

**F# 3.0:**

# **Strongly Typed Programming in the Information Rich World**

Don Syme, Principal Researcher,  
Microsoft Research, UK



**Today's talk is very simple**

**Proposition 1**  
**The world is incredibly**  
**information-rich**

**Proposition 2**  
**Modern financial enterprises are**  
**incredibly information-rich**

**Proposition 3**  
**Our languages are information-**  
**sparse**

# **Proposition 4**

## **This is a problem**

**(especially for strongly typed programming)**

**With F# we want to help fix this...**

**The mechanism we're adding  
to F# is called **Type Providers****

**LINQ + Type Providers**

**=**

**Language Integrated Data and  
Services**



# Two aims today

**Demonstrate what we're doing in F# 3.0**

**Explore applications in a range of data spaces**

**But first...**

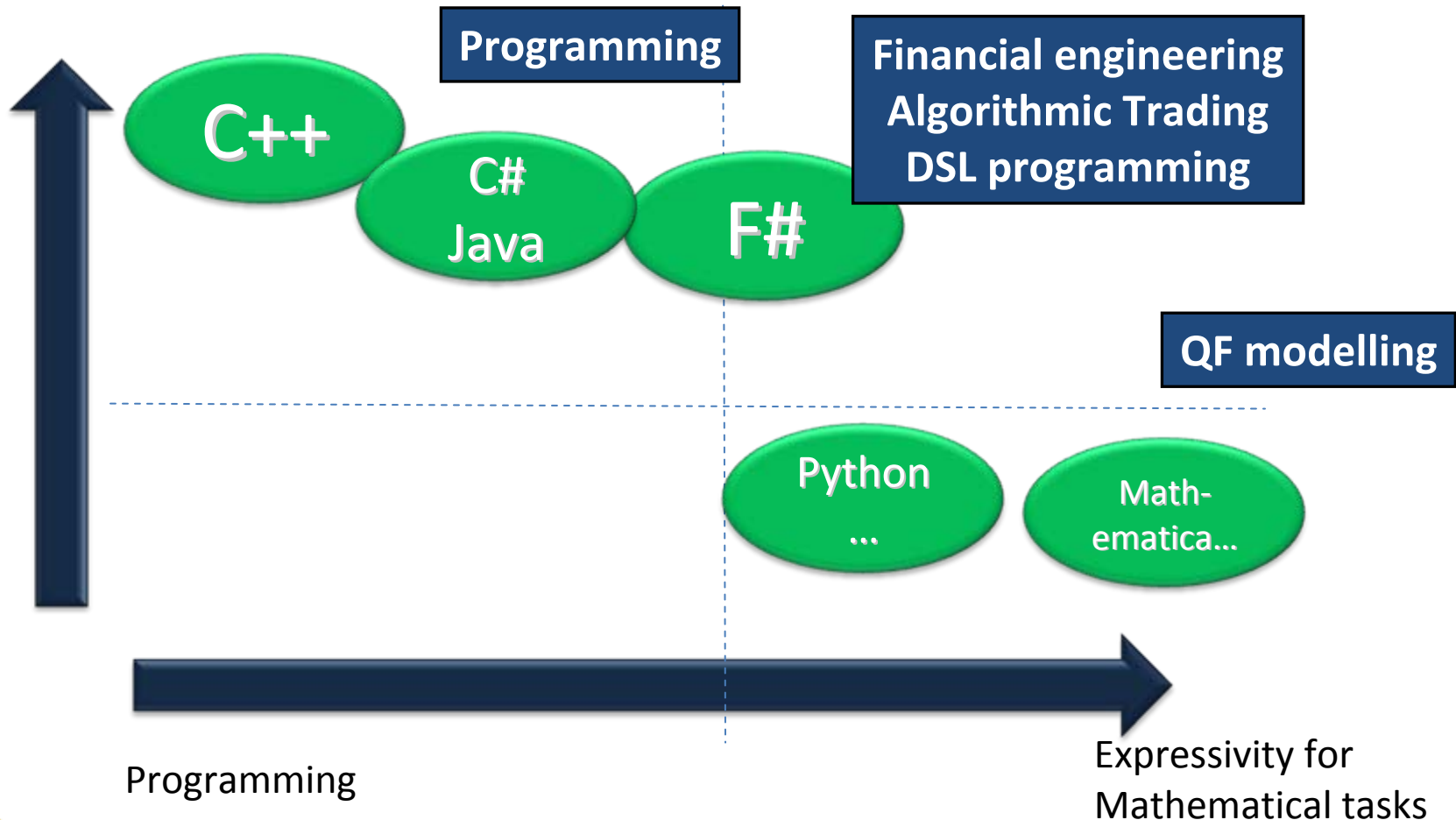
**What is F#?**

# F# is...

...a **practical, supported, interoperable, functional language** that allows you to write **simple code** to solve **complex problems**.

