Десятая независимая научно-практическая конференция «Разработка ПО 2014»

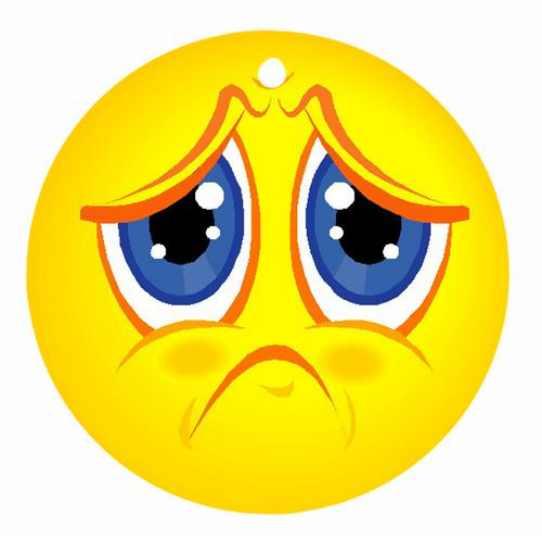
23 - 25 октября, Москва

Реактивное функциональное программирование на F# в увлекательеых примерах

Д.Сошников, Я.Кириленко

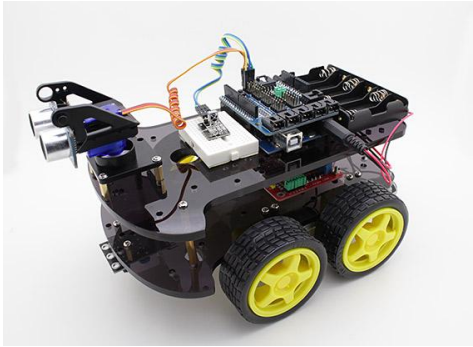
Microsoft/МФТИ/МАИ, СПбГУ

В этой лекции вы не услышите «введение в F#»



В этой лекции вы не услышите про монады и корекурсию

Зато вы увидите живого робота...



О чем хочется поговорить:

- Реактивное программирование в функциональном стиле
- Построение распределенных систем в реактивном стиле
- Примеры…

Реактивное программирование

- Программирование в терминах реакции на события

```
let handle s (ea : MouseEventArgs) =
   if ea.LeftButton=MouseButtonState.Pressed then
       let pos = ea.GetPosition(window.Root)
       Canvas.SetLeft(window.el,pos.X)
       Canvas.SetTop(window.el,pos.Y)

window.cnv.MouseMove.AddHandler(
     MouseEventHandler(handle))
```

```
void OnLoad()
{ cnv.MouseMove += cnv_MouseMove; }

void cnv_MouseMove(object sender, MouseEventArgs e)
{
    if (e.LeftButton==MouseButtonState.Pressed)
    {
        var p = e.GetPosition(cnv);
        Canvas.SetLeft(el, p.X);
        Canvas.SetTop(el, p.Y);
    }
}
```

```
let sub = window.cnv.MouseMove.AsObservable()
          |> Observable.filter(fun x->x.LeftButton=MouseButtonState.Pressed)
          |> Observable.map (fun x-> x.GetPosition(window.Root))
          |> Observable.subscribe (fun pos -> Canvas.SetLeft(window.el,pos.X); Canvas.SetTop(window.el,pos.Y))
```

```
Observable.FromEventPattern<MouseEventArgs>(this,"MouseMove")
    .Where(x=>x.EventArgs.LeftButton==MouseButtonState.Pressed)
    .Select(p => p.EventArgs.GetPosition(cnv))
    .Subscribe(p => { Canvas.SetLeft(el, p.X); Canvas.SetTop(el, p.Y); });
```

Стили обработки данных

Списки (коллекции)	[1..1000] |> List.map fib |> List.filter (fun x->x%3=0) |> List.head	In Memory
Ленивые последова-тельности (IEnumerable)	seq {1..1000} |> Seq.map fib |> Seq.filter (fun x->x%3=0) | Seq.head ReadLines "huge.txt" |> Seq.filter (fun s->s.Contains("hi")) |> Seq.length	Pull ∞
События (реактивное программи-рование)	ConsoleRead() |> Event.filter (fun s->s.Contains("hi")) |> Event.scan (fun acc _ -> acc+1) 0 |> Event.add (printfn "%d")	Push ∞

