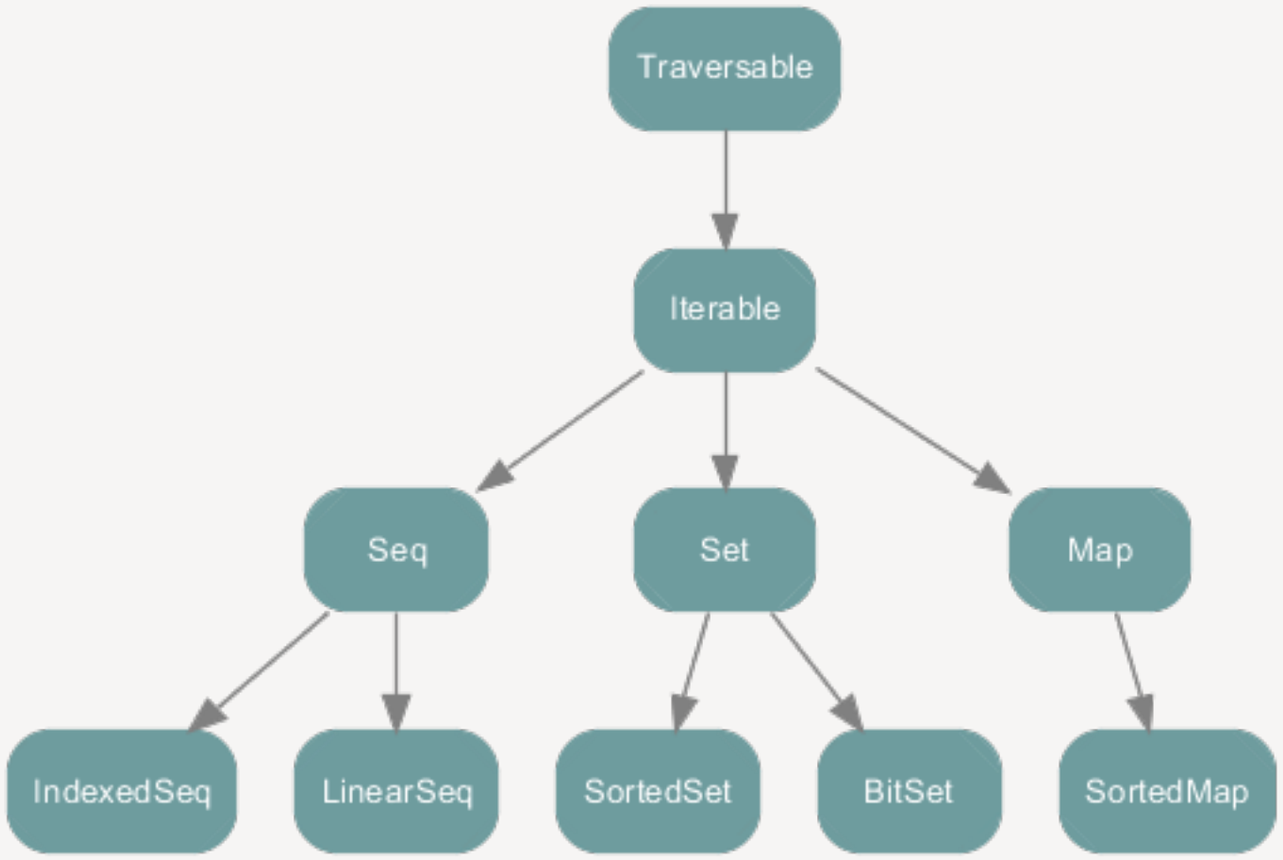
SCALA COLLECTION

吴雪峰@Thoughtworks

2014年6月15日



Tuple

Option

SCALA.COLLECTION

列表 LIST

```
val numbers = List(1, 2, 3, 4)
```

集 SET 没有重复

```
val numbers = Set(1, 1, 2)
res0:immutable.Set[Int] = Set(1, 2)
```

键值对 MAP

```
val dic = Map("foo" -> "bar")
dic("foo") //"bar"
```

元组 TUPLE 不同类型集合

```
val hostPort = ("localhost", 80)
scala> hostPort._1
res0: String = localhost

scala> hostPort._2
res1: Int = 80

hostPort match {
  case ("localhost", port) => ...
  case (host, port) => ...
}
```

