Realizzare il drag'n'drop con DHTML

Come inserire nei nostri siti Web una funzionalità – quella del drag'n'drop – utilizzata in modo sistematico da tutti gli utenti con un PC con una qualsiasi versione di Windows o con sistemi operativi Windows like almeno nell'interfaccia

di Francesco Sblendorio e Nicola Tino

È ormai diventato una rarità trovarsi davanti a un PC che non abbia installato una versione qualsiasi di Windows o che non utilizzi sistemi operativi che ne riproducano perlomeno l'interfaccia; proprio quest'ultima caratteristica infatti ha decretato il tramonto inarrestabile di sistemi operativi e applicativi a sola interfaccia testuale, che terrorizzavano e di conseguenza allontanavano gli utenti meno esperti.

È ormai un gesto del tutto naturale trascinare una qualsiasi icona su quella del cestino di windows *per cancellarla, oppure copiare dei file da una cartella ad un'altra semplicemente selezionando e trascinando la stessa nella cartella di destinazione.*

Questa funzionalità chiamata drag'n'drop, *per la sua intuitività e facilità di esecuzione, rappresenta uno degli aspetti che, più di altri, hanno avvicinato l'utente all'utilizzo di applicativi e del PC stesso.*

A questo punto è lecito domandarsi se non sia il caso di pensare, con l'esplosione di Internet e di siti Web di tutti i tipi, all'inserimento di tale funzionalità nei nostri siti in modo da mantenere una coerenza con le operazioni che di solito l'utente compie, e fornire al nostro sito un aspetto innovativo che si distacchi in modo evidente dalle solite interfacce Web.

Svilupperemo un insieme di routine abbastanza generali per poterle utilizzare nelle nostre pagine Web.

Per avere un'idea di ciò che è possibile realizzare con questo insieme di routine si può dare uno sguardo al sito http://www.websamba.com/webpoker *o agli esempi reperibili presso ftp://ftp.infomedia.it/pub/Login/Listati/*

Queste routine si ispirano a quelle pubblicate in [1].

Compatibilità con i vari browser

I due principali browser presenti sul mercato (Internet Explorer e Netscape Navigator) hanno differenti implementazioni di DHTML, in quanto hanno un differente modello ad oggetti del documento (DOM). È attraverso questo modello che ci si riferisce alla varie parti che compongono una pagina Web: immagini, *layer* (o livelli), *form*, ecc.

Nel nostro caso utilizzeremo pesantemente ciò che Netscape chiama *layer*, ovvero porzioni di documento che possono essere manipolate, spostate, fatte sparire e ricomparire indipendentemente dal resto del contesto.

Dobbiamo fare in modo di riconoscere su quale browser stiamo rendendo la pagina ed agire diversamente a seconda di questo. Cominciamo quindi a costruire un file javascript, denominato *crossbrowser.js*, che si occuperà della gestione dei cosiddetti *layer* e potrà essere utilizzato in una pagina Web inserendo le seguenti righe nella sezione *<HEAD>...</HEAD>*:

```
<script type="text/javascript"
                src="crossbrowser.js"></script>
```

Le prime righe di questo file sono quelle presenti nel **Listato 1**.

LISTATO 1

Individuazione del browser

```
layerRef="";
styleRef="";
isN4=0;
isIE=0;
isN6=0;
// document.all                  => EXPLORER 4 o
                                    superiore
// document.getElementById => NETSCAPE 6
// document.layers               => NETSCAPE 4
if (document.all)
{
  layerRef=".all['";
  styleRef="'].style";
  isIE=1;
}
else if (document.getElementById)
{
  layerRef=".getElementById('";
  styleRef="').style";
  isN6=1;
}
else // (document.layers)
{
  layerRef=".layers['";
  styleRef="']";
  isN4=1;
}
```

LISTATO 2

Esempio di dichiarazione dei livelli

```
// FRAMMENTO DI CODICE CSS
DIV.livello
{
 position: absolute;
}

// FRAMMENTO DI CODICE
                              HTML

...

<div id="foto" class="livello">
  <img src="fotografia.gif">
</div>

<div id="toolbar" class=
                          "livello">
  <img src=
          "barrastumenti.gif">
</div>
```

