



RX - Reactive Extensions

Library for asynchronous & event-based
programming

Jonas Bandi

<http://blog.jonasbandi.net>
jb@jonasbandi.net
twitter: @jbandi



DNUG Bern Sponsoren



About me



Jonas Bandi
At TechTalk since 2009
Committer to SpecFlow
<http://blog.jonasbandi.net>
Twitter: @jbandi

TechTalk is a software development and consulting company with ~60 people located in Vienna, Budapest and Zürich. We focus on Scrum and .NET.



Hinweis: Verwendung für eigene Präsentationen nur mit Einverständnis des Autors

The audience?

- Who knows RX?
- Who is working with JavaScript
- Who is working with asynchronous systems?
- Who is familiar with functional programming concepts?



Agenda

- Introduction to Reactive Programming
- Concepts in .NET 4
- Reactive Extensions Framework
- Demos

Let's discuss!

**"We're still figuring
this stuff out.
All of us."
- Jay Fields**



Thoughts on developer Testing:

<http://blog.jayfields.com/2009/02/thoughts-on-developer-testing.html>

Reactive Programming?

Year 8084:



C# 7.3:

```
var a = 10;  
var b = a + 1;  
a = 11;  
b = a + 1;
```

P##: destiny operator

```
var a = 10;  
var b <= a + 1;  
a = 20;  
Assert.AreEqual(21, b);
```

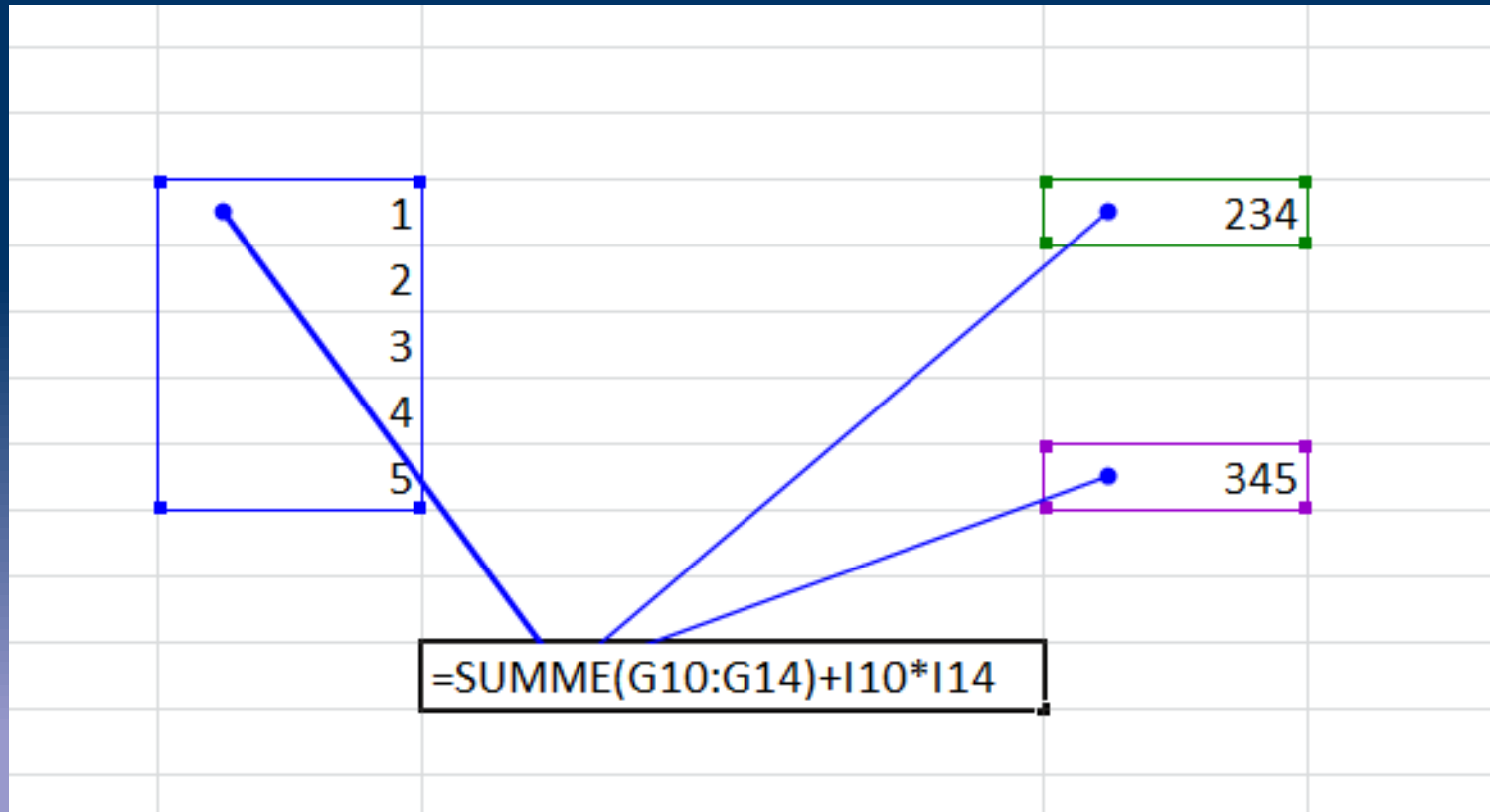
Reactive Programming

Reactive Programming is a programming paradigm oriented around data flows and the propagation of change.

- Wikipedia

Reactive Programming?

- But Excel does just that ...



Ok ... nothing new here ...

