SIND SIE AGIL GENUG FÜR F#? - MIT F# DEN ENTWICKLUNGSPROZESS VERBESSERT

MATTHIAS DITTRICH, AIT GMBH

@MATTHI__D | GITHUB | AITGMBH.DE

WHO AM I?

- uses lots of languages

  (AutoIt, C#, C++, Java, Python, F#)
- AIT GmbH & Co KG. - consulting & development
- currently working on a message based architecture (Protbuf, AMQP)
- long time Gentoo Linux user
- maintainer of RazorEngine
- Paket, FAKE (5) and FSharp.Formatting
- Autor of lots of other projects

ROADMAP

- **What is F#**
- A quick introduction
- Does the language make a difference?
- What is doing F# differently?
- Adoption?
- Challenges?

WHAT IS F#

"F# is a mature, open source, cross-platform, functional-first programming language. It empowers users and organizations to tackle complex computing problems with simple, maintainable and robust code."

fsharp.org

WHAT IS F#

"F# was so easy to pick up we went from complete novices to having our code in production in less than a week."

Jack Mott from O'Connor's Online

fsharp.org/testimonials

WHAT IS F#

"Most successful projects I have written have all been in F#."

Colin Bull, talking about enterprise software

WHAT IS F#

Just another .NET language

WHAT IS F#

- Open Source
- Functional-First -> multi-paradigm
- Less error prone
- Expressive
- Less Work

WHAT IS F#

- Javascript - Fable
- Type providers (Meta programming) -> JSON (+ REST Apis, for example WorldBank), XML, PowerShell, Python, "R", SQL, Registry, WMI, FileSystem, HTML, Excel, CSV
- Quotations (Meta programming)
- Computation Expressions (Meta programming) -> async, sequence, cloud, asyncSeq or your own

ROADMAP

- What is F#
- **A quick introduction**
- Does the language make a difference?
- What is doing F# differently?
- Adoption?
- Challenges?

F# IS C# ON STEROIDS

```
1: var t = 5;
```

```
1: let u = 5
```

- No semi colons required
- u is immutable by default (a symbol no variable)

F# IS C# ON STEROIDS

```
1: public int Add(int a, int b) {
2:     int c = a + 1;
3:     return c + b;
4: }
```

```
1: let add a b =
2:     let c = a + 1
3:     c + b
```

- No return keyword, last value is returned.
- No types needed, compiler will figure it out.
- No start- and end-brace is needed, whitespace counts.

F# IS C# ON STEROIDS

```
1: public class Person {
2:     public string Name { get; } // C#7
3:     public Person(string name) { Name = name; }
4: }
```

```
1: type Person (name:string) =
2:     member x.Name = name
```

- No start- and end-brace is needed, whitespace counts.
- Single constructor by default (helps you design better classes)
- Constructor parameters are private fields out of the box

F# IS C# ON STEROIDS

```
1: public interface IPerson {
2:     public string Name { get; }
3: }
```

```
1: type IPerson =
2:     abstract Name : string
```

- Interface = Type without implementations and without constructor
- You can do everything you would expect: abstract classes, namespaces, public, private, internal, ...

MATCH = SWITCH ON STEROIDS

```
 1: var a = o as A;
 2: if (a != null)
 3: {
 4:     //...
 5: }
 6: var b = o as B;
 7: if (b != null && someCondition)
 8: {
 9:     // ...
10: }
```

```
1: match o with
2: | :? A as a -> //...
3: | :? B as b when someCondition -> //...
```

CONSISTENT SYNTAX

```
try {
  // ...
}
catch (AException a) {
  // ...
}
catch (BException b) {
  if (!someCondition) throw;
  // ...
}
```

```
try
  // ...
with
| :? AException as a -> //...
| :? BException when someCondition -> //...
```

RECORDS

```
1: public class Person {
2:     public string Name { get; }
3:     public string Address { get; }
4:     public Person(string name, string address) {
5:         Name = name; Address = address } }
```

```
1: type Person =
2:     { Name : string; Address : string option }
3:
4: let createPerson name = { Name = name; Address = None }
5: let printPerson p =
6:     match p with
7:     | { Name = "Lars" } -> "the boss"
8:     | _ -> p.Name
```

- Equality members.
- Pattern matching.
- Immutable by default.

ROADMAP

- What is F#
- A quick introduction
- **Does the language make a difference?**
- What is doing F# differently?
- Adoption?
- Challenges?

DOES THE LANGUAGE MAKE A DIFFERENCE?

- The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities.
- The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.