Notiamo in particolare le variabili *isIE*, *isN4*, *isN6*. Come si intuisce, dalla fine del **Listato 1**, saranno tutte a zero tranne una, che identifica il browser (rispettivamente *Internet Explorer*, *Netscape Navigator 4.x*, *Netscape Navigator 6.x*). Infatti anche le due differenti release di Navigator implementano un differente modello del documento, in quanto Netscape 4 implementa specifiche fortemente proprietarie (all'epoca della sua uscita, il W3C non aveva ancora pubblicato le specifiche oggi considerate da tutti standard). Con l'uscita di Netscape 6 le cose sono cambiate e le specifiche W3C sono seguite molto da vicino (e ciò che andava bene con la versione 4 è al 100% *incompatibile* con la 6).
Vediamo ora come è stato effettuato il riconoscimento: non viene esaminata una stringa con il nome del browser; vengono piuttosto testate alcune proprietà del DOM

- la proprietà "document.layers" esiste solo in Netscape 4.x
- la proprietà "document.all" esiste in Internet Explorer 4.x ed Internet Explorer 5.x
- la proprietà "document.getElementById" (W3C) esiste in Internet Explorer 5.x e Netscape 6.x

Come si può notare, Explorer 5.x è compatibile all'indietro, quindi effettuiamo prima il test su *document.all* (si individua Explorer >=4 e si esce), poi *document.getElementById* (si individua Netscape 6 e si esce), infine se ho fallito, significa che il browser è Netscape 4. Le variabili *layerRef* e *styleRef* saranno utilizzate nella funzione *obj()* descritta in seguito.

Polling degli eventi

Proseguendo nella scrittura di *crossbrowser.js* troviamo le seguenti istruzioni:

```
navigatorCaptureEvents()

function navigator
                    CaptureEvents()
{
 if (isN4)
 { document.captureEvents
          (Event.MOUSEDOWN |
           Event.MOUSEMOVE |
           Event.MOUSEUP ); }
}
```

Questa è una funzione eseguita solo con Netscape Navigator 4.x ed impone al browser di fare il polling degli eventi *onMouseDown*, *onMouseMove*, *onMouseUp*. Negli gli altri browser ciò viene fatto in automatico (con il termine "polling degli eventi" si indica un loop potenzialmente infinito, interrotto dal verificarsi di un certo evento. Questo evento può essere la pressione di un tasto, il movimento del mouse, ecc.).

Come dichiarare i *layer*

Vediamo ora come dichiarare i *layer* con i quali dobbiamo lavorare: un esempio è nel **Listato 2**. Si tratta di un frammento di codice CSS in cui si dichiara una classe con la proprietà *position: absolute*; successivamente, bisogna indicare nell'HTML il contenuto dei layer; per un esempio vedere sempre il **Listato 2**. Notare che si è evitato l'utilizzo del tag proprietario *<LAYER>* (Netscape 4).

Referenziare un *layer*

Per poter utilizzare un *layer* è necessario referenziarlo. Come abbiamo detto sopra, i vari browser parlano "lingue" diverse. Il problema è risolto costruendo una stringa-comando usando le variabili *layerRef* e *styleRef* di cui sopra e passando la stringa così costituita alla funzione JavaScript *eval*. La funzione che fa tutto ciò è stata chiamata *obj()* ed è riportata nel **Listato 3**. Essa riceve in input il *nome* di un *layer* ed, a seconda del browser, restituisce (supponendo che abbia in ingresso la stringa "foto"):

- Internet Explorer: obj('foto') = document.all['foto'].style
- Netscape 4.x: obj('foto') = document.layes['foto']
- Netscape 6.x (W3C): obj('foto') = document.getElementById('foto').style

In questo modo possono essere referenziate in maniera uniforme

LISTATO 3

Funzione di referenziazione di un livello

```
// Restituisce il layer con nome "layerName"
function obj(layerName)
{

 if (layerName == "")
   return null;
 else
 {
   return eval("document"+layerRef+layerName+styleRef);
 }
}
```