JavaScript:

```
var a = 1;  
var b = function(){return a + 1};  
a = 2;  
alert(b());
```

C# 3.5:

```
var a = 1;  
Func<int> b = () => a + 1;  
a = 2;  
Assert.AreEqual(3, b());
```

The observer pattern

Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

Design Patterns

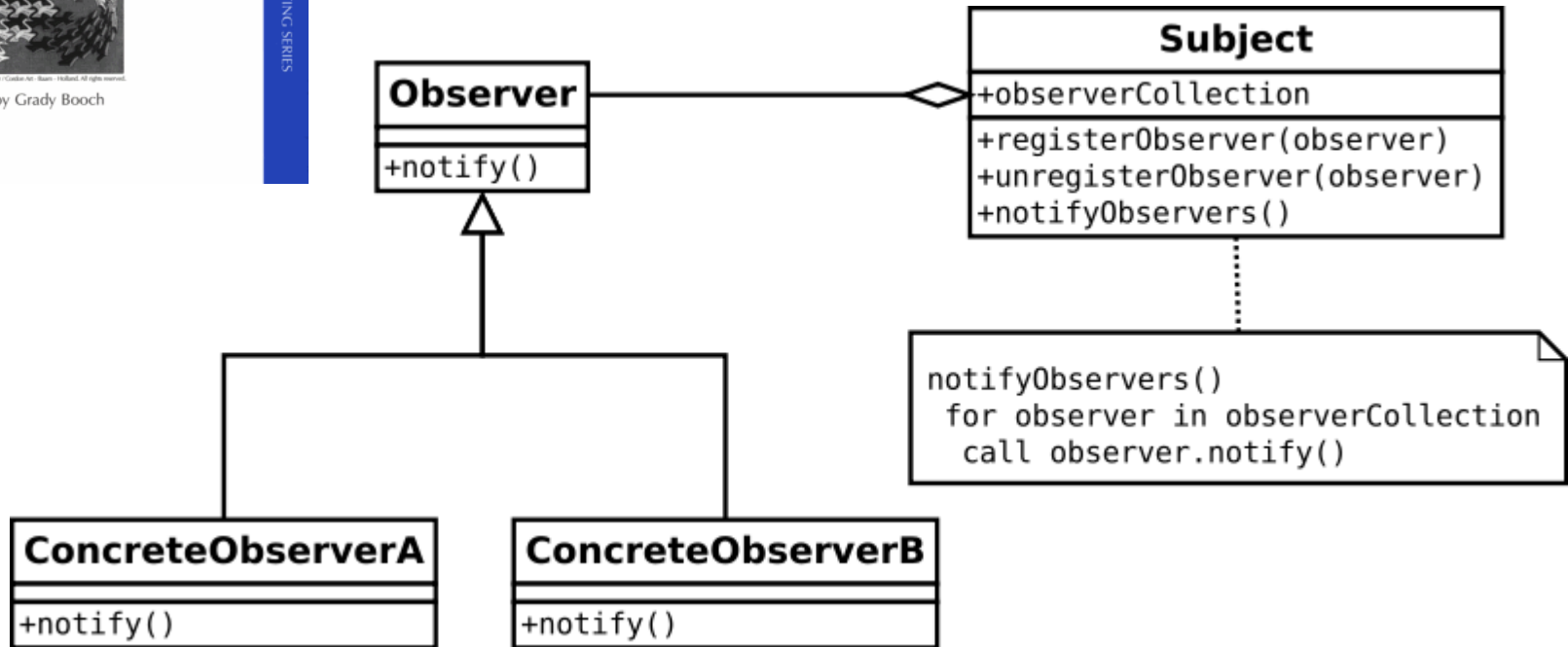
Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES



From Patterns to Language Constructs

- Events & Delegates
- Anonymous Methods
- Lambdas

- For many use-cases the Observer Pattern becomes obsolete
 - Still important for communication

.NET Events

```
// Define a delegate named LogHandler, which will encapsulate
// any method that takes a string as the parameter and returns no value
public delegate void LogHandler(string message);

// Define an Event based on the above Delegate
public event LogHandler Log;
```

```
// By Default, create an OnXXXX Method, to call the Event
protected void OnLog(string message)
{
    if (Log != null)
    {
        Log(message);
    }
}
```

```
// Subscribe the Functions Logger and fl.Logger
myClass.Log += new MyClass.LogHandler(Logger);
myClass.Log += new MyClass.LogHandler(fl.Logger);
```

Problems with .NET Events

- Attach/Detach -
 - Detach has to be called manually
- Not composable
 - Result: complicated ad-hoc statemachines
 - Example: Drag & Drop
- No concept for synchronization
- Difficult to test

.NET events are not first class language constructs ...

How to pass around?

Hidden data source

```
form1.MouseMove += (sender, args) => {  
    if (args.Location.X == args.Location.Y)  
        // I'd like to raise another event  
};
```

Lack of composition

```
form1.MouseMove -= /* what goes here? */
```

Resource maintenance?

Erik Meijer



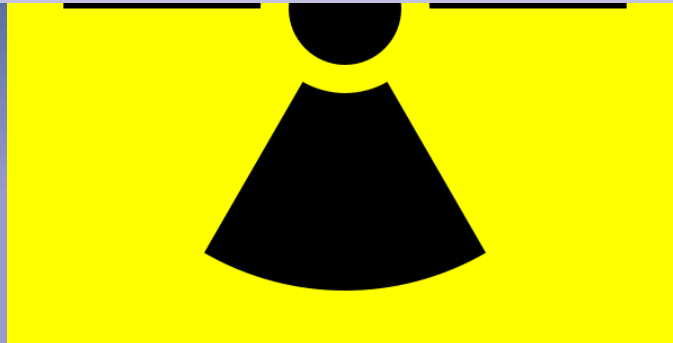
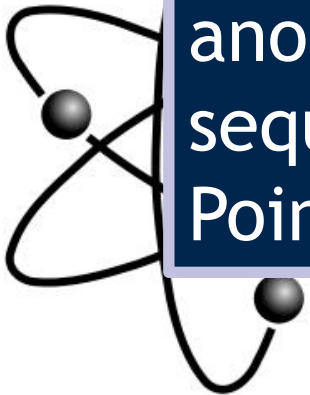
- Former University Professor
 - Functional programming
 - Haskell
- Head of *Cloud Programmability Team* at Microsoft
- „Inventor“ of LINQ
- „Inventor“ of RX

Events are collections...

