Delegate:
Showcase 2/3 years old F# Codebase
@ Prosa 2016-02-23

F#unctional Copenhageners Meetup Group
(MF#K)

Overview – very few slides

- About me
- F#unctional Copenhageners Meetup Group (MF#K)
- How did we start …
- … where are we now …
- … where are we going to
- Showcase Codebase
- Missing syntactic sugar in F#
- Q & A

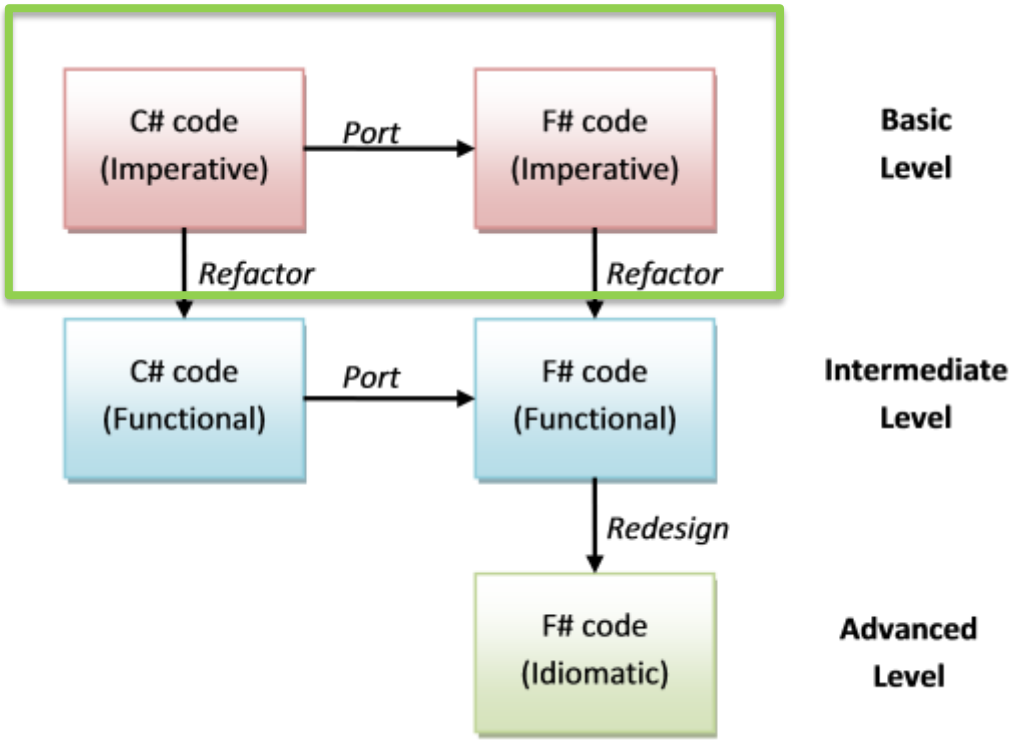
About me (very shortly)

- Ramón Soto Mathiesen
- MSc. Computer Science DIKU and minors in Mathematics HCØ
- Managing Specialist / CTO of CRM Department @ Delegate A/S
  - ER-modeling, WSDL, OData (REST API)
- F# / C# / JavaScript / C++
- Blog: http://blog.stermon.com/

F#unctional Copenhageners Meetup Group (MF#K)

