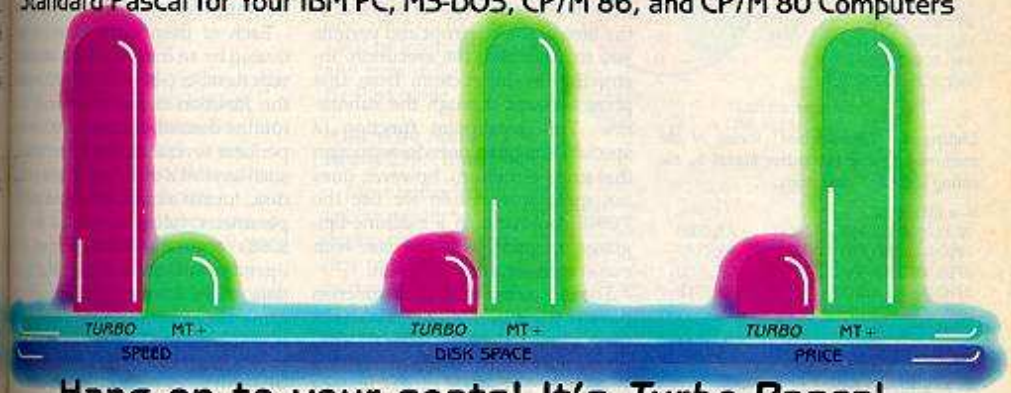
Where are Programming Languages Going?

Anders Hejlsberg
Microsoft Technical Fellow
andersh@microsoft.com

PASCAL $49.95

Offer

Standard Pascal for Your IBM PC, MS-DOS, CP/M 86, and CP/M 80 Computers



Hang on to your seats! It's *Turbo* Pascal.

	Turbo Pascal	JRT	MT+
8 & 16 bit	YES	NO	YES
Editor	YES	NO	NO
Generate Direct Code	YES	NO	YES
Execute Run Time Library Directly from Source Code	YES	NO	NO
Compilation Speed ¹	1 s.	46 s.	69 s. includes linking
Execution Speed ¹	6 s.	69 s.	8 s.
Space	28K including editor	85K +editor	168K +editor
Price	$49.95	$39.95	$595.00

¹ Benchmark data based on EightQueens in "Algorithms + Data Structures = Programs" by N. Wirth (Prentice-Hall, publisher)
Turbo Pascal is a trademark of Borland International. MT+ is a trademark of MT Microsystems. JRT Pascal is a product of JRT. CP/M is a trademark of Microproc.

Dealer and Distributor inquiries welcome.

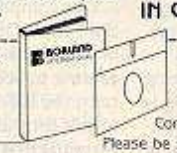
There has never been a Pascal compiler this good with so many powerful features. We know what you've been waiting for: a true Pascal compiler that works fast, offers a full screen editor, and has a great price.

Turbo Pascal has it all. First, we've included a built-in, interactive full screen, Wordstar compatible editor; it not only lets you correct errors, but during program compilation the cursor even jumps directly to the error and waits for your correction. No kidding. Second, it takes only 28K of disk space, including the editor; and on your microcomputer you need all the space you can get. Turbo Pascal is 10 to 70 times faster during compilation, as well as execution than Digital Research's MT+ or JRT Pascal.

Hard to believe your good fortune on the price? Don't worry. We're Borland, and we produce only quality, state-of-the-art software. Companies such as Micro Pro, Morrow Computers, Access and others distribute our software products, so you can't go wrong.

Place your order today. And we'll ship your *Turbo Pascal* out fast. For VISA and MasterCard orders call toll free:

1-800-227-2400 X 968
IN CA: 1-800-772-2666 X 968



***Turbo Pascal* $49.95 + $2.00 shipping per copy.**

Check ___ Money Order ___
VISA ___ MasterCard ___
Card #: _____
Exp date: _____ Shipped UPS.

My system is: 8 bit ____ 16 bit ____
Operating system: CP/M 80 ____
CP/M 86 ____ MS DOS ____ PC DOS ____
Computer: _____ Disk Format: _____
Please be sure model number and format are correct.

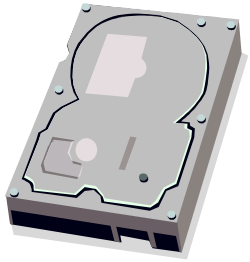
NAME: _____
ADDRESS: _____
CITY/STATE/ZIP: _____
TELEPHONE: _____

California residents add 6½% sales tax. Outside North America add $15.00 for airmail, or $5.00 for surface mail. Checks must be on a U.S. bank.

