Monads

Monads

Monads are a kind of types which have some fundamental ops

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

object MonadTemplate {
  def apply(value: A): MonadTemplate[A]
}
```

also called *bind*

also called *pure*

Examples of monads: List, Option, IO (Cats Effect), ZIO

Operations must satisfy the *monad properties ("laws")*

left-identity

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

right-identity

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

associativity

```scala
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))
```

Example: Option

Left-identity

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

actual implementation in code

Right-identity

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

Some(x).flatMap(x => Option(x)) = Some(x)
None.flatMap(x => Option(x)) = None
```

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)

None.flatMap(f).flatMap(g) = None
None.flatMap(x => f(x).flatMap(g)) = None
```

same

same

Scala rocks