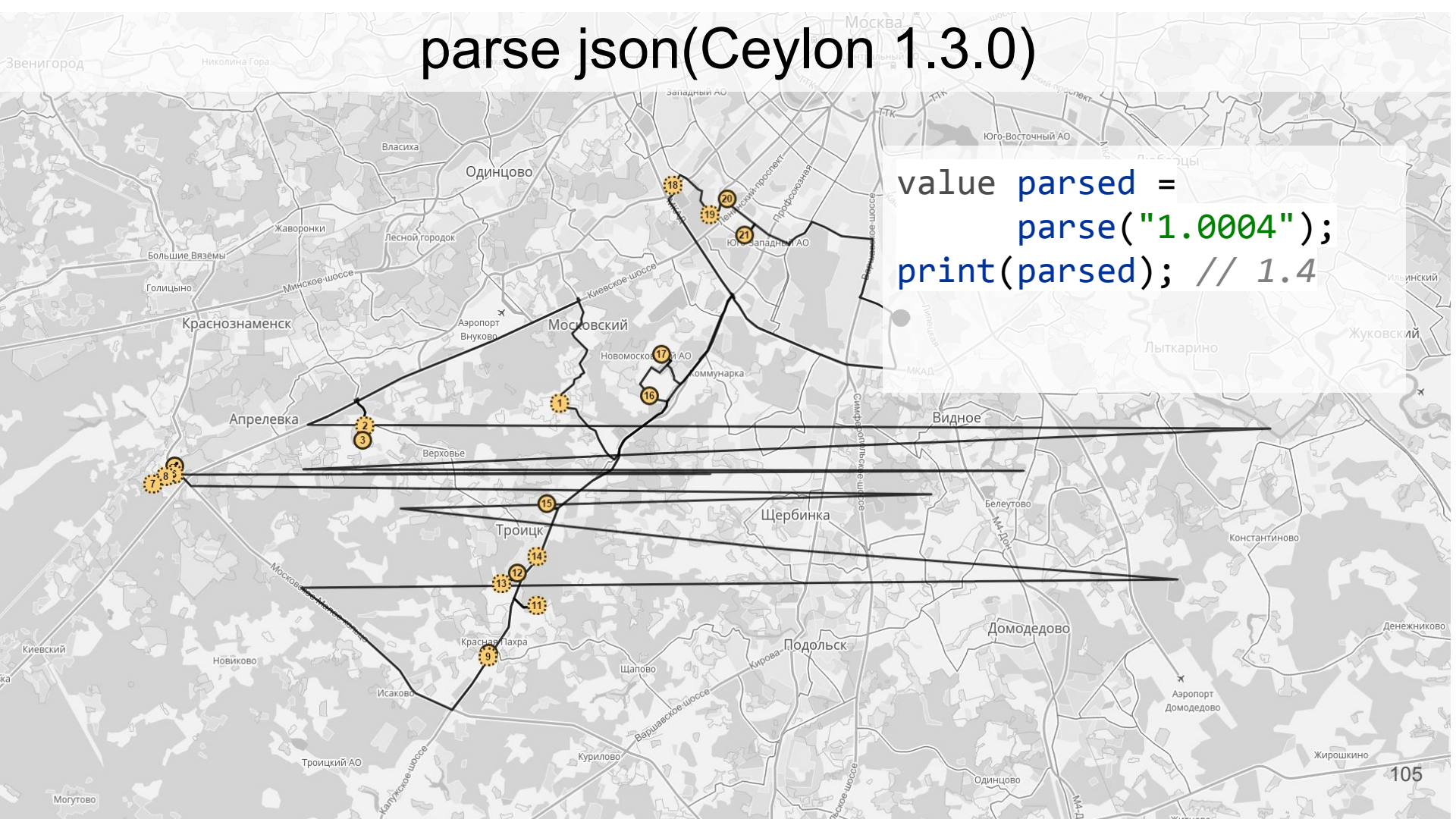
Group (1.2.2)

```
value bigArray = [for (i in 1..1000000) i];
value afterGroup =
    bigArray.group( (element) => "gr" ).get("gr");
assert (exists afterGroup);
variable i = 0;
afterGroup.each( (item) {
   if (i % 100 == 0) {
       print("``i`` ``now()``");
   }
   ++i;
});
```

