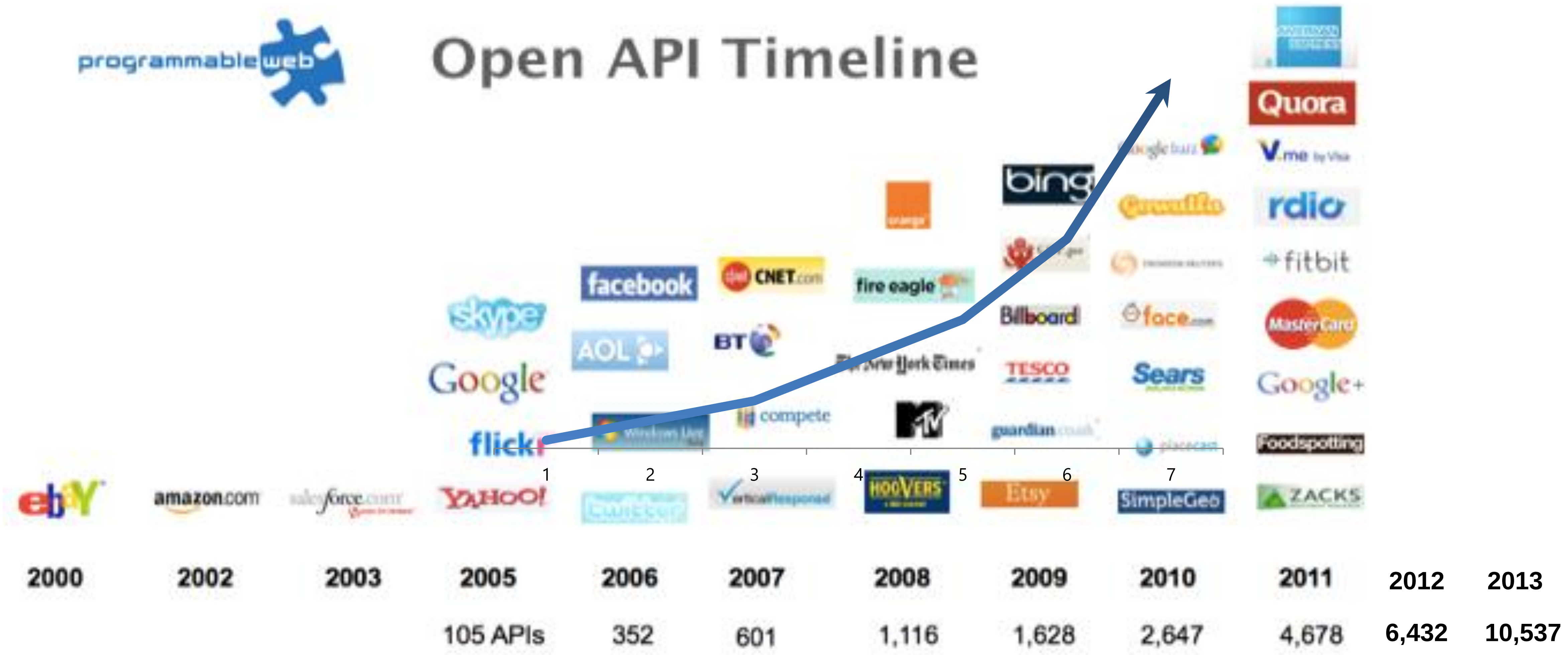
Making Magic with F# Type Providers

Dr Kenji Takeda (@ktakeda1)
Microsoft Research

#fsharp

Proposition 1
We are living through an information revolution

The Information Revolution



Proposition 2

Our programming languages are **information-sparse**

Proposition 3
This is a big problem

especially for statically typed languages
(Java, C#, F#, VB, …)

We need to bring information **into** the language…

At **internet-scale, strongly tooled, strongly typed**

But before we get into that…

Part 1

Functional-first Programming and F#

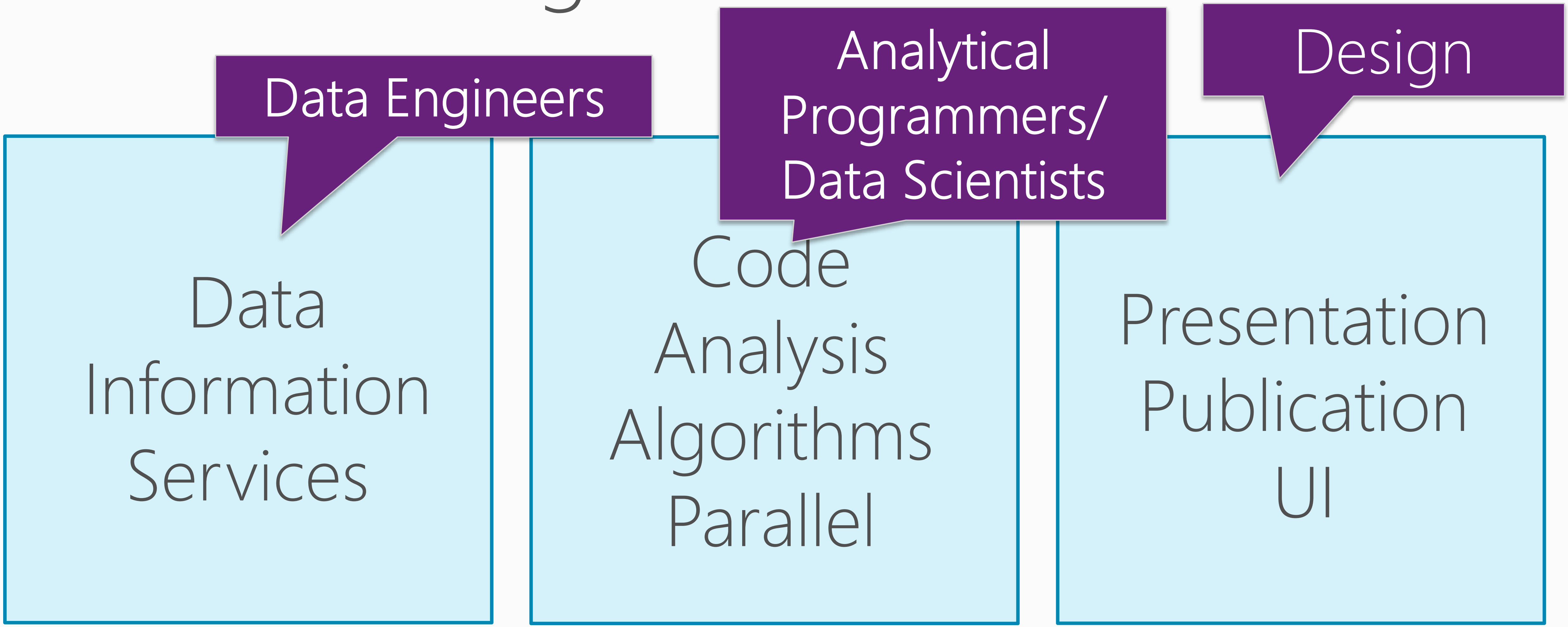
F# is free, open source, cross platform, independent

fsharp.org

Microsoft contribute to F#, and so do many others

The Visual F# Tools from Microsoft are supported, enterprise-ready and come with Visual Studio

Understanding the Situation



Data Engineers

Data
Information
Services

Analytical Programmers/ Data Scientists

Code
Analysis
Algorithms
Parallel

Design

Presentation
Publication
UI

The Recurring Business Situation

"I lead a team developing…"

- Analytical Components
- Data-rich Services
- Analytical Components
- Data-rich Services
- Analytical Components
- …

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 Complexity a Problem?

Intractable Models → Can’t enter markets

Intractable Services → Can’t deliver services

Intractable Analytical Components → Can’t ship

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

Why?

Observation #1

At the core of every functional-first language is:

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)

We 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

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

Correctness

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

Time to Market

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

Correctness

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

Efficiency

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

Time to Market

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

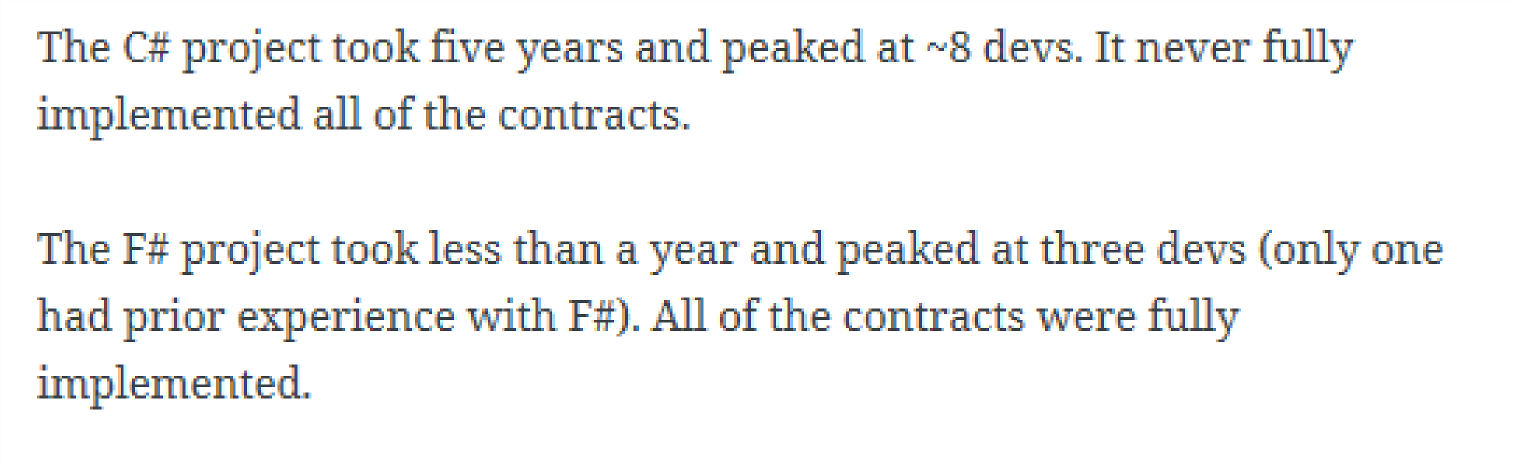
Correctness

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

Example #1 (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/

Example #1 (Simon Cousins, Energy Sector)

