Succeeding with Functional-First Languages in Industry

Dr. Don Syme
F# Community Contributor,
Principal Researcher, Microsoft
@dsyme

Today: Some Simple Observations About

Data Engineering
Data Pipelines
Analytical Programming
Game Server Engines
Calculation Engines
Coding …

with

F#
(and associated technologies)

Let's focus on the industry perspective and its correspondence to the technical features of F#

Based on informal observations of many successful F# adoptions

fsharp.org/testimonials

also observations from OCaml, Scala, Erlang…

F# is free, open source, cross platform, independent

fsharp.org

F# for Android

http://fsharp.org/use/android

F# for iOS

http://fsharp.org/use/ios

F# in Emacs

https://github.com/fsharp/fsharpbinding/

Part 1

What's the Situation? What's the Problem?

I will use a standard methodology for communicating “complex” products.

S	• Situation
P	• Problem
I	• Implication
N™	• Need

“SPIN Selling”, Rackham

The Recurring Business Situation

“I lead a team developing…”

- Data Processing Pipelines
- Insurance Calculation Engines
- Service Implementations
- Trading Platforms
- Market Simulators
- Server-side Game Engines
- …

The Recurring Business Situation

“I lead a team developing…”

- Analytical Components
- Analytical Services
- Analytical Components
- Analytical Services
- Analytical Components
- …

Understanding the Situation

Data Engineers

Data
Information
Services

Analytical Programmers/ Data Scientists

Code
Analysis
Algorithms
Parallel

Design

Presentation
Publication
UI

The Recurring Business Problems

Time to Market

Efficiency

Correctness

Complexity

- for analytical components

Is Time to Market a Problem?

Late Models → Missed market opportunities

Financial model

Late Services → Users have gone elsewhere

Gaming service

Late Components → Millions evaporate

Ad ranking engine

Is Correctness a Problem?

Buggy Models → Major risks to institutions

Quant model

Buggy Services → Users walk away

Gaming service

Buggy Analytical Components → Millions leak away

Ad ranking engine

Is Efficiency a Problem?

Slow Models → Can’t assess the institution daily

Financial model

Slow Services → Massive loss of online business

Insurance quote service

Slow Analytical Components → Can’t scale to web

Ad ranking engine

Is Complexity a Problem?

Intractable Models → Can’t enter markets

Intractable Services → Can’t deliver services

Intractable Analytical Components → Can’t deliver

The Recurring Business Problems

Time to Market

Efficiency

Correctness

Complexity

- for analytical components and services

What's the Need?

Analytical programmers delivering **correct**, **efficient components** in the enterprise, **on-time**

This is one set of problems that functional-first programming helps solve

Part 2 - Why?

Observations and Examples

Observation #1

At the core of every functional-first language is this:

simple, correct, robust code for solving **complex problems**

Observation #2

A highly interoperable language allows **rapid, non-intrusive deployment** and **integration** of components

… functional-first code is a part of a larger solution. With F# your code can be rapidly integrated and deployed.

Observation #2 cont.

Interoperable languages remove **entire phases** from the analytical software development process.

...no R → C#
...no Mathematica → C++
...no Excel → Java

Observation #3

Strongly-typed functional-first languages maintain efficiency

...as good as C# and Java, and sometimes C++

Observation #4

Strongly-typed functional languages help analytical programmers tackle more complex problems

…more time in the domain, less time on nulls and object hierarchies.

Recap – How Functional-first Helps

Simple, correct, robust code

Interoperability eliminates entire phases

Strong-typing gives efficiency

Analytical developers empowered to solve complex problems

Example #1 (power company)

I have written an application to balance the national power generation schedule … for an energy company.

…the calculation engine was written in F#.

The use of F# to address the complexity at the heart of this application clearly demonstrates a sweet spot for the language … algorithmic analysis of large data sets.

Simon Cousins (Eon Powergen)

Example #1 (power company)

Time to Market

Efficiency

Interoperation ... Seamless. The C# programmer need never know.

Parallelism ...The functional purity ... makes it ripe for exploiting **the inherent parallelism in processing vectors of data**.

Correctness

Units of measure ... a huge time saver...it **eradicates a whole class of errors**.

Time to Market

Code reduction... ... ve matrices...higher order functions eat these for breakfast with **minimal fuss, minimal code. Beautiful.**

Time to Market

Exploratory programming ...Working with F# Interactive allowed me to **explore the solution space more effectively**.

Correctness

Lack of bugs... Functiona feel strange. .. once the type checker is satisfied **that's often it, it works**.

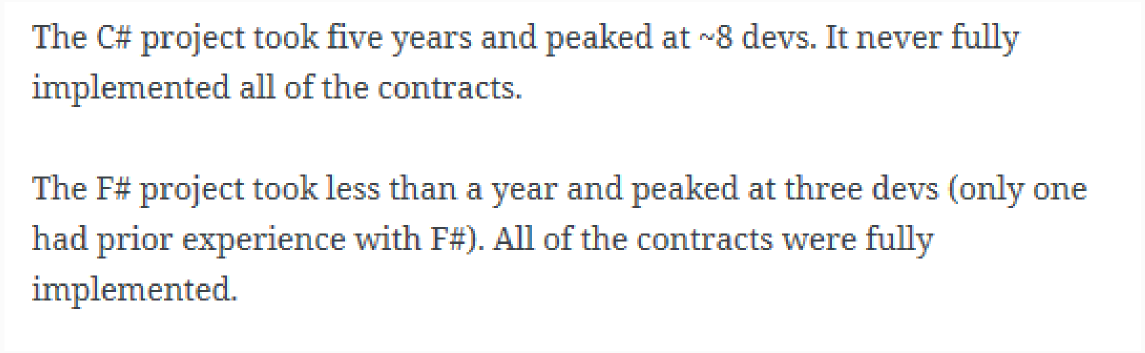
Correctness

Unit testing ...a joy to test. **There are no complex time-dependent interactions to screw things up....**

A related analysis (Simon Cousins, Energy Sector)

350,000

lines of C# OO
by offshore team



The C# project took five years and peaked at ~8 devs. It never fully implemented all of the contracts.

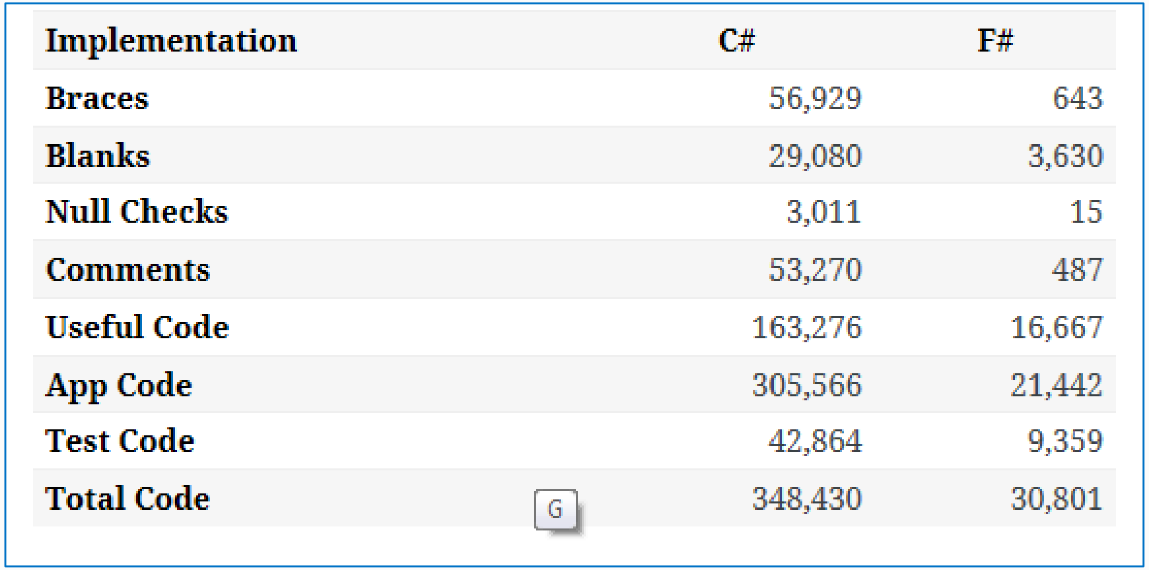
The F# project took less than a year and peaked at three devs (only one had prior experience with F#). All of the contracts were fully implemented.

30,000

lines of robust F#, with
parallel +more features

An application to evaluate the revenue due from Balancing Services contracts in the UK energy industry

http://simontcousins.azurewebsites.net/does-the-language-you-use-make-a-difference-revisited/



Implementation	C#	F#
Braces	56,929	643
Blanks	29,080	3,630
Null Checks	3,011	15
Comments	53,270	487
Useful Code	163,276	16,667
App Code	305,566	21,442
Test Code	42,864	9,359
Total Code	348,430	30,801

A related analysis (Simon Cousins, Energy Sector)

Zero

bugs in deployed system

"F# is the safe choice for this project, any other choice is too risky"

An application to evaluate the revenue due from Balancing Services contracts in the UK energy industry

http://simontcousins.azurewebsites.net/does-the-language-you-use-make-a-difference-revisited/