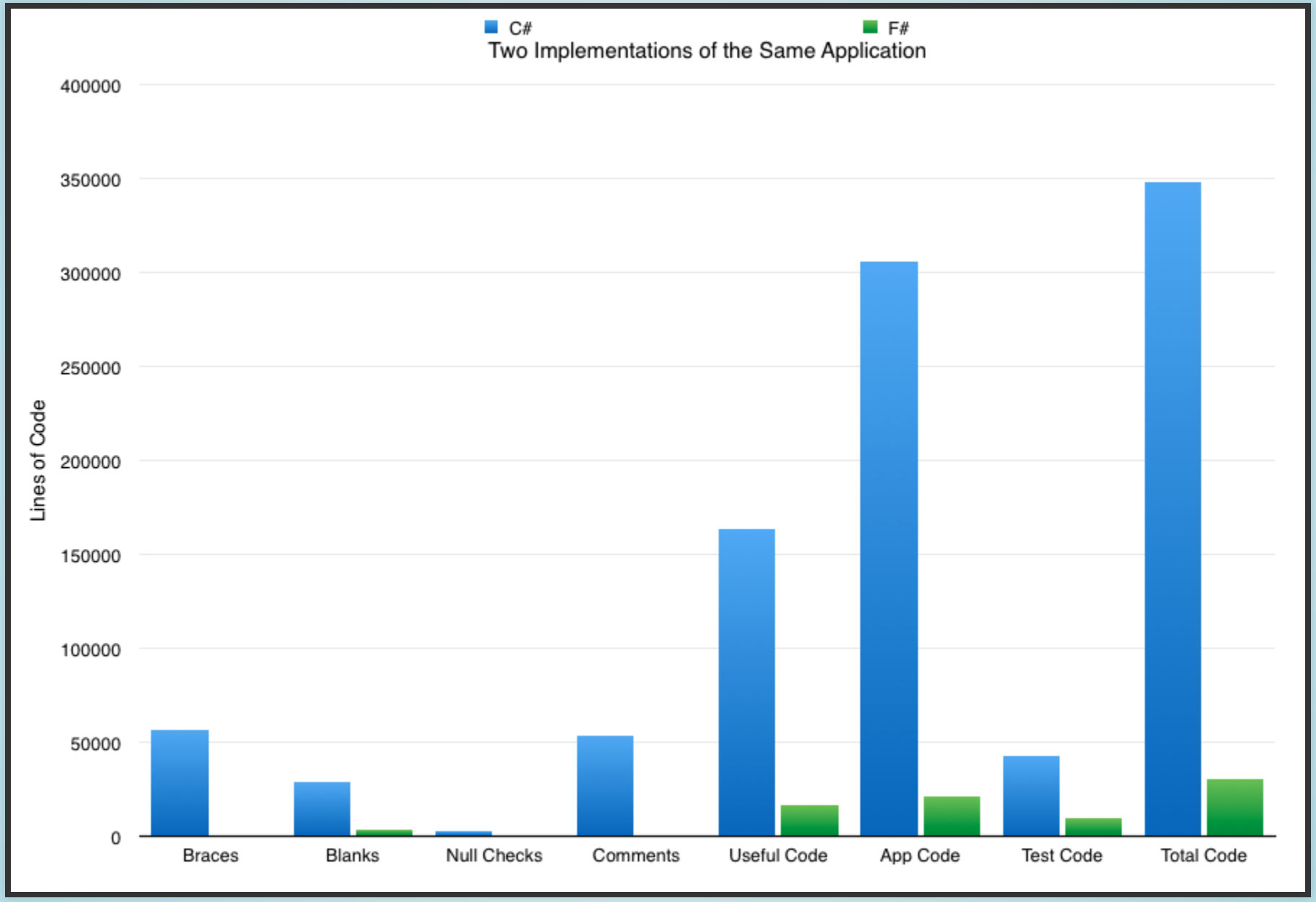
Edsger W. Dijkstra

DOES F# MAKE A DIFFERENCE?

What do people say about C# and F#?

- Real enterprise system
- Two different teams
- Same set of contracts, complex contracts
- Analysis afterwards

DOES F# MAKE A DIFFERENCE?



DOES F# MAKE A DIFFERENCE?

Implementation	C#	F#
Braces	56,929	643
Blanks	29,080	3,630
Null Checks	3,011	15
Comments	53,270	487
Useful Code	163,276	16,667
App Code	305,566	21,442
Test Code	42,864	9,359
Total Code	348,430	30,801

DOES F# MAKE A DIFFERENCE?

- The C# project took five years and peaked at ~8 devs. It never fully implemented all of the contracts.
- The F# project took less than a year and peaked at three devs (only one had prior experience with F#). All of the contracts were fully implemented.

F# makes a difference

ROADMAP

- What is F#
- A quick introduction
- Does the language make a difference?
- **What is doing F# differently?**
- Adoption?
- Challenges?

WHAT IS DOING F# DIFFERENTLY?