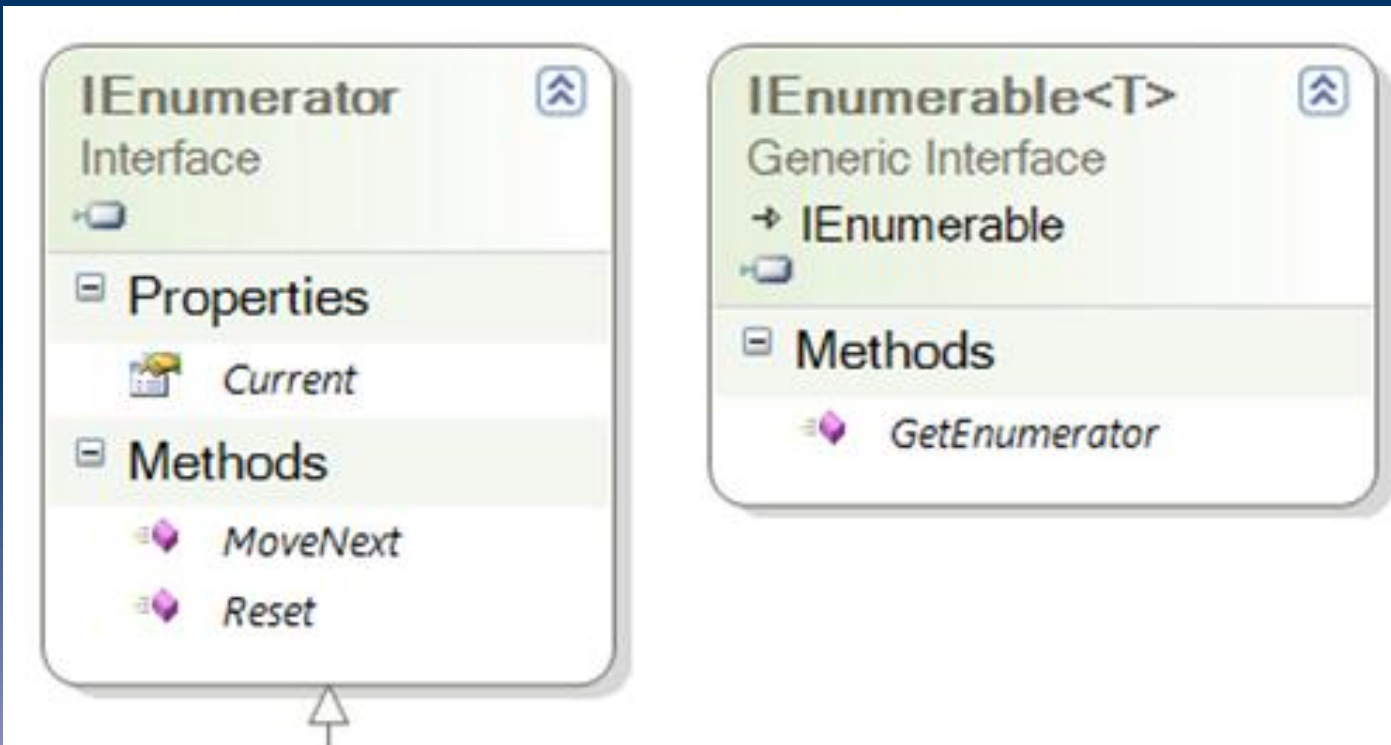
Different perspectives on the same thing..

Did you ever regard the `MouseMove` event as a collection of `Point` values?