选项 OPTION 表示有可能包含值

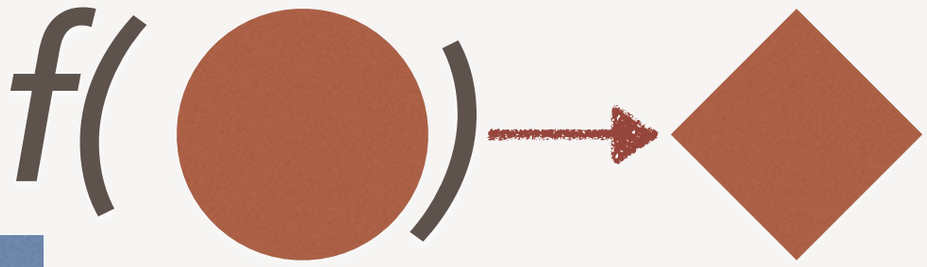
*Option*是抽象类，有两个子类： *Some[T]* 或 *None*

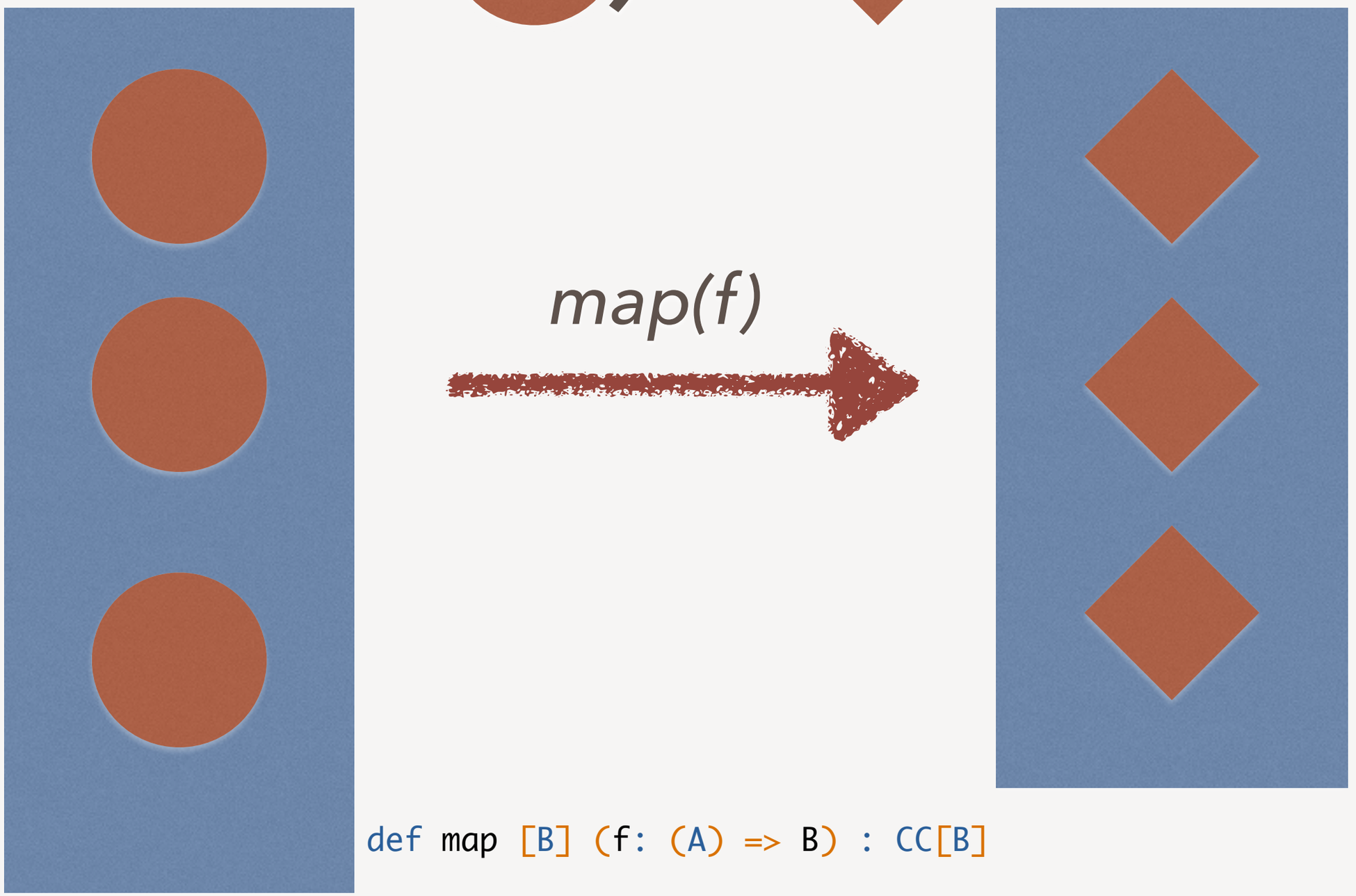
```
scala> val numbers = Map(1 -> "one", 2 -> "two")
numbers: Map[Int,String] = Map((1,one), (2,two))

scala> numbers.get(2)
res0: Option[java.lang.String] = Some(two)

scala> numbers.get(3)
res1: Option[java.lang.String] = None


val result = res0.getOrElse(0)
```

函数组合子

- MAP
- FOREACH
- FILTER
- ZIP
- PARTITION
- FIND
- DROP AND DROPWHILE
- FOLDRIGHT AND FOLDLEFT
- FLATTEN
- FLATMAP
- 扩展函数组合子

