

Developing and Integrating WebSharper Applications with Facebook

```
let FlashHidingArgs =  
  Class "FB.FlashHidingArgs"  
  |> Protocol [  
    "state" =? T<string>  
    "elem" =? T<Element>  
  ]
```

```
let InitOptions =  
  Pattern.Config "FB.InitOptions"  
  Required = []  
  Optional =  
    [ "appId", T<string>  
      "cookie", T<bool>  
      "school"
```



<http://www.intellifactory.com>
sales@intellifactory.com

3x F# MVP – 2010, 2011, 2012
(Most Valuable Professional)

Coauthor of **4 F# books**, 3 of them
with Don Syme, the designer of F#

CEO of IntelliFactory,
The F# Company

Regular **speaker** in numerous
conferences and developer
workshops



Steering Committee member of
the Commercial Users of
Functional Programming (CUFP)
workshop, representing F#

F# Books I coauthored

Expert F# - 2007

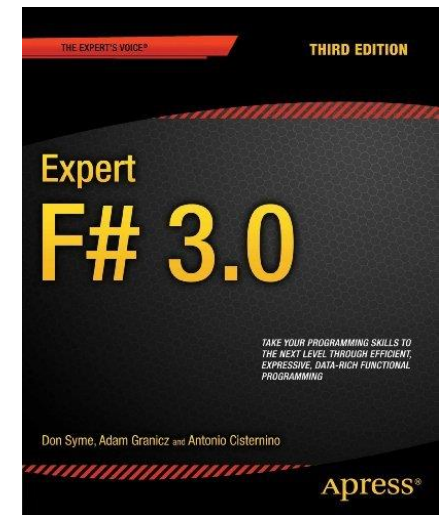
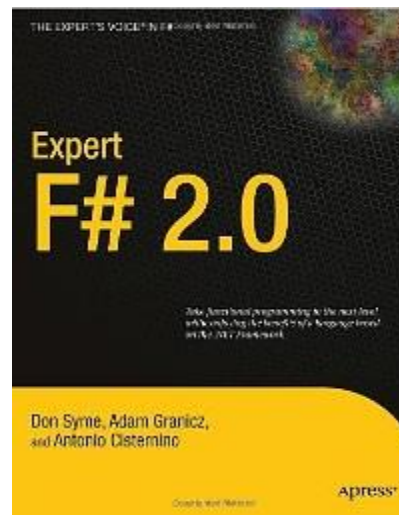
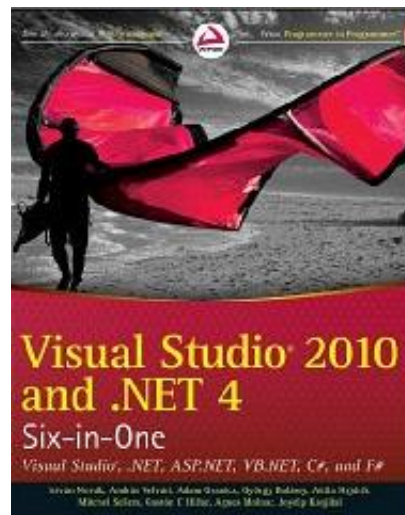
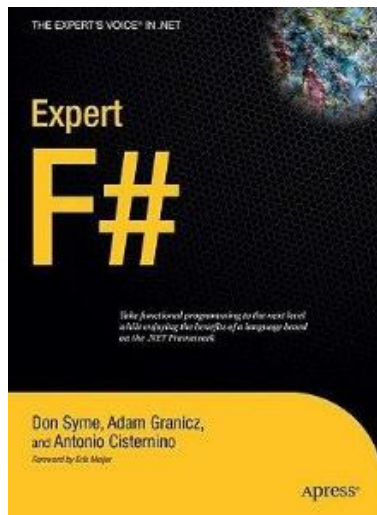
Expert F# 2.0 – 2010

Visual Studio 2010 and .NET 4 Six-in-One – 2010

Expert F# 3.0 – 2012

More publications at

<http://intellifactory.com/publications.aspx>



First F# consultancy worldwide, founded in 2004

We do **research** and **product development**

Development in F# - we built up **significant expertise** in 100+ projects

Combining creative ideas and cutting edge research

Agile processes, mature development practices

Emphasis on **quick prototyping** and **getting things done**



- **Mature, enterprise-ready framework**
- Write **all your server+client code in F#**
- Get a **complete web or mobile application**
- Interface with **any** client-side JS library via F#
- **Powerful functional abstractions**
- **Automatic** resource management
- **Safe URLs**, type-safe URLs
- and much-much more...



Develop applications with

Less code - 50-90% less

Quicker to develop – on average we find 2-3x

Easier to maintain – Significant \$\$ savings



A quick introduction

... is a web development platform that enables you to:

Program entire web applications in F#

Like "GWT for F#" – write F# instead of JavaScript

Develop, next to ordinary client-server applications,
entirely **client-based web applications** (--> HTML5 and mobile)

Develop client-based web applications in F#

HOW?

... is a web development platform that gives you:

Nearly the entire F# language, most notably

Data types – unions, tuples, records, lists, seqs, sets, maps, etc.

Constructs – pattern matching, active patterns, computation expressions, etc.