Конкурс:

Как коротко (без использования fun) записать условие в выражении Seq.filter (fun x -> x%3=0)

Практический пример: мера доброты

```
ReadLines “Война_и_мир.txt”
     |> Seq.filter (fun s -> s<>"")
     |> Seq.scan (fun (ch,s) x ->
                      if x.StartsWith("CHAPTER") then (x,s) else (ch,x)) ("","")
     |> Seq.map (fun (ch,s) -> (ch,MoodAnalyzer.Weight s))
     |> Seq.groupBy fst
     |> Seq.map (fun (ch,wt) -> (ch,wt |> Seq.map snd |> Seq.sum))
     |> FSharpChart.Bar
```

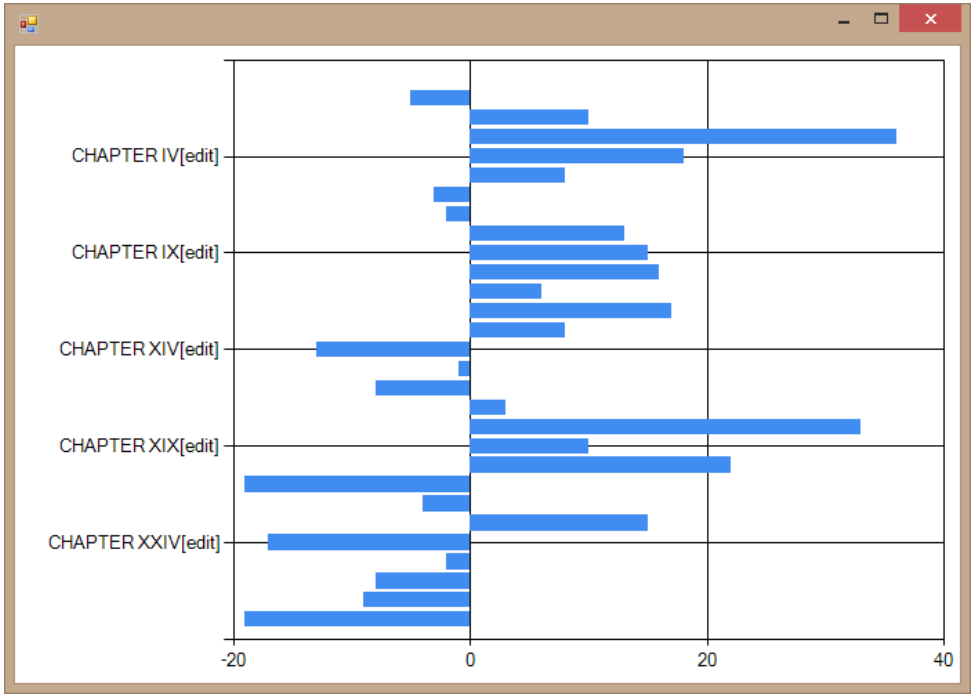
```
let ReadSet fn =
   ReadLines fn |> Seq.fold (swap Set.add) Set.empty

let Positive = ReadSet @"positive.txt"
let Negative = ReadSet @"negative.txt"

let (|Positive|Negative|Neutral|) w =
  if Set.contains w PositiveWords then Positive
  elif Set.contains w NegativeWords then Negative
  else Neutral
```

```
let Weight (s:string) =
   s.ToLower().GoodSplit()
    |> Array.fold (fun acc x ->
                     match x with
                      | Positive -> acc+1
                      | Negative -> acc-1
                      | Neutral -> acc) 0
```

