

F# for Trading



Phillip Trelford
Trayport

F# Community



London, United Kingdom



Founded Feb 4, 2010

Members	442
Group reviews	17
Upcoming Meetups	2
Past Meetups	22
Our calendar	
Follow us	



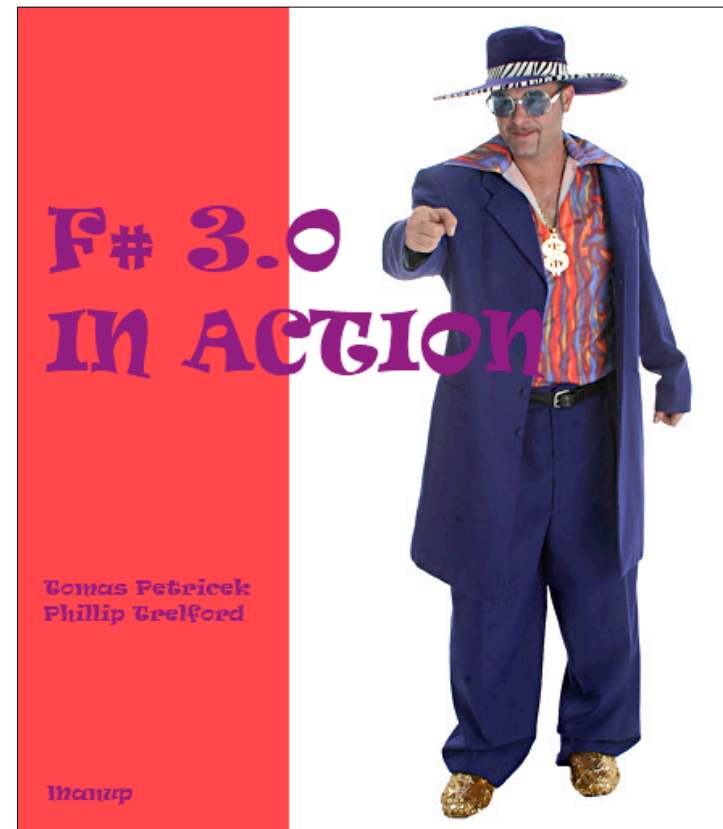
New York, NY

Founded Nov 3, 2010

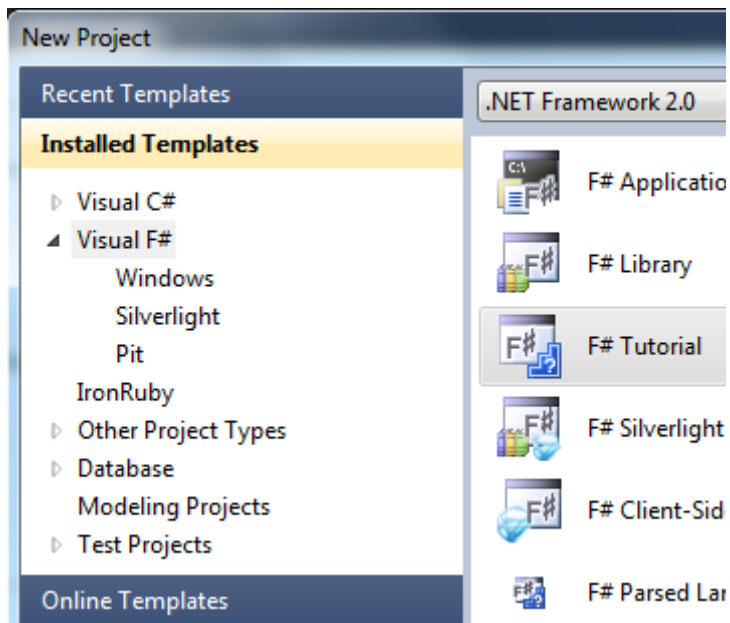
F#ers	429
Group reviews	27
Past Meetups	20
Our calendar	
Follow us	

F# Language

- Strongly Typed
- Functional First
- Object Orientated
- Open Source
- First Class .Net language
- In Visual Studio