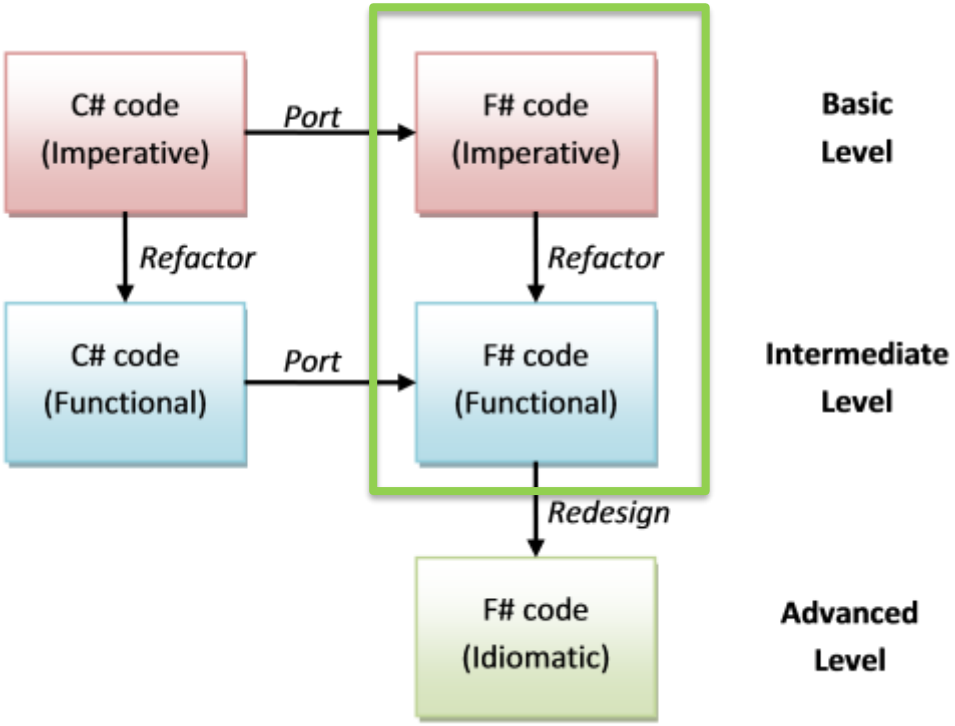
- F#unctional Copenhageners Meetup Group will try to get more and more software projects to be based on functional programming languages.
- We mainly focus on F# and Haskell, but other functional programming languages like Scala, Lisp, Erlang, Clojure, etc. are more than welcome.
- We expect to meet at least twelve times a year (last Tuesday every month), if not more, to share experiences with regards of the use of functional programming languages in software projects that are in / or heading to production.

How did we start …



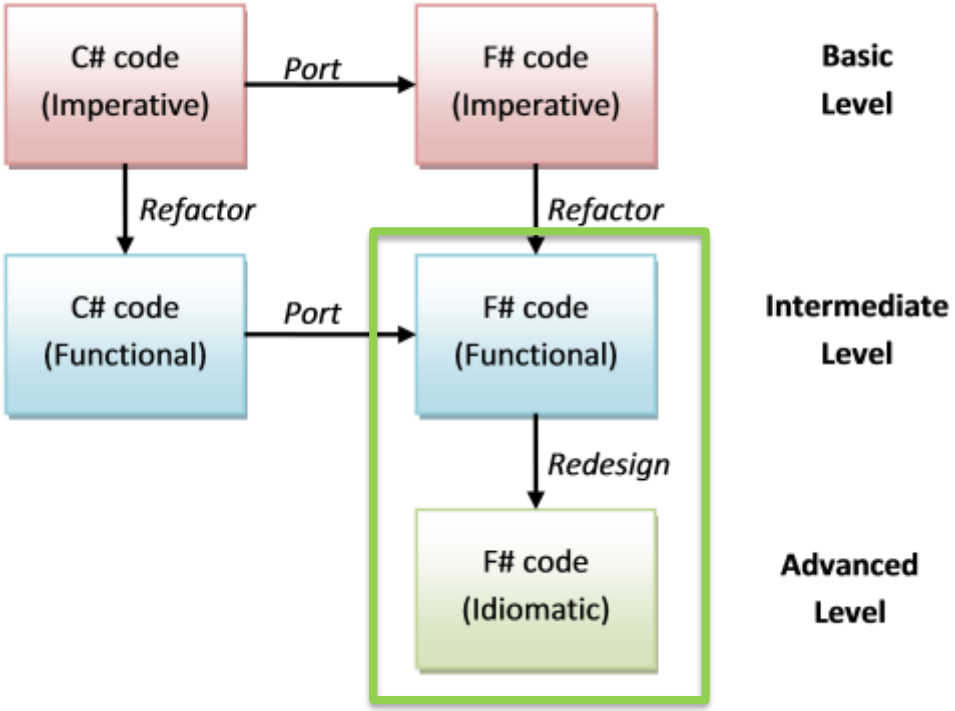
- Porting from C# to F#:
  - http://fsharpforfunandprofit.com/posts/porting-to-csharp-intro/

… where are we now …



- Porting from C# to F#:
  - http://fsharpforfunandprofit.com/posts/porting-to-csharp-intro/

