Monads

Monads Rock!

Intro

Monads are a kind of types which have some fundamental ops.

```
trait MonadTemplate[A] {
  def unit(value: A): MonadTemplate[A]
  def flatMap[B](f: A => MonadTemplate[B]): MonadTemplate[B]
}
```

also called *pure* or *apply*

also called *bind*

List, Option, Try, Future, Stream, Set are all *monads.*

Operations must satisfy the *monad laws:*

left-identity

```
unit(x).flatMap(f) == f(x)
```

right-identity

```
aMonadInstance.flatMap(unit) == aMonadInstance
```

associativity

```
m.flatMap(f).flatMap(g) == m.flatMap(x => f(x).flatMap(g))
```

Example: List

Left identity:

```
List(x).flatMap(f) =
f(x) ++ Nil.flatMap(f) =
f(x)
```

Right identity:

```
list.flatMap(x => List(x)) =
list
```

Associativity:

```
[a b c].flatMap(f).flatMap(g) =
(f(a) ++ f(b) ++ f(c)).flatMap(g) =
f(a).flatMap(g) ++ f(b).flatMap(g) ++ f(c).flatMap(g) =
[a b c].flatMap(f(_).flatMap(g)) =
[a b c].flatMap(x => f(x).flatMap(g))
```

Another Example: Option

Left identity:

```
Option(x).flatMap(f) = f(x)
```

```
Some(x).flatMap(f) = f(x)
```

Right identity:

```
opt.flatMap(x => Option(x)) = opt
```

```
Some(v).flatMap(x => Option(x)) =
Option(v) =
Some(v)
```

Associativity

```
o.flatMap(f).flatMap(g) = o.flatMap(x => f(x).flatMap(g))
```

```
Some(v).flatMap(f).flatMap(g) = f(v).flatMap(g)
Some(v).flatMap(x => f(x).flatMap(g)) = f(v).flatMap(g)
```

Scala rocks