LISTATO 4

Funzioni generiche per l'utilizzo dei livelli

```
// mostra il layer "layerRef"
function showLayer(layerRef)
{
 layerRef.visibility = 'visible';
}

// nasconde il layer "layerRef"
function hideLayer(layerRef)
{

 layerRef.visibility = 'hidden';
}

// muove il layer "layerRef" ad (absX,absY)
function moveTo(layerRef, absX, absY)
{
 if (isN6)
 {
  layerRef.left = absX+"px";
  layerRef.top  = absY+"px";
 }
 else
 {
  layerRef.left = absX;
  layerRef.top  = absY;
 }
}
```

(attraverso il nome del *layer*) le *proprietà*: ad esempio posso indicare (o assegnare) la distanza dal bordo sinistro di un layer con *obj('nomelayer').left*.
Completano la nostra mini-libreria di compatibilità le funzioni *hideLayer()*, *showLayer()* e *moveTo()*, riportate nel **Listato 4**. Tutte e tre queste funzioni ricevono come parametro di input un *riferimento a layer* (non il nome, in quanto il riferimento è ottenibile tramite la funzione *obj()*). Riserviamo particolare attenzione a *moveTo()*, che se riconosce come browser Netscape 6, concatena la stringa "px" (ovvero *pixel*) ai parametri numerici che riceve: infatti Netscape 6 richiede anche le unità di misura.
Una piccola nota: se si vuole utilizzare il *drag'n'drop* bisogna riposizionare tutti i layer da trascinare (e tutti quelli che dovranno "catturare" altri layer) tramite javascript e la funzione *moveTo()*, tipicamente nell'evento *onLoad* di *<BODY>*. Questo in particolare è necessario in Netscape 6 ed Internet Explorer.

Drag'n'Drop

Una volta affrontata e risolta la problematica della scrittura di codice compatibile per i due maggiori browser in circolazione, possiamo passare all'illustrazione del codice che implementerà tale funzionalità in un qualsiasi documento Web. In primo luogo, per una maggiore comprensione del codice stesso, descriviamo le operazioni da compiere all'interno del documento Web. Terminata tale fase introduttiva, passeremo all'illustrazione della libreria *"dragndrop.js"*.

Preparazione del documento Web per l'utilizzo del drag'n'drop

La prima operazione da compiere è quella di includere le librerie *"crossbrowser.js"* e *"dragndrop.js"* nel documento.
Fatto questo, si passerà all'individuazione dell'insieme dei layer sui quali il *drag'n'drop* dovrà agire e dell'insieme dei layer che invece avranno il compito di catturare ed eventualmente attivare un servizio, nel momento in cui un layer verrà rilasciato su di esso. Per definire questi due insiemi, è necessario dichiarare due array:

- *floatingElements*, in cui saranno elencati i nomi dei layer che usufruiranno del *drag'n'drop*
- *blackHoles*, che invece conterrà i nomi dei layer che cattureranno e faranno partire eventuali servizi associati al rilascio.

Un aspetto importante di quanto detto è che non è richiesta una *pre-dichiarazione* di tali livelli in un eventuale foglio di stile associato al documento; non è necessario così sapere a priori quanti livelli devono essere presenti nel documento.
Ciò che invece è necessario definire in un foglio di stile (esterno o interno) associato al documento, è una classe nella quale inserire la proprietà *position:absolute*. Essa forza il layer a posizionarsi esattamente alle coordinate (top, left) ad esso associate.
Un'altra dichiarazione necessaria è quella di una variabile, *ray*, che individui per i blackhole un quadrato virtuale entro il quale far catturare il livello (una volta "oltrepassato" tale quadrato, si dovrà far tornare il livello alla sua posizione originale).
Terminata tale fase, è necessario definire le posizioni dei layer: a tal proposito si definisce una funzione *init()* che è richiamata dall'evento *OnLoad* del *<BODY>*. Essa assegnerà le posizioni dei vari layer utilizzando la funzione moveTo precedentemente illustrata; tale funzione dovrà inoltre *obbligatoriamente* assegnare ai vari layer la proprietà *zIndex*.
Una ulteriore funzione da definire nel documento è quella denominata *holeHandler*; essa infatti è invocata al momento in cui si rilascia un layer e quindi è utilizzata per personalizzare le operazioni da compiere al verificarsi di tale evento.
L'ultima operazione da eseguire è chiaramente la definizione dei layer nella pagina.
Il **Listato 5** illustra un esempio di pagina HTML nella quale è utilizzato il *drag 'n' drop*.