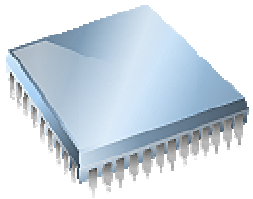
BORLAND INTERNATIONAL

Borland International
4807 Scotts Valley Drive
Scotts Valley, California 95066

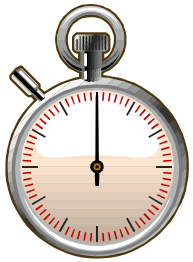
Since then…



x 100,000



x 10,000



x 1000

But…

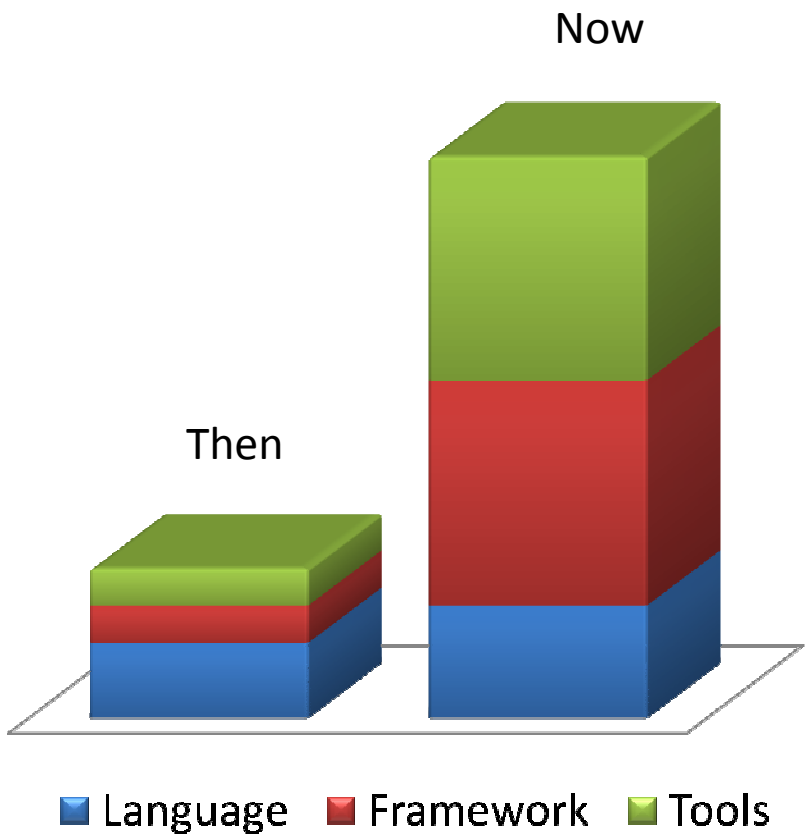
```
program HelloWorld;
var
  I: Integer;
begin
  for I := 1 to 10 do
    WriteLn('Hello World');
end.
```

x ?

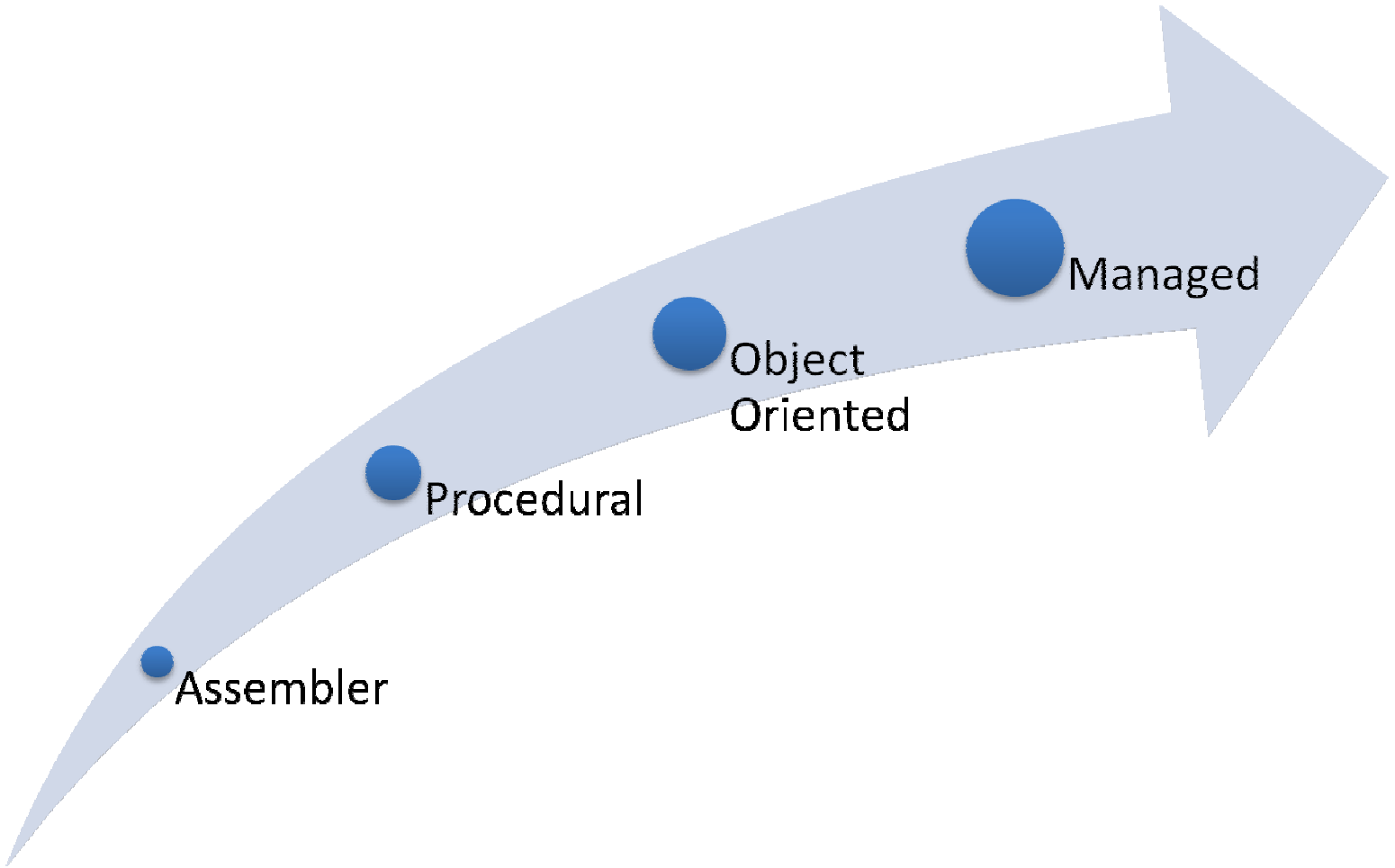
=

```
using System;

class HelloWorld
{
  static void Main(string[] args) {
    for (int i = 0; i < 10; i++)
      Console.WriteLine("Hello
World");
  }
}
```