Visual Studio



```
/// A very simple constant integer
let int1 = 1

/// A second very simple constant integer
let int2 = 2

/// Add two integers
let int3 = int1 + int2

// Functions on integers
// -----

/// A function on integers
let f x = 2*x*x - 5*x + 3

/// The result of a simple computation
let result = f (int3 + 4)

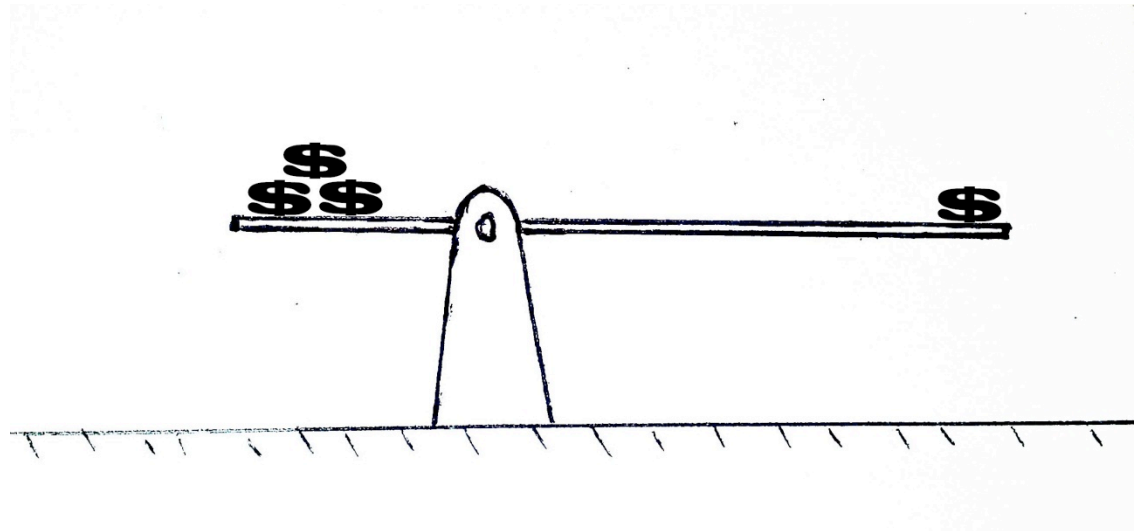
/// Another function on integers
let increment x = x + 1
```

Joule Energy Trading Screen



Leverage

- Existing code
- Domain modelling
- Computation
- Concurrency
- Libraries




Specification by Example

cucumber 1.2.1

Behaviour Driven Development with elegance and joy

INSTALL > `gem install cucumber`

 Download  Subscribe

Authors

Aslak Helleøy

2,111,952 **194,961**
total downloads for this version

Owners



TickSpec: Breakpoint in text file

Addition.txt ▢ ✕

Feature: Addition

In order to avoid silly mistakes

As a math noob

I want to be told the sum of two numbers

Scenario: Add two numbers



Given I have entered 50 into the calculator

And I have entered 70 into the calculator

When I press add

Then the result should be 120 on the screen

Thoughtworks Tech Radar

March 2012

*“F# is excellent at concisely expressing business and **domain** logic.”*

*“Developers trying to achieve explicit business logic within an application may opt to express their **domain in F#** with the majority of **plumbing code in C#**.”*