Using F# StdLibs & many .NET namespaces -- *easy to extend*

Addressing third-party JavaScript libraries -- *easy to extend*

Uniform, Extensible Development Model

... is a web development platform that gives you:

Powerful abstractions

Pagelets, formlets, flowlets, sitelets

Automatic resource management (CSS, JS, etc.)

Web programming abstractions

... is a web development platform that gives you:

ASP.NET and ASP.NET MVC integration for compatibility

Via server controls that add client-side functionality

Visual Studio 2008/2010/2012 integration

Project templates

Compiler integration

Single-click deployment

Seamless environment integration

Pagelets

Dynamic, client-side functionality in F#

Sitelets

Composable, type safe web applications

Formlets

Composable, type safe web forms

Dependent formlets – web forms with element dependencies

Flowlets – sequences of web forms

Extensions

Type safe mappings of JavaScript libraries

Slidelets

Modeling mobile and Windows 8 UIs

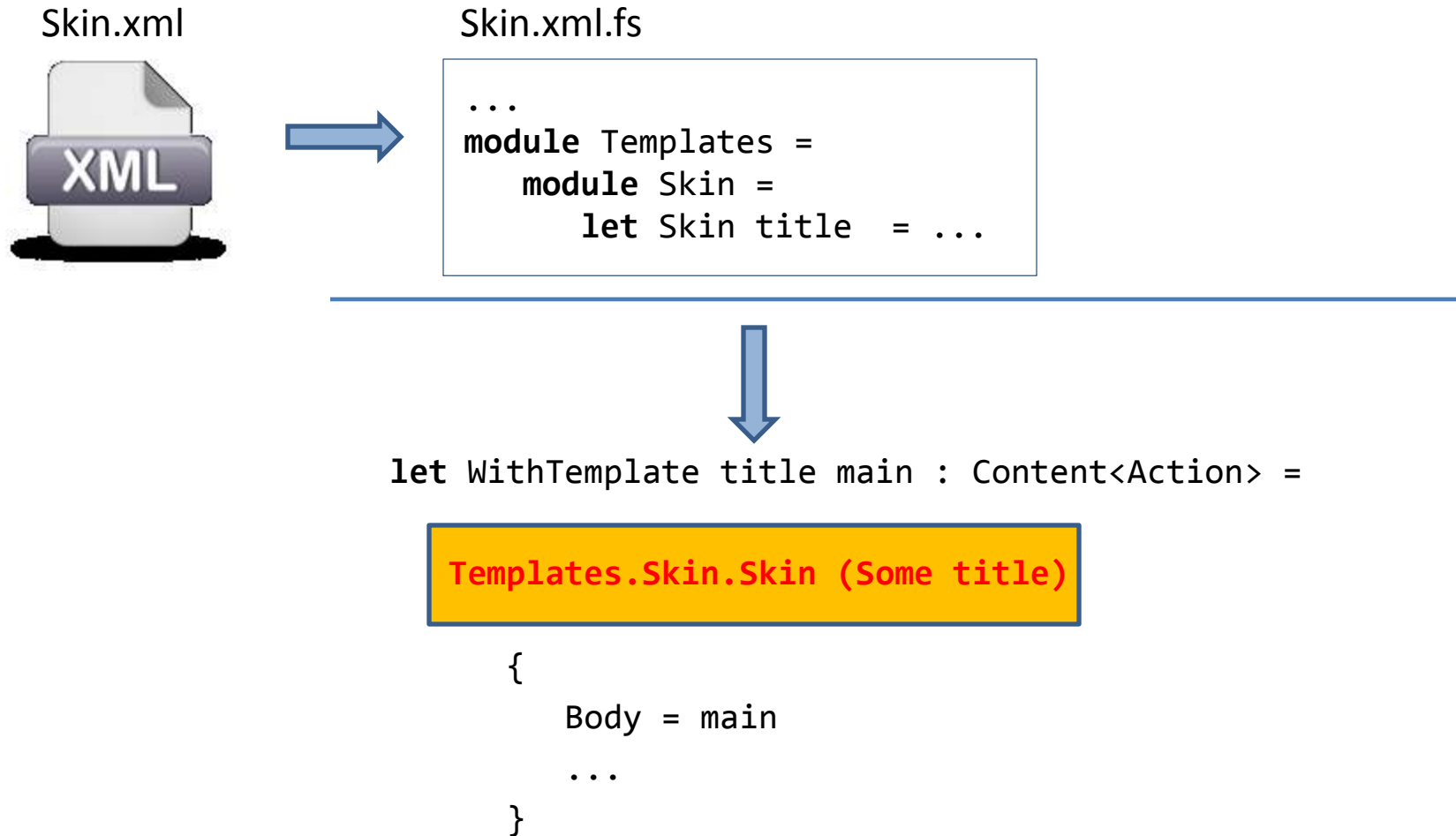
- Type-safe
- Composable
- First-class

Parameterized over a union type:

```
/// Actions that correspond to the different pages in the site.
```

```
type Action =  
    | Home  
    | Contact  
    | Protected  
    | Login of option<Action>  
    | Logout  
    | Echo of string
```