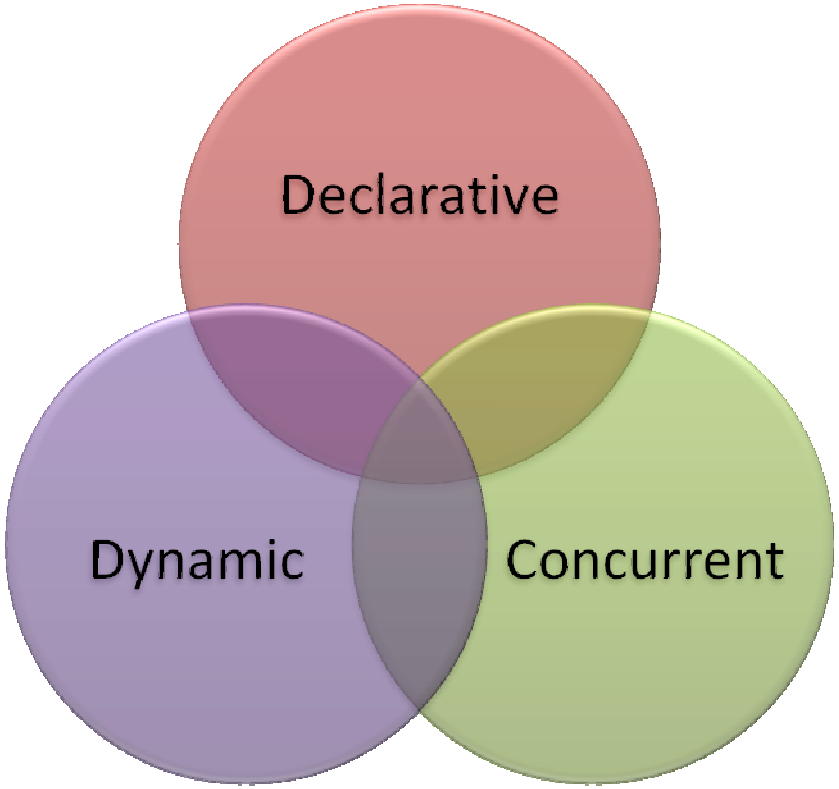
Languages, Frameworks, Tools



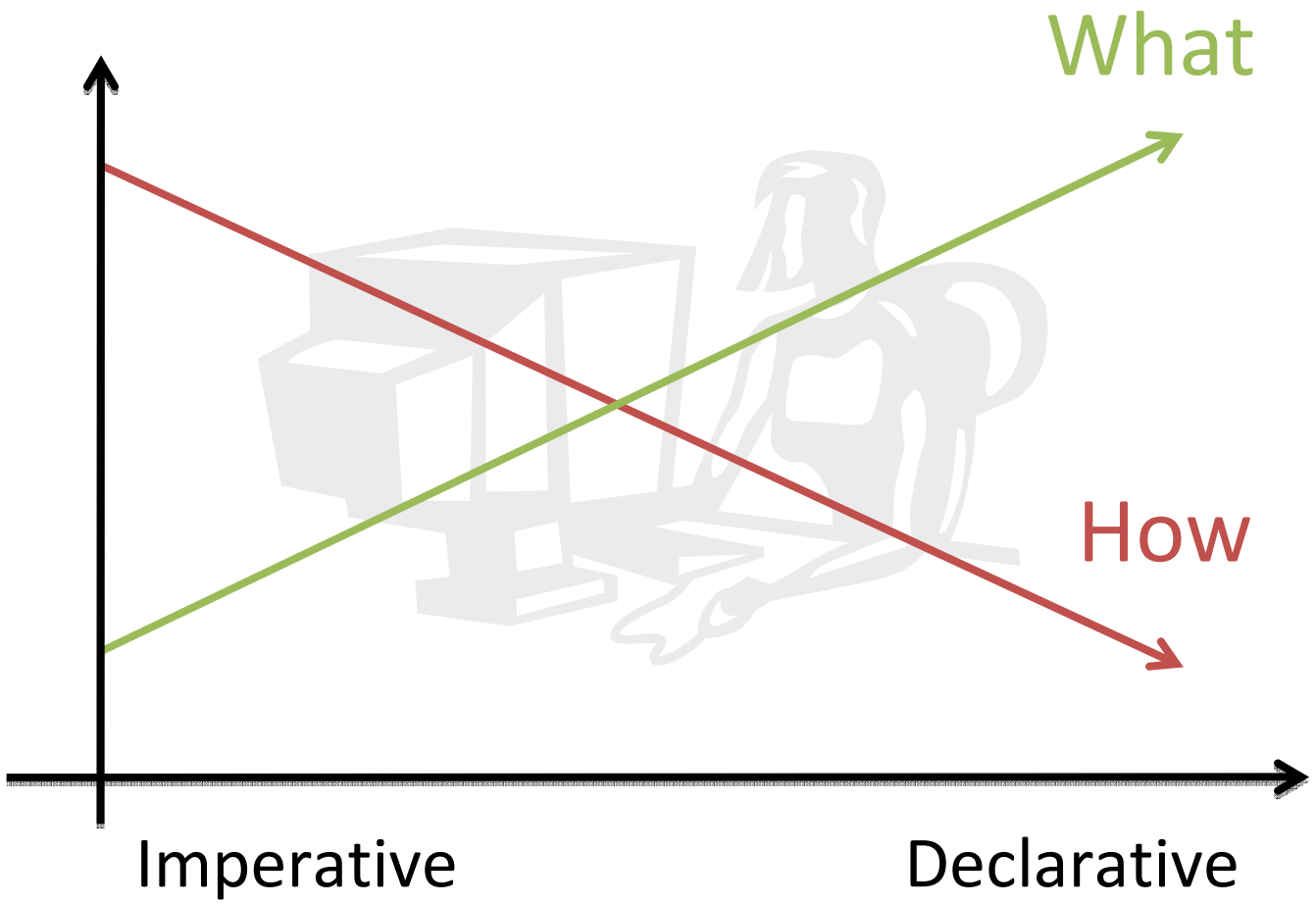
Increasing Abstraction Level



Trends



Declarative Programming

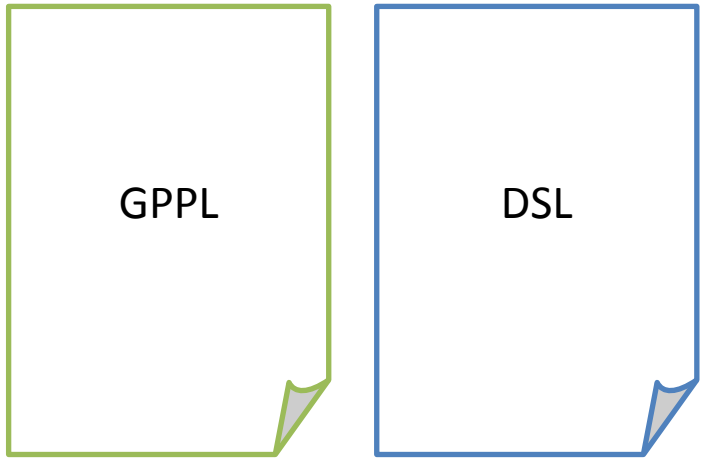


Domain Specific Languages

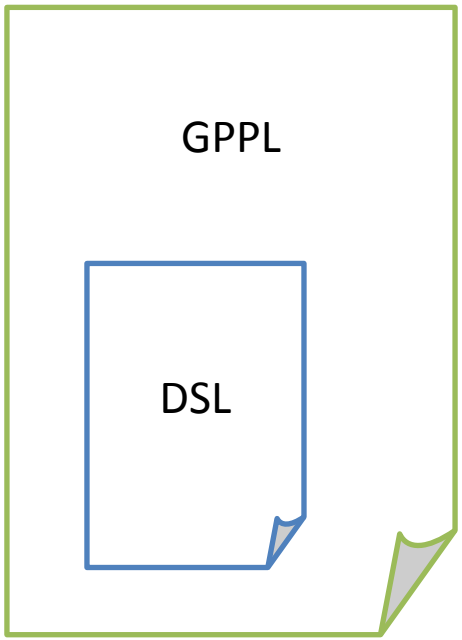
"… a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and/or a particular solution technique."

Source: Wikipedia

Domain Specific Languages



External DSL



Internal DSL

External DSLs

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" indent="yes"/>
   <xsl:template match="/persons">
      <root><xsl:apply-templates select="person"/></root>
   </xsl:template>
   <xsl:template match="person">
      <name id="{@username}"><xsl:value-of select="name"/></name>
   </xsl:template>
</xsl:stylesheet>
```

