



Jet loves F#. Here's why!

Rachel Reese | @rachelreese | @JetTechnology

+

A brief history



Programming languages evolve
just like regular languages.

Mother Tongues

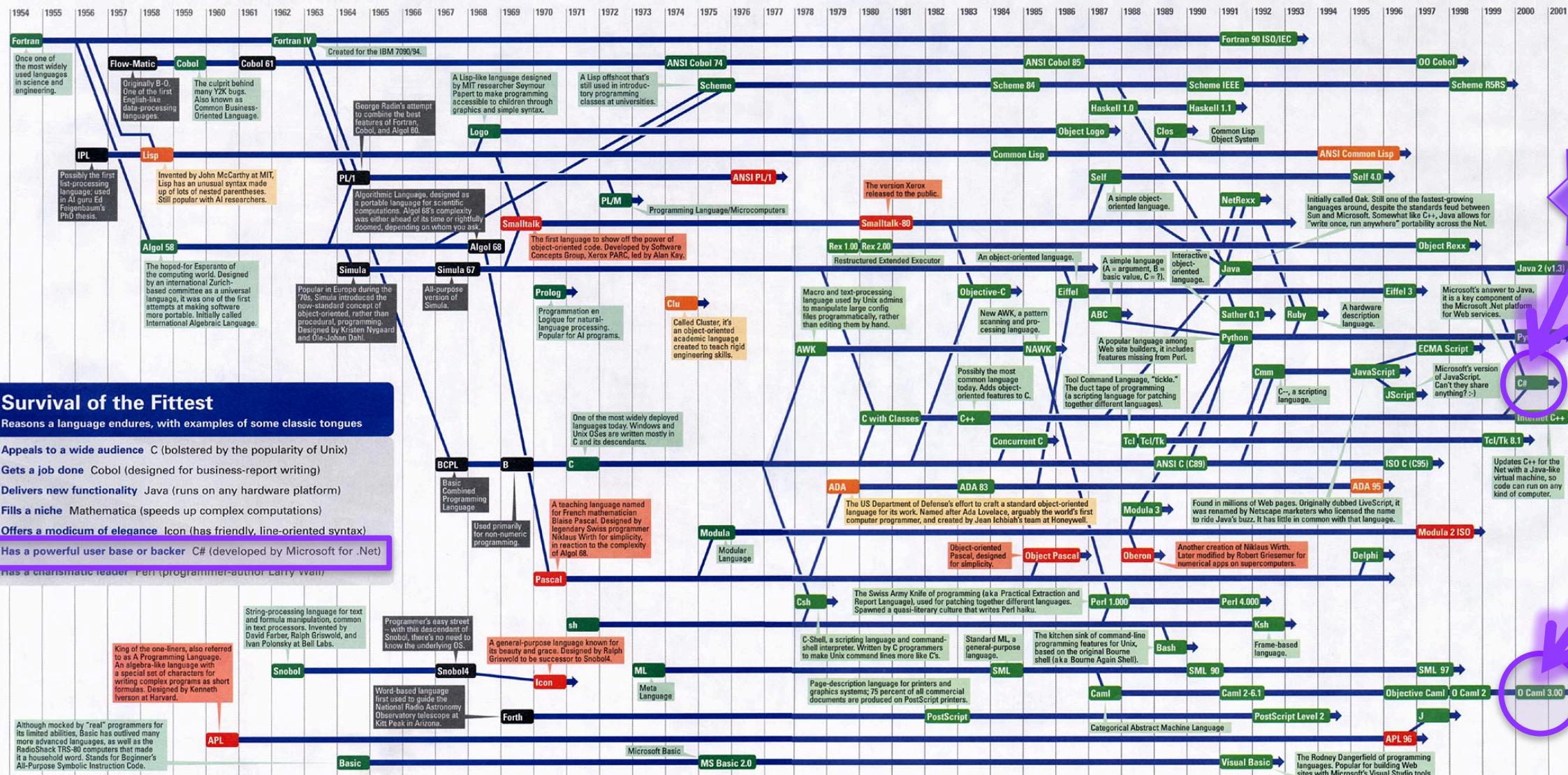
Tracing the roots of computer languages through the ages

Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java, and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers – electronic lexicographers, if you will – aim to save, or at least document, the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua francas. Among the most endangered are Ada, APL, B (the predecessor of C), Lisp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Museum in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at www.informatik.uni-freiburg.de/Java/misc/lang_list.html. – Michael Menduno

Key	
1954	Year Introduced
Active:	thousands of users
Protected:	taught at universities; compilers available
Endangered:	usage dropping off
Extinct:	no known active users or up-to-date compilers
→	Lineage continues





Why F#?