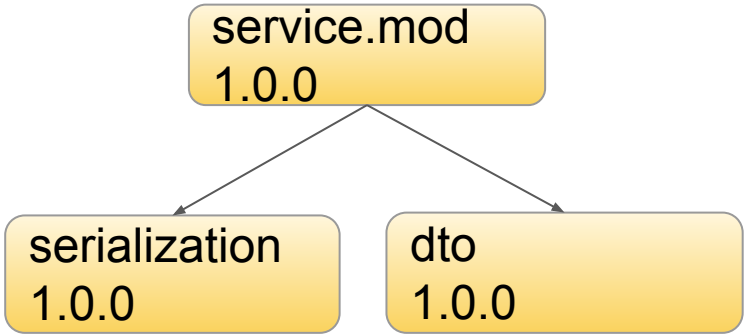
Group (1.2.2)

```
value bigArray = [for (i in 1..1000000) i];
value afterGroup =
    bigArray.group( (element) => "gr" ).get("gr");
assert (exists afterGroup);
variable i = 0;
afterGroup.each( (item) {
   if (i % 100 == 0) {
       print("``i`` ``now()``");
   }
   ++i;
});
```

parse json(Ceylon 1.3.0)



Сериализация



- защита модулей препятствует десериализации
- десериализация работает, если запускать с --flat-classpath

Сериализация

```
shared class Test(
  shared String a = "Hello",
  shared Integer i = 100,
  shared List<String> list =
    ArrayList<String>{"TST"}
) {}
```

Episode VI

ВОЗВРАЩЕНИЕ CEYLON

Episode VI

ВОЗВРАЩЕНИЕ CEYLON

Несмотря на проблемы, Ceylon весьма неплохо себя зарекомендовал. Было достаточно большое количество типичных задач, для которых было важно пользоваться DRY принципом, максимально компактно в DSL стиле описывать задачи предметной области. И Ceylon весьма неплохо себя зарекомендовал как язык для построения DSL и библиотек.

Совместимость с Java

```
CompletableFuture.supplyAsync(() => true);

CompletableFuture.supplyAsync(
    object satisfies Supplier<Object> {
        shared actual Object get() {
            return true;
        }
    });

CompletableFuture.supplyAsync(toJavaSupplier(() => true));
```

Совместимость с Java

```
javaArrayList.stream().forEach(print);


for (entry in jhashMap.entrySet()) {
   print(entry.key);
}
```

Java 9 - пока не работает

```
org.jboss.modules.ModuleNotFoundException: javax.xml:7
```

Совместимость с Java

```
shared void ceylonPrintAll([String*] data) {
   print(data);
}
```

ceylon compile --verbose=code

Совместимость с Java

```
public final class ceylonPrintAll_ {

    private ceylonPrintAll_() {
    }

    public static void ceylonPrintAll(
            final .ceylon.language.Sequential<
                ? extends .ceylon.language.String> data) {
        .ceylon.language.print_.print(data);
    }
}
```

Совместимость с Java

```
Sequence<? extends String> tuple =
        (Sequence<? extends String>)
Tuple.instance(String.$TypeDescriptor$, new Object[]{
                String.instance("Hello world")
        });
ceylonPrintAll_.ceylonPrintAll(tuple);


ceylonPrintAll(toCeylonStringTyple("Hello world"));
```

Микросервисы

- написать hello world микросервис
- описывать Html средствами языка
- важна строгая типизация
- хотим писать как можно быстрее с нуля
- хотим использовать свои наработки

module.ceylon

```
module helloService "1.0.0" {
    import ceylon.http.server "1.3.3";
    import ceylon.html "1.3.3";
    import ourcompany.utils "1.0.0";
}
```

module.ceylon

```
module helloService "1.0.0" {
    import ceylon.http.server "1.3.3";
    import ceylon.html "1.3.3";
    import ourcompany.utils "1.0.0";
}
```

module.ceylon

```
module helloService "1.0.0" {
    import ceylon.http.server "1.3.3";
    import ceylon.html "1.3.3";
    import ourcompany.utils "1.0.0";
}
```

module.ceylon

```
module helloService "1.0.0" {
    import ceylon.http.server "1.3.3";
    import ceylon.html "1.3.3";
    import ourcompany.utils "1.0.0";
}
```

module.ceylon

```
module helloService "1.0.0" {
    import ceylon.http.server "1.3.3";
    import ceylon.html "1.3.3";
    import ourcompany.utils "1.0.0";
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

run.ceylon

