Магия F#
для обработки данных: монады, провайдеры типов, и немного машинного обучения

Дмитрий Сошников

Технологический евангелист, Microsoft

dmitryso@microsoft.com @shwars

.NEXT

Обо мне

> Microsoft
> МФТИ, НИУ ВШЭ, МАИ

> Автор книги: Функциональное программирование на F#

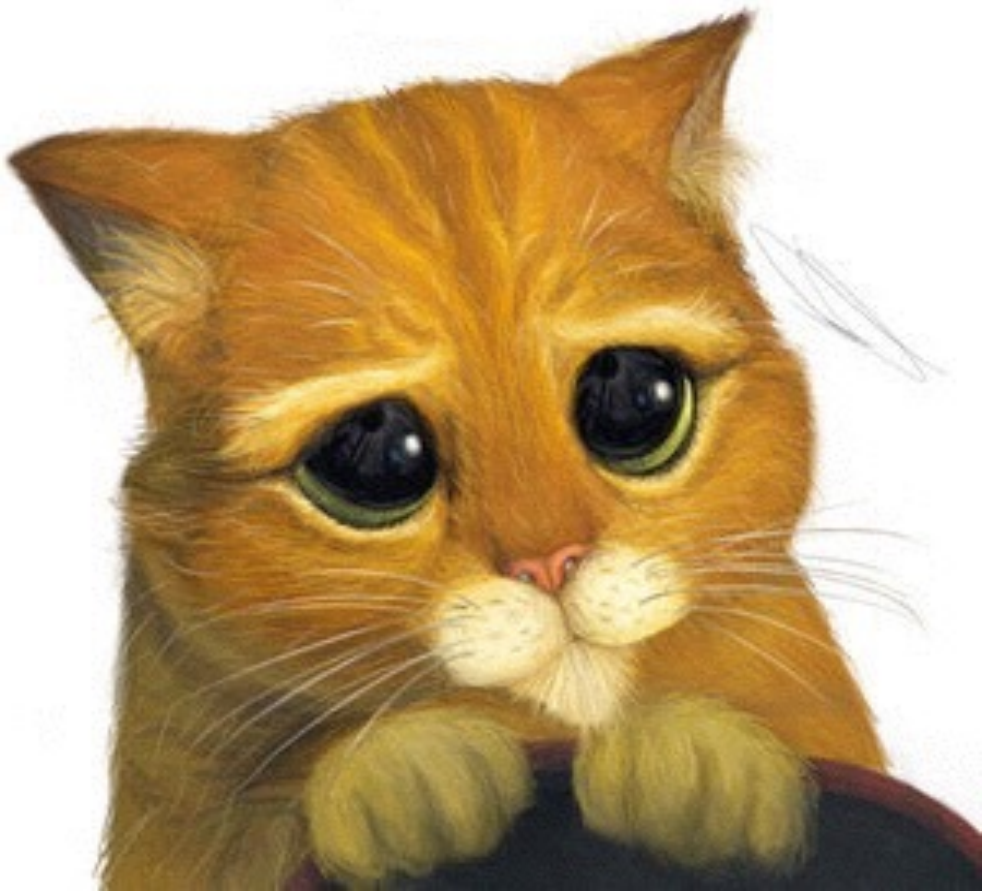
Вдохновление для этого доклада

F#

Доклад по F# на DevCon 2014, 2015 – лучший доклад конференции…

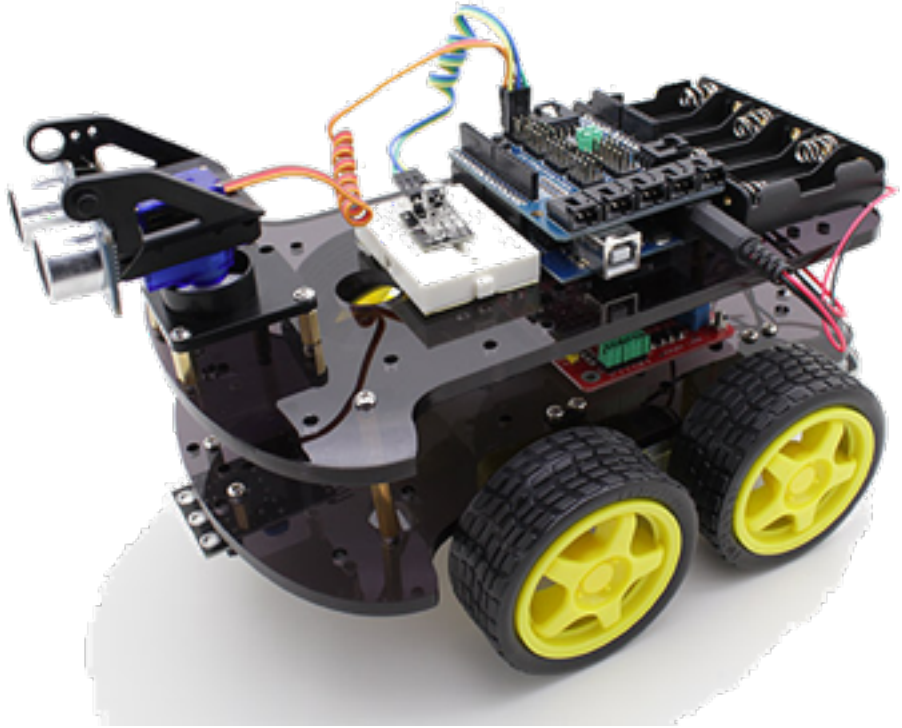
«Используйте F#, или вы сделаете котёнка несчастным…»

