JQuery/Versione stampabile

Wikibooks, manuali e libri di testo liberi.
< JQuery

Indice

1 Introduzione
  1.1 Per cominciare
  1.2 Le basi di jQuery
2 Selettori
  2.1 I selettori
  2.2 I filtri di selezione
3 Le actions
  3.1 Le actions
  3.2 DOM Treversing
4 Gestire gli eventi
  4.1 Gestire gli eventi
5 Effetti
  5.1 Gli effetti
  5.2 Visibilità
  5.3 Dissolvenza
  5.4 Scorrimento
  5.5 Animazioni di elementi HTML
  5.6 Interrompere un effetto
  5.7 La funzione di Callback
6 Modificare il DOM
  6.1 Manipolare il contenuto dei tag
  6.2 Modificare gli attributi
  6.3 Aggiungere elementi al DOM
  6.4 Manipolare le proprietà CSS degli elementi
  6.5 Dimensioni degli elementi e della window
7 jQuery per AJAX
  7.1 Note
8 Crediti

Introduzione

Per cominciare

La libreria jQuery è immagazzinata in un singolo file JavaScript, che contiente tutti i metodi jQuery, e che può essere scaricata dal sito jquery.com in due formati:

- development: non compresso, da 252Kb, utilizzabile in fase di sviluppo
- production: compresso, da 32Kb, da utilizzare online, che garantisce una notevole leggerezza per il server.

Non è necessaria nessuna installazione, è sufficiente includere il file "`jquery-versione.js`" direttamente nella pagina web in questo modo:

```
<script src="jquery-versione.js"></script>
```

avendo cura di inserire il codice all'interno della sezione `<head>`.

In alternativa è possibile, senza scaricare il package, includere un riferimento alla libreria disponibile in remoto sul sito ufficiale del progetto:

```
<script src="http://code.jquery.com/jquery-latest.js"></script>
```

Oppure richiamare, sempre da remoto, la libreria ospitata sui server di Google:

```
<script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery.min.js"></script>
```

o di Microsoft:

```
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.0.js"></script>
```

Il vantaggio di richiamare da remoto la libreria è che l'utente potrebbe già aver scaricato il file in questione nella cache del browser e per questo gli verrebbe evitato di doverlo riscaricare con conseguente diminuzione del tempo di caricamento della pagina.

Le basi di jQuery

I comandi jQuery si basano sull'assunto che il documento HTML contiene oggetti che possono essere richiamati attraverso i loro id o classi, esattamente come in CSS.

La sintassi di base è `$(selector).action()`

- il simbolo del dollaro è un alias per jQuery
- il selettore, tra parentesi tonde, serve a selezionare elementi HTML
- l'action è l'azione che si vuole applicare all'elemento selezionato

Essendo $ un alias, sarebbe possibile usare anche la sintassi jQuery(), ma per brevità si usa $().

È buona pratica di programmazione far eseguire il codice jQuery dopo che la pagina sia stata caricata. Per questo ci viene in aiuto la funzione $(document).ready(function())

```
$(document).ready(function(){
   // qui vanno i metodi jQuery...
});
```

che è un listener di eventi che viene richiamato quando il DOM è pronto. Questo permette anche di inserire il codice Javascript (quindi anche jQuery) prima del tag body, nella sezione head. E' possibile usare anche questa sintassi ultracompatta:

```
$(function(){
   // metodi jQuery...
});
```

Alternativamente si può usare il metodo $(window).load(): la differenza sta nel fatto che questo viene richiamato quando tutti gli elementi (anche quelli grafici) sono stati già richiamati, e non solo il document. Infatti i browser basati su webkit (Chrome e Safari) "sanno" la dimensione di un'immagine solo dopo che questa è stata caricata.

Per rendere più facile la manutenzione del codice, è consigliabile creare un file separato con estensione .js dove mettere tutti i metodi jQuery, e richiamarlo nella sezione head di ogni pagina che ne fa uso in questo modo:

```
<script src="my_jquery_functions.js">
```

Selettori

I selettori

I selettori permettono di selezionare e manipolare gli elementi HTML. Gli elementi possono essere identificati attraverso l'id, la classe, gli attributi, i valori degli attributi e altro ancora.

Vediamo come avverrebbe questo con Javascript e confrontiamolo con jQuery:

```
document.getElementById("id_element");
// ho utilizzato Javascript
$("#id_element");
// ho utlilizzato jQuery
```

È evidente come il codice jQuery sia più agile e compatto.

La sintassi è la stessa dei selettori CSS, con in più alcuni selettori specifici.

Per selezionare tutti gli elementi che hanno lo stesso tag è sufficiente quindi usare come selettore il tag stesso:

```
$("p")
```

Ad esempio:

```
$(document).ready(function(){
   $("button").click(function(){
      $("p").hide();
    });
});
```

quando l'utente clicca sul bottone tutti gli elementi paragrafo vengono nascosti.

Il selettore `#id` usa l'attributo `id` di un tag HTML per trovare uno specifico elemento. Infatti l'id dovrebbe essere unico all'interno di una pagina HTML; se si è dato a più elementi lo stesso id, viene preso solo il primo, perché questo selettore ritorna sempre un unico elemento.

