Schuster Stefan

Advanced JavaScript in FAT Clients

JSF Days*08

irian | THE JAVA EXPERTS

Don’t panic!

No JSF

No Java

Topics

- What/Why?
- Examples
- Dojo
- How-to
  - Start
  - Structure & Organization
  - Implementation

What/Why?

What/Why?

1. What are we talking about?
   - JavaScript - Pure JavaScript
2. Why do we talk about JavaScript?
   - Initially: Ajax
   - In the meantime: JS indispensable

What/Why?

1. Areas of application:
   - Validation
   - UI enhancements (show/hide)
   - AJAX
   - RIA/SPA
     - Rich Internet Application
     - Single Page Application

What/Why?

1. **What is an RIA/SPA?**
   - ***Rich Internet applications*** *(RIA) are web applications that have the features and functionality of traditional desktop applications. RIAs typically transfer the processing necessary for the user interface to the web client...*

     http://en.wikipedia.org/wiki/Rich_Internet_application - 15.02.2008

What/Why?

1. Advantages: Usability
   - Direct feedback
   - More responsive
   - Desktop-like widgets and usage possible

What/Why?

1. Advantages: Application types
   - Some applications impossible within page-based architecture
     - WYSIWYG
       - Rich text formatting
       - Interactive graphics
     - Collaborative applications

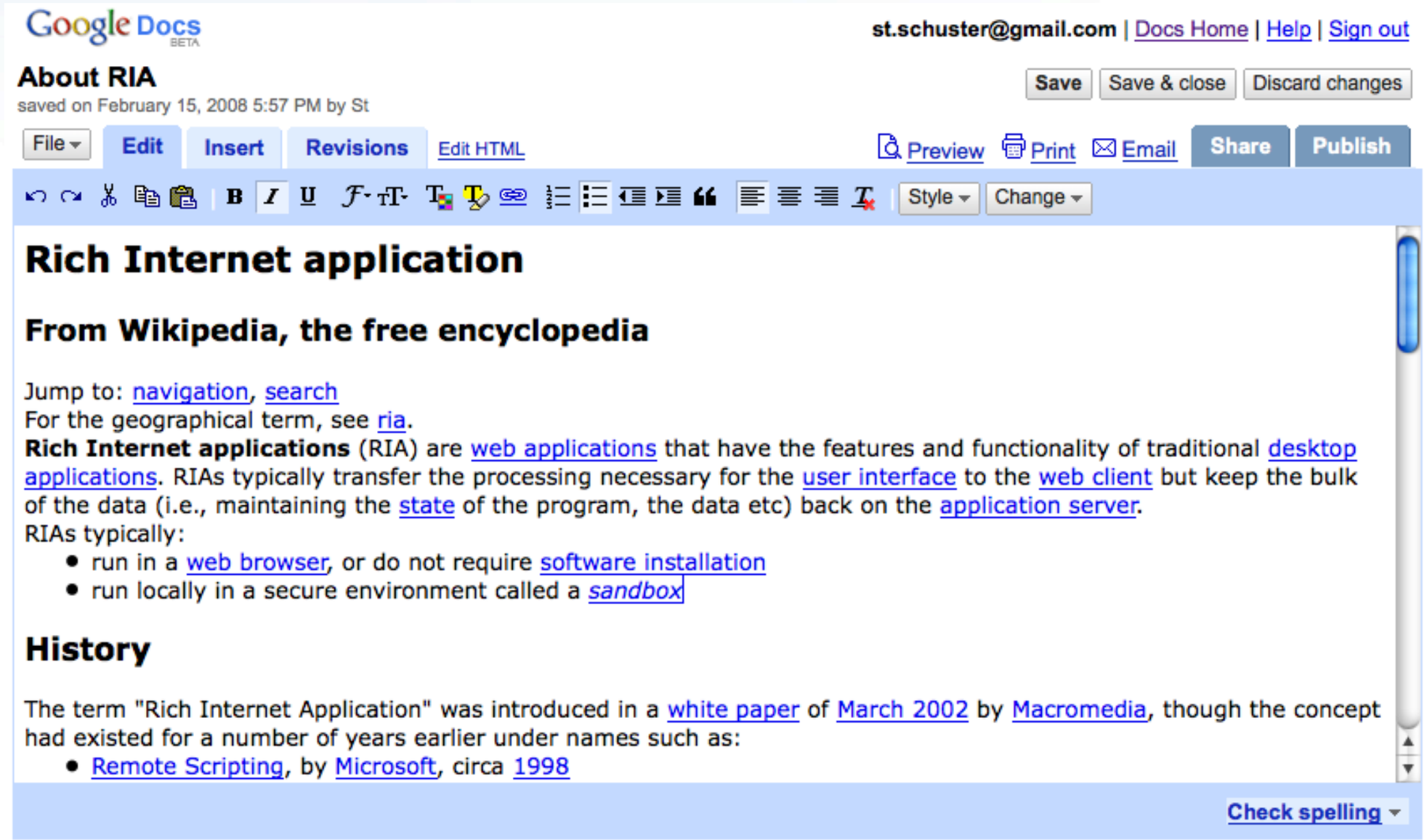
What/Why?

