Plugin jQuery per gestire le mappe di Google

Autore: **Lorenzo Forti** (lorenzo.forti@gmail.com)

Il codice che segue permette di creare una mappa con le API Google Maps partendo da un array di indirizzi. L'array da costruire può contenere sia coordinate latitudine/longitudine che indirizzi da geocodificare. L'unica dipendenza richiesta è jQuery.

Rispetto ai precedenti due script analizzati in precedenza (gui e gui), questo codice non ha dipendenze da plugin di terze parti (sono necessari jQuery e Google Maps ovviamente). Molte le opzioni gestite mentre la funzione più importante è che gli indirizzi geocodificati vengono gestiti senza restrizioni.

Infatti, mentre non c'è nessun problema di prestazioni se si hanno già latitudine e longitudine, bisogna tenere conto dei limiti imposti da Google per la geocodifica. Lo script, nello specifico, imposta un timeout tra una chiamata e l'altra ed offre la possibilità di scegliere il numero massimo di tentativi per ogni indirizzo.

Una demo è presente in questa pagina. Il debug è stato lasciato attivo quindi potete aprire la console "strumenti per sviluppatori" di Google e leggere ciò che viene fatto. Mi raccomando di non usare Firebug perchè, per un bug noto, Google Maps dà errore.

Per scaricare il plugin potete cliccare qui.

Come si può notare possono essere istanziate più mappe nella stessa pagina. Le opzioni di default sono le seguenti:

```
// parametri di default
var defParams = {
 centerFirstItem   : false,
 mapAutoFit   : true,
 mapAsyncFit   : true,
 mapEmptyShow   : false,
 exposeMap   : true,
 zoom    : 5,
 queryTimeout   : 300,
 queryAttemps   : 5,
 infoMaxWidth   : "",
 defaultAddrText   : "",
 labelOnMarker   : false,
 labelList   : "ABCDEFGHIJKLMNOPQRSTUVWXYZ123456789",
 labelColor   : "#000",
 labelSize   : "16px",
 iconCommUrl   : "",
 iconSizeW   : 31,
 iconSizeH   : 40,
 templateOpen   : "",
 templateClose   : "",
        callbackName   : "",
 center    : {lat: 41.890251, lng: 12.492373},
 mapTypeControl   : true,
 mapTypeControlOptions   : {position: google.maps.ControlPosition.TOP_RIGHT},
 navigationControl  : true,
 mapTypeId   : google.maps.MapTypeId.ROADMAP,
 isDebug    : false
};
```

Sono abbastanza "parlanti" quindi non mi soffermerò troppo sulla loro spiegazione. L'unica che voglio affrontare è "exposeMap".

Se impostata a "true" permette di rendere "pubblico" l' oggetto mappa istanziato. Utile in quei casi in cui si voglia integrare il codice con delle funzioni che avete già oppure sia necessaria fare il resize di una mappa da un link "esterno" al plugin. Infatti uno dei problemi di Google Maps è che le mappe non funzionano se inserite dentro un div che inizialmente ha proprietà "display: none".

Ad esempio prendiamo queste due callback:

```
$.fn.drawMap().callbackResize('#map_canvas3');
var miamappa = $.fn.drawMap().getMapObj('#map_canvas3')
```

La prima può essere agganciata a qualsiasi elemento che ha la funzione di mostrare un div inizialmente nascosto. Così facendo, una volta che il div assume la proprietà "display: block" (o comunque diversa da "none") viene fatto il resize della mappa contenuta del div passato come riferimento e, di fatto, la mappa stessa diventa visibile.

La seconda callback sever a mettere dentro una variabile l'oggetto della mappa contenuta del div passato come riferimento. In questo modo potete interagire con la mappa attraverso le vostre funzioni.

Infine ecco un esempio di come caricare correttamente sia le API Google Map in maniera asincrona che la callback che istanzia la mappa.

Caricamento API Google e plugin:

```
 var gskey = 'VOSTRA_GOOGLE_KEY';
 $.getScript('http://maps.googleapis.com/maps/api/js?key=' + gskey +
'&callback=$.fn.callbackMap');
```

Caricamento Callback:

```
(function($) {
/**
* $.fn.callbackMap
*
* callback senza parametri che richiama la drawMap con parametri
*/
$.fn.callbackMap = function() {

 $("#map_canvas1").drawMap({
  "mapAutoFit": false,
  "centerFirstItem": true,
                "callbackName": function() { //VOSTRA CALLBACK },
  "markers":
   {data: "indirizzo 1", lat: 41.88249, lng: 12.53575, icon:
"http://www.url.it/marker-custom.png"},
   {data: "indirizzo 2", lat: 41.87531, lng: 12.62019},
   {data: "indirizzo 3", lat: 41.87905, lng: 12.56401},
   {data: "indirizzo 4", lat: 41.87943, lng: 12.56812},
   {data: "indirizzo 5", lat: 41.87967, lng: 12.56914},
   {data: "indirizzo 6", lat: 41.88167, lng: 12.56357},
   {data: "Via Laurentina 100", lat: "", lng: "", address:"Via Laurentina 100, Roma"},
   {data: "Via Flaminia 100", address:"Via Flaminia 100, Roma, Italia"},
   {data: "Via Trionfale 100", address:"Via Trionfale 100, Roma, Italia"},
   {data: "Via Cola di rienzo 100", lat: "", lng: "", address:"Via Cola di rienzo 100, Roma,
Italia"},
   {data: "Via Appia 100", lat: "", lng: "", address:"Via Appia 100, Roma, Italia"},
   {data: "Via Salaria 100", lat: "", lng: "", address:"Via Salaria 100, Roma, Italia"},
   {data: "Via Mondello 100",c, address:"Via Mondello 100, Roma, Italia"}
  ]
```

```
 });

};

})(jQuery);
```

Quando avete il parametro "**address**" potete omettere o lasciare vuoto i parametri "**lat**" e "**lng**". È indifferente.

Con queste semplici regole css le mappe sono anche responsive. È solo un esempio di come possono essere fatte:

```
.container {
 margin: 0 auto;
 width: 80%;
 height: 150px;
}

.mappa {
 margin: 0;
 padding: 0;
 width:100%;
 height: 100%;
}

/* test media query per responsive */
@media all and (min-width: 768px) {

 .container {
  width:60%;
  height: 450px;
 }
}
```