Д.Сошников



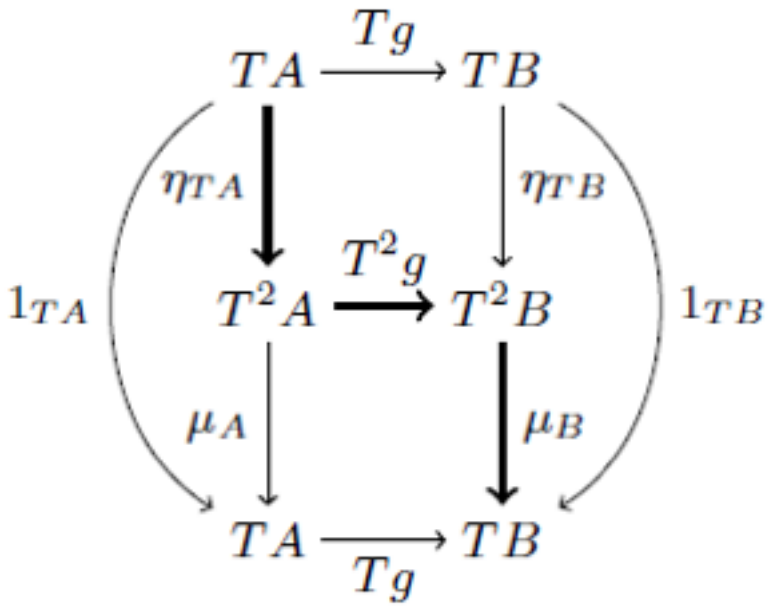
«Что это за презентация без котиков и роботов…»

Д.Сошников, 2014



«Что это за презентация без котиков и монад…»

Д.Сошников, 2015



Выводы

- Презентации с котиками всем нравятся
- Презентации с роботами всем нравятся
- F# интересен сообществу разработчиков

Гипотезы

- Презентации с монадами всем нравятся
- Презентации с демонстрациями всем нравятся
- Не всё ещё используют F# в своём инструментарии, и с этим надо что-то делать…

