

F# is the future of OSS .NET

Open FSharp 2017, San Francisco

@lenadroid



Wow!
F# is so cool!
Powerful and
functional



**Community
is
Power**

The background of the image features several thick, light green brushstrokes that are scattered across the white space. These strokes vary in length and orientation, some pointing downwards and others more horizontally, creating a textured, artistic effect.

Independence



Impact



Voice



Openness

Should I use F# for my project?

Yes

No



What makes F# F#

- + Functional first, and more paradigms!
- + Strong typing and type inference
- + Default immutability
- + Fully OSS!
- + Cross platform
- + .NET interoperability
- + F# interactive, explorative programming
- + Pattern matching
- + Type providers
- + Computation Expressions

IDEs

- + VS Code
- + Xamarin
- + Visual Studio
- + Rider
- + Atom with Ionide
- + Sublime
- + Vim
- + Emacs
- + MonoDevelop
- + More

OSes and platforms

- + Windows
- + Linux
- + OS X
- + Mobile (iOS, Android, etc.)
- + IoT (i.e. Azure IoT)
- + Docker
- + GPUs
- + JS ecosystem through Fable

Myths about F#

- + It is only for scientific projects and math
- + It is beyond complicated to get started with

The image features the two hosts of MythBusters, Adam and Jamie, standing behind a large, stylized sign that reads "MYTHBUSTERS". Adam is on the left, wearing a black t-shirt and glasses. Jamie is on the right, wearing a white lab coat, a black beret, and glasses. The background is a dark blue space-themed scene with stars and faint mathematical formulas like "2(B+C)" and "E=MC^2". A blue diamond-shaped logo is positioned above the hosts. The sign they are holding has a metallic, rusted appearance with the word "MYTHBUSTERS" in bold, white, block letters.

MYTHBUSTERS

C#

```
public class Item
{
    public Item(string id, string name, double price)
    {
        Id = id;
        Name = name;
        Price = price;
    }

    public string Id { get; }

    public string Name { get; }

    public double Price { get; }
}

public interface IShoppingCart
{
    IShoppingCart AddItem(Item item);
    IShoppingCart Empty();
}

public class EmptyShoppingCart : IShoppingCart
{
    public IShoppingCart AddItem(Item item)
    {
        return new NonEmptyShoppingCart(ImmutableList.Create(item));
    }

    public IShoppingCart Empty()
    {
        return this;
    }
}

public class NonEmptyShoppingCart : IShoppingCart
{
    public NonEmptyShoppingCart(ImmutableList<Item> items)
    {
        Items = items;
    }

    public IShoppingCart AddItem(Item item)
    {
        return new NonEmptyShoppingCart(Items.Add(item));
    }

    public IShoppingCart Empty()
    {
        return new EmptyShoppingCart();
    }

    public ImmutableList<Item> Items { get; }
}
```

```
type Item =
{ Id: string
  Name: string
  Price: float }

type ShoppingCart =
{ Items: Item list }
member this.AddItem item = { this with Items = item :: this.Items }
static member Empty = { Items = [] }
```

F#

C#

```
public interface ICommand { }

public class AddItem : ICommand
{
    public AddItem(Item item)
    {
        Item = item;
    }

    public Item Item { get; set; }
}

public class Buy : ICommand
{
    public static Buy Instance { get; } = new Buy();
    private Buy() { }
}

public class Leave : ICommand
{
    public static Leave Instance { get; } = new Leave();
    private Leave() { }
}

public class GetCurrentCart : ICommand
{
    public static GetCurrentCart Instance { get; } = new GetCurrentCart();
    private GetCurrentCart() { }
}
```

F#

```
type Command =
| AddItem of Item
| Buy
| Leave
| GetCurrentCart
```

MYTH BUSTED!



Mark Gray

@MarkRGray

Following



Just starting to teach my 9 year old nephew programming, and it is amazing how easy he is grasping the concepts using #fsharp

6:48 AM - 12 Aug 2017

3 Retweets 21 Likes



1



3



21



Tweet your reply



Mark Gray @MarkRGray · Aug 12



Replying to @MarkRGray

Tabs v spaces: he chooses spaces naturally :-D #TheImportantThings



1



2



vaskir 🍅 @kot_2010 · Aug 14

because tabs are invisible magic.