Example #2: F# in Finance

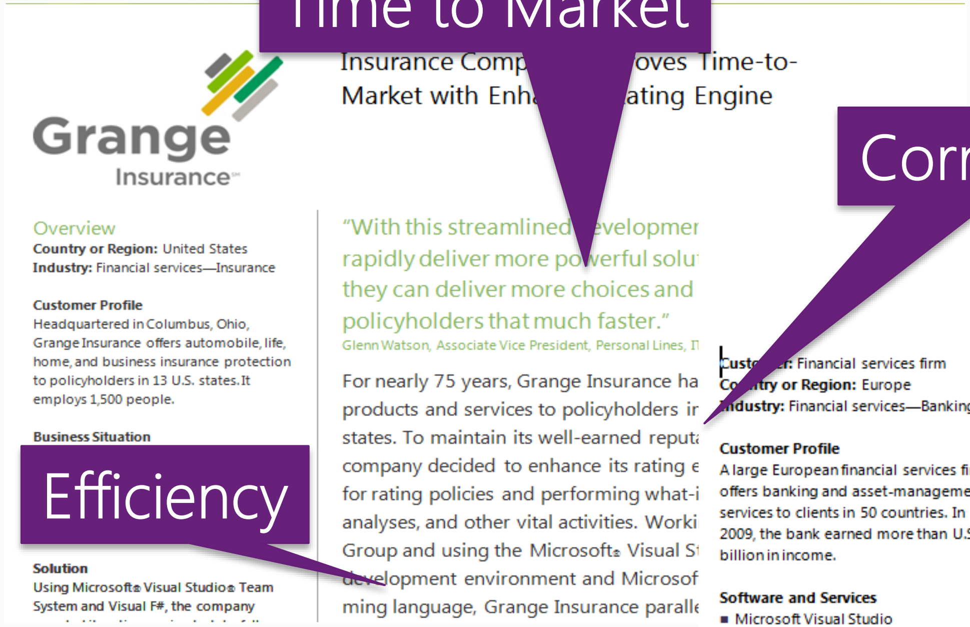
Time to Market

Correctness

Time to Market

Efficiency

Correctness



Grange Insurance℠

Insurance Comp oves Time-to-Market with Enh ating Engine

Overview
Country or Region: United States
Industry: Financial services—Insurance

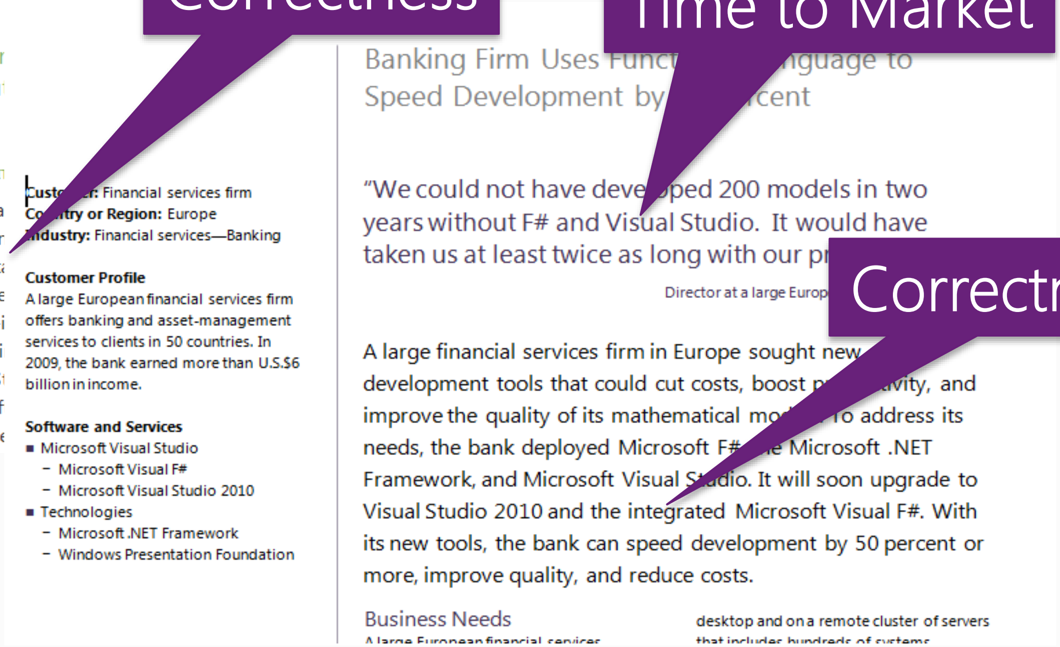
Customer Profile
Headquartered in Columbus, Ohio, Grange Insurance offers automobile, life, home, and business insurance protection to policyholders in 13 U.S. states. It employs 1,500 people.

Business Situation

Solution
Using Microsoft® Visual Studio® Team System and Visual F#, the company

"With this streamlined evelopmer rapidly deliver more powerful solut they can deliver more choices and policyholders that much faster."
Glenn Watson, Associate Vice President, Personal Lines, I

For nearly 75 years, Grange Insurance ha products and services to policyholders ir states. To maintain its well-earned reputa company decided to enhance its rating e for rating policies and performing what-i analyses, and other vital activities. Worki Group and using the Microsoft® Visual St development environment and Microsof ming language, Grange Insurance paralle



Banking Firm Uses Funct nguage to Speed Development by rcent

Custo r: Financial services firm
Co try or Region: Europe
ndustry: Financial services—Banking

Customer Profile
A large European financial services firm offers banking and asset-management services to clients in 50 countries. In 2009, the bank earned more than U.S.$6 billion in income.

Software and Services
- Microsoft Visual Studio
  - Microsoft Visual F#
  - Microsoft Visual Studio 2010
- Technologies
  - Microsoft .NET Framework
  - Windows Presentation Foundation

"We could not have deve ped 200 models in two years without F# and Visual Studio. It would have taken us at least twice as long with our pr

Director at a large Europ

A large financial services firm in Europe sought new development tools that could cut costs, boost p ivity, and improve the quality of its mathematical mo To address its needs, the bank deployed Microsoft F# e Microsoft .NET Framework, and Microsoft Visual S dio. It will soon upgrade to Visual Studio 2010 and the integrated Microsoft Visual F#. With its new tools, the bank can speed development by 50 percent or more, improve quality, and reduce costs.

Business Needs
A large European financial services

desktop and on a remote cluster of servers that includes hundreds of systems

http://fsharp.net

Example #3: F# in Insurance

Time to Market

Complexity

I work for a large ac al company... ...Despite adopting Agile/Scrum ...the usual delays, comp ations and sometimes ...failures.

Efficiency

We used F#, and quickly created a system which would perform the necessary calculations highly efficiently, in parallel, and with a perfect match to the spreadsheet results.

All of the advantages which are commonly touted for F# do play ut in practice. *Immutability, Easy Parallelisation, Expressiveness, Test ility, Conciseness, Flexibility, Productivity*

[Company name omitted]

Correctness

fsharp.org/testimonials

Example #4: Finance trading platform

F# + C# for Trading Front End

Leverage F#'s features:

- extensive type system
- asynchronous workflows, agents and immutable types
- rich pattern matching and parser support

TRADING AT YOUR FINGERTIPS
Powering over 13,000 screens worldwide

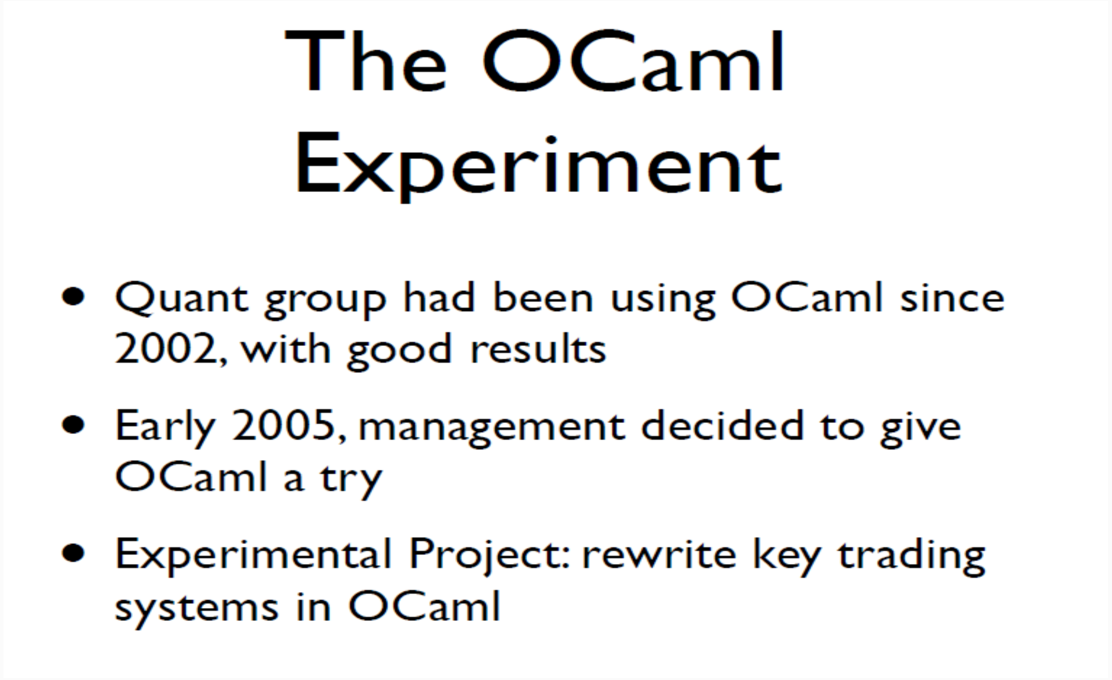
Time to Market

"Experienced F# developers regularly solve problems in days that would take weeks using more traditional languages…solving complex problems in an elegant highly maintainable manner"

Complexity

Phil Trelford, Trading Platform Company

Example #5: OCaml @ Jane St



The OCaml Experiment

- Quant group had been using OCaml since 2002, with good results
- Early 2005, management decided to give OCaml a try
- Experimental Project: rewrite key trading systems in OCaml

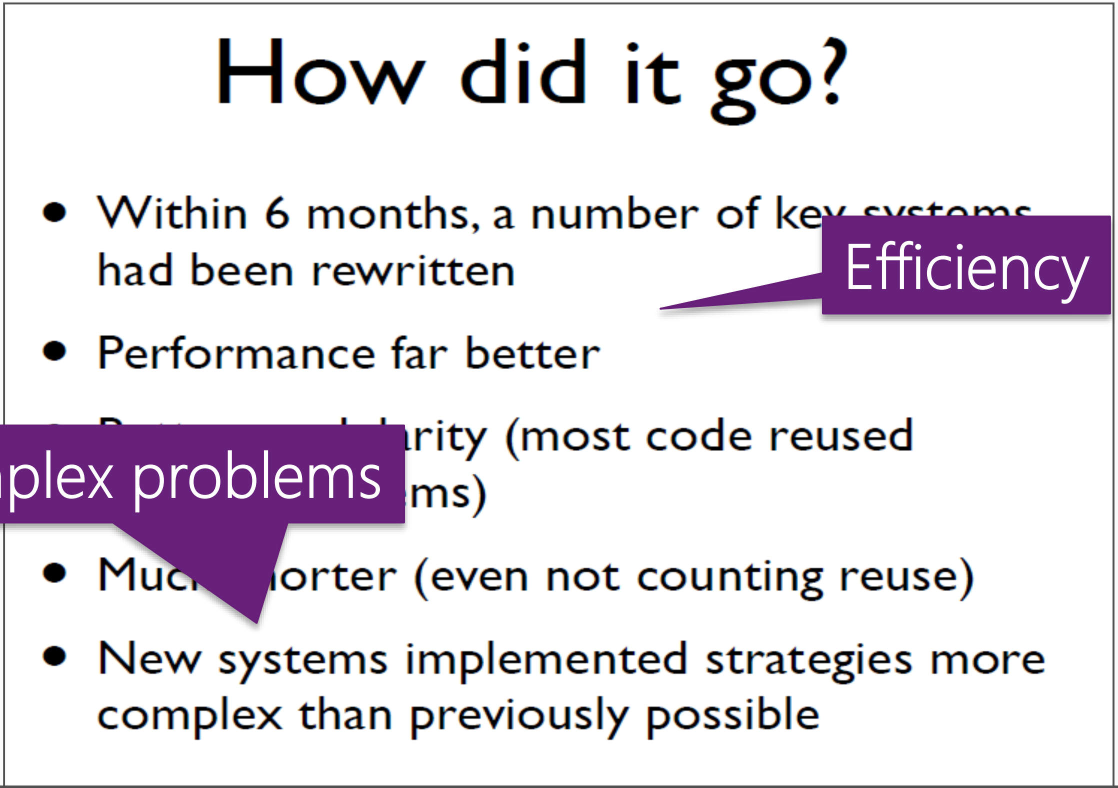
Robustness

Performance

Readability

source: http://www.janestcapital.com/yaron_minsky-cufp_2006.pdf

Example #5: OCaml @ Jane St



How did it go?

