Combining advanced JavaScript/DOM techniques and Ajax to build better User Interfaces
Or:
Create Happy Users
Thomas Fuchs

- **wollzelle**
- Core team member of Ruby on Rails
- **script.aculo.us**
- **mir.aculo.us**
- Branding/Design/Web
- fluxiom Digital Asset Management
  http://www.fluxiom.com
- http://www.wollzelle.com/
JavaScript, as seen by the browser

Source: http://www.pollux.franken.de/KNF/
JavaScript Frameworks make you...
...write less code...
...code more elegantly...
…deal less with browser bugs…
...and ultimately stay happy.
Concentrate on your application...
...and not on browser bugs.
1.5

prototype
Prototype is JavaScript, reloaded
Ajax
Basic Ajax calls
(HTML snippets)
Ajax Forms
Ajax behaviours

Server-side JavaScript generation:
Prototype automatically evaluates JavaScript when Content-type: text/javascript is set

The hot topic of 2006!
Hooks for integration with UI enhancements (like activity indicators)
Extensions to built-in object types
alert('This is a test'.gsub(/s/, '(s)'));

alert('This is a long string'.truncate(10));

alert('String with <tag/>'.stripTags());

var i = 3;
i.times(function(v){ alert(v) });
Enumerables
[1,2,3].each( function(n){
    alert(n);
});

alert(['Prototype','script.aculo.us','Ruby on Rails'].join(', '));

alert([1,2,[3,4,[5,6]]].flatten().collect( function(v) {
    return v*v;
}).join('/'))
$ and $$
if($('blah').visible()) { ... }

$$('p.hint').each(Element.hide);

$$('p.hint').each(function(e){
  e.hide();
});
Direct Element extensions
Prototype 1.4

```
if(Element.visible('blah'))
Element.hide('blah');
```

VS

Prototype 1.5

```
if($('blah').visible())
$('blah').hide();
```
.update()
.replace()
.remove()
.visible()
.setStyle()
.addClassName()
(...and lots more)
Events
(without cross-browser pain)
Event.observe('some_input','blur',
    function(){ alert('blur!') }
);
cheat sheet!

http://www.snook.ca/archives/000531.php
Of course, just think of it as magic.

Sam Stephenson, author of Prototype
Visual effects
Drag and Drop
Sortable lists demo

this is the first list
- Item 1 from first list.
- Item 2 from first list.
- Item 3 from first list.

and now the second list
- DRAG HERE Item 1 from second list.
- DRAG HERE Item 2 from second list.
- DRAG HERE Item 3 from second list.

2 lines of code!
Ready-to-use controls
Real-time search
Autocompleting
Drag and Drop
Inplace editor
Slider
Plus some extras

JavaScript unit testing, DOM Builder
Docs & Help

- **Wiki:**

- **Mailing list:**

- **IRC channel:**
  `#prototype on freenode.net`
event: Selectors

var Rules = {
  '*icons a:mouseover': function(element) {
    var app = element.id;
    new Effect.BlindDown(app + '-content',
    {queue: 'end', duration: 0.2});
  },
  '*icons a:mouseout': function(element) {
    var app = element.id;
    new Effect.BlindUp(app + '-content',
    {queue: 'end', duration: 0.2});
  }
};

http://encytemedia.com/event-selectors/
Lightbox JS

http://www.huddletogether.com/projects/lightbox2/
Windows

Example
Click on the links below to pop-up windows

Open a window
Open another
Open a window with position/size
Open a modal window

I have also written a Javascript. It opens a window and you can write messages in this window with the `debug(message)` function.

You can view and/or download it here: `debug.js` or view it in live.

http://blogus.xilinus.com/pages/javawin
Visual effects
Sliding panels and homegrown effects
Advanced engine for visual effects

Time-based

Complex effects easily possible
More than just „eye candy“

Visual feedback

Loading times „feel shorter“

Can help move away from proprietary plugins
On board

Often-used effects

„Yellow Fade Technique“
Fade in and out
Sliding panels
Puff, Shake, etc.
<div id="message" style="display:none">
    Hello world!
</div>

<script type="text/javascript">
    // <![CDATA[
    new Effect.Appear('message');
    // ]]></script>
<style type="text/css" media="screen">
  *panel {
    background: #f00;
    position: absolute;
    left: 200px;
    width: 400px;
  }
  *panel div {
    border: 2px solid #f00;
    padding: 20px;
  }
</style>