MYTH BUSTED!

Java engineers exploring F#



Data Science engineers exploring F#



Paige Bailey

@DynamicWebPaige

Following



It's functional like Clojure; you can use .NET tooling; & it's supported in Jupyter notebooks?

Okay: [@lenadroid](#) just convinced me to try F#

9:50 AM - 25 Aug 2017 from Redmond, WA

5 Retweets 21 Likes



C#? F#?

* Is it harder to
learn F# than C#?

* (no)

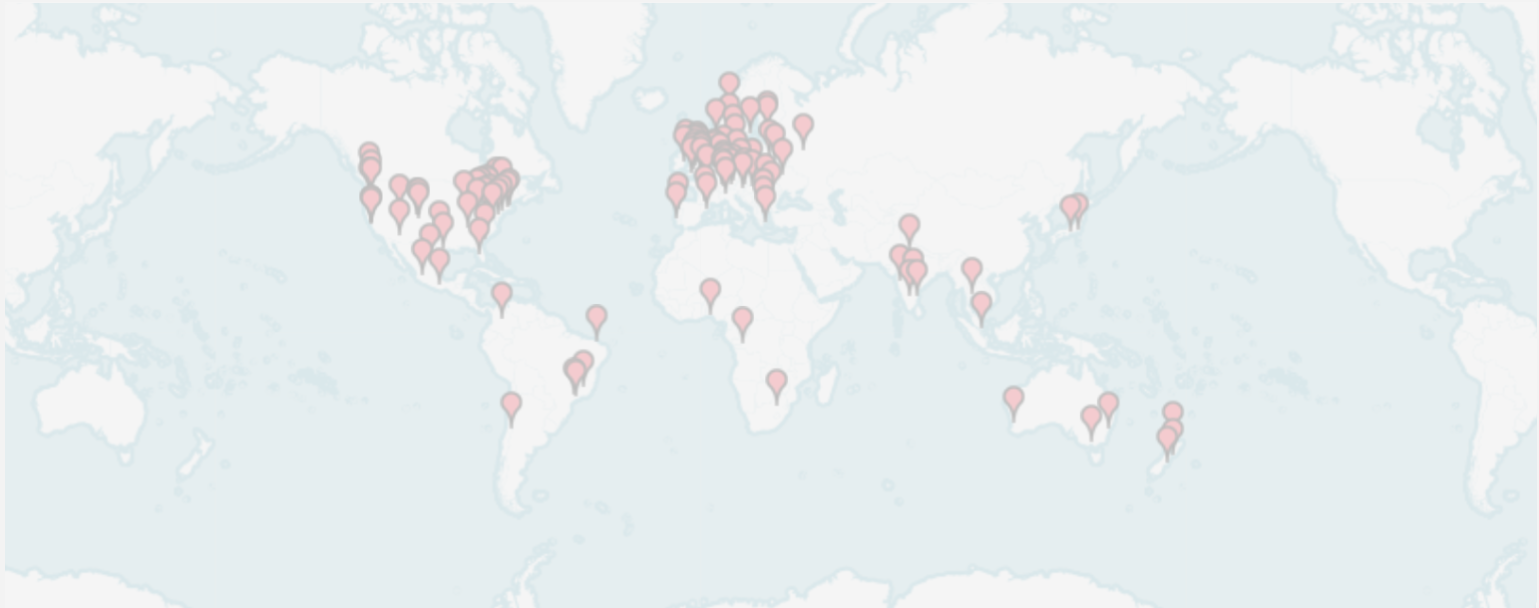
- Faster “time to market”
- Correctness in business logic
- Ease of maintenance
- Freedom of options, fully OSS
- Helpful and friendly community
- Cutting edge language features
- Smaller code base
- Any C# programmer can learn F#
- Can use F# with existing C# libraries