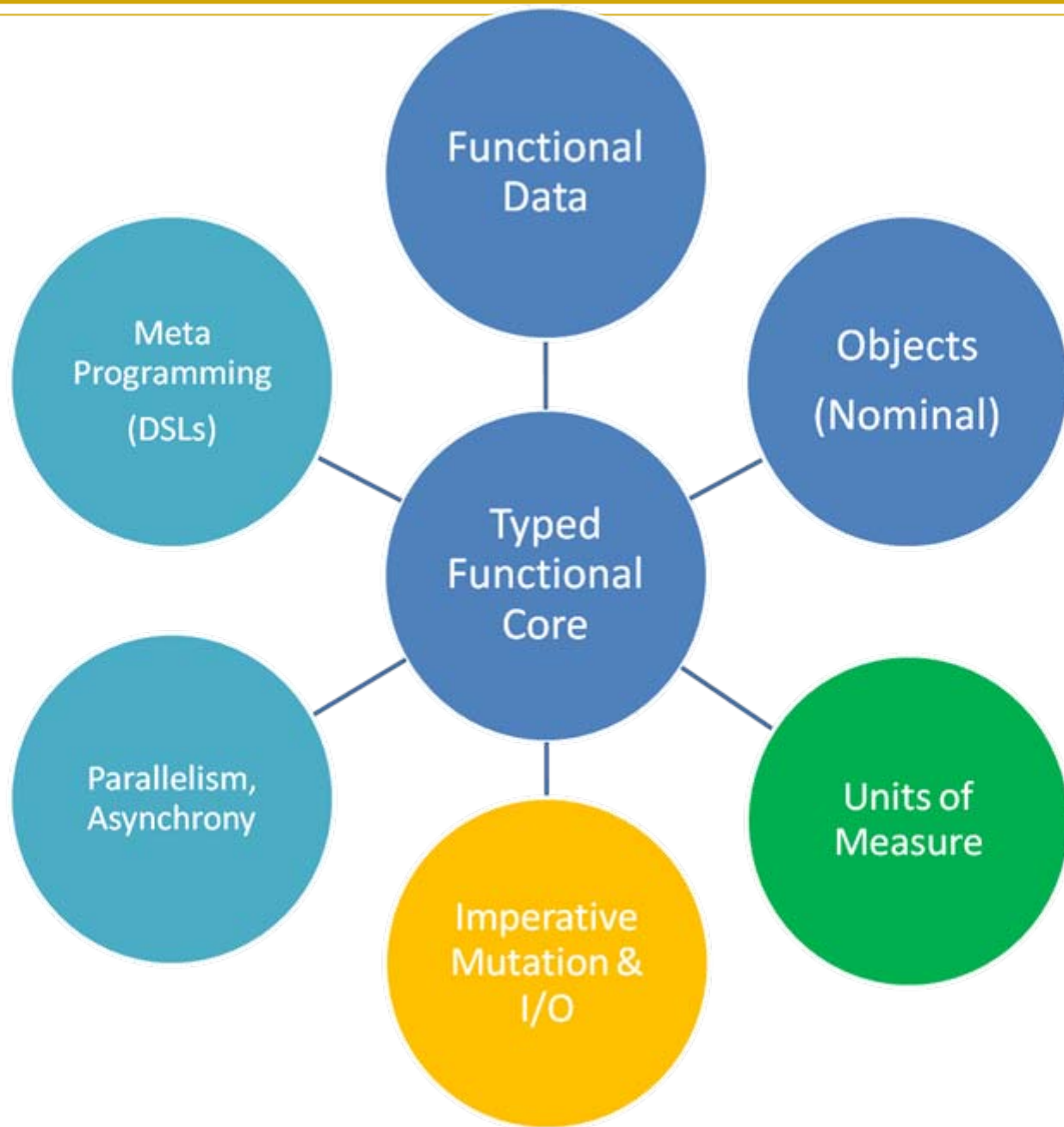
# Crossing boundaries

“Fresh Code” Performance  
Professional Development



# Why is F# appealing in finance?

- Functional programming fits with financial work
  - Programmatic modelling
  - Compositional parallel & GPU programming
  - Domain specific languages – internal and external
  - Efficient execution
- Plays differently for different roles:
  - **Quants** contribute to component development
  - **Architects** explore hard problems fluently
  - **Developers** tackle parallel and async programming