```
SELECT Name, Address, City
FROM Customers
WHERE Country = "Denmark"
```

```
rm -f /tmp/listing.tmp > /dev/null 2>&1
touch /tmp/listing.tmp
ls -l [a-z]*.doc | sort > /tmp/listing.tmp
lpr -Ppostscript_1 /tmp/listing.tmp
rm -f /tmp/listing.tmp
```

Internal DSLs

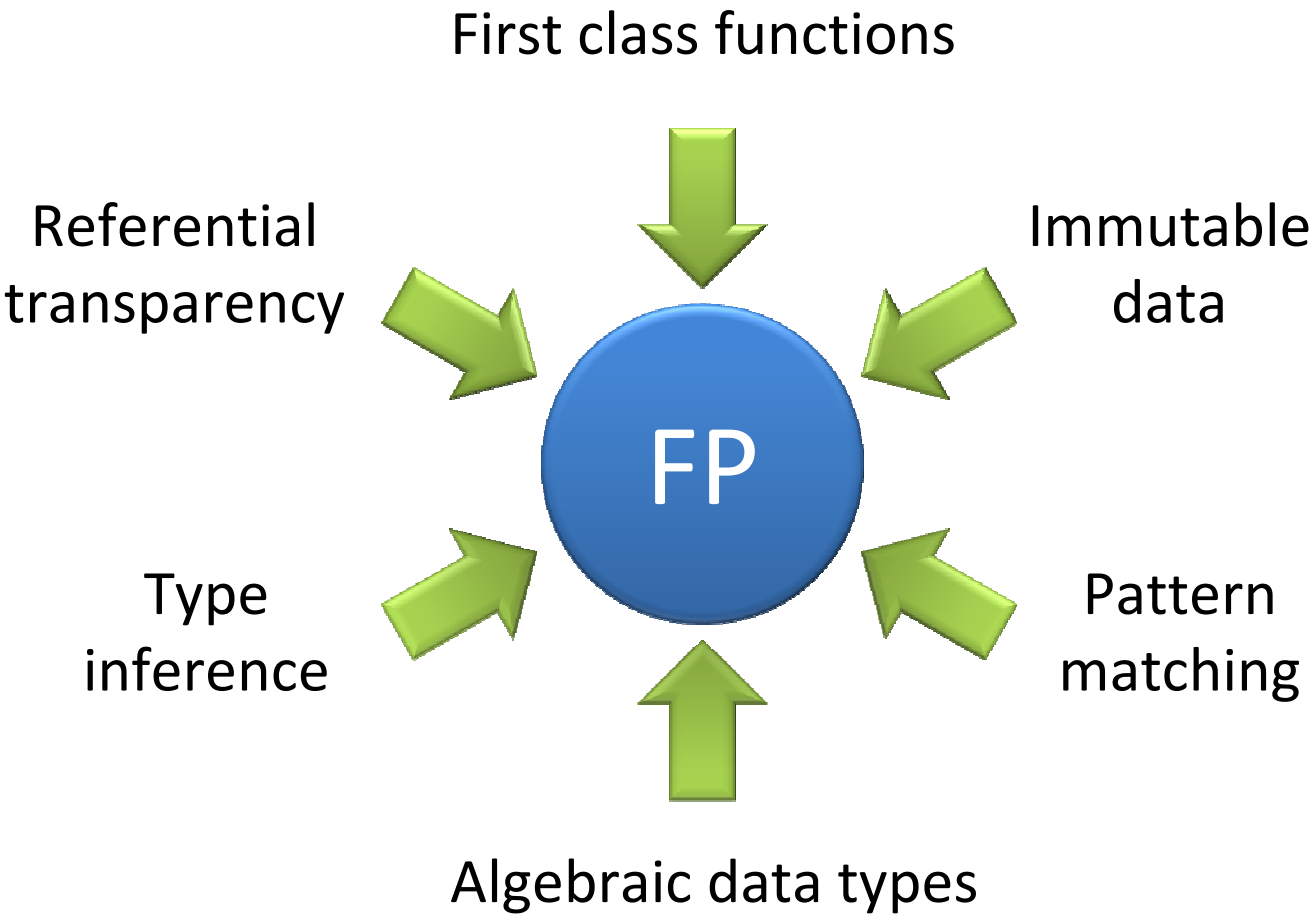
```
var query = db.Customers
    .Where(c => c.City == "London")
    .OrderBy(c => c.CompanyName)
    .Select(c => c.CompanyName);
```

```
class Order < ActiveRecord::Base