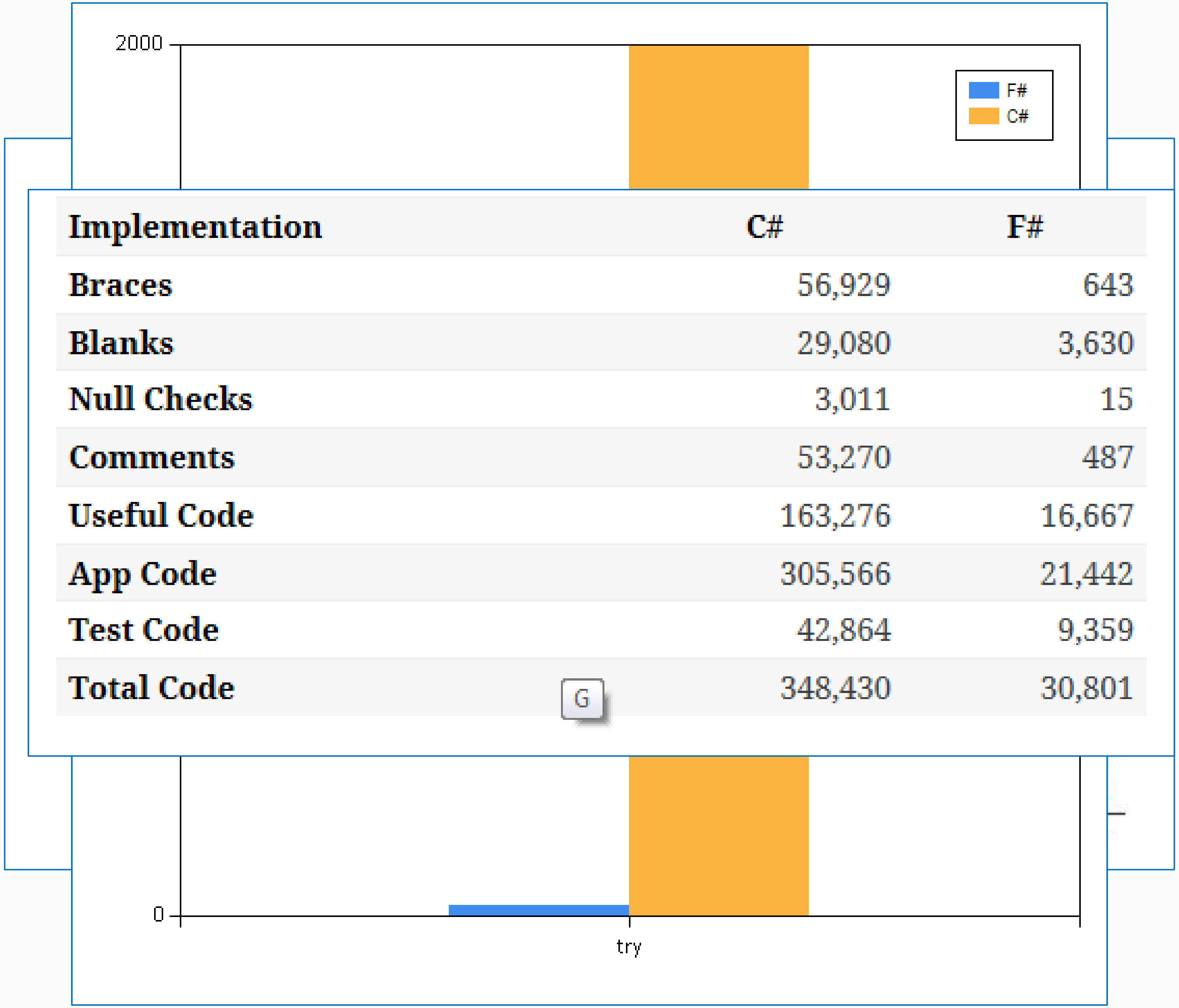
Zero

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

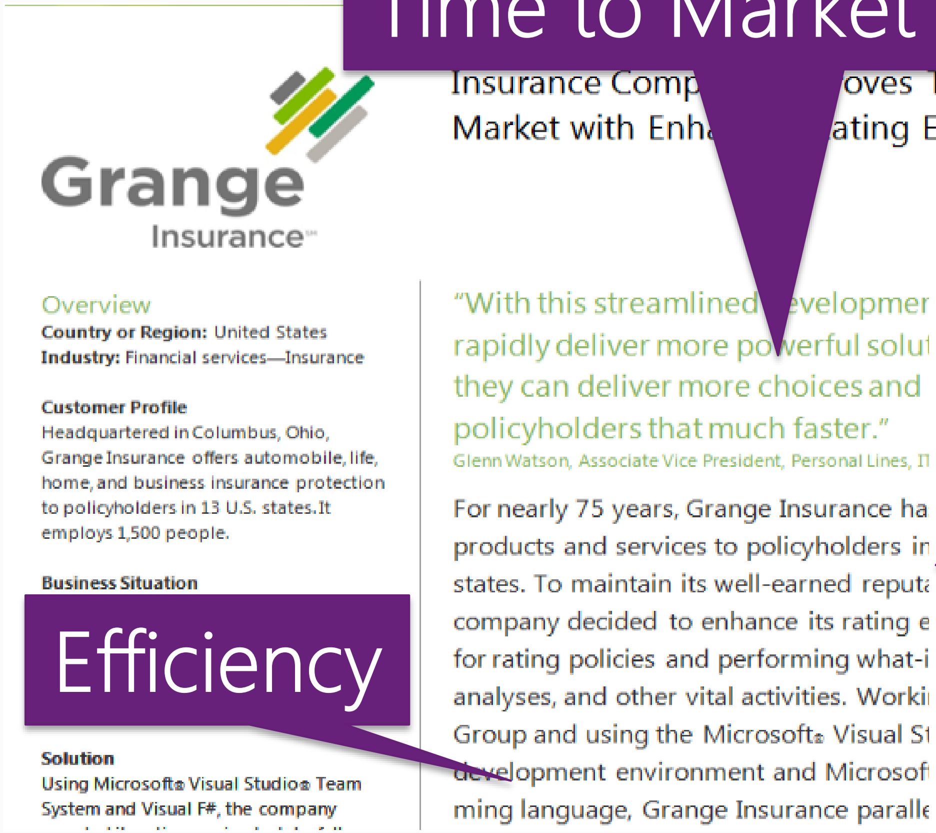
http://simontcousins.azurewebsites.net/does-the-language-you-use-make-a-difference-revisited/



2000
0
F#
C#
try

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

Example #2: F# in Finance



Time to Market

Efficiency

Grange Insurance

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

Overview

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

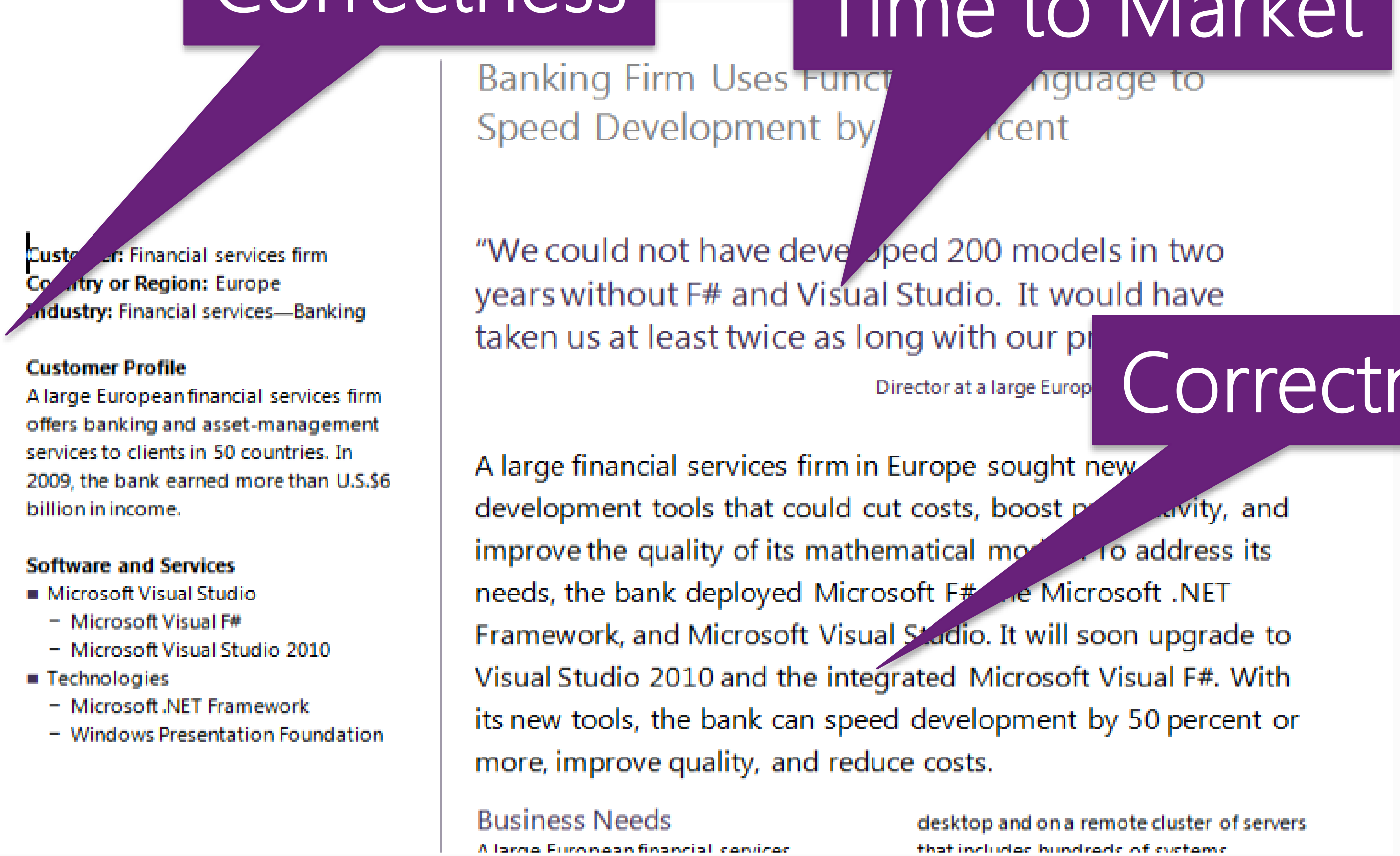
Customer Profile
Headquartered in Columbus, Ohio, Grange Insurance offers automobile, life, home, and business insurance protection to policyholders in 13 U.S. states. It employs 1,500 people.