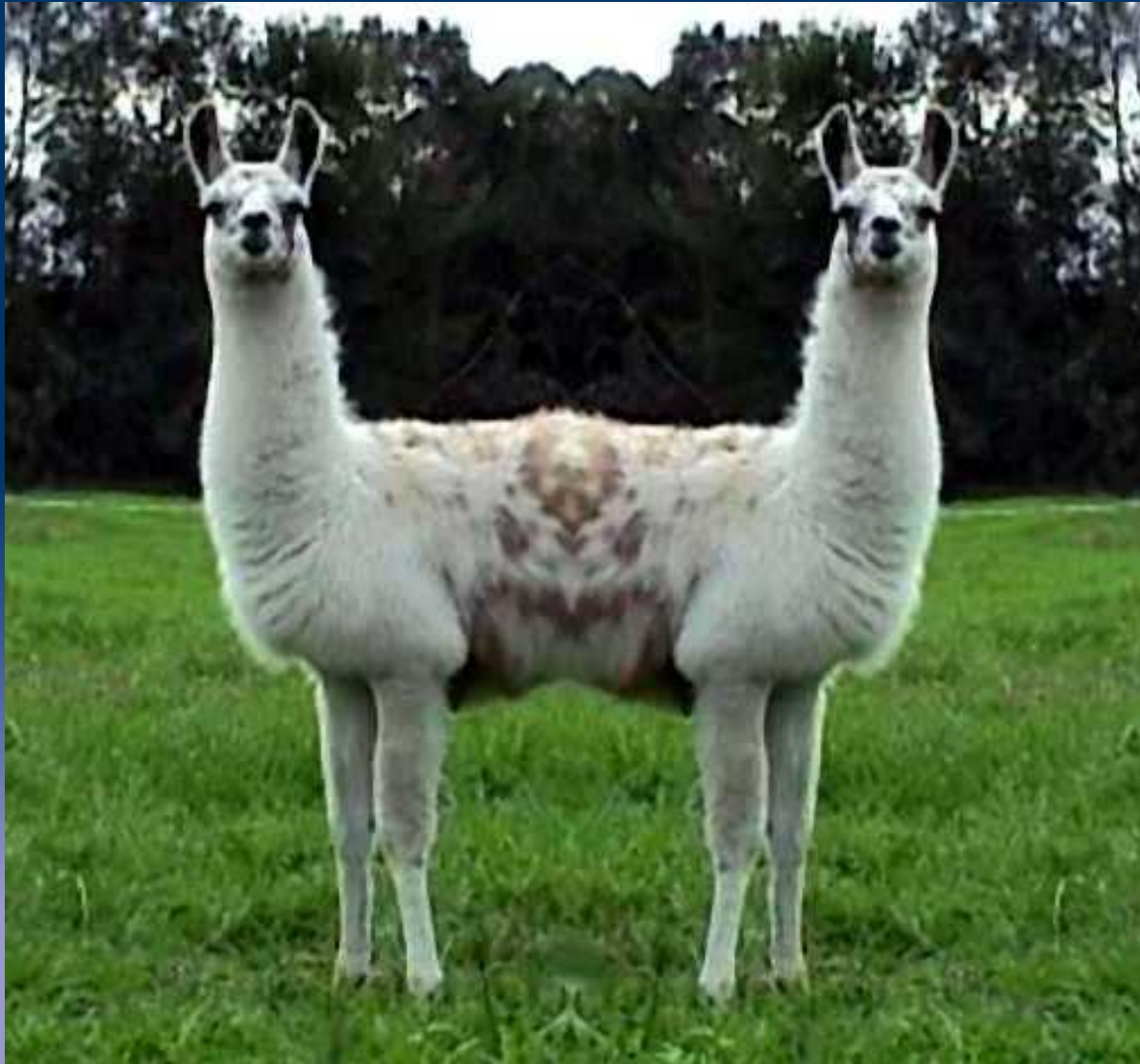
In the world of Rx, we see events as just another concrete form of observable sequences: your mouse is a database of `Point` values!



IEnumerator & IEnumerable



Applying Duality ...






RX: Interfaces


IObserver<T> 
Generic Interface


 **Methods**

-  *OnCompleted*
-  *OnError*
-  *OnNext*

IObservable<T> 
Generic Interface


 **Methods**

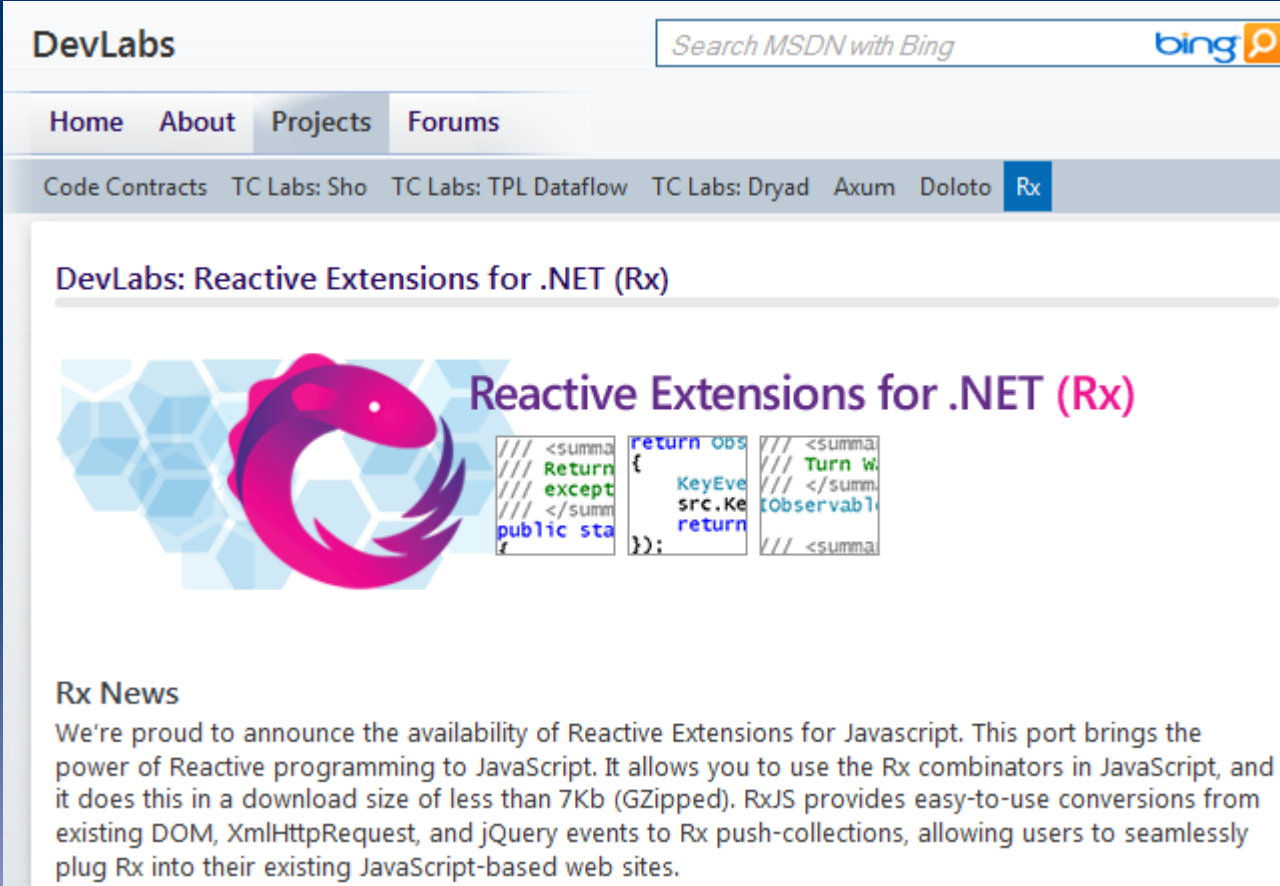
-  *Subscribe*

Part of .NET 4.0 BCL

That's it ... Questions?