LISTATO 5

Documento web che usa il drag'n'drop

```
<html>
  <head>
    <title>Prova Drag'n'Drop</title>
      <style>
        div.livello
          {
            position:absolute;
          }
      </style>

      <script type="text/javascript" src="crossbrowser.js"></script>
      <script type="text/javascript" src="dragndrop.js"></script>
      <script type="text/javascript">

       var blackHoles=['hole1','hole2','hole3'];
       var floatingElements=['card1','card2','card3'];
       var ray=30;

       function init()
         {
           moveTo(obj("hole1"),20,150);
           moveTo(obj("hole2"),110,150);
           moveTo(obj("hole3"),200,150);
           moveTo(obj("card1"),20,20);
           moveTo(obj("card2"),110,20);
           moveTo(obj("card3"),200,20);
           obj("hole1").zIndex = 0;
           obj("hole2").zIndex = 0;
           obj("hole3").zIndex = 0;
           obj("card1").zIndex = 1;
           obj("card2").zIndex = 1;
           obj("card3").zIndex = 1;
         }

       function holeHandler()
         {
          alert("Hai spostato il layer "+returnedElementName+
                                          " sul layer "+selectedHole);
         }

</script>
</head>

<body onload="init()">
   <div class="livello" id="card1">
     <img src="fiori11.gif">
   </div>

   <div class="livello" id="card2">
     <img src="cuori12.gif">
   </div>

   <div class="livello" id="card3">
     <img src="picche1.gif">
   </div>

   <div class="livello" id="hole1">
     <img src="hole.gif">
   </div>

   <div class="livello" id="hole2">
     <img src="hole.gif">
   </div>

   <div class="livello" id="hole3">
     <img src="hole.gif">
   </div>

</body>

</html>
```

Sviluppo del codice per il Drag'n'Drop: preliminari

Passiamo ora all'illustrazione della libreria *dragndrop.js*. La prima sezione di codice presente è la seguente:

```
var selectedElement
var selectedElementName="";
var returnedElement;
var returnedElementName;
var selectedHole="";
var offsetX, offsetY;
var old_X, old_Y, old_Z

document.onmousedown = grabElement
document.onmousemove = dragElement
document.onmouseup = releaseElement
```

Come è possibile notare si dichiarano una serie di variabili globali, che vengono utilizzate dalle varie funzioni implementate nella libreria. La parte di codice che invece merita una spiegazione è quella rappresentata dalle ultime tre istruzioni: qui infatti si comunicano al browser le funzioni da richiamare quando l'utente effettua determinate azioni.
In particolare si sta comunicando di invocare la funzione *grabElement* quando si preme il pulsante sinistro del mouse, la funzione *dragElement* quando si sposta il mouse sullo schermo, e la funzione *releaseElement* quando invece si rilascia il pulsante (sinistro) del mouse.

Sviluppo del codice per il Drag'n'Drop: selezione di un layer

La prima funzione implementata nella libreria prende il nome di *setSelectedElement*, ed ha il compito di individuare il livello che l'utente seleziona con il mouse.
Sono evidenti tre sezioni di codice: una per Netscape 4, una per Internet Explorer e una per Netscape 6. La sezione riservata a Netscape 4 opera nel seguente modo: si memorizzano in due variabili (*clickX*, *clickY*) il punto in cui l'utente ha cliccato con il mouse; tali coordinate sono ottenute dalle proprietà *pageX*, *pageY* dell'oggetto *evt* passato per parametro; l'oggetto *evt* è una