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, IT

For nearly 75 years, Grange Insurance ha products and services to policyholders in 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



Correctness

Time to Market

Correctness

Customer: Financial services firm
Country or Region: Europe
Industry: 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

Banking Firm Uses Funct nguage to Speed Development by rcent

"We could not have developed 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 Studio. 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: F# in Biotech

…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

Efficiency

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 #5: F# at Kaggle

Taming Complexity

At Kaggle we initially chose F ata 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 #6: F# in Advertisement Ranking & Rating @ Microsoft

Time to Market

Around 95% of the code in these projects 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 #7: 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 millions of active 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 #8: 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 #9: F# for Consulting

Our bids for tendered contracts in quan are re he 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

Summary – The Data Agrees

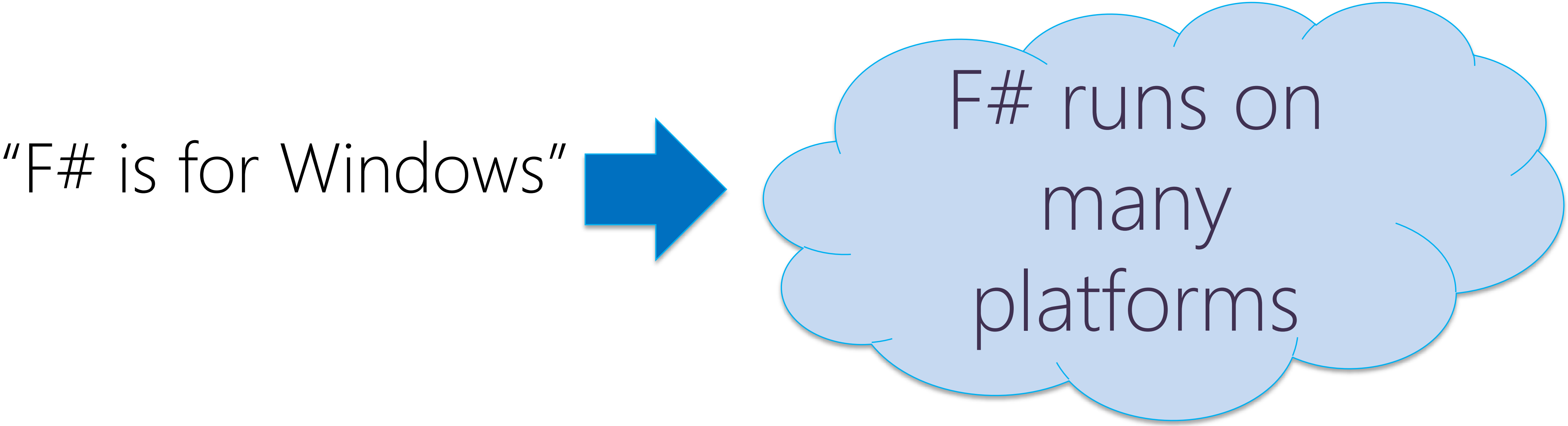
Simple, correct, robust code

Interoperability improves time-to-market

Strong-typing gives efficiency

Analytical developers empowered to solve complex problems

F# is changing…

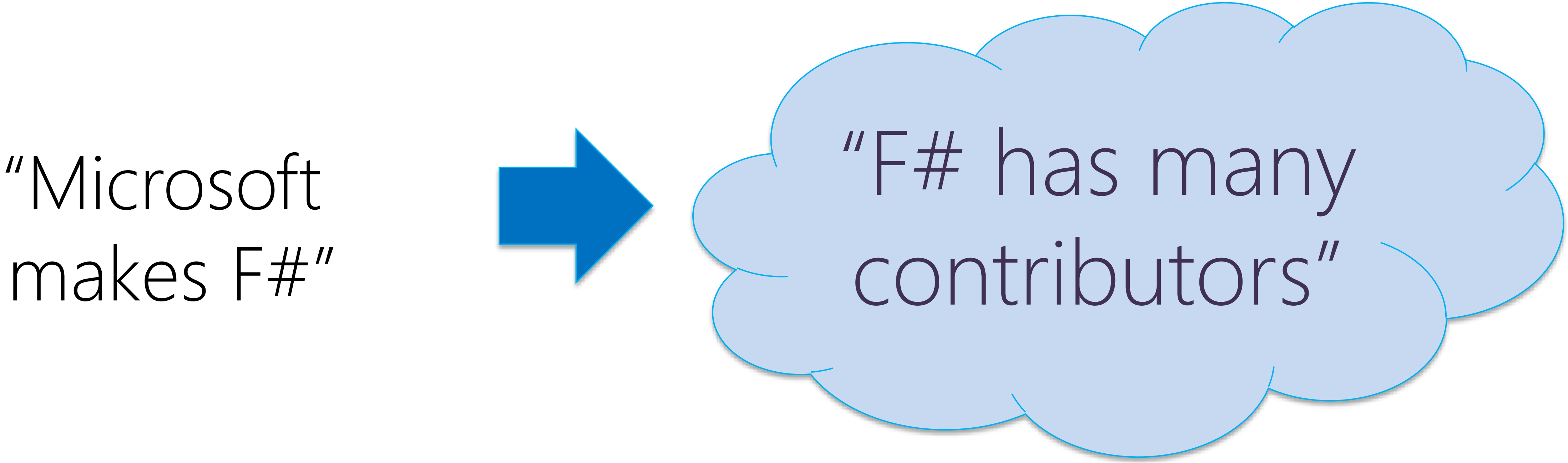


“F# is for Windows”

F# runs on many platforms

Overview

F# is changing…

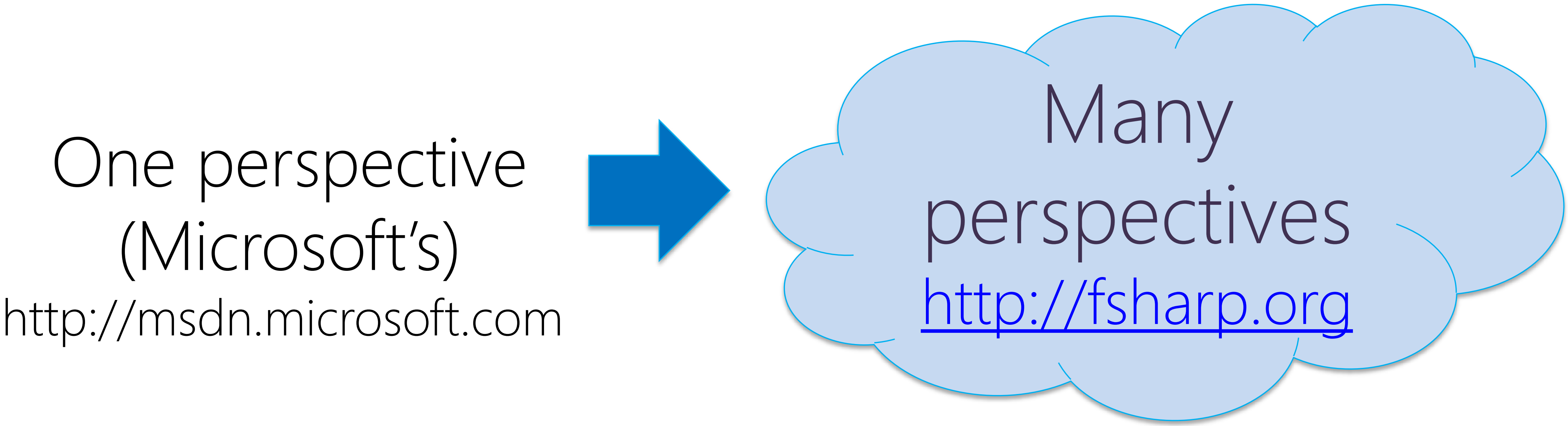


“Microsoft makes F#”

“F# has many contributors”

Overview

F# is changing…



One perspective
(Microsoft's)
http://msdn.microsoft.com

Many perspectives
http://fsharp.org

F# for Android

fsharp.org/use/android

F# for Linux, Mac

fsharp.org/use/linux
fsharp.org/use/mac

F# for iOS

fsharp.org/use/ios

Give your iOS world some F# spice!