- Within 6 months, a number of key systems had been rewritten
- Performance far better
- Better modularity (most code reused across systems)
- Much shorter (even not counting reuse)
- New systems implemented strategies more complex than previously possible

Efficiency

Solve complex problems

source: http://www.janestcapital.com/yaron_minsky-cufp_2006.pdf

Example #6: F# in Biotech

Efficiency

…F# rocks - building algorithms for DNA processing and it's like a drug. 12-15 at Amyris use F#… A complete genome resequencing pipeline with interface, algs, reporting in ~5K lines and it has been incredibly reliable, fast and easy to maintain.. A suffix tree in 150 lines that can index 200,000 bases a second

Correctness

F# v. Python: F# has been phenomenally useful. I would be writing a lot of this in Python otherwise and F# is more robust, 20x - 100x faster to run and faster to develop.

Time to Market

Darren Platt, Amyris BioTechnologies

Example #7: F# in Advertisement Ranking & Rating @ Microsoft

Time to Market

Around 95% of the code in these rojects has been developed in F#.

F# allowed for rapid development of prototypes, and thus also rapid

Taming Complexity

on of the underlying mathematical models.

Complex algorithms, for example to compute Nash equilibria in game theory, can be expressed succinctly.

Correctness

Units of measure reduced the chance of errors dramatically: Prices, probabilities, derivatives, etc. can already be kept apart at compile time.

Example #8: F# at Kaggle

Taming Complexity

At Kaggle we initially chose F data analysis algorithms because of its expressiveness.

Correctness

We've found ourselves moving more and more of application ...into F#. The F# code is shorter, easier to read, easier to refactor, and, because of the strong typing, contains far fewer bugs.

Time to Market

As our data analysis tools have developed, we've seen domain-specific constructs emerge very naturally. As our codebase gets larger, we become more productive.

fsharp.org/testimonials

Example #9: F# for Machine Learning at Microsoft

I wrote the first prototype of the click prediction system deployed in Microsoft AdCenter in F# in a few days.

Time to Insight

For a machine learning scientist, speed of experimentation is the critical factor to optimize.

Unlike C# and C++, F# was designed for this mode of interaction. Switching to F# was liberating and exhilarating.

Correctness

The world is moving toward functional programming with good justifications: the code is cleaner and easier to debug in a distributed environment.

Dr. Patrice Simard, Microsoft Distinguished Engineer, fsharp.org/testimonials

Example #10: F# for Consulting

Our bids for tendered contracts in quan … are re … the price of competitors because of the increa … d productivity we … #.

Correctness

Efficiency

We are regularly able to deliver correct, robust, performant solutions on-time, which is what our customers value most.

Time to Market

Daniel Egloff, QuantAlea Consulting, Zurich

http://fsharp.org/testimonials

Example #11: F# for Social Gaming

F# is becoming an increasingly important part of our server cture that supports our mobile and web-based social games with ctive users. F# first came to prominence in our technology stack in the implementation of the rules engine for our social slots games which by now serve over **700,000 unique players** and **150,000,000 requests per day** at peaks of **several thousand requests per second**.

Efficiency

The F# solution offers us an order of magnitude increase in productivity and allows one developer to perform the work that are performe dedicated developers on an existing Java-based solution, and supporting our agile approach and bi-weekly release cycles.

Time to Market

Yan Cui, Lead Server Engineer
http://fsharp.org/testimonials

Example #12: F# for Insurance

One of the world's larges **Efficiency** npanies have F# code in production, are starting several more projects in F#

They migrated some of their number crunching and business logic to F# and are so happy with the results (10x faster and 10x less code vs their Visual C++ 6) that they are proposing to migrate 1,600,000 lines of code to F#. In particula **Time to Market** d F# easy to learn and use.

... my predecessor developed an entire pension quote calculator (typically scheduled to take 300-400 man days) entirely in F# in under 100 days with no prior F# experience at all. Performance is 10× better than the C++ that it replaces because the new code avoids unnecessary copying and exploits multicore parallelism.

Aviva

http://fsharp.org/testimonials

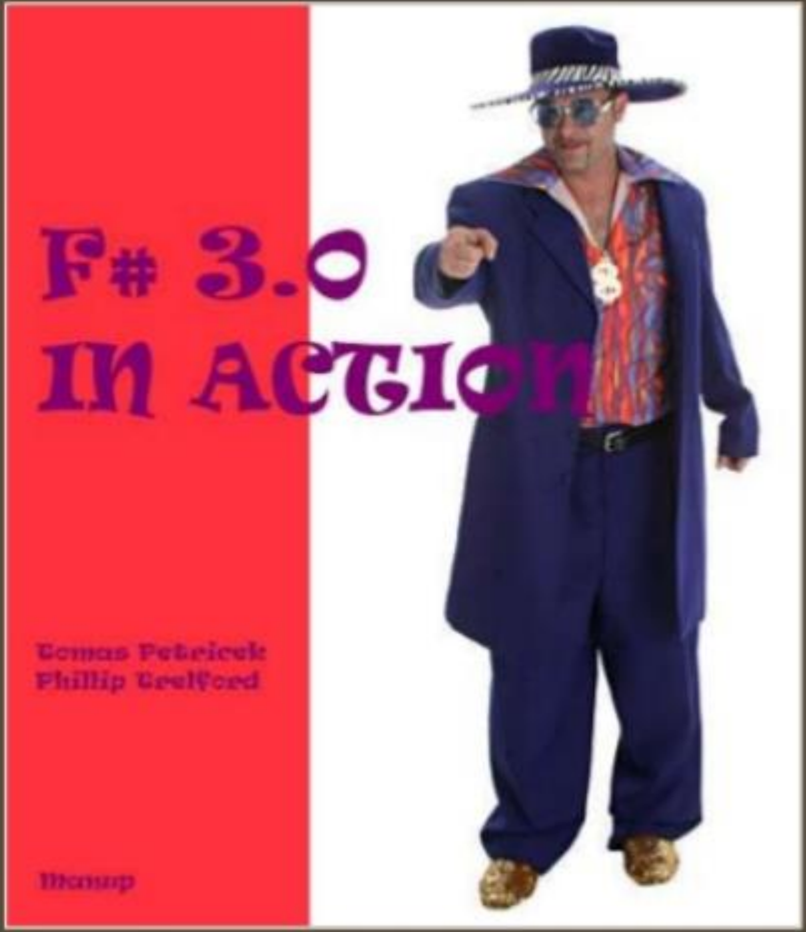
F# FOR PROFIT

Time to Market

Efficiency

Correctness

Complexity



Summary – The Data Agrees

- Simple, correct, robust code
- Interoperability improves time-to-market
- Strong-typing gives efficiency
- Analytical developers empowered to solve complex problems

Part 3 – Topics on F# in Practice

Part 3 – Topics on F# in Practice

Topic - Data

You can easily find out more about…

F# Basics

F# for Data Science

F# for GPUs

F# for Cloud Data

F# for Pricing

F# for DSLs

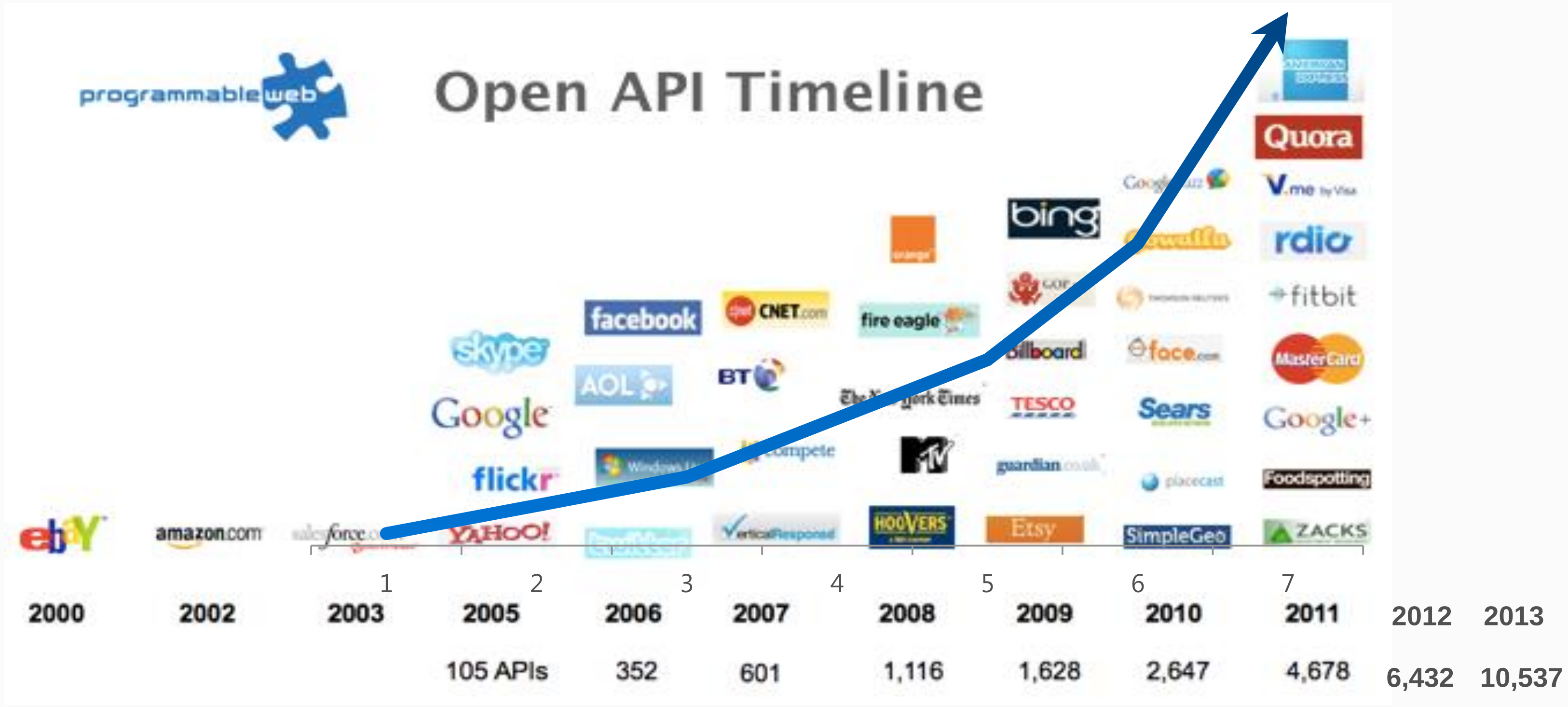
F# + R

F# + Excel

F# Deep Data Integration

Proposition 1
The world is information-rich

The Information Revolution



Data is like water…

Data is like water…

- Everyone needs it. Everyone knows where to get it.
- Nobody is sure where it really came from, or goes to.
- …really knows its true cost, or true value.
- …likes to pay for it, or to share it.
- …knows how much is wasted.
- You might get washed away by it.
- You only find out it was bad after you have drunk it.

