Core functions for destructuring

Ken Q Pu, Faculty of Science, Ontario Tech University

Comprehensive Guide

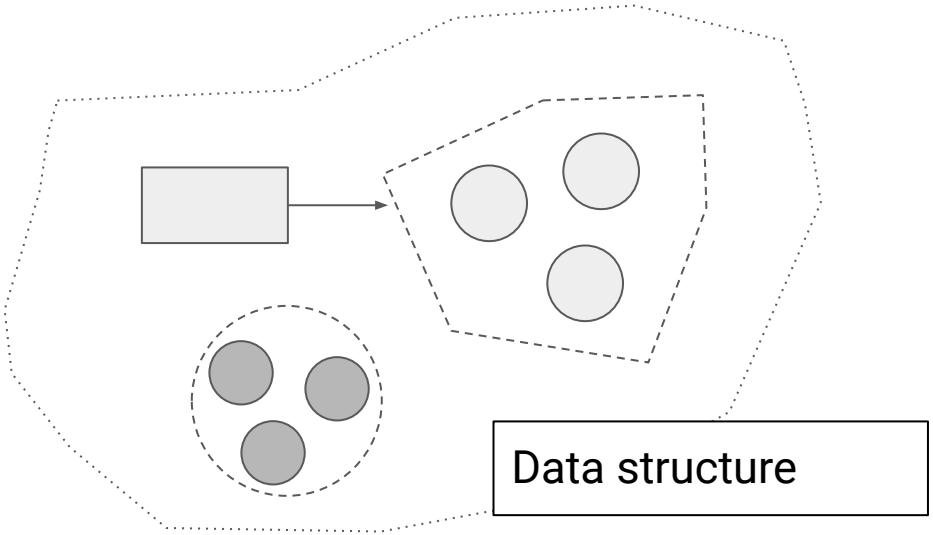
https://clojure.org/api/cheatsheet

Ken Q Pu, Faculty of Science, Ontario Tech University

Data Destructuring

Destructure: *To dismantle*.