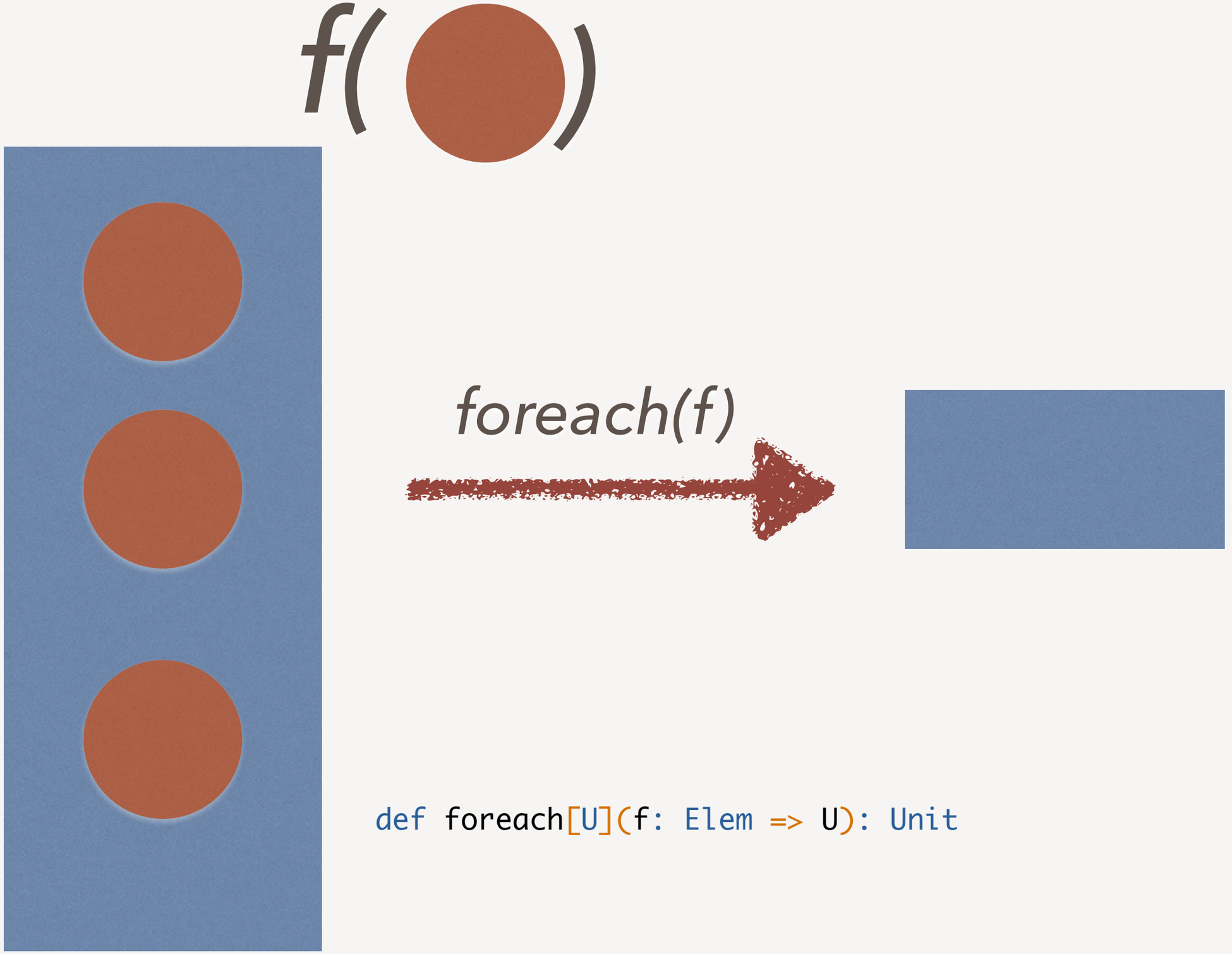
MAP





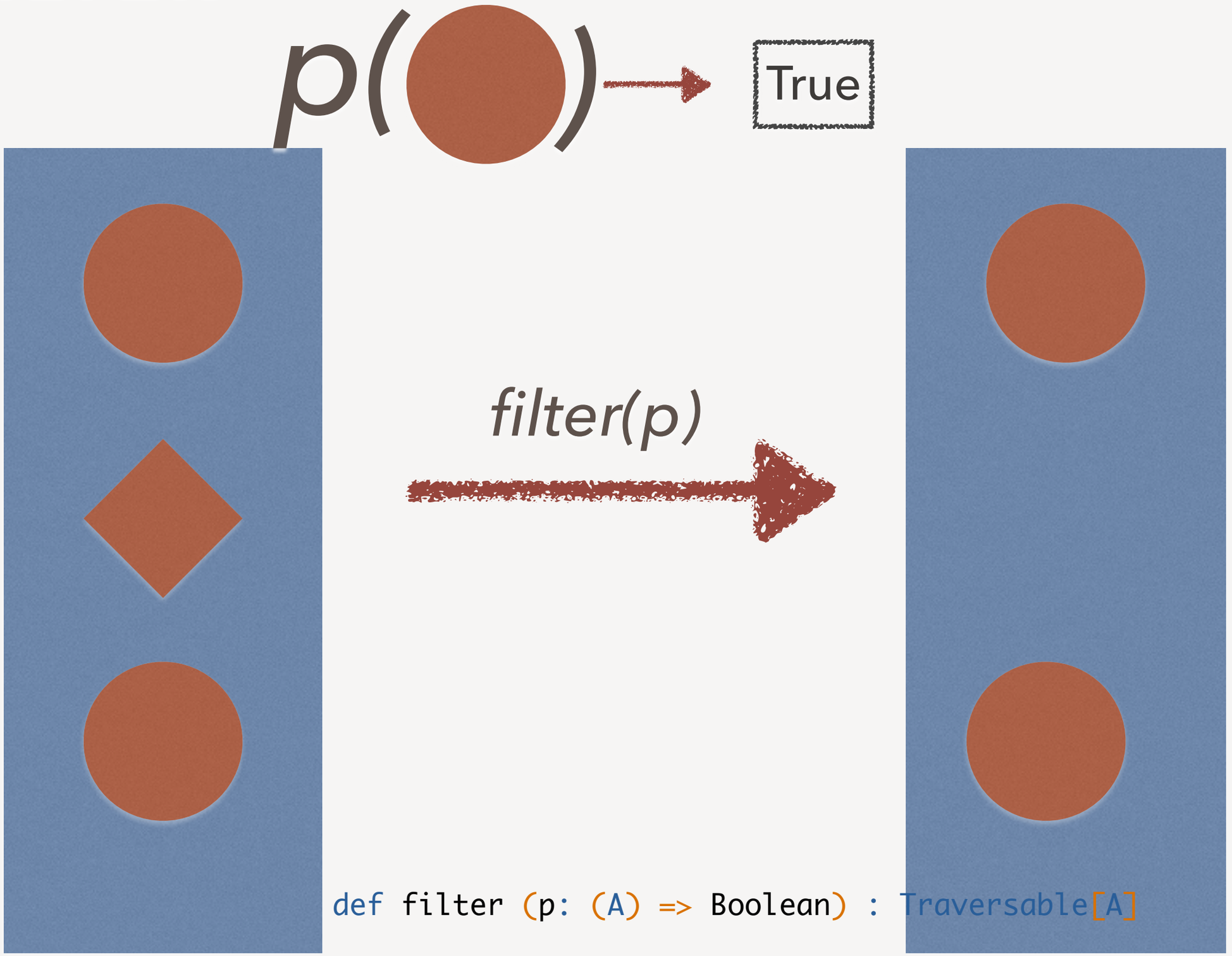

```
def map [B] (f: (A) => B) : CC[B]
```