LISTATO 6

Gestione del trascinamento e rilascio di un livello

```
function grabElement(evt)
{
 setSelectedElement(evt)
 if (selectedElement)
 {
  if (isN4 || isN6)
  {
   offsetX = evt.pageX - parseInt(selectedElement.left);
   offsetY = evt.pageY - parseInt(selectedElement.top);
  }
  else
  {
   offsetX = window.event.offsetX;
   offsetY = window.event.offsetY;
  }
  return false; // se devo trascinare
 }
 else
  return true;  // se non devo trascinare
}

function dragElement(evt)
{
 if(selectedElement)
 {
  if (isN4 || isN6)
  {moveTo( selectedElement, (evt.pageX - offsetX),
                                          (evt.pageY - offsetY) );

  }
  else
  {
   moveTo( selectedElement, (window.event.clientX - offsetX),
                                  (window.event.clientY - offsetY));
   return false;
  }
 }
 else
  return true;
}

function releaseElement(evt)
{
 if (selectedElement)
  if (isElementCaptured())
  {
   selectedElement.zIndex = old_Z;
   returnedElement = selectedElement;
   selectedElement = null;
   holeHandler();
  }
 selectedElement = null;
}
```

istanza della classe che gestisce per l'appunto gli eventi scatenati dall'utente. Successivamente la funzione controlla con un ciclo che tale punto appartenga ad uno dei layer elencati nel vettore *floatingElements* privilegiando quelli con *zIndex* più alto. In caso affermativo, si procede all'assegnazione di tale livello alla variabile *selectedElement*, in modo che possa essere eventualmente manipolato da altre funzioni, mentre si assegna alla variabile *selectedElementName* il nome del layer selezionato.

Del tutto simile è la porzione di codice dedicato a Netscape 6; la differenza si trova nella condizione che verifica se il punto selezionato sia interno o meno al layer in esame. In Netscape 6 infatti, non è presente il metodo *"clip"* utilizzato invece da Netscape 4 per indicare l'area in ampiezza e altezza di un layer. Inoltre le proprietà *left* e *top* dei livelli in Netscape 6 sono stringhe (contengono anche l'unità di misura; esempio "120px"); quindi, nel leggerle è necessario un *parseInt* per renderle numeriche.

Del tutto diversa è invece la sezione di codice dedicata ad Internet Explorer; per scoprire quale elemento è stato selezionato si deve richiamare la proprieta *srcElement* dell'oggetto *window. event*, che fornisce l'elemento che ha generato l'evento (nel nostro caso il clic del mouse). Successivamente, con la proprietà *style* dell'oggetto *imgObj.parentElement*, si ottiene il livello attuale, che sarà assegnato alla variabile *selectedElement*. Il nome del layer selezionato è ottenuto invece con l'invocazione della proprietà *"id"* dell'oggetto *imgObj.parentElement*, e tale valore sarà assegnato alla variabile *selectedElementName*; si salvano infine alcune informazioni sul layer individuato, ad esempio la sua posizione di partenza, il suo *zIndex*, e tale valore è aumentato (ponendolo uguale a 100) in modo che nel trascinamento il layer nel suo spostamento risulti sempre in primo piano rispetto a tutto il resto del documento. Quanto illustrato è presente nel **Listato 8**, scaricabile dal sito Ftp://ftp.infomedia.it/pub/Login/listati

Sviluppo del codice per il *Drag'n'Drop*: determinare le coordinate del livello

La prossima funzione illustrata è quella che prende il nome di *grabElement*; come già detto, essa è richiamata automaticamente dal browser ogni volta che l'utente preme il tasto sinistro del mouse. Il suo scopo è quello di determinare quale livello è stato selezionato e qual è lo scarto del cursore: il livello selezionato è determinato dalla funzione *setSelectedElement* alla quale si passa come parametro l'oggetto *evt*.

Una volta individuato il layer, se il browser utilizzato è Netscape si calcola lo scarto del cursore facendo la differenza tra il punto dello schermo su cui si è cliccato

LISTATO 7

Cattura di un livello

```
function isElementCaptured()
{
 lunVect=blackHoles.length;
      captured=false;
 sE_top=parseInt(selectedElement.top);   //sE= selectedElement
 sE_left=parseInt(selectedElement.left);
      while (lunVect > 0 && captured == false )
       {
  testObj = obj(blackHoles[lunVect-1])
  if (testObj) bH_top=parseInt(testObj.top);     //bH= blackHole
                                                              attuale
  if (testObj) bH_left=parseInt(testObj.left);

  if (testObj && (sE_top >= bH_top - ray) &&    //condizione di
                                                              cattura
              (sE_top <= bH_top + ray) &&
              (sE_left >= bH_left - ray) &&
              (sE_left <= bH_left + ray) )
  {
   selectedHole = blackHoles[lunVect-1];
   captured = true;
                 selectedElement.top = testObj.top;
                 selectedElement.left = testObj.left;
                selectedElement.zIndex = parseInt(testObj.
                                                          zIndex) + 1;
  }
  else
   lunVect—;
 }
      if( !captured)
 {
  selectedElement.top  = old_X;
  selectedElement.left = old_Y;
            selectedElement.zIndex = old_Z;
 }
      return captured;
}
```