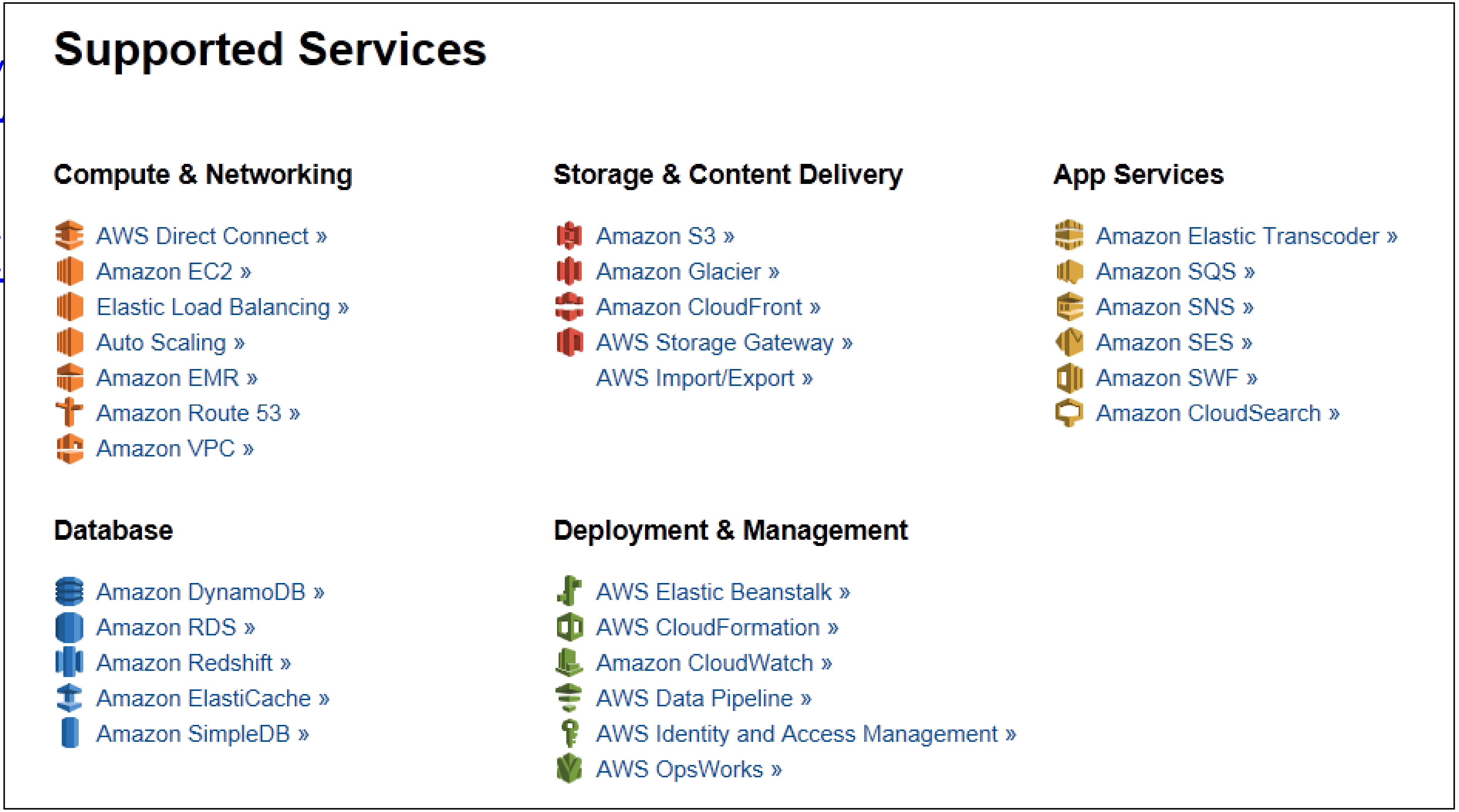
Xamarin

Microsoft

Amazon Web Services .NET SDKs

http:/

https:



Supported Services

Compute & Networking

- AWS Direct Connect »
- Amazon EC2 »
- Elastic Load Balancing »
- Auto Scaling »
- Amazon EMR »
- Amazon Route 53 »
- Amazon VPC »

Storage & Content Delivery

- Amazon S3 »
- Amazon Glacier »
- Amazon CloudFront »
- AWS Storage Gateway »
- AWS Import/Export »

App Services

- Amazon Elastic Transcoder »
- Amazon SQS »
- Amazon SNS »
- Amazon SES »
- Amazon SWF »
- Amazon CloudSearch »

Database

- Amazon DynamoDB »
- Amazon RDS »
- Amazon Redshift »
- Amazon ElastiCache »
- Amazon SimpleDB »

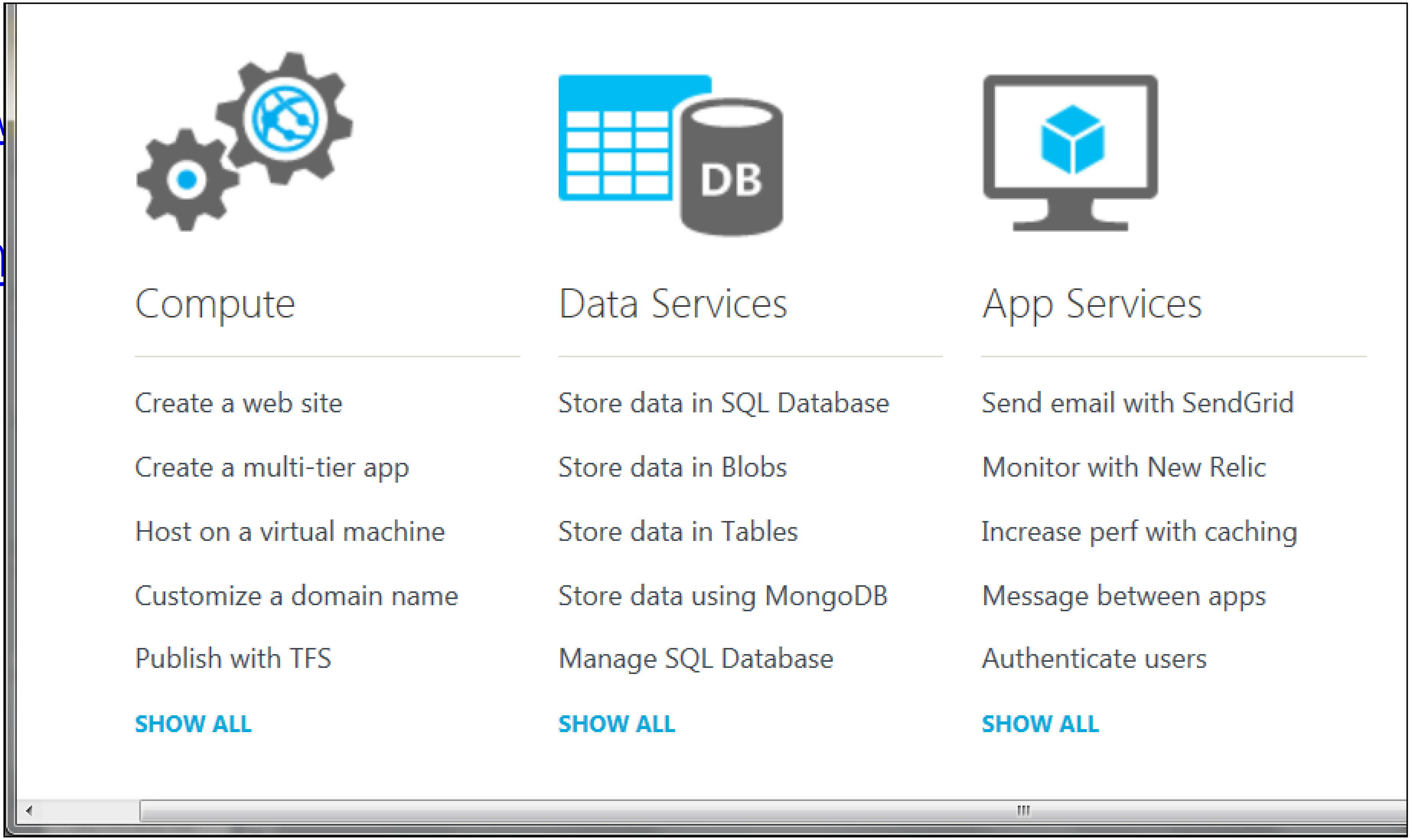
Deployment & Management

- AWS Elastic Beanstalk »
- AWS CloudFormation »
- Amazon CloudWatch »
- AWS Data Pipeline »
- AWS Identity and Access Management »
- AWS OpsWorks »