  belongs_to :customer
  has_many :details
end
```

```
Pattern socialSecurityNumber = Pattern.With.AtBeginning
    .Digit.Repeat.Exactly(3)
    .Literal("-").Repeat.Optional
    .Digit.Repeat.Exactly(2)
    .Literal("-").Repeat.Optional
    .Digit.Repeat.Exactly(4)
    .AtEnd;
```

LINQ Demo

Functional Programming



Imperative vs. Functional

$$x = x + 1$$

Imperative vs. Functional

$$y = x + 1$$

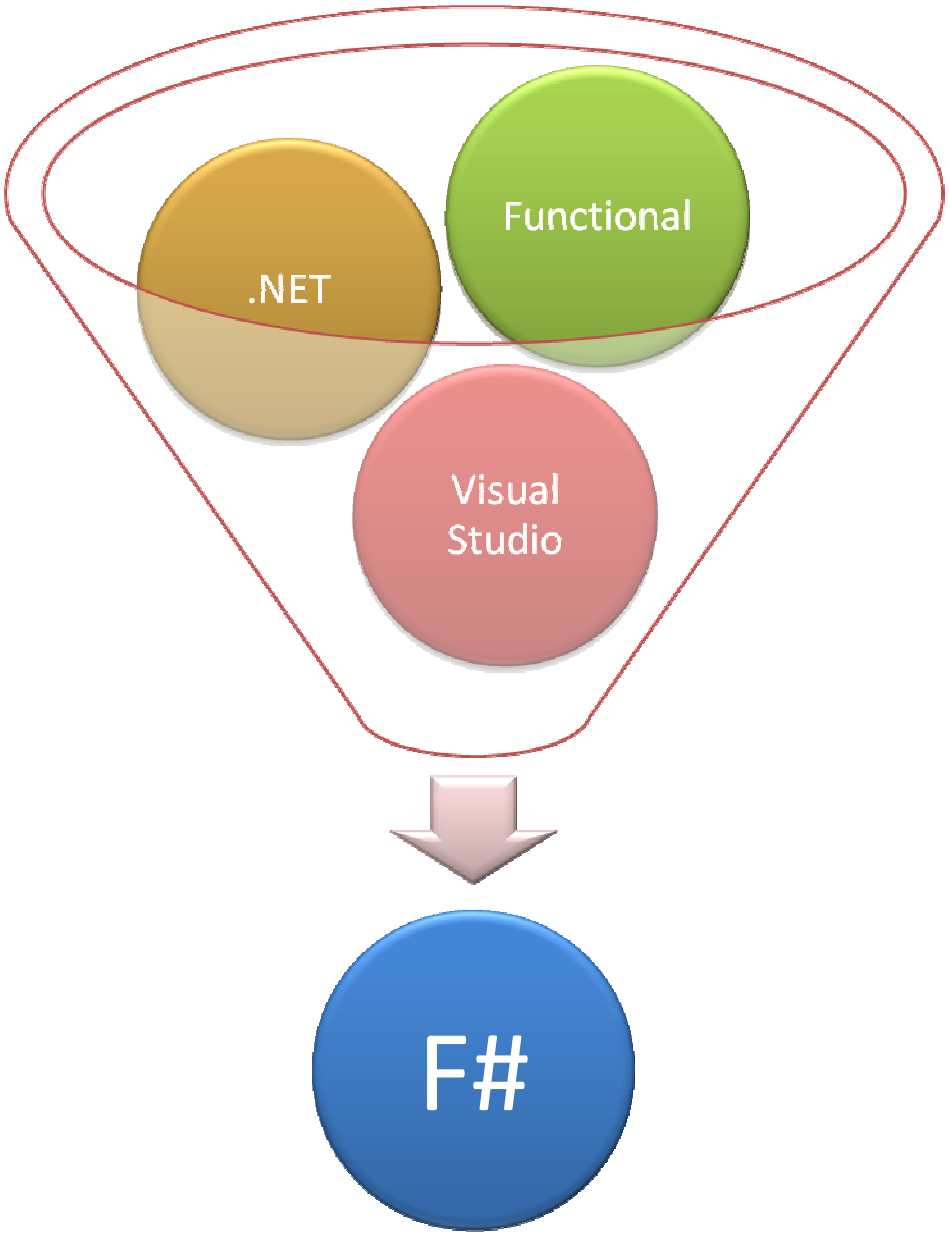
Imperative vs. Functional

Imperative

