Functional Collections

A Functional Set

A Functional Collection: *Set*

```scala
val set = Set(1,2,3)
```

Set instances are callable (they have *apply*)

```scala
set(2) // true
set(42) // false
```

Set instances are callable like functions.
The *apply* method *always* returns a value: true/false.

=> Sets behave like actual functions.

Sets ARE functions!

```scala
trait Set[A] extends (A) => Boolean with ...
```

Moar Functional Collections!

Functional Seq

Sequences are "callable" through an integer index:

```scala
trait Seq[+A] extends PartialFunction[Int, A] {
  def apply(index: Int): A
}
```

“give me the element at *index* in the sequence”

```scala
val numbers = List(1, 2, 3)
numbers(1) // 2
numbers(3) // java.lang.IndexOutOfBoundsException
```

Seqs are partially defined on the domain [0, ..., length – 1]

Sequences are partial functions!

Functional Map

Maps are "callable" through their keys:

```
trait Map[A, +B] extends PartialFunction[A, B] {
  def apply(key: A): B
  def get(key: A): Option[B]
}
```

"give me the *value* associated to *key*"

```
val phoneMappings = Map(2 -> "ABC", 3 -> "DEF")
phoneMappings(2) // "ABC"
phoneMappings(1) // java.lang.NoSuchElementException
```

A map is defined on the domain of its keys.

Maps are partial functions!

Scala rocks