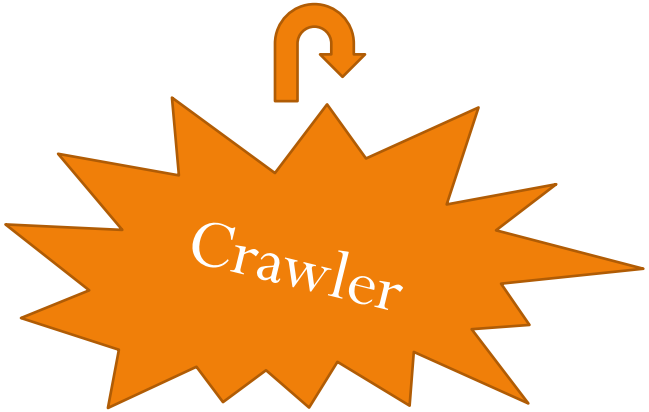
Вывод:

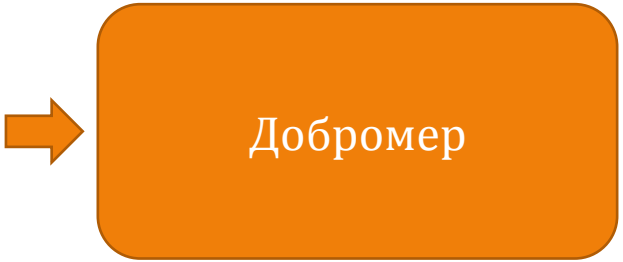
Война и Мир – добрый роман!

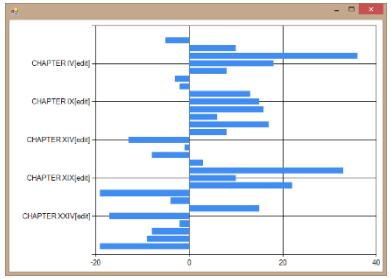
(по крайней мере в переводе на английский)

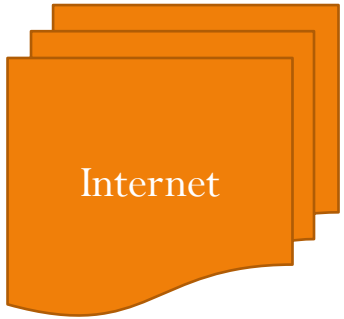


Индексируем интернет









Индексируем интернет

```
let Crawl url =
        async {
            let! s = httpAsync url
            return (s,s.Split('"') |> Seq.filter (fun s-> s.StartsWith("http://") && (s.EndsWith("html")||s.EndsWith("/"))))
            }
```

```
type Crawler() =
    let event = new Event<_>()
    let rec CrawlerAgent =
        MailboxProcessor.Start(fun inbox ->
        let rec loop() = async {
            let! url = inbox.Receive()
            let! (cont,urls) = Crawl url
            event.Trigger (url,cont)
            urls |> Seq.iter (fun x->CrawlerAgent.Post x)
            if (not Async.DefaultCancellationToken.IsCancellationRequested)
            then return! loop()
        }
        loop()
        )
    member this.StartIndexing(url) = CrawlerAgent.Post(url)
    member this.StopIndexing() = Async.CancelDefaultToken()
    member this.NewFeed = event.Publish
```

```
#load @"MoodAnalyzer.fsx"

let cr = new Crawler()

cr.StartIndexing("http://yandex.ru")
cr.NewFeed
 |> Event.map (fun (url,cont) -> MoodAnalyzer.Weight cont)
 |> FSharpChart.FastLine

cr.StopIndexing()
```

Анатомия Event

- let evt = new Event<T>()
- evt.Publish – интерфейс для подписки на события
- evt.Publish |> Event.map f |> Event.filter p
  |> Event.add(fun x -> …)
- evt.Trigger(x : T) – инициирует событие

Интереснее: настроение твиттера

```
type TwitterStreamSample(token,secret) =
    let event = new Event<_>()
    let listener =
        async {
                let req = WebRequest.Create(streamSampleUrl) // more OAUTH magic
                use! resp = req.AsyncGetResponse()
                use stream = resp.GetResponseStream()
                use reader = new StreamReader(stream)
                while ((not reader.EndOfStream)&&(not IsCancellationRequested)) do
                    let z = reader.ReadLine()
                    event.Trigger text
                }
    member this.StartListening() = Async.Start(listener)
    member this.StopListening() = Async.CancelDefaultToken()
    member this.NewFeed = event.Publish
```