1. Advantages: Server load
   - Server only has to provide the data
   - Rendering gets done on the client
   - Fewer requests to handle

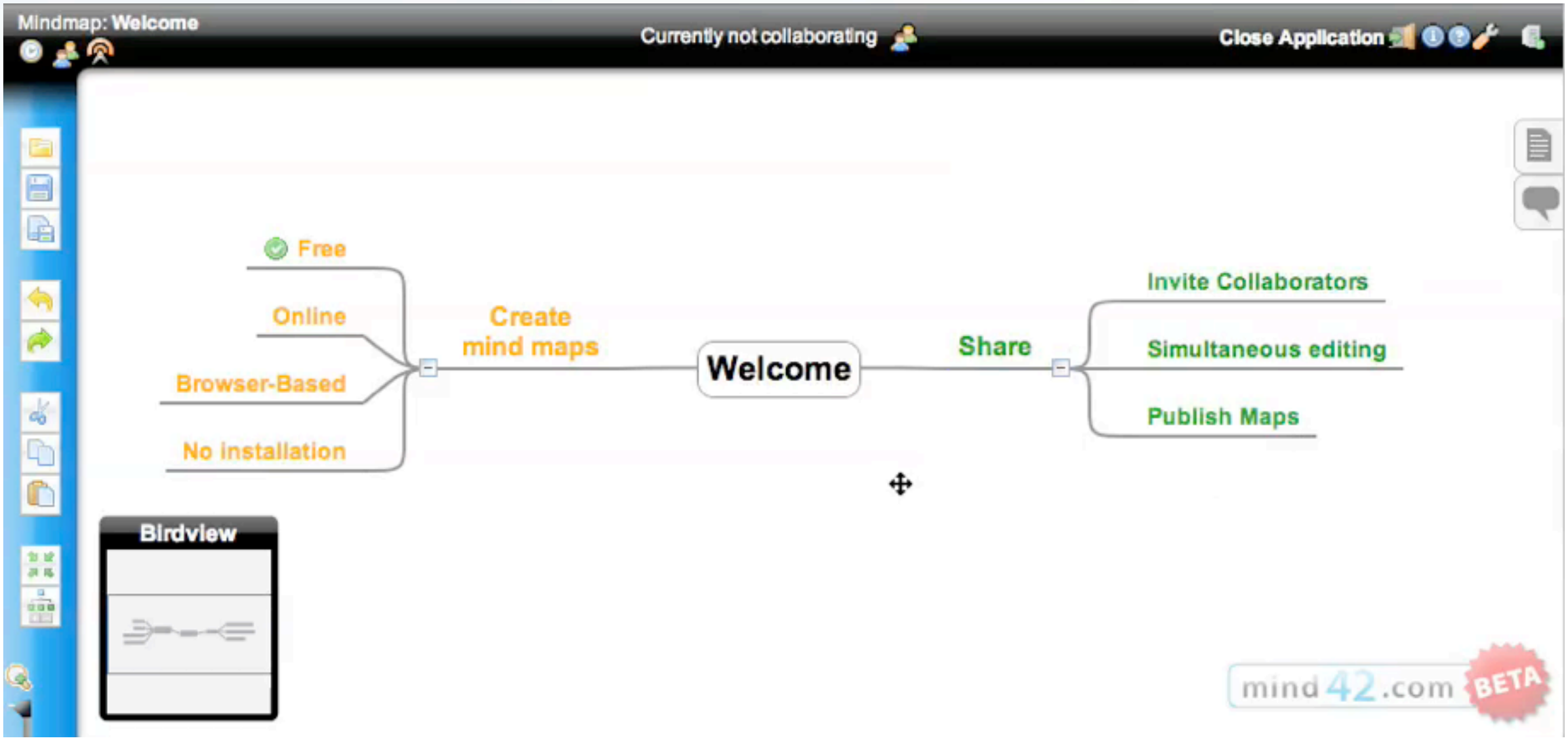
What/Why?

1. Advantages: Offline
   - Server only needed for data
   - Browser will have an integrated database:
     - Google Gears
     - HTML 5
   - Only data source has to be changed ...
     - ... in disregard of synchronisation

Example: Google Docs



Example: Mind42



What/Why?

1. Counter-arguments
   - Security
   - Complexity
     - Know-How
     - Additional work (bad tools)
- Type of applications
  - SPA for some simple forms? NO!

Dojo

JS Frameworks - General

1. Abstraction areas
   - Cross Browser Help
     - Event handling
     - JS API (Arrays, window object, ...)
   - DOM manipulation
   - Widgets
   - Effects
   - ...

Dojo

1. Many Libraries
   - Prototype/Scriptaculous
   - jQuery
   - YUI
   - Ext JS
   - MooTools
   - Qooxdoo
   - ...

Dojo

1. Why I prefer Dojo:
   - Module based
   - dojo.require
   - Build System

Dojo

1. Module based
   - Small & fast set of core features
   - Separate modules for every use-case

Events	Ajax	DOM	Effects
String	Date	Colour	Math
I18N	DnD	Widgets	Offline
Crypto	Charting	GFX	Comet
Templates	SQL	Validation	CSS

Dojo

1. dojo.require:
   - Namespacing
     - dojo.date.locale
   - AJAX based on demand loading
     - dojo/date/locale.js
     - Ideal for development

Dojo

1. Build System:
   - Based on dojo.require
     - recursive resolution of requires
     - creation of single JS file of resolved requires
     - Minification / Obfuscation of code possible
   - Single obfuscated file means less HTTP requests and smaller file size

Dojo

1. Additional goodies:
   - dojo.declare for OO object definition
   - dojo.connect
     - Normalized DOM connection
     - Connections to every other method

How-to

How-to: Start

1. If the client should work with the data, we need a:

Data model

How-to: Start

1. JavaScript models basically consist of:
   - Strings
   - Numbers
   - Booleans
   - Arrays
   - Objects

How-to: Start

1. No good Dojo-specialized tooling available
   - Model classes hand-coded
   - Getters/Setter hand-coded
   - But generators could be written pretty easily

How-to: Implementation

1. Example: irian.myApp.model.User

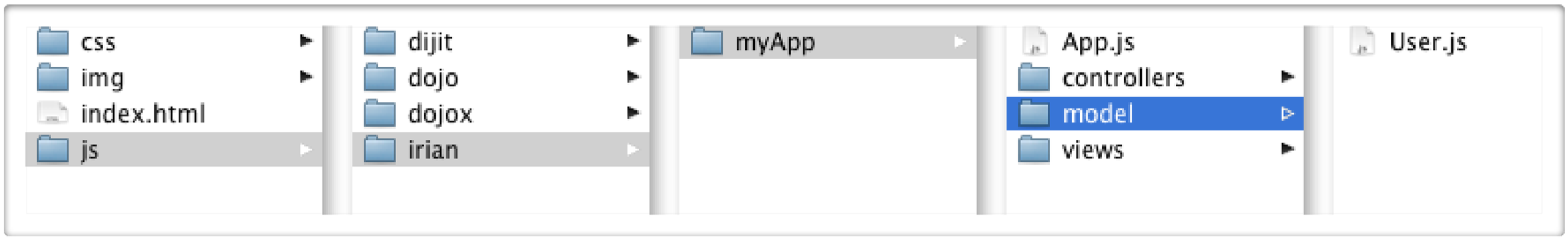
```
dojo.provide("irian.myApp.model.User");

dojo.declare("irian.myApp.model.User", null, {
  constructor: function(id, username) {
    this._id = id || 0;
    this._username = username || "";
  },
  getId: function() {
    return this._id;
  },
  setId: function(id) {
    this._id = id;
  },
  ...
});
```

How-to: Start

1. Based on the model you could implement:
   - Needed business logic in JavaScript
   - Communication
     - Format e.g. JSON
     - Method-based
     - Data-based

How-to: Structure & Oranization

1. Based on dojo.require
   - One file per “class”
   - Folders for namespaces



How-to: Structure & Organization

1. Model
   - Business Logic
1. View
   - DOM manipulation
   - Templating
   - Widget initialization
1. Controller
   - Event handling

How-to: Structure & Organization

1. Controllers
   - Inits the view
   - Implements event handlers
   - Initiates all the real work

How-to: Implementation

1. Controller: Example

```
dojo.provide("irian.myApp.controllers.Login");
dojo.require("irian.myApp.views.Login");

dojo.declare("irian.myApp.controllers.Login", null,
{
  constructor: function() {
    this.view = new irian.myApp.views.Login(this);
  },
  show: function(div) {
    this.view.show(div);
  },
  onLogin: function() {
    //Do something
  }
});
```

How-to: Structure & Organization

1. Views
   - Know their controller (for event setup)
   - Finally show the mask
     - innerHTML
     - templating
     - DOM manipulation

How-to: Structure & Organization

1. Views:
   - innerHTML faster than DOM access
     - `http://www.quirksmode.org/dom/innerhtml.html`
   - JS Templating to separate presentation from JS
     - dojox.dtl (Django Template Language)
     - Trimpath JavaScript Templates
       - `http://code.google.com/p/trimpath/wiki/JavaScriptTemplates`

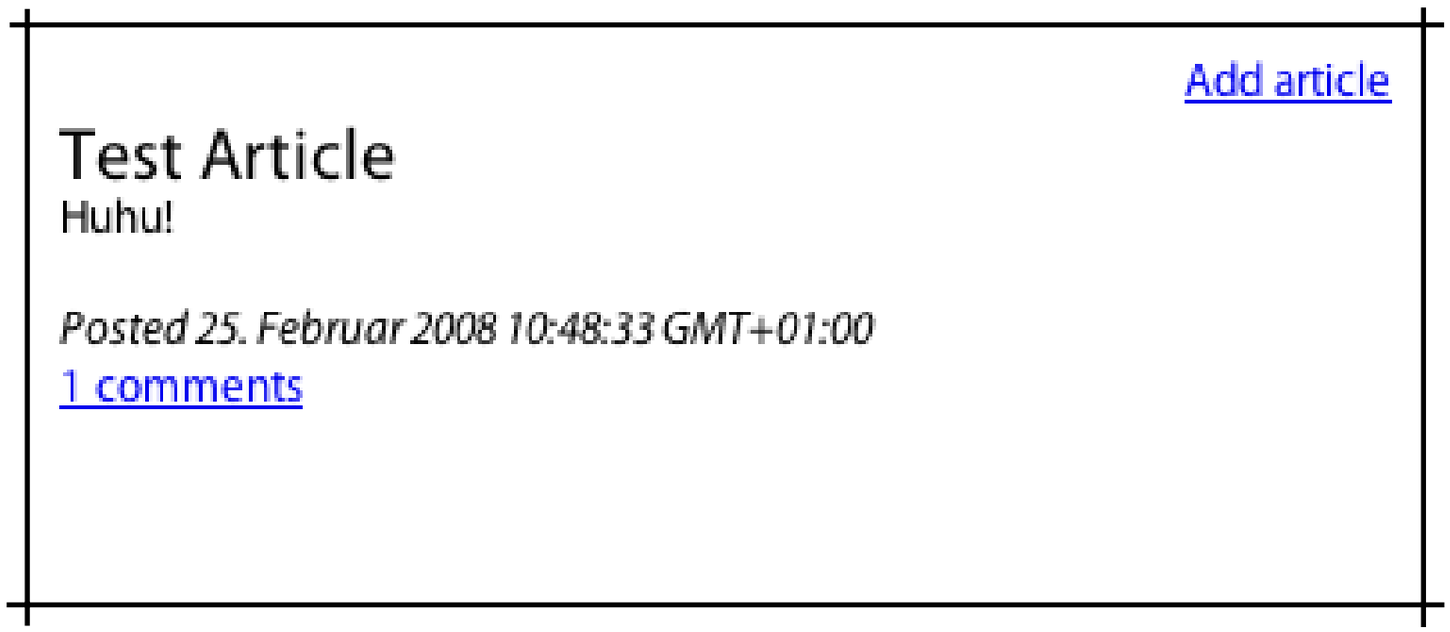
How-to: Implementation

1. View: Example

```
dojo.provide("irian.myApp.views.Login");

dojo.declare("irian.myApp.views.Login", null, {
  constructor: function(controller) {
    this.controller = controller;
    this.div = null;
  },
  show: function(div) {
    this.div = div;
    //Fill the div
    dojo.connect(dojo.byId("loginButton"), "onclick",
                          this.controller, "onLogin");
  }
});
```

Example

1. A simple working example:
   - Pure JavaScript Proof Of Concept Mini Blog



Example

1. JS Blog
   - Simplified structure
     - Model
     - View-Controller
   - No templating (inner-html)
   - No server
   - No persistence

Example

DEMO

Download:
http://www.sschuster.net/presentations/barcamp/200709/bcBlog.zip

Example

- More advanced example application with Offline support

http://www.sschuster.net/mutt

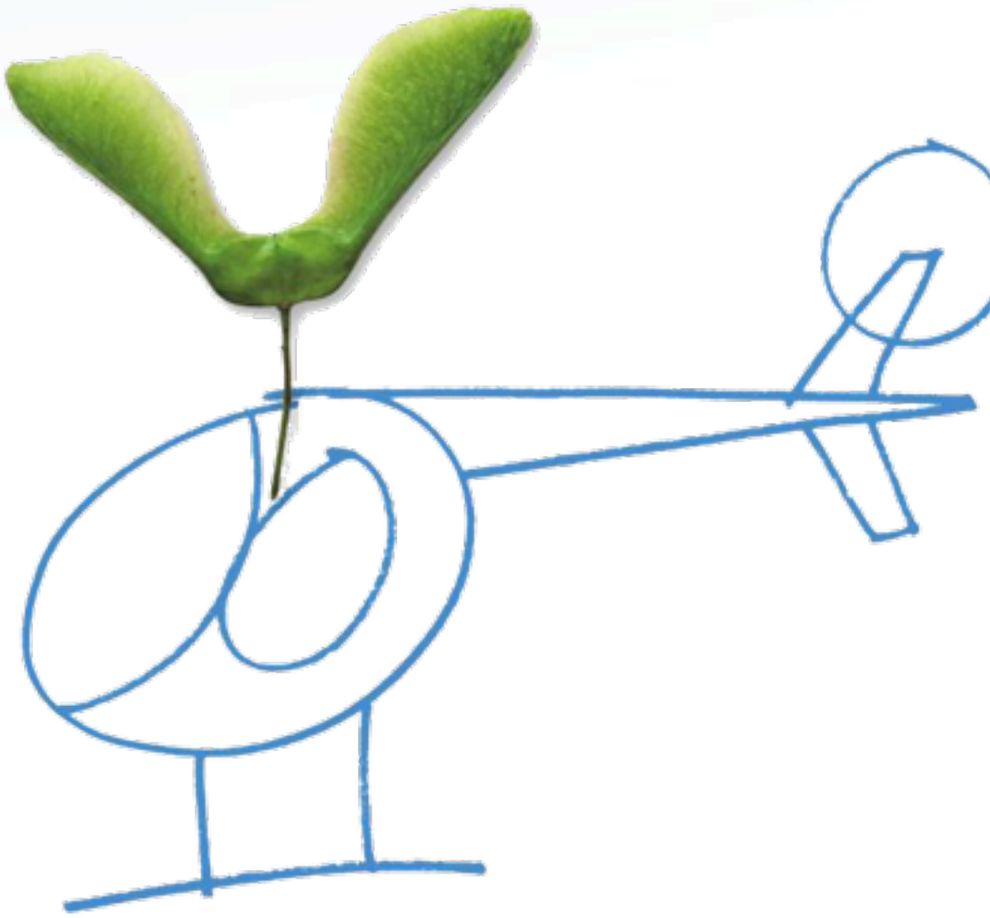
Conclusion

1. JavaScript is a full programming language
   - design patterns applicable
   - business logic realizable
2. "Endless" possibilities for client programming
3. Examples + hands on code could give and idea

Ressources

- http://javascript.crockford.com
- http://developer.yahoo.com/yui/theater/

Thank you for your attention



There is always an idea in the beginning.