- Layout, Readability
- Naming
- Understandable and Expressive
- Abstractions

LAYOUT: WHITESPACE

```
1: public int Method() {
2:     return 3;
3: }
```

Or

```
1: public int Method ()
2: {
3:     return 3;
4: }
```

Two competing rules in C-like languages

LAYOUT: WHITESPACE

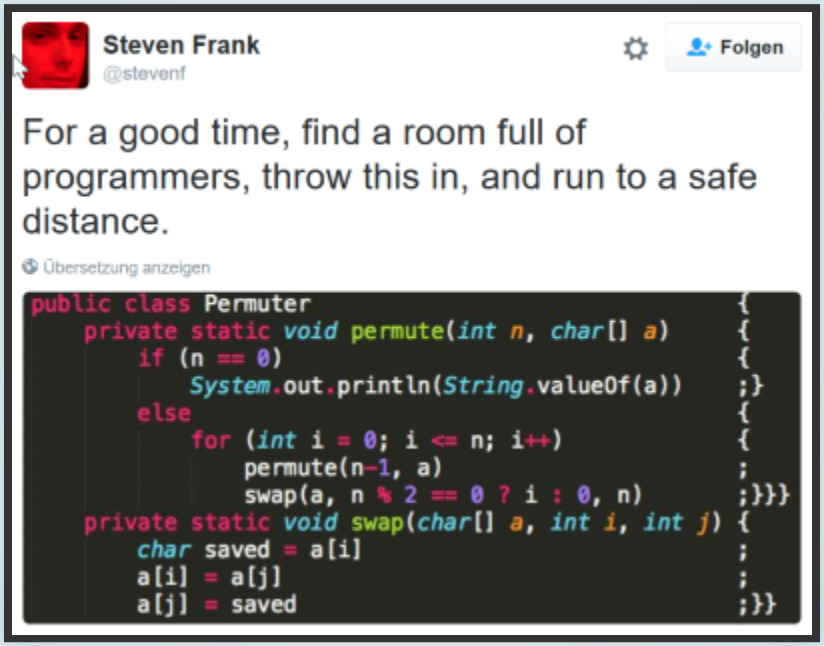
There should be one - and preferable only one - obvious way to do it.

the Zen of Python

LAYOUT: WHITESPACE

Language	Compiler	Human
C#	{}	{} and whitespace
F#	whitespace	whitespace

LAYOUT: WHITESPACE



Steven Frank
@stevenf

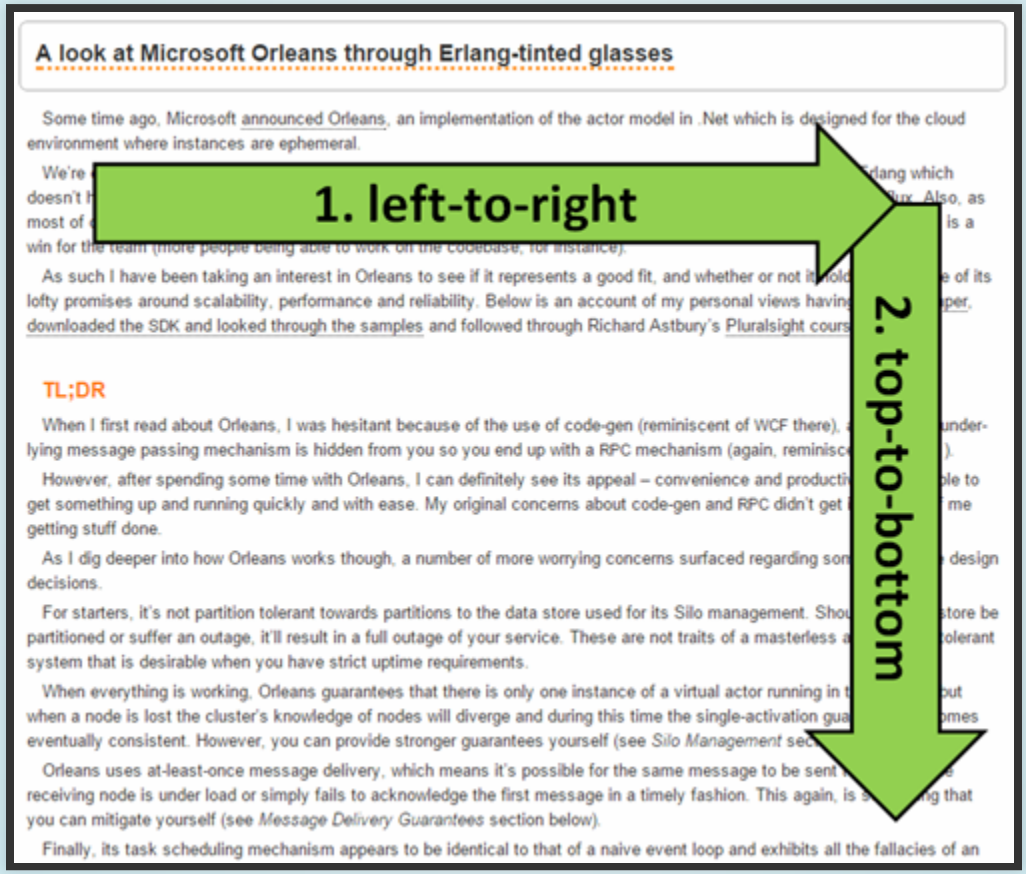
Folgen

For a good time, find a room full of programmers, throw this in, and run to a safe distance.

