

# Functional Code in C#

Version 9 and beyond

Oliver Sturm • @olivers[@fosstodon.org] • oliver@oliversturm.com



# Oliver Sturm

- Training Director at DevExpress
- Consultant, trainer, author, software architect and developer for 30 years
- Contact: [oliver@oliversturm.com](mailto:oliver@oliversturm.com)

# 2007 at BASTA Conference: 10 Cool Things You Can Do With C# 3.0

1. Sets
2. Empty anonymous lists
3. Ruby style iterations
4. Control.Invoke
5. Fluent interfaces
6. Dynamic querying
7. Ruby style ranges

9. Specifications

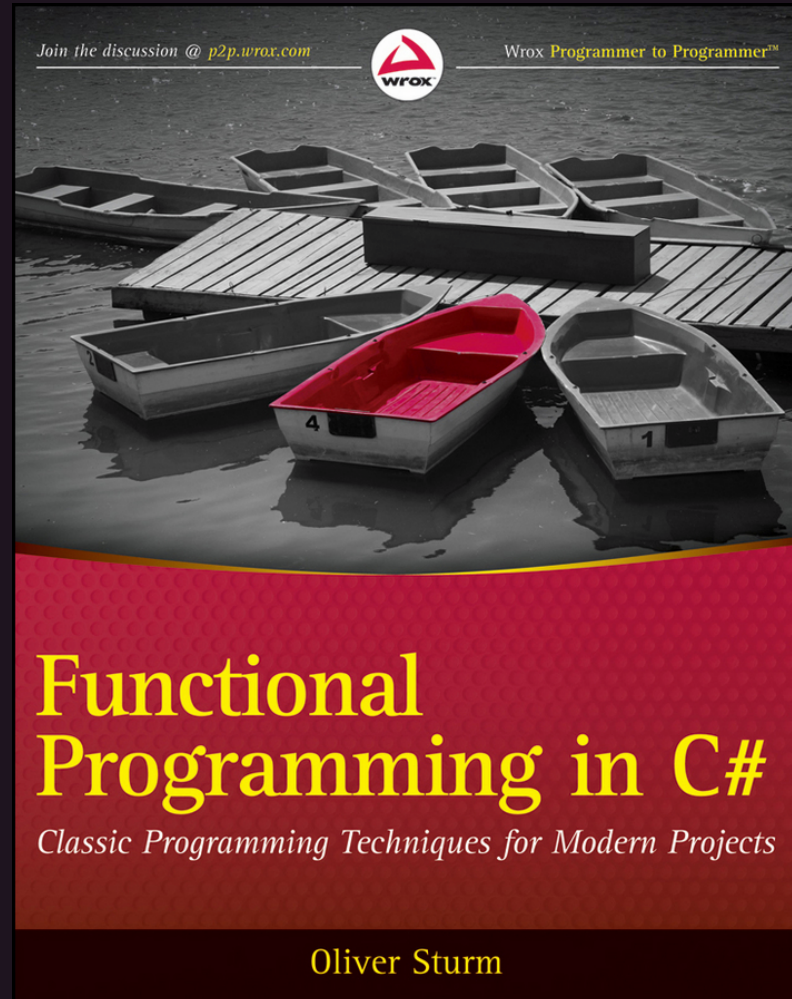
## 10. Smells like Functional Programming

*This simple example demonstrated how to use higher order functions and lambda expressions to do some calculations, instead of imperative techniques.*

# Since then: many more FP in C# talks

- Basics: Lambdas, closures, standard LINQ features
- Higher order functions known from other languages: Map, Filter, Reduce
- Functional Modularization: Currying, Partial Application, Composition
- Concurrency using FP patterns
- Immutable Data
- 2011 Book: Functional Programming in C#
- Accompanying library: FCSlib
- Monads

# Functional Programming in C#, The Book



# Agenda

- What's FP in C# all about?
- New features in C# relevant to FP
- Immutable objects with `record` (cloning)
- Updating FCSlib
- Advanced Pattern Matching

# What's FP in C# all about?

- Applying the style you like to the language you must use - it's a niche!
- Preference to write functional code
- Use functions as reusable blocks
- Learn from other (pure!) FP languages

# Demo

What does FP in C# look like?



# Monads - Interlude

In practical terms, a Monad ...

- is a thing that encapsulates a value and adds some extra information
- allows operations executed against the value, returning a new Monad

```
class Monad<T> {  
    public Monad<T>(T value);  
    public Monad<U> Bind(Func<T, Monad<U>> f);  
}
```

- Example: `Option` (or `Maybe`), `Either`

# Demo (cont.)

What does FP in C# look like?

# New features in C# relevant to FP

- `using static` (6)
- Tuples (7)
- Deconstruction (7)
- Pattern matching (7, really 8 and 9)
- Expression-bodied members (6, 7)
- Throw expressions (7)
- Local functions (7, 8)
- Nullable reference types (8)
- Records (9)

# Demo

A look at records and clones

# What's the point of **record** / **with** ?

- It's all about isolation - different views on generations of data
- Imperative programming is all about change: state modification
- Something to recognize: change is bad
  - Problems when sharing access to data
  - Locking & Co required when parallelizing
  - Friday afternoon debugging problem
- **Idea: let's not change things**
  - **In the real world, this means establishing "views" to create the illusion of change and preserve isolated states**

# Updating FCSlib

In summary, here are important changes I made to the FCSlib code:

- Lots of fixes for nullable reference types. Not sure yet that I've made all the right decisions. E.g. `Option<T>` ?
- Use expression-bodied members where possible
- Implement all tests without the `Functional.` prefix - `using static`
- Considered using records - but no obvious advantage in most FCSlib classes
- Pattern matching. We need to talk about pattern matching.

# Pattern Matching

- C# 7: `value is Type t`, also in `switch`
- C# 8: switch expressions, property, tuple, positional patterns, deconstruction
- C# 9: logical (`and`, `or`, `not`), relational (operators) and structural (parens) enhancements
- C# 10: nested property patterns

Clearly an important feature! Complex expression-based logic implementations would look pretty bad using only ternary expressions!

# Demo

## Pattern Matching



# Sources

- This presentation:  
<https://oliversturm.github.io/functional-cs9>
- PDF download:  
<https://oliversturm.github.io/functional-cs9/slides.pdf>
- Repository with sample code:  
<https://github.com/oliversturm/functional-cs9-samples>
- FCSlib: <https://github.com/oliversturm/FCSlib>

# Thank You

Please feel free to contact me about the content anytime.

Oliver Sturm • @olivers[@fosstodon.org] • oliver@oliversturm.com