FOREACH



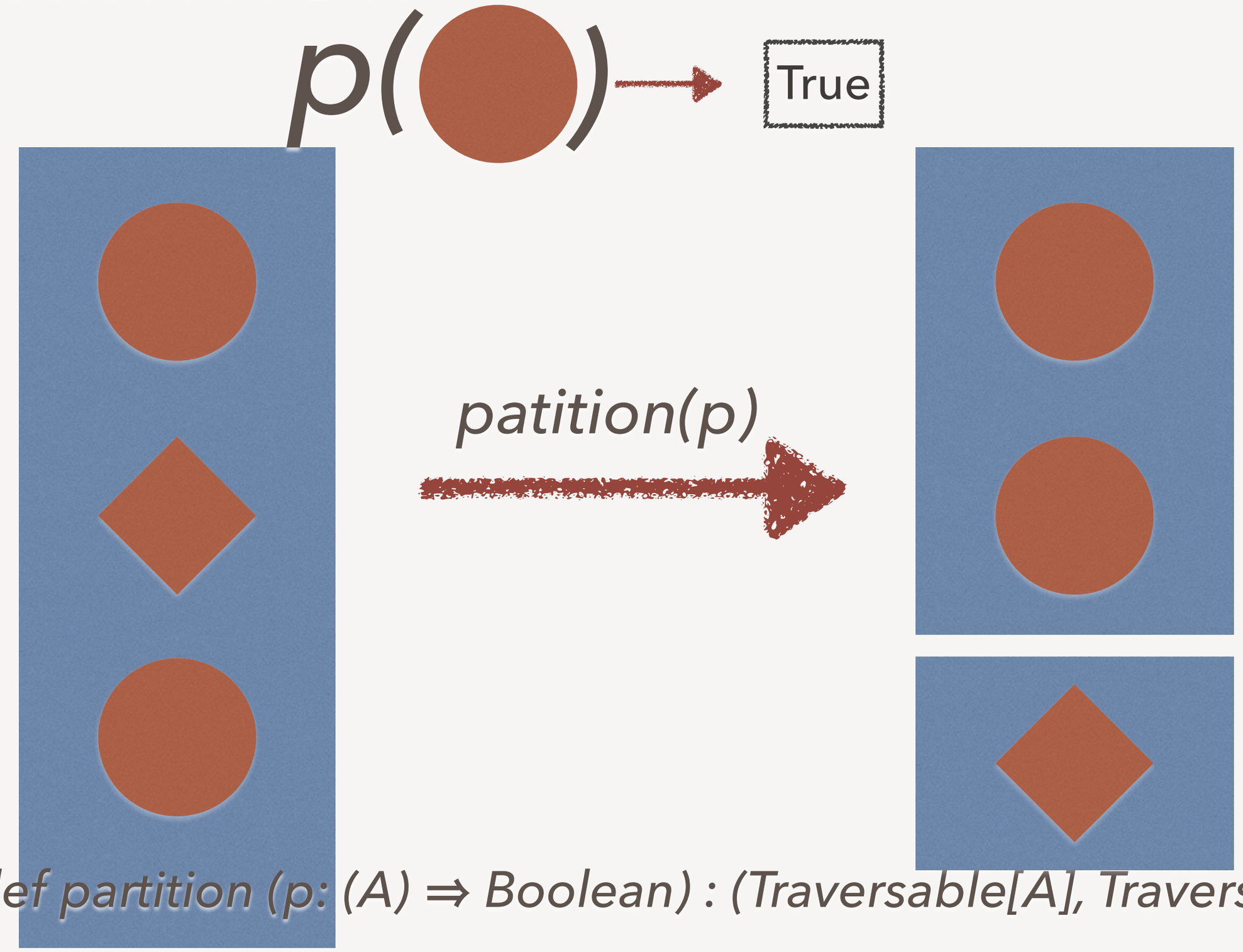

```
def foreach[U](f: Elem => U): Unit
```

FILTER



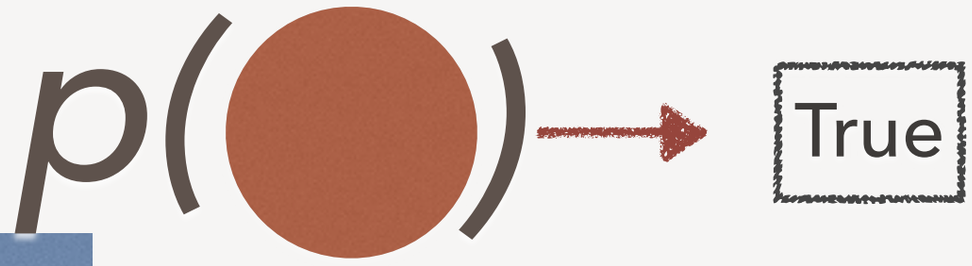

```
def filter (p: (A) => Boolean) : Traversable[A]
```