Что вы могли не знать про F#

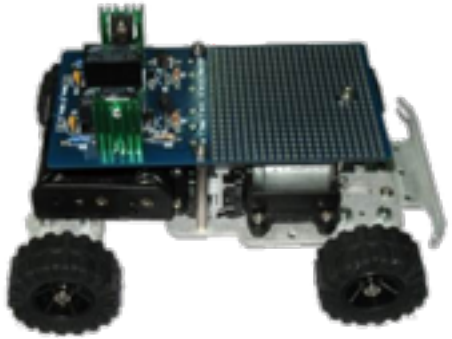
Постараемся не больше, чем 3 минуты…

.NEXT

since 2010…

F# - Open Source

http://fsharp.org



Microsoft Azure

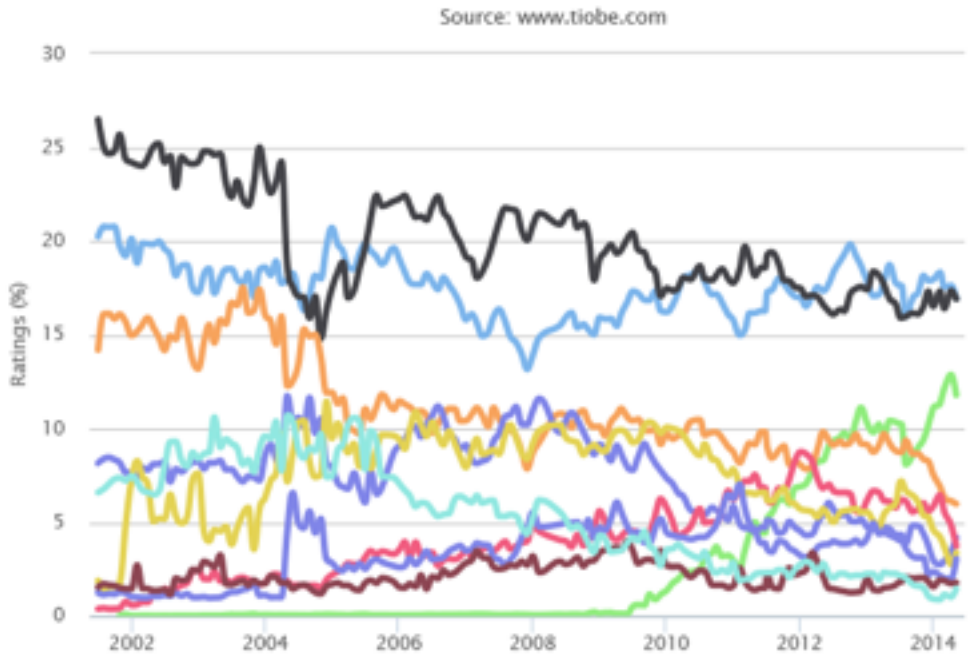
Certified

F# - самый популярный функциональный язык программирования

TIOBE Programming Language Index

F# - #16
Lisp - #23
ML - #27
Scala - #31
Scheme - #37
Haskell - #38
Erlang - #42

F# - самый популярный функциональный язык программирования
F# - самый быстро-растущий по популярности язык
F# - также популярен, как Паскаль/Delphi



F# может сломать вам мозг...

Как описать факториал без явной рекурсии и без использования переменных?

```
let rec fact n = if n=1 then 1 else n*fact(n-1)
```

```
let fact = y ((<<)(cond ((=)0)(k 1))((<<)(s(*))((>>)(c(-)1))))
```

Что нас ждет сегодня…

Тезисы

1. Богатые функциональные абстракции F# позволяют легко реализовывать сложные вещи

Комбинаторные парсеры vs. lexx/yacc

```
let num = pint32
let list x = (pstring "[" >>. sepBy num (pstring ",") .>> pstring "]") x
CharParsers.run list "[1,2,3,4]“
```

```
let num_ws = pint32 .>> spaces
let str_ws s = pstring s .>> spaces
let list x = (str_ws "[" >>. sepBy num_ws (str_ws ",") .>> str_ws "]") x
CharParsers.run list "[1, 2, 3,4 ]“
```

ЕЯ-интерфейс для черепашки

```
type Command =
  | Forward of int
  | Back of int
  | Left
  | Right
  | Stop
```

```
let s x = pstring x .>> spaces
let rec sl = function
   | [x] -> s x
   | h::t -> s h >>. sl t
let K x _ = x
let num = pint32 .>> spaces
```

```
let cmdsep = s "then" <|> sl ["after";"that"]
let forward = (s "forward" <|> sl ["go";"forward"] |> attempt) >>. (num |>>
Forward)
let back = (s "back" <|> sl ["go";"back"] |> attempt) >>. (num |>> Back)
let left = (s "left" <|> sl ["turn";"left"] |> attempt) |>> (K Left)
let right = (s "right" <|> sl ["turn";"right"]) |>> (K Right)
let stop = (s "stop" <|> sl ["go";"to";"sleep"]) |>> (K Stop)
let simplecommand = (s "please" <|> sl ["i";"beg";"you"] <|> s "")
                    >>. (forward <|> back <|> left <|> right <|> stop)
let command = sepBy simplecommand cmdsep .>> eof

let ParseCommand (s:string) = s.ToLower()
   |> CharParsers.run command
   |> function
       | Success(res,_,_) -> res
       | _ -> []

ParseCommand "please turn right after that forward 10 then i beg you turn left then
```

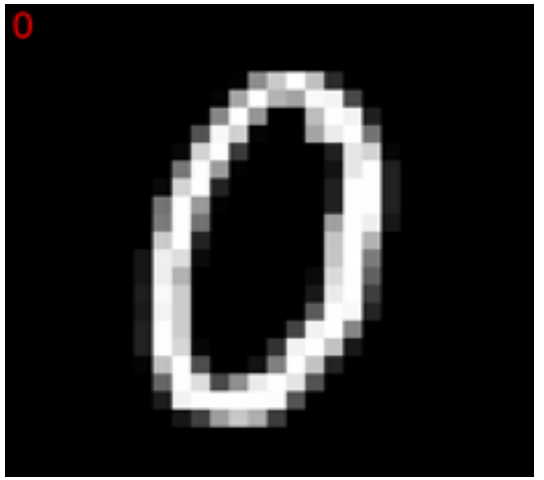
Тезисы

1. Богатые функциональные абстракции F# позволяют легко реализовывать сложные вещи
2. F# - прекрасный язык для обработки данных

