TinySelf: Tiny object-oriented language

Working with mutable data in F#

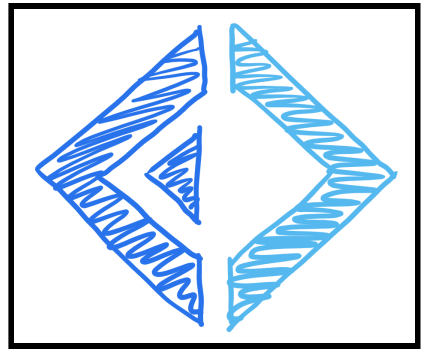
Tomas Petricek, Charles University

tomas@tomasp.net
@tomasp.net
https://tomasp.net
https://d3s.mff.cuni.cz/teaching/nprg077

Mutable records in F#

Defining mutable objects

- Records with **`mutable`** fields
- We could use classes too



Equality and records

- Still use structural equality by default
- Not if records (can) contain functions!
- **`ReferenceEquality`** attribute to override

Charles University

Mutable records

```
type Person =
  { mutable Name : string
    mutable Book : string option }

let setName n p =
  p.Name <- n
let setBook b p =
  p.Book <- Some b

let x = { Name = "Bill"; k = None }
x |> setName "William"
x |> setBook "Alice in Wonderland"

match x with
| { Book = Some book } ->
  printfn "%s likes %s" x.Name book
| _ ->
  printfn "%s is sad :-(" x.Name
```

Helper functions

Make code a bit nicer

Can support **|>** pipe

Pattern matching

Same as immutable

Nice data extraction!

Charles University

Demo

Working with mutable records

TinySelf programming style

Different than before!

Everything is an `Objekt`
Type definition stays
We change what we put in!

Uniformity has drawbacks
Everything type checks!

Demo

TinySelf object visualizer

TinySelf programming style

Different than before!

Everything is an `Objekt`
Type definition stays
We change what we put in!

Uniformity has drawbacks
Everything type checks!

Helper methods

Simplify object construction

```
let makeString s =
  makeDataObject [
    makeParentSlot "parent*"
      stringPrototype
    makeSlot "value"
      (makeSpecial(String s))
    makeAssignmentSlot "value"
  ]
```

Charles University