DevLabs: Reactive Extensions



The screenshot shows the DevLabs website interface. At the top left is the "DevLabs" logo. To its right is a search bar with the text "Search MSDN with Bing" and the Bing logo. Below the search bar is a navigation menu with links for "Home", "About", "Projects", and "Forums". A secondary navigation bar contains links for "Code Contracts", "TC Labs: Sho", "TC Labs: TPL Dataflow", "TC Labs: Dryad", "Axum", "Doloto", and "Rx", with "Rx" highlighted in a blue box. The main content area features the heading "DevLabs: Reactive Extensions for .NET (Rx)". Below this heading is a large graphic with a blue hexagonal background and a pink circular logo. To the right of the logo is the text "Reactive Extensions for .NET (Rx)". Below the graphic is a code snippet showing C# code for a method that returns an Observable. The code is as follows:

```
/// <summary>  
/// Return  
/// except  
/// </summary>  
public sta  
f  
return Obs  
{  
    KeyEve  
    src.Ke  
    return  
};
```

Below the code snippet is the heading "Rx News" and a paragraph of text:

We're proud to announce the availability of Reactive Extensions for Javascript. This port brings the power of Reactive programming to JavaScript. It allows you to use the Rx combinators in JavaScript, and it does this in a download size of less than 7Kb (GZipped). RxJS provides easy-to-use conversions from existing DOM, XMLHttpRequest, and jQuery events to Rx push-collections, allowing users to seamlessly plug Rx into their existing JavaScript-based web sites.

<http://msdn.microsoft.com/en-us/devlabs>

Using the RX framework

- .NET 3.5 & 4, Silverlight 3 & 4, WindowsPhone
 - 3 Dlls: System.Reactive, System.Interactive, System.CoreEx
- JavaScript
 - Libraries: rx.js (30kb), rx.jquery.js ...

Creating Observable Collections

Demo 1

Observable collections are first class language constructs ...

Objects can be passed

Source of Point values

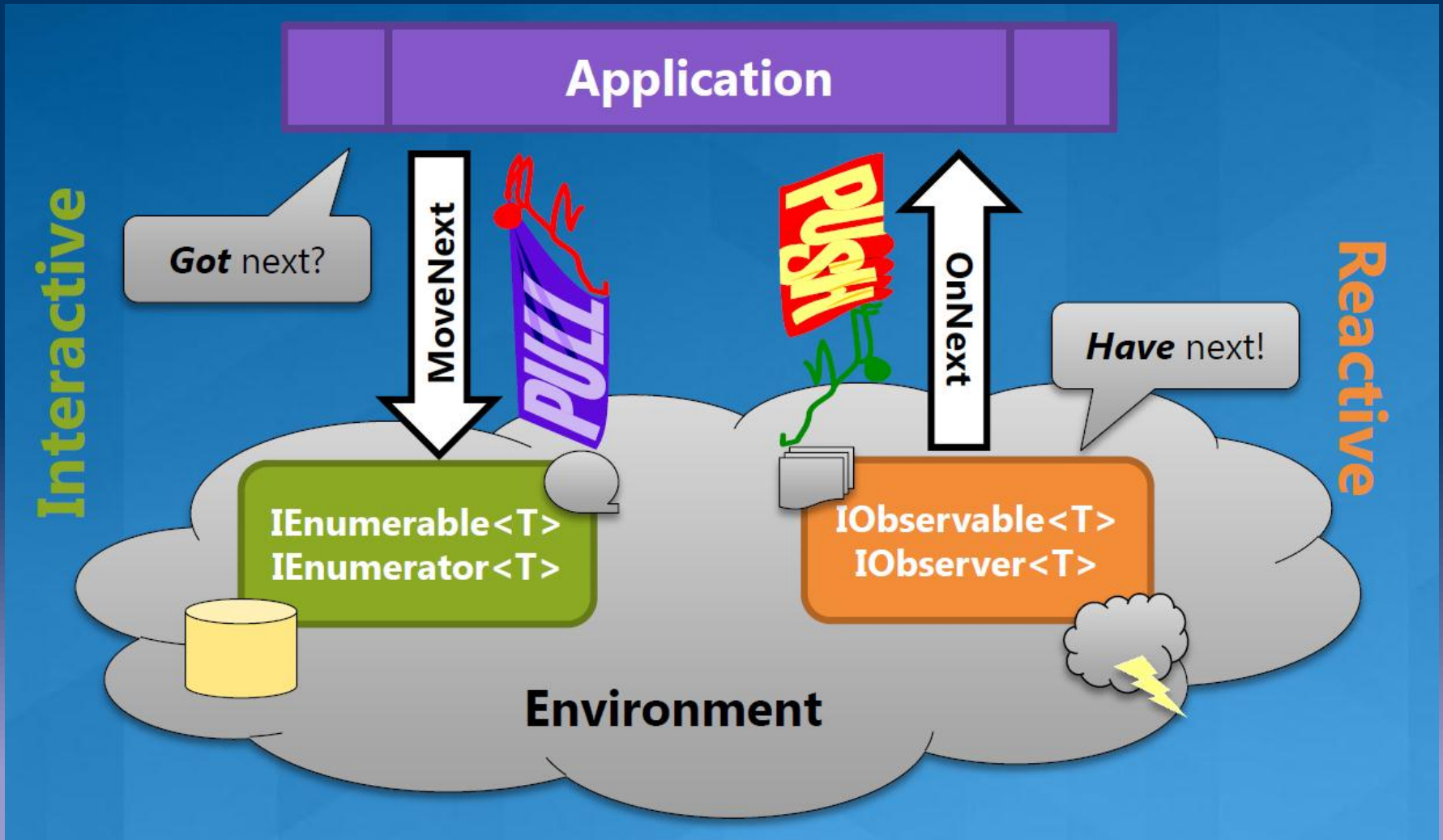
```
IObservable<Point> mouseMoves =  
    Observable.FromEvent(frm, "MouseMove");  
var filtered = mouseMoves  
    .where(pos => pos.X == pos.Y);
```

Can define operators

```
var subscription = filtered.Subscribe(...);  
subscription.Dispose();
```

Resource maintenance!