```
public class Point
{
    public int x;
    public int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void MoveBy(int dx, int dy) {
        x += dx;
        y += dy;
    }
}
```

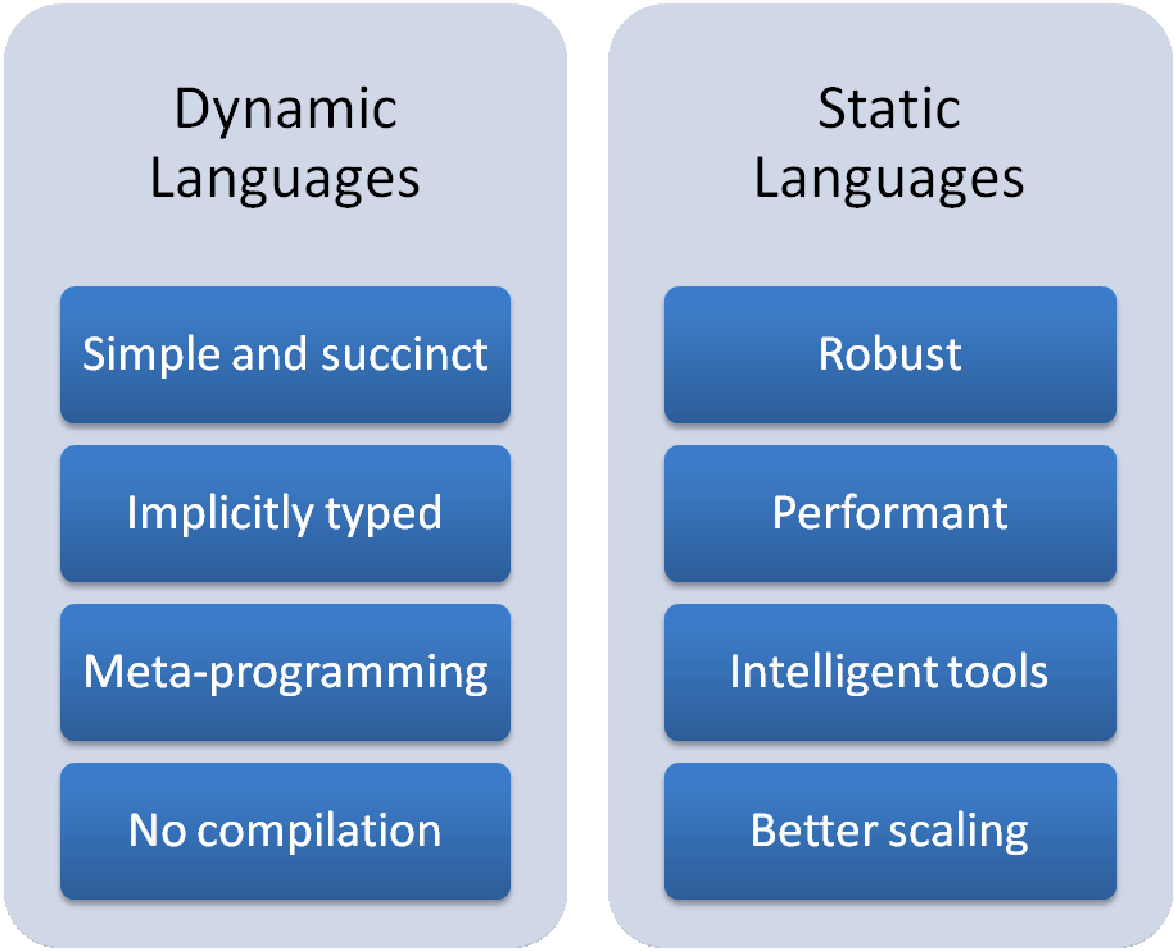
```
public class Point
{
    public readonly int x;
    public readonly int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public Point MoveBy(int dx, int dy) {
        return new Point(x + dx, y + dy);
    }
}
```

Functional


.NET
Functional
Visual Studio
F#

F# Demo

Dynamic vs. Static



Meta-Programming

“Meta-programming is the writing of programs that write or manipulate other programs (or themselves) as their data…”

Source: Wikipedia

```
class Order < ActiveRecord::Base
  belongs_to :customer
  has_many :details
end
```

.NET Dynamic Programming

IronPython | IronRuby | C# | VB.NET | Others…

Dynamic Language Runtime

Statement Trees | Dynamic Dispatch | Call Site Caching

Object Binder | JavaScript Binder | Python Binder | Ruby Binder | COM Binder

Static *and* Dynamic