Работа с данными и немного Machine Learning

- http://www.kaggle.com/c/digit-recognizer
- Множество примеров рукописных цифр
- Цель – научиться автоматически распознавать цифры
- Обучающая выборка - 50,000 примеров
- Тестовая выборка - 20,000 “неизвестных” цифр

- Пример: https://github.com/shwars/FSharpCodingDojo/tree/master/DigitRecognizer



Что такое классификатор?

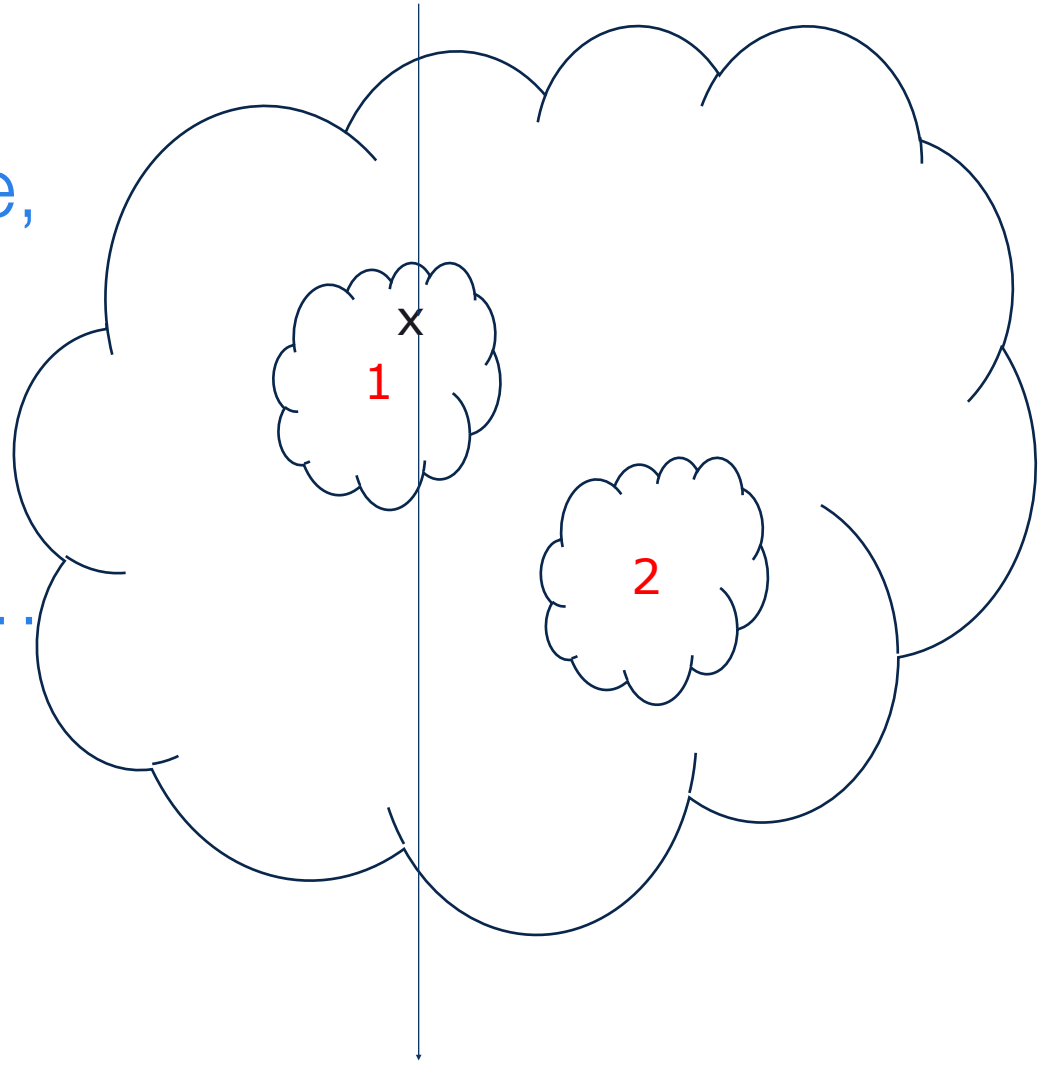
- На вход поступает неизвестная точка в некотором пространстве, на выходе – класс, к которому она принадлежит