Actually these days it’s more like a flood…

The Problem

Our programming tools are data-sparse

getting data **into** a programming language is tiresome, error prone and boring

We need to bring data **into** the language...

At internet scale, strongly tooled, strongly typed

Problem: Integrate all of freebase.com

“as if it were a library”

>40M entities, >1Billion facts, >24,000 types, >65,000 properties

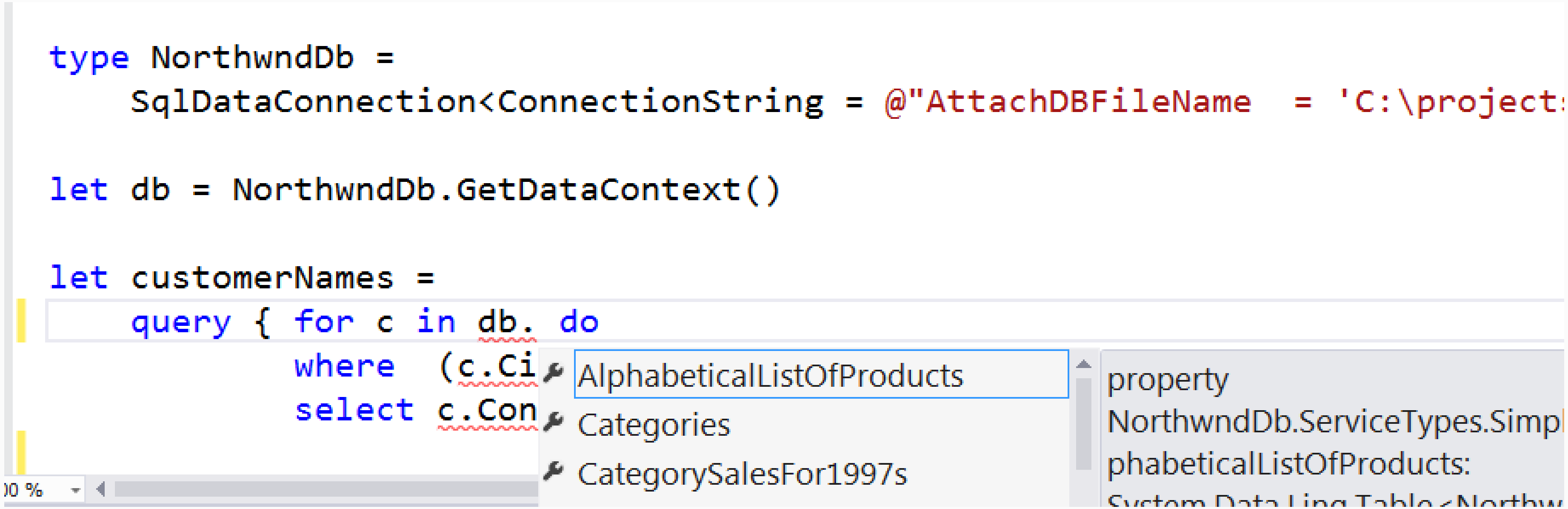
Demo

F# + Freebase

An F# type provider for deep, robust integration of web data

All your types are belong to us….

SQL #1

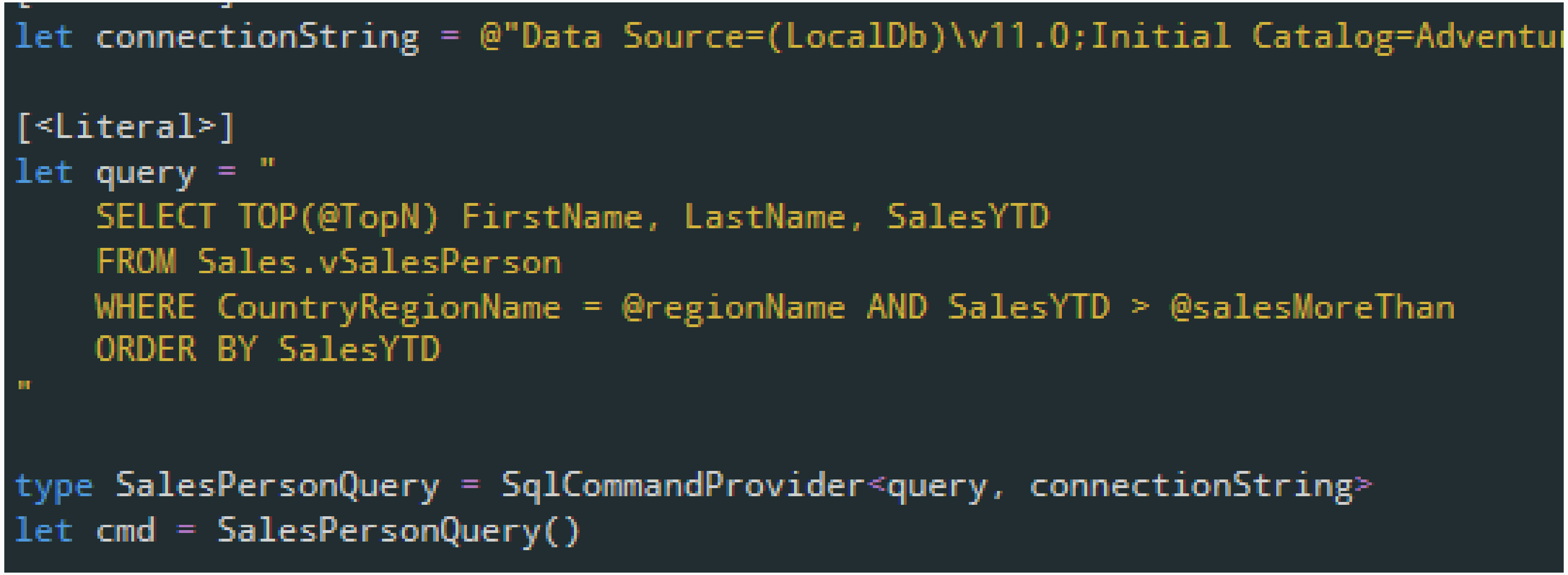


```
type NorthwndDb =
    SqlDataConnection<ConnectionString = @"AttachDBFileName  = 'C:\project

let db = NorthwndDb.GetDataContext()

let customerNames =
    query { for c in db. do
                where  (c.Ci
                select c.Con
```

```
AlphabeticalListOfProducts
Categories
CategorySalesFor1997s
```

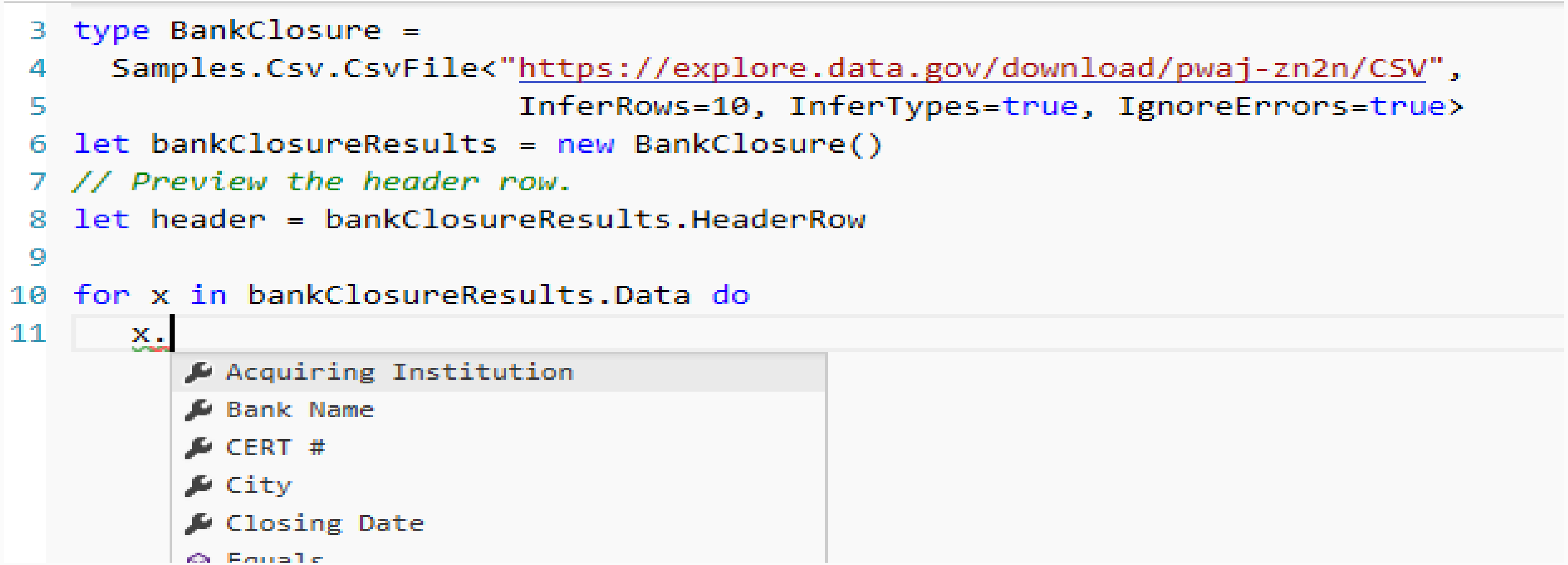
```
property
NorthwndDb.ServiceTypes.Simpl
phabeticalListOfProducts:
System.Data.Linq.Table<Northw
```