Strongly-typed, safe URLs



Runtime-checked, safe URLs

Main.html



```
module Skin =  
  type Page =  
    {  
      Body : Content.HtmlElement list  
    }
```

```
let MainTemplate =  
  let path = Path.Combine(__SOURCE_DIRECTORY__, "Main.html")
```

```
Content.Template<Page>(path)  
  .With("body", fun x -> x.Body)
```

```
let WithTemplate body : Content<Action> =  
  Content.WithTemplate MainTemplate <| fun context ->  
    {  
      Body = body context  
    }
```


Composing web applications from pieces

```
let EntireSite =  
    let home = Sitelet.Content ...  
    let authenticated = Sitelet.Protect filter <| ...  
    let basic = Sitelet.Infer <| fun action -> ...  
  
    Sitelet.Sum  
        [  
            home  
            authenticated  
            basic  
        ]
```

- Type-safe
- Composable
- First-class

Dependent formlets – can express dependencies

Flowlets – provide step-by-step rendering

Layout – via combinators such as

`Formlet.Horizontal` and `Formlet.Vertical`

```
let TB label msg =  
    Controls.Input ""  
    |> Validator.IsNotEmpty msg  
    |> Enhance.WithValidationIcon  
    |> Enhance.WithTextLabel label
```

General form:

```
Formlet.Yield (fun v1 v2 ... vn -> <compose all vi's>)  
<*> formlet1  
<*> formlet2  
...  
<*> formletn
```

WebSharper formlets - Example

```
type Person = { Name: string; Email: string }
```

```
[<JavaScript>]
```

```
let PersonFormlet () : Formlet<Person> =  
    let nameF = TB "Name" "Empty name not allowed"  
    let emailF = TB "Email" "Please enter a valid email address"  
    Formlet.Yield (fun name email -> { Name = name; Email = email })  
    <*> nameF  
    <*> emailF  
    |> Enhance.WithSubmitAndResetButtons  
    |> Enhance.WithLegend "Add a New Person"  
    |> Enhance.WithFormContainer
```



Add a New Person

Name 

Email 

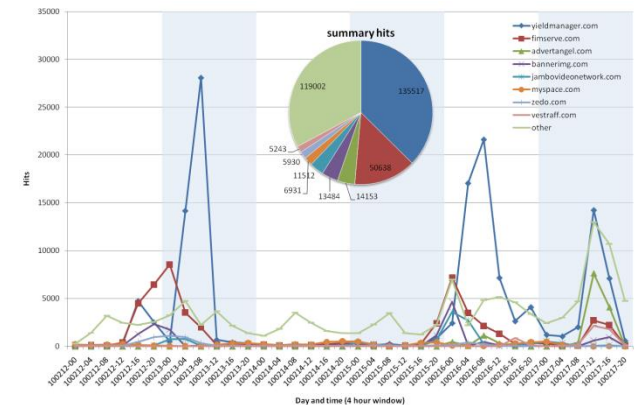
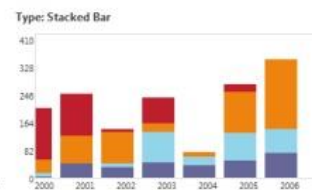
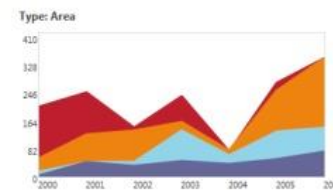
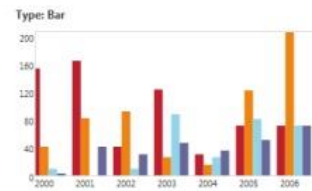
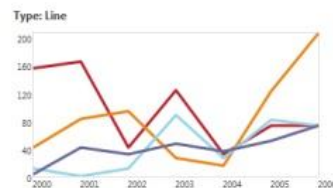
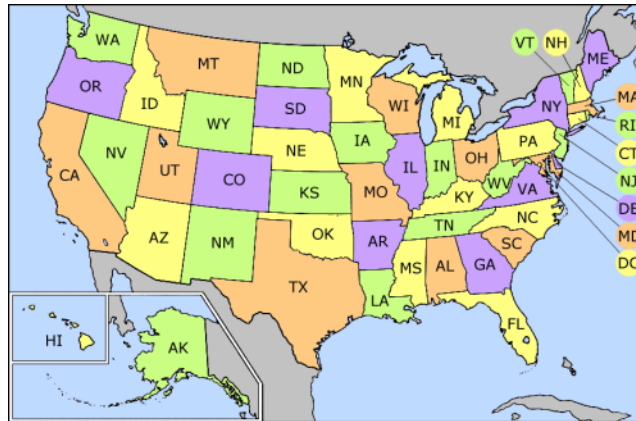
Available for various UI control set libraries such as:

- jQuery UI
- Yahoo UI
- Ext JS
- **jQuery Mobile**

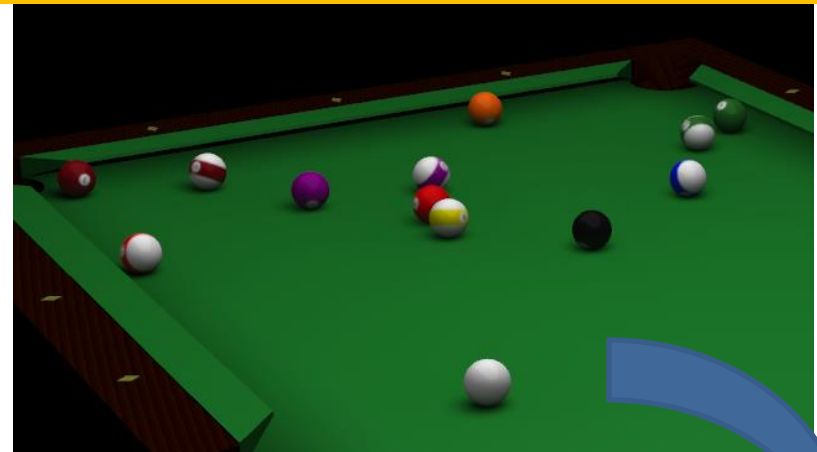


Other extensions

- GIS
 - Google Maps
 - Bing Maps
- Visualization
 - Infovis
 - Protovis
 - Google Visualization



- HTML5
 - WebGL
 - O3D
 - GLMatrix



Dozens of extensions available at:

`http://websharper.com/extensions`

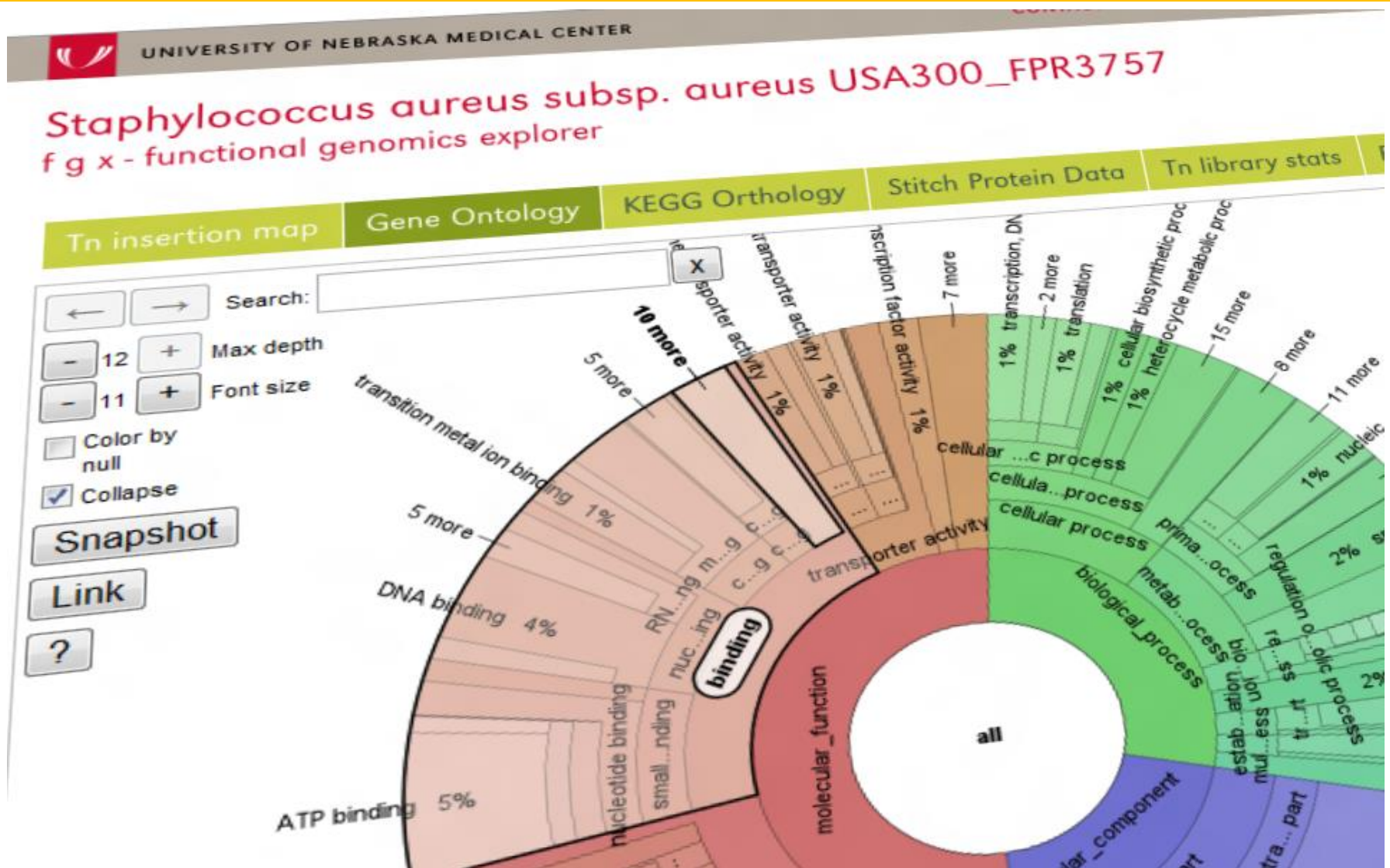
- Exposes mobile capabilities to JavaScript
- Provides the necessary packaging
- Enables quick and seamless multi-targeting
- Scales into the cloud