Thousands of Github repositories use F#

Dozens of thousands users all over the world

More than 140 contributors to fsharp/fsharp

More than 5,150 members in F# meetups everywhere



Personal top favorites
OSS F# projects

MBrace cloud { ... }

Run scalable, distributed data parallel workflows in the cloud

```
1: open MBrace.Core
2: open MBrace.Flow
3:
4: let numberOfDuplicates =
5:     CloudFlow.OfCloudFilesByLine ["container/data0.csv" ; "container/data1.csv"]
6:     |> CloudFlow.map (fun line -> line.Split(','))
7:     |> CloudFlow.map (fun tokens -> int tokens.[0], Array.map int tokens.[1 ..])
8:     |> CloudFlow.groupBy (fun (id,_) -> id)
9:     |> CloudFlow.filter (fun (_,values) -> Seq.length values > 1)
10:    |> CloudFlow.length
11:    |> cluster.Run
```


MBrace.Core - Cloud Programming Made Simple

Confused by the cloud? Cloud computation and data can be simple, if using the right framework. MBrace.Core helps the cloud empower you, not enslave you.

- ✓ Cloud Workflows
- ✓ Concurrent
- ✓ Fault Tolerant
- ✓ High Availability
- ✓ Functional Cloud Data Flows
- ✓ Partitioned Cloud Vectors
- ✓ Abstract and Reuse Common Cloud Patterns
- ✓ Strong Typing For Cloud Data
- ✓ Full F# and C# Support
- ✓ Cloud Vendor Neutral
- ✓ 100% Open Source
- ✓ Extensible Data Serialization

MBrace.Azure - Included Features

Whether new to Azure or an advanced Azure developer, MBrace.Azure brings Azure storage and compute to your fingertips.

- ✓ Cloud Scripting from Your Editor
- ✓ Programmatic Data Upload
- ✓ Automatic Code Transport
- ✓ Extensible Data Serialization
- ✓ Fault Tolerance Options
- ✓ Full F# and C# Support
- ✓ Job Creation and Control
- ✓ Integrated Cloud Logging
- ✓ Smooth Transitions From Scripts to Code
- ✓ Nuget Packages
- ✓ All MBrace.Core Features
- ✓ Interoperate With Azure Services
- ✓ Local Prototyping
- ✓ Native CPU Performance
- ✓ 100% Open Source



Lightweight web server for F# web applications

Simple non-blocking web server

```
open System
```

```
open System.Threading
```

```
open Suave
```

```
[<EntryPoint>]
```

```
let main argv =
```

```
    let cts = new CancellationTokenSource()
```

```
    let conf = { defaultConfig with cancellationToken = cts.Token }
```

```
    let listening, server = startWebServerAsync conf (Successful.OK "Hello World")
```

```
    Async.Start(server, cts.Token)
```

```
    printfn "Make requests now"
```

```
    Console.ReadKey true |> ignore
```

```
    cts.Cancel()
```

```
    0 // return an integer exit code
```

Routing in Suave

```
open Suave
open Suave.Filters
open Suave.Operators
open Suave.Successful

let app =
  choose
    [ GET >=> choose
      [ path "/hello" >=> OK "Hello GET"
        path "/goodbye" >=> OK "Good bye GET" ]
      POST >=> choose
        [ path "/hello" >=> OK "Hello POST"
          path "/goodbye" >=> OK "Good bye POST" ] ] ]

startWebServer defaultConfig app
```



Samples: ▾

ES2015

Compile

Run

Fable v1.2.0, FCS v14.0.2, Babel v6.26

```
1 let factorial n =
2     let rec loop i acc =
3         match i with
4         | 0 | 1 -> acc
5         | _ -> loop (i-1) (acc * i)
6     loop n 1
7
8 factorial 10
9 |> printfn "%i"
10
```

```
1 import { printf } from "fable-core/String";
2 export function factorial(n) {
3     const loop = function (i, acc) {
4         loop: while (true) {
5             const $var1 = i === 0 ? [0] : i === 1 ? [0] : [1];
6
7             switch ($var1[0]) {
8                 case 0:
9                     return acc | 0;
10
11                 case 1:
12                     const $var2 = i - 1;
13                     acc = acc * i;
14                     i = $var2;
15                     continue loop;
16             }
17         }
18     };
19
20     return loop(n, 1) | 0;
21 }
22 replLog(printf("%i"))(factorial(10));
```

F# for Web


 **SAFE Stack**
safe @safe_stack Following

Hi all! We're looking forward to engage with [#fsharp](#) community to make web applications easier than ever over the coming weeks!


