Clojure 数据类型介绍

刘家财
http://liujiacai.net/

提纲

- 数据类型介绍
  - 基本 Scalars
  - 组合 Collections
- 数据类型特性
  - immutable
  - lazy
  - persistent

数据类型介绍

数据类型——基本

- Number (long, double, ratio...)
- Bool
- String
- Character
- Symbol
- Keyword

数据类型——基本

```
boot.user> (type 2)
java.lang.Long
boot.user> (type 1/3)
clojure.lang.Ratio
boot.user> (type 3.14)
java.lang.Double
```

```
boot.user> (type "hello")
java.lang.String
boot.user> (type \a)
java.lang.Character
```

```
boot.user> (type true)
java.lang.Boolean
boot.user> (type false)
java.lang.Boolean
```

```
boot.user> (type 'a)
clojure.lang.Symbol
boot.user> (type :a)
clojure.lang.Keyword
```

数据类型——基本

除了 nil 与 false，其他值均为 真值（truthiness）。

```
(if 0
  true
  false)
```

真还是假?

symbol/keyword

- Symbol: 标识符(identifiers)，指向其他值。在macro 中非常有用

- keyword：与symbol类似，区别在于其值指向自身， fast equality tests。一般用在map中。

Symbol：程序体的主要**构**件

```
(defn hello [greeting]
  (println "Hello " greeting))
```

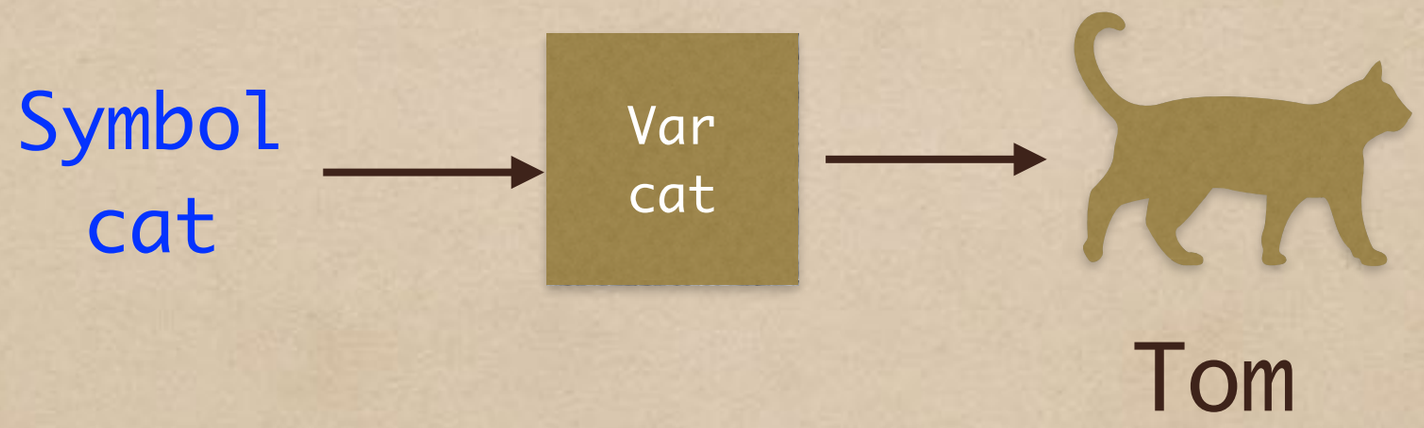
blue: symbol

red: string

```
defn -> #'clojure.core/defn
hello -> #'hello
println -> #'clojure.core/println
greeting -> ${local env}
```

Var 变量

```
(def cat "Tom")
```




```
user> (resolve 'cat)
#'user/cat
```

数据类型——集合

- List
- Vector/Queue
- Map
- Set

List

```
user=> '(1 2 3)
(1 2 3)
user=> (= (list 1 2) (list 1 2))
true
user=> (type '(1 2 3))
clojure.lang.PersistentList
user=> (conj '(1 2 3) 4)
(4 1 2 3)
```

Vector as Stack

```
user=> [1 2 3]
[1 2 3]
user=> (type [1 2 3])
clojure.lang.PersistentVector
user=> (vector 1 2 3)
[1 2 3]
user=> (vec (list 1 2 3))
[1 2 3]
user=> (conj [1 2 3] 4)
[1 2 3 4]
user=> ([:a :b :c] 1)
:b
```

先进后出

从vector中取元素

	nth	Get	Vector as fn
Vector is nil	Nil	Nil	Throw exception
Index out of range	Throw exception	Nil	Throw exception
Support not found	Yes	Yes	No

Queue

```
user=> (def schedule
         (conj clojure.lang.PersistentQueue/EMPTY
               :wake-up :shower :brush-teeth))

;;=> <-(:wake-up :shower :brush-teeth)-<

(peek schedule)
;;=> :wake-up
(pop schedule)
;;=> <-(:shower :brush-teeth)-<

(rest schedule)
;; ⚠️这里返回的是 seq, 不再是 queue
;;=> (:shower :brush-teeth)
```

先进先出

Map

```
user=> {:name "Clojure 分享" :author "jiacai"}
{:name "Clojure 分享" :author "jiacai"}
user=> (assoc {:name "Clojure 分享" :author "jiacai"} :rates "五星")
{:name "Clojure 分享" :author "jiacai" :rates "五星"}
user=> (dissoc {:name "Clojure 分享" :author "jiacai" :rates "五星"} :rates)
{:name "Clojure 分享" :author "jiacai"}
user=> (get {:name "Clojure 分享" :author "jiacai"} :name)
"Clojure 分享"
user=> ({:name "Clojure 分享" :author "jiacai"} :name)
"Clojure 分享"
user=> (:name {:name "Clojure 分享" :author "jiacai"})
"Clojure 分享"
```