SQL #2



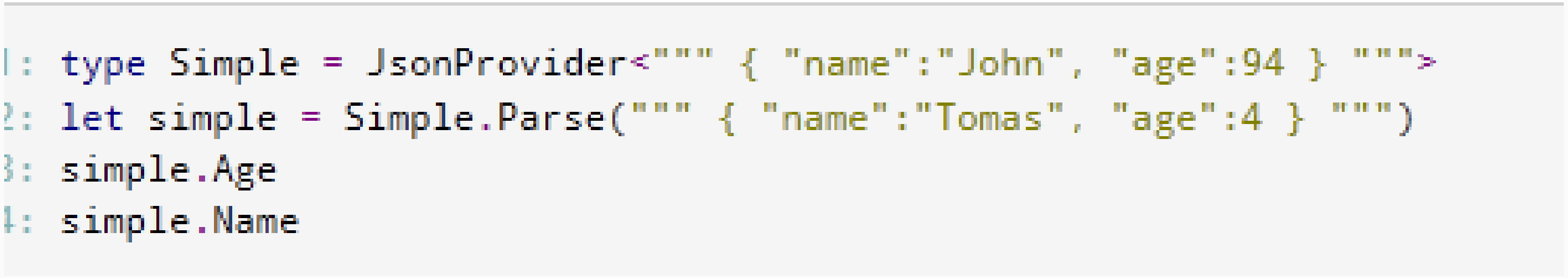
```
let connectionString = @"Data Source=(LocalDb)\v11.0;Initial Catalog=Adventu

[<Literal>]
let query = "
    SELECT TOP(@TopN) FirstName, LastName, SalesYTD
    FROM Sales.vSalesPerson
    WHERE CountryRegionName = @regionName AND SalesYTD > @salesMoreThan
    ORDER BY SalesYTD
"

type SalesPersonQuery = SqlCommandProvider<query, connectionString>
let cmd = SalesPersonQuery()
```

CSV



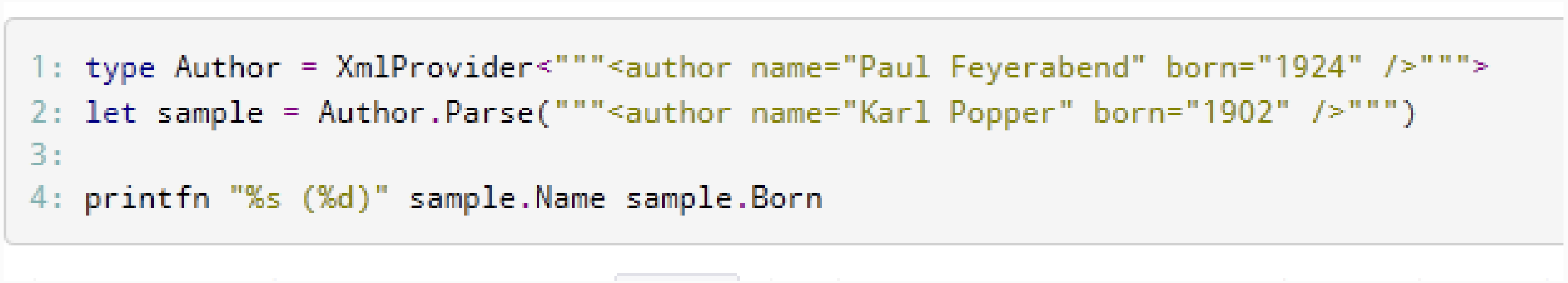
```
type BankClosure =
  Samples.Csv.CsvFile<"https://explore.data.gov/download/pwaj-zn2n/CSV",
                      InferRows=10, InferTypes=true, IgnoreErrors=true>
let bankClosureResults = new BankClosure()
// Preview the header row.
let header = bankClosureResults.HeaderRow

for x in bankClosureResults.Data do
    x.
```

JSON



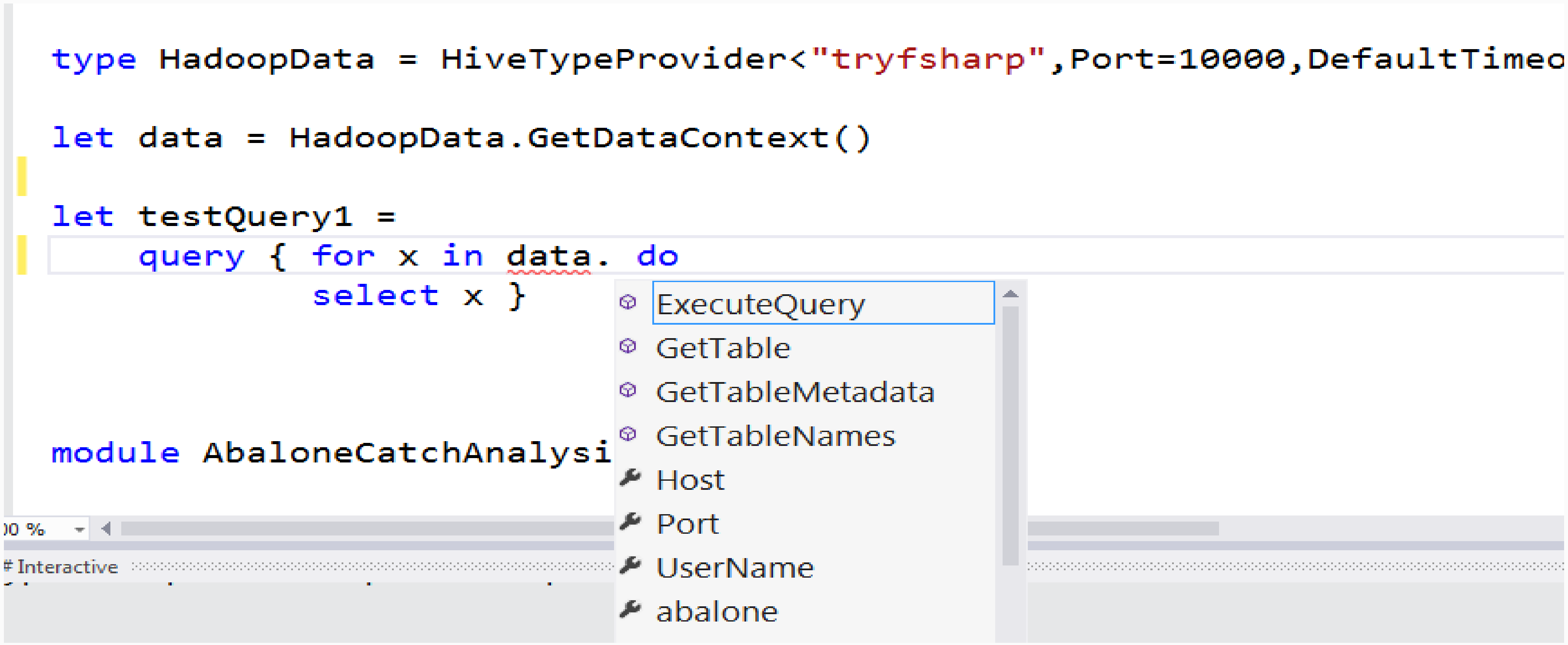
```
type Simple = JsonProvider<""" { "name":"John", "age":94 } """>
let simple = Simple.Parse(""" { "name":"Tomas", "age":4 } """)
simple.Age
simple.Name
```

XML



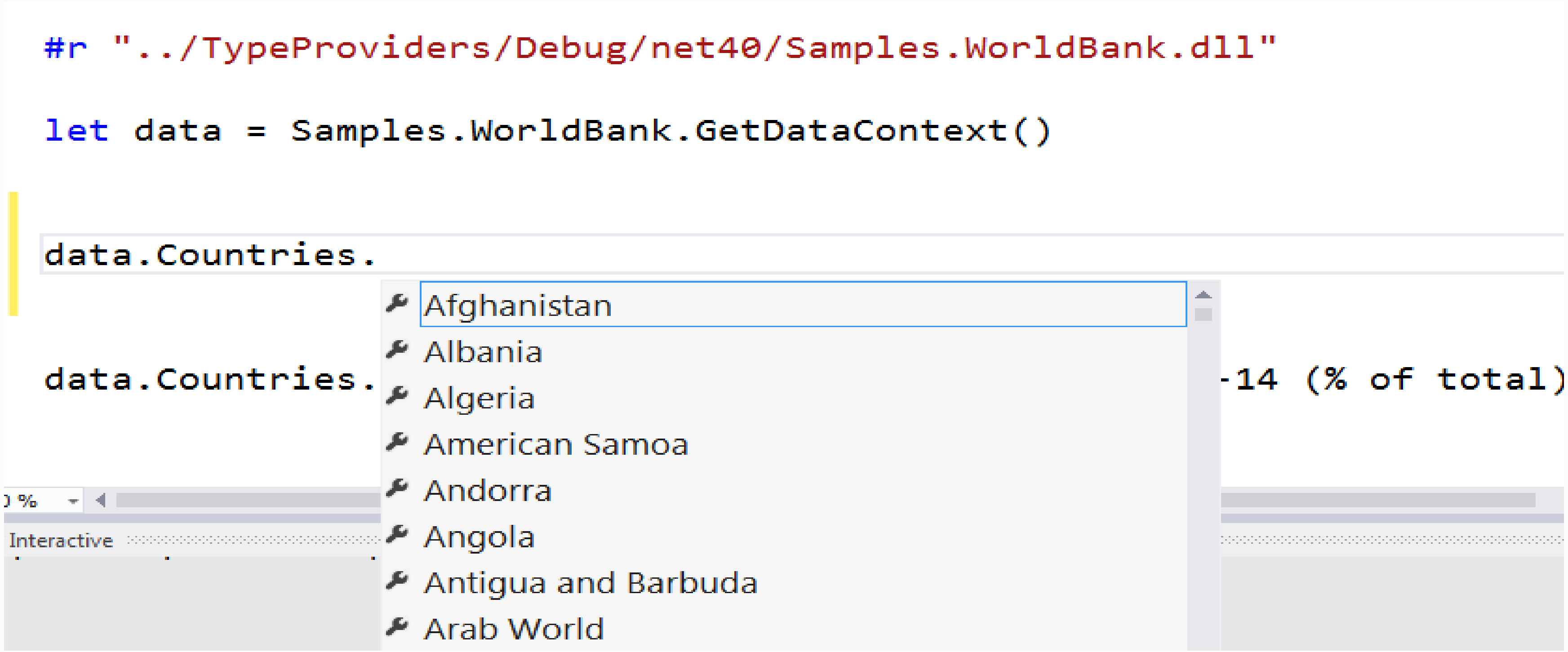
```
type Author = XmlProvider<"""<author name="Paul Feyerabend" born="1924" />""">
let sample = Author.Parse("""<author name="Karl Popper" born="1902" />""")

printfn "%s (%d)" sample.Name sample.Born
```

