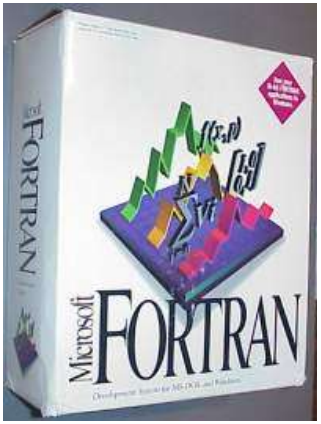
F# Eye for the C# guy

Leon Bambrick, secretGeek.net

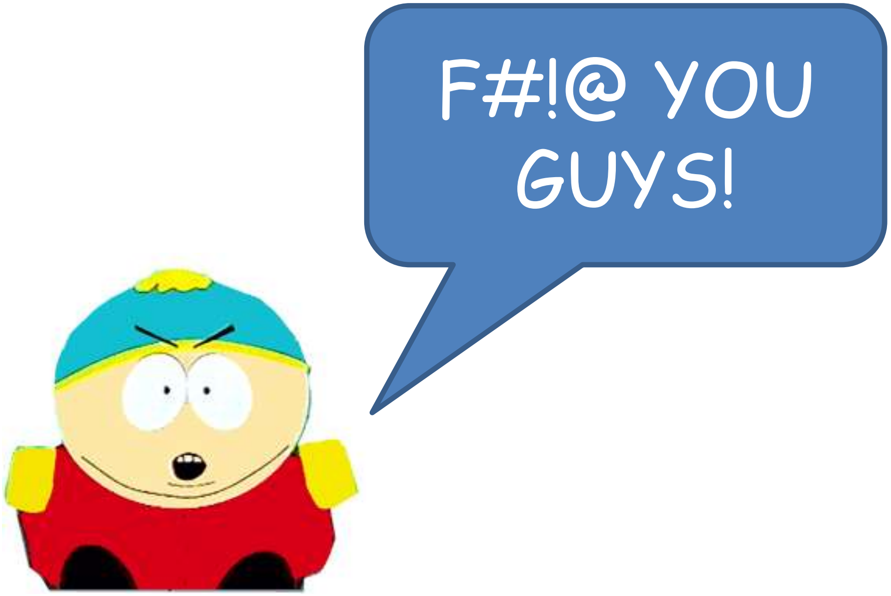
WTF$^{\#}$?



F#

…it's

Fortran.net


F#!@ YOU GUYS!

F#

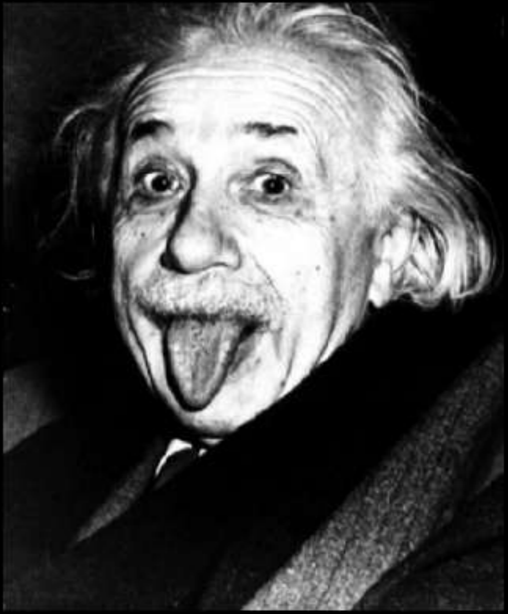
An Academic

Language

Reserved

For

Scienticians?



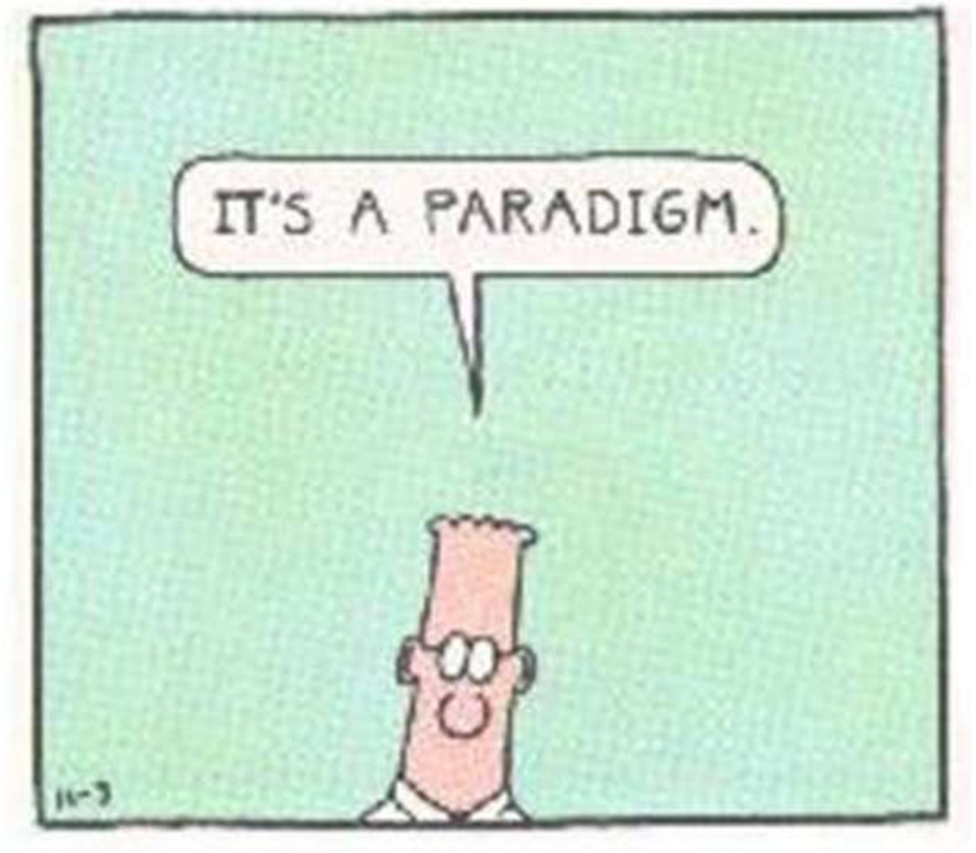
None of that.

Rather:

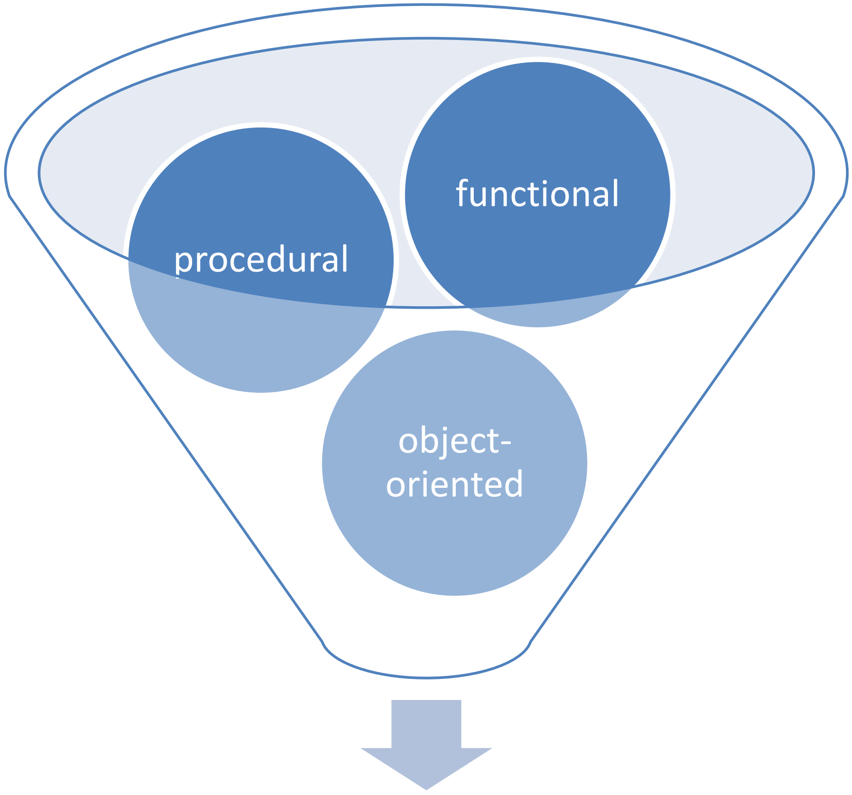
- General Purpose Language
- Ideal for Real World Development

Friendly. Approachable.

