Clojure@Echo 01

clojure.org

Числа

```
Long       1
Ratio      1/3
Double     1.0
BigInt     367868838682168188l6N
BigDecimal 3.14159265358M
```

Числа

Операции

Нормальное равенство:

```
(== 1 1.0) => true
```

Проверка на переполнение:

```
+  -  *  / quot rem
```

Авто-promoting:

```
+’ -’ *’   inc’ dec’
```

Без проверок:

```
unchecked-add…
```

Строки и символы

```
java.lang.String      “abc”
java.lang.Character   \a \b \c
```

Keywords

```
(:abc mymap)
  <=>  (mymap :abc)
  <=>  (get mymap :abc)

(:abc mymap 0)
  <=>  (mymap :abc 0)
  <=>  (get mymap :abc 0)
```

Коллекции

+ Иммутабельны
+ Персистентны
+ Шарят структуру
+ Гарантии по скорости
+ Interop в Джаву (как родные)
+ Абстрактные (count, conj, seq)

ISeq

+ Делается из любой коллекции
+ Даже из Java arrays и Iterables
+ Могут быть ленивыми

ISeq

Применения

first

rest

cons

ISeq

Применения-2

distinct filter remove for keep keep-indexed cons concat lazy-cat mapcat cycle interleave interpose rest next fnext nnext drop drop-while nthnext for take take-nth take-while butlast drop-last for flatten reverse sort sort-by shuffle split-at split-with partition partition-all partition-by map pmap mapcat for replace reductions map-indexed seque first ffirst nfirst second nth when-first last rand-nth zipmap into reduce set vec into-array to-array-2d frequencies group-by apply not-empty some reduce seq? every? not-every? not-any? empty? some filter doseq dorun doall realized? seq vals keys rseq subseq rsubseq lazy-seq repeatedly iterate repeat range line-seq resultset-seq re-seq tree-seq file-seq xml-seq itera-

Списки

```
‘(a b c)  <=>  (list a b c)
```

Быстрое добавление в начало

Остальное медленно :)

Векторы

```
  [a b c]      <=>  (vector 1 2 3)
 ([a b c] 0)   <=>  (get [a b c] 0)
```

+ Дерево с коэфф. ветвления 32
+ Доступ по индексу $O(\log_{32} N)$
+ Быстрое добавление в конец
+ rseq

Словари (maps)

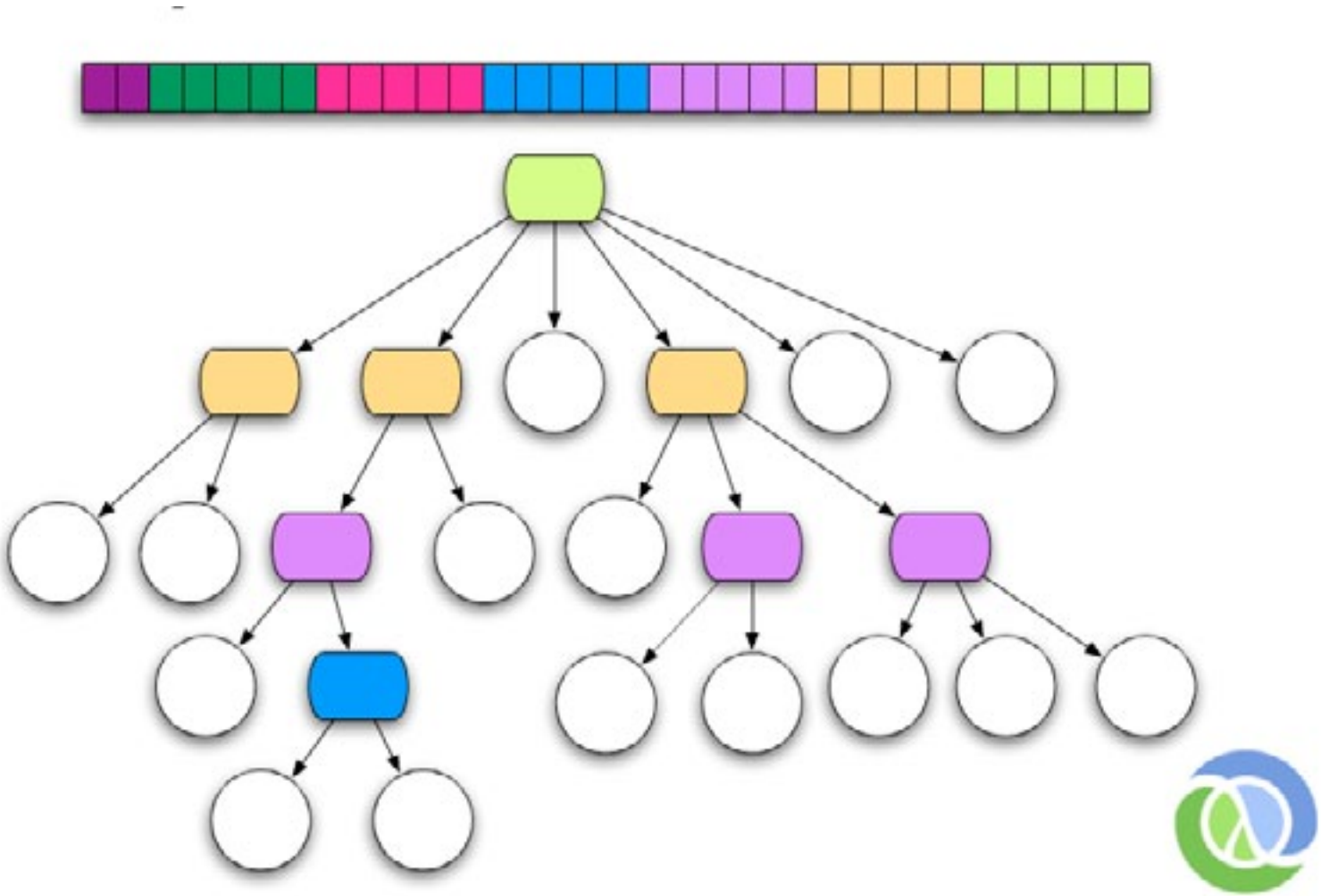
```
{:a 1, :b 2}

({…} :a)     <=>  (get {…} :a)
({…} :a 0)   <=>  (get {…} :a 0)
```