Testimonials

“ I have now delivered three business critical projects written in F#. I am still waiting for the first bug to come in. ”

- Simon Cousins
UK-based power company

The F# solution offers us an order of magnitude increase in productivity and allows one developer to perform the work [of] a team of dedicated developers... ”

- Yan Cui
Lead server Engineer, Gamesys

“ The F# code is consistently shorter, easier to read, easier to refactor and contains far fewer bugs. ”

- Kaggle, a predictive modelling company



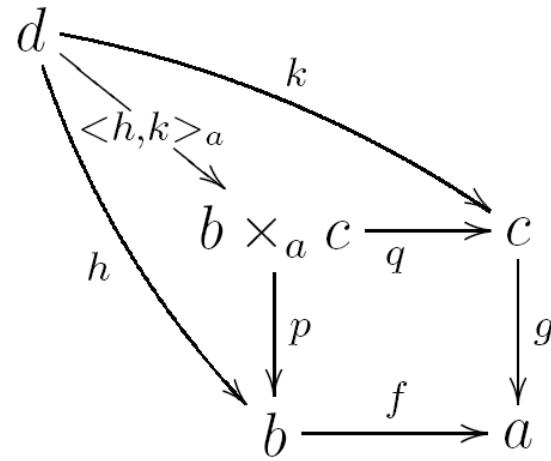
All of this seems incredible, right?



So why doesn't everyone use F#?



There's a lot of math behind programming languages like F#.



$\mathcal{C}^{\text{add. cat.}} = X \oplus R$
Abelian
Brave
Inv. to -algebra.
Azumaya
Brauer
 $\mathcal{D}\mathcal{F}(X) = X \rightarrow Y \rightarrow Z \rightarrow \Sigma X$

op. f-theory. $= I$ cat. $\sim (-)/\text{homotopy arr.}$ mor. \mathcal{G}
- $D(A)$? \sim What is a tria. cat.? $K(A)$ sh. -
pure math | (Comp. sci.) | localize arr. [$A \oplus B$]
- Add. Cat. | $\mathbb{A}_{\mathbb{R}}$ -enriched | \mathbb{Z} | $D(A)$
(+ zero obj.) \mathbb{N} | + fin. biproduct. (direct sum) | \mathbb{Z} | \mathbb{Z}
+ kernel cokernel | \mathbb{Z} | \mathbb{Z} | \mathbb{Z}
+ hom. th. H | \mathbb{Z} | \mathbb{Z} | \mathbb{Z}
 $\mathbb{Z} \xrightarrow{F} \mathbb{Z} \xrightarrow{DF} \mathbb{Z}$

functor $F: \mathcal{C} \rightarrow \mathcal{D}$

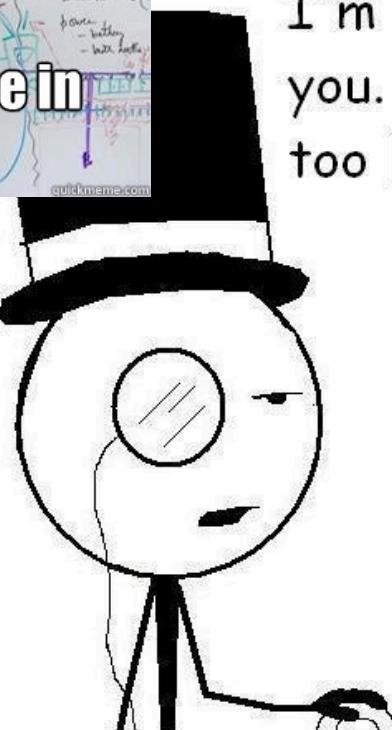
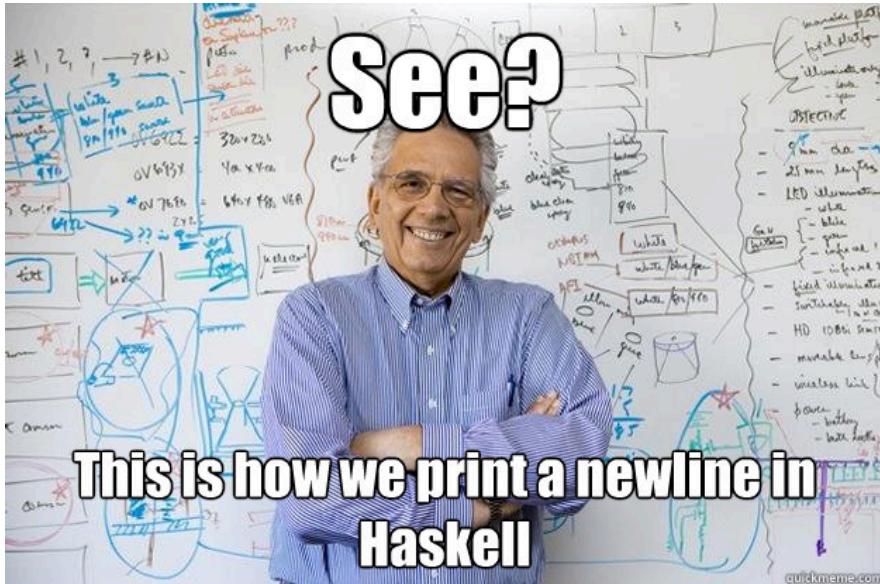
$$\begin{array}{ccccc} U & \xrightarrow{h} & V & \xrightarrow{g} & W \\ \downarrow & & \downarrow & & \downarrow \\ F(U) & \xrightarrow{F(h)} & F(V) & \xrightarrow{F(g)} & F(W) \end{array}$$