  classify(x) = 1

- В нашем случае классы = 0, 1, … 9

- Точка = изображение цифры

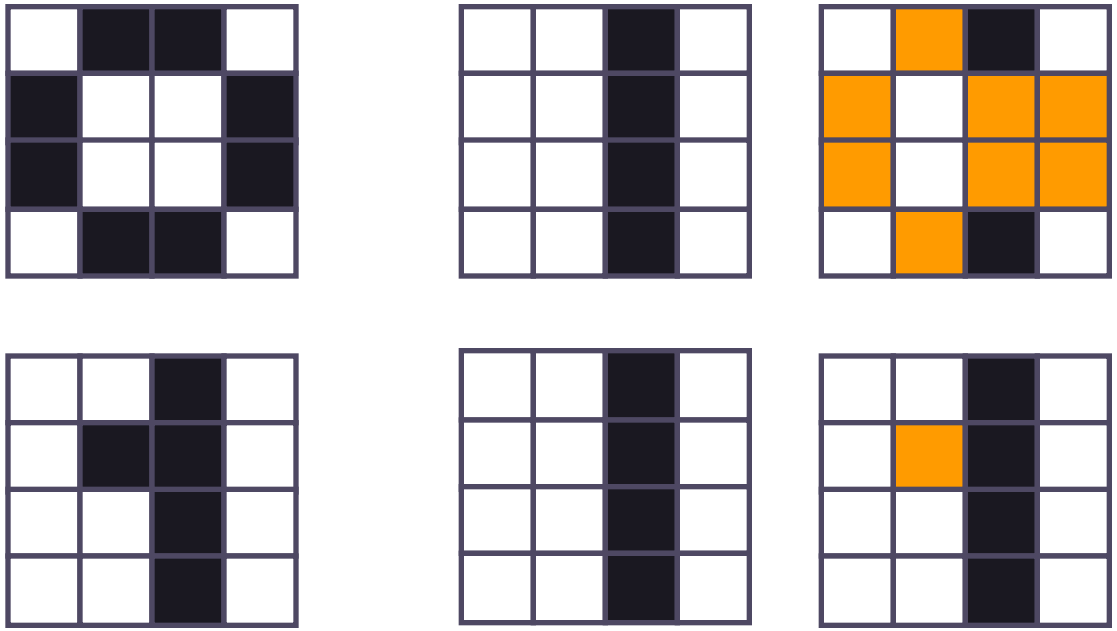
  784-мерное пространство



KNN-классификатор

- KNN = K Nearest Neighbors
- На входе – неизвестная точка
- Смотрим на список всех доступных примеров
- Находим K ближайших соседей
- Предсказываем класс, которые встречается в большинстве этих примеров

Пример: 1-nearest neighbor (2)