+ Bit-partitioned hash tries
+ Поиск $O(\log_{32} N)$ (hash-map)
  или $O(\log N)$ (sorted-map)

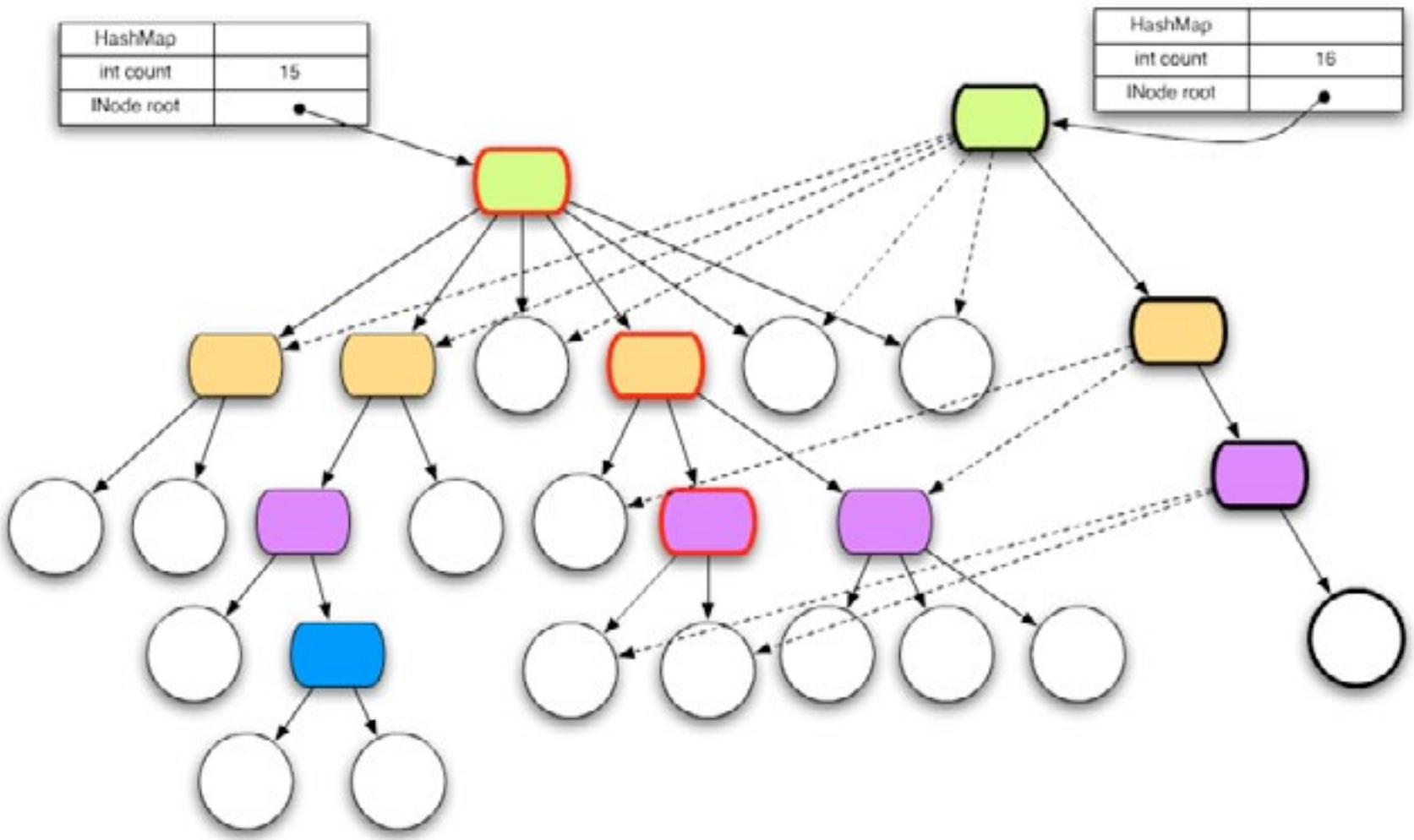
Словари (maps)