$\in \text{category } \mathcal{C}$ $\in \text{category } \mathcal{D}$

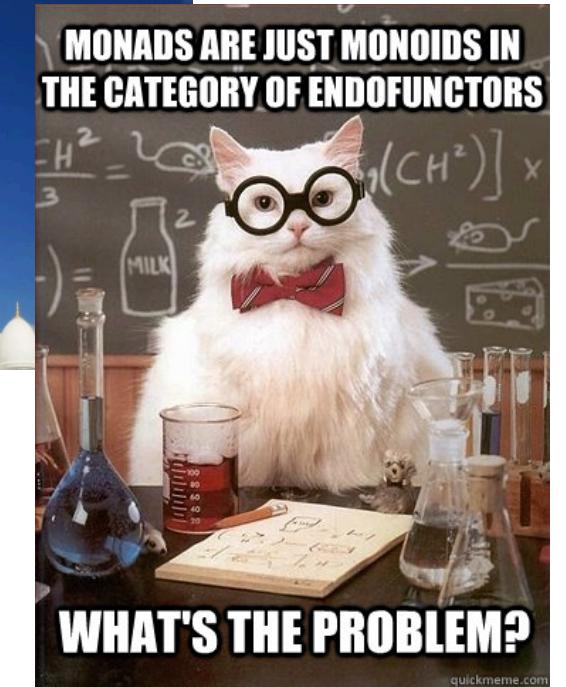
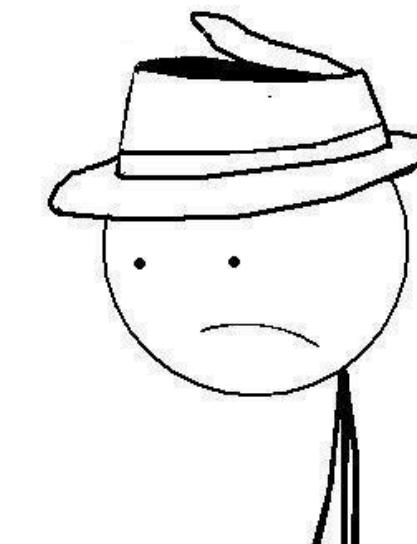
F

+

So they have this reputation.



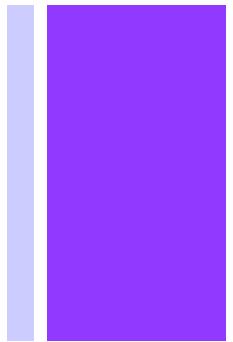
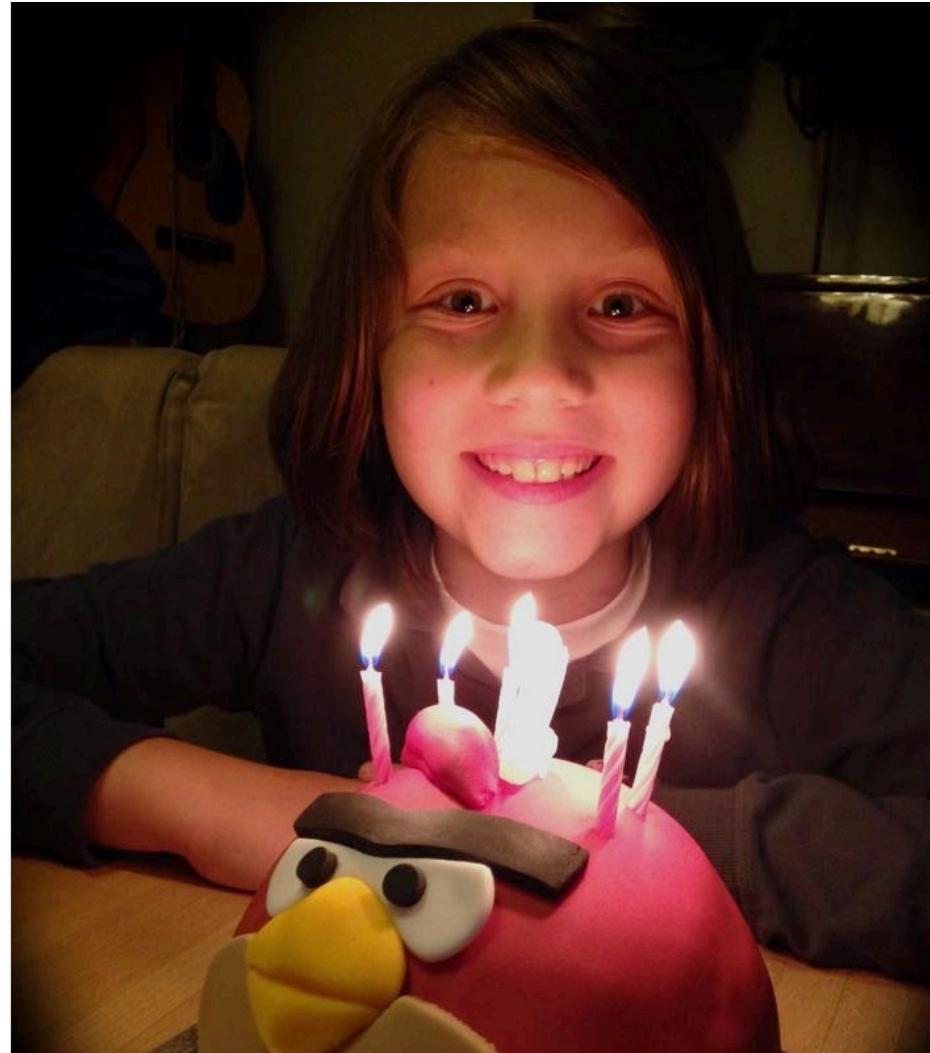
I'm sorry, I can't hear you. Your inferiority is too loud.