$$D = \sqrt{255^2 + 255^2 \ldots + 255^2}$$

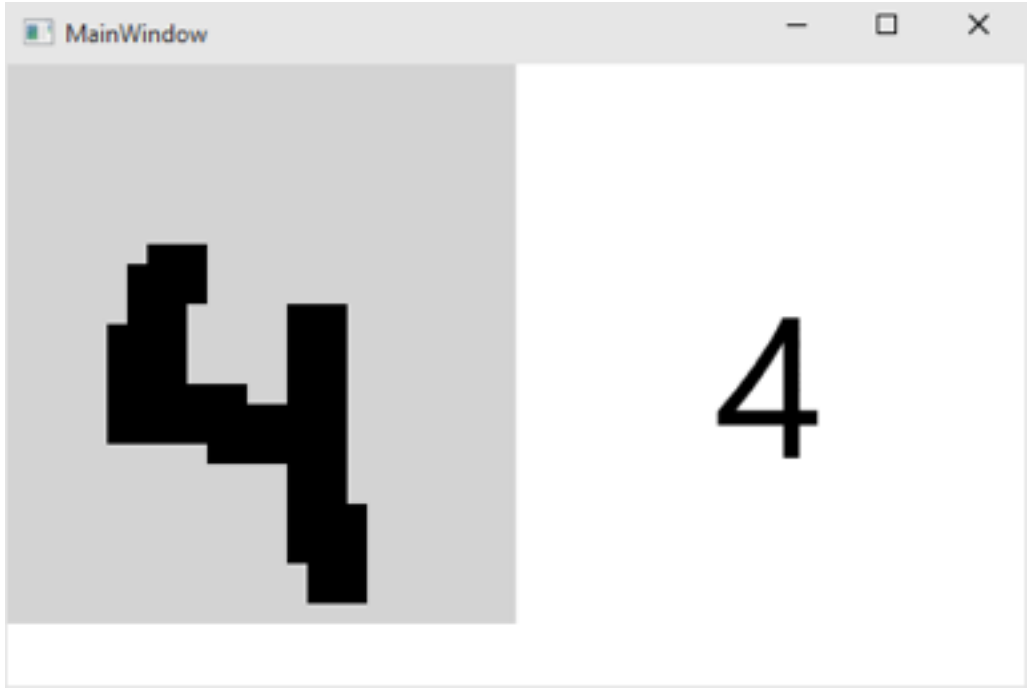
$$D = \sqrt{255^2}$$

Попиксельно сравниваем изображения

Вычислим расстояние между неизвестным и доступными примерами

Тезисы

1. Богатые функциональные абстракции F# позволяют легко реализовывать сложные вещи
2. F# - прекрасный язык для обработки данных
3. F# - прекрасно интероперирует с C# и .NET



Тезисы

1. Богатые функциональные абстракции F# позволяют легко реализовывать сложные вещи
2. F# - прекрасный язык для обработки данных
3. F# - прекрасно интероперирует с C# и .NET
4. F# инновационный и взаимодействует с данными через Type Providers

Type Providers

```
open Microsoft.FSharp.Data.TypeProviders
type crimeData = ODataService<"https://api.datamarket.azure.com/
data.gov/Crimes/v1/">
let context = crimeData.GetDataContext(Credentials=credentials);

let res =
 query {
  for c in context.CityCrime do
  where (c.State="Washington")
  select (c.City,c.Burglary)
 }
```

Тезисы

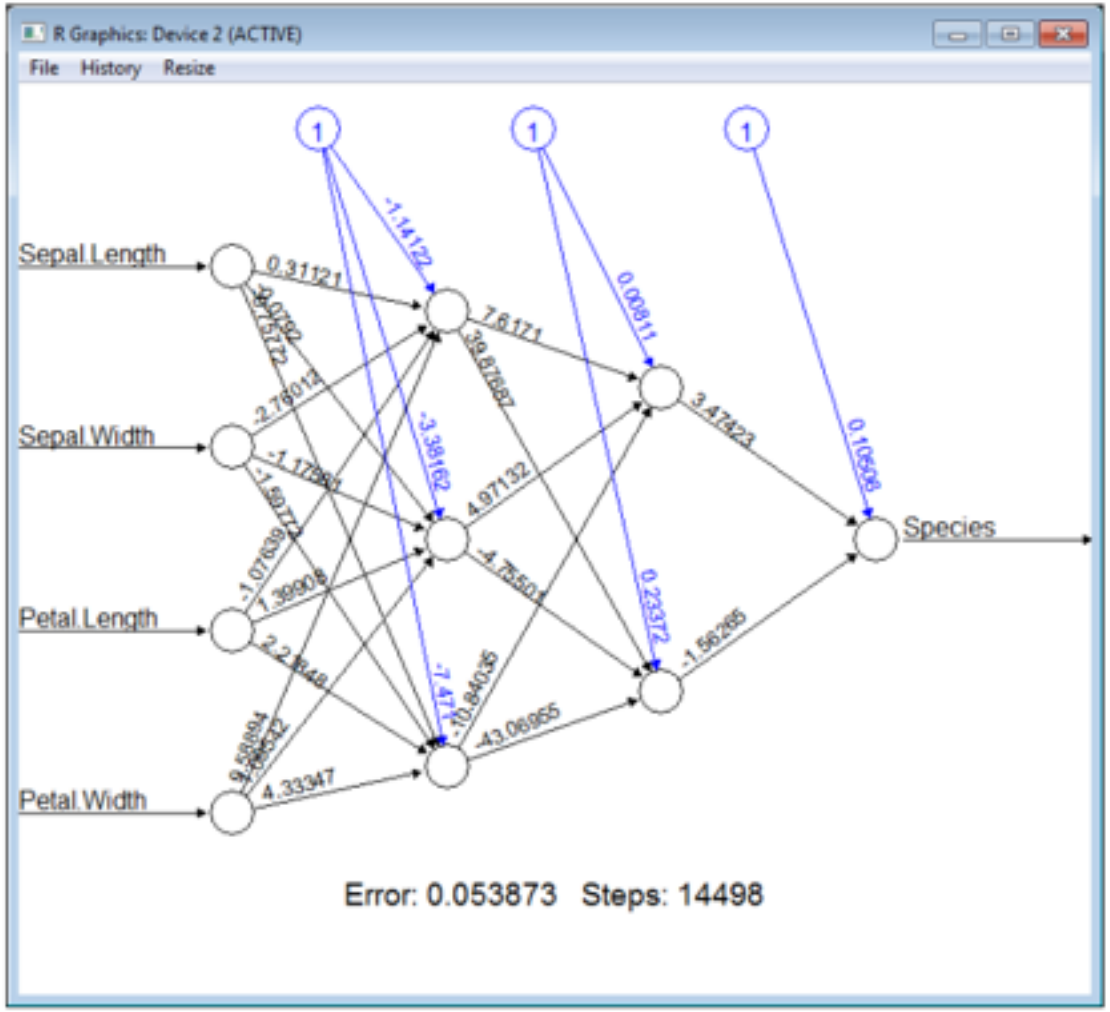
1. Богатые функциональные абстракции F# позволяют легко реализовывать сложные вещи
2. F# - прекрасный язык для обработки данных
3. F# - прекрасно интероперирует с C# и .NET
4. F# инновационный и взаимодействует с данными через Type Providers
5. F# - прекрасен для Data Scientist!