Pull vs. Push



Convert between both worlds

- Convert between both worlds

```
// Introduces concurrency to enumerate and signal...  
var xs = Enumerable.Range(0, 10).ToObservable();
```

```
// Removes concurrency by observing and yielding...  
var ys = Observable.Range(0, 10).ToEnumerable();
```

Cold Observables vs. Hot Observables

Cold observables

```
var xs = Observable.Return(42);
```

Triggered by subscription

```
xs.Subscribe(Console.WriteLine); // Prints 42  
xs.Subscribe(Console.WriteLine); // Prints 42 again
```

Hot observables

```
var mme = Observable.FromEvent<MouseEventArgs>  
    (from, "MouseMove");
```

Mouse events going
before subscription

```
mme.Subscribe(Console.WriteLine);
```



- Create Observables from Events
- Basic Query Operators
- Composition

Demo 2

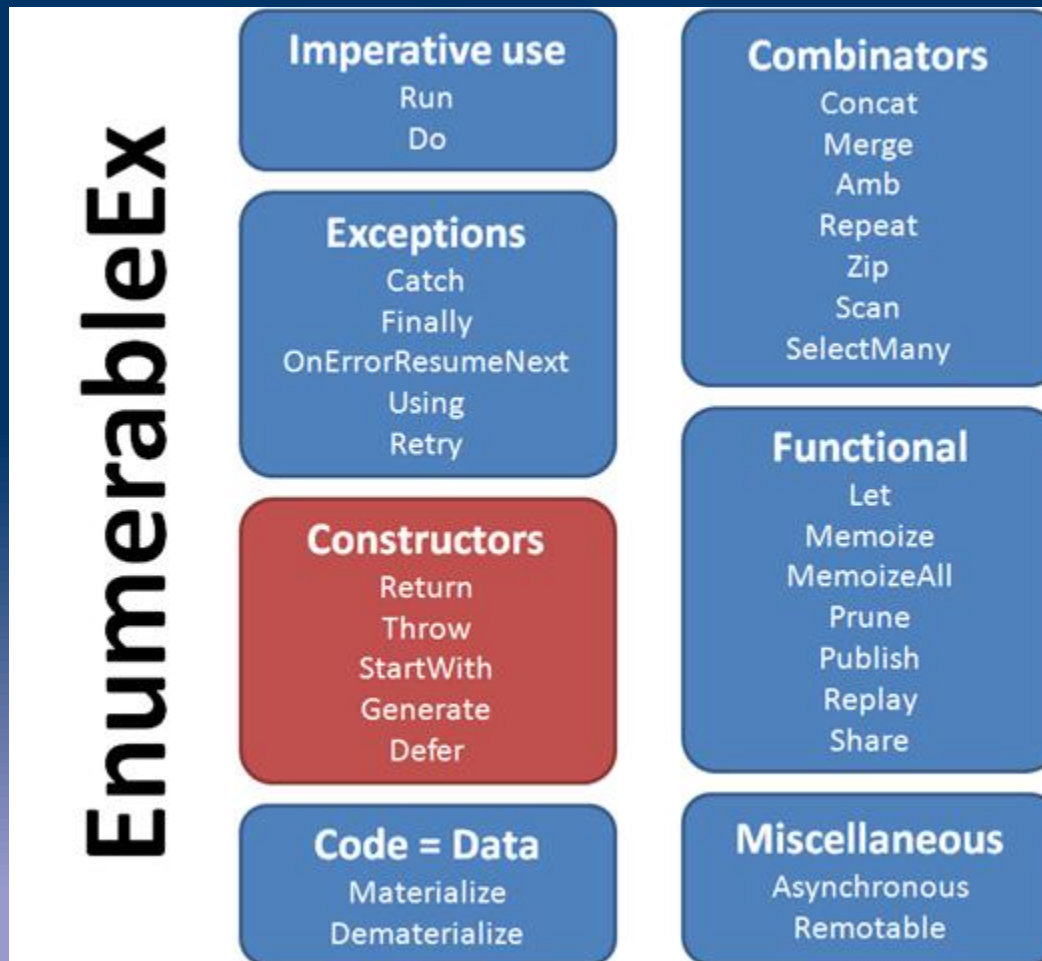
„Linq to Events“

„Query the Future“

Asynchronity

- RX abstracts and hides the concept of asynchronity and concurrency
- ... but it is still there ...
- RX introduces the notion of a Scheduler
 - The Scheduler is an abstraction for Threads, Threadpool, WPF Dispatcher, TPL ...

