Ajax - javascript intro

Andrea Ferracani
andreaferracani@gmail.com

Javascript è uno dei tre elementi fondamentali del web design:

- XHTML: struttura semantica del documento
- CSS: positioning e style
- DOM scripting: enhance not provides comportamento ed interazione

Enhancement or degradation sono essenziali nell'uso di javascript, perchè ajax risiede e dipende da una tecnologia.

L'accessibilità dell'informazione è la cosa primaria

Javascript è un linguaggio interpretato da un web browser e dunque risiede su software e tecnologie che possono essere molto diverse.

Dunque i requisiti fondamentali sono:

- Standard compliant
- Maintainable: facile da riusare, ad oggetti
- Accessibile, anche per chi non ha javascript enabled
- Usabile: efficace, efficiente e soddisfacente

Dunque la parte di scripting deve essere separata dala struttura del documento. Si fa come per i CSS. La sintassi giusta è questa:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External File JavaScript</title>
<script type="text/javascript" src="source.js"></script>
</head>
<body>
<h1>External File Example</h1>
</body></html>
```

Uno script importato in questo modo serve ad interagire conil DOM.

DOM: Document object model.
Non significa jvascript, è uno web standard che rappresenta il documento.

Specifiche: http://www.w3.org/DOM - Specificano gli oggetti, le funzioni e le proprietà che linguaggi come javascript devono implementare per interagire con la struttura

Esistono 3 livelli di specifiche del DOM, la prima del 1998, il core; la seconda del 2000 e una piuttosto recente divise a loro volta in numerose sottosezioni con specifiche per task. La difficoltà è che non tutti i browsers seguono o implementano le specifiche stesse.

JAVASCRIPT INCLUDE

Bisogna separare il behaviour dal markup.

Ci sono diversi modi per inserire javascript in una pagina web ma non tutti sono giusti.

Inline javascript:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Inline JavaScript</title>
</head>
```

```
<body>
<h1>Inline Example</h1>
<script type="text/javascript">
//JavaScript Code
</script>
</body>
</html>
```

ma stiamo mischiando comportamento e markup, così anche nel caso che lo mettessimo nell'head (solo a un livello minore)

es.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Head JavaScript</title>
<script type="text/javascript">
//JavaScript Code
</script>
</head>
<body>
<h1>Head Example</h1>
</body>
</html>
```

Correct method: external source file

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External File JavaScript</title>
<script type="text/javascript" src="source.js"
></script>
</head>
<body>
<h1>External File Example</h1>
</body>
</html>
```

JAVASCRIPT prefix:

Google

L'attributo target non è xhtml script, dunque uiamo js.

<a href="javascript:window.open
('http://www.google.com');">
Google

ma il prefisso può gestire solo una funzione e inoltre

se restituisce qualcosa la pagina iniziale viene sovrascritta. Possiamo usare una funzione che non ritorna niente

Vedi sample 1_start/samples/include.

Altrimenti possiamo usare il gestore onclick.

```
<a href="#" onclick="window.open('http://www.
google.com');">
google</a>
```

ma nel caso js sia disabilitato non funziona il link

Soluzioni migliori:

<a href="http://advanceddomscripting.com"
onclick="this.href=
'javascript:popup
(\'http://ww.google.com\');"">
Google

best: <a href="http://www.google.com" onclick="
popup(this.href);return false;">google

return false; previene l'azione di default che è
seguire il link

Ma cmq i nostri script sono sempre mischiati al markup, dunque dimenticare tutto ciò.

```
// Add a load event to alter the page
Mine.addEvent(window,'load',function(W3CEvent) {
// Locate all the anchor tags with the popup class
in the document
var popups = Mine.getElementsByClassName
('popup', 'a');
for(var i=0 ; i<popups.length ; i++ ) {
// Add a click event listener to each anchor
Mine.addEvent(popups[i],'click',function(W3CEvent)
```

```
{
// Open the window using the href attribute
window.open(this.href);
// Cancel the default event
Mine.preventDefault(W3CEvent);
});
}
})
```

Un buon esempio di separazione script - markup

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

VERSION CHECK

è concettualmente sbagliato il browser sniffing:

```
if ( pitcher.name == 'Addison' ) {
pitcher.screwball.throw();
} elseif ( pitcher.name == 'Stephanie' ) {
pitcher.fastball.throw();
} else {
alert( 'Sorry, you can't throw the ball.' );
}
```

se restringono le funzionalità solo per alcuni

CAPABILITY DETECTION

```
if ( pitcher.screwball ) {
pitcher.screwball.throw();
} elseif ( pitcher.fastball ) {
pitcher.fastball.throw();
} else {
alert( 'Sorry, you can't throw the ball.' );
}

quindi if (document.body) {
// code that relies on document.body
} (netscape 4 ed explorer 3 non lo supportano)
```

Il controllo sulle funzioni dipende da quello che ciascuno deve fare, non ha senso controllarle tutte.

per esempio per le funzioni del DOM

```
var W3CDOM = document.createElement &&
document.getElementsByTagName;
function
exampleFunctionThatRequiresW3CMethods() {
if(!W3CDOM) return;
// Code using document.createElement()
// and document.getElementsById()
}
```

La funzione principe del DOM

```
function $(id) {
    return document.getElementById(id);
}
```

In questo caso il proplema principale è quello del namespace. La funzione $ che ho definito potrebbe entrare in conflitto con altre. Prototype usa la stessa notazione.