# Simplicity: Functions as Values




F#

```
type Command = Command of (Rover -> unit)

let BreakCommand =
    Command(fun rover -> rover.Accelerate(-1.0))

let TurnLeftCommand =
    Command(fun rover -> rover.Rotate(-5.0<degs>))
```



```
abstract class Command
{
    public virtual void Execute();
}
abstract class RoverCommand : Command
{
    protected Rover Rover { get; private set; }

    public RoverCommand(MarsRover rover)
    {
        this.Rover = rover;
    }
}
class BreakCommand : RoverCommand
{
    public BreakCommand(Rover rover) : base(rover)
    {
    }
    public override void Execute()
    {
        Rover.Rotate(-5.0);
    }
}
class TurnLeftCommand : RoverCommand
{
    public TurnLeftCommand(Rover rover) : base(rover)
    {
    }
    public override void Execute()
    {
        Rover.Rotate(-5.0);
    }
}
```

# Simplicity: Functional Data

```
let swap (x, y) = (y, x)
```

A red rounded square icon with the text 'F#' in white.

```
let rotations (x, y, z) =  
  [ (x, y, z);  
    (z, x, y);  
    (y, z, x) ]
```

```
let reduce f (x, y, z) =  
  f x + f y + f z
```

```
Tuple<U,T> Swap<T,U>(Tuple<T,U> t)  
{  
    return new Tuple<U,T>(t.Item2, t.Item1)  
}
```

```
ReadOnlyCollection<Tuple<T,T,T>> Rotations<T>(Tuple<T,T,T> t)  
{  
    new ReadOnlyCollection<int>  
        (new Tuple<T,T,T>[]  
            { new Tuple<T,T,T>(t.Item1,t.Item2,t.Item3);  
              new Tuple<T,T,T>(t.Item3,t.Item1,t.Item2);  
              new Tuple<T,T,T>(t.Item2,t.Item3,t.Item1); });  
}
```

```
int Reduce<T>(Func<T,int> f,Tuple<T,T,T> t)  
{  
    return f(t.Item1) + f(t.Item2) + f (t.Item3);  
}
```

A purple rounded square icon with the text 'C#' in white.



# Understanding F#

**F# 3.0**

Data Access,  
Information,  
Services,  
External DSLs

**F# 2.0**

Transformation,  
Analysis,  
Algorithms,  
Code,  
Parallel,  
Internal DSLs

C# (designers)  
F# (code)  
WebSharper  
(HTML5)  
Javascript...