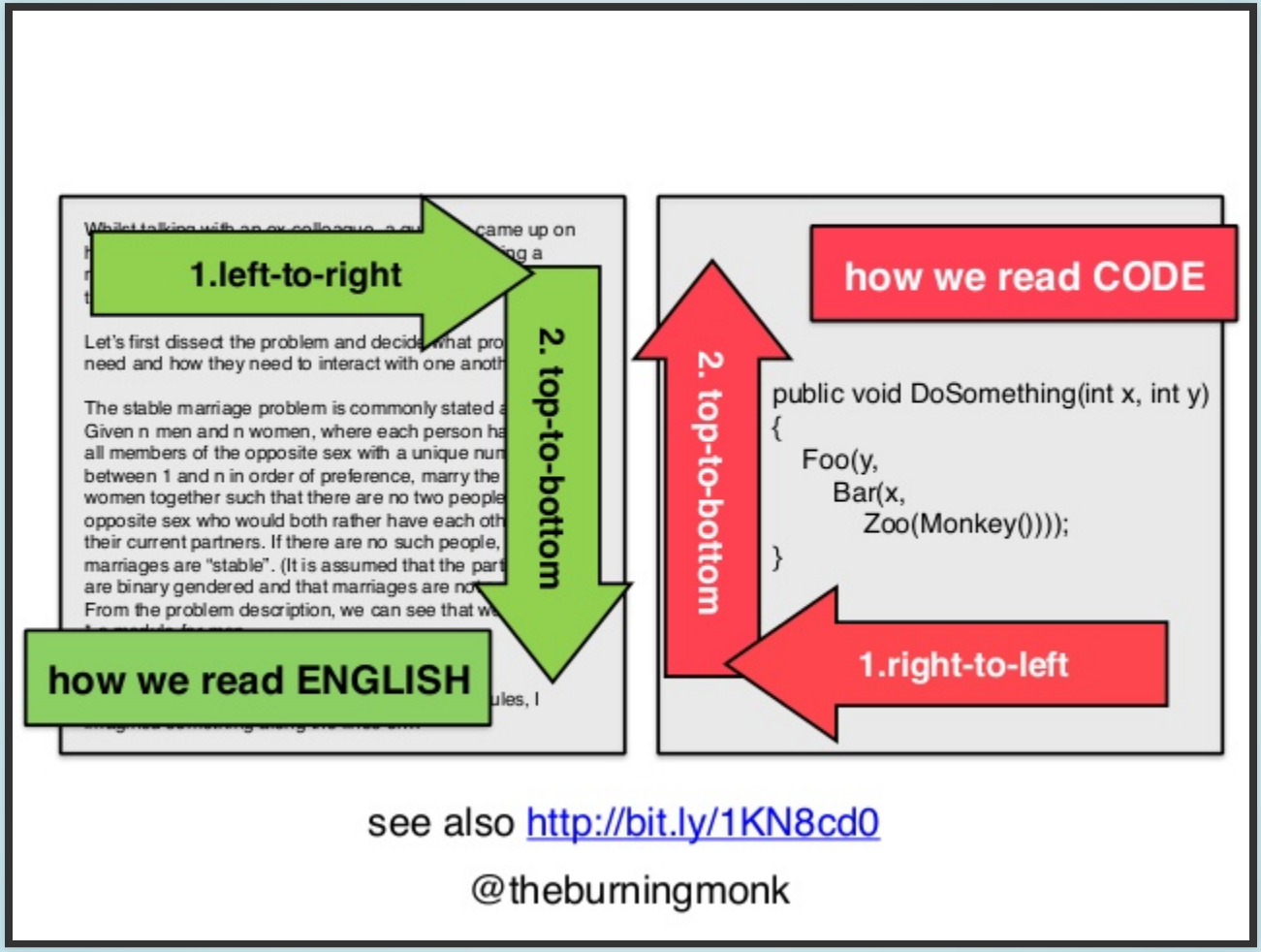
Übersetzung anzeigen

```
public class Permuter                                   {
    private static void permute(int n, char[] a)        {
        if (n == 0)                                     {
            System.out.println(String.valueOf(a))       ;}
        else                                            {
            for (int i = 0; i <= n; i++)                {
                permute(n-1, a)                         ;
                swap(a, n % 2 == 0 ? i : 0, n)          ;}}}
    private static void swap(char[] a, int i, int j) {
        char saved = a[i]                               ;
        a[i] = a[j]                                     ;
        a[j] = saved                                    ;}}
```