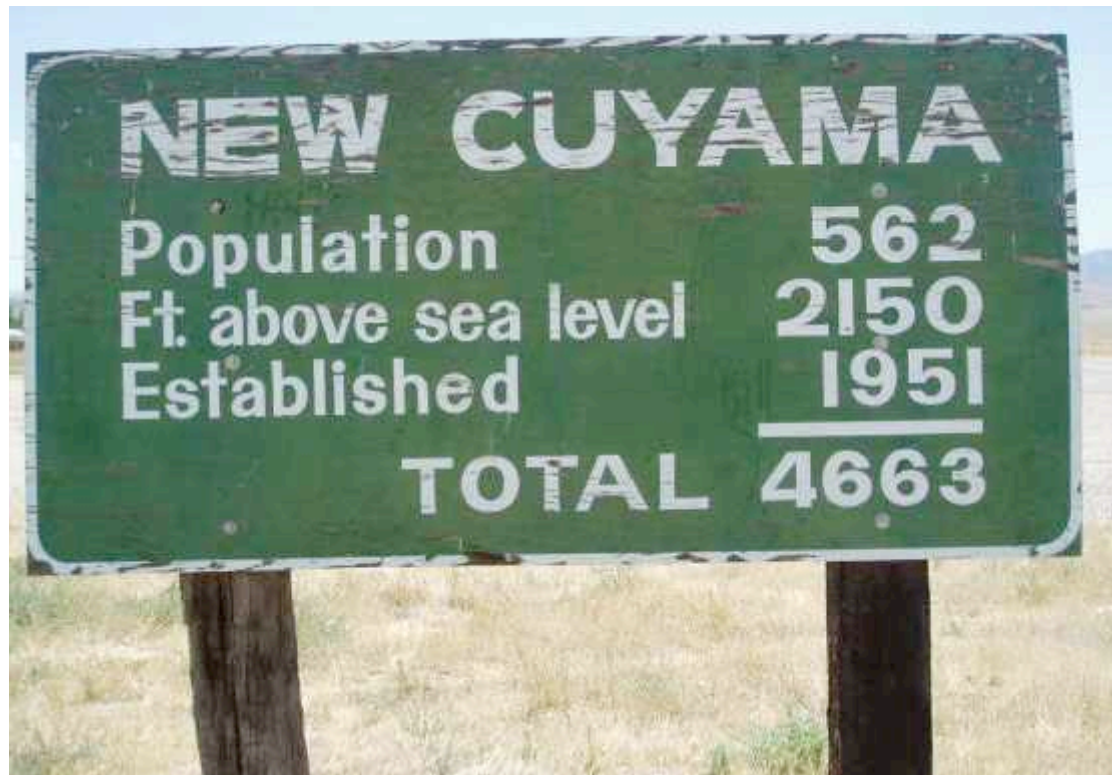
Order - Record type

```
type Order = {  
    Side    : Side  
    LimitPrice : Price  
    Quantity : Quantity  
    IsAllOrNone : bool  
}
```

Time in Force - Union type

```
type TimeInForce =  
  | Immediate  
  | GoodForDay  
  | GoodTillDate of DateTime
```

Units of Measure



Excel - Type Provider

Excel

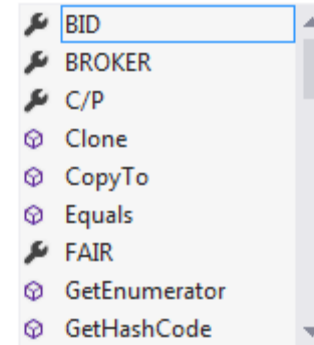
	A	B	C	D	E
1	SEC	UNDERLYING	STRATEGY	STYLE	MATURITIES
2	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89
3	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89
4	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89
5	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89
6	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89
7	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89
8	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89
9	ASI	JPY-NIKKEI 225	DIV_SWAP	A	01-DEC-89

Visual Studio

```
#r @".\bin\Debug\ExcelTypeProvider.dll"

open Samples.FSharp.ExcelProvider

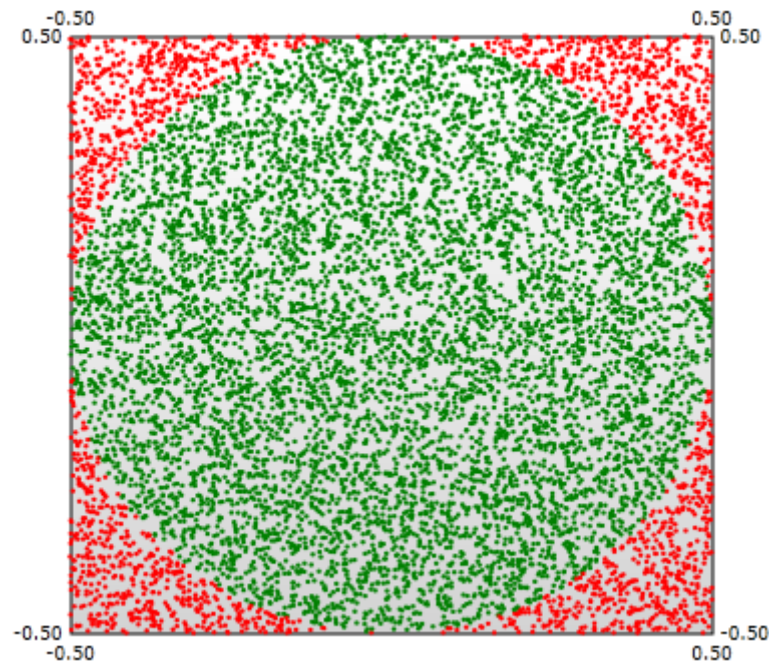
let file = new ExcelFile<"BookTest.xls">
let firstrow = file.Data |> Seq.head
firstrow.
```



