Monads!

Mike Hadlow
@mikehadlow
http://mikehadlow.blogspot.com
mikehadlow@suteki.co.uk

What we'll talk about…

- What is a Monad?
- A (very) brief history of Monads.
- Lots of C# code ☺
- Thoughts on C# vs F# vs Haskell.

https://github.com/mikehadlow/Suteki.Monads

“Amplified Types”

- Collections: IEnumerable<string>, IList<int>
- Nullable<int>
- Task<string>
- They wrap and ‘enhance’ simple types.
- They all require boiler plate to access their wrapped values.

Amplified type composition

- Monads allow us to compose amplified types naturally without boiler plate.
- To be a Monad, a Whatever<T> must implement two methods:

```
Whatever<T> ToWhatever<T>(T value) // AKA unit

Whatever<B> Bind<A,B>(Whatever<A> a, Func<A, Whatever<B>> func)
```

Maths!

- Category theory 1940s.
- ~~You need to understand Category theory to understand Monads.~~

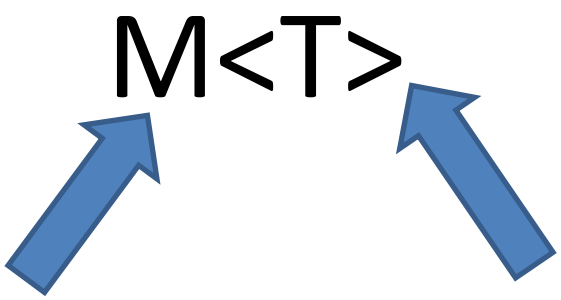
Haskell

- Haskell is a pure lazy functional programming language.
- No side effects. No guaranteed order of execution.
- Monads first introduced by Eugenio Moggi and Philip Wadler to enable side effecting functions.
- Many applications of Monads in Haskell.

Code!

The limitations of Linq & C#

- No control structures (if/else, loops)
- We can't define a Monad in C# because we don't have "types of types".



This can't be generic

This is generic

F# Computation Expressions

```
type Identity<'a> = Identity of 'a

let getValue (a : Identity<'a>) = match a with Identity x -> x
let mreturn x = Identity x
let bind (a : Identity<'a>) (f : 'a -> Identity<'b>) = f
(getValue a)

type IdentityBuilder() =
    member x.Bind(a, f) = bind a f
    member x.Return(a) = mreturn a

let identity = new IdentityBuilder()
let result = identity {
    let! a = Identity 4
    let! b = Identity 3
    return a + b
    }

printfn "result = %A" (getValue result)
```

Haskell ‘do’ notation

```
data Identity a = Identity a

getValue (Identity a) = a

instance Monad Identity where
    return a = Identity a
    (>>=) a f = f $ getValue a

main = putStrLn $ show $ getValue $ do
    a <- Identity 4
    b <- Identity 3
    return (a + b)
```

Where next?

- My Monad series on Code Rant
- Wikipedia Monad Page
- Wes Dyer – The Marvel of Monads
- Read a good Haskell Book:
  - Learn you a Haskell for Great Good
  - Read World Haskell

Questions?