Реализация



Словари (maps)

Шаринг структуры



Множества

```
#{a b c}  <=>  (hash-set a b c)

(#{1 2 3} 1)
  <=>  (get #{1 2 3} 1)  ==> 1
(#{1 2 3} 4)
  <=>  (get #{1 2 3} 4)  ==> nil
```

Мультиметоды

dispath function + cases

```
(defmulti area
  (fn [shape]
    (get shape :type)))

;; (defmulti area :type)
```

Мультиметоды

```
(defmethod area :square [this]
  (* (:side this) (:side this)))

(defmethod area :circle [this]
  (* Math/PI (:radius this) (:radius this)))

(area {:type :circle, :radius 10})
(area {:type :square, :side 2})
```

Мультиметоды

Диспатчинг по типу

```
(defmulti foo class)

(defmethod foo java.lang.String [s]
  (format “%s” s))
(defmethod foo java.lang.Integer [i]
  (format “%i” i))
(defmethod foo nil [_]
  “nothing here”)
(defmethod foo :default [_]
  :oops)
```

Протоколы

Диспатчинг по типу
  первого аргумента
Быстрые
Не создают иерархий и отношений
Расширяются в любой момент

Протоколы

Пример

```
(defprotocol Pr
  (foo [x])
  (bar [x y]))

(defrecord Foo [a b c]
  Pr
  (foo [x] a)
  (bar [x y] (+ c y)))
```

Протоколы

Пример-2

```
(extend-protocol Pr
  String
    (foo [x] ...)
  IPersistenVector
    (foo [x] ...)
    (bar [x y] ...)
  nil
    (bar [x y] ...)
  Object
    (bar [x y] ...))
```

Протоколы

Пример-3

```
(extend-type Foo
  Countable
    (cnt [c] ...)
  Pr
    (foo [x] ...)
    (bar [x y] …))
```

Интероп

```
(new Date)  <==>  (Date.)

obj.method(args)  <==>  (.method obj args)
(.toUpperCase “fred”)

(.-prop obj)
(set! (.-prop obj) v)

(System/getProperty “java.vm.version”)
Math/PI
```

Интероп

Расширяем Джаву

```
(proxy MouseListener []
  (onMouseClick [e] …)
  (onMouseOver [e] …))

(reify Object
  (toString [this] f))
```

Интероп
Type hints

```
(set! *warn-on-reflection* true)

(defn len [^String x]
  (.length x))

(defn hinted
  (^String [] “a“))
```

Интероп

Примитивы

```
(int …), (float …), …
aget, aset, int-array, ints…
amap, areduce
^ints, ^floats, …
```

Никита Прокопов

tonsky.livejournal.com

Echo, Ульяновск

12 июля 2012

aboutecho.com

echorussia.ru

Обсуждение лекций:

tonsky.livejournal.com/tag/clojure