quickmeme.com

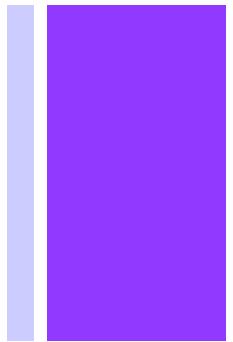
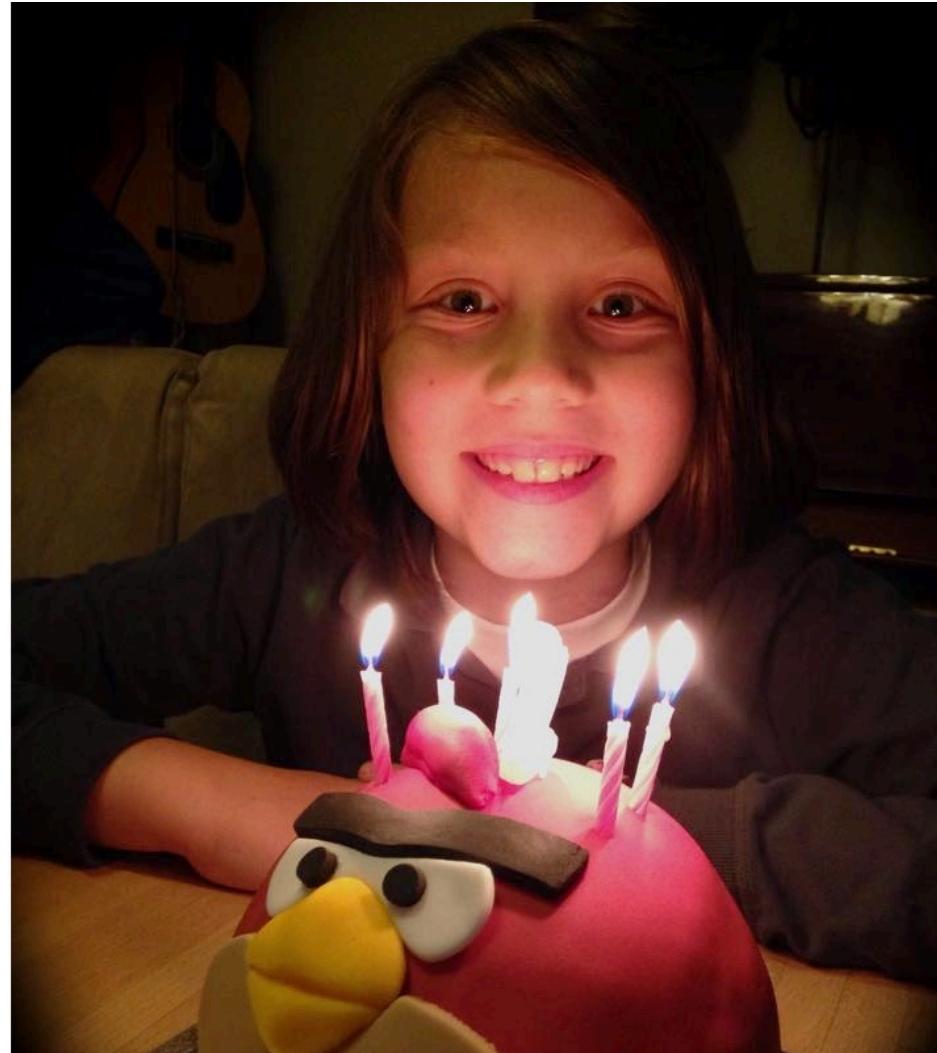
+

This is Sean.