Azure .NET SDKs

http://www

https://gith



Compute

Create a web site

Create a multi-tier app

Host on a virtual machine

Customize a domain name

Publish with TFS

SHOW ALL

Data Services

Store data in SQL Database

Store data in Blobs

Store data in Tables

Store data using MongoDB

Manage SQL Database

SHOW ALL

App Services

Send email with SendGrid

Monitor with New Relic

Increase perf with caching

Message between apps

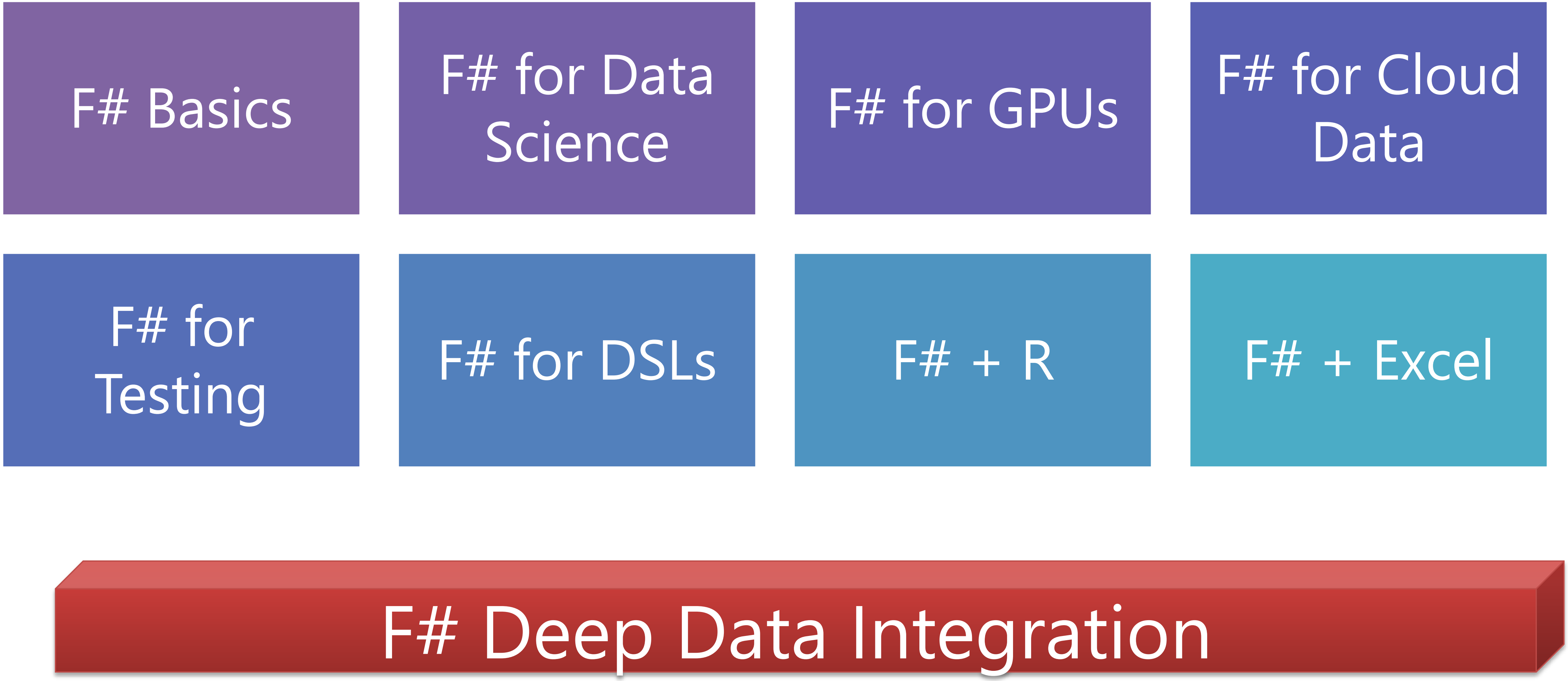
Authenticate users

SHOW ALL

Back to the main topic…

The Main Topic

You can easily find out more about…



F# Basics

F# for Data Science

F# for GPUs

F# for Cloud Data

F# for Testing

F# for DSLs

F# + R

F# + Excel

F# Deep Data Integration

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

Demo

Problem: Integrate all of freebase.com

“as if it were a library”

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

A Type Provider is….

"Just like a library"

"A design-time component that computes a space of types and methods on-demand…"

"An adaptor between data/services and the .NET type system…"

"On-demand, scalable compile-time provision of type/module definitions…"

Theme #1

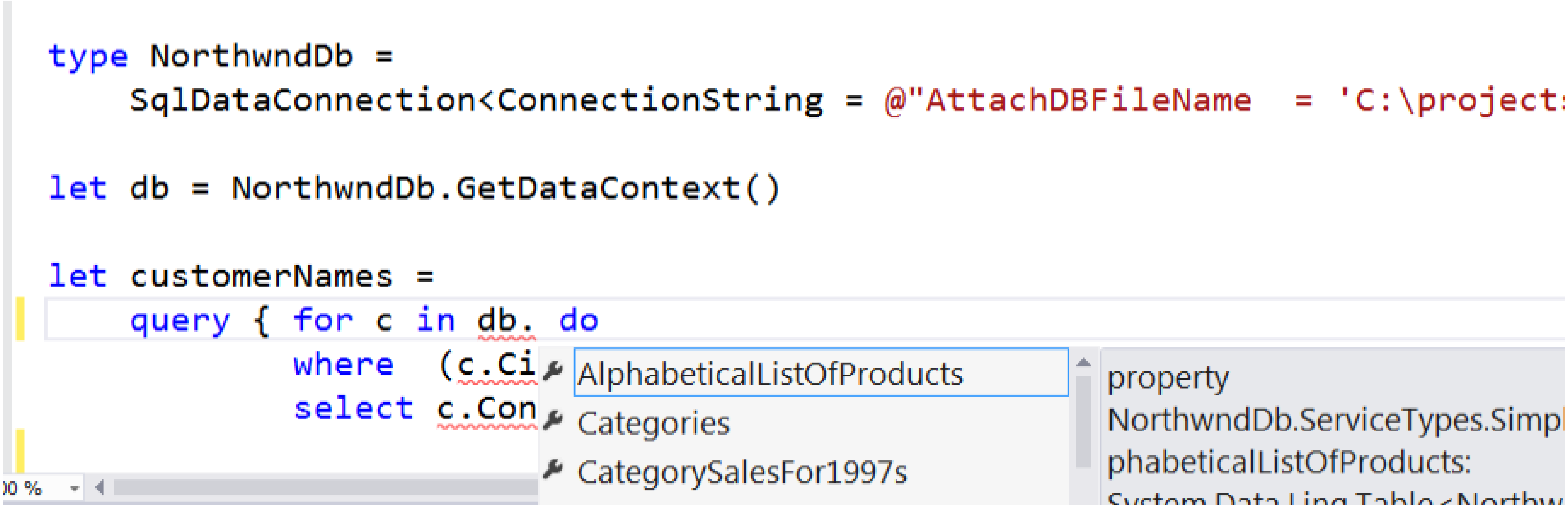
On-Demand Types = Internet Scalable Magic

Theme #2

Many Data Sources, One Mechanism

All your types are belong to us….

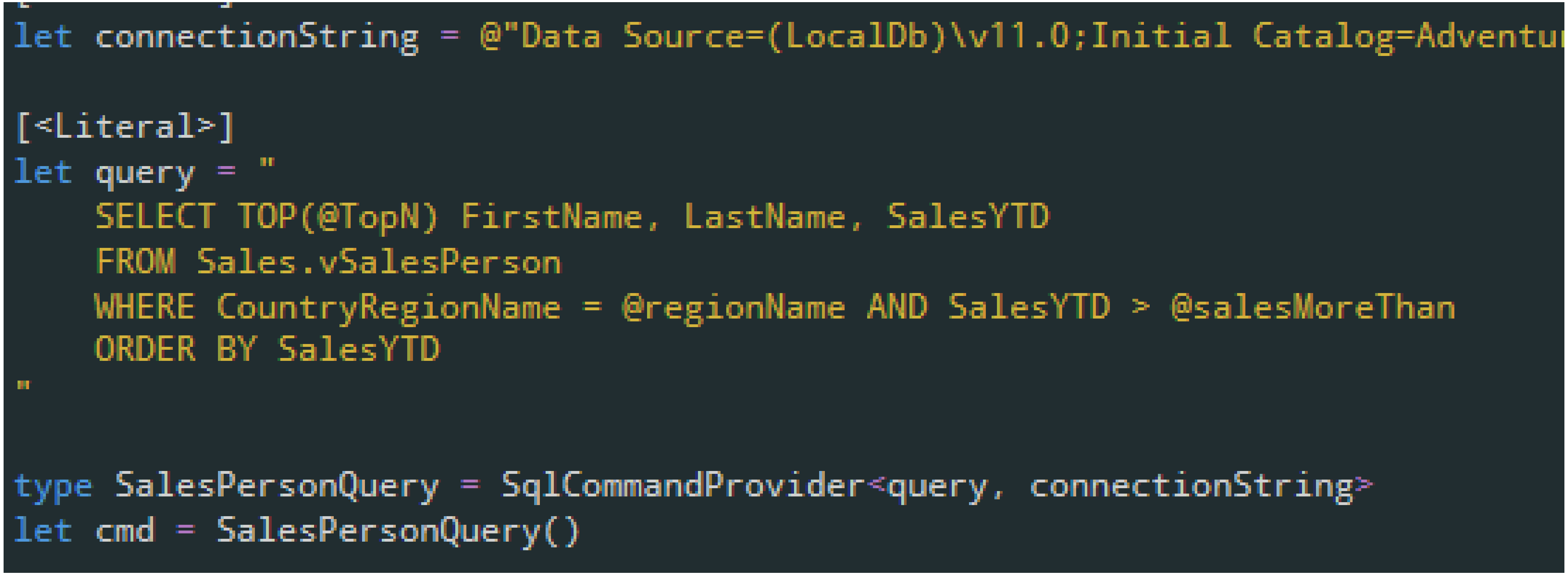
SQL #1



```
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
```