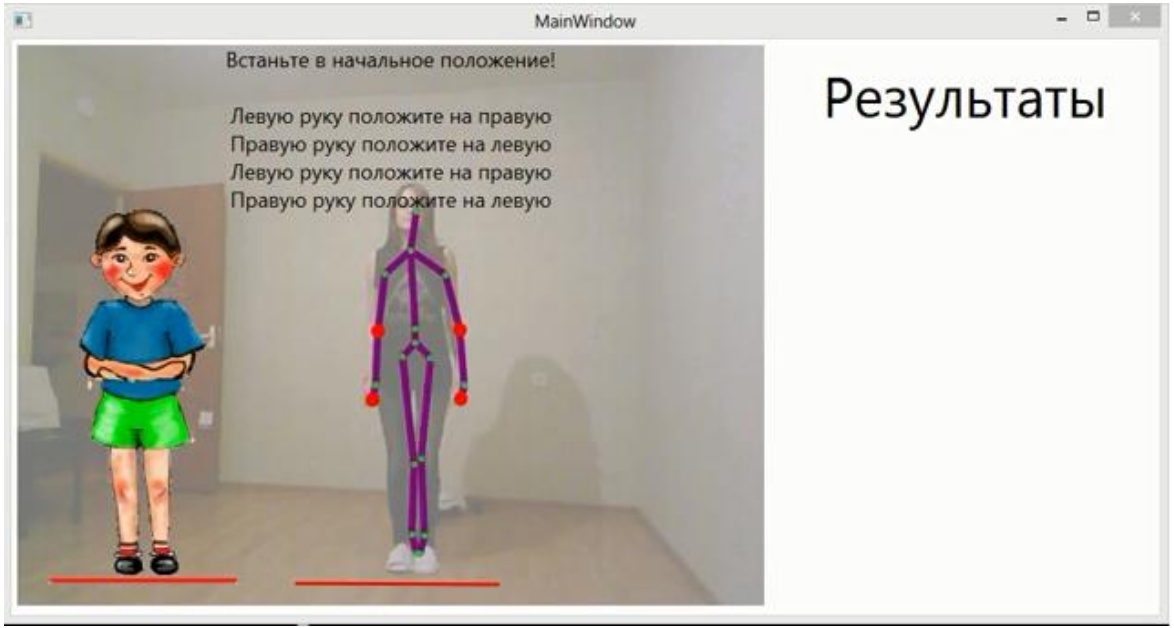
Событийная работа с NUI-контроллерами

- NUI-контроллеры Kinect, Leap Motion – выдают информацию по кадрам, 30fps

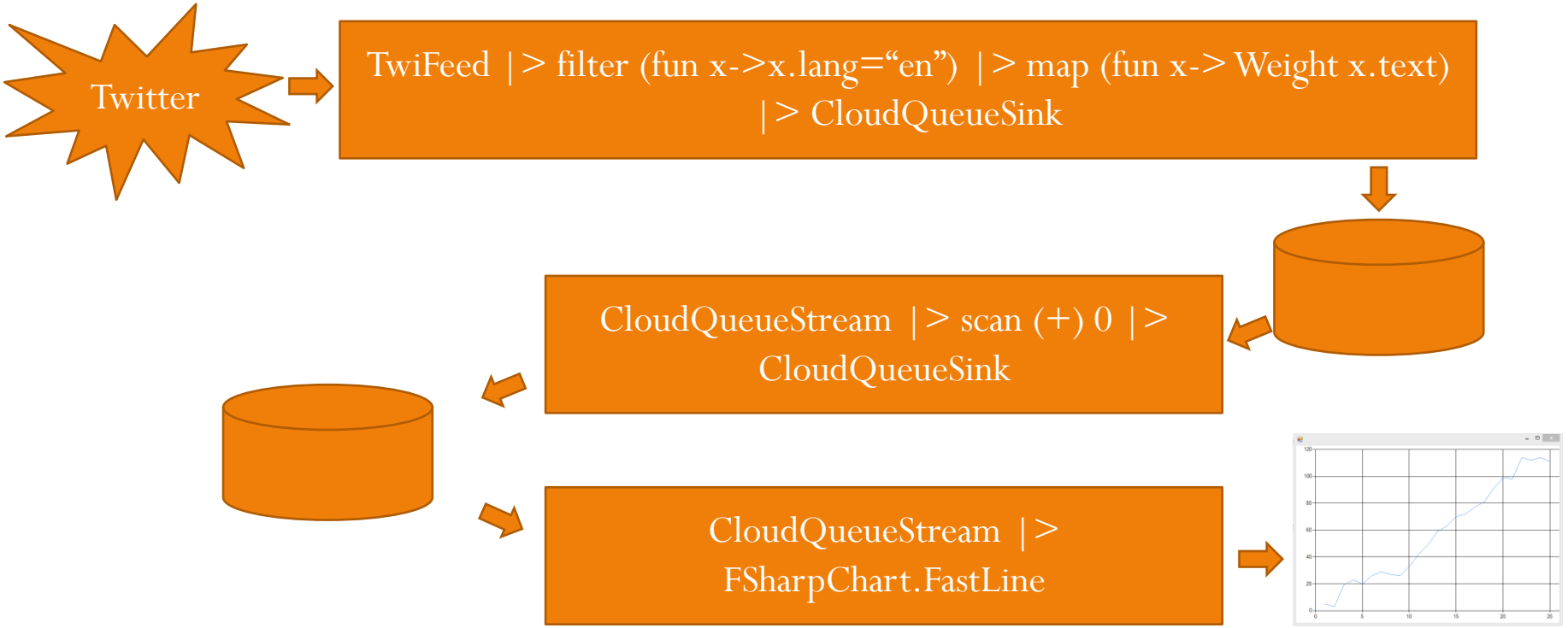
```
type LeapMotion() =
    let event = new Event<_>()
    let cntr = new Leap.Controller()
    let rec loop = async {
                    let f = cntr.Frame()
                    post (fun()->event.Trigger(f))
                    do! Async.Sleep(50)
                    if (not IsCancellationRequested) then return! loop