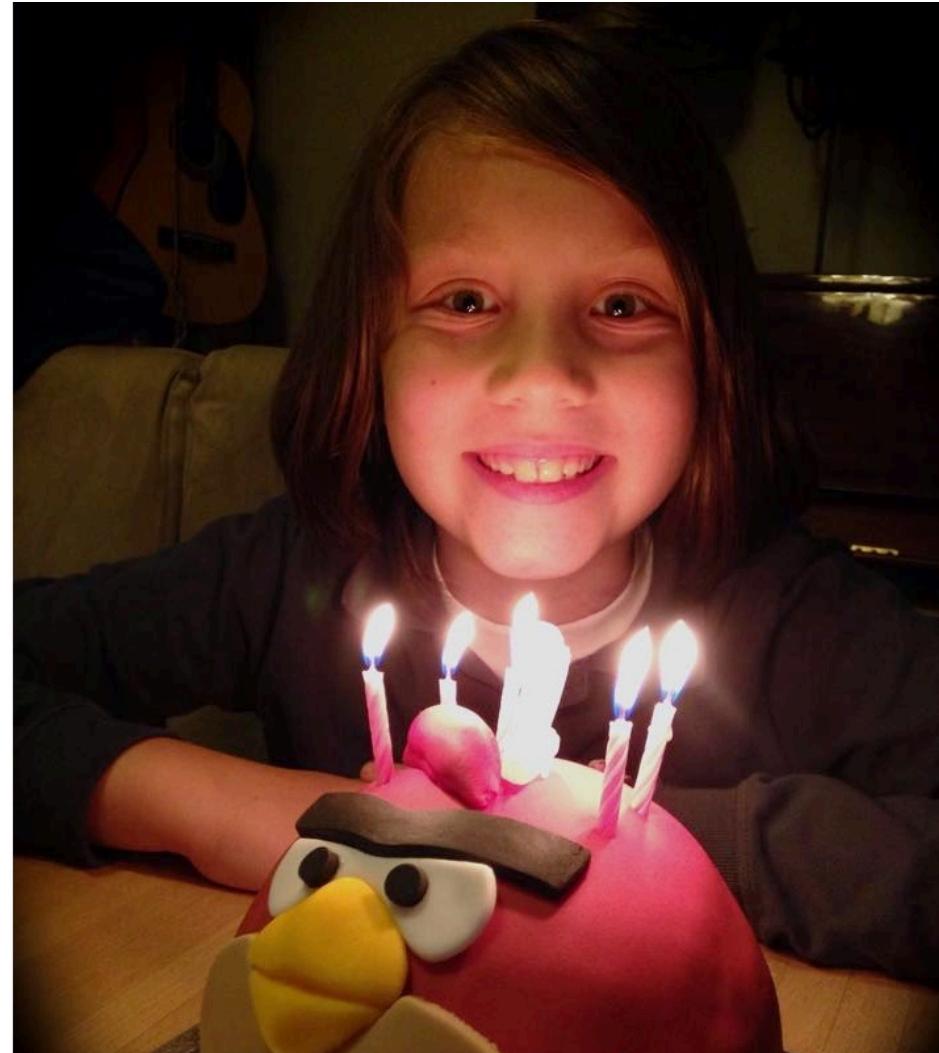
+

Sean is 8.



+

Sean spoke at a 2000-person technical conference this year. On F#.



+

Why F#?

Concise code & expanded feature set



Concise code

C# - 23 lines

```
using System.Net
using System
using System.IO

class WebPageDownloader
{
    public TResult FetchUrl<TResult>(
        Func<string, StreamReader, TResult> callback,
        string url)
    {
        var req = WebRequest.Create(url);
        using (var resp = req.GetResponse())
        {
            using (var stream = resp.GetResponseStream())
            {
                using (var reader = new StreamReader(stream))
                {
                    return callback(url, reader);
                }
            }
        }
    }
}
```

F# - 10 lines

```
open System.Net
open System
open System.IO

let fetchUrl callback url =
    let req = WebRequest.Create(Uri(url))
    use resp = req.GetResponse()
    use stream = resp.GetResponseStream()
    use reader = new IO.StreamReader(stream)
    callback reader url
```



Concise code

C#

```
public abstract class Transport{ }

public abstract class Car : Transport {
    public string Make { get; private set; }
    public string Model { get; private set; }
    public Car (string make, string model) {
        this.Make = make;
        this.Model = model;
    }
}

public abstract class Bus : Transport {
    public int Route { get; private set; }
    public Bus (int route) {
        this.Route = route;
    }
}

public class Bicycle: Transport {
    public Bicycle() {
    }
}
```

F#

```
type Transport =
| Car of Make:string * Model:string
| Bus of Route:int
| Bicycle
```

Same info as C#!

Trivial to pattern match on!

F# pattern matching

C#

Name	Description	Example
Constant pattern	Any numeric, character, or string literal, an enumeration constant, or a defined literal identifier	1.0, "test", 30, Color.Red
Identifier pattern	A case value of a discriminated union, an exception label, or an active pattern case	Some(x) Failure(msg)
Variable pattern	<i>identifier</i>	a
as pattern	<i>pattern as identifier</i>	(a, b) as tuple1
OR pattern	<i>pattern1 pattern2</i>	([h] [h; _])
AND pattern	<i>pattern1 & pattern2</i>	(a, b) & (_ , "test")
Cons pattern	<i>identifier :: list-identifier</i>	h :: t
List pattern	[<i>pattern_1; ... ; pattern_n</i>]	[a; b; c]
Array pattern	[<i>pattern_1; ..; pattern_n</i>]	[a; b; c]
Parenthesized pattern	(<i>pattern</i>)	(a)
Tuple pattern	(<i>pattern_1, ..., pattern_n</i>)	(a, b)
Record pattern	{ <i>identifier1 = pattern_1; ... ; identifier_n = pattern_n</i> }	{ Name = name; }
Wildcard pattern	-	-
Pattern together with type annotation	<i>pattern : type</i>	a : int
Type test pattern	:? <i>type</i> [as <i>identifier</i>]	:? System.DateTime as dt
Null pattern	null	null