```
Calculator calc = GetCalculator();
int sum = calc.Add(10, 20);
```

```
object calc = GetCalculator();
Type calcType = calc.GetType();
object res = calcType.InvokeMember("Add",
    BindingFlags.InvokeMethod, null,
    new int[] { 10, 20
int sum = Convert.ToIn
```

```
ScriptObject calc = GetCalculator();
object res = calc.Invoke("Add", 10, 20);
int sum = Convert.ToInt32(res);
```

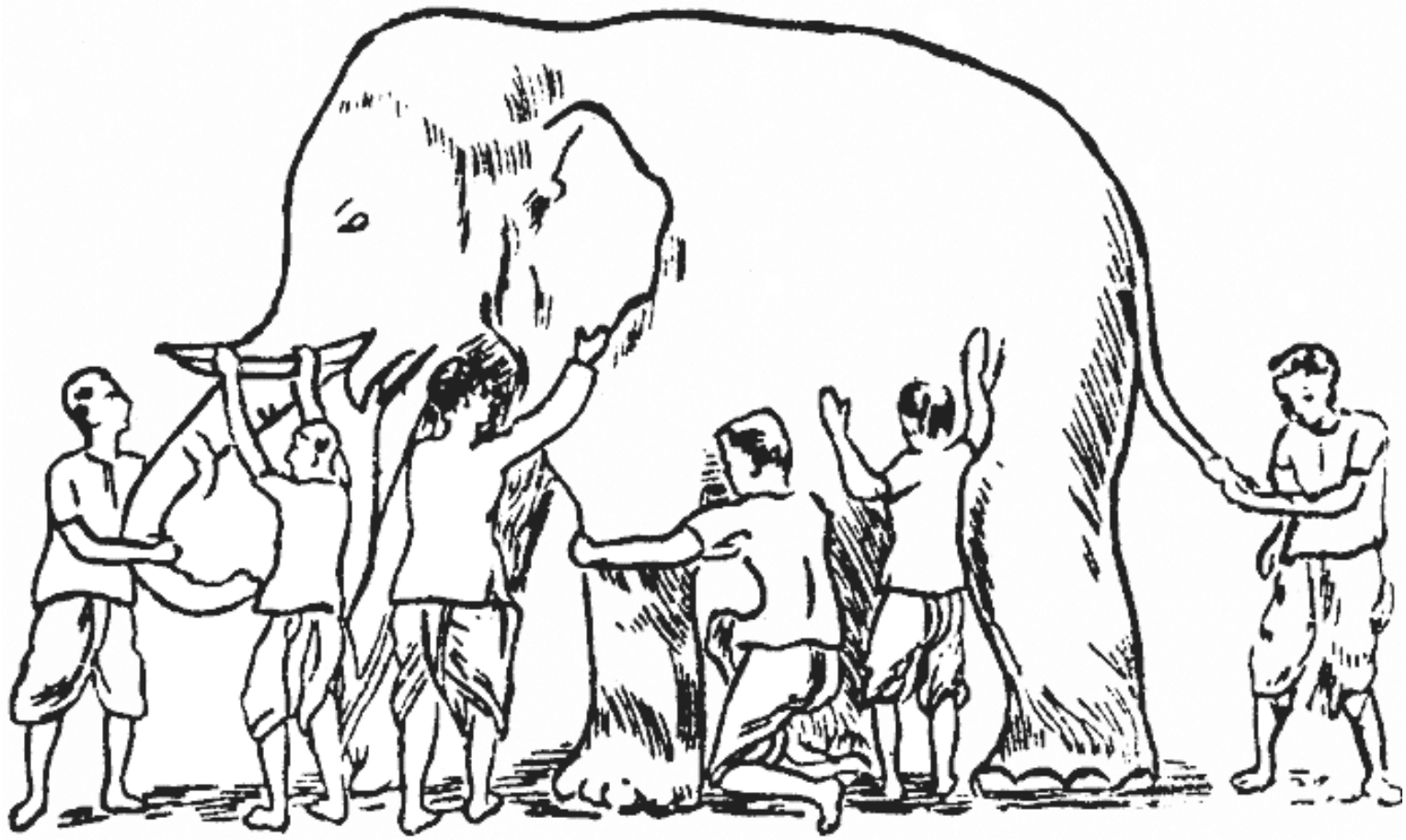
Statically typed to be dynamic