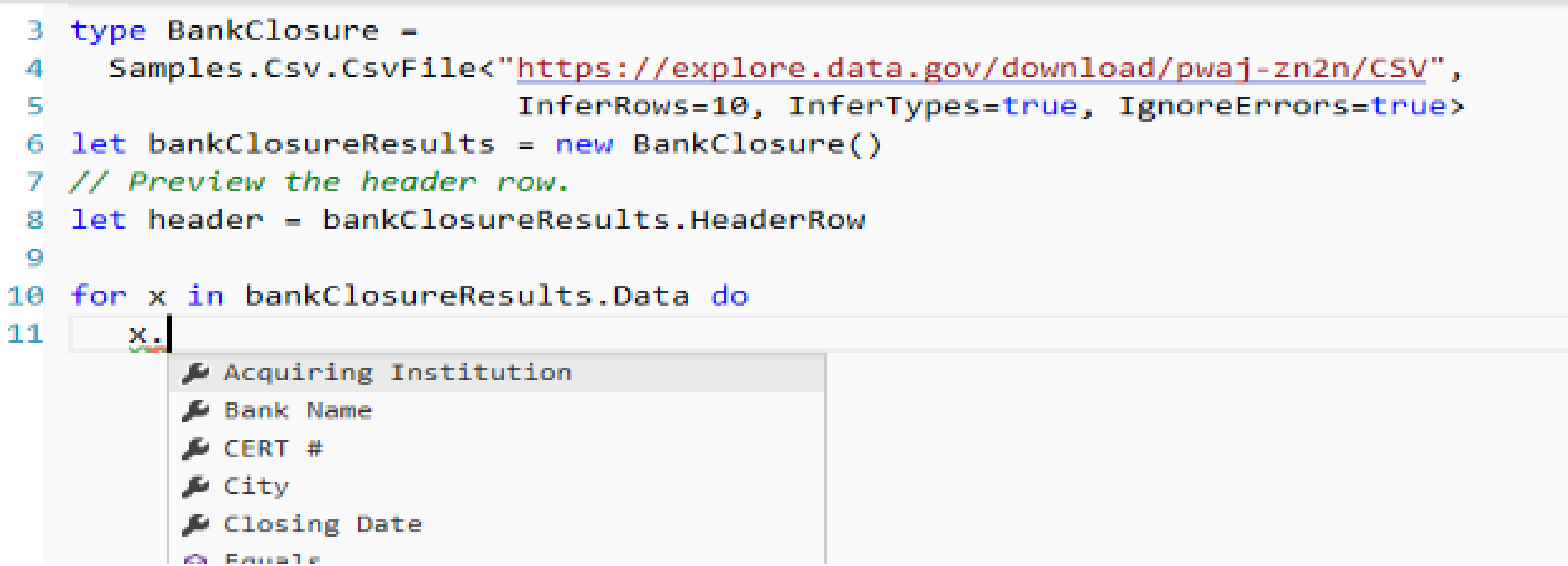
AlphabeticalListOfProducts
Categories
CategorySalesFor1997s
property
NorthwndDb.ServiceTypes.Simpl
phabeticalListOfProducts:
System.Data.Linq.Table<Northw
00 %

SQL #2



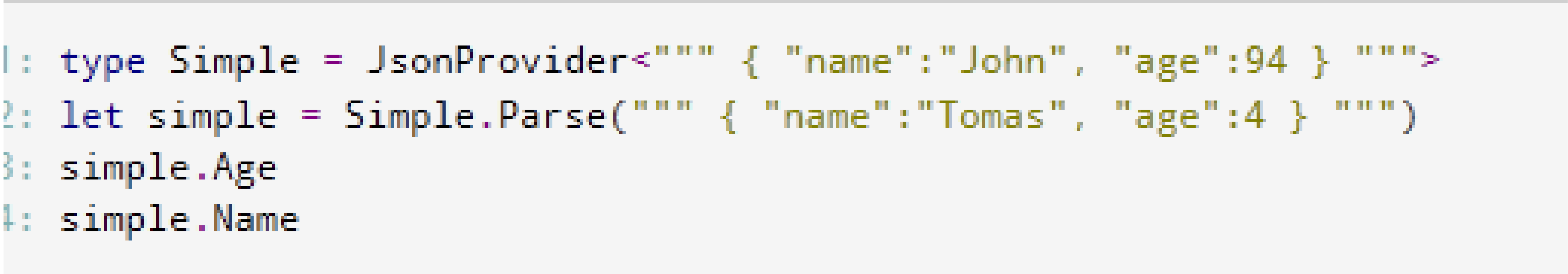
```
let connectionString = @"Data Source=(LocalDb)\v11.0;Initial Catalog=Adventur

[<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



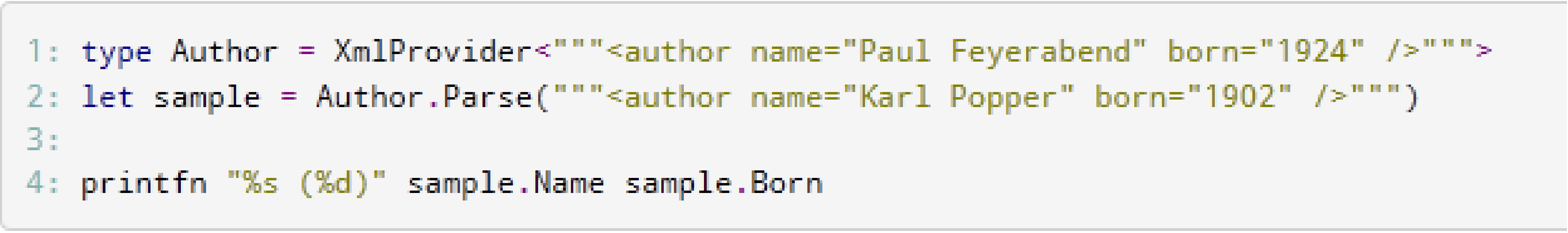
```
type BankClosure =
  Samples.Csv.CsvFile<"https://explore.data.gov/download/pwaj-zn2n/CSV",
                       InferRows=10, InferTypes=true, IgnoreErrors=true>
let bankClosureResults = new BankClosure()
// Preview the header row.
let header = bankClosureResults.HeaderRow

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

JSON



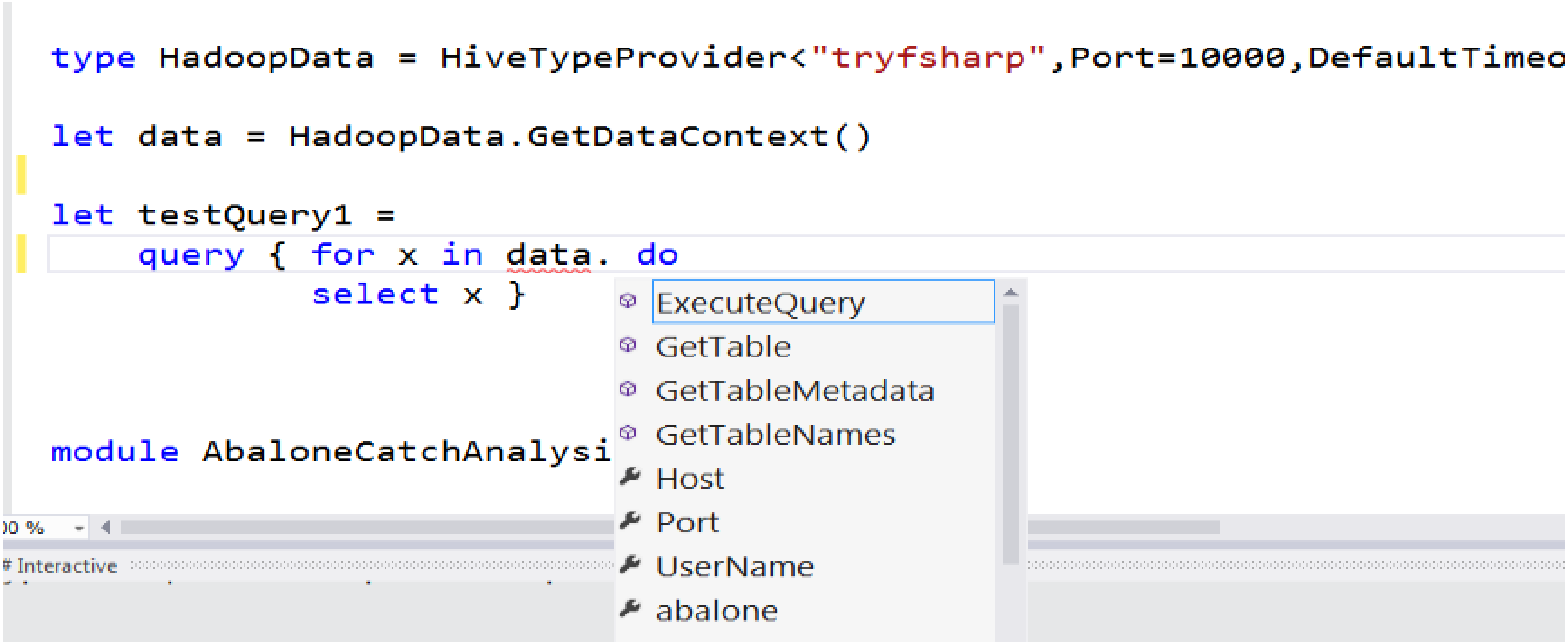
```
type Simple = JsonProvider<""" { "name":"John", "age":94 } """>
let simple = Simple.Parse(""" { "name":"Tomas", "age":4 } """)
simple.Age
simple.Name
```

XML



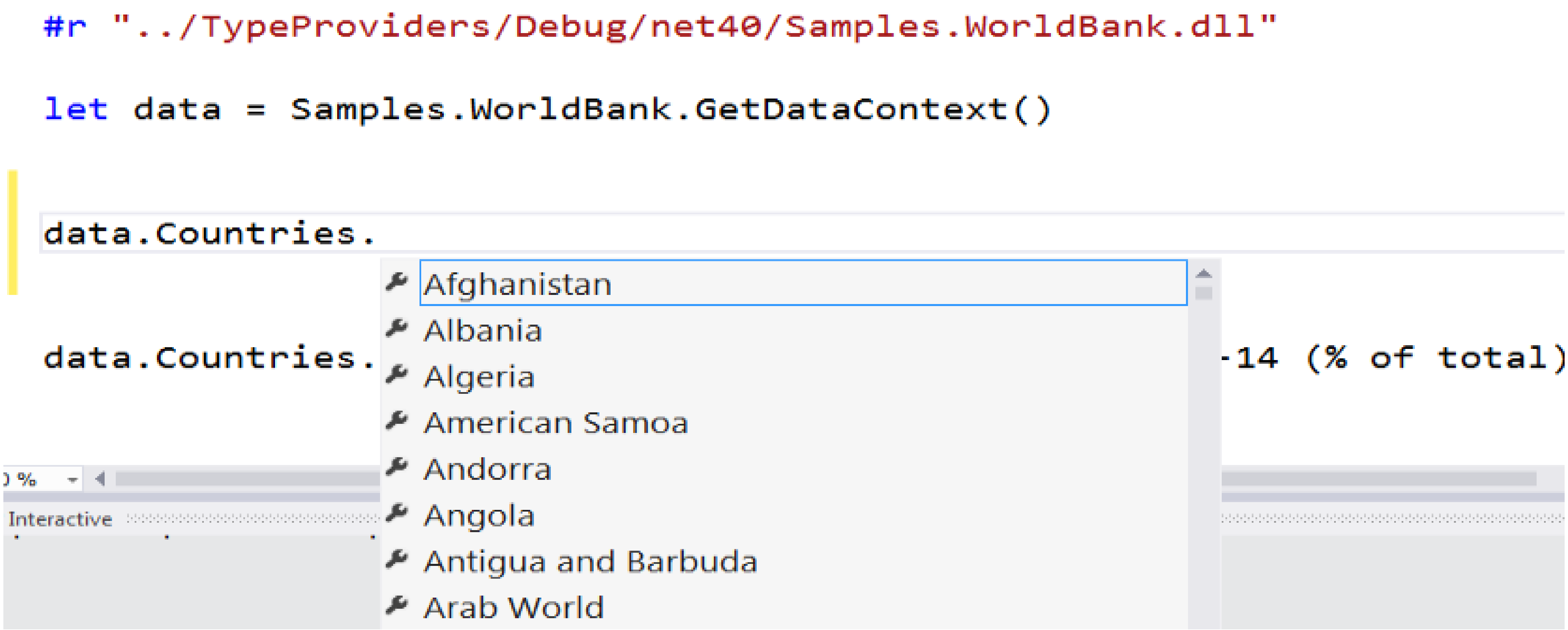
```
type Author = XmlProvider<"""<author name="Paul Feyerabend" born="1924" />""">
let sample = Author.Parse("""<author name="Karl Popper" born="1902" />""")

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

Hadoop/Hive



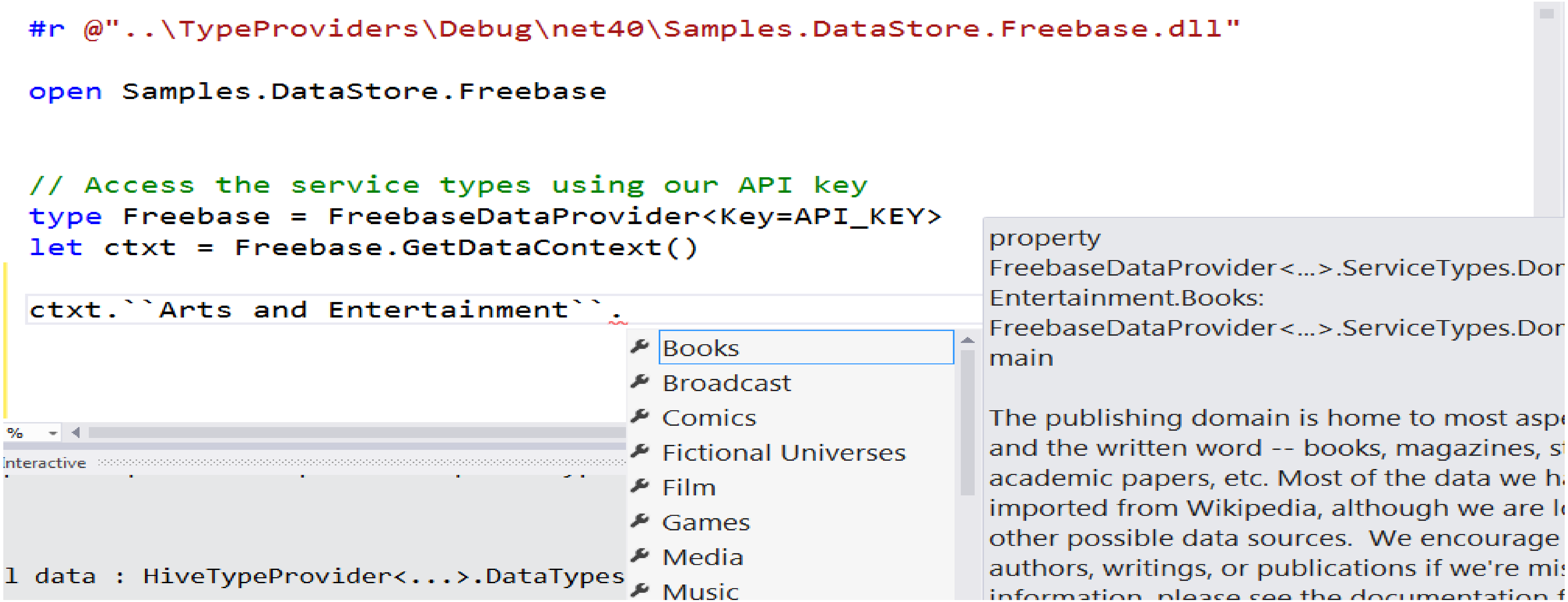
```
type HadoopData = HiveTypeProvider<"tryfsharp",Port=10000,DefaultTimeo

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



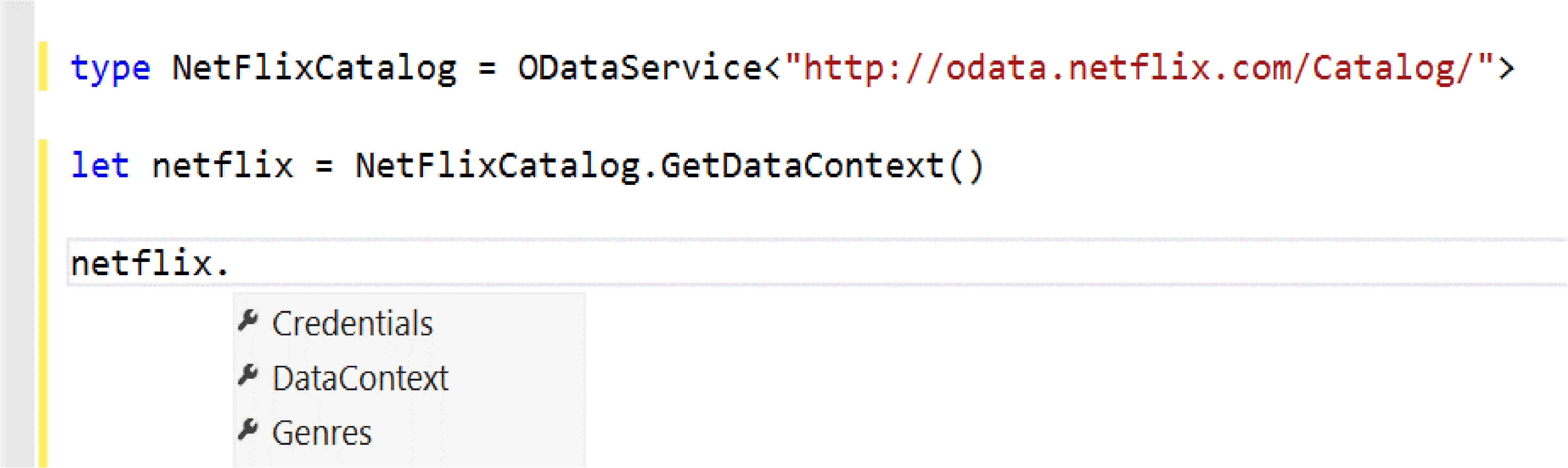
```
#r @"..\TypeProviders\Debug\net40\Samples.DataStore.Freebase.dll"

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``.
```

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 mis
information, please see the documentation f

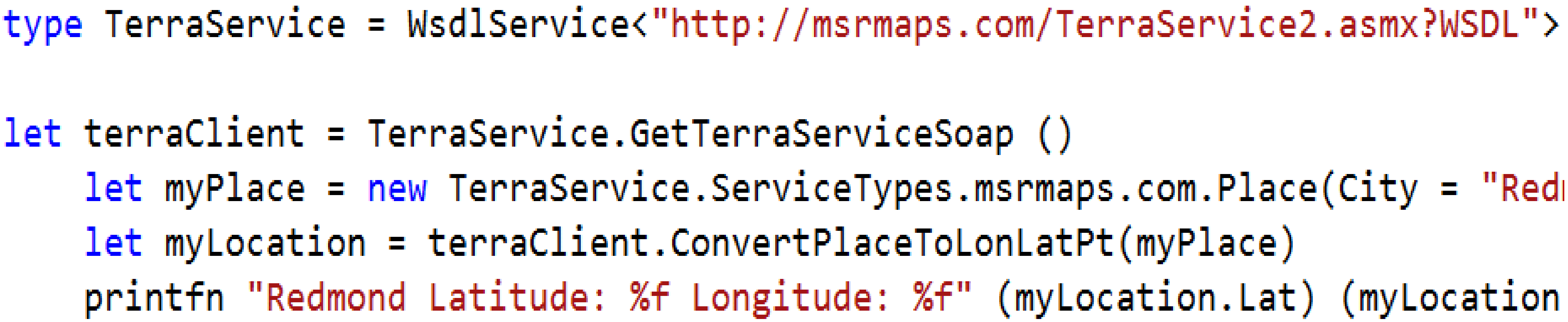
```
l data : HiveTypeProvider<...>.DataTypes
```