           }
    member this.EventFeed = event.Publish
    member this.Start() = Async.Start loop
    member this.Stop() = Async.CancelDefaultToken()
```

Использование в реальной жизни

- Совместный проект Российского государственного университета физической культуры, спорта, молодёжи и туризма и лаборатории MAILabs
- Оценка показателей здоровья и физического развития детей при помощи Kinect



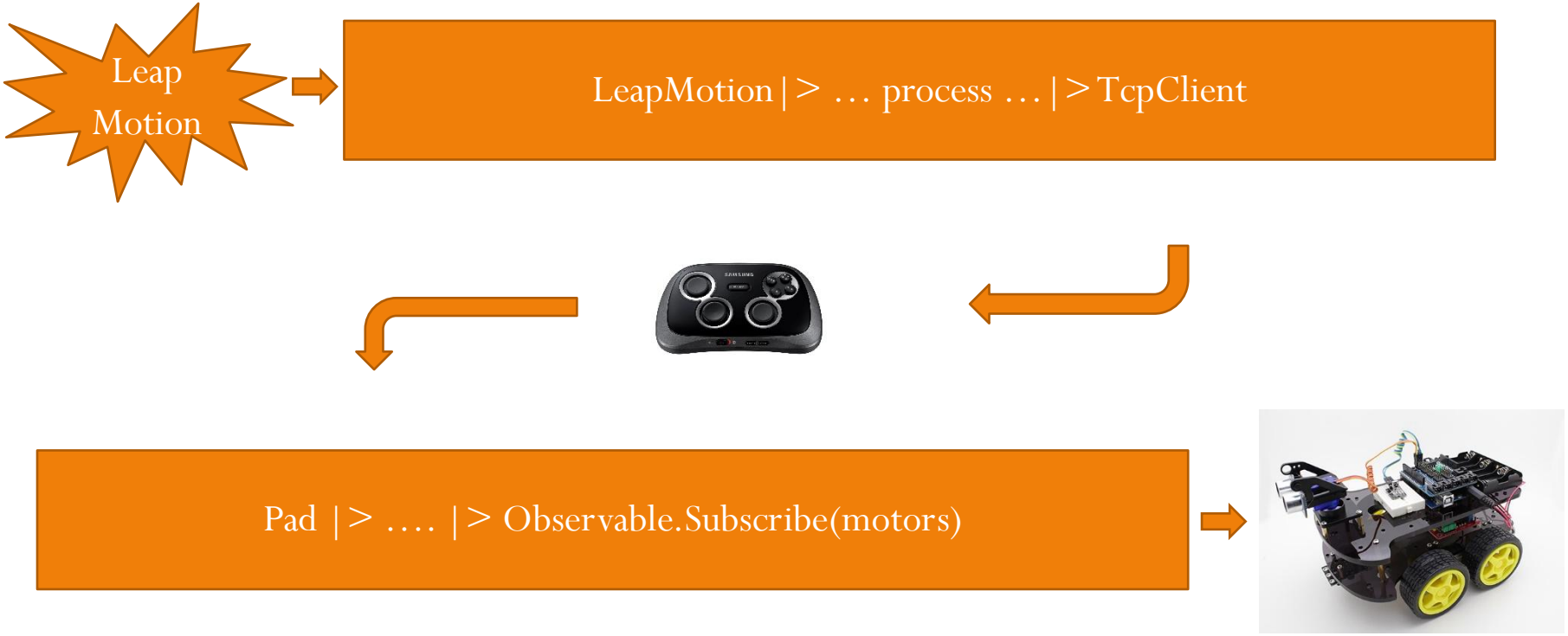
Распределенное реактивное программирование в облаке