A Multi-Paradigm Language!



a what?



F#

30 second review:

3

Big

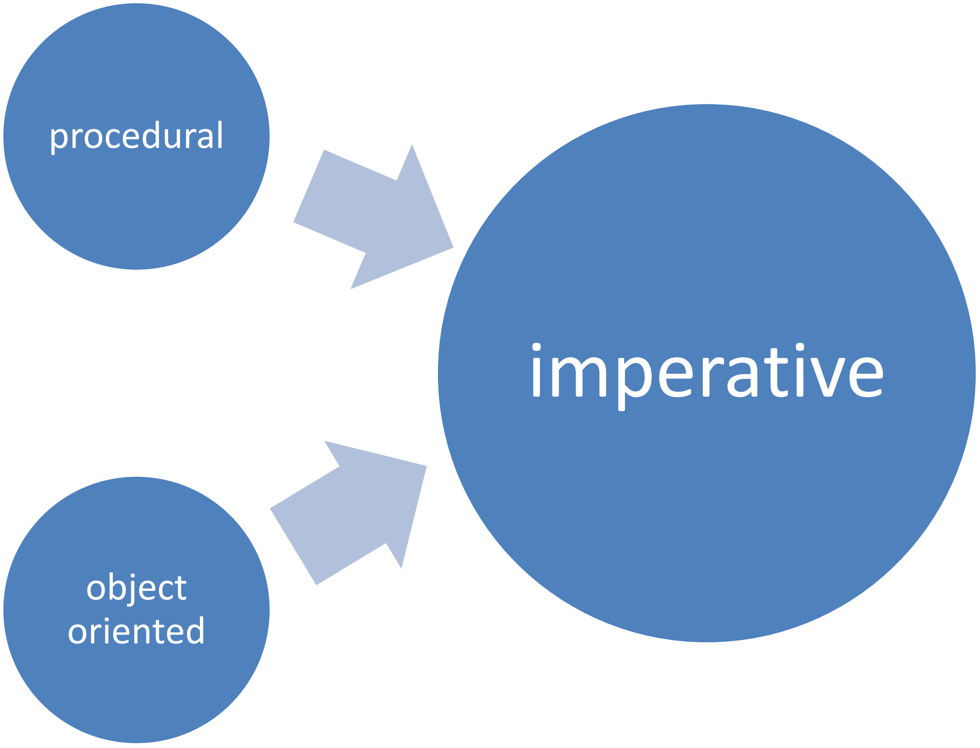
Paradigms

procedural

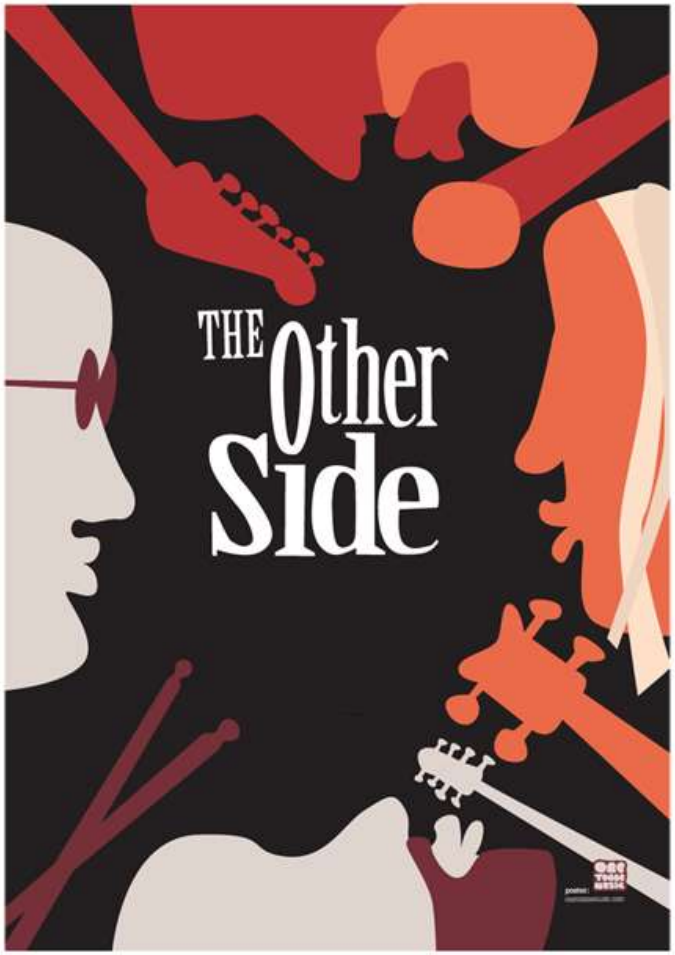
- Do this, then that, then that
- Useful abstraction over machine code
- Assembly language, Fortran, C, Pascal

object oriented

- Useful abstraction over procedural
- Define types, methods, members
- Inheritance, polymorphism, overloading
- C++, VB.net, C#, J#


procedural
object oriented
imperative

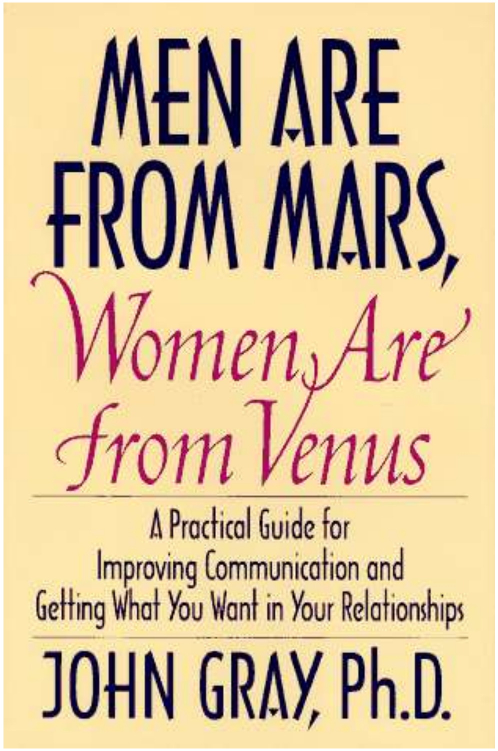
Functional = The Other Side

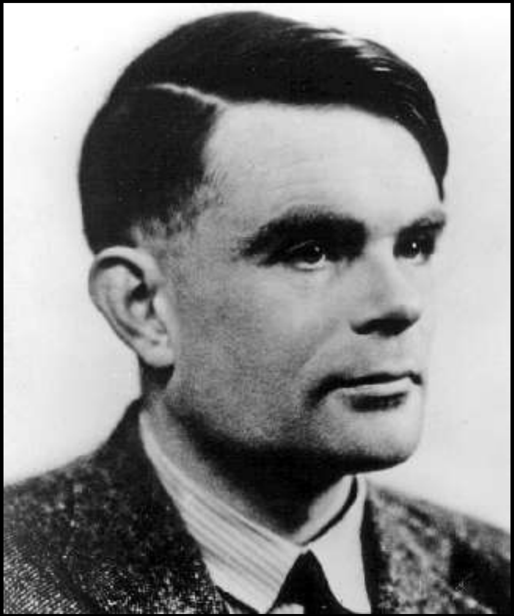


functional

No common ancestor with C

No matter how far back you go!