```
shared void run() {
    value server = newServer {
        service = (Request request, Response response) {
            Endpoint { path = equals("/hello");
                sendOk(response, html, Html {
                    Body {
                        H1 {"Hello world"}
                    }
                }.string));
            };
        }
    };
    server.start(SocketAddress(getMyIp(),getPort()));
}
```

Распределенная задача

- миллиард заявок
- сгруппировать по году, месяцу
- подсчитать суммарные расходы
- вычисление года заявки - обычная Java функция. В ней есть if
- расходы считаем тоже формулой
- при расчете не используем пред агрегацию, индексы, все грубой силой
- уложиться за 3 секунды

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices) -> GroupInfo(20, indexToYearFunc,
                                         yearToIndexFunc(invoices)),
        monthGroup(invoices)
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```

DSL для кластерных вычислений

```
executePreprocessedGroupQuery(
   (Storage<Invoice> invoices)
   => [noFilteringFunc<Invoice>(),
       [yearGroup(invoices) -> GroupInfo(20, indexToYearFunc,
                                         yearToIndexFunc(invoices)),
        monthGroup(invoices) -> GroupInfo(12, monthFromIndexFunc,
                                          monthToIndexFunc(invoices))
       ],
       [sumAggregator -> expenses(invoices)]
      ], distributedExecutorService);
```


Episode VII
ПРОБУЖДЕНИЕ CEYLON

Episode VII

ПРОБУЖДЕНИЕ CEYLON