LAYOUT: STRUCTURE



LAYOUT: STRUCTURE



LAYOUT: STRUCTURE

```
1: let doSomething x y =
2:     Monkey()
3:     |> zoo
4:     |> bar x
5:     |> foo y
```

NAMING

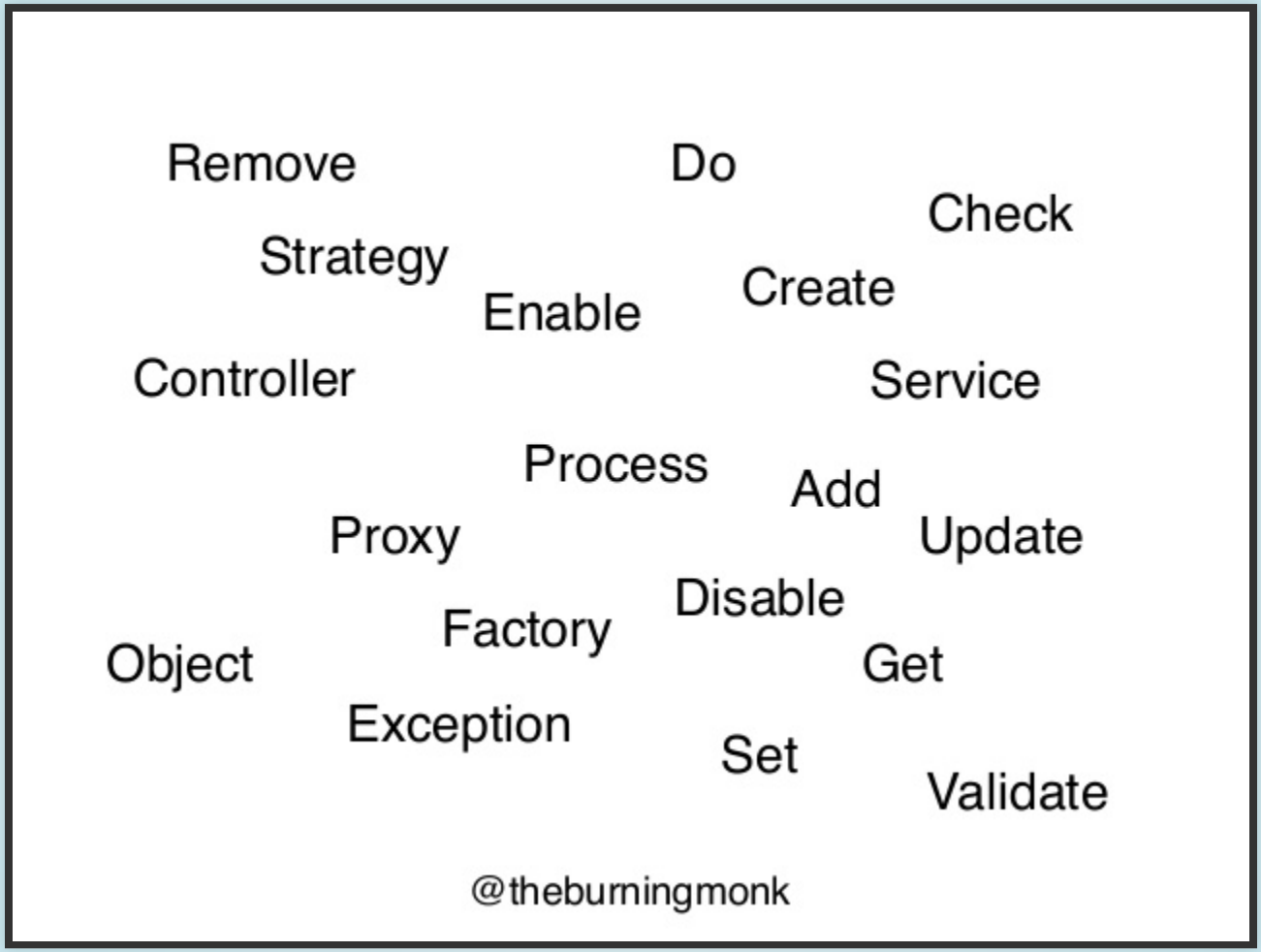
"There are only two hard things in Computer Science: cache invalidation and naming things."

Phil Karlton

"Names are the one and only tool you have to explain what a variable does in every place it appears, without having to scatter comments everywhere."