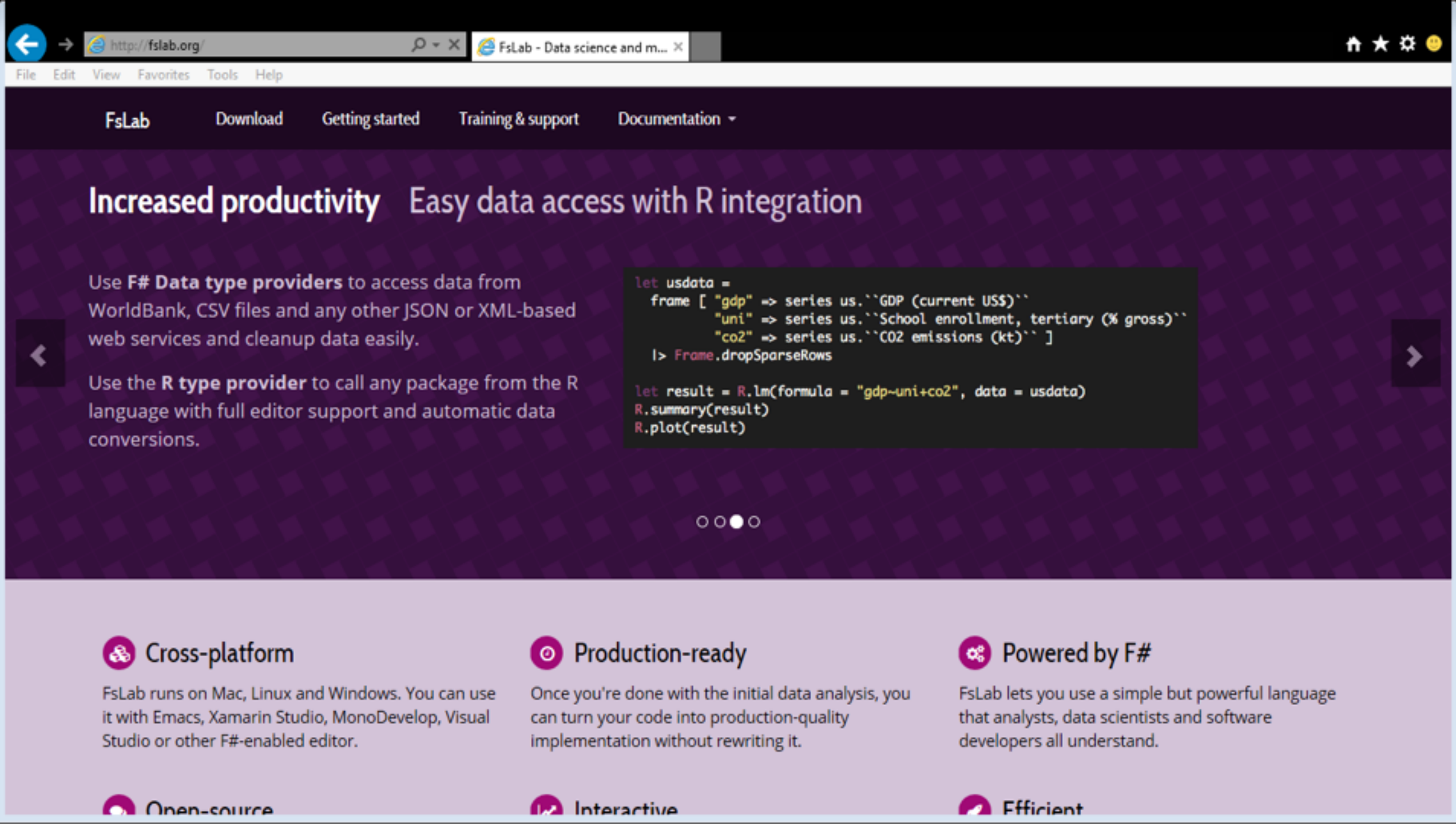
Deedle Frame Library

```
let titanic =
Frame.ReadCsv("titanic.csv").GroupRowsBy<int>("Pclass")

let byClass =
  titanic.GetColumn<bool>("Survived")
  |> Series.applyLevel fst (fun s ->
      series (Seq.countBy id s.Values))
  |> Frame.ofRows
  |> Frame.sortRowsByKey
  |> Frame.indexColsWith ["Died"; "Survived"]
```