Hadoop/Hive



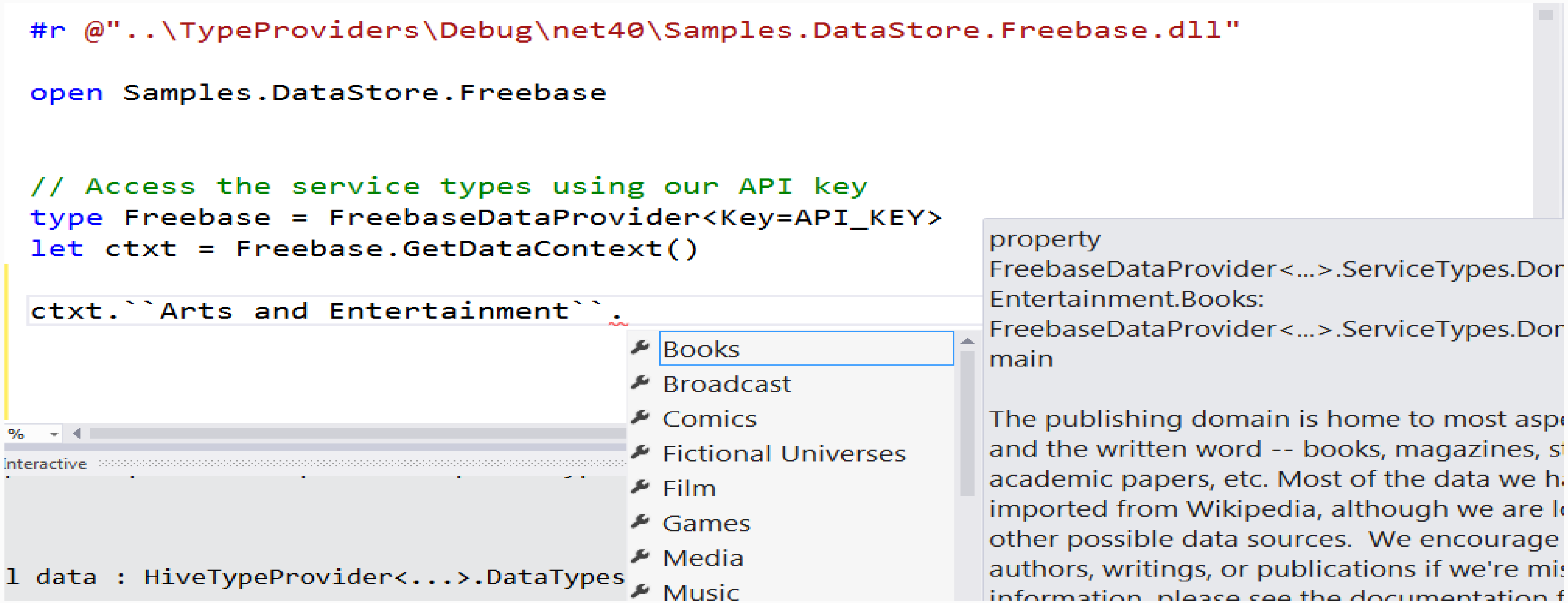
```
type HadoopData = HiveTypeProvider<"tryfsharp",Port=10000,DefaultTimeo

let data = HadoopData.GetDataContext()

let testQuery1 =
    query { for x in data. do
                select x }

module AbaloneCatchAnalysi
```

World Bank



```
#r "../TypeProviders/Debug/net40/Samples.WorldBank.dll"

let data = Samples.WorldBank.GetDataContext()


data.Countries.


data.Countries.                                          -14 (% of total)
```

Freebase



```
#r @"..\TypeProviders\Debug\net40\Samples.DataStore.Freebase.dll"

open Samples.DataStore.Freebase


// Access the service types using our API key
type Freebase = FreebaseDataProvider<Key=API_KEY>
let ctxt = Freebase.GetDataContext()

ctxt.``Arts and Entertainment``.
```

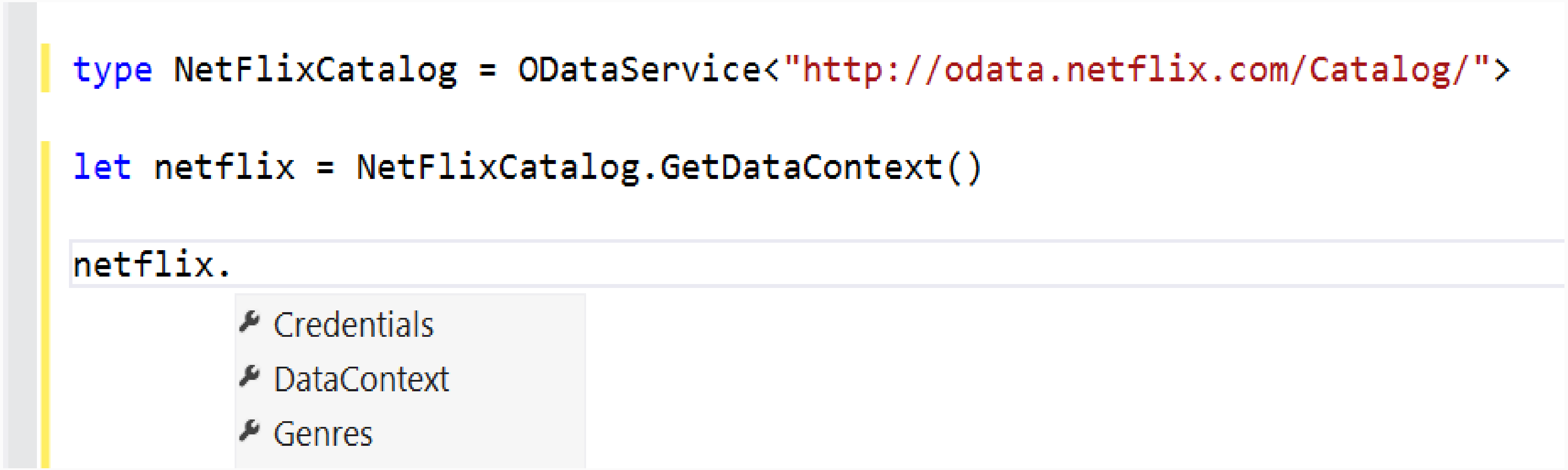
```
Books
Broadcast
Comics
Fictional Universes
Film
Games
Media
Music
```

property
FreebaseDataProvider<...>.ServiceTypes.Dor
Entertainment.Books:
FreebaseDataProvider<...>.ServiceTypes.Dor
main

The publishing domain is home to most aspe
and the written word -- books, magazines, s
academic papers, etc. Most of the data we h
imported from Wikipedia, although we are l
other possible data sources. We encourage
authors, writings, or publications if we're mi
information, please see the documentation f

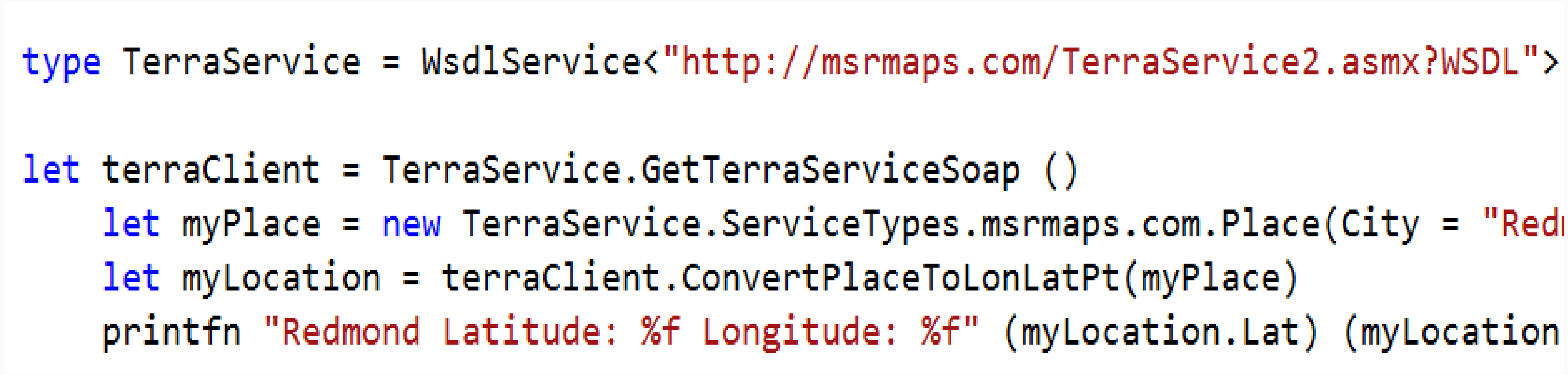
```
l data : HiveTypeProvider<...>.DataTypes
```

OData



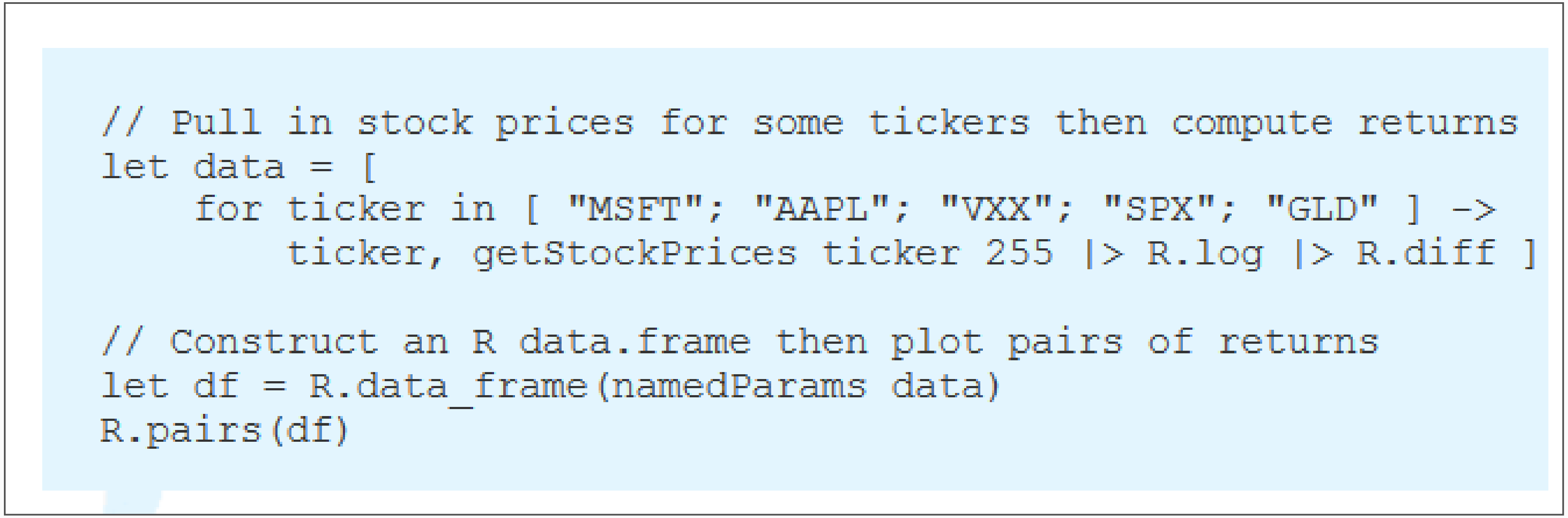
```
type NetFlixCatalog = ODataService<"http://odata.netflix.com/Catalog/">

let netflix = NetFlixCatalog.GetDataContext()

netflix.
```