MEN ARE FROM MARS, Women Are from Venus
A Practical Guide for Improving Communication and Getting What You Want in Your Relationships
JOHN GRAY, Ph.D.



V

Alan Turing v Alonzo Church
Cage Match of Death

functional

- Focus on results not process
- Decompose problem into ‘functions’
- Lisp, Scheme, Haskell, ML, Erlang

functional?

- Visual Basic has functions...

functional?

- Visual Basic has functions...

does that make it 'functional' ?

functional?

FUNCTION $\neq$ "method that returns a value"

functional?

FUNCTION $\neq$ "method that returns a value"

Think:

"mathematical function“

"formula"

"equation "

Purely functional...

Avoid Side-Effects!

Purely functional…

Avoid Mutation!

Purely functional…

No Variables!

Only Functions!

Purely functional...

Same input -> Same output!

Purely functional…

No Shared State

Why bother?

- Pure functions can be executed in parallel without interfering with one another

Why bother?

- Pure functions can be executed in parallel without interfering with one another
- Pure functions can be “perfectly” cached

Why bother?

- Pure functions can be executed in parallel without interfering with one another
- Pure functions can be “perfectly” cached
- Pure functions can be “partially” applied

Why bother?

- Pure functions can be executed in parallel without interfering with one another
- Pure functions can be "perfectly" cached
- Pure functions can be "partially" applied
- Functions can receive and return functions, for which all of the above hold true

Why bother?

- Pure functions can be executed in parallel without interfering with one another
- Pure functions can be “perfectly” cached
- Pure functions can be “partially” applied
- Pure functions can return functions, for which all of the above still hold true
- Allows for greater “modularity”

What’s the catch?

- “Hello world” is a side effect
- Custom runtimes a-plenty

What’s the catch?

- “Hello world” is a side effect
- Custom runtimes a-plenty
- Smug Lisp weenies

Functional is the new OO

Some stuff is now cheap!

Functional is the new OO

Some stuff is now cheap!

- Ram
- Disk
- Cores

Functional is the new OO

- Some stuff is now cheap!
  - Ram
  - Disk
  - Cores
- Some stuff remains expensive!

Functional is the new OO

- Some stuff is now cheap!
  - Ram
  - Disk
  - Cores
- Some stuff remains expensive!
  - Real Time
  - Concurrency
  - Locking

This tips the balance toward higher abstractions

Genealogy of F$^{\#}$ …

- Theorem proving and ISWIM

Genealogy of F$^{\#}$ …

- Theorem proving and ISWIM begat:
  - ML “Meta Language”

Genealogy of F$^{\#}$ …

- Theorem proving and ISWIM begat:
  - ML "Meta Language", which begat:
    - CAML

Genealogy of F$^{\#}$ …

- Theorem proving and ISWIM begat:
  - ML “Meta Language”, which begat:
    - CAML, which in turn begat
      - OCaml

Oh!

Genealogy of $F^{\#}$ …

- Theorem proving and ISWIM begat:
  - ML “Meta Language”, which begat:
    - CAML, which in turn begat
      - OCaml, which in turn begat
        - F#

… a sort of OCaml.net (and more)

WTF$^{\#}$?

- First official functional language on .net
- Deep support thanks to Generics

WTF$^{\#}$?

- First official functional language on .net
- Deep support thanks to Generics
- Recently assimilated by dev-div

Code!

```
//F#
let a = 2
```

Code!

```
//F#
let a = 2
```

≠

```
//C#
int a = 2
```

Code!

```
//F#
let a = 2
```

More like

```
//C#
//a function!
static int a()
{
 return 2;
}
```

More Code!

```
//F#
#light
open System
let a = 2
Console.WriteLine a
```

```
//C#
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static int a()
        {
            return 2;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(a);
        }
    }
}
```

More Code!

```
//F#
#light
open System
let a = 2
Console.WriteLine a
```

```
//C#
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static int a()
        {
            return 2;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(a);
        }
    }
}
```

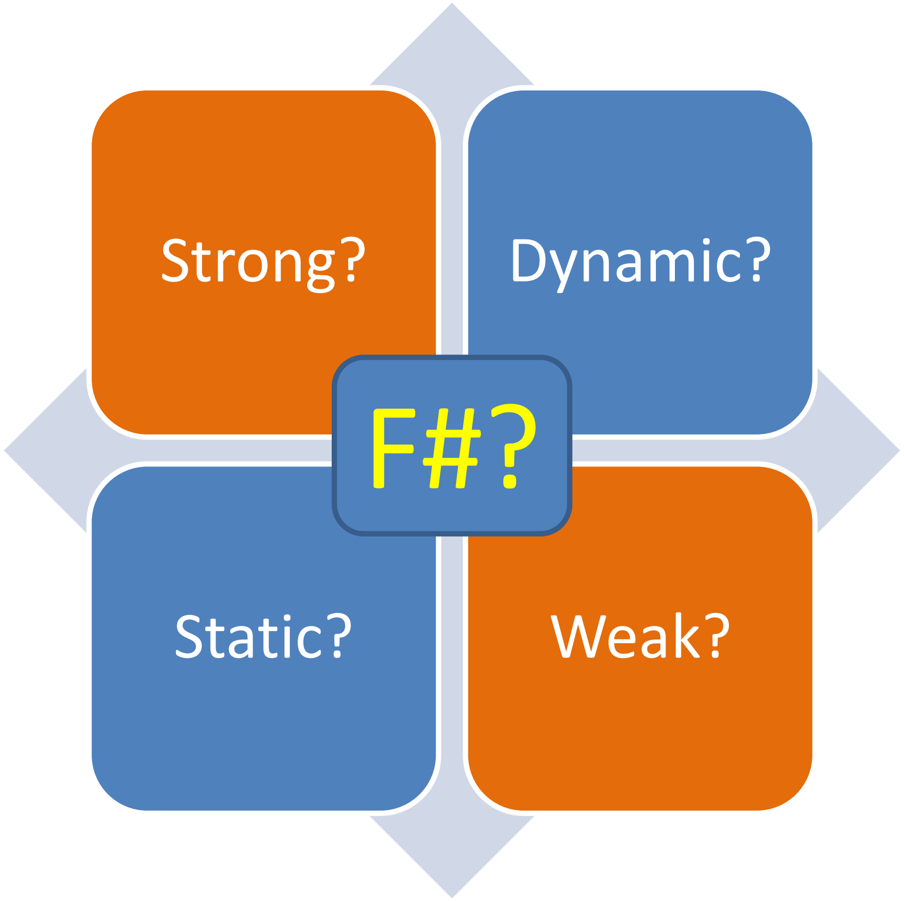
More Noise Than Signal!