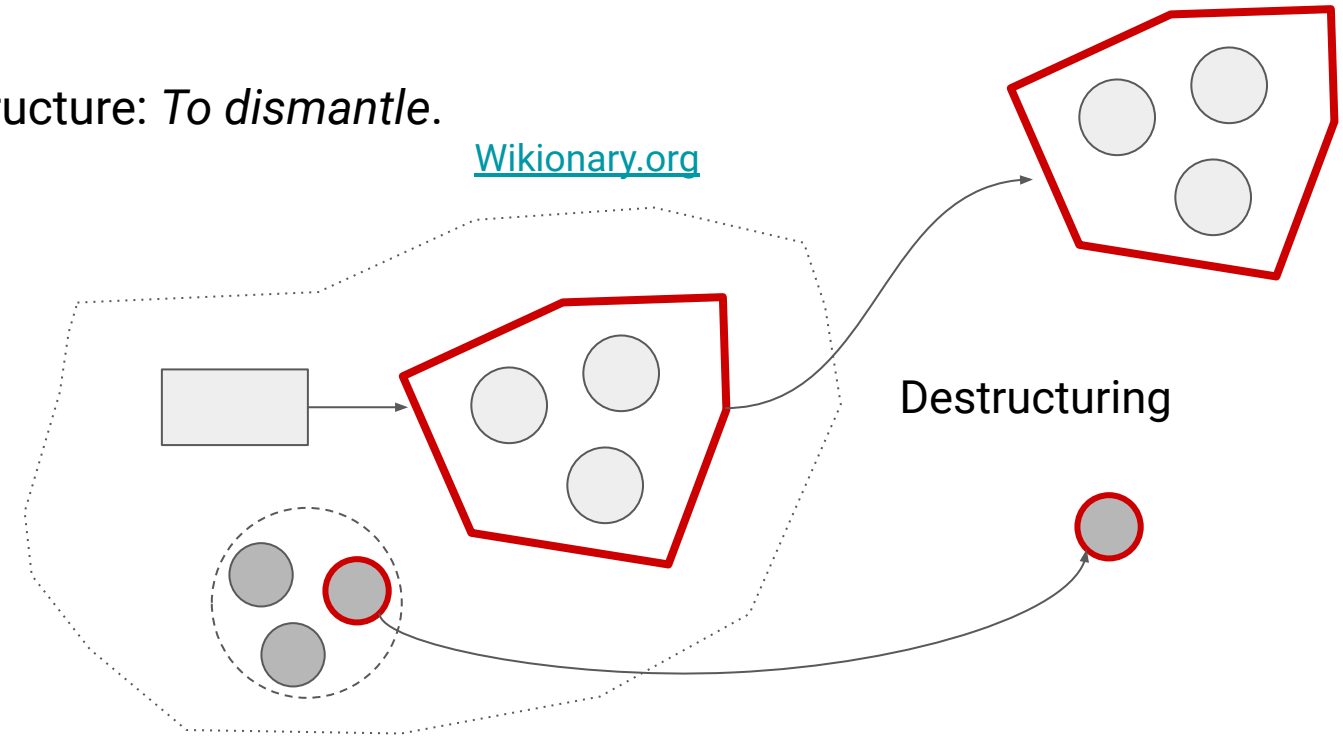
Wikionary.org



Data Destructuring

Destructure: *To dismantle*.

Wikionary.org

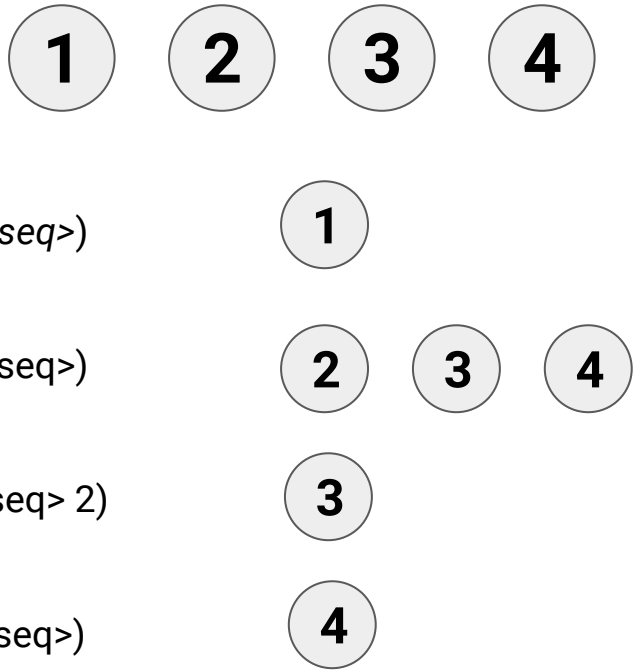


Ken Q Pu, Faculty of Science, Ontario Tech University

Sequence Destructuring

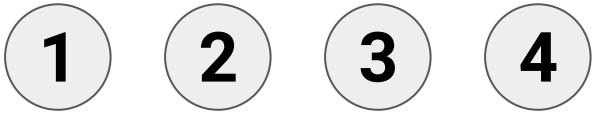
Ken Q Pu, Faculty of Science, Ontario Tech University

Sequence Destructuring

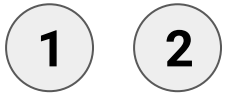


Lists, vectors, even hashmaps are treated as sequences through a standard set of destructuring functions.

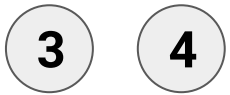
More sequence destructuring




```
(take 2 seq)
```

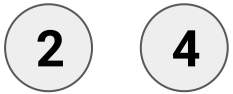



```
(drop 2 seq)
```




```
(take-while (fn [n] (<= n 3)) seq)
```


```
(filter even? seq)
```



More sequence destructuring functions

```
second
last
ffirst
drop-while
take-last
butlast
drop-last
```

Ken Q Pu, Faculty of Science, Ontario Tech University

Vector destructuring

```
(def my-vec ["a" "b" "c" "d"])

(my-vec 2) ⇒ "c"

(nth my-vec 2) ⇒ "c"

(get my-vec 2) ⇒ "c"

(subvec my-vec 2 4) ⇒ ["c" "d"]
```

Vectors can be used as functions. It takes an index as a parameter.