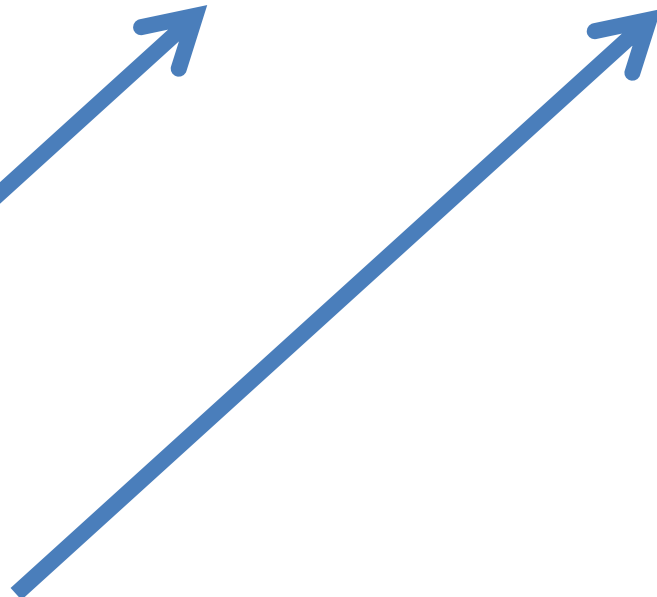
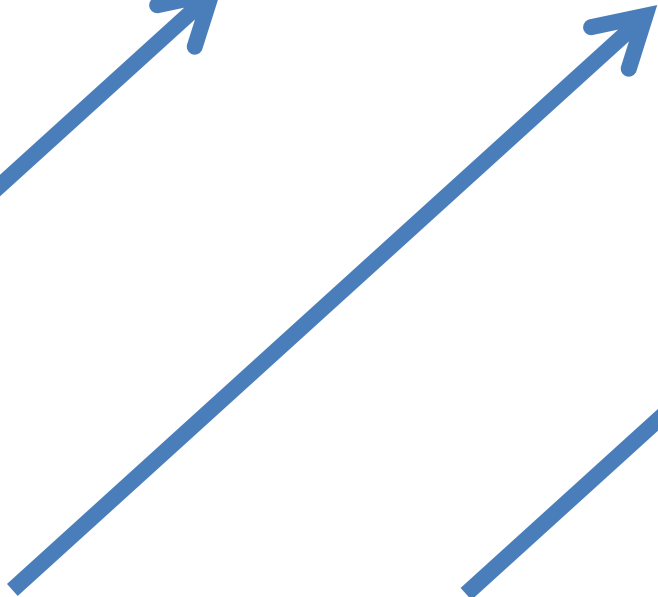
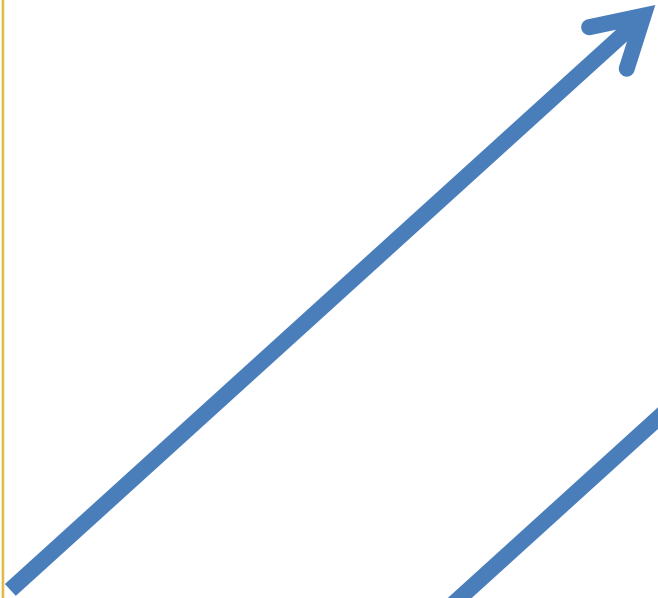
Presentation,  
Publication,  
Experience

# The Big Trends

**WEB**

**MULTICORE**

**DATA**



# 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)

# Examples #2/#3: Finance companies



## 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

To maintain its competitive standing and its reputation among agents for being easy to do business with, Grange Insurance needed to keep its rating engine working at top performance.

## Solution

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

## Insurance Company Improves Time-to-Market with Enhanced Rating Engine

*"With this streamlined development cycle, we can rapidly deliver more powerful solutions that our policyholders can deliver more choices and better policyholders that much faster."*

*Glenn Watson, Associate Vice President, Personal Lines, IT, Grange*

For nearly 75 years, Grange Insurance has offered products and services to policyholders in more than 13 states. To maintain its well-earned reputation and its reputation among agents for being easy to do business with, Grange Insurance needed to keep its rating engine working at top performance. To address its needs, the bank deployed Microsoft F#, the 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.

**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 Functional Language to Speed Development by 50 Percent

*"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 previous tools."*

*Director at a large European financial services firm*

A large financial services firm in Europe sought new development tools that could cut costs, boost productivity, and improve the quality of its mathematical models. To address its needs, the bank deployed Microsoft F#, the 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 customers