Trick: una funzione anonima che si autoesegue:

```
(function(){
//your code
})();
```

Le prime due parentesi restituiscono la funzione allo script e le successive parentesi vuote la eseguono. Così ogni funzione è nello scope della anonymous function e non possono essere chiamate dall'esterno del namespace

```
(function(arg){
    alert(arg);
})('This will show in the alert box');
```

quindi:

```
(function(){
function $(id) {
return document.getElementById(id);
}
```

```
function alertNodeName(id) {
    alert($(id).nodeName);
}
window['myNamespace'] = {};
window['myNamespace']['showNodeName']
=       alertNodeName;
})();
```

tutto avviene all'interno della funzione, ne suo scope, dunque se si chiama il metodo $, si sta chiamando il metodo definito in tale funzione

window['myNamespace']['showNodeName']

così invece dò la possibilità di eseguire la funzione anche all'esterno, da altri script, ma sempre nell'ambito del mio namespace

il segreto di javascript e di ogni linguaggio di programmazione ad oggetti, è di poter definire appunto degli oggetti che racchiudono delle funzionalità senza dover ogni volta riscrivere il codice

```
(function(){
//The Mine namespace
if(!window.Mine) { window['Mine'] = {} }

function isCompatible(other) { };
window['Mine']['isCompatible'] = isCompatible;

function $() { };
window['Mine']['$'] = $;

function addEvent(node, type, listener) { };
window['ADS']['addEvent'] = addEvent;
```

```
function removeEvent(node, type, listener) { };
window['Mine']['removeEvent'] = removeEvent;

function getElementsByClassName(className, tag,
parent){ };

window['Mine']['getElementsByClassName'] =
getElementsByClassName;

function toggleDisplay(node,value) { };
window['Mine']['toggleDisplay'] = toggleDisplay;
```

```
function insertAfter(node, referenceNode) { };
window['Mine']['insertAfter'] = insertAfter;

function removeChildren(parent) { };
window['Mine']['removeChildren'] = removeChildren;

function prependChild(parent, newChild) { };
window['Mine']['prependChild'] = prependChild;
})();
```

Vedi 1_start/html

```
function isCompatible(other) {
// Use capability detection to check requirements
if( other===false
|| !Array.prototype.push
|| !Object.hasOwnProperty
|| !document.createElement
|| !document.getElementsByTagName
) {
return false;
}
return true;
}
window['Mine']['isCompatible'] = isCompatible;
```

```
if(Mine.isCompatible()) {
// Codice che usa Mine library
}



function $() {
var elements = new Array();
// Find all the elements supplied as arguments
for (var i = 0; i < arguments.length; i++) {
var element = arguments[i];
// If the argument is a string assume it's an id
```

```
if (typeof element == 'string') {
element = document.getElementById(element);
}
// If only one argument was supplied, return the
// element immediately
if (arguments.length == 1) {
return element;
}
// Otherwise add it to the array
elements.push(element);
}
// Return the array of multiple requested elements
```

```
return elements;
};
window['Mine']['$'] = $;



function addEvent( node, type, listener ) {
// Check compatibility using the earlier method
// to ensure graceful degradation
if(!isCompatible()) { return false }
if(!(node = $(node))) { return false; }
if (node.addEventListener) {
```

```
// W3C method
node.addEventListener( type, listener, false );
return true;
} else if(node.attachEvent) {
// MSIE method
node['e'+type+listener] = listener;
node[type+listener] = function(){
node['e'+type+listener](window.event);
}
node.attachEvent( 'on'+type, node[type+listener] );
return true;
}
```

```
// Didn't have either so return false
return false;
};
window['Mine']['addEvent'] = addEvent;



function removeEvent(node, type, listener ) {
if(!(node = $(node))) { return false; }
if (node.removeEventListener) {
node.removeEventListener( type, listener, false );
return true;
```

```
} else if (node.detachEvent) {
// MSIE method
node.detachEvent('on'+type, node[type+listener]);
node[type+listener] = null;
return true;
}
// Didn't have either so return false
return false;
};
window['Mine']['removeEvent'] = removeEvent;
```

http://shark.emileaben.com/cgi-bin/regex.pl

```
function getElementsByClassName(className, tag,
parent){
parent = parent || document;
if(!(parent = $(parent))) { return false; }
// Locate all the matching tags
var allTags = (tag == "*" && parent.all) ? parent.all :
parent.getElementsByTagName(tag);
var matchingElements = new Array();
// Create a regular expression to
// determine if the className is correct
```

```
className = className.replace(/\-/g, "\\-");
var regex = new RegExp("(^|\\s)" + className +
"(\\s|$)");
var element;
// Check each element
for(var i=0; i<allTags.length; i++){
element = allTags[i];
if(regex.test(element.className)){
matchingElements.push(element);
}
}
// Return any matching elements
return matchingElements;
};
```

```
}
}
// Return any matching elements
return matchingElements;
};
window['Mine']['getElementsByClassName'] =
getElementsByClassName;


function toggleDisplay(node, value) {
if(!(node = $(node))) { return false; }
if ( node.style.display != 'none' ) {
node.style.display = 'none';
} else {
```

```
node.style.display = value || '';
}
return true;
}
window['Mine']['toggleDisplay'] = toggleDisplay;


function insertAfter(node, referenceNode) {
if(!(node = $(node))) return false;
if(!(referenceNode = $(referenceNode))) return
false;
return referenceNode.parentNode.insertBefore(
node, referenceNode.nextSibling
);
```

```
};
window['Mine']['insertAfter'] = insertAfter;
```