Kleisli

Kleisli

Wrap functions returning F[_] instances

```
val func1: Int => Option[String] = x => if (x % 2 == 0) Some(s"$x is even") else None
val func2: Int => Option[Int] = x => Some(x * 3)

import cats.data.Kleisli
import cats.instances.option._ // FlatMap[Option]
val func1K: Kleisli[Option, Int, String] = Kleisli(func1)
val func2K: Kleisli[Option, Int, Int] = Kleisli(func2)
val func3K: Kleisli[Option, Int, String] = func2K andThen func1K
```

Lots of convenience functions

- map, flatMap
- apply
- andThen
- traverse

Used for

- function composition when they return F[_] types
- dependency injection – Reader!

Cats rock