Hashmap Destructuring

Ken Q Pu, Faculty of Science, Ontario Tech University

Hashmap review

```
{ "Jack" 55
  "Jill" 56
  "Joe"  79 }
```

```
{ ["Jack", "Bauer"]  55
  ["Jill", "Deacon"] 56
  ["Joe", "Biden"]   79 }
```

Anything can be a key in a hashmap

```
{ :first-name "Joe"
  :last-name  "Biden"
  :age        79 }
```

Whenever possible, we should use keywords to represent attribute names.

Getting value by a given key

```
(def my-map {"Jack" 55
             "Jill" 56
             "Joe"  76})
```

Top-level symbols should be avoided.

Don't do this in practice.

```
(get my-map "Jack")
⇒ 55
```

```
(get my-map "Albert")
⇒ nil
```

```
(get my-map "Albert" -1)
⇒ -1
```

Getting value by a given key

```
(def my-map {"Jack" 55
             "Jill" 56
             "Joe"  76})
```

Top-level symbols should be avoided.

Don't do this in practice.

```
(my-map "Jack")
⇒ 55

(my-map "Albert")
⇒ nil
```

A hashmap can be used as a function with the key as parameter. It will return the value, or nil.

Getting value by a given **keyword** key

```
(def president {:first-name "Joe"
                :last-name "Biden"
                :age 76})
```

```
(:first-name president)
⇒ "Joe"
```

```
(:hobby president)
⇒ nil
```

Nested hashmaps

```
(def myclass
 {:title "Calculus"
  :lectures [{:day :Monday
              :start [11 0]
              :end   [12 30]
              :room  "UA1010"}
             {:day :Wednesday
              :start [14 0]
              :end   [15 0]}]
  :instructor {:first-name "Albert"
               :last-name "Einstein"
               :office "UA3030" }})
```

Nested hashmaps

```
(def myclass
 {:title "Calculus"
  :lectures [{:day :Monday
              :start [11 0]
              :end   [12 30]
              :room  "UA1010"}
             {:day :Wednesday
              :start [14 0]
              :end   [15 0]}]
  :instructor {:first-name "Albert"
               :last-name "Einstein"
               :office "UA3030"}})
```

```
(get myclass :instructor)

⇒ {:first-name "Albert"
    :last-name "Einstein"
    :office "UA3030"}
```

```
(get (get myclass :instructor) :office)

⇒ "UA3030"
```

```
(get-in myclass [:instructor :office])

⇒ "UA3030"
```

Nested hashmaps

```
(def myclass
 {:title "Calculus"
  :lectures [{:day :Monday
              :start [11 0]
              :end   [12 30]
              :room  "UA1010"}
             {:day :Wednesday
              :start [14 0]
              :end   [15 0]}]
  :instructor {:first-name "Albert"
               :last-name "Einstein"
               :office "UA3030"}})
```

```
(get
 (get
  (get myclass :lectures) 0) :room)

⇒ "UA1010"
```

```
(get-in myclass [:lectures 0 :room])

⇒ "UA1010"
```

```
(:room (first (:lectures myclass)))

⇒ "UA1010"
```

Nested hashmaps

```
(def myclass
 {:title "Calculus"
  :lectures [{:day :Monday
              :start [11 0]
              :end   [12 30]
              :room  "UA1010"}
             {:day :Wednesday
              :start [14 0]
              :end   [15 0]}]
  :instructor {:first-name "Albert"
               :last-name "Einstein"
               :office "UA3030"}})
```

```
(defn duration [myclass]
 (let [lectures (:lectures myclasses)]
  (loop [total 0
         xs lectures]
    (if (empty? xs) total
      (recur (+ total (length (first xs)))
             (rest xs))))))


(defn length [lecture]
 (let [start-hr (get-in lecture :start 0)
       start-m (get-in lecture :start 1)
       end-hr (get-in lecture :end 0)
       end-m (get-in lecture :end 1)]
  (- (+ (* 60 end-hr) end-m)
     (+ (* 60 start-hr) start-m))))
```

Ken Q Pu, Faculty of Science, Ontario Tech University

Getting keys

```
(def president {:first-name "Joe"
                :last-name "Biden"
                :age 76})
```

```
(keys president)
⇒ (:first-name :last-name :age)
```

```
(vals president)
⇒ (76)
```