Реактивное программирование в .NET: Reactive Extensions

- IObserver<T>
  - void OnNext(T)
  - void OnError(Exception exn)
  - void OnCompleted()
- IObservable<T>
  - IDisposable Subscribe(IObserver O)
- В Observable можно оборачивать списки (IEnumerable), события и другие привычные объекты

Пример: управление роботом



Платформа: ТРИК

- http://trikset.com
- Разработка на базе кафедры системного программирования СПбГУ
- Мощный контроллер, допускающий .NET Framework, WiFi
- Обвязка на базе стандартного конструктора
- Полный стек технологий для обучения:
  - Визуальная среда программирования
  - .NET (C#, F#)
  - Низкий уровень (C)

Программируем робота на F#

```
use model = new Model()
let motorL = model.Motor.["M1"]
let motorR = model.Motor.["M2"]
let sensor = model.AnalogSensor[“A1”]
```

```
motorR.SetPower(20)
motorL.SetPower(-20)
```

```
let dist x =
    if x<20 || x>80 then 0
    else
        if x<40 then 40
        elif x>60 then -40
        else 0

let power = sensor.ToObservable().Select(dist).DistinctUntilChanged()
use l = power.Subscribe(motorL)
use r = power.Subscribe(motorR)
```

Интересные темы на будущее

- Использование Signal-R для быстрой передачи сообщений через облако
- «Прозрачные» реактивные конвейеры с переносом части вычислений в облако через технологии типа m-brace
- Учет распределенности в формальной модели реактивного функционального программирования

Мораль:

```
List |> List.map (fun x->x*2) |> List.filter (fun x -> x>0)
```

```
Sequence |> Seq.map (fun x->x*2) |> Seq.filter (fun x -> x>0)
```

```
EventSource|> Event.map (fun x->x*2) |> Event.filter (fun x -> x>0)
  |> Event.add (fun x-> …)
```

F# - высокодекларативный язык

При одинаковом синтаксисе процесс вычисления может сильно отличаться

Это позволяет эффективно обрабатывать большие данные

В том числе в модели реактивного программирования

Реактивное программирование (сравнительно) легко распределяется

При этом:

F# есть в облаке Microsoft Azure

F# есть на клиенте (Kinect, Leap Motion, W8/WP)

F# - open source

F# имеет сильное сообщество (http://fsharp.org)

F# почти не проприетарный

F# используется на практике (даже в России)

F# используется в преподавании в вузах России (МФТИ, НИУ ВШЭ, СПбГУ, НГУ, МАИ, …)

F# - прекрасен!

Используйте F#!

Дмитрий Сошников

dmitri@soshnikov.com
facebook.com/shwars
twitter.com/shwars
vk.com/shwars
blog.soshnikov.com

© 2014 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries.
The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.