

# Practical JavaScript

4mir Salihefendic  
[amix@amix.dk](mailto:amix@amix.dk)

# Who am I?

- Skeletonz CMS, GreyBox, Orango Spell and Todoist > Lots of JavaScript
- Employed at Cabo Communications as web developer working with Zimbra (Java + JavaScript)
- Employed at Galt Networks as web developer (Python + JavaScript)

# Plan

- Disco and Spin [\(demo\)](#)
- Why use JS libraries? [\(code\)](#)
- Show-case of 3 libraries [\(code\)](#)  
Prototype, jQuery and AJS
- Development in JavaScript [\(demo\)](#)
- Creating DOM smartly [\(code\)](#)
- JavaScript DOM trick [\(code\)](#)
- bind function [\(code\)](#)
- Implementing sortable lists [\(demo\)](#)

# Disco & Spin

- Google image search for disco:  
<http://images.google.com/images?q=disco>
- Paste code in address bar:  
<http://amix.dk/upload/awt/spin.txt>  
<http://amix.dk/upload/awt/disco.txt>
- JavaScript isn't used only for fluffy effects  
- anymore!

# JavaScript libraries

- Prototype (<http://www.prototypejs.org/>)
- mootools (<http://mootools.net/>)
- Dojo (<http://dojotoolkit.org/>)
- jQuery (<http://jquery.com/>)
- Yahoo UI (<http://developer.yahoo.com/yui/>)
- MochiKit (<http://www.mochikit.com/>)
- AJS (<http://orangoo.com/labs/AJS/>)
- ....!

# Why use JS libs?

- To show the power of a JavaScript library we are going to rewrite ajax-demo using AJS

# Standard JS

```
var timeout;
function buttonClicked() {
  if (http.readyState == 0 || http.readyState == 4) {
    var v = document.getElementById("x").value;
    sendRequest("http://www.brics.dk/ixwt/echo?X="+
      encodeURIComponent(v), responseReceived);
  } else {
    window.clearTimeout(timeout);
    timeout = window.setTimeout(buttonClicked, 500);
  }
}

function responseReceived() {
  if (http.readyState == 4)
    try {
      if (http.status == 200) {
        var d = document.createElement("div");
        d.innerHTML = http.responseText;
        var t = d.getElementsByTagName("table")[0];
        var r = document.getElementById("result");
        r.replaceChild(t, r.firstChild);
      } else
        alert("Error " + http.status);
    } catch (e) {
      alert(e);
    }
}
```

And this code even has some abstraction (sendRequest)!

# AJS

```
function buttonClicked(input) {  
  var d = getRequest('http://www.brics.dk/ixwt/echo');  
  d.addCallback(function(html, req) {  
    var doc = HTML2DOM(html);  
    RCN($('result'), $bytc('table', null, doc)[0]);  
  });  
  d.sendReq({X: input.value});  
}
```

7 vs. 26 lines of code

Higher abstraction = Faster coding

Libraries also solve browser inequalities

# Prototype

- Popular and widely used (RoR, Slashdot ...)
- Extends DOM and JavaScript  
particularly elements, arrays and hashes
- OOP JavaScript
- Introduced the \$ function
- No effects
- Java of Javascript
- ~50 KB
- <http://www.prototypejs.org/>

# Prototype snippets

```
//AJAX
new Ajax.Request('/profile', {
  method: 'post',
  parameters: $H({'action': 'check_username',
                 'username': $F('username')}),
  onSuccess: function (j) {
    //...
  }
});
```

```
//JavaScript OOP
var Person = Class.create({
  initialize: function(name) {
    this.name = name;
  },
  say: function(message) {
    return this.name + ': ' + message;
  }
});
```

```
//Iteration
[el1, el2].each( fn(el, i) );
```

# jQuery

- Centered around chaining and binding methods to objects and elements
- Encapsulated - i.e. does not extend
- No OOP
- Supports XPath and CSS 1-3 selectors inside the \$ function
- The Perl/Ruby of JavaScript (“slow”, but convenient)
- Simple effects included
- ~50KB
- <http://jquery.com/>

# jQuery snippets

```
//Chaining
```

```
$("#p.surprise").addClass("ohmy").show("slow");
```

```
//Iteration
```

```
$([el1, el2]).each(fn(i))
```

```
//AJAX
```

```
$.ajax({ url: '/profile',  
        data: {'action': 'check_username',  
              'username': $('#username').val()},  
        type: 'post',  
        success: function (json) {  
            //...  
        }  
});
```

```
//Effects
```

```
$("#p").click(function () {  
    $("#p").fadeOut("slow");  
});
```

