Ajax - libraries

Andrea Ferracani
andreaferracani@gmail.com

Le librerie ajax sono serie di oggetti di cui possiamo accedere funzioni e proprietà che permettono di adempiere ad alcuni tasks che vanno dall'odinaria gestione del DOM a tools molto più complessi che possono gestire effetti visuali o interazioni con un server.

Daremo un'occhiata a:

- DOMAssistant - http://www.domassistant.com/
- JQuery - http://jquery.com/
- MochiKit - http://mochikit.com/

- Prototype - http://prototypejs.org/
- Yahoo user interface (YUI) - http://developer.yahoo.com/yui/
- Scriptaculous - http://script.aculo.us/

Ci concentreremo su questi aspetti:

- Gestione DOM
- Gestione eventi
- Lo stile
- Comunicare con Ajax

Quale scegliere?

- DOMAssistant - è leggero, indicato per la gestione del DOM, non ha metodi di comunicazione client server. Permette di agire sugli elementi solo 'byId' e 'byClassName'. Non sovrascrive i metodi di altre librerie se importate. Si può usare il namespace: DOMAssistant.$().
- JQuery - molti più metodi per la gestione del DOM: per id, per regole CSS e sintassi XPath anche combinati insieme. Comunicazione ed eventi. C'è una grossa community e documentazione dietro

se si usa con altre librerie come Prototype bisogna fare attenzione a non sovrascrivere un metodo comune come $ e all'inizio dello script si usa jQuery.noConflict();

- MochiKit è una libreria abbastanza completa che oltre ai metodi per il DOM, per la gestione eventi, include anche effetti e colori, drag and drop ed anche un pannello di debug. Uso: ifilter (...)
  or
  MochiKit.Iter.ifilter(...)

quando viene caricato nella pagina tutti i metodi sono registrati al namespace window. Per prevenirlo bisogna includere questo codice nell'head della pagina HTML:

```
<script type="text/javascript">
// Prevent collisions
MochiKit = {__export__: false};
</script>
<script src="/path/to/MochiKit.js" type="
text/javascript"></script>
```

- Prototype: ottimo per la manipolazione del DOM. Famosa la funzione $. Ha il difetto di non avere namespace e quindi lavora male con altre librerie.
- Yahoo user interface: è una libreria completa di tanti piccoli widgets. L'unica difficoltà è capire quale usare. Yahoo namespace.

DOM TESTING

MOCHIKIT SAMPLE

```
// MochiKit library
// Locate all the a.browser anchors within the #browserList
id.
var browserAnchors = MochiKit.DOM.
getElementsByTagAndClassName(
'a',
'browser',
MochiKit.DOM.$('browserList')
);
```

Vedi MochiKit/Mine/1_dom_man.js

DOM TESTING

YUI SAMPLE

```
var browserAnchors = YAHOO.util.Dom.
getElementsByClassName(
'browser',
'a',
YAHOO.util.Dom.get('browserList')
);
```

Vedi YUI/Mine/1_dom_man.js

DOM TESTING

DOM ASSISTANT SAMPLE

```
var browserAnchors = DOMAssistant.$$('browserList').
elmsByClass('browser','a');
```

Vedi DOMAssistant/Mine/1_dom_man.js

DOM TESTING

JQUERY SAMPLE

```
var browserAnchors = jQuery('#browserList').find('a.browser');

var value = jQuery('#browserList').find('a')[0].firstChild.
nodeValue;
```

CSS SELECTOR

```
var browserAnchors = jQuery('#browserList a.browser');
```

Vedi JQuery/Mine/1_dom_man.js

DOM TESTING

PROTOTYPE SAMPLE

```
var browserAnchors = $('browserList').
getElementsByClassName(
'browser'
).findAll(
function(e) {
return e.nodeName == 'A';
}
);
```

DOM TESTING

BYCSSSELECTOR

```
var browserAnchors = $$('#browserList a.browser');
```

Vedi Prototype/Mine/1_dom_man.js

Ogni metodo dei selettori CSS ne supporta tutta una serie:

- * - seleziona tutti gli elementi
- tag - seleziona tutti i tag elements
- tagA tagB seleziona tutti i tagB descendant di tagA
- #id e tag#id
- .className e tag.className
- combinations: #id ul li a.className

Vedi JQuery/Mine/1_dom_man.js

CSS 2.1 Attribute selectors

- tag[attr] seleziona tutti i tag che hanno quel particolare attributo
- tag[attr=value] seleziona tutti i tag con un particolare valore nell'attributo
- tag[attr~=value] seleziona tutti i tag in cui un valore è contenuto nel valore dell'attributo
- tag[attr^=value] seleziona tutti i tag in cui un valore è uguale all'inizio del valore dell'attributo
- tag[attr$=value] seleziona tutti i tag in cui un valore è uguale alla fine del valore dell'attributo
- tag[attr|=value] seleziona tutti i tag con un valore che corrisponde alla prima parte di un

valore d'attributo separato da trattino (es: article-news)/* don't works with jquery */