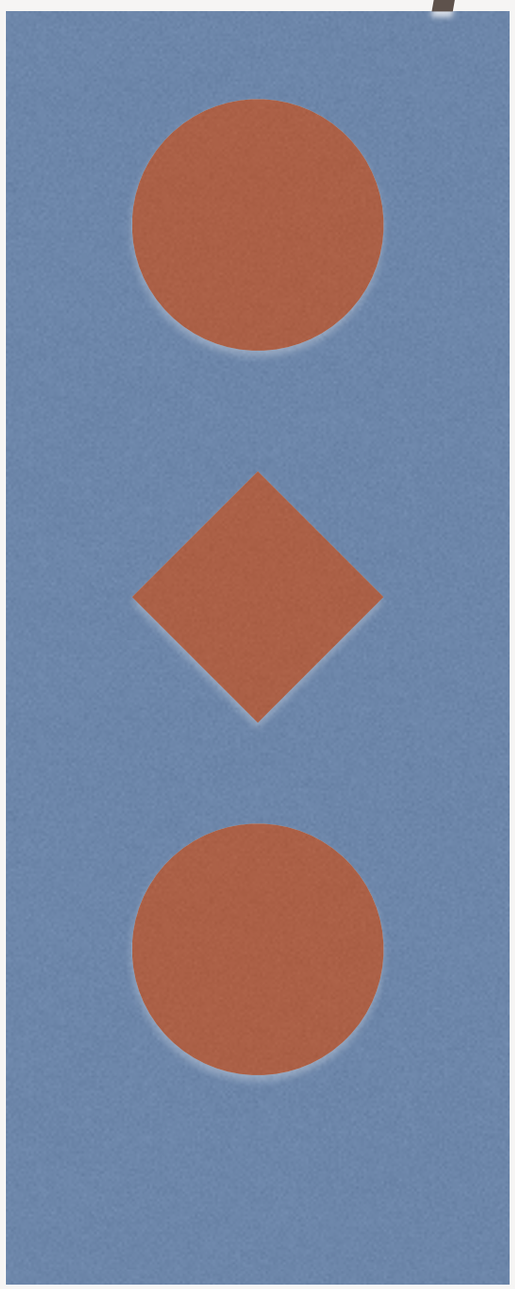
PARTITION

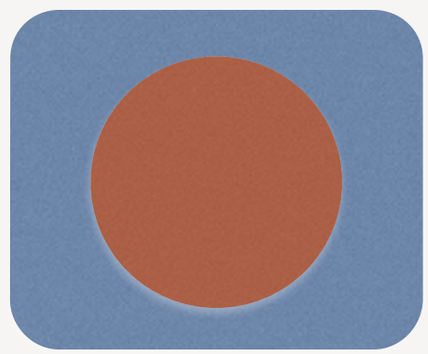


def partition (p: (A) ⇒ Boolean) : (Traversable[A], Traversable[A])

FIND

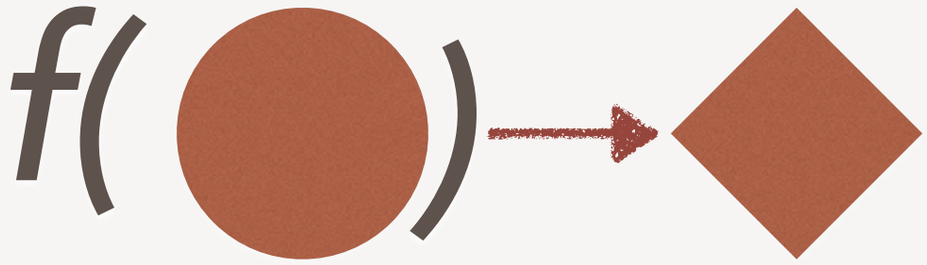


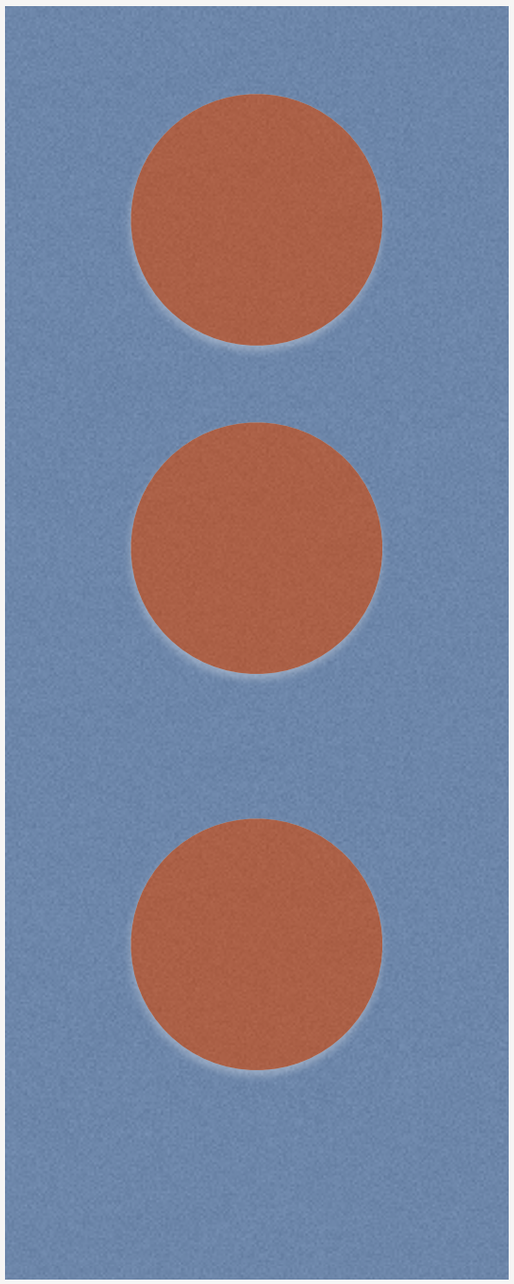




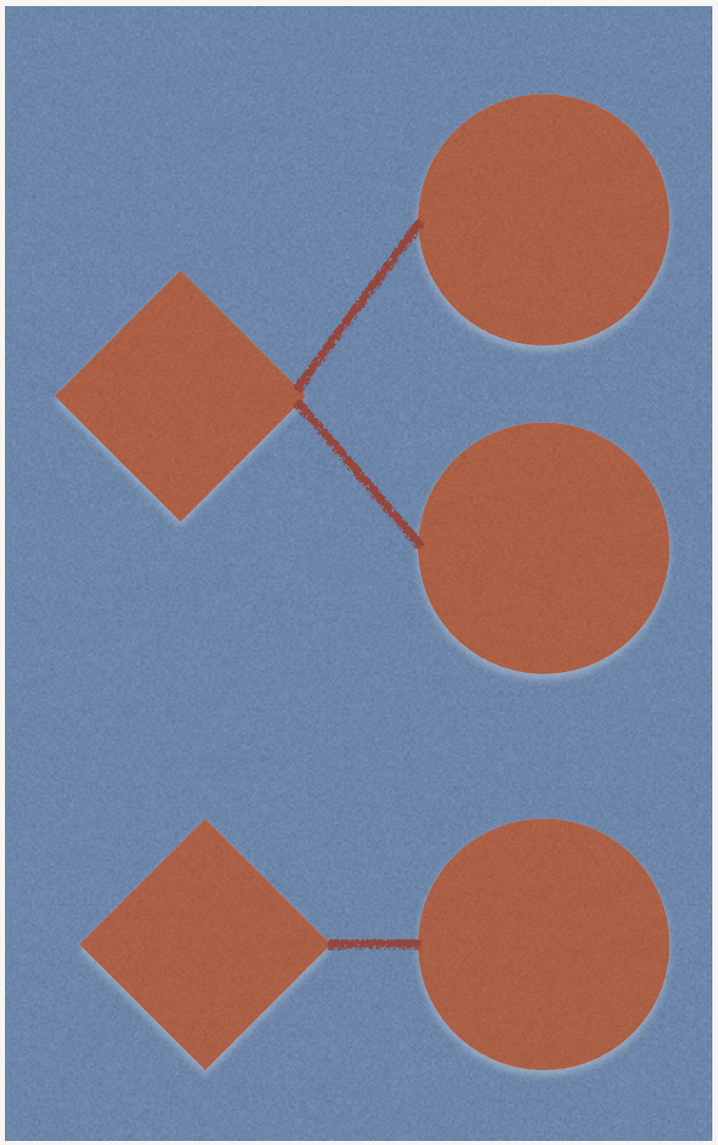
```
def find (p: (A) => Boolean) : Option[A]
```