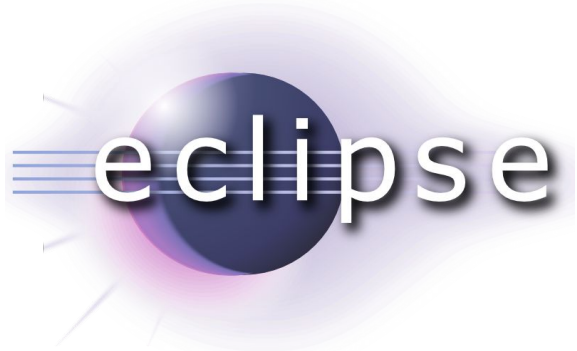
Итого, с точки зрения кода и читаемости Ceylon весьма себя неплохо показал на практике.

Нужно было создавать современный стек полностью с нуля, поднимать CI и CD.

В начале

- Нет CI
- Нет взаимодействия между модулями
- Ручной деплой

IDE



- с самого начала
- несколько больше возможностей

- начиная с версии 1.3.0
- меньше функционала
- JetBrains часто меняет API

« 1 2 3 4 ... 13 »

List of modules (page 2/13 of 241 modules)

ceylon.http.server	Ceylon Http Server Platform Module	1.3.3	JVM		quintesse
ceylon.interop.browser	Ceylon Browser Interop Platform Module	1.3.3	JavaScript		quintesse
ceylon.interop.java	Ceylon / Java Interoperability Platform Module	1.3.3	JVM		quintesse
ceylon.interop.persistence	Ceylon Persistence Interop Platform Module	1.3.3	JVM		quintesse
ceylon.interop.spring	Ceylon Spring Interop Platform Module	1.3.3	JVM		quintesse
ceylon.io	Ceylon IO Platform Module	1.3.3	JVM		FroMage
ceylon.json	Ceylon JSON Platform Module	1.3.3	JVM JavaScript	★★★★★	FroMage
ceylon.language	Ceylon Language Module	1.3.3	JVM JavaScript		FroMage
ceylon.locale	Ceylon Locale Platform Module	1.3.3	JVM JavaScript		FroMage
ceylon.logging	Ceylon Logging Platform Module	1.3.3	JVM JavaScript		FroMage
ceylon.math	Ceylon Math Platform Module	1.3.3	JVM		tombentley
ceylon.net	Ceylon Network Platform Module	1.2.2	JVM		FroMage
ceylon.numeric	Ceylon Numeric Platform Module	1.3.3	JVM JavaScript		quintesse
ceylon.openshift	Ceylon OpenShift Platform Module	1.3.3	JVM		FroMage