… where are we going to

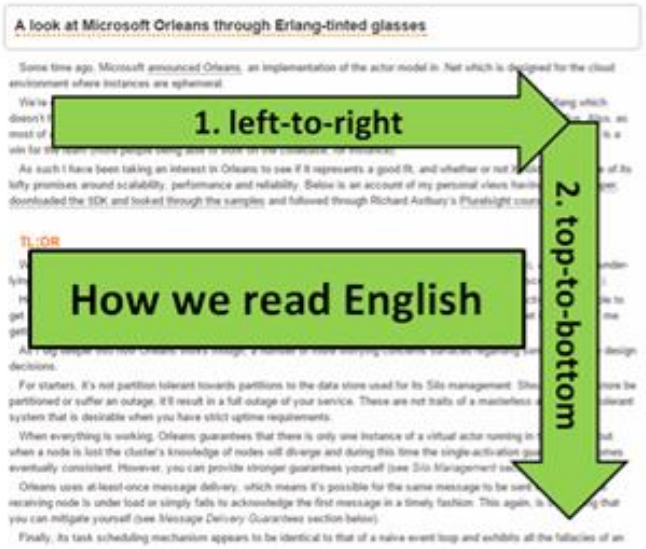


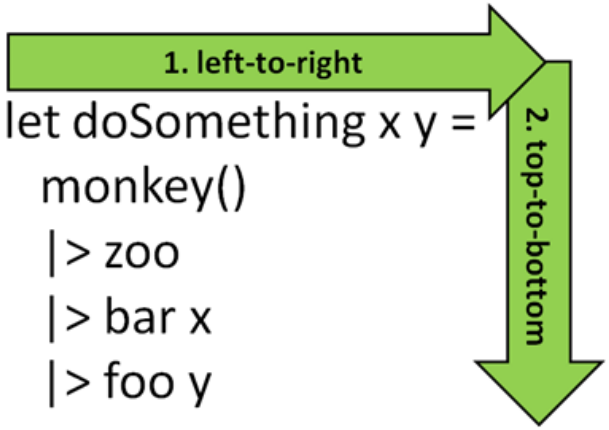
- Porting from C# to F#:
  - http://fsharpforfunandprofit.com/posts/porting-to-csharp-intro/

Showcase Codebase

- Daxif: https://github.com/delegateas/Delegate.Daxif

Showcase Codebase





- How to read F# code?
  - Left-right and top-down (also for projects) but start with interfaces ☺

Missing syntactic sugar in F#

```fsharp
[<AutoOpen>]
module Either =
   type ('a,'b) Either = Choice<'a,'b>
   let Left  x : Either<'a,'b> = Choice1Of2 x
   let Right x : Either<'a,'b> = Choice2Of2 x

   let (|Left|Right|) = function Choice1Of2 x -> Left x | Choice2Of2 x -> Right x
   let bind f = function | Right a -> f a  | Left b -> Left b
   let inline (>>=) m f = bind f m

   let left  = function | Left s -> Some s | Right _ -> None
   let right = function | Left _ -> None   | Right f -> Some f

[<AutoOpen>]
module Result =
   let success x = Either.Right x
   let failure x = Either.Left x
   let succeeded x = Either.right x
   let failed    x = Either.left x
```

- Based on Control.Monad.Either by re-using Choice Type:
  - https://hackage.haskell.org/package/category-extras-0.52.0/docs/Control-Monad-Either.html

Missing syntactic sugar in F#

- This code:

```
let foobar x y =
    success x / y
    failure ex -> ex
```

- Should be syntactic sugar for this:

```
let foobar x y =
    try   x / y     |> success
    with ex -> ex |> failure
```

- Based on Control.Monad.Either by re-using Choice Type:
  - https://hackage.haskell.org/package/category-extras-0.52.0/docs/Control-Monad-Either.html

Missing syntactic sugar in F#

```
List.init 10 (foobar)
|> List.map(fun dividedBy -> dividedBy 1)
|> List.choose(succeeded)

3 |> foobar 22 >>= foobar 33 >>= foobar 44 |> succeeded
```

```
val foobar : x:int -> y:int -> (exn,int) Either
>
val it : int list = [0; 1; 2; 3; 4; 5; 6; 7; 8; 9]
>
val it : int option = Some 11
```

- Based on Control.Monad.Either by re-using Choice Type:
  - https://hackage.haskell.org/package/category-extras-0.52.0/docs/Control-Monad-Either.html

Questions?

U want more?

- Code is available @:
  - https://github.com/delegateas/Delegate.Daxif
- Slides will be available @ MF#K (Files)
- Sign up @ MF#K for:
  - More ***fun***
  - Hands-on:
    - None so far …
  - Talks:
    - In the pipeline talks about:
      - Upcoming: Nothing yet, you want to give a talk?
- MF#K would like to thank our sponsor(s):

Forbundet af It-professionelle

PROSA