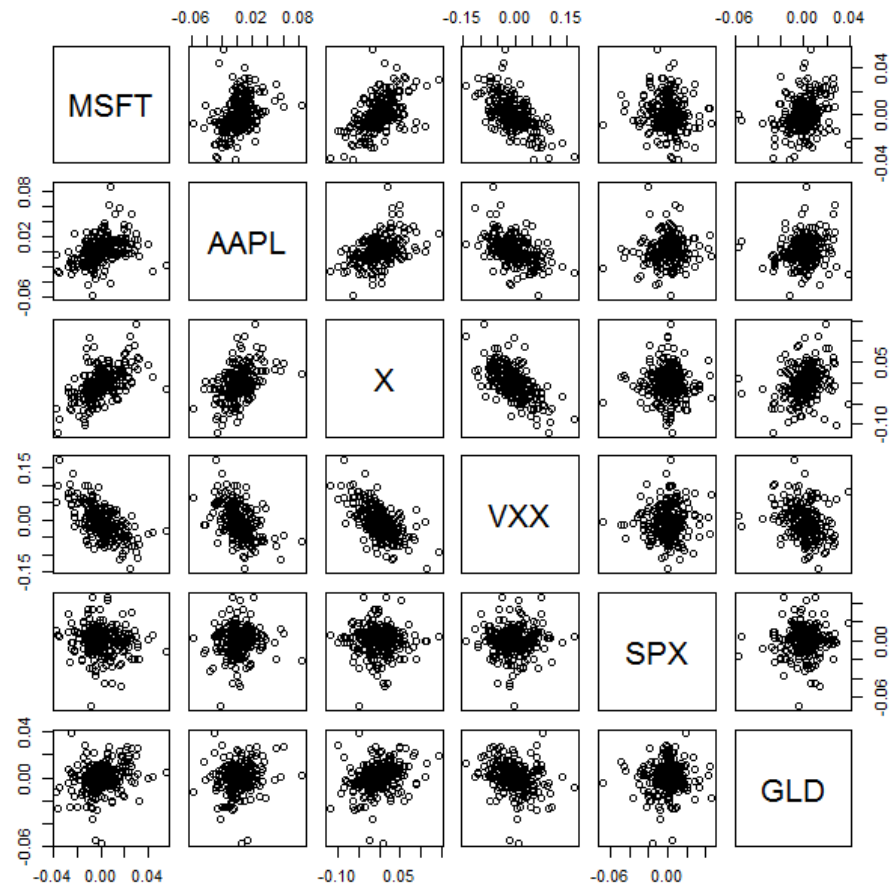
F# for Computation

- Standard F# tools
 - F# language and core libraries
- Workstation libraries
 - Math.NET Numerics (open source)
 - StatFactory FCore
- Distributed libraries
 - Microsoft Cloud Numerics
 - Nessos {m}brace cloud

Monte Carlo estimation of Pi



R - Type Provider



ShowDialog - Async workflows

```
type System.Windows.Window with
  member window.ShowDialog() =
    async {
      // Show the window
      window.Show()
      // Disable all other windows
      allWindows() |> Seq.filter ((<>) window) |> Seq.iter disableWindow
      // Await window closing
      do! window.Closing |> Async.AwaitEvent |> Async.Ignore
      // Enable all other windows
      allWindows() |> Seq.filter ((<>) window) |> Seq.iter enableWindow
    } |> Async.StartImmediate
```