Expanded feature set: Option types



Tomas Petricek

@tomaspetricek



Following

> @davetchepak "What can C# do that F# cannot?" NullReferenceException :-)



Reply



Retweeted



Favorite



More



Expanded feature set: Units of measure

```
[<Measure>] type F // degrees Fahrenheit
[<Measure>] type C // degrees Celsius
[<Measure>] type mi // miles
[<Measure>] type km // kilometers
[<Measure>] type hr // hour

let WindChill_US (T:float<F>) (v:float<mi/hr>) =
    35.74<F> + 0.6215 * T-35.75<F> * float(v) ** 0.16 + 0.3965 * T * float(v) ** 0.16

let WindChill_CA (T:float<C>) (v:float<km/hr>) =
    13.12<C> + 0.6215 * T-11.37<C> * float(v) ** 0.16 + 0.4275 * T * float(v) ** 0.16
```



Expanded feature set: Type providers

*Building Data Centric Apps with
the ADO.NET Entity Framework*

2nd Edition
thoroughly Revised

920 pages!

Programming

Entity
Framework

O'REILLY®

Copyrighted Material

Julia Lerman

```
1  #r "FSharp.Data.TypeProviders"
2  #r "System.Data.Linq"
3  #r "System.Data.Entity"
4  #load @"..\packages\FSharp.Charting.0.90.9\FSharp.Charting.fsx"
5
6  open Microsoft.FSharp.Data.TypeProviders
7  open System.Data.Linq
8  open System.Data.Entity
9  open FSharp.Charting
10
11 // Sum & graph landing counts by airline -- all flights, for all time
12 module SqlDataConnectionSample =
13     type internal SFOData = SqlEntityConnection<ConnectionStringName = "SFO", ForceUpdate=true, Pl
14         let internal SFOContext = SFOData.GetDataContext()
15
16     let internal SFOInfo =
17         query { for data in SFOContext.RawLandingsData do
18             where (not (data.PublishedAirline.Contains("United Airlines"))) // again, numbers
19             groupValBy data.LandingCount data.PublishedAirline into g
20             let total = query { for group in g do sumBy group }
21             select (g.Key, total)
22         } |> Chart.Column
23
24
25     // Add landing counts for a new airline.
26     let internal NewSFO =
27         SFOData.ServiceTypes.RawLandingsData.CreateRawLandingsData(0, 201501., "RachelsAirline", "
28
29
30     SFOContext.DataContext.AddObject("RawLandingsData", NewSFO)
31     SFOContext.DataContext.SaveChanges()
32
```