Mike Mahemoff

LEGO NAMING



methodnamer.com

NAMING: HIGHER ORDER FUNCTIONS

```
1: words
2: |> Array.map (fun x -> x.Count)
3: |> Array.reduce (+)
```

- Smaller scopes
- Shorter names (When x, y, z are great variable names)
- Fewer things to name

NAMING: SHORT NAMES

"The length of a name should be related to the length of the scope. You can use very short variable names for tiny scopes, but for big scopes you should use longer names."

***Clean Code**: Robert C. Martin*

NAMING: SMALLER THAN OBJECT?

```
1: public interface IConditionChecker
2: {
3:     bool CheckCondition();
4: }
```

```
1: public interface ICondition
2: {
3:     bool IsTrue();
4: }
```

NAMING: SMALLER THAN OBJECT?

```
1: type Condition = unit -> bool
```

```
1: using Condition = System.Func<System.Boolean>;
```

No abstraction is too small.

NAMING: OBJECT EXPRESSIONS

```
1: enterpriseCrew.OrderBy(
2:     (fun c -> c.Current),
3:     { new IComparer<Occupation> with
4:         member __.Compare(x, y) =
5:             x.Position.CompareTo(y.Position) })
```

- No need to define a class, no need to name it.
- `_` and `__` used to explicitly give something no name.

EXPRESSIVENESS: "EVERYTHING" IS AN EXPRESSION

```
1: var variable = null;
2: try {
3:     variable = TrySomethingRisky()
4: } catch (AException) {
5:     variable = fallback1;
6: } catch (BException) {
7:     variable = fallback2;
8: }
9: return DoSomething(variable);
```

```
1: let symbol =
2:     try
3:         TrySomethingRisky()
4:     with
5:       | :? AException -> fallback1
6:       | :? BException -> fallback2
7: DoSomething(symbol)
```

EXPRESSIVENESS: "EVERYTHING" IS AN EXPRESSION

```
1: let symbol =
2:     try
3:         TrySomethingRisky()
4:     with
5:       | :? AException -> fallback1
6:       | :? BException -> fallback2
7: DoSomething(symbol)
```

```
1: try
2:     TrySomethingRisky()
3: with
4:   | :? AException -> fallback1
5:   | :? BException -> fallback2
6: |> DoSomething
```

- No need to name the thing.

EXPRESSIVENESS: "EVERYTHING" IS AN EXPRESSION

```
1: var variable = condition ? Value1 : fallback;
```

```
1: let variable = if condition then Value1 else fallback
```

SINGLE PASS COMPILER

DEMO

EXPRESSIVENESS: TYPE PROVIDER

DEMO

EXPRESSIVENESS: COMPUTATION EXPRESSION

```
1: async {
2:     let! results =
3:         [ "http://www.mbrace.io/"
4:           "http://www.nessos.gr/" ]
5:         |> List.map downloadAsync
6:         |> Async.Parallel
7:
8:     return results |> Array.sumBy(fun r -> r.Length)
9: }
```

Like async/await in C# but the concept behind is more powerful.

EXPRESSIVENESS: COMPUTATION EXPRESSION

```
 1: let downloadCloud url = downloadAsync url |> Cloud.OfAsync
 2:
 3: cloud {
 4:     let! results =
 5:         [ "http://www.mbrace.io/"
 6:           "http://www.nessos.gr/" ]
 7:         |> List.map downloadCloud
 8:         |> Cloud.Parallel
 9:
10:     return results |> Array.sumBy(fun r -> r.Length)
11: }
```

mbrace

ABSTRACTIONS

"Your abstractions should afford right behaviour, whilst make it impossible to do the wrong thing."

"Make illegal states unrepresentable."

ABSTRACTIONS: MAKE ILLEGAL STATES UNREPRESENTABLE

Business rules:

- A contact has a name.
- A contact has an address.
  - an email address
  - a postal address
  - both