WSDL



```
type TerraService = WsdlService<"http://msrmaps.com/TerraService2.asmx?WSDL">

let terraClient = TerraService.GetTerraServiceSoap ()
    let myPlace = new TerraService.ServiceTypes.msrmaps.com.Place(City = "Red
    let myLocation = terraClient.ConvertPlaceToLonLatPt(myPlace)
    printfn "Redmond Latitude: %f Longitude: %f" (myLocation.Lat) (myLocation
```

R



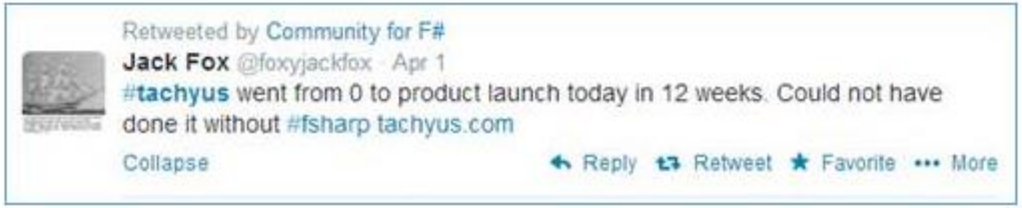
```
// Pull in stock prices for some tickers then compute returns
let data = [
    for ticker in [ "MSFT"; "AAPL"; "VXX"; "SPX"; "GLD" ] ->
        ticker, getStockPrices ticker 255 |> R.log |> R.diff ]

// Construct an R data.frame then plot pairs of returns
let df = R.data_frame(namedParams data)
R.pairs(df)
```

SQL #2 - Application

Tachyus is a Silicon Valley startup that aims to be "***a Data Start-Up for the Oil Industry***". They aim to create an array of sensors and mobile applications to help oil and gas producers better record and analyze their wells. According to the New York Times coverage:

> *The start-up represents an anomaly of sorts in Silicon Valley. Many new businesses focus on high-technology products for the Internet or green technology, but Mr. Sloss and his co-founders, Paul Orland and Francisco LePort, have instead homed in on the decidedly older and dirtier business of drilling for hydrocarbons.*

Last week Tachyus announced that it has raised $6M in funding from a group led by Founders Fund. At the time of the announcements, one of the Tachyus engineers announced that they went from "from zero to product launch in 12 weeks" and "we couldn't have done it without F#". Founnder Paul Orland commented "we are using 100% F#"



Retweeted by Community for F#
Jack Fox @foxyjackfox · Apr 1
#**tachyus** went from 0 to product launch today in 12 weeks. Could not have done it without #fsharp tachyus.com
Collapse ↩ Reply ⟲ Retweet ★ Favorite ••• More

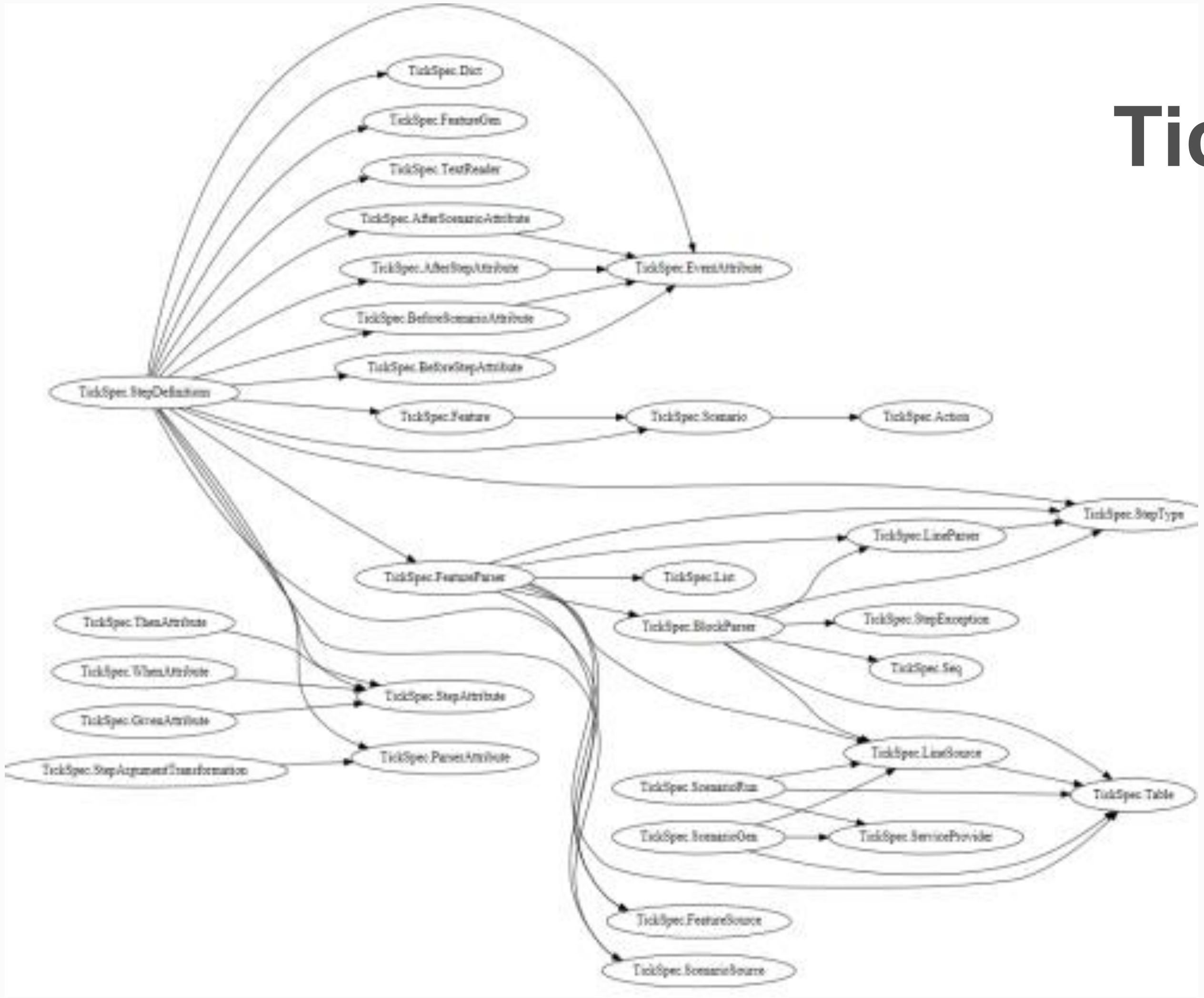
Part 3 – Topics on F# in Practice

Topic – Managing Complexity in the Large

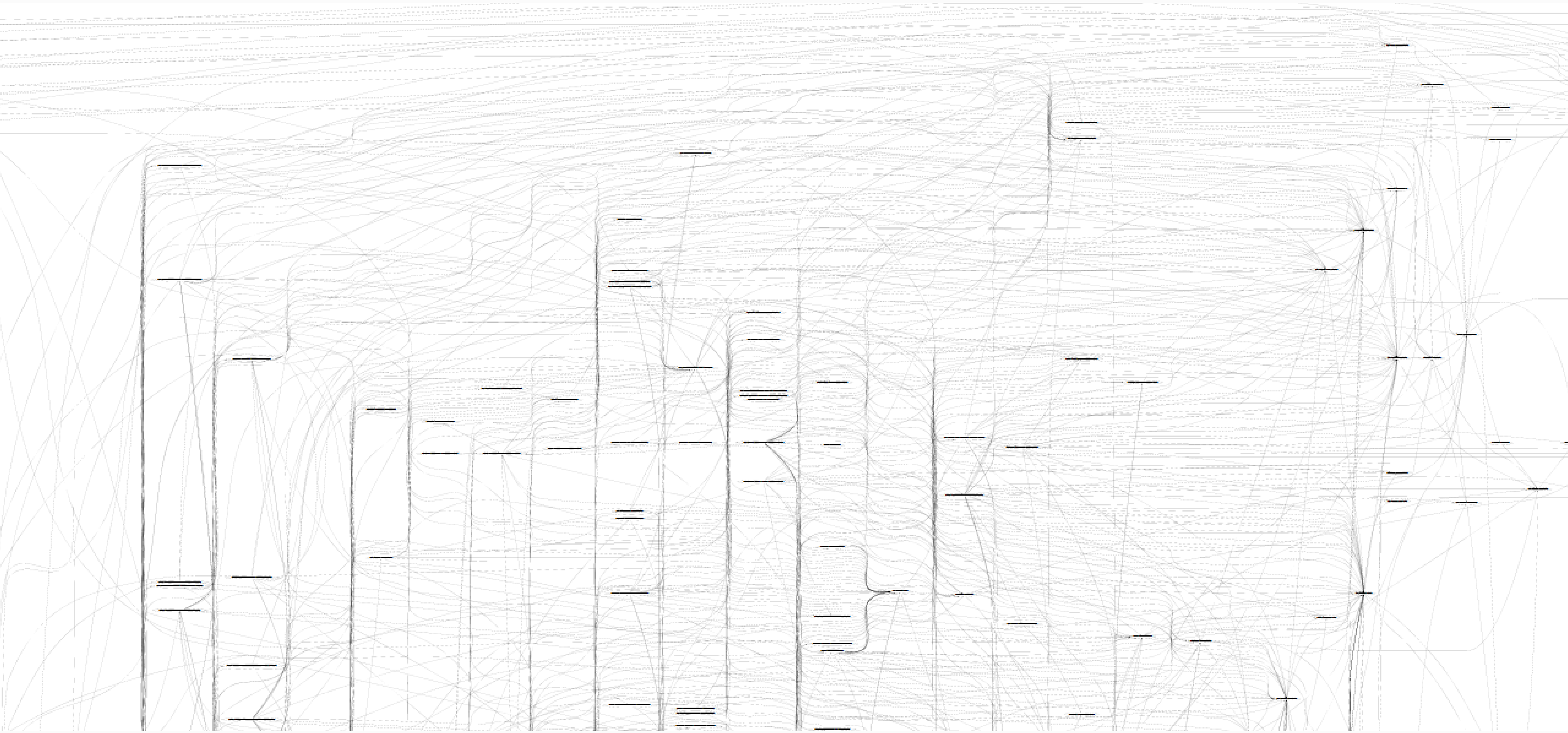
The dependency structure of some real-world OO and functional-first projects