RX Operators



RX Sandbox

The screenshot shows the Rx Sandbox application window. On the left is a tree view of operators, with 'Merge' selected under 'StandardOperators' > 'Combinators'. The main area is titled 'Merge' and contains the following information:

- Description: Merges an observable sequence of observable sequences into an observable sequence.
- Signature: `(a, b, c) => Merge(new [] {a, b, c})`
- Inputs: Three input fields labeled 'a : String', 'b : String', and 'c : String', each with 'OnNext', 'OnError', and 'OnCompleted' buttons and a 'History' label.
- Marble diagram: A diagram with three horizontal lines labeled 'a', 'b', and 'c'. Line 'a' has two green circles followed by a vertical bar. Line 'b' has one green circle, then a gap, then another green circle, followed by a vertical bar. Line 'c' has one green circle, then a gap, then another green circle, followed by a vertical bar. A fourth line below 'c' has six green circles followed by a vertical bar.
- Buttons: 'Reset' and 'Close' buttons are at the bottom.

At the bottom right of the window, there is a page number '12' and a scrollbar.

So far ...

- Events are first class objects
- Leverage the power of LINQ
- Operations on observable collections are composable
- Declarative declarations of data flow
- Separation of data-flow and actions

Combining Events: Example D&D

```
var mouseDown = from evt in Observable.FromEvent<MouseButtonEventArgs>(
    this, "MouseDown")
    select evt.EventArgs.GetPosition(this);

var mouseUp = from evt in Observable.FromEvent<MouseButtonEventArgs>(
    this, "MouseUp")
    select evt.EventArgs.GetPosition(this);

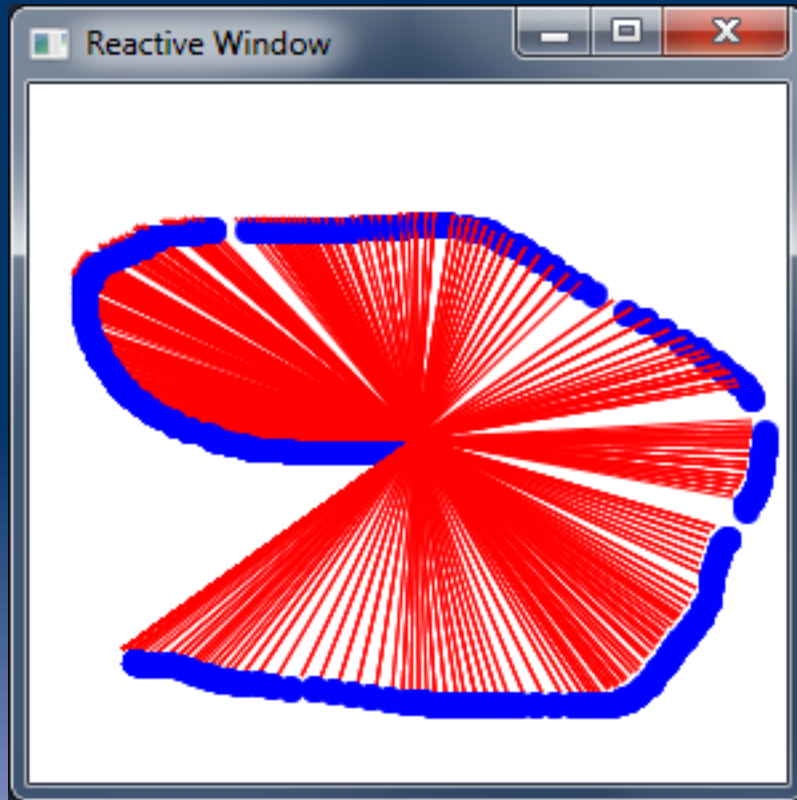
var mouseMove = from evt in Observable.FromEvent<MouseEventArgs>(
    this, "MouseMove")
    select evt.EventArgs.GetPosition(this);
```

```
var mouse =
    from start in mouseDown
    where VisualTreeHelper.FindElementsInHostCoordinates(
        start, rectangle).Count() > 0
    from delta in mouseMove.StartWith(start).TakeUntil(mouseUp)
    .Let(mm => mm.Zip(mm.Skip(1), (prev, cur) =>
        new { X = cur.X - prev.X, Y = cur.Y - prev.Y })))
    select delta;
```

```
mouse.Subscribe(delta =>
{
    Canvas.SetLeft(rectangle, Canvas.GetLeft(rectangle) + delta.X);
    Canvas.SetTop(rectangle, Canvas.GetTop(rectangle) + delta.Y);
});
```