Module info

Name	**FroMage** / ceylon.io Ceylon IO Platform Module
Description	This module allows you to read and write to streams, such as files, sockets and pipes. See the `ceylon.io` package for usage examples. Standard platform modules belonging to the Ceylon SDK
Category	SDK The Ceylon SDK
Last Published	Aug 21, 2017
Stats	**Downloads (JVM):** 22918 **Downloads (JS):** 0 **Source downloads:** 4626
Module links	Home Code repository Issue tracker (118 open issues) Imported By Browse

List of published versions

1.3.3	JVM JVM: 1.2.x, 1.3.x (latest)	Published Aug 21, 2017	Browse	View docs

This module allows you to read and write to streams, such as files, sockets and pipes.

See the `ceylon.io` package for usage examples.

Platform: Java

By: Stéphane Épardaud

License: Apache Software License

Packages	
ceylon.io	This package lets you create `FileDescriptor` objects, which represent open streams, such as files, sockets, or pi
ceylon.io.readers	

Dependencies

package ceylon.io

Values Functions Interfaces Classes

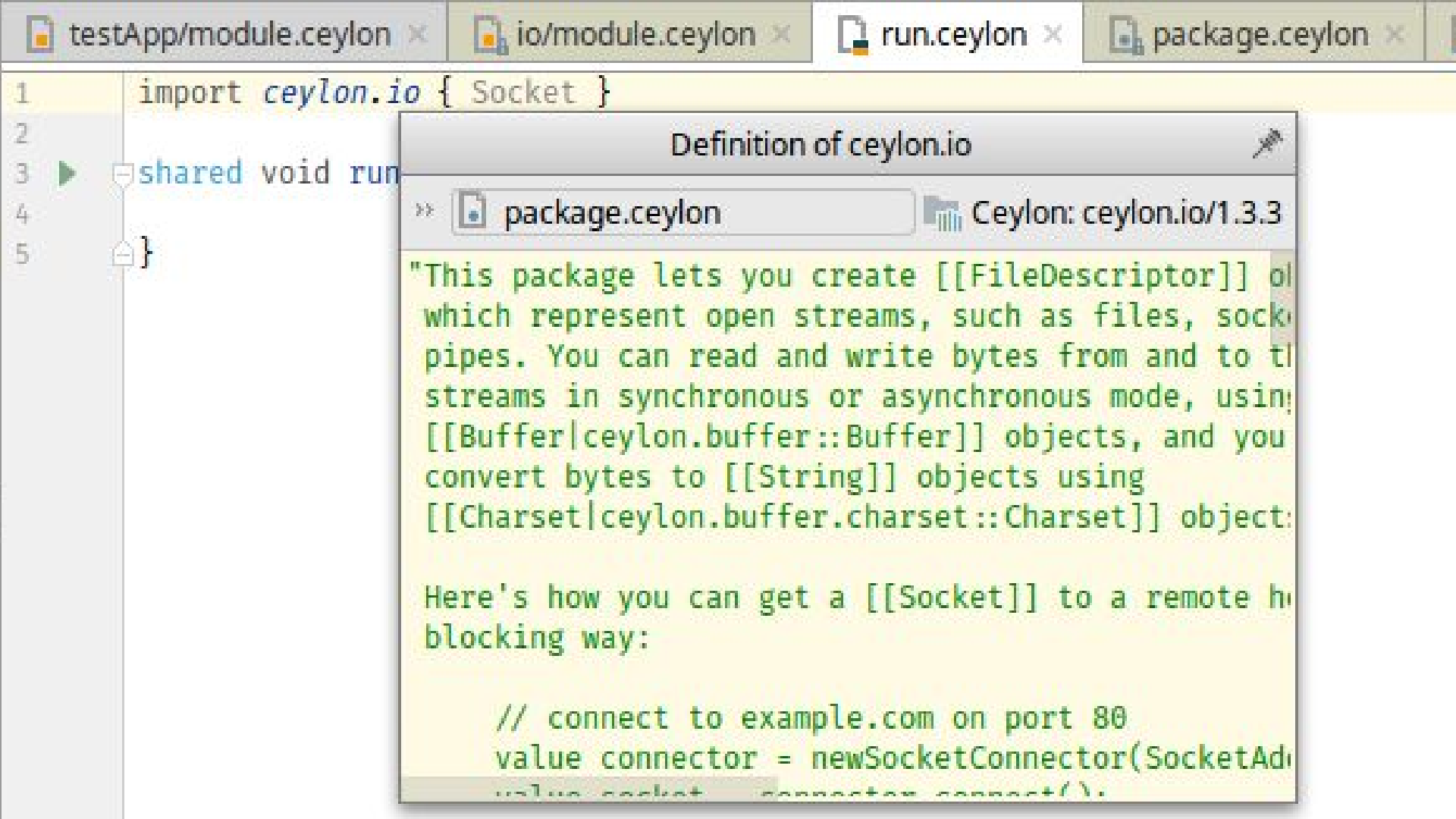
This package lets you create `FileDescriptor` objects, which represent open streams, such as files, sockets, or pipes. You can read and wri to those streams in synchronous or asynchronous mode, using Buffer objects, and you can convert bytes to `String` objects using Charset

Here's how you can get a `Socket` to a remote host in a blocking way:

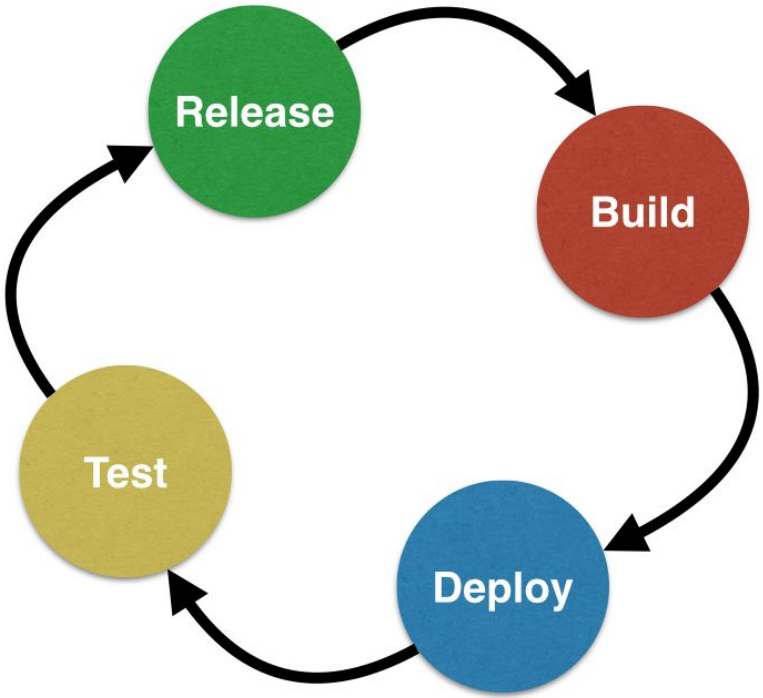
```
// connect to example.com on port 80
```