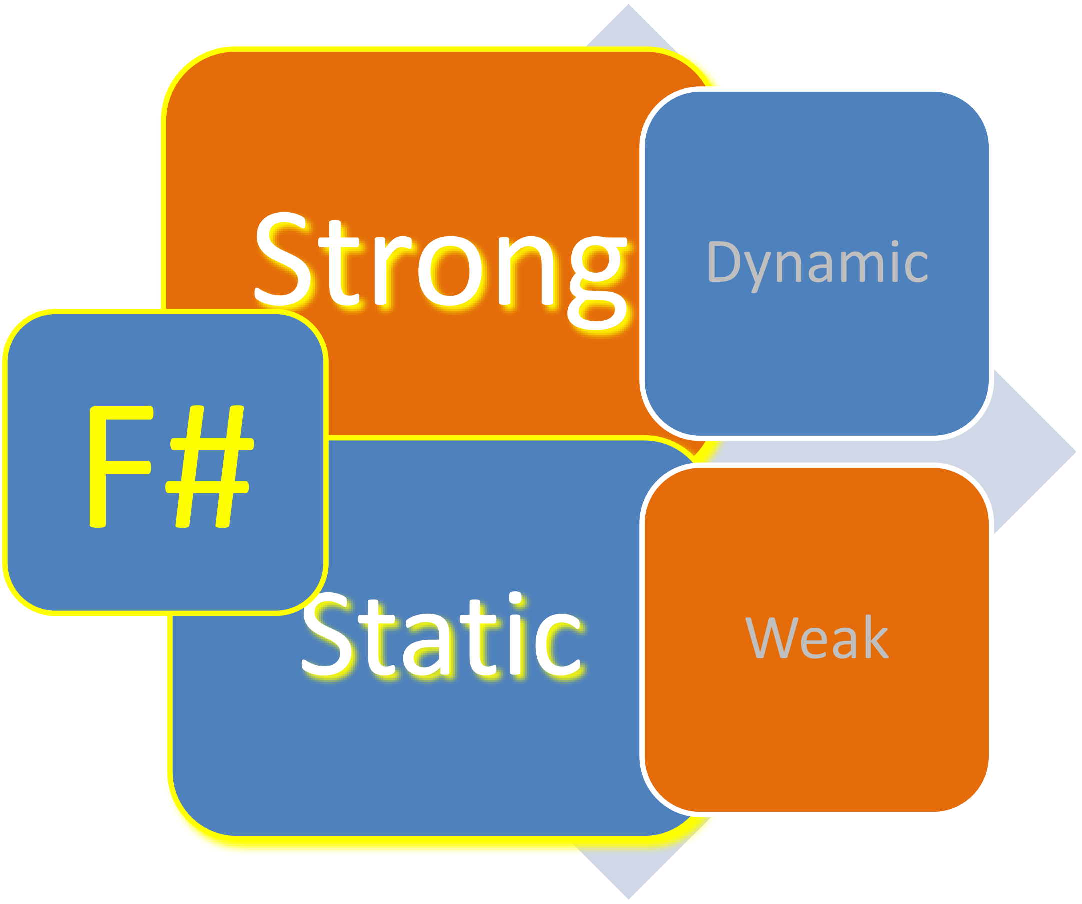
More Code!

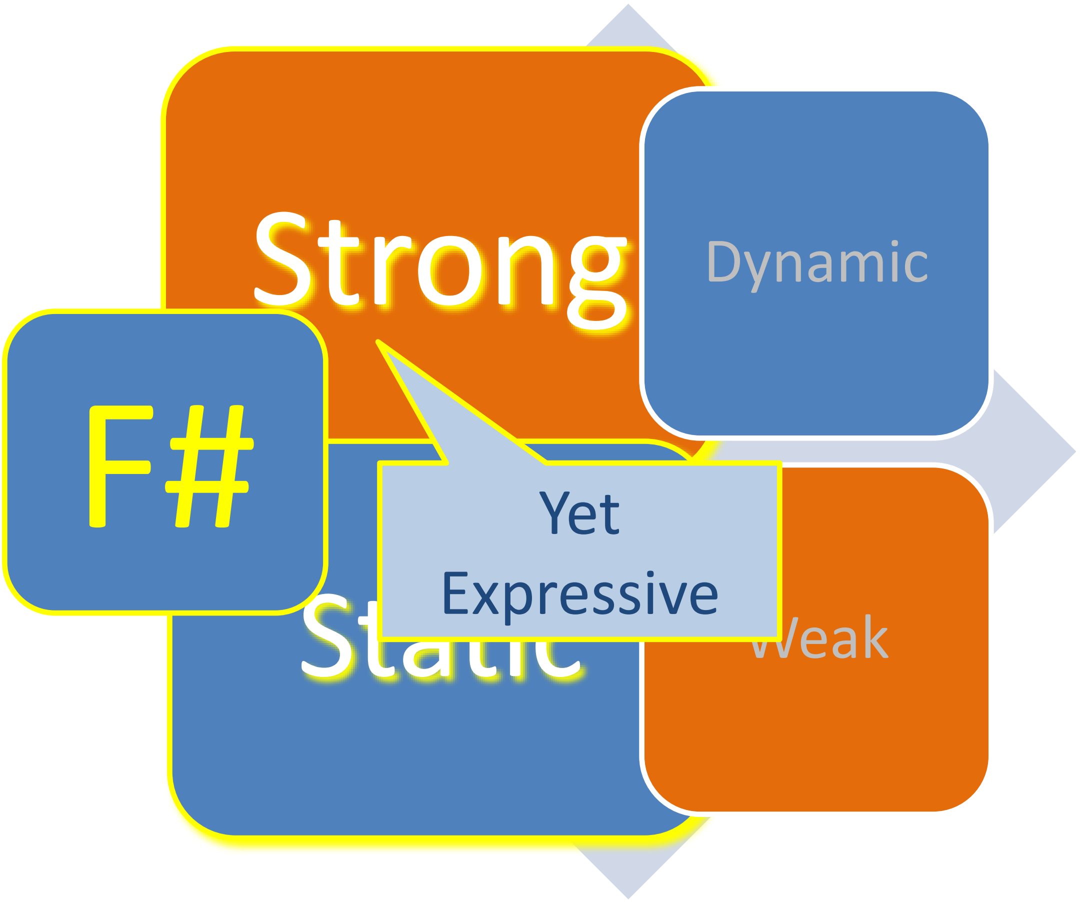
```fsharp
//F#
#light
open System
let a = 2
Console.WriteLine a
```