GROUPBY



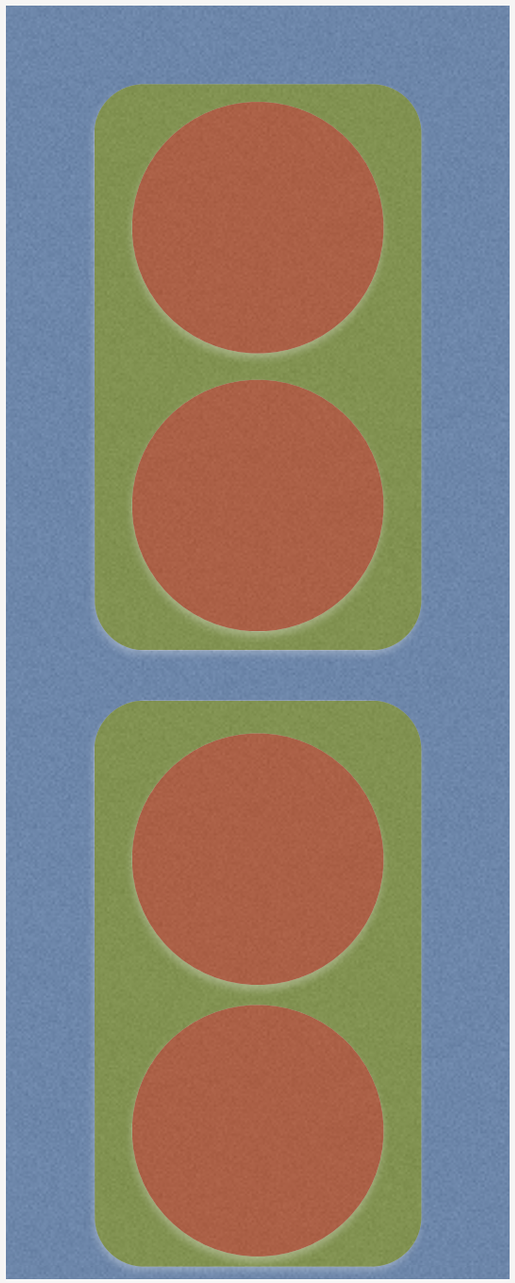


groupBy(f)

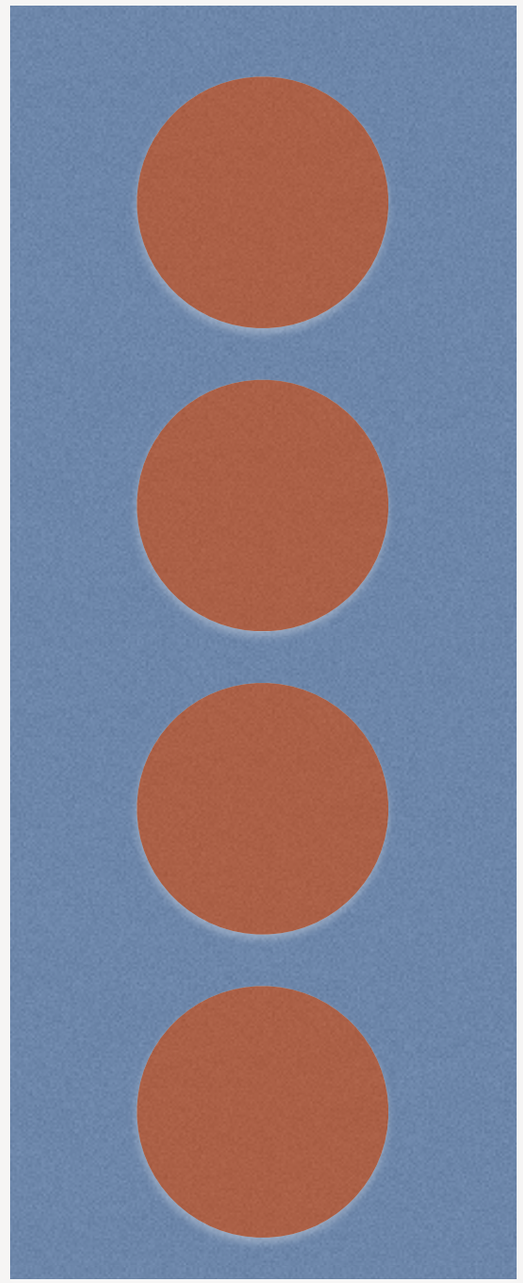


```
def groupBy [K] (f: (A) => K) : Map[K, Traversable[A]]
```