```
Finally, here's how you can read and write asynchronously
to the same socket:

    void readAndWriteAsync(String request, Socket socket) {
        Selector select = newSelector();
        // encode and write as we can
        socket.writeAsync(select, stringToByteProducer(ascii, request));
        // read, decode and print as we can
        socket.readAsync(select, byteConsumerToStringConsumer(utf8, (String
        // run the event loop
        select.process();
    }
"
by("Stéphane Épardaud")
shared package ceylon.io;
```


testApp/module.ceylon
io/module.ceylon
run.ceylon
package.ceylon
1 import ceylon.io { Socket }
2
3 shared void run
4
5 }
Definition of ceylon.io
package.ceylon
Ceylon: ceylon.io/1.3.3
"This package lets you create [[FileDescriptor]] o
which represent open streams, such as files, sock
pipes. You can read and write bytes from and to t
streams in synchronous or asynchronous mode, usin
[[Buffer|ceylon.buffer::Buffer]] objects, and you
convert bytes to [[String]] objects using
[[Charset|ceylon.buffer.charset::Charset]] object
Here's how you can get a [[Socket]] to a remote h
blocking way:
// connect to example.com on port 80
value connector = newSocketConnector(SocketAd

CI



TeamCity

- Есть поддержка ceylon для ant, maven, gradle
- Написали скрипт сборки и прогонки тестов
- поставить плагин поддержки TAP

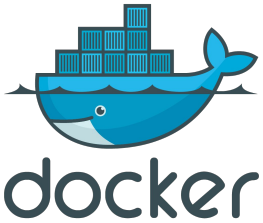
Выход новой версии ceylon

- Выпустить новую версию библиотек с поддержкой новой версии
- Пересобрать сами проекты
- Неужели это все обязательно делать каждый раз?

overrides.xml

```
<overrides xmlns="http://www.ceylon-lang.org/xsd/overrides">
    <set module="ceylon.language" version="1.3.3"/>
    <set module="com.redhat.ceylon.langtools.classfile"
         version="1.3.3"/>
    <set module="com.redhat.ceylon.module-resolver"
version="1.3.3"/>
    <set module="com.redhat.ceylon.model" version="1.3.3"/>
    <set module="com.redhat.ceylon.common" version="1.3.3"/>
</overrides>
```

Технологии


docker


kubernetes


APACHE
kafka


mongoDB


Microsoft
SQL Server


hazelcast


TeamCity


Ceylon


RxJava


GitLab CI


Java 8


Infinispan


VERT.X

Episode VIII

ПОСЛЕДНИЙ CEYLON

Episode VIII

ПОСЛЕДНИЙ CEYLON

Перейдем к заключению. Сейчас планируется выход версии 1.3.4. Язык развивается, исправляются баги. Недавно Ceylon перешел под крыло Eclipse, что дает надежду на безоблачное будущее.

Feed

Recent posts

Ceylon 1.3.3 is now available
by Gavin King
21 August 2017

Ceylon to move to the Eclipse Foundation
by Stéphane Épardaud & Gavin King
21 August 2017

Ceylon 1.3.2 is now available

Team blog

Ceylon 1.3.3 is now available

Gavin King 21 August 2017 **release** **news** **progress** **1.3.3**

Ceylon 1.3.3 is a significant minor release of the Ceylon language, with over 200 issues closed. This release introduces the `restricted` annotation, allowing more sophisticated access control, and `else case` in `switch` statements, features full support for npm scopes and Maven classifiers, allows `static` members of interfaces, and freely allows constructor and method overloading in Ceylon code marked `native("jvm")`.

A major goal of this release was to enable improvements to Vert.x's Ceylon language APIs.

This is the last release of Ceylon 1.3. The next release of Ceylon will be 1.4.0, after migration of the project to the Eclipse Foundation.

IDE Changes

Eclipse Ceylon

Eclipse Ceylon consists in a number of components:

- The Ceylon distribution, which is composed of the following modules:
  - ceylon-typechecker (compiler front-end), which includes the parser, AST and type-checker
  - ceylon-model: describes a type-checked Ceylon program (this is what the type-checker outputs), and provides a

Выводы

- Ceylon вполне применим в продакшене
- Ceylon хорошо подходит для разработки прототипов
- Ceylon легко осваивается новичками
- Ceylon показывает достаточную производительность

Ссылки

1) https://ceylon-lang.org/
2) https://habrahabr.ru/post/330412/
3) https://habrahabr.ru/post/330796/

Q & A

Рычков Н.Н

Малютин М.С.
malyutin.mikhail@gmail.com
@MikhailMalyuti2