6:10 AM - 22 Sep 2017

11 Retweets 14 Likes 

 2  11  14 



 **Don Syme** @dsyme · 4h
Replying to @safe_stack
Link? ;)

 2   1 

 **SAFE Stack** @safe_stack · 4h
Whoops. safe-stack.github.io is a great place to start.

More F# for the Web and serverless



WebSharper.



F# and Docker containers



Kubernetes



Hashicorp Nomad



Apache Mesos



Azure Container Instances



Docker Swarm



Amazon EC2 Container Service

Endless opportunities...

Paket

Better dependency management

`paket.dependencies`

`paket.lock`

`paket.references`

`paket.template`

Expecto

- + Tests are first class values
- + More flexibility and leverage when writing tests
- + Parallel and async by default
- + Integrates with Ionide, and has a VS adapter
- + Works well with FsCheck

```
open Expecto
```

```
let tests =
```

```
  test "A simple test" {
```

```
    let subject = "Hello World"
```

```
    Expect.equal subject "Hello World" "The strings should equal"
```

```
  }
```

```
[<EntryPoint>]
```

```
let main args =
```

```
  runTestsWithArgs defaultConfig args tests
```

Expecto is 

“Type provider” F# projects

Data formats type providers

SQL type provider

R type provider

Swagger type provider



INTRODUCTION

F# Development

Ionide includes all the necessary features you'd find in a modern IDE - autocomplete, tooltips, document formatting, syntax and error highlighting, and many more.

```
Test.fs - test
1 module Test
2
3 []
4 string [] -> int
5 let main argv =
6     printfn "%A" argv
7     0 // return an integer
8
9 nullArg
```

All this is a result of
community work

What can each of us do to make F#
even better?

Solve issues and ask questions

- + Attempt to solve it yourself first
- + Contribute to documentation
- + Write answers on StackOverflow or Quora (even if it seems easy now)
- + Join FSSF and participate in F# mentorship program
- + Ask and discuss questions on Twitter [#fsharp](#)!
- + Start your project on Github! Look at ["up-for-grabs"](#) items.

Develop your ideas. Everybody has one

- + Don't ignore your ideas
- + Discuss it with F# Community
- + Tweet [#fsharp](#)
- + Join F# Slack
- + Bring FSSF Board's attention to it
- + You will always find advice and help!

F# Ninjas, share your experience

- + Live stream your F# coding, mentor beginners
- + Write blog posts and create videos to help others
- + Walk through your contributions to ecosystem & tools
- + Share knowledge about F# compiler
- + Experiment
- + Create next new revolutionary F# project, **continue to contribute!**

Live Streaming F#



Krzysztof Cieślak

@k_cieslak

Following



In around 30 min I'll try streaming some [#fsharp](#) / Fable / Ionide hacking -



k_cieslak - Twitch

F#, Fable and Ionide hacking - adding profiler to VSCode

[twitch.tv](https://www.twitch.tv/k_cieslak)

6:29 PM - 20 Jul 2017

7 Retweets 10 Likes



Maintaining an F# project?

- + Mark items for new contributors as “up-for-grabs”
- + Maintain your documentation up to date
- + Create examples on how to contribute
- + Welcome new ideas and appreciate new contributions

Companies that use F#

- + Publicly state the fact that you use F#
- + Share your F# success stories
- + Submit a testimonial <http://fsharp.org/testimonials>
- + Write blog posts on how F# helps you achieve more

You **will** attract more talent from the market!

Expand F# usage at your job

- + Create prototypes in F#
- + Do it gradually, use it with existing C# code
- + Teach your colleague F#
- + Demonstrate how powerful, quick, simple, concise and efficient F# is
- + Emphasize faster time-to-market of F# code
- + Clearly show your boss that you save time, money and support efforts

F# is the future of OSS .NET
because
we are making it so.