Per selezionare un elemento con uno specifico id, si scrive il simbolo cancelletto # seguito dall'id dell'elemento:

```
$("#test")
```

Per selezionare gli elementi in base alla classe, si scrive il nome della classe preceduto da un punto. Come per i CSS è possibile combinare i selettori, per ottenere ad esempio tutti i paragrafi che hanno la stessa classe:

```
$("p.class1")
```

Per selezionare elementi che hanno uno stesso attributo, si scrive l'attributo racchiuso da parentesi quadre:

```
$("[href]")
```

È possibile selezionare elementi anche in base alla gerarchia degli elementi nella pagina ("selettori gerarchici"):

- **elementi figli**: sono quegli elementi contenuti in un altro
- **elementi fratelli**: sono quegli elementi che sono allo stesso livello di un altro

Per selezionare un elemento figlio si scrive:

```
$("elemento_genitore elemento_figlio")
```

Ad esempio:

```
$("ul#menu li");
```

seleziona tutti le voci dell'elenco con id menu.

Per selezionare tutti gli elementi direttamente discendenti da un elemento padre, e non anche quelli annidati, si scrive:

```
$("elemento1 > elemento2")
```

Per selezionare un elemento fratello si scrive:

```
$("elemento1 + elemento2")
```

Ad esempio:

```
$("label + input")
```

seleziona tutti gli elementi input preceduti direttamente da un elemento label.

I filtri di selezione

I **filtri di selezione**, che assomigliano agli pseudo-elementi nei CSS, servono a raffinare la selezione andando a prendere elementi specifici o gruppi di elementi che non sono identificati da tag, e vanno sempre applicati ai selettori visti in precedenza. La sintassi è:

```
$("elemento:filtro")
```

Un primo gruppo di filtri può servire a selezionare elementi in base alla posizione rispetto all'elemento contenitore:

- **first** identifica il primo elemento figlio contenuto nell'elemento
- **last** identifica l'ultimo elemento figlio conenuto nell'elemento
- **odd** identifica gli elementi dispari tra quelli contenuti
- **eq(n)** identifica l'elemento n contenuto nell'elemento

I filtri basati sugli attributi selezionano gli elementi che hanno uno stesso attributo:

```
$('elemento[attributo]')
```

o lo stesso valore di un attributo:

```
$('elemento[attributo="valore"]')
```

Un altro gruppo di filtri serve a selezionare elementi in base alla visibilità:

- **hidden** seleziona gli elementi nascosti

oppure in base a un'esclusione:

- **not(elemento)** identifica tutti gli elementi tranne quelli indicati tra parentesi

Un particolare gruppo di filtri è dedicato ai form. In questo caso i comandi sono ancora più sintetici. Ad esempio:

```
$("input[type='text']") // è la versione completa del comando
$(":text") // è la versione abbreviata dello stesso comando
```

seleziona tutti gli elementi input di tipo text. Per trovare tutte le scelte fatte da un utente sui checkbox della pagina si scrive:

```
$(":checked")
```

Le actions

Le actions

Le **actions** sono i metodi che vengono applicati agli elementi HTML selezionati attraverso i selettori visti nel capitolo precedente. Con le actions è possibile manipolare gli elementi del document dal lato client.

DOM Treversing

I primi metodi che vedremo sono quelli per scorrere gli elementi della pagina. Questi metodi possono essere utilizzati in alternativa ai selettori, ma hanno il vantaggio di essere più sintetici.

- **contents()** seleziona tutti gli elementi figli dell'elemento su cui è applicato
- **find(elemento)** seleziona un elemento specifico tra quelli contenuti in un elemento

Ad esempio:

```
$("#id_elemento").find("p").hide();
```

nasconde i paragrafi all'interno dell'elemento con l'id specificato.

Per trovare gli elementi di pari livello (sibling), si usano:

- **next()** seleziona l'elemento di pari livello seguente
- **prev()** seleziona l'elemento di pari livello precedente

Per risalire l'albero del DOM invece si usano:

- **parent()** seleziona gli elementi di livello immediatamente superiore che contengono l'elemento al quale è applicato
- **parents()** seleziona tutti gli elementi contenitori dell'elemento

Entrambi questi due metodi accettano un selettore opzionale come parametro per raffinare la ricerca.

Gestire gli eventi

Gestire gli eventi

jQuery (e Javascript) serve a rendere le pagine dinamiche utilizzando la programmazione lato client, quindi il codice jQuery viene eseguito solo quando l'utente attiva un evento al quale è associata una funzione jQuery, che in questo caso prende il nome di "gestore di eventi".

Abbiamo già visto la funzione `$(document).ready(function)`, che si attiva quando è terminato il caricamento del document. Un'altra funzione usata di frequente è `$(selector).click(function)`, che si attiva quando l'utente clicca su uno specifico elemento. Ad esempio:

```
$(document).ready(function(){
 $("button").click(function(){
    $("p").hide();
 });
});
```

quando l'utente clicca sul bottone vengono nascosti i paragrafi.

Atri gestori di eventi sono:

- **$(selector).mouseover(function)** si attiