Clarity of Intent

Three Features of F# which Lead to ~~More-Readable~~ BETTER! Code

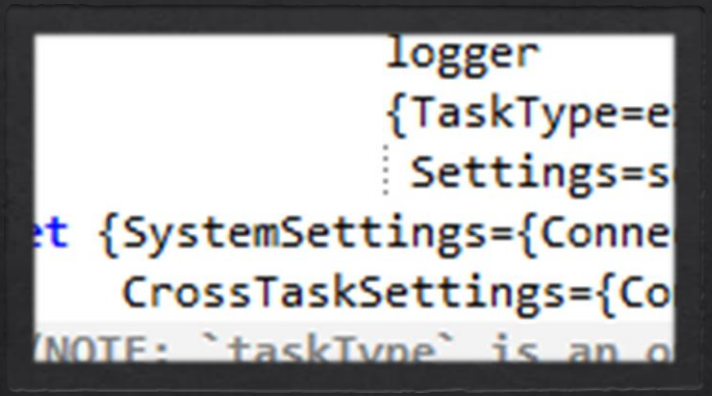
Paulmichael Blasucci (Tuesday, 9 July, 2013)

> Even if we all wrote perfect documentation all of the time, code can hardly be considered reusable if it's not readable.

-- Guido van Rossum, creator of Python

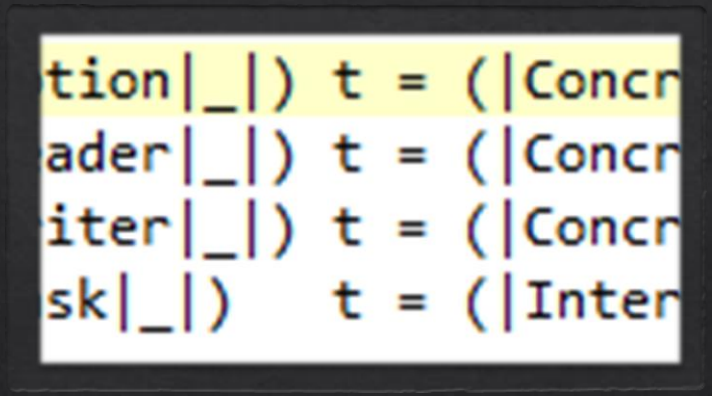
1. Code is read far more often than it is written.
2. Any language other than assembly is for communicating with humans.
3. Nearly all human activity is improved by clear communication.
4. Therefore "better" means: easier to understand the intended purpose and the steps taken to achieve that purpose.

Features: One, Two, and Three.



1. Pattern Decomposition

Unpacking the "shape" of data



2. Active Recognizers

Transforming representations of data

3. Units of Measure

Enhancing the context of data

Pattern Decomposition

- Special application of pattern matching
- Allows one to think in terms of “shapes” of data
- Exemplar of declarative programming style
- Largely symmetrical to data composition
- Easily nested and/or combined with other language features
- Works with most of F#’s composite structures (sorry POCOs)
  - Tuples, Records, Discriminated Unions, Linked Lists, Arrays

Active Recognizers

- Also known as “Active Patterns”
- Conceptually like “views for structured data" [1]
- Plug into match expressions and pattern decomposition
- Are actually special-case functions
  - Thus they may be: curried, composed, and/or partially applied
- Three flavors of recognizers
  - Total Pattern: transforms one shape into another, represents a complete domain
  - Partial Pattern: transforms part of a shape, represents an incomplete domain
  - Parameterized Pattern: like a Total or Partial, but with configuration arguments

Unit-of-Measure Annotations

- Gives meaningful context to rational and integral values
- Addition, subtraction, and comparison require homogenous units
- Multiplication and division give rise to composite (esp. heterogeneous) units
- Units may have static members (useful for constants and conversions)
- Types and operations may be defined in terms of generic units (i.e. placeholders)
- Units are enforced by the compiler – but erased (i.e. not available at run-time)
- Developer must provide any unit conversion logic (e.g. metric to imperial)
- Many common scientific (SI) units defined in core library

Conclusion

Pattern decomposition, active recognizers, and unit-of-measure annotations are three features of F# which lead to better code -- code which more clearly conveys the intentions, suppositions, and directions of it's author.

Errata

More about F#

- http://fsharp.net
- http://fsharp.org
- http://tryfsharp.org
- http://fssnip.net

More about Me

- http://github.com/pblasucci
- http://twitter.com/pblasucci
- http://linkedin.com/in/pblasucci
- http://pblasucci.wordpress.com