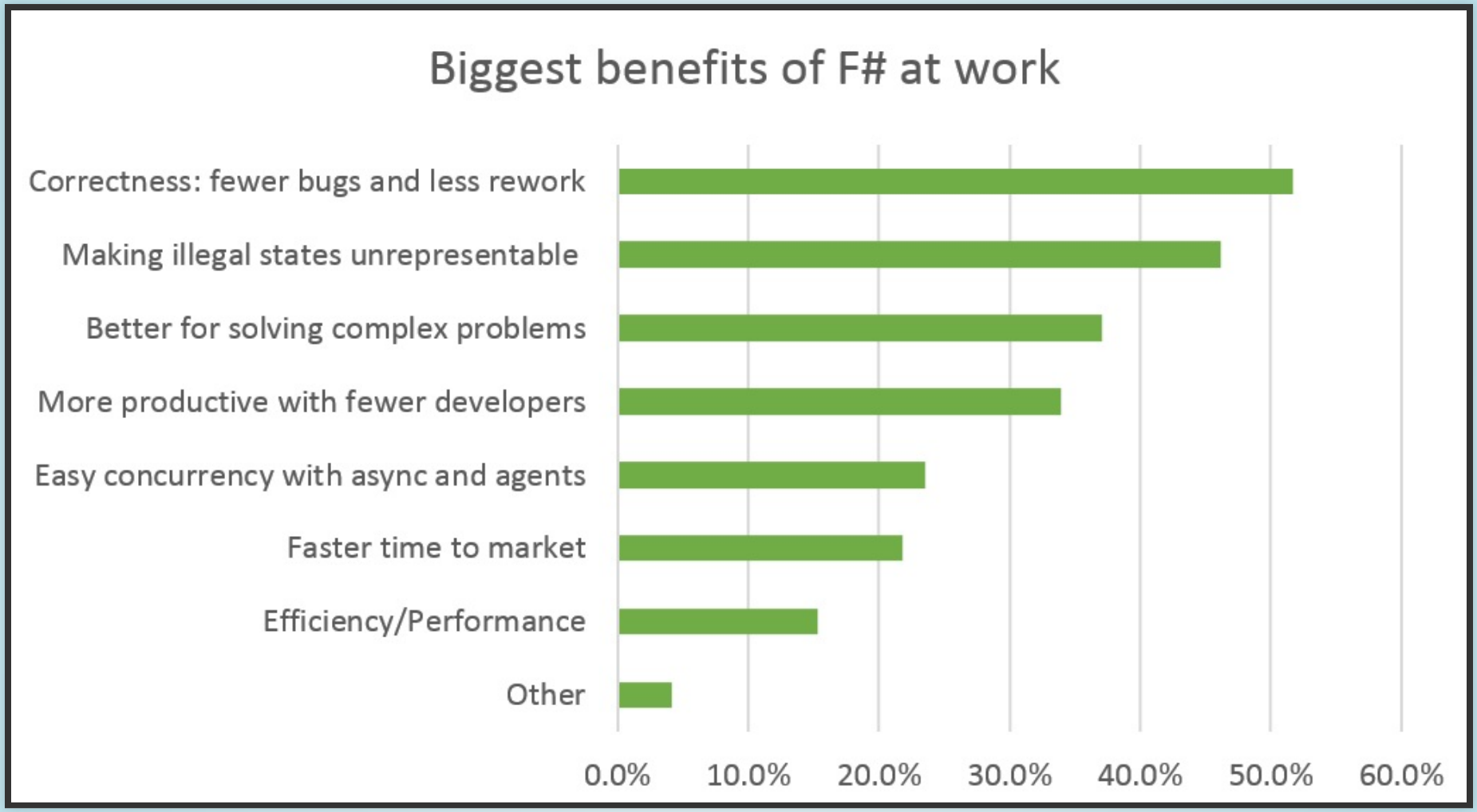
EIN VORSCHLAG IN C#

```
 1: public class Contact {
 2:   public string Name { get; }
 3:   public EmailAddress EMail { get; }
 4:   public PostalAddress Address { get; }
 5:   public Contact(string name, EmailAddress email,
 6:       PostalAddress address) {
 7:     if (email == null && address == null)
 8:       throw new ArgumentException("Invalid!");
 9:     Name = name;
10:     EMail = email;
11:     Address = address;
12: }
```

ABSTRACTIONS: MAKE ILLEGAL STATES UNREPRESENTABLE

```
1:  // Discriminated union
2:  type ContactInfo =
3:      | EmailOnly of EmailAddress
4:      | PostOnly of PostalAddress
5:      | EmailAndPost of EmailAddress * PostalAddress
6:
7:  type Contact = { Name : string; ContactInfo : ContactInfo }
8:
9:  // pattern match -> switch on steriods
10: let printContact contact =
11:     match contact with
12:     | EmailAndPost (email, _)
13:     | EmailOnly email -> sprintf "EmailContact: %A" email
14:     | PostOnly post -> sprintf "Post-Contact: %A" post
```

WHAT IS DOING F# DIFFERENTLY?



F# Annual Survey 2016

ROADMAP

- What is F#
- A quick introduction
- Does the language make a difference?
- What is doing F# differently?
- **Adoption?**
- Challenges?

ADOPTION?

Low risk

- Runs on CLR, mono and the new "dotnetcore" + Javascript
- Open Source
- Good Interop
- Back-out to C# / Javascript

@simontcousins

ADOPTION?

- Self taught
- Hire .NET developers, not language X
- Production code in a week
- Functional programmer in a month

@simontcousins

ADOPTION?

- Baby steps: Don't try to introduce a new language and a new paradigm
- Language first: Explicit interfaces, Syntax, Records vs Classes vs Modules
- Paradigm second: Let the language guide you

Colin Bull

ADOPTION?