## Part 2

# F# 3.0 Information Rich Programming

# A Challenge

**Task #1: A Chemistry Elements Class Library**

**Task #2: Repeat for all Sciences, Businesses, ...**

# Language Integrated Web Data

demo

# **A Type Provider is....**

**“A compile-time component that provides a computed space of types and methods on-demand ...”**

**“A compiler plug-in...”**

**“An adaptor between data/services and the .NET type system...”**



```

// Freebase.fsx
// Example of reading from freebase.com in F#
// by Jomo Fisher
#r "System.Runtime.Serialization"
#r "System.ServiceModel.Web"
#r "System.Web"
#r "System.Xml"

open System
open System.IO
open System.Net
open System.Text
open System.Web
open System.Security.Authentication
open System.Runtime.Serialization

[<DataContract>]
type Result<'TResult> = {
    [<field: DataMember(Name="Code")> Code:string
    [<field: DataMember(Name="Result")> TResult:'TResult
    [<field: DataMember(Name="Message")> Message:string
}

[<DataContract>]
type ChemicalElement = {
    [<field: DataMember(Name="Name")> Name:string
    [<field: DataMember(Name="BoilingPoint")> BoilingPoint:string
    [<field: DataMember(Name="AtomicMass")> AtomicMass:string
}

let Query<'T>(query:string) : 'T =
    let query = query.Replace("'", "\"")
    let queryUrl = sprintf "http://api.freebase.com/api/service/mqlread?query=%s"
        "{\\"query\\":\"+query+"}"

    let request : HttpWebRequest = downcast WebRequest.Create(queryUrl)
    request.Method <- "GET"
    request.ContentType <- "application/x-www-form-urlencoded"

    let response = request.GetResponse()

    let result =
        try
            use reader = new StreamReader(response.GetResponseStream())
            reader.ReadToEnd();
        finally
            response.Close()

    let data = Encoding.Unicode.GetBytes(result);
    let stream = new MemoryStream()
    stream.Write(data, 0, data.Length);
    stream.Position <- 0L

    let ser = Json.DataContractJsonSerializer(typeof<Result<'T>>)
    let result = ser.ReadObject(stream) :?> Result<'T>
    if result.Code <> "/api/status/ok" then
        raise (InvalidOperationException(result.Message))
    else
        result.Result

let elements = Query<ChemicalElement
array>("[{'type':'/chemistry/chemical_element','name':null,'boiling_point':null,'atomic_mass':null}]")

elements |> Array.iter(fun element->printfn "%A" element)

```

**How would we do this previously?**

**Note: F# itself still contains no data**

**Open architecture**

**You can write your own type  
provider**

# Language Integrated Data Market Directory

demo

**LMax**

demo

# SQL

```
type Data = SqlConnection<"Server='.\SQLEXPRESS'..">
```

```
let db = SQL.GetDataContext()
```

```
db.Customers
```



Fluent, Typed  
Access To  
SQL

# OData

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

```
let service = Netflix.GetDataContext()
```

```
service.Titles
```



Fluent, Typed  
Access To  
OData

# Web Services

```
type Data = Wsd1Service<"http://www.xignite.com/xFutures.asmx?WSDL">  
  
let financials = Data.GetServiceContext()  
  
financials.GetQuotes "IBM"
```



Fluent, Typed  
Access To  
WSDL

# F# 3.0: Queries

```
let avatarTitles =  
    query { for t in netflix.Titles do  
            where (t.Name.Contains "Avatar")  
            select t }
```



# F# 3.0: Queries

```
let avatarTitles =  
    query { for t in netflix.Titles do  
            where (t.Name.Contains "Avatar")  
            sortBy t.Name  
            select t }
```

# F# 3.0: Queries

```
let avatarTitles =  
    query { for t in netflix.Titles do  
            where (t.Name.Contains "Avatar")  
            sortBy t.Name  
            select t  
            take 100 }
```

## **Conclusion 1**

**Huge Information Spaces can be  
Software Components**

## **Conclusion 2**

**Multiple data standards with one  
simple mechanism**

## **Conclusion 3**

**Integrated Data Access Empowers  
Both Programmers and Analysts**

## **Summary**

**The financial world is massively  
information rich**

**Our enterprise financial programming needs  
to be information-rich too**

**Information-richness changes how we think  
about programming and analysis**

# Thank You!

# Questions?

Contacts: [dsyme@microsoft.com](mailto:dsyme@microsoft.com)  
[www.fsharp.net](http://www.fsharp.net), <http://blogs.msdn.com/b/fsharpteam>  
Twitter: @dsyme, #fsharp

# F# and Open Source

F# 2.0 compiler+library open source drop

Apache 2.0 license

Runs on Linux, Mac, Windows, Browser



# F# for the Browser & Web

[www.tryfsharp.org](http://www.tryfsharp.org)

(F# Console + Tutorials)

[pitfw.posterous.com](http://pitfw.posterous.com)

(F# to JS/HTML5, Community)

[websharper.com](http://websharper.com)

(F# to JS/HTML5, Product)