FLATTEN

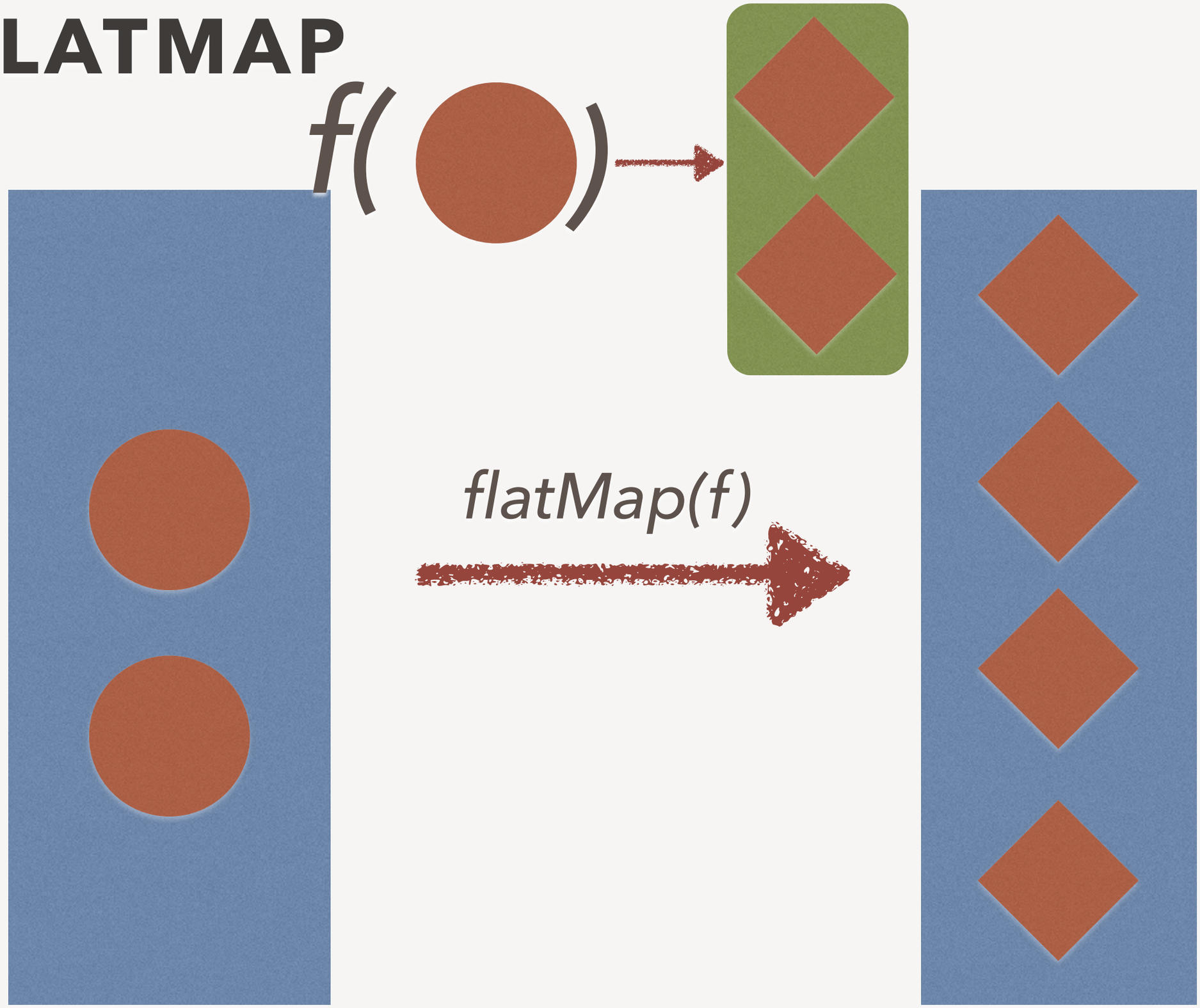


flatten



```
def flatten[a](l: List[List[a]]): List[a]
```

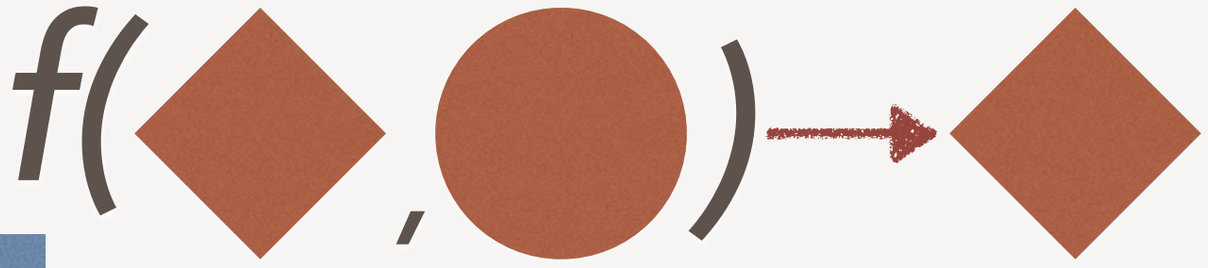
FLATMAP

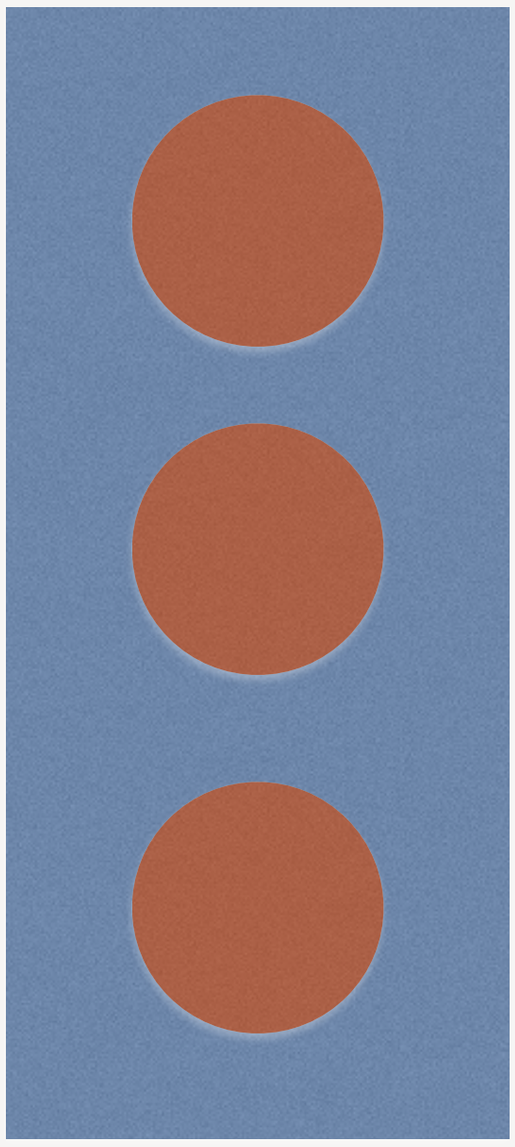



```
def flatMap[B](f: A => Option[B]): Option[B]
```