- Start with Build
- msbuild is not fun
- FAKE

Colin Bull

ROADMAP

- What is F#
- A quick introduction
- Does the language make a difference?
- What is doing F# differently?
- Adoption?
- **Challenges?**

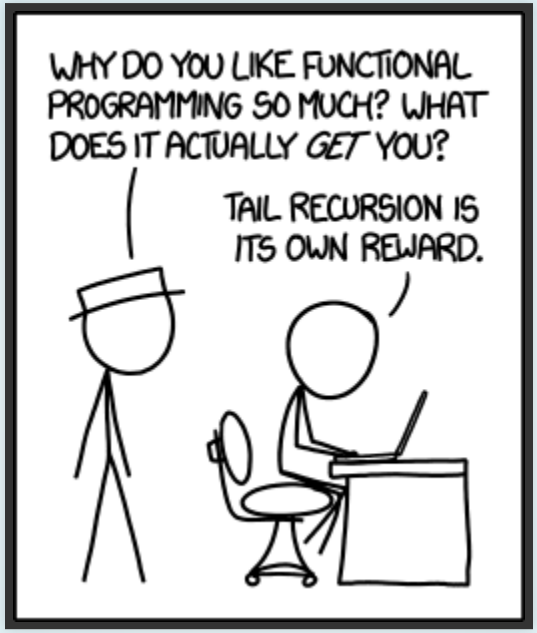
CHALLENGES?

- Designer Support
- After C#
- Before C#

CHALLENGES?

- People are too puritanical about purity.
  - be pragmatic and solution focused
  - functional purity not for the sake of itself

CHALLENGES?



XKCD

CHALLENGES?

- Explicit is better than implicit.
- Simple is better than Complex. Complex is better than Complicated.
- Special cases aren't special enough to break the rules. Although practicality beats purity.
- If the implementation is hard to explain, it's a bad idea.

the Zen of Python

```
let memoize f =
  let cache = System.Collections.Generic.Dictionary<_, _>()
  fun x ->
    if cache.ContainsKey(x) then cache.[x]
    else let res = f x
         cache.[x] <- res
         res

let somePureLongRunningFunc i =
  System.Threading.Thread.Sleep (2000)
  i + 1

let fastFunc = memoize somePureLongRunningFunc


```

CHALLENGE YOURSELF!

- "Practice does not make perfect. Only perfect practice makes perfect"
- "Perfection is not attainable. But if we chase perfection, we can catch excellence"

Vince Lombardi

CHALLENGE YOURSELF!

- "Programming languages have a devious influence: They shape our thinking habits."

Edsger W. Dijkstra

- "One of the most disastrous thing we can learn is the first programming language, even if it's a good programming language."

Alan Kay

THANKS (AND THINGS TO READ/WATCH IN DEPTH)

- 7 ineffective coding habits many F# programmers don't have
- Does the language make a difference
- FSReveal (but really the whole F# community)
- Real world functional programming
- Luca Bolognese
- Enterprise F#
- Folien

BONUS: FAKE

DEMO (these slides)

BONUS: PAKET

Root folder: paket.lock, paket.dependencies

paket.dependencies - Notepad

File Edit Format View Help

```
source https://www.nuget.org/api/v2

nuget Castle.Core
nuget Castle.Windsor
nuget EntityFramework
nuget LinqKit framework: >= net45
nuget log4net 2.0.0
nuget Microsoft.Bcl
```

paket.lock - Notepad

File Edit Format View Help

```
NUGET
  remote: https://nuget.org/api/v2
  specs:
    Castle.Core (3.3.3)
    Castle.Windsor (3.3.0)
      Castle.Core (>= 3.3.0)
    EntityFramework (6.1.3)
    LinqKit (1.1.2) - framework: >= net45
      EntityFramework (>= 6.0.2)
    log4net (2.0.0)
    Microsoft.Bcl (1.1.10)
```

Every project folder has own: paket.references

paket.references - Notepad

File Edit Format View Help

```
EntityFramework
```

paket.references - Notepad

File Edit Format View Help

```
Castle.Core
Castle.Windsor
EntityFramework
LinqKit
Newtonsoft.Json
```

paket.references - Notepad

File Edit Format View Help

```
Castle.Core
Castle.Windsor
Microsoft.Bcl.Async
Microsoft.Bcl
```

BONUS: PAKET

- Not a simple NuGet.exe replacement
- Package via csproj/fsproj (no nuspec needed)
- GitHub and Http (file based) dependencies.
- Caching
- Git dependencies (repository -> temporary replace nuget dependency)
- Support for most stuff (Roslyn Analyzers, Content files, ...)
- Doesn't support PowerShell scripts (by design)