Rx

notifications.

```
BufferWithTimeOrCount(  
    TimeSpan.FromSeconds(0.01),  
    50);
```

MiniRx

```
int closure = i;
mouseMove
    .Select(e => e.GetPosition(canvas))
    .Delay(closure * 100)
    .OnDispatcher()
    .Subscribe(pos =>
    {
        Canvas.SetLeft(label, pos.X + closure * 10);
        Canvas.SetTop(label, pos.Y);
    });
```

F# Agents - MailboxProcessor type

```
Agent.Start (fun inbox -> async {  
    while true do  
        let! instruction =  
  
instruction.Receive()  
    instruction  
    |> market.Update  
    |> notify } )
```

The future - `{m}brace` the cloud

- Programming model that looks sequential but executes distributed
 - `async { }`
 - `...`
 - `cloud { }`

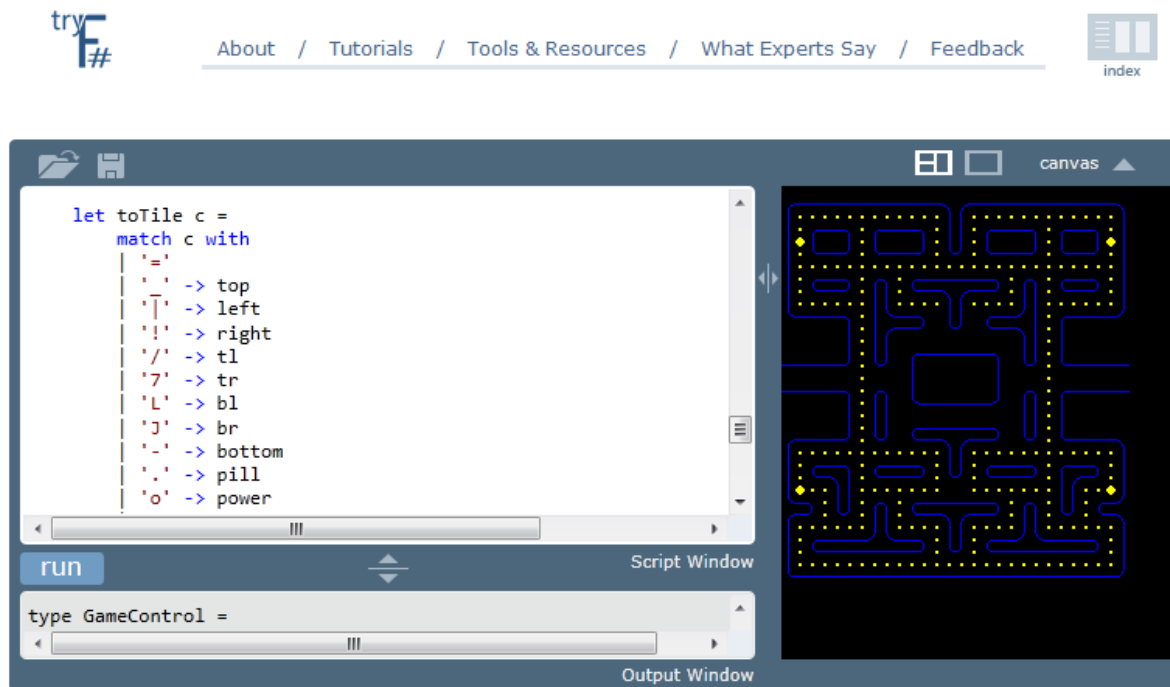
.Net 4.5 Garbage Collector

Latency Mode	Application Scenarios
Batch	Command line Server Side
Interactive	LOB applications
LowLatency	Trading Screens
SustainedLowLatency	Financial Applications

Meet the F#ers



Try F#: <http://tryfsharp.org>



Microsoft

Research

©2011 Microsoft Corporation. All rights reserved.

[Contact Us](#) | [About Microsoft Research](#) | [Privacy](#) | [Terms of](#)

Questions?

Me

Twitter: @ptrelford

Feed: <http://trelford.com/blog>

F#

Twitter: #fsharp

Feed: <http://fpound.net>