e la posizione del layer; se invece il browser usato è Internet Explorer, tale calcolo è eseguito in automatico utilizzando le proprietà *offsetX*, *offsetY* dell'oggetto *window.event*; la funzione restituisce *false* se è possibile trascinare il livello, *true* altrimenti.

Sviluppo del codice per il *Drag'n'Drop*: trascinamento dei livelli

Un'altra funzione presente nella libreria è quella denominata *dragElement*. In tale funzione, se è stato selezionato un layer (ovvero se *selectedElement* è diverso da *null*), si sposta il layer nella posizione ottenuta dalla differenza tra la posizione cliccata dall'utente e lo scarto del cursore precedentemente determinato. Anche in questa funzione vi è la distinzione tra i due browser; la differenza consiste infatti nella modalità di acquisizione del punto cliccato sullo schermo (*evt.pageX* e *evt.pageY* per Netscape, *window.event.clientX* e *window.event.clientY* per Internet Explorer).

Sviluppo del codice per il *Drag'n'Drop*: gestire il rilascio dei livelli

Infine l'ultima funzione implementata è quella che è richiamata in automatico nel momento in cui l'utente rilascia il tasto sinistro del mouse; essa non fa altro che verificare l'avvenuta selezione di un layer e il suo bloccaggio da parte di uno degli elementi dell'insieme *blackHole*. In caso affermativo, si invoca la funzione *holeHandler* e si riporta il layer selezionato al suo originale valore di *zIndex*.
Infine, si effettua il salvataggio del riferimento del layer in una variabile (*returnedElement*) e del suo nome (*returnedElementName*). Quanto illustrato negli ultimi tre paragrafi è codificato nel **Listato 6**.

Sviluppo del codice per il Drag'n'Drop: cattura dei livelli

La funzione che controlla la cattura di un layer da parte di un *blackHole* prende il nome di *isElementCaptured*.
I passi eseguiti da tale funzione sono i seguenti:

1. si prelevano il *top* e il *left* del layer selezionato
2. per ciascun elemento appartenente all'insieme *blackHole*, si verifica che il rilascio sia avvenuto all'interno del "quadrato virtuale" tracciato intorno a ogni elemento di *blackHoles*
3. se il punto 2 ha esito positivo, si effettua il salvataggio del nome del layer (nella variabile *selectedHole*) su cui è stato effettuato il rilascio, e si blocca il layer spostato nella stessa posizione del layer bloccante
4. se il punto 2 ha esito negativo, si riporta il layer al punto di partenza.

La funzione ritorna *true* se è stato effettuato il bloccaggio, *false* altrimenti. Il codice di questa funzione è illustrata nel **Listato 7**.

Conclusioni

Questo è quanto basta per l'utilizzo del *drag'n'drop* in un documento Web, un servizio che oltre ad essere gradevole da un punto di vista grafico, è molto utile e intuitivo per chiunque voglia accedere al vostro sito e sfruttarne le funzionalità offerte.

Bibliografia

[1] Dan Livingston, Micah Brown – *CSS e DHTML* – ed. Tecniche Nuove – 2000 – ISBN 88-481-1082-7

Francesco Sblendorio è laureando in Informatica presso l'Università degli Studi di Bari. Si occupa di programmazione in C/C++, Visual Basic e di problematiche legate al Web. Può essere contattato tramite e-mail all'indirizzo sblendorio@infomedia.it

Nicola Tino è studente di informatica presso l'Università degli Studi di Bari. Il suo interesse principale è lo sviluppo di interfacce adattive in applicazioni Web-based. Può essere contattato tramite e-mail all'indirizzo tino@infomedia.it