- JavaScript is the IL ...
 - of client-side web applications

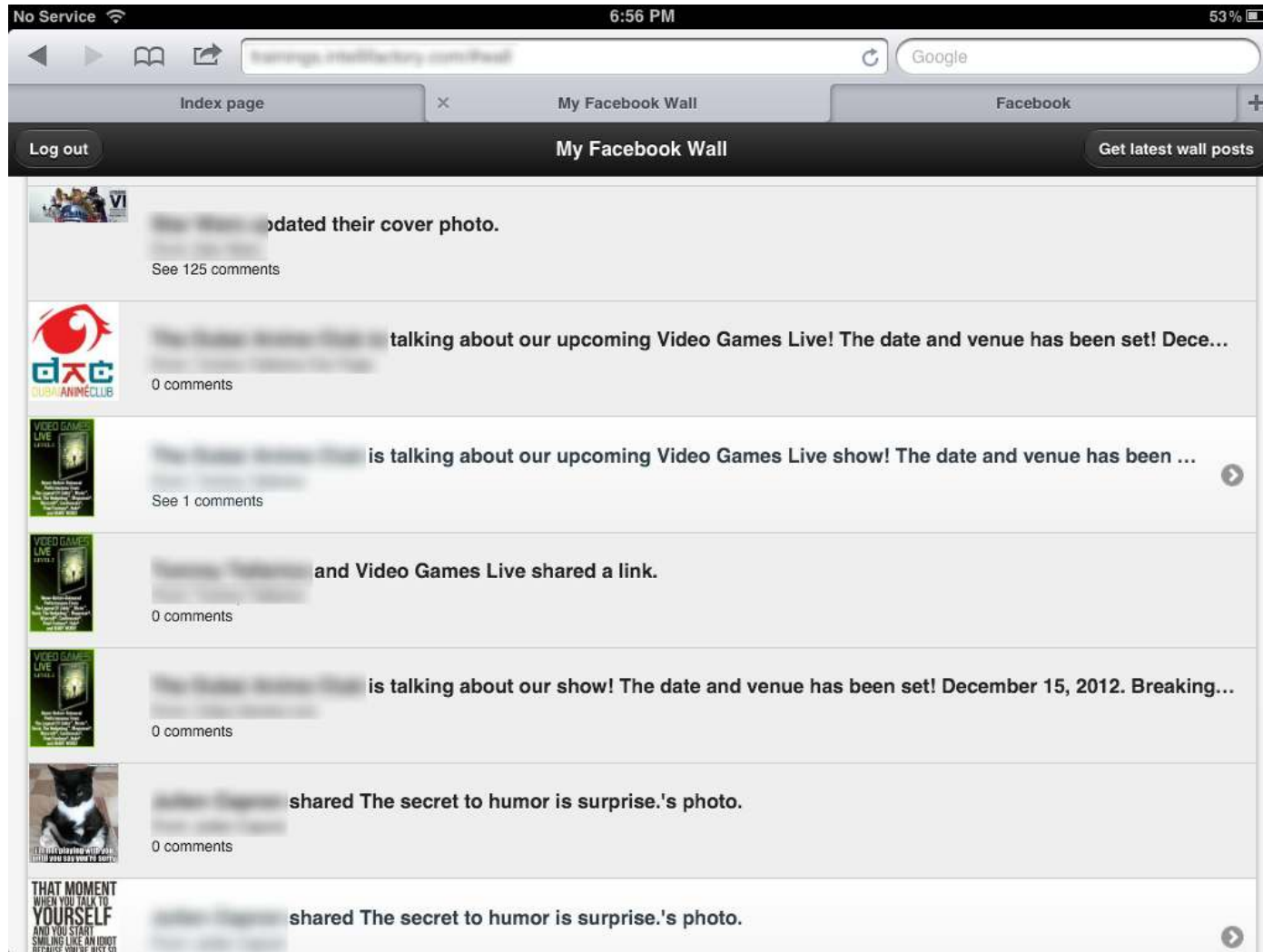
and is becoming the IL ...

 - of desktop applications: Windows 8
 - of mobile applications: Android, WP7, etc.
- Some even write the server side in JavaScript

HTML5 Applications



My Facebook Wall – The Application



My Facebook Wall – Configuration

Create New App

App Name: [?] 

App Namespace: [?]

Web Hosting: [?] ☐ Yes, I would like free web hosting provided by Heroku ([Learn More](#))

By proceeding, you agree to the [Facebook Platform Policies](#)

Continue **Cancel**



TestApplication1234
App ID: 111915 [redacted]
App Secret: 620fb6 [redacted] (reset)
 (edit icon)

My Facebook Wall – Configuration

Basic Info

Display Name: [?]

Namespace: [?]

Contact Email: [?]

App Domains: [?]