Combining Events: Demo

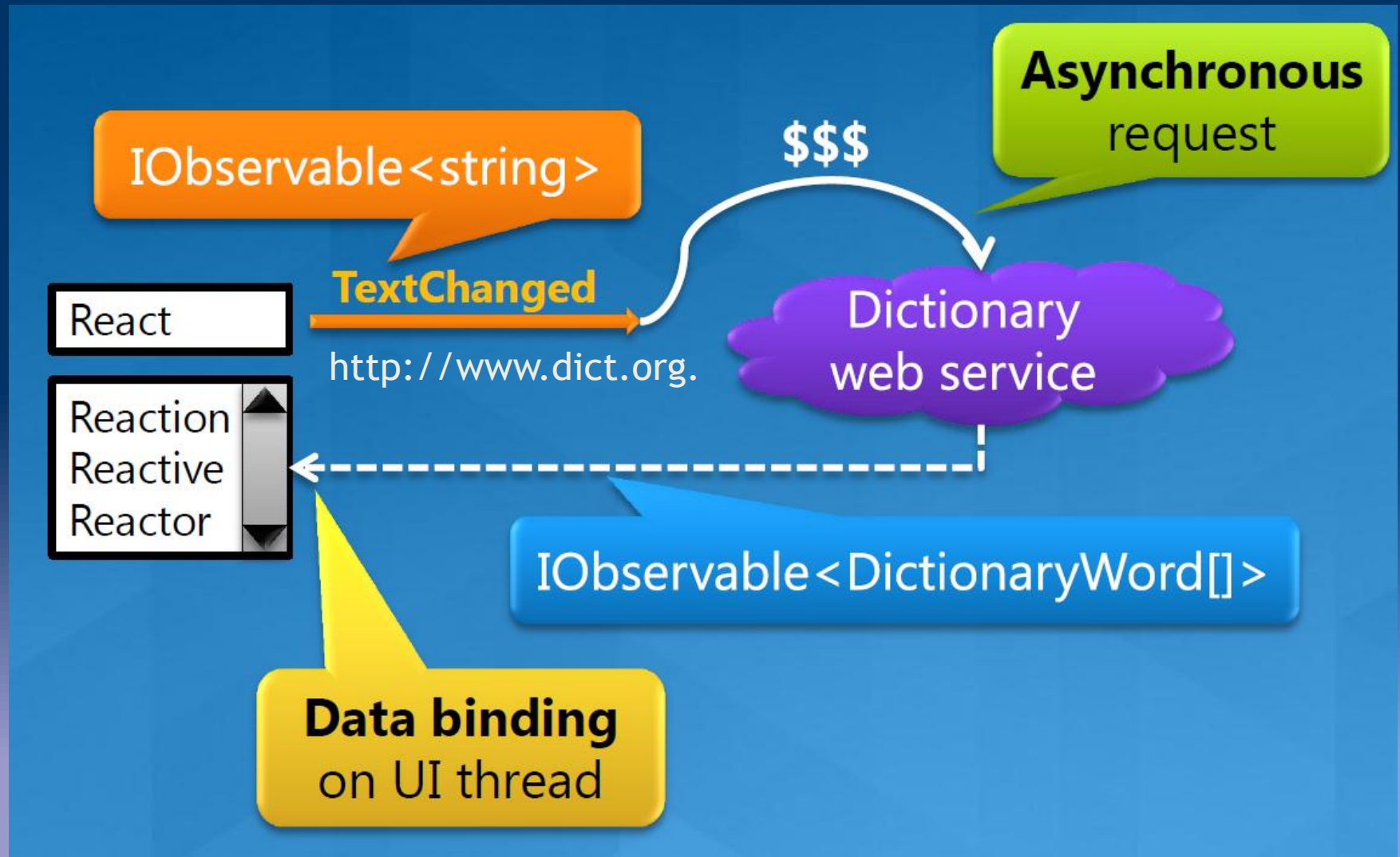
Simple Drawing



Asynchronous Data Sources

- Web Services / Cloud / Mobile
- Create Observables from Async Method Pattern
 - Begin : IAsyncResult
 - End

Demo 3



Debugging / Testing

- Debugging
 - Debugging schwierig...
 - Operators:
 - .Do , .Finally, .TimeStamp
- Testing
 - Representing asynchronous data sources as first-class objects is a big help

Summary

Way *simpler* with Rx

$$(f \circ g)(x) = f(g(x))$$

Rx is a library for *composing*
asynchronous and event-based programs
using *observable collections*.

Queries! LINQ!

JavaScript

Demo

Real World

- Silverlight
- Testing -> Abstract Asynchrony

- Silverlight Apps
 - <http://timecard.codeplex.com/>

- MVVM / Databinding -> Events
 - <https://github.com/xpaulbettsx/ReactiveUI>

- ESB
 - <http://masstransit-project.com/>
 - <https://github.com/MassTransit/MassTransit/tree/master/src/MassTransit.Reactive>

- Port to ActionScript (Adobe Flex)
 - <https://github.com/richardszalay/rxas>

Links

- <http://rxwiki.wikidot.com/>
- <http://rxwiki.wikidot.com/101samples>
- <http://code.google.com/p/rx-samples/>
- <https://github.com/joeldart/RX-Example>

The async ecosystem in .NET

- TPL
 - Framework for distribution of work (parallelism)
- Async features in C# 5.0
 - manage asynchrony from a language level
- RX
 - framework to compose and orchestrate asynchronous data streams

Podcast with Bart de Smet

<http://bit.ly/e2GjF>

PUBER UNS > PODCAST

PODCAST "ON TECHTALK'S MIND"

Als Vorreiter für innovative Technologien und Methoden ist es uns ein Anliegen, dass auch unsere Kunden von diesem Vorsprung profitieren. In regelmäßigen Abständen diskutieren wir die heißesten Themen mit anerkannten Experten aus aller Welt.

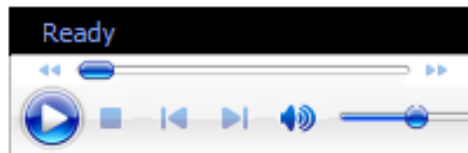
[Podcast \(RSS\) abonnieren](#)

[Podcast im iTunes-Store abonnieren](#)

13.4.2011 - Reactive Extensions Framework - An Interview with Bart De Smet

Jonas Bandi (links), Software Developer von TechTalk in Zürich, sprach mit Bart De Smet über das Reactive Extensions Framework.

Bart De Smet ist Software Development Engineer bei Microsoft und arbeitet an den Reactive Extensions for .NET.



Dauer: 1 h 11 min

TechTalk Schweiz sucht Mitarbeiter!

TechTalk ist eine Entwicklungsfirma mit ca. 60 Mitarbeiter in Wien und Budapest und Zürich. Wir fokussieren auf die .NET Plattform und auf Scrum.

Wir suchen erfahrene Entwickler für den Aufbau des Standorts Schweiz:

- Mehrjährige Erfahrung in diversen Projekten, idealerweise in unterschiedlichen Branchen
- Grosse Flexibilität, Eigeninitiative und Selbstmotivation
- Erfahrung und Bereitschaft für On-Site Einsätze bei Kunden

Wir bieten:

- Aufbau eines jungen Unternehmens in einem kleinem Team
- Möglichkeit eines Aufenthaltes in Wien / Budapest
- Weiterbildung und Verwirklichung eigener Ideen

Fragen und Diskussion