- tag[attr!=value] seleziona tutti i tag il cui attributo è diverso da un valore
- tagA > tagB seleziona tutti i tagB figli di tagA
- tagA + tagB seleziona tutti i tagB fratelli di tagA che lo seguono immediatamente (nextSibling)
- tagA ~ tagB seleziona i tagA che hanno un previuos sibling tagB /* non fa in JQuery */

PSEUDOCLASS E PSEUDOELEMENT SELECTORS

- tag:nth-child seleziona i tag figli alla nth posizione rispetto ai loro padri

- tag:nth-last-child lo stesso contando dal basso
- tag:nth-of-type seleziona il nth sibling del suo tipo
- tag:nth-last-of-type lo stesso dal basso
- tag:first-child
- tag:last-child
- tag:first-of-type
- tag:last-of-type
- tag:only-child seleziona il tag unico figlio del padre
- tag:only-sibling seleziona i tag che sono l'unico sibling del loro tipo
- tag:empty seleziona i tag senza figli

- tag:enabled
- tag:disabled
- tag:checked
- tag:not(s) seleziona i tag che non hanno un selettore particolare

JQUERY e XPATH

- jQuery('tag[@attr]') seleziona tutti i tag con un attributo
- jQuery('tag[@attr=value]')
- jQuery('tag[@attr^=value]') match con l'inizio del valore dell'attributo

- jQuery('tag[@attr$=value]') match con la fine del valore
- jQuery('tag[@attr*=value]') contiene il valore
- jQuery("/html/body//p")
  jQuery("/*/body//p") absolute path
- jQuery("a",this)
  jQuery("p/a",this) relative path
- jQuery("//div//p")
- jQuery("//div/p")
- jQuery("//input[@checked]")
- jQuery("//a[@ref='nofollow']")
- jQuery("p:last")
- jQuery("p:eq(0)")
- jQuery("p:lt(5)")

- jQuery("p:gt(2)")
- jQuery('ul/li') == jQuery('ul>li')
- jQuery('input[@name=street]').val();
- jQuery('input[@type=radio][@checked]')
- jQuery("p:first").css("fontWeight","bold");
- jQuery("div:hidden").show();
- jQuery("div:contains('scared')").hide();

Può essere definita una callback per selezionare degli elementi custom dato che i selettori standard selezionano di solito l'elemento più a destra e quindi pe es. non è possibile trovare tutte gli elementi che hanno una sola immagine come figli facendo return true; o return false;

```
// YUI library callback filter
var singleImageAnchors = YAHOO.util.Dom.
getElementsBy(function(e) {
// Look for A node with one child image
return (e.nodeName == 'A' &&
e.getElementsByTagName('img').length == 1);
});
```

```
// MochiKit library callback filter
var singleImageAnchors = MochiKit.Iter.ifilter(
function(e) {
// Make sure there is only one child image
return (e.getElementsByTag('img').length === 1);
},
document.getElementsByTagName('a')
);

// Prototype library callback filter
var singleImageAnchors = $$('a').findAll(function(e)
{
return (e.descendants().findAll(function(e) {
// Locate all image elements
```

```
return (e.nodeName == 'IMG');
}).length == 1);
});

// jQuery library callback filter
var singleImageAnchors = jQuery('a').filter
(function() {
// Make sure there is only one child image
return (jQuery('img',this).length == 1)
});
```

CREATING ELEMENTS

DomAssistant:

```
DOMAssistant.$(id).create(name, attr, append, content)
// Create a child <div> element in #content with an id of
// myDiv and a className of justAdded and set its content
// to the text "I'm a brand new div!":
$("content").create("div", {
id : "myDiv",
className : "justAdded"
}, true, "I'm a brand new div!");
```

JQuery:

```
// Find all the <li> elements in ul#list1 and
// relocate them to children of the ul#list2
$('ul#list1 li').appendTo("ul#list2");
```

MochiKit:

```
var newDiv = MochiKit.DOM.createDOM(
'DIV',
{'class': 'justAdded'},
'I\'m a brand new div!'
);
MochiKit.DOM.$('content').appendChild(newDiv);
```

Prototype cleanWhitespace:
i browser implementano in modi diversi la gestione degli spazi bianchi, IE considera gli spazi come figli degli elementi.

```
var node = $('example').cleanWhitespace().
firstChild
```

COLLISIONS

```
//YUI
YAHOO.util.Dom.getRegion(String | HTMLElement
| Array) restituisce top, left, bottom, right position

var region1 = YAHOO.util.Dom.getRegion('region1');
var region2 = YAHOO.util.Dom.getRegion('region2');
if(region1.intersect(region2)) {
    alert('The regions intersect!');
}
```

Vedi YUI/Mine/1_dom_man.js

ITERATING

```
// Prototype library iteration using each()
// Iterate over the list of singleImageAnchors
hasOneImage class.

singleImageAnchors.each(function(e,i){
e.addClassName('hasOneImage');
});

e = element
i = index
```

```
// jQuery

singleImageAnchors.each(function(i){
    jQuery(this).addClass('hasOneImage');
});

// MochiKit

MochiKit.Iter.forEach(singleImageAnchors,
function(e){
MochiKit.DOM.addElementClass(e,'hasOneImage');
});
```