Category: [?] Choose a sub-category

Hosting URL: [?] <https://sultry-ridge-4321.herokuapp.com/> (Remove)

Sandbox Mode: [?] ☐ Enabled ☒ Disabled

Binding the Facebook API

```
namespace IntelliFactory.WebSharper.Facebook

module Definition =
    ...
    let FB =
        Class "FB"
        |> [ ...
            "login" => (LoginResponse ^-> T<unit>) * !?LoginOptions ^-> T<unit>
            "getLoginStatus" => (LoginResponse ^-> T<unit>) ^-> T<unit>
            "api" => T<string>?url * !?T<string>?``method`` * !?T<obj>?options
                * (T<obj> ^-> T<unit>)?callback ^-> T<unit>
        ]
        |> Requires [Res.FacebookAPI]

    let Assembly =
        Assembly [
            Namespace "IntelliFactory.WebSharper.Facebook" [
                ...; FB
            ] ...
        ]
```

Defining resources

Resources are annotated on code units and their use is automatically tracked.

```
module Definition =  
    open IntelliFactory.WebSharper.InterfaceGenerator  
    open IntelliFactory.WebSharper.Dom  
  
    module Res =  
        let FacebookAPI =  
            Resource "FacebookAPI" "https://connect.facebook.net/en_US/all.js"
```


Defining classes – useful operators

- `=? : Member`

```
let FlashHidingArgs =  
  Class "FB.FlashHidingArgs"  
  |> Protocol [  
    "state" =? T<string>  
    "elem" =? T<Element>  
  ]
```

Defining optional members

- Typical in most JavaScript libraries
- Used heavily in configuration objects

```
let InitOptions =  
    Pattern.Config "FB.InitOptions" {  
        Required = []  
        Optional =  
            [  
                "appId", T<string>  
                "cookie", T<bool>  
                ...  
                "hideFlashCallback", FlashHidingArgs ^-> T<unit>  
            ]  
    }
```

Defining safe enumerations

- Typical in most JavaScript libraries
- Used heavily in configuration objects

```
let UserStatus =  
    Pattern.EnumStrings "FB.UserStatus"  
        ["connected"; "not_authorized"; "unknown"]
```

Walk through in
... In Visual Studio ...

Main points:

Performing Login/Logout

```
A [Attr.HRef "#"] -< [<"Login" or "Logout" depending on status>]  
|>! OnClick (fun el ev ->  
    FB.GetLoginStatus <| fun resp ->  
        if resp.Status = UserStatus.Connected then  
            FB.Logout updateStatus  
        else  
            FB.Login(updateStatus, LoginOptions(Scope = "read_stream"))  
)
```

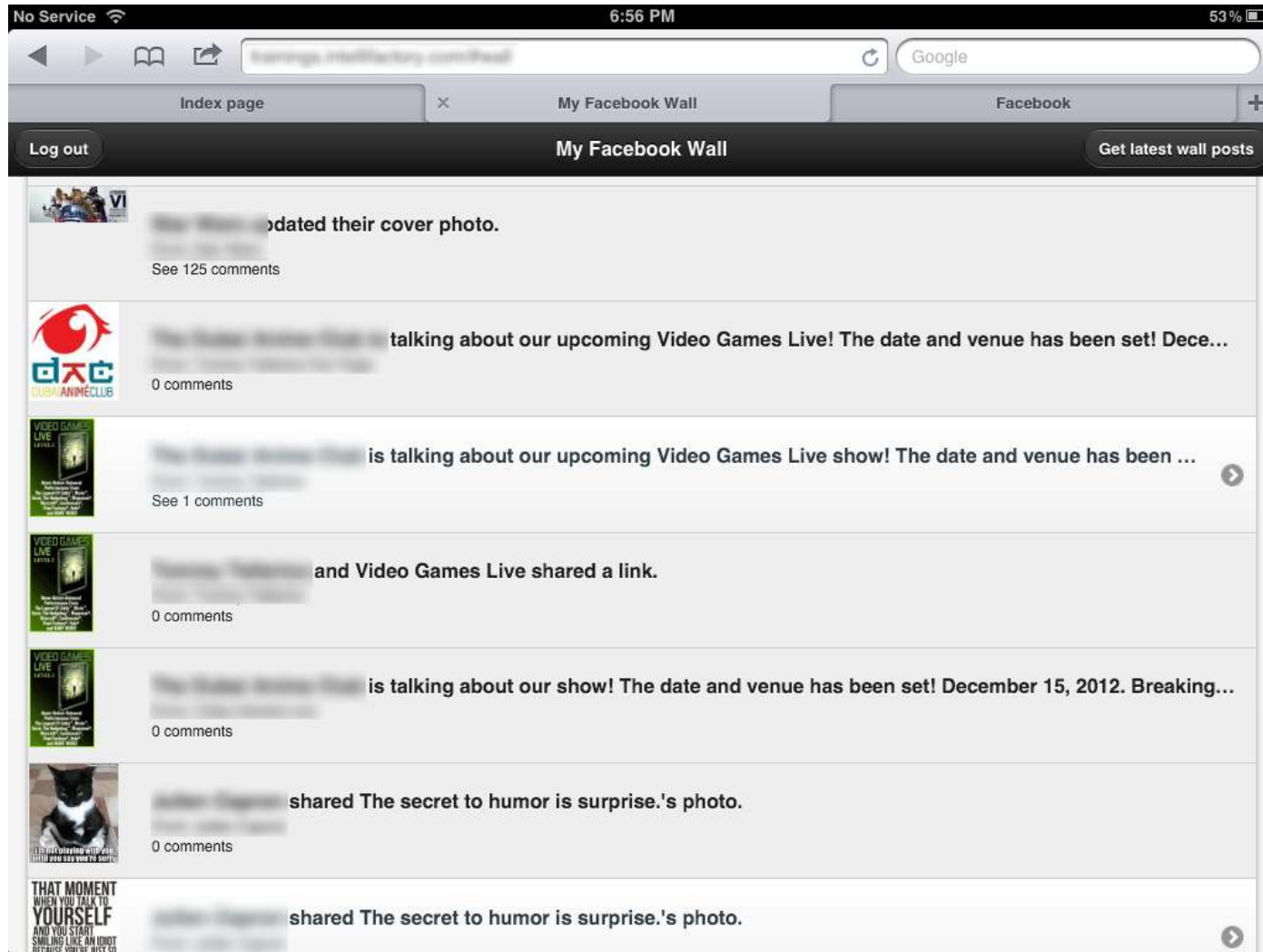
Main points:

Getting wall posts

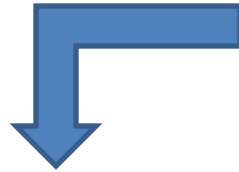
```
btnGetPosts
|>! OnClick (fun el ev ->
    Mobile.Instance.ShowPageLoadingMsg("a", "Receiving wall posts...")
    FB.Api("/me/home", fun o ->
        wall.Clear()
        o?data |> Array.iter (fun x ->
            let message = x?message ||? x?story ||? x?caption
            LI [
                yield H6 [Text message]
                yield P [Text ("From: " + x?from?name)]
                ...
            ]
        |> wall.Append
        Mobile.Instance.HidePageLoadingMsg()
    )
    JQuery.Of(wall.Body) |> ListView.Refresh
)
```

Deployment to a public URL, as configured in the Facebook application

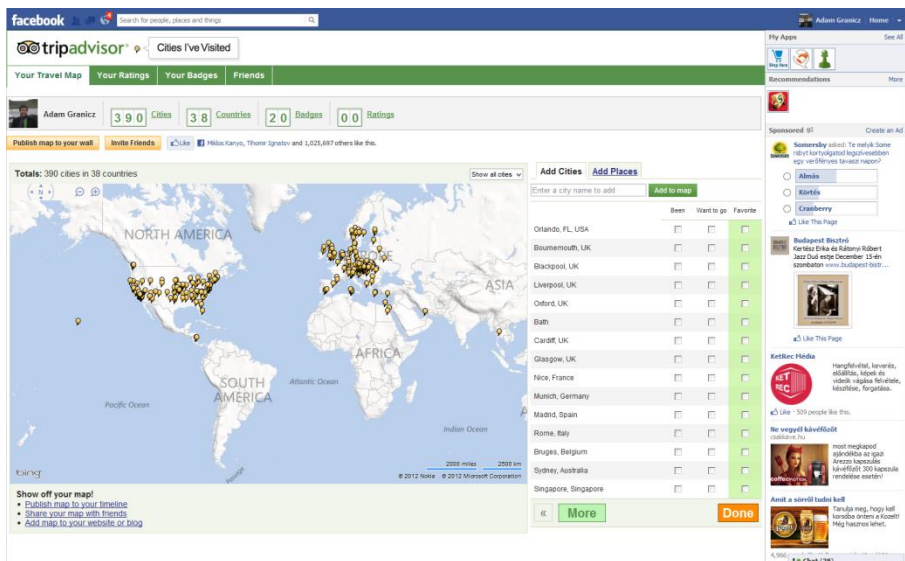
Application – Running!



Same code base



Facebook Application



Mobile Application



Summary – WebSharper

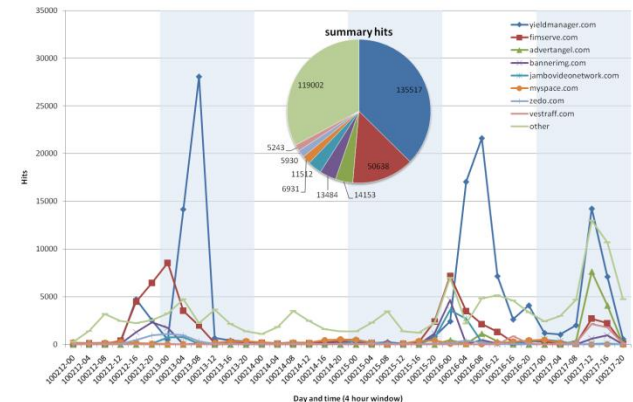
F# + WebSharper gives:

- Massive **productivity gain**
- Access to a **growing market** opportunity
- Quick path to **multiple mobile platforms**
- Scaling from **desktop** and to the **cloud**



Advocates functional programming

- Using **more powerful** abstractions
- **Cutting development time**
- Producing **shorter, more maintainable code**



Wait, it's not over!

Aggregates and catalogs FP content about:

Q&A

Events/Conferences

Courses

User Groups

Blogs

Jobs

Developers

etc...



The screenshot shows the FPish website interface. At the top, there's a navigation bar with links for Home, Profile, Inbox, Answers, Events, Courses, Groups, Blogs, Jobs, and Developers. Below this, there's a section for "Subscribe to browse" and "Benefits". The main content area displays "569 developers found" with search filters for terms and location. A bar chart shows the distribution of developers by technology, and a world map shows their geographical distribution. Below the map, there are sample profiles for Loic Denuziere and Adam Granicz. On the right side, there's a "Developer tags" section listing various functional programming topics and their counts.