Set

```
user=> #{:a :b :c}
#{:a :c :b}
user=> (conj #{:a :b :c} :d)
#{:a :c :b :d}
user=> (conj #{:a :b :c} :a)
#{:a :c :b}
user=> (disj #{:a :b :c} :a)
#{:c :b}
user=> (contains? #{1 2 3} 3)
true
user=> (set [:a :b :c])
#{:a :c :b}
```

数据类型特性

数据类型特性

- Sequential
- immutable，不可变性
- Persistent 持久化

(Seq)uential

- Clojure 里面很多算法都是用seq 来表示
- 一个 seq 是一个**逻辑**的列表（vector/list/map/set）
- 不是基于 cons，而是基于 Seq 接口
- 大部分 seq 相**关**函数具有惰性（lazy）的特性

seq 接口

```
(first seq)

(rest seq)

(cons item seq)
```

seq lib函数: filter,remove,concat,mapcat....

immutable,不可**变**性

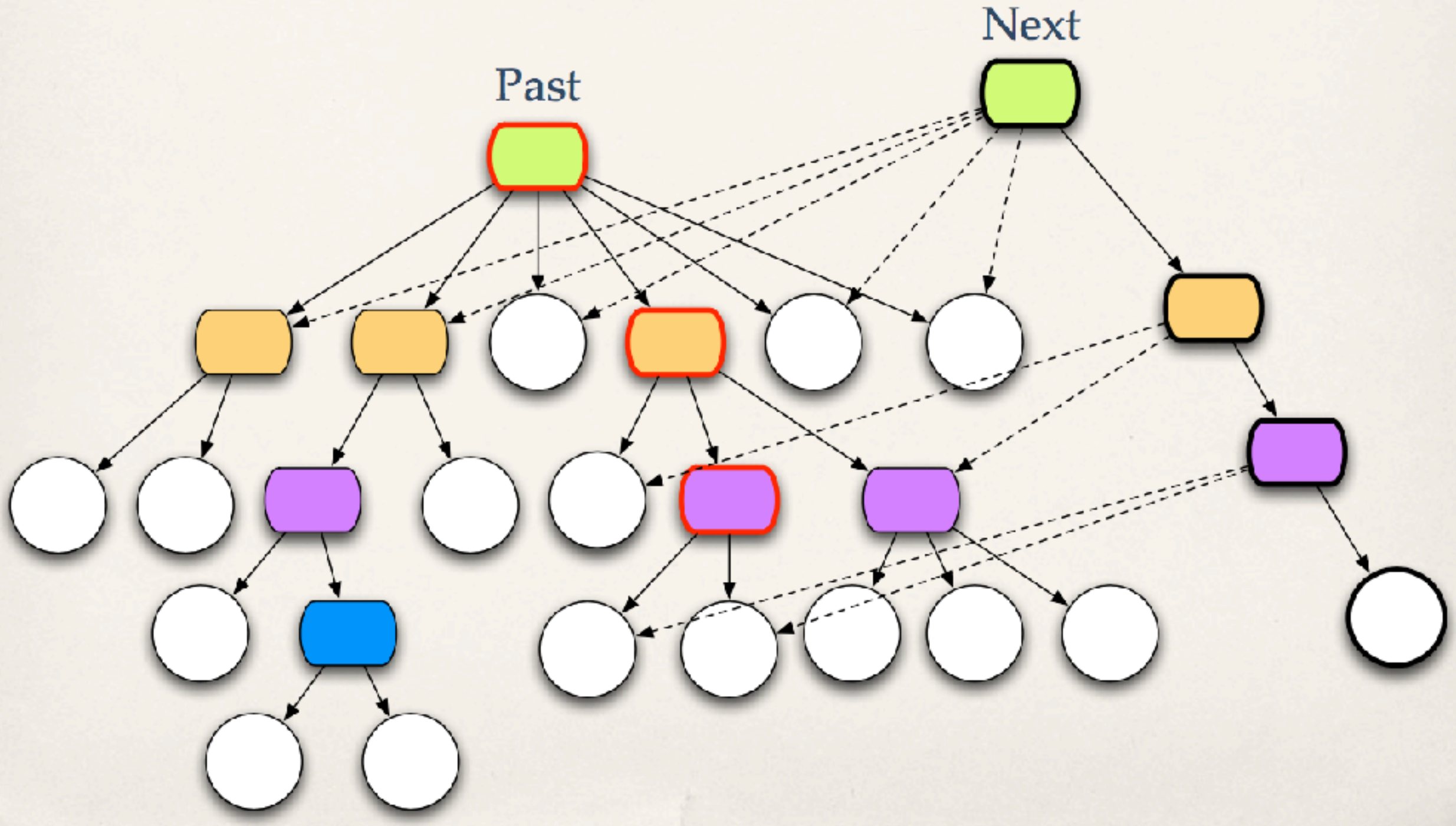
- 所有集合类型都是不可**变**的
- 修改一个集合，返回一个新的集合类型
- 适用于并**发**

immutable,不可变性

```
user> (def arr [:a :b])
#'user/arr
user> (conj arr :c)
[:a :b :c]
user> arr
[:a :b]
user> (pop arr)
[:a]
user> arr
[:a :b]
```

persistent，持久性

- 不是指保持到磁盘
- 解决不可变性带来的性能问题
- 一个研究领域：Purely functional data structure



下一讲细谈

Thank You.

群名称：SICP读书群

群　号：119845407

公众号