# AJS

- Includes the best from MochiKit
- Aimed to build DOM (and not decorate it)
- Purely functional
- Polymorphic: The size is variable!
- Simple effects included
- Simple drag and drop abstraction
- OOP and simple \$ included
- ~40KB
- Mostly used by me :)
- <http://orangoo.com/labs/AJS/>

# AJS snippets

```
//DOM building
var span;
var p_elm = P( span = SPAN(IMG({'src': 'hottie.gif'})) );
addClass('span', 'blue_n_black');
```

```
//Iteration
map([e11, e12], fn);
```

```
//AJAX
var d = loadJSONDoc(url);
d.addCallback(
    function(o) {
        alert(o);
    });
d.sendReq({name: 'fluffy'});
```

```
//Effects
fx.setHeight($('div'), { from: 0, to: 500});
```

# Develop like a pro!

- **Firebug** > God's gift to the web developer!
- Profile and debug JavaScript!
- Quickly find errors (trace-back)
- Inspect and edit HTML (live)
- Tweak CSS
- Inspect JavaScript objects
- Evaluate JavaScript
- Monitor network
- .... and more!
- <http://www.getfirebug.com/>

# Creating DOM smartly (I)

- We would like to write:

```
<p><span class="blue_n_black"></span></p>
```

- In pure DOM:

```
var p = document.createElement("p");  
var span = document.createElement("span");  
span.className = "blue_n_black";  
var img = document.createElement("img");  
img.src = "hottie.gif";  
p.appendChild(span);  
span.appendChild(img);
```

- Hacky, hacky:

```
var p = document.createElement("p");  
p.innerHTML = '<span class="blue_n_black"></span>';
```

# Creating DOM smartly (2)

- We create some DOM functions:

```
var P = function() { return createDOM.apply(this, ["p", arguments]); };  
var IMG = function() { return createDOM.apply(this, ["img", arguments]); };  
var SPAN = function() { return createDOM.apply(this, ["span", arguments]); };
```

- createDOM is a special function -> <http://amix.dk/uploads/MagicDOM.js>

- Now we can do following:

```
P(SPAN({'class': 'blue_n_black'}, IMG({'src': 'hottie.gif'})))
```

# JavaScript DOM trick (I)

- Standard way by using DOM:

```
var table = document.createElement('table');
table.className = 'fluffy';
var tbody = document.createElement('tbody');
var tr_1 = document.createElement('tr');
var tr_2 = document.createElement('tr');
table.appendChild(tbody);
tbody.appendChild(tr_1);
tbody.appendChild(tr_2);
```

- Using AJS DOM:

```
var table = TABLE({'class': 'fluffy'});
var tbody = TBODY();
var tr_1 = TR();
var tr_2 = TR();
ACN(table, tbody);
ACN(tbody, tr_1);
ACN(tbody, tr_2);
```

# JavaScript DOM trick (2)

- Using AJS DOM + JS trick

```
var tbody, tr_1, tr_2;  
var table = TABLE({'class': 'fluffy'},  
    tbody = TBODY(  
        tr_1 = TR(),  
        tr_2 = TR()  
    )  
);
```

- Standard DOM **hides the structure** of HTML
- This trick **preserves the structure** of HTML -  
makes code much more readable!

# Scope “problem”

- The “problem”:

```
function Agent(name) {  
    this.name = name;  
}  
Agent.prototype = {  
    alertName: function() {  
        alert('Name: ' + this.name);  
    }  
}  
var fluffy = new Agent('Fluffy 007');  
document.onclick = fluffy.alertName;
```

- “name: undefined” is alerted on document click!
- The method has lost its scope!?
- But we have a solution: **bind method**

# Scope “problem”

- Purely functional implementation:

```
bind = function (fn, obj) {  
    return function () {  
        fn.apply(obj, arguments);  
    };  
}
```

- With it we can do:

```
var fluffy = new Agent('Fluffy 007');  
document.onclick = bind(fluffy.alertName, fluffy);
```

- Bind is essential when we work with  
AJAX/events!

# Scope “problem”

- Prototype based implementation :

```
Function.prototype.bind = function (obj) {  
    var method = this;  
    return function () { method.apply(obj, arguments); }  
}
```

- With it we can do:

```
var fluffy = new Agent('Fluffy 007');  
document.onclick = fluffy.alertName.bind(fluffy);
```

- Prototype vs. functional...? :-)

# Implementing sortable lists

- It's possible to create **amazing interfaces** with very few lines of code
- Sortable lists implementation in less than 100 lines:  
[http://orangoo.com/AJS/examples/sortable\\_list.html](http://orangoo.com/AJS/examples/sortable_list.html)
- Drag and drop (20 lines):  
[http://orangoo.com/AJS/examples/drag\\_n\\_drop.html](http://orangoo.com/AJS/examples/drag_n_drop.html)

Questions?