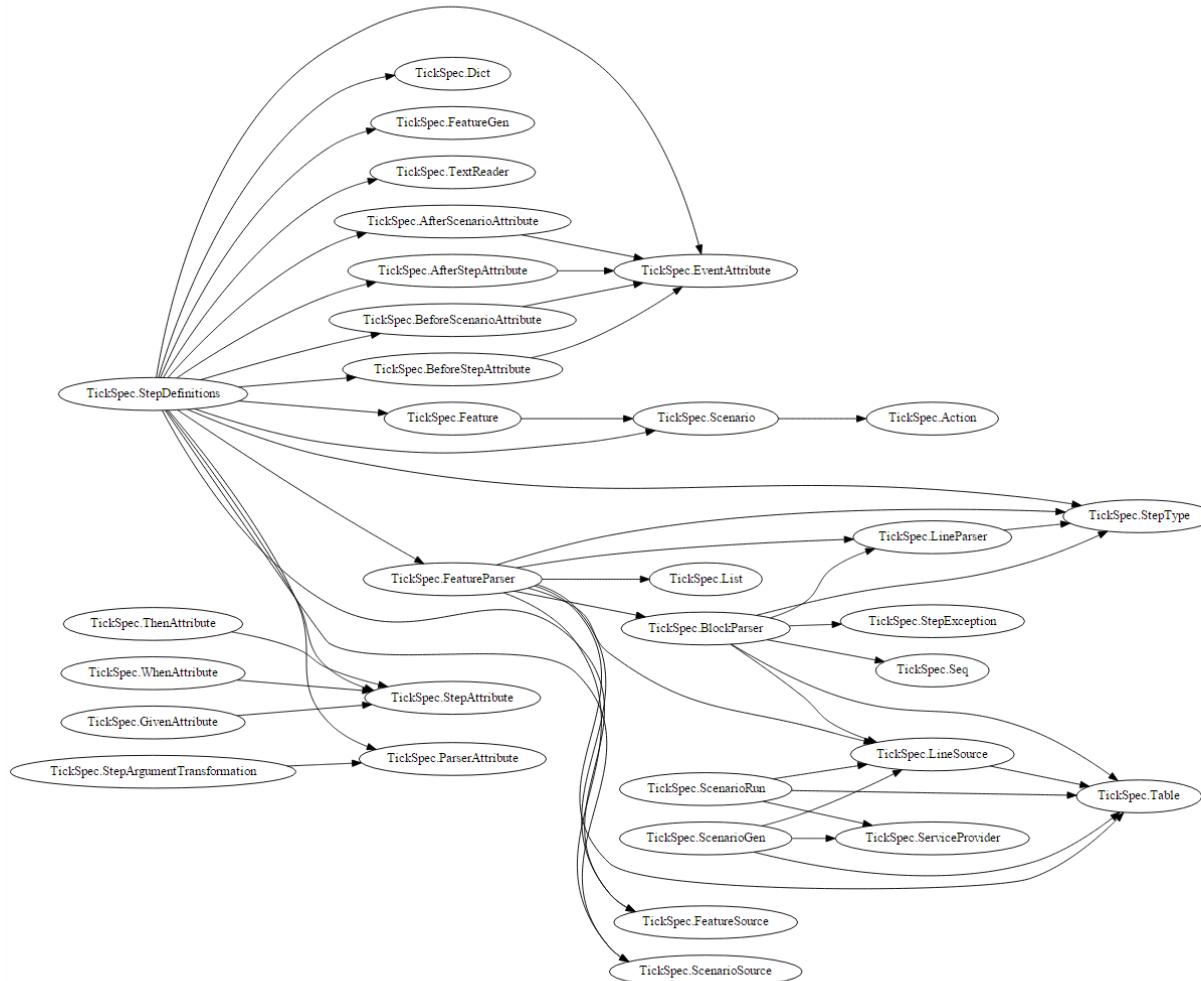
+

Why F#?

Dependencies

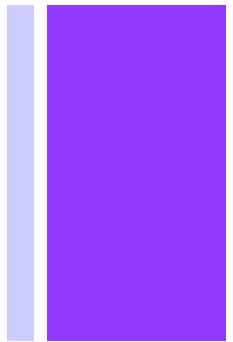


TickSpec – an F# project



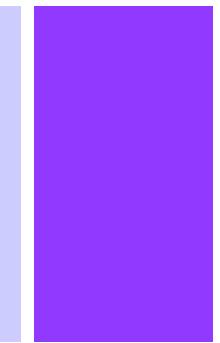
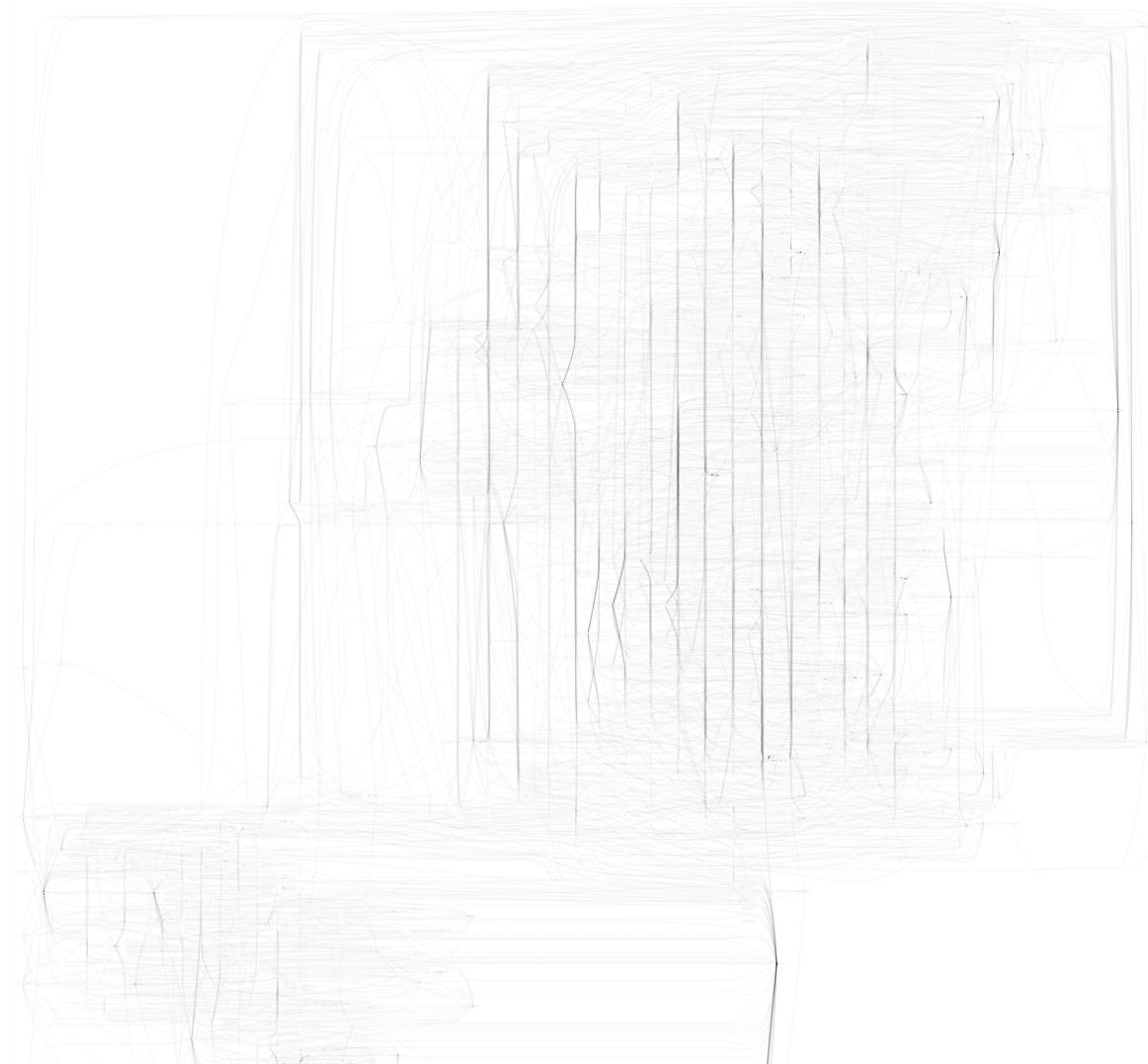
+