Developer tags:

- haskell × 532
- web × 299
- scripting × 269
- unix × 269
- type systems × 167
- dsl design × 166
- formal verification × 160
- teaching × 155
- translation × 150
- m × 140
- concurrent × 134
- generic × 118
- gui × 118
- agile × 108
- functional × 103
- parser combinators × 102
- graphics × 83
- arrows × 78
- network × 78
- metaprogramming × 77
- system programming × 75
- parallelism × 73
- physics simulation × 62
- packaging × 58
- signal processing × 47
- natural language × 46
- database × 45
- numerical computation × 45
- denotational design × 41
- ghc × 41
- machine learning × 38
- attribute grammar × 37
- reactive × 37
- reverse engineering × 37
- transactional business applications × 34
- categories × 33

Feature-rich, online, cloud-hosted development environment

Provides syntax highlighting, semantic and type checking on the fly

Supports multi-project solutions, switching between them with a single click

Automated building, versioning, dependency tracking

Uses no Silverlight, requires no installed extra software or browser plugins

Full F# language support

Multi-project solutions

Web and mobile Apps

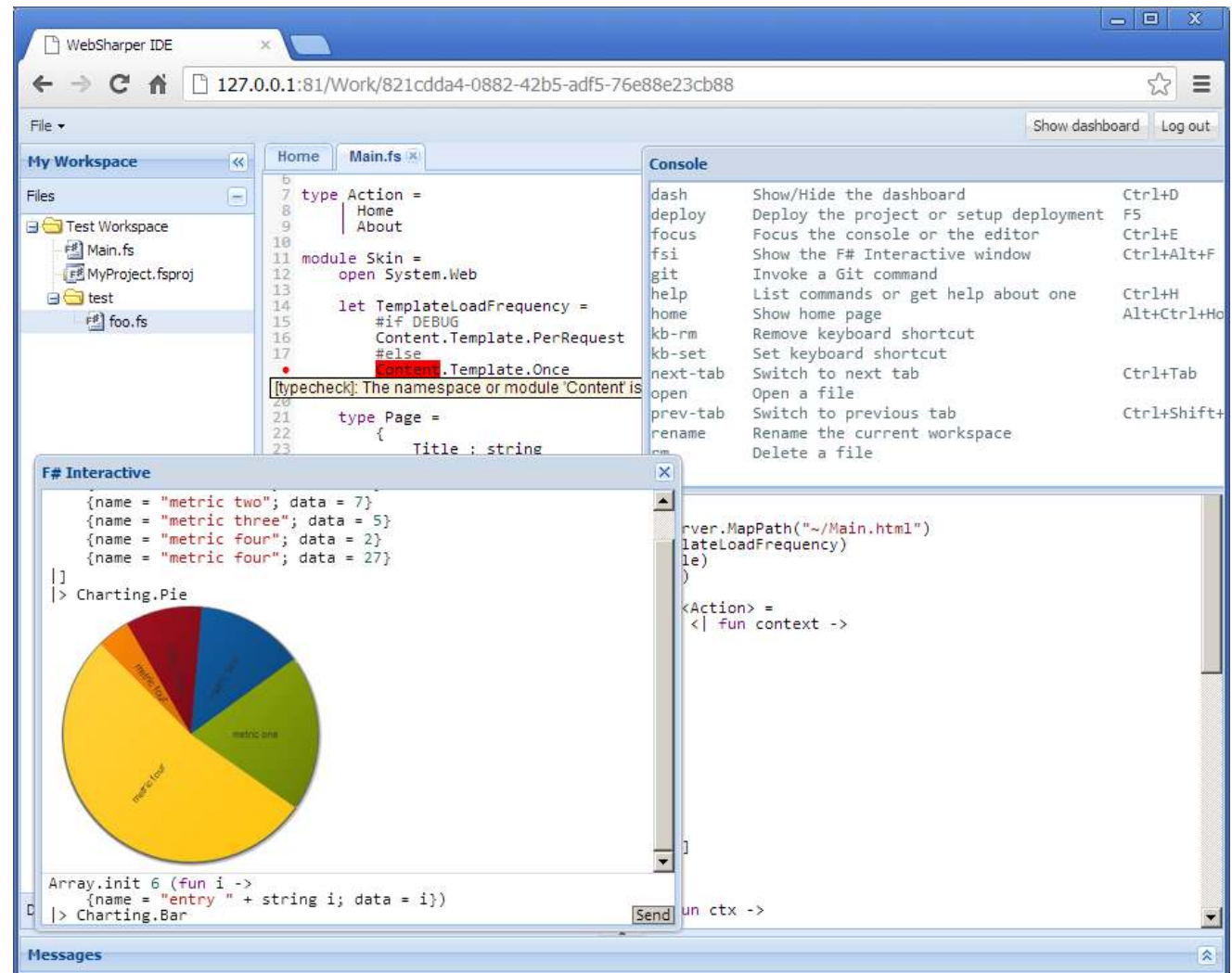
Syntax highlighting

On the fly type checking

Interactive exploration

Integration with data

Support for type
providers





Find out more at:

<http://intellifactory.com>

<http://websharper.com>

<http://infoq.com/articles/WebSharper>

Twitter: **#granicz**, **#websharper**
granicz.adam@intellifactory.com