# Functional Programming: A Terribly Civilised Introduction

A British Guide to Category Theory, Functors, Monoids, and Monads

WorldSMEGraphs

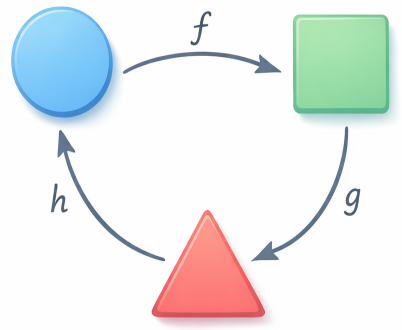
# Functional Programming: A Terribly Civilised Introduction


- Or: How I Learned to Stop Worrying and Love the Monad

# Right Then, Let's Be Honest

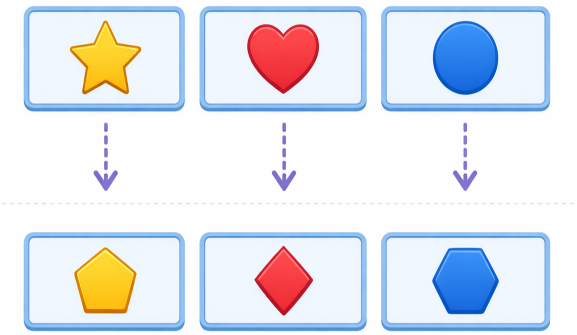
- You're here because someone said "monads" in a meeting
- And you nodded wisely whilst dying inside
- Don't worry, we've all been there
- (Some of us for rather longer than we'd care to admit)
- Today we shall fix that. Probably. Maybe. We'll see.

# Category Theory: It's Less Scary Than It Sounds




- Invented by mathematicians in the 1940s
- (They needed something to do during the war besides codebreaking)
- It's really just about "things" and "arrows between things"
- Objects + Morphisms = That's literally it
- You've been doing this all along, you just didn't know it had a fancy name

# Functors: The Art of Preserving Structure

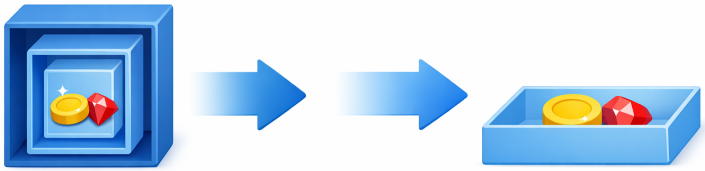


- A Functor is a "structure-preserving mapping"
- (In plainer English: .Select() in LINQ, .map() in JavaScript)
- It takes things in boxes and transforms them whilst keeping them boxed
- The box might be: List, Option, Task, Observable...
- Quite useful, really. Terribly sensible.

# Monoids: Things That Combine Nicely

- Two rules: things can be combined, and there's an "identity" thing
- Numbers with addition: 1 + 2 = 3, and 0 + anything = anything
- Strings with concatenation: "Hello" + "World", and "" + anything = anything
- Lists: [1,2] ++ [3,4], and [] ++ anything = anything
- It's like the Queen's rules of etiquette, but for data

# Monads: Sorry About This One

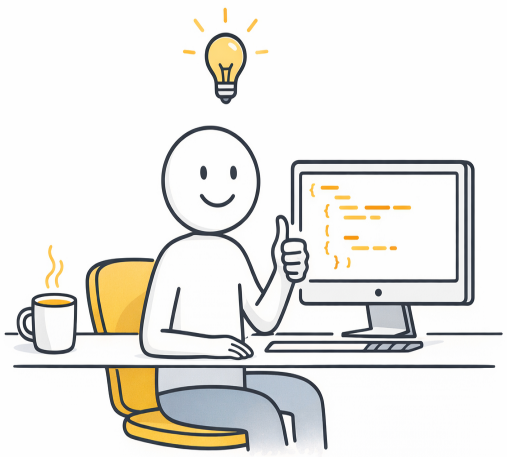


- Right, deep breath. Here we go.
- A Monad is a Functor that can also "flatten" nested structures
- .SelectMany() in LINQ, .flatMap() in JavaScript
- Promise.then() chains? That's a Monad.
- Optional?.value chaining? Also a Monad.
- You've been using Monads for years. Surprise!

# Putting It All Together (Without Panic)

- Category Theory → The framework for thinking about structure
- Functors → Transform contents, preserve structure
- Monoids → Combine things sensibly
- Monads → Functors that handle nested complexity
- Together: A complete vocabulary for composition
- Like LEGO, but for your code. Rather satisfying, actually.

# What to Actually Do With This



- Use .map() and .flatMap() with confidence now
- Recognize patterns: "Ah, that's a Functor situation"
- Compose small functions into larger pipelines
- Handle errors with Either/Result types (Monads!)
- Impress colleagues with restrained accuracy
- Make better tea. (That last one is optional but recommended.)

# Any Questions?

- (Besides "Why did this take so long to explain?")
- Perfectly reasonable to still feel slightly confused
- Rome wasn't built in a day, and neither was Haskell comprehension
- Resources for further reading in the appendix
- Feel free to pretend you understood everything. We won't tell.

# Thank You for Your Patience


- You've been a wonderful audience.
- Now go forth and map things!
- ■ ■ ■