R Type Provider




http://fslab.org/
FsLab - Data science and m...
File Edit View Favorites Tools Help
FsLab
Download
Getting started
Training & support
Documentation
Increased productivity
Easy data access with R integration
Use F# Data type providers to access data from WorldBank, CSV files and any other JSON or XML-based web services and cleanup data easily.
Use the R type provider to call any package from the R language with full editor support and automatic data conversions.
let usdata =
frame ["gdp" => series us.``GDP (current US$)``
"uni" => series us.``School enrollment, tertiary (% gross)``
"co2" => series us.``CO2 emissions (kt)``]
|> Frame.dropSparseRows
let result = R.lm(formula = "gdp~uni+co2", data = usdata)
R.summary(result)
R.plot(result)
Cross-platform
FsLab runs on Mac, Linux and Windows. You can use it with Emacs, Xamarin Studio, MonoDevelop, Visual Studio or other F#-enabled editor.
Production-ready
Once you're done with the initial data analysis, you can turn your code into production-quality implementation without rewriting it.
Powered by F#
FsLab lets you use a simple but powerful language that analysts, data scientists and software developers all understand.
Open-source
Interactive
Efficient

Тезисы

1. Богатые функциональные абстракции F# позволяют легко реализовывать сложные вещи
2. F# - прекрасный язык для обработки данных
3. F# - прекрасно интероперирует с C# и .NET
4. F# инновационный и взаимодействует с данными через Type Providers
5. F# - прекрасен для Data Scientist!
6. В F# отлично реализована параллельность и асинхронность

Тезисы

1. Богатые функциональные абстракции F# позволяют легко реализовывать сложные вещи
2. F# - прекрасный язык для обработки данных
3. F# - прекрасно интероперирует с C# и .NET
4. F# инновационный и взаимодействует с данными через Type Providers
5. F# - прекрасен для Data Scientist!
6. В F# отлично реализована параллельность и асинхронность
7. За счет монадических выражений F# позволяет прозрачно переносить вычисления в облако!

{m}brace

http://m-brace.net

Чтобы попробовать:

- Запустить свой кластер на Microsoft Azure с помощью http://briskengine.com
- Скачать и попробовать tutorial: https://github.com/mbraceproject/MBrace.StarterKit/archive/master.zip

Вычисления на кластере

```
let Result =
    [ 32 .. 42 ]
    |> List.map(fun i -> cloud { return (fib i) })
    |> Cloud.Parallel
    |> cluster.Run
```

Parametric Sweep

```
let ParamSweep =
    Parameters
    |> List.map(fun p -> cloud {
          if Success(p) then return Some(p) else return None } )
    |> Cloud.Choice
    |> cluster.CreateProcess
```

CloudFlow

```
let streamComputationJob =
    [| 1..100 |]
    // Convert the data into multiple cloud-parallel data streams
    |> CloudFlow.ofArray

    // Map/filter/map
    |> CloudFlow.map (fun num -> num * num)
    |> CloudFlow.filter (fun num -> num < 2500)
    |> CloudFlow.map (fun num -> if num % 2 = 0 then "Even" else "Odd")

    // Reduce
    |> CloudFlow.countBy id

    // Collect the results back to a single array
    |> CloudFlow.toArray
    |> cluster.Run
```

Финальная демонстрация

Асинхронность + Парсеры + Реактивное программирование

Используйте F#!

Дмитрий Сошников

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

В презентации использованы фотографии Linus Bohman, Martin Biskoping, распространяемые по лицензии Creative Commons

©2015 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics 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.