SpecFlow (

TickFlow (F


TickSpec.Dict
TickSpec.FeatureGen
TickSpec.TextReader
TickSpec.AfterScenarioAttribute
TickSpec.AfterStepAttribute
TickSpec.EventAttribute
TickSpec.BeforeScenarioAttribute
TickSpec.BeforeStepAttribute
TickSpec.StepDefinitions
TickSpec.Feature
TickSpec.Scenario
TickSpec.Action
TickSpec.StepType
TickSpec.LineParser
TickSpec.FeatureParser
TickSpec.List
TickSpec.ThenAttribute
TickSpec.BlockParser
TickSpec.StepException
TickSpec.Seq
TickSpec.WhenAttribute
TickSpec.StepAttribute
TickSpec.GivenAttribute
TickSpec.LineSource
TickSpec.ParserAttribute
TickSpec.StepArgumentTransformation
TickSpec.ScenarioRun
TickSpec.Table
TickSpec.ScenarioGen
TickSpec.ServiceProvider
TickSpec.FeatureSource
TickSpec.ScenarioSource

“Entity Framework” (OO)



(and that’s just 1/4 of the graph…)

Part 3 – Topics on F# in Practice

Topic – Integration

Typical F# Topics

F# Basics

F# for Data Science

F# for GPUs

F# + Excel

F# for Pricing

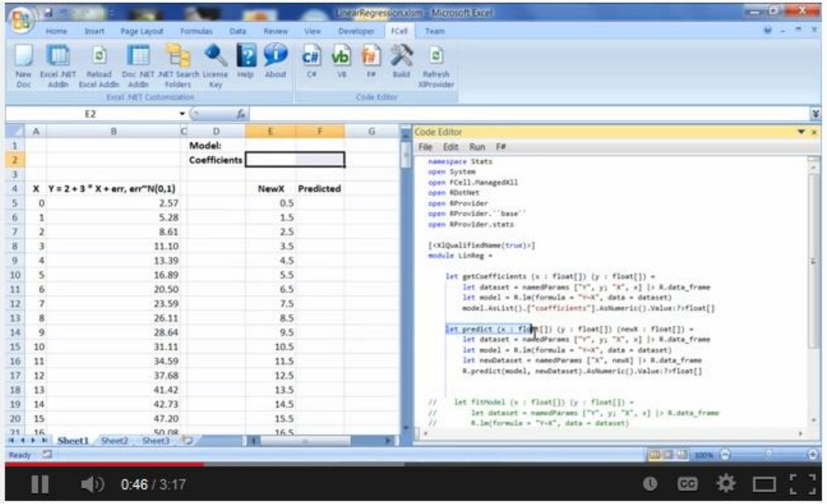
F# for DSLs

F# + R

F# Deep Data Integration

Functional + R + Excel Integration

via fcell.io



	A	B	C	D	E	F	G
1				Model:			
2				Coefficients			
3							
4	X	Y = 2 + 3 * X + err, err~N(0,1)			NewX	Predicted	
5	0	2.57			0.5		
6	1	5.28			1.5		
7	2	8.61			2.5		
8	3	11.10			3.5		
9	4	13.39			4.5		
10	5	16.89			5.5		
11	6	20.50			6.5		
12	7	23.59			7.5		
13	8	26.11			8.5		
14	9	28.64			9.5		
15	10	31.11			10.5		
16	11	34.59			11.5		
17	12	37.68			12.5		
18	13	41.42			13.5		
19	14	42.73			14.5		
20	15	47.20			15.5		
21	16	50.08			16.5		

```fsharp
namespace Stats
open System
open FCell.ManagedXll
open RDotNet
open RProvider
open RProvider.``base``
open RProvider.stats

[<XlQualifiedName(true)>]
module LinReg =

    let getCoefficients (x : float[]) (y : float[]) =
        let dataset = namedParams ["Y", y; "X", x] |> R.data_frame
        let model = R.lm(formula = "Y~X", data = dataset)
        model.AsList().["coefficients"].AsNumeric().Value:?>float[]

    let predict (x : float[]) (y : float[]) (newX : float[]) =
        let dataset = namedParams ["Y", y; "X", x] |> R.data_frame
        let model = R.lm(formula = "Y~X", data = dataset)
        let newDataset = namedParams ["X", newX] |> R.data_frame
        R.predict(model, newDataset).AsNumeric().Value:?>float[]


//    let fitModel (x : float[]) (y : float[]) =
//        let dataset = namedParams ["Y", y; "X", x] |> R.data_frame
//        R.lm(formula = "Y~X", data = dataset)
```

Part 3 – Topics on F# in Practice

Topic – GPU Execution

Typical F# Topics

F# Basics

F# for Data Science

F# for GPUs

F# + Excel

F# for Pricing

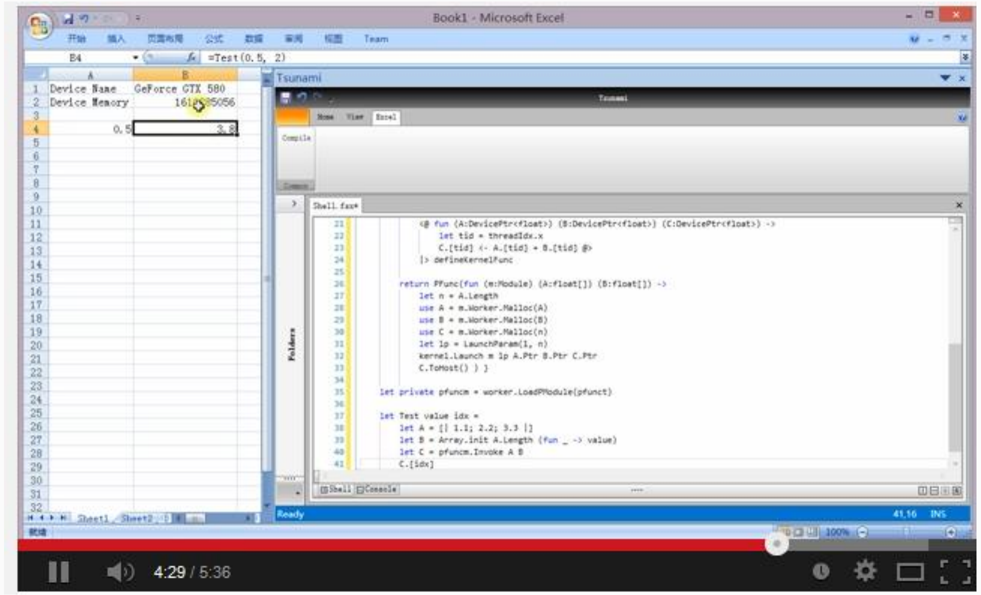
F# for DSLs

F# + R

F# Deep Data Integration

Functional + GPGPU

F# + FCell + QuantAlea



```
        <@ fun (A:DevicePtr<float>) (B:DevicePtr<float>) (C:DevicePtr<float>) ->
            let tid = threadIdx.x
            C.[tid] <- A.[tid] + B.[tid] @>
        |> defineKernelFunc

    return PFunc(fun (m:Module) (A:float[]) (B:float[]) ->
        let n = A.Length
        use A = m.Worker.Malloc(A)
        use B = m.Worker.Malloc(B)
        use C = m.Worker.Malloc(n)
        let lp = LaunchParam(1, n)
        kernel.Launch m lp A.Ptr B.Ptr C.Ptr
        C.ToHost() ) }

let private pfuncm = worker.LoadPModule(pfunct)

let Test value idx =
    let A = [| 1.1; 2.2; 3.3 |]
    let B = Array.init A.Length (fun _ -> value)
    let C = pfuncm.Invoke A B
    C.[idx]
```

	A	B
1	Device Name	GeForce GTX 580
2	Device Memory	161[illegible]5056
3		
4	0.5	3.8

Summary

Functional-first programming is the

safe choice

for many programming tasks in industry

Training, learning and community are key!

Summary – F#

Open, cross-platform, strongly typed, efficient, rock-solid stable

The safe choice for functional-first

F#

Unbeatable, practical, scalable data integration

Tooling for Windows, Linux, OSX, Android, iOS and more

To find out more…

Learn F# at tryfsharp.org (including financial)

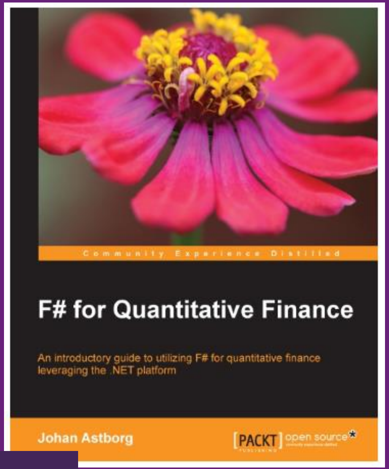
Lots of resources at fsharp.org

Join the Copenhagen Functional Meetup Group!

Testimonials at fsharp.org/testimonials

Over 100 videos at fsharp.org/videos

Questions?

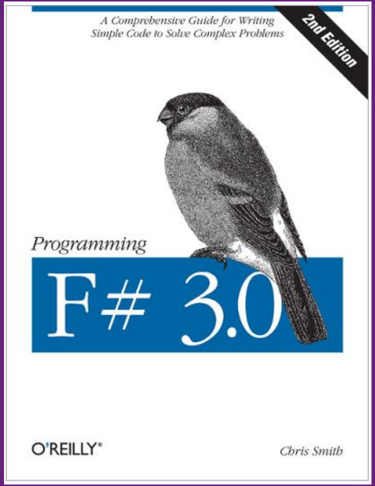

Community Experience Distilled

F# for Quantitative Finance

An introductory guide to utilizing F# for quantitative finance leveraging the .NET platform

Johan Astborg

[PACKT] open source


A Comprehensive Guide for Writing Simple Code to Solve Complex Problems

2nd Edition

Programming

F# 3.0

O'REILLY

Chris Smith

tryfsharp.org