REDUCE

REDUCELEFT



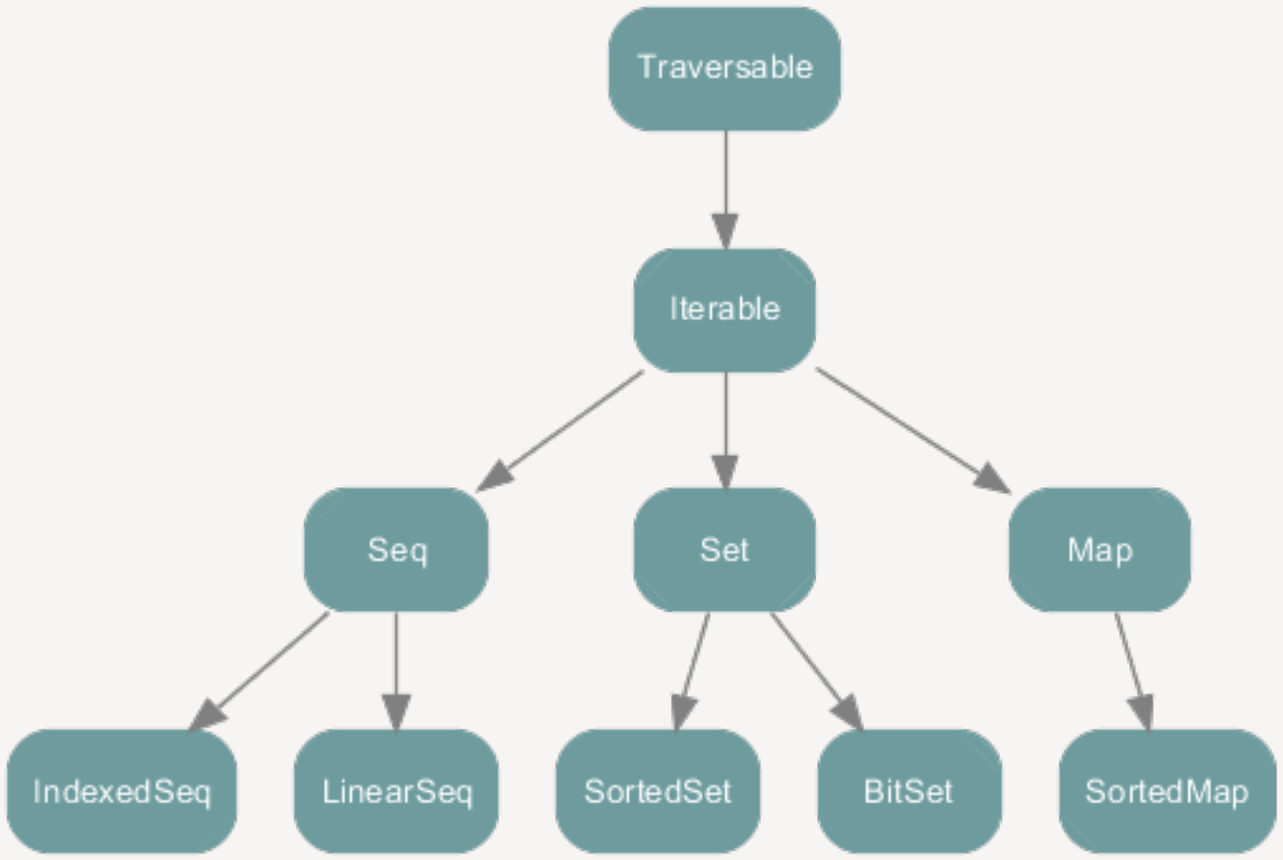



```
def reduceLeft[B >: A](op: (B, A) => B): B
```

TRAVERSABLE

```
def foreach[U](f: A => U): Unit
def map(f: A => B)
def isEmpty: Boolean
def takeWhile(p: A => Boolean)
def dropWhile(p: A => Boolean)
def span(p: A => Boolean)
def splitAt(n: Int)
def filterNot(p: A => Boolean)
def filter(p: A => Boolean)
def find[A](p: A => Boolean):Option[A]
def collect(pf: PartialFunction[A, B])
//List(1,2,"SS").collect{case i:Int => i}
def partition(p: A => Boolean)
def groupBy[K](f: A => K):
immutable.Map[K, Repr]
def forall(p: A => Boolean): Boolean
def exists(p: A => Boolean)
```

```
def head: A
def headOption: Option[A]
def tail:
def last: A
def lastOption: Option[A]
def init: Repr
def take(n: Int)
def drop(n: Int)
def tails: Iterator[Repr]
def view
def flatMap[B, That](f: A => GenTraversableOnce[B])

def foldLeft[B](z: B)(op: (B, A) => B)
def reduceLeft[B >: A](op: (B, A) => B)
```



Tuple

Option

SCALA.COLLECTION

更多子类

- HashSet和HashMap 的快速查找， 这些集合的最常用的形式。

- TreeMap 是SortedMap的一个子类， 它可以让你进行有序访问。

- Vector 快速随机选择和快速更新。

- Range 等间隔的Int有序序列。你经常会在for循环看到