```
dynamic calc = GetCalculator();
int sum = calc.Add(10, 20);
```

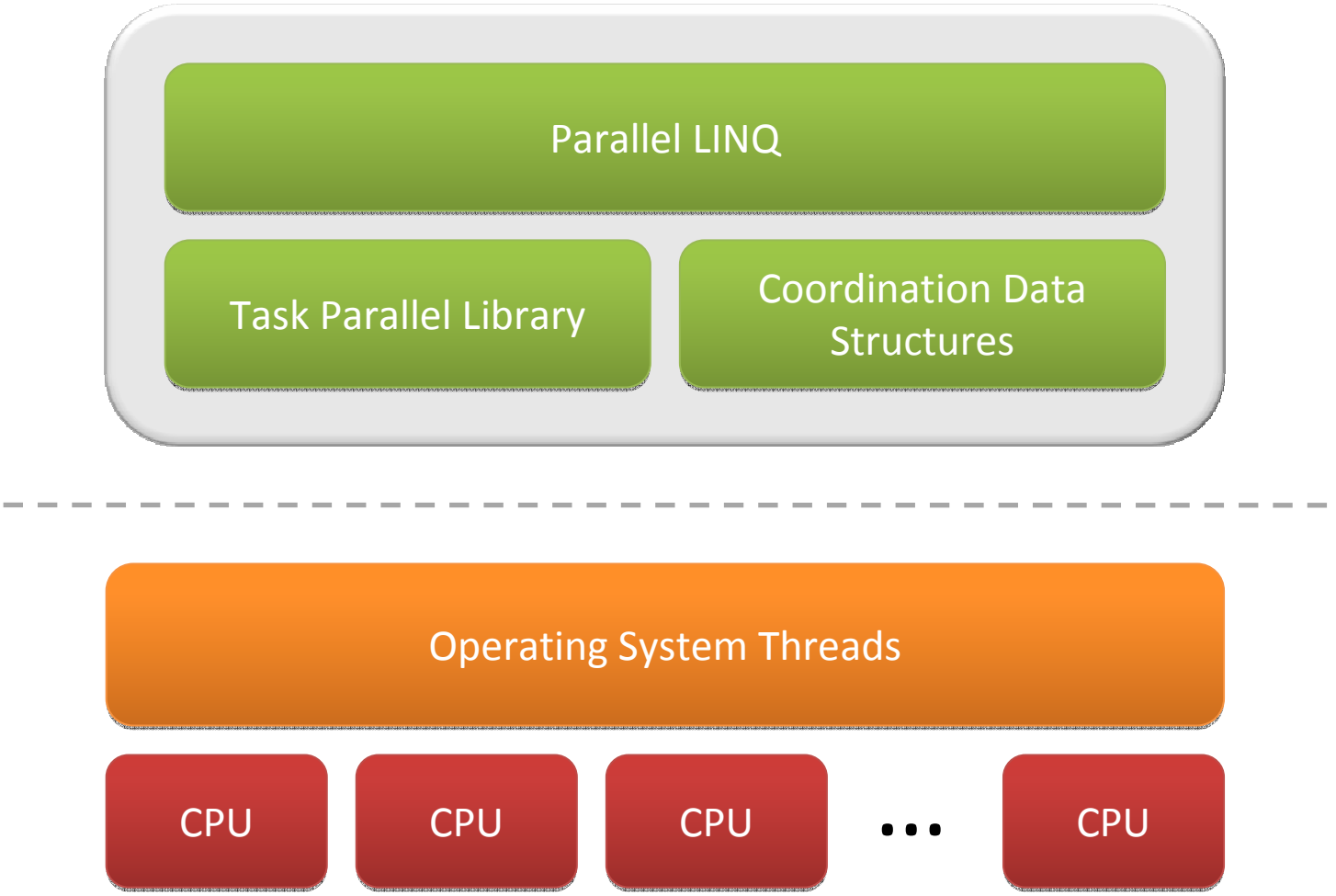
Dynamic conversion

Dynamic method invocation

Concurrency

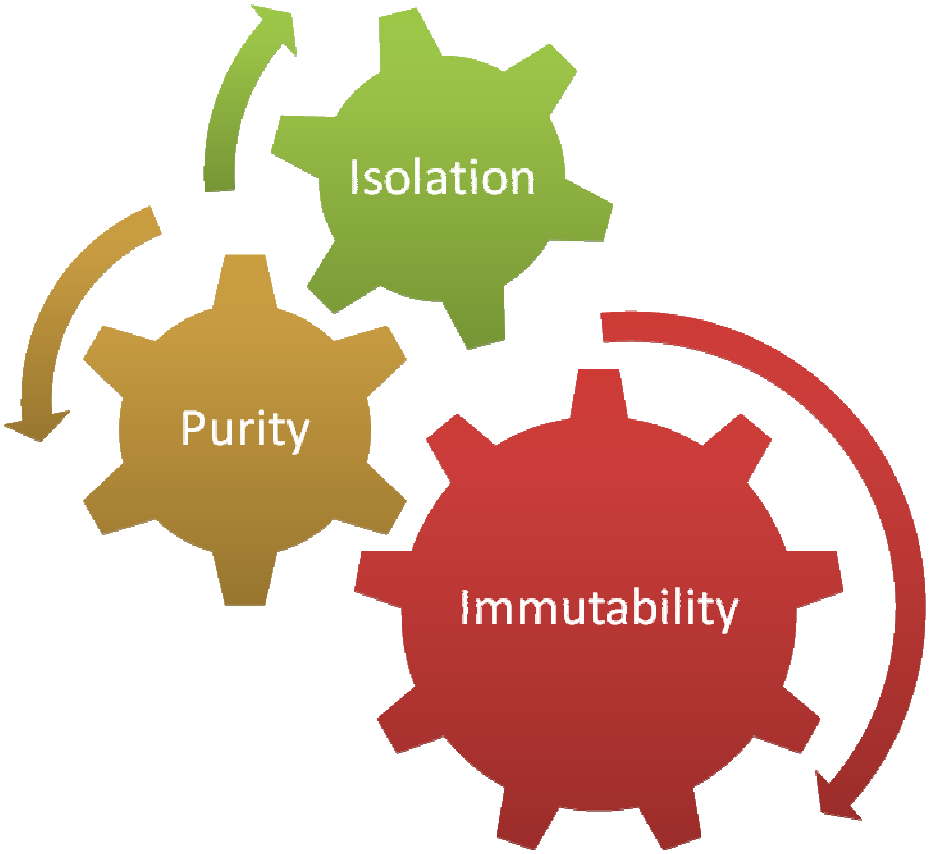


Parallel Extensions for .NET



Parallel Extensions Demo

Potential Language Constructs



Interesting Times

Resources

http://msdn.microsoft.com/fsharp

http://msdn.microsoft.com/concurrency

Microsoft®

Your potential. Our passion.™