SpecFlow – a comparable C# project





Entity Framework (C#)





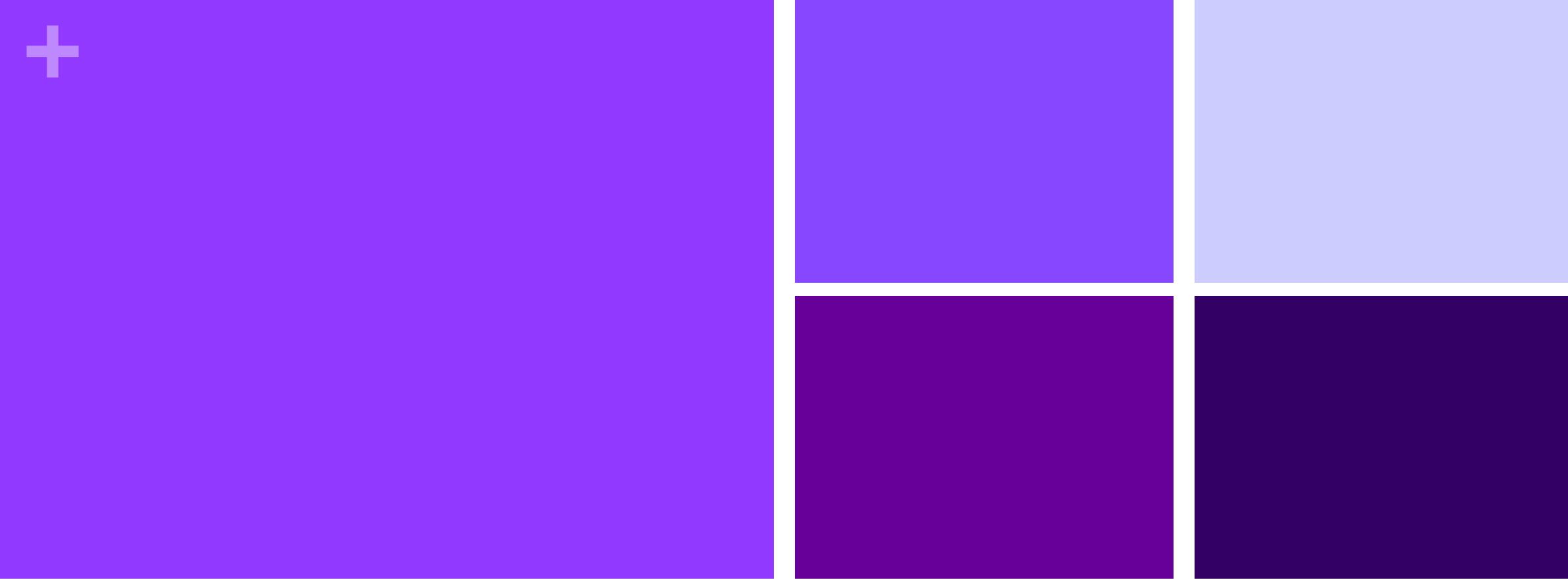
Additional libraries to check out

- FsCheck: <https://github.com/fsharp/FsCheck>
- Canopy: <http://lefthandedgoat.github.io/canopy/>
- FAKE: <http://fsharp.github.io/FAKE/>
- Paket: <http://fsprojects.github.io/Paket/>
- Type Providers:
 - Powershell: <http://fsprojects.github.io/FSharp.Management/PowerShellProvider.html>
 - FSharp.Data: <http://fsharp.github.io/FSharp.Data/>
 - FSharp.Configuration: <http://fsprojects.github.io/FSharp.Configuration/>



General resources & additional reading

- General resources
 - Me!
 - Team leads
 - Slack - #fsharp channel
 - [F# chat on SO](#)
 - <http://fsharp.org/>
 - Twitter: #fsharp
 - [F# channel on Functional Programming Slack](#)
- Additional reading
 - [F# for Fun and Profit](#)
 - [F# Weekly](#)
 - [Try F#](#)



Jet loves F#. Here's why!

Rachel Reese | @rachelreese