<div id="panel" style="display:none"></div>
  Hello world!
</div></div>

<a href="#" onclick="Effect.toggle('panel','slide'); return false">
  Show/hide panel
</a>
christof wagner
portfolio 1 / portfolio 2 / portfolio 3 / portfolio 4
publikationen / biografie / referenzen / kontakt

zolargasse 15/12a / 1070 wien / österreich /
+43 664 185 71 46 / foto@christofwagner.com

copyright © 2006 christof wagner
A field guide to **scriptaculous**

**Combination Effects**

**The basic syntax**

- `Effect.EffectName('dom_id')`

  *effects require just a DOM ID*

- `Effect.EffectName('dom_id', {option:value, option:value})`

  *you can use options for customizing options are optional!*

**Highlight syntax**

- `Effect.Highlight('dom_id', {startcolor:'hex', endcolor:'hex'})`

  *optional, if you don’t want yellow fade to white*

  *this bit’s critical*

**Default options**

- `delay: seconds, in decimals`
  - delay before starting effect

- `from: 0.0 to 1.0 to: 0.0 to 1.0`
  - stages of the effect, 0 being no effect;
  - a Fade set to `to: .5` would fade half-way

- `duration: seconds, in decimals`

- `transition: transition method`
Do-it-yourself effects

Very easy to do

Automatically use all the features from the visual effects engine
Complete tutorial & demo code:

http://www.thinkvitamin.com/features/ajax/create-your-own-ajax-effects
Activity indicators
Show what’s going on
„There‘s something going on, wait a sec“

Revival of the animated GIF

Both „per use“ and „global“ indicators possible
„Local“ indicator for a specific request, show/hide indicator image
„Global“ indicator

```javascript
Ajax.Responders.register({
  onCreate: function() {
    if ($('busy') && Ajax.activeRequestCount > 0)
      Effect.Appear('busy', {duration: 0.5, queue: 'end'});
  },
  onComplete: function() {
    if ($('busy') && Ajax.activeRequestCount === 0)
      Effect.Fade('busy', {duration: 0.5, queue: 'end'});
  }
});
```

for all Ajax requests, appear/fade an indicator image
Need an indicator image?

AjaxLoad

Generator

Indicator type: Arrows
Background color: #FFFFFF
Foreground color: #FA8B8B

Preview

Press "Generate it"

Top 10

1. 2508
2. 1919
3. 639
4. 597

http://www.ajaxload.info/
Annoyances & Debugging
Lessons learned from a year of Ajax
Not all Browsers are created equal.
CSS
CSS can eliminate or save on server accesses

HTML sizes are reduced

Be creative, CSS is extremely capable
JavaScript performance
DOM-API can be slow
Firefox is the slowest (better in 1.5)
Caching references in JavaScript vars
\`.innerHTML\` is your friend
Venkman profiling
Ajax Security
Ajax is as secure as any browser request

So: Don‘t trust it

Always verify user submitted data on the server
Which data format?
Trivial updates: HTML

Complex updates: JavaScript

Use HTTP caching

Don‘t overengineer

be pragmatic!
Some helpful hints
Local calls only

- Ajax can only call originating server
- Also works with file:// URLs
- Use server-side proxying if you want to tie in other services
The Back button

• The browser back button works by returning the user to the last „non-ajax“ loaded page
• Keep that in mind, especially for public sites
• Workarounds available, but aren‘t working with all browsers
display: none

- Don’t use display:none in external CSS
- Browser can’t determine the original (built-in) display value of an element (like "block" for DIVs, and "inline" for SPANs)
- Use the style attribute instead:
  `<div style="display:none">...</div>`
Internet Explorer image cache

- Internet Explorer doesn’t use its cache when HTML page fragments that are inserted via Ajax are loading images (IMG tag)

- Some proprietary HTTP headers exist to work around this (hey, it’s Microsoft!):

The verboten IDs of Internet Explorer

- IE 6 has some bugs regarding naming ID attributes
- Don’t use: “length“, “item“, “namedItem“, “tags“ or “urns“

http://meyerweb.com/eric/thoughts/2005/08/29/reserved-id-values/
JavaScript objects

```javascript
new Ajax.Updater('test', 'update.php', {
  onComplete: function() {}
  ,
  insertion: Insertion.Top
  ,
});
```

This code works in Firefox, but doesn't in Safari.

Always test in all browsers you want to support!
Debugging JavaScript
No rocket surgery here
The first rule of AJAX debugging: Use Firefox
The second rule of AJAX debugging: Use Firefox
Firebug
Version 0.4 will include a lightweight JavaScript debugger
Web Developer Extension
Venkman JavaScript Debugger also does profiling!
This site demonstrates the **Ruby on Rails 0.13** AJAX helpers, which use script.aculo.us effects, drag-and-drop and autocompletion.
The third rule: Test with all browsers you want to support
Safari Web Inspector
Microsoft Script Debugger for IE

„It’s more like Web 3.0“
What can you actually do with all this?
Available now

www.fluxiom.com