OData



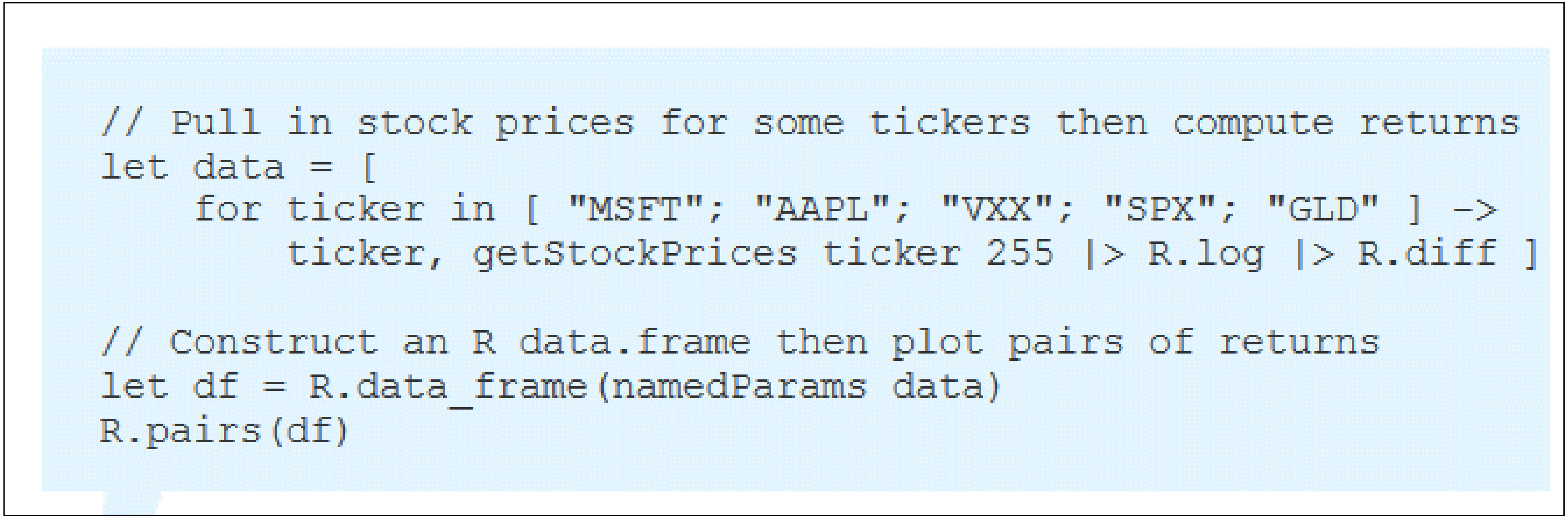
```
type NetFlixCatalog = ODataService<"http://odata.netflix.com/Catalog/">

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
```

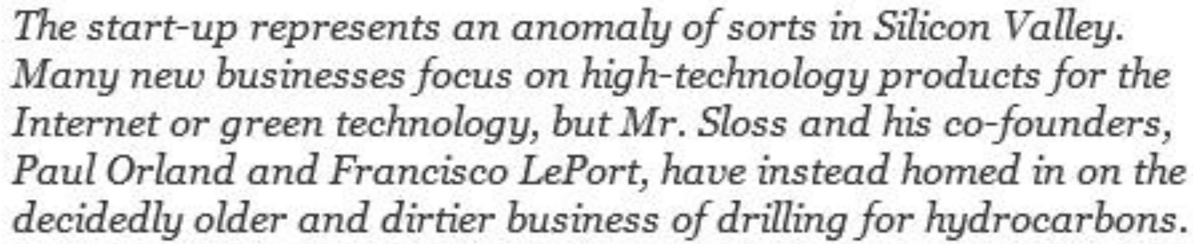
R



```
// 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)
```

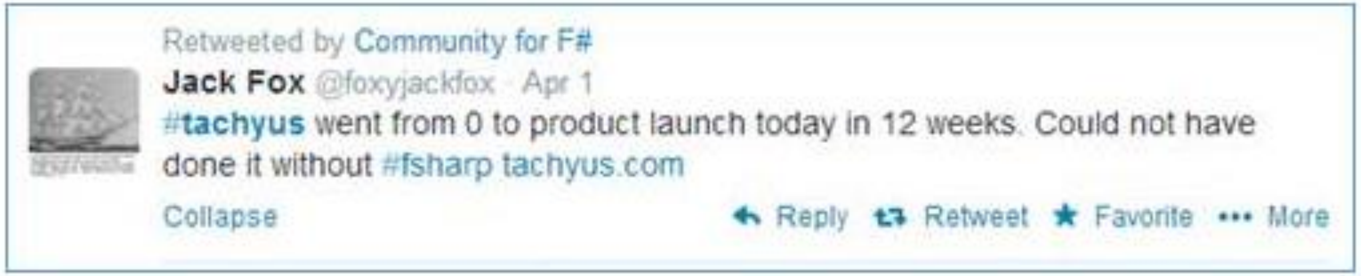
SQL #2 - Application

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#

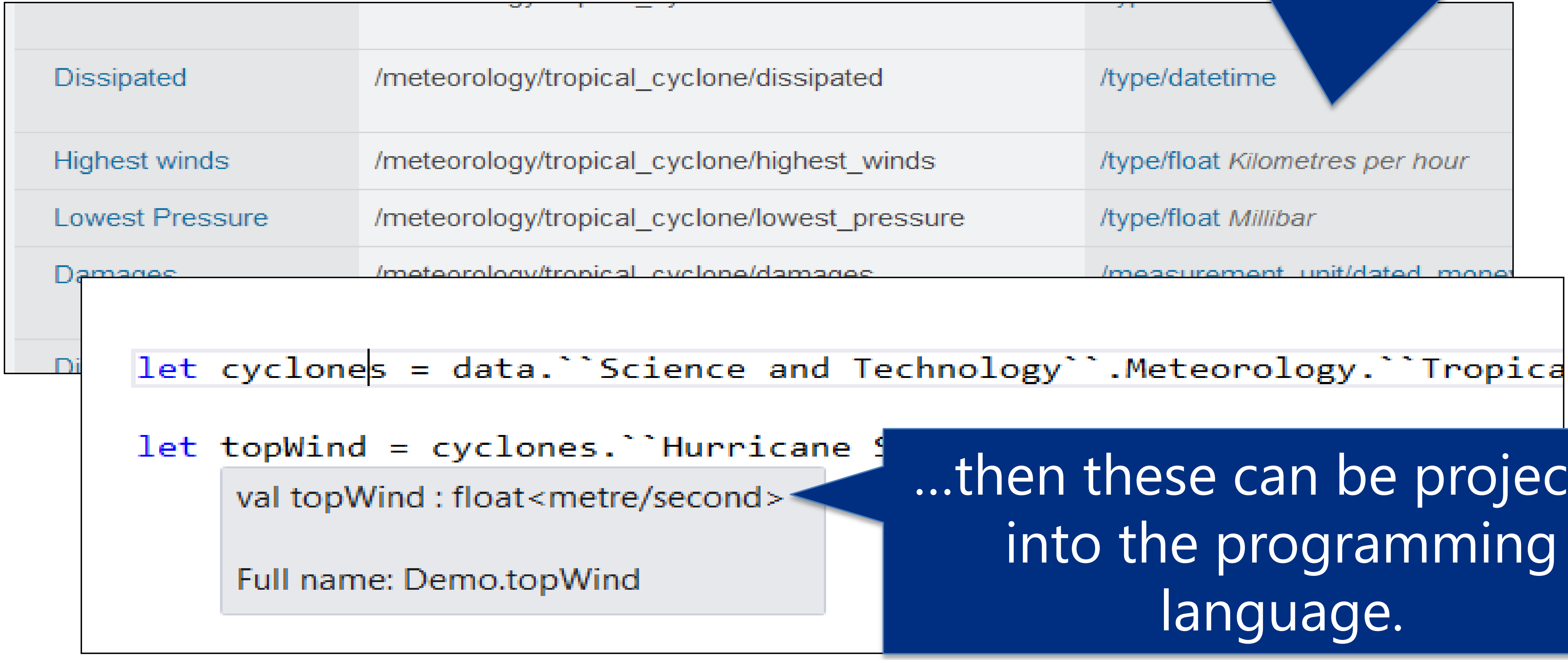
Jack Fox @foxyjackfox · Apr 1

#**tachyus** went from 0 to product launch today in 12 weeks. Could not have done it without #fsharp tachyus.com

Providing Units of Measure

via F#'s Units of Measure

If the metadata contains units…



Dissipated	/meteorology/tropical_cyclone/dissipated	/type/datetime
Highest winds	/meteorology/tropical_cyclone/highest_winds	/type/float *Kilometres per hour*
Lowest Pressure	/meteorology/tropical_cyclone/lowest_pressure	/type/float *Millibar*
Damages	/meteorology/tropical_cyclone/damages	/measurement_unit/dated_money
Di		

```
let cyclones = data.``Science and Technology``.Meteorology.``Tropica

let topWind = cyclones.``Hurricane S
```

```
val topWind : float<metre/second>

Full name: Demo.topWind
```

…then these can be projected into the programming language.

FSharp.Data

fsharp.github.io/FSharp.Data

on NuGet, use it in Visual Studio today

Summary

Scalable (meta)data integration into programming is a key challenge of our era

“F# type providers” are a simple, powerful point in the design space

The techniques have many, many applications

People use this “for real” in production F# systems

In Summary

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

The safe choice for enterprise data programming

F#

Unbeatable data integration

Visual F# - tooling you can trust from Microsoft

http://fsharp.org

Questions?

Give your life
an F# edge!

http://fsharp.org

Microsoft