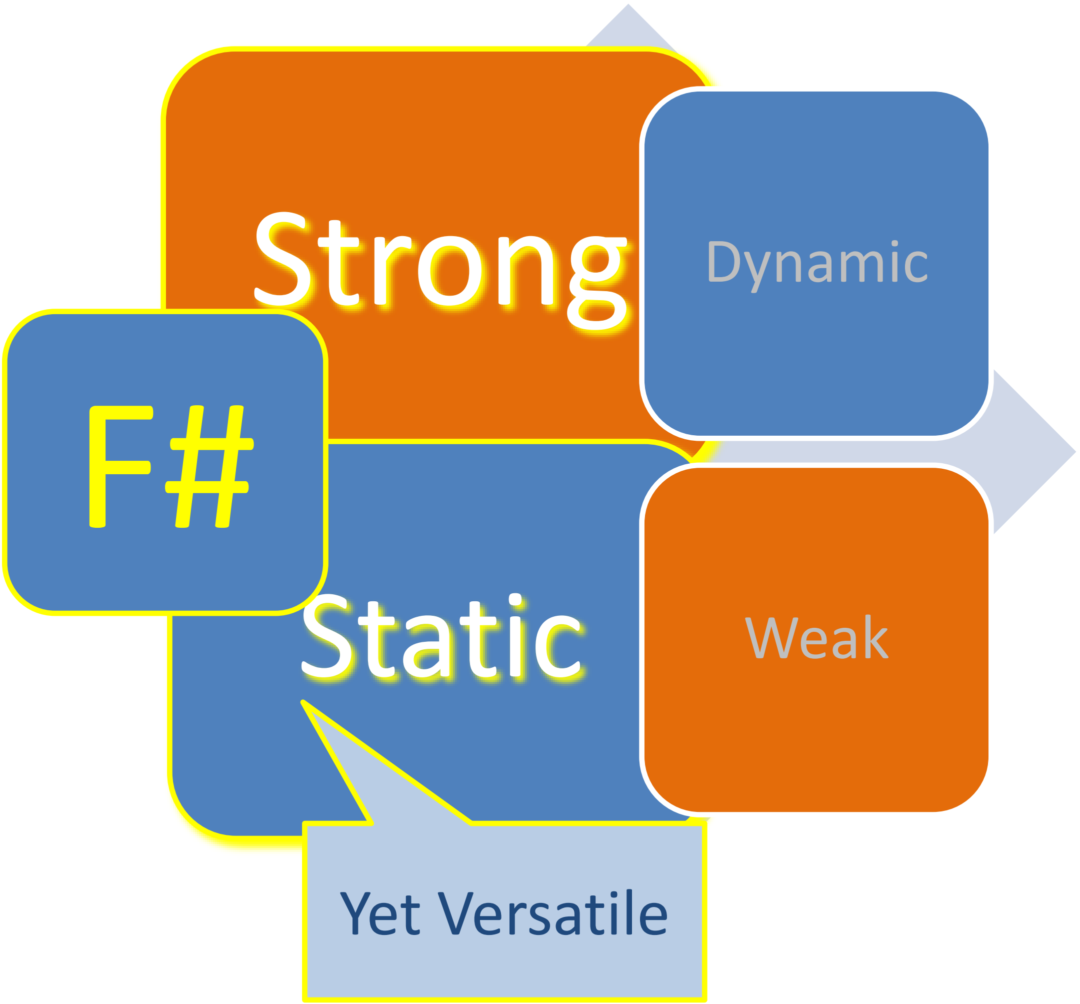
Looks Weakly typed?
Maybe Dynamic?

```csharp
//C#
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static int a()
        {
            return 2;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(a);
        }
```


Strong?
Dynamic?
F#?
Static?
Weak?


Strong
Dynamic
F#
Static
Weak


Strong
Dynamic
F#
Yet Expressive
Static
Weak


Strong
Dynamic
F#
Static
Weak
Yet Versatile

More Code!

```
//F#
#light
open System
let a = 2
Console.WriteLine a
```

Type Inference

```
//C#
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static int a()
        {
            return 2;
        }

        static void Main(string[] args)
        {
            Console.WriteLine(a);
        }
    }
}
```

Immutable by default

```
let a = 2
let a = 3
```

```
error: FS0037: Duplicate definition of value 'a'
```

simple function…

```
let square x = x * x

> val square : int -> int

square 5

> val it : int = 25
```

simple function...

```
let square x = x * x

> val squar

square 5

> val it : int = 25
```

Parameter

simple function…

```
let square x = x * x

> val square : int -> int

square 5

> val it
```

“Signature”

Discriminated union types

```
type NullableInt =
| Value of int
| Nothing of unit
```

Discriminated unions example

```
type Weapon =
| Knife
| Gun
| Bomb
```

Pattern Matching

```
type Weapon =
 | Knife
 | Gun
 | Bomb
```

```
//block any weapon!
let block w =
    match w with
     | Knife
     | Gun  -> disarm w
     | _ -> difuse w
```

Pattern Matching

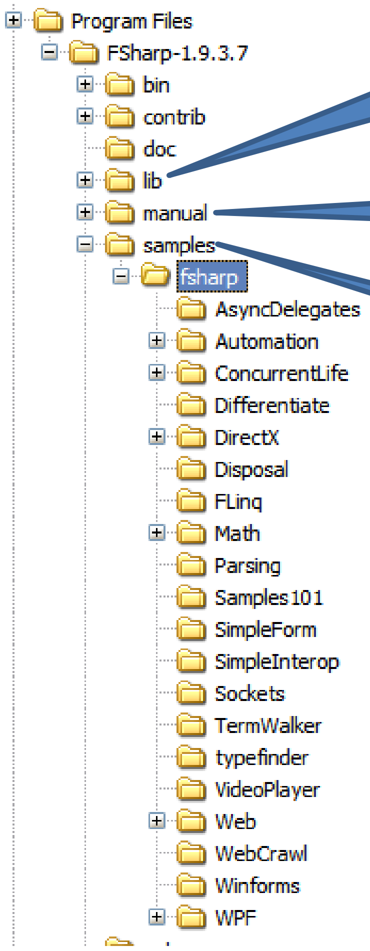
```
type Weapon =
 | Knife
 | Gun
 | Bomb
```

```
//block any weapon
let block w =
    match w with
     | Knife
     | Gun  -> disarm w
     | _ -> difuse w
```

```
block Gun
block Knife
block Bomb
```

Lazy is a virtue

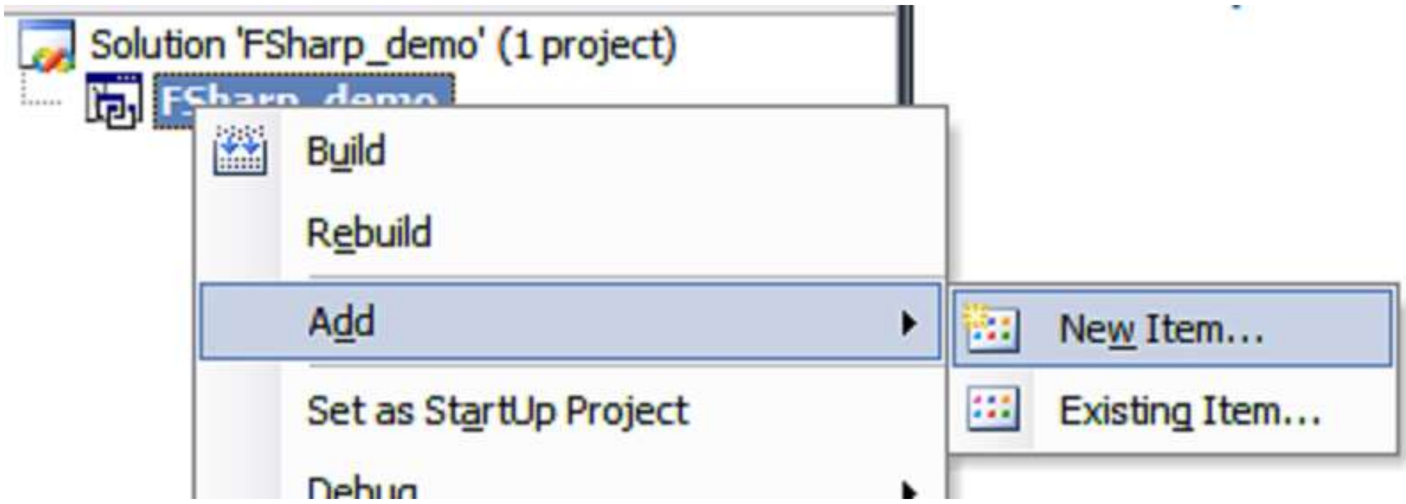
```
let lazy_square x =
    lazy ( print_endline “thinking...“
          x * x )

let lazy_square_ten = lazy_square 10

//first time: “thinking...”
Lazy.force (lazy_square_ten)

//second time: no thinking, just result
Lazy.force (lazy_square_ten)
```

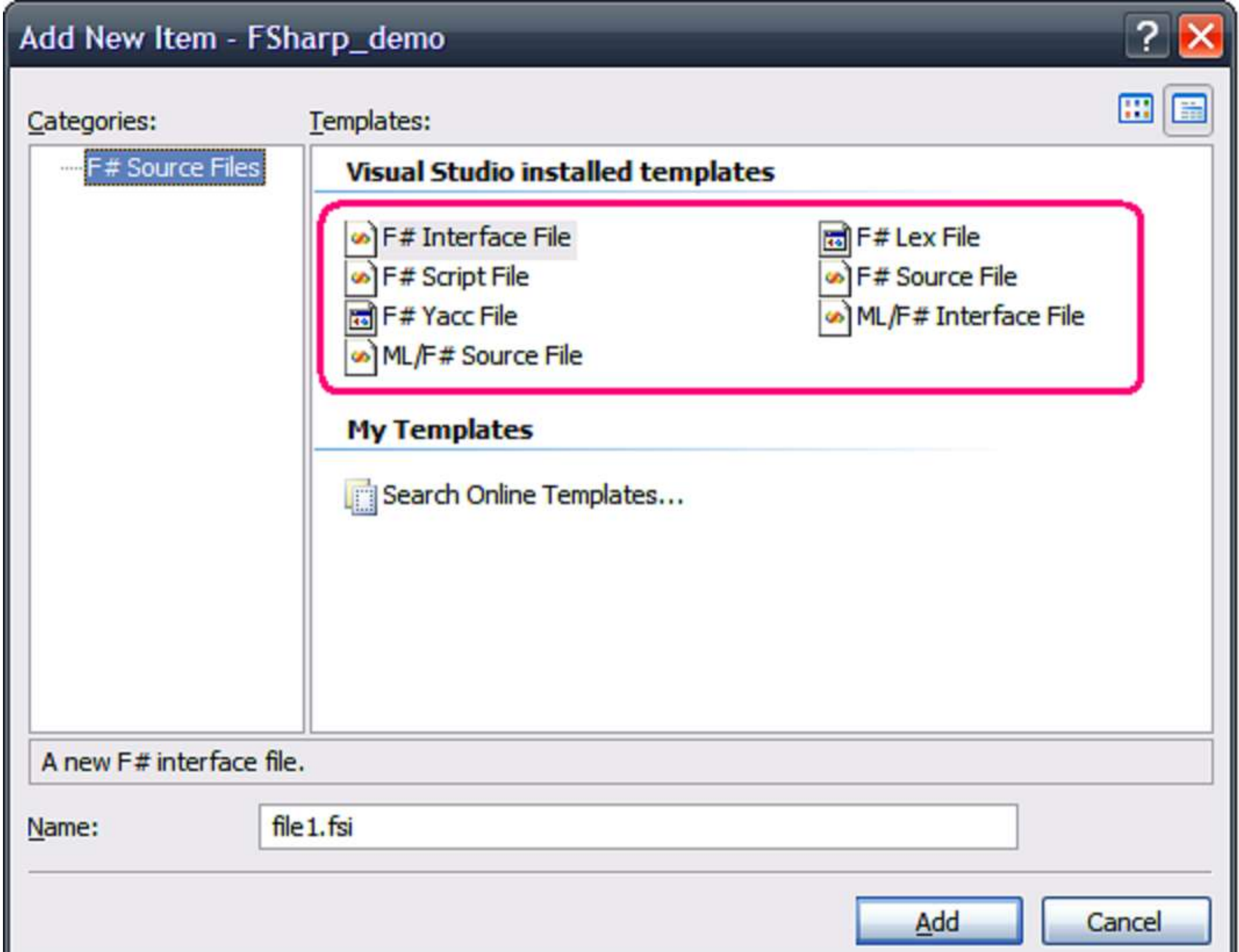

Program Files
FSharp-1.9.3.7
bin
contrib
doc
lib
manual
samples
fsharp
AsyncDelegates
Automation
ConcurrentLife
Differentiate
DirectX
Disposal
FLinq
Math
Parsing
Samples101
SimpleForm
SimpleInterop
Sockets
TermWalker
typefinder
VideoPlayer
Web
WebCrawl
Winforms
WPF

Useful libraries

Neat manual

Awesome Samples

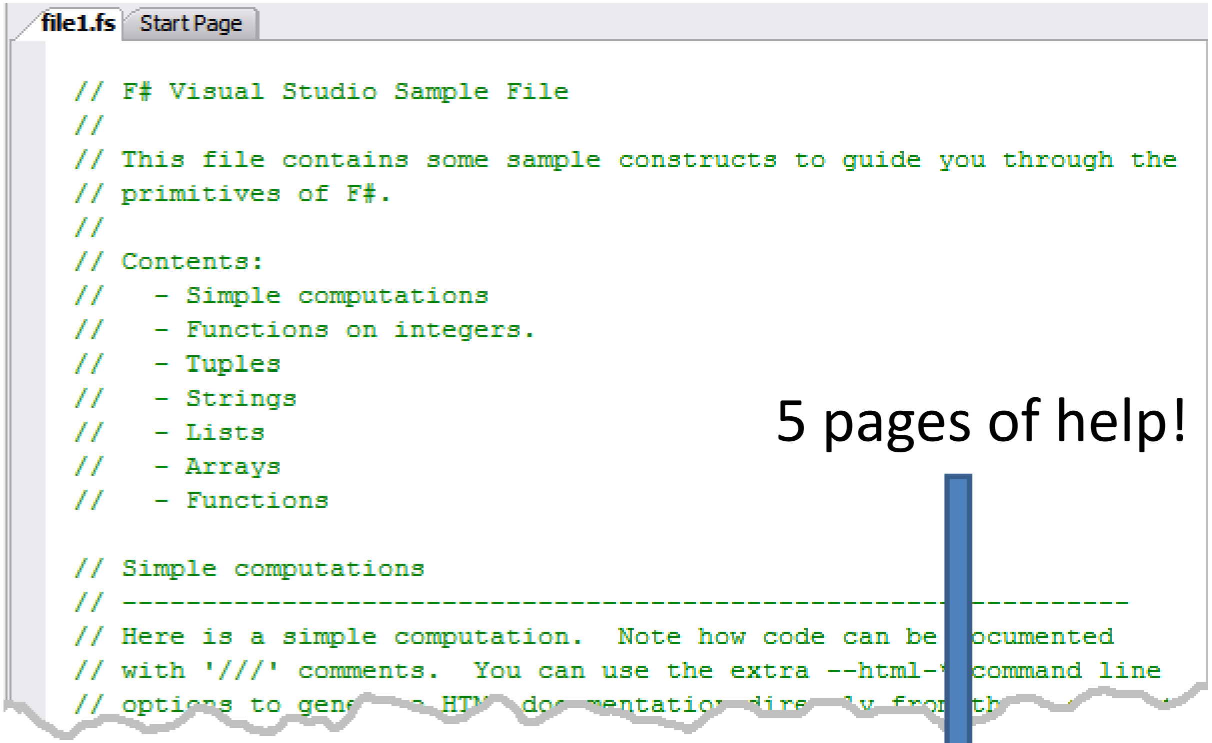

Solution 'FSharp_demo' (1 project)
FSharp_demo
Build
Rebuild
Add
Set as StartUp Project
Debug
New Item...
Existing Item...



A new F# interface file.

Name: file1.fsi

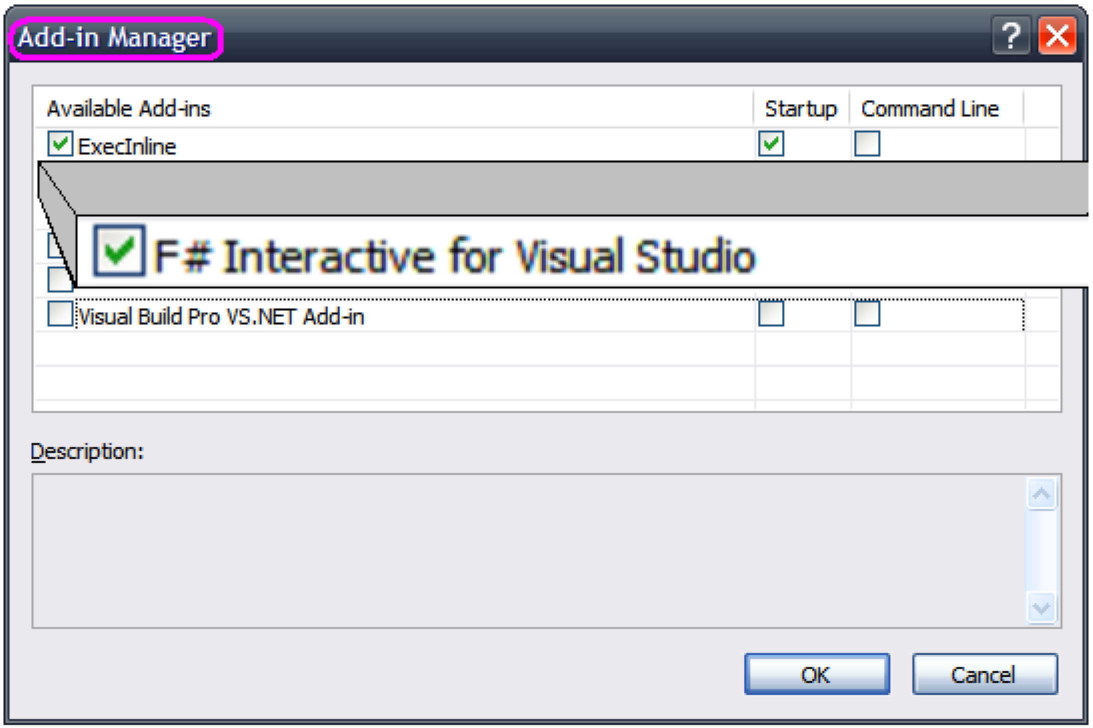
“Empty” source file...



```
// F# Visual Studio Sample File
//
// This file contains some sample constructs to guide you through the
// primitives of F#.
//
// Contents:
//   - Simple computations
//   - Functions on integers.
//   - Tuples
//   - Strings
//   - Lists
//   - Arrays
//   - Functions

// Simple computations
// --------------------------------------------------------------
// Here is a simple computation.  Note how code can be ocumented
// with '///' comments.  You can use the extra --html-  command line
// options to gene      HTM  documentation dire    y from th
```

5 pages of help!

Make sure F# Interactive is running!



Add-in Manager

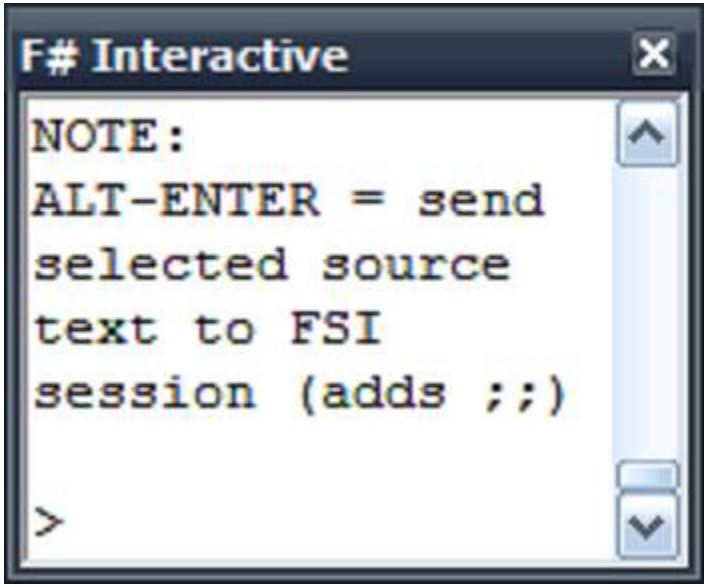
Available Add-ins	Startup	Command Line
✓ ExecInline	✓	☐
✓ F# Interactive for Visual Studio		
☐ Visual Build Pro VS.NET Add-in	☐	☐

Description:

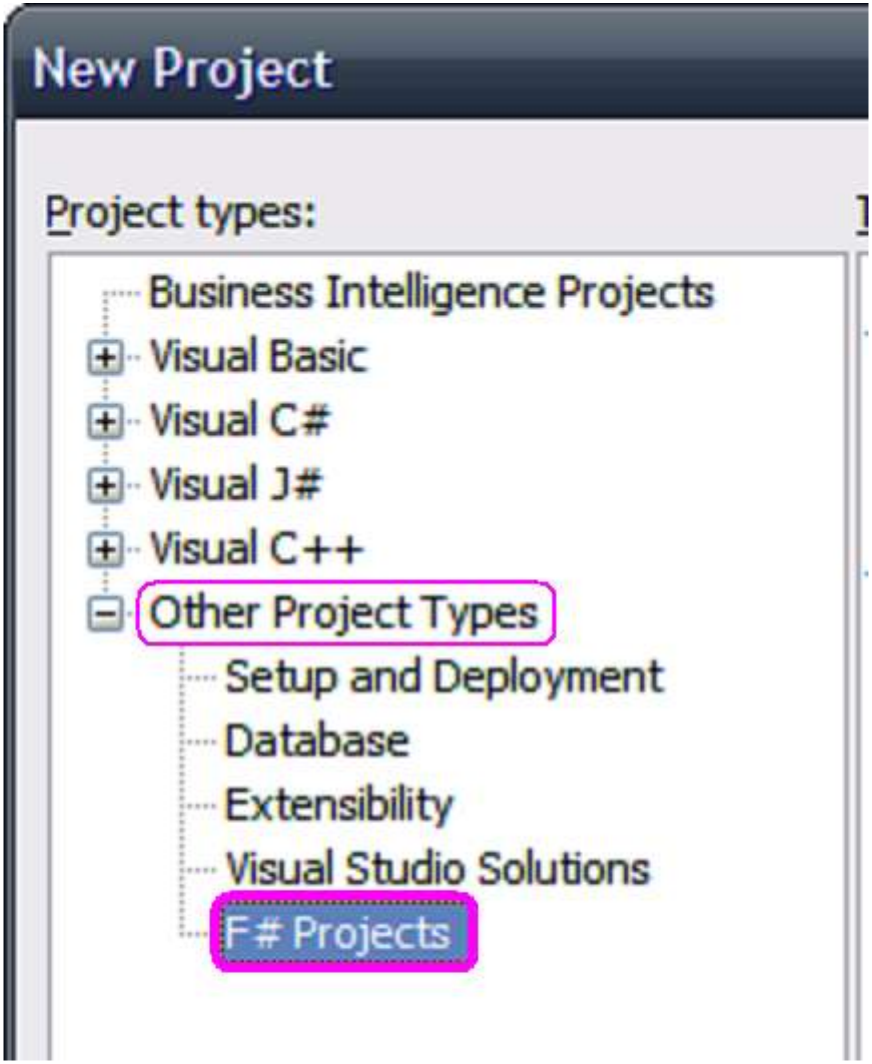
F# Interactive:

It’s the bomb!

F# Interactive:



```
NOTE:
ALT-ENTER = send
selected source
text to FSI
session (adds ;;)


>
```


New Project
Project types:
Business Intelligence Projects
Visual Basic
Visual C#
Visual J#
Visual C++
Other Project Types
Setup and Deployment
Database
Extensibility
Visual Studio Solutions
F# Projects

Why learn F#?

Why learn F#?

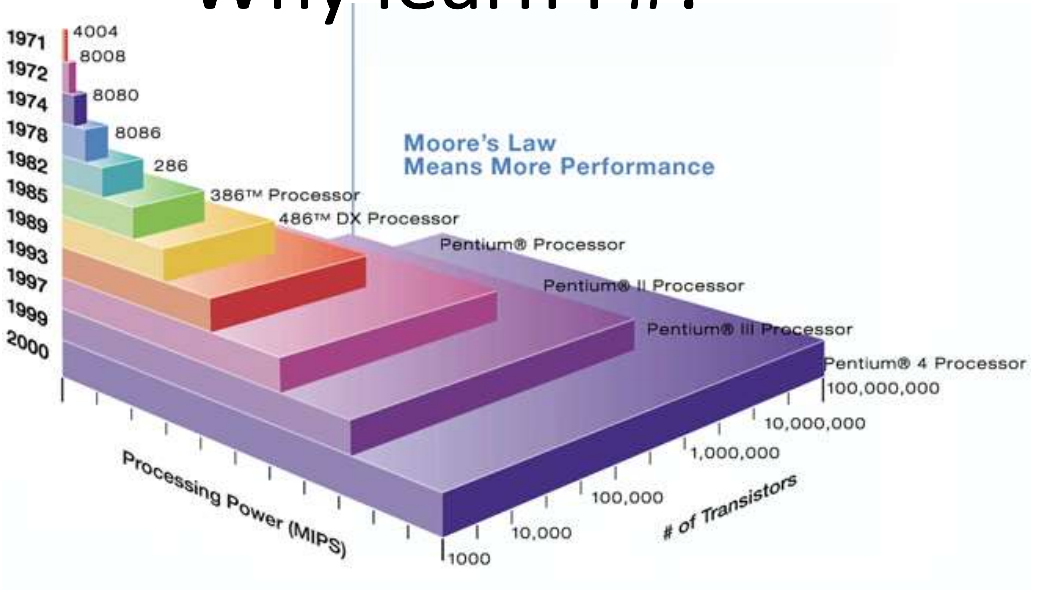
- See where C# and VB.net are headed

Why learn F#?

- See where C# and VB.net are headed

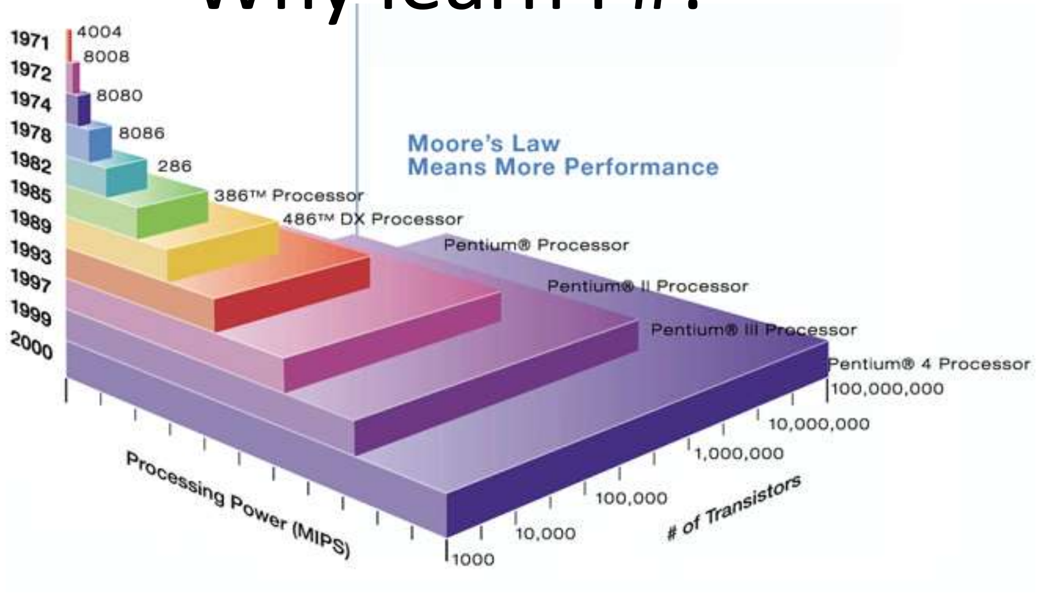
- Learn one new language per year

Why learn F#?



Moore's Law Ran Out!

Why learn F#?



Moore's Law Ran Out!

(again, maybe)

- Data Grows Quickly
- But # of Dimensions Grows much faster!
- And semi-structured data outgrowing structured
- Entropy Increasing
- Complexity is through the roof!

Hence: Again with the donkey

"Software gets slower faster than hardware gets faster"

--Wirth's Law

~~Lisp~~ **F#** is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use ~~Lisp~~ **F#** itself a lot."

- ~~Eric Raymond~~ (lb)

Some Applications of F#

- Map/Reduce over internets
- Financial Analysis
- In process SQL Data Mining
- XNA Games Development
- Web tools, Compile F# to Javascript

Game Programming

Game Programming

- 3D Animation
- Rendering
- Shading
- Simulation (e.g. physics)
- Collision Detection
- AI Opponents

8 Ways to Learn

- FSI.exe
- Samples Included
- Go to definition
  - See the source!
- Lutz' Reflector
- http://cs.hubfs.net
- Codeplex Fsharp Samples
- Books
- ML

Acknowledgements

- Cartman
- Einstein
- Dilbert
- Alan Turing
- Alonzo Church
- Godzilla
- Gears of war

- John Hughes, *Why Functional Programming Matters,* http://www.math.chalmers.se/~rjmh/Papers/whyfp.html
- Robert Pickering, *Foundations of F#,* http://www.apress.com/book/view/1590597575
- Slava Akhmechet, *Functional Programming For The Rest of Us,* http://www.defmacro.org/ramblings/fp.html
- Steve Yegge, *Execution In the Kingdom of Nouns,* http://steve-yegge.blogspot.com/2006/03/execution-in-kingdom-of-nouns.html
- P. J. Landin, *The Next 700 Programming Languages* http://www.cs.utah.edu/~wilson/compilers/old/papers/p157-landin.pdf
- Tim Sweeney, *The Next Mainstream Programming Language*, http://www.st.cs.uni-sb.de/edu/seminare/2005/advanced-fp/docs/sweeny.pdf
- Tomas Petricek, *F# Web Tools, Ajax Made Simple*, http://www.codeplex.com/fswebtools
- Herb Sutter, *The Free Lunch Is Over - A Fundamental Turn Toward Concurrency in Software*, http://www.gotw.ca/publications/concurrency-ddj.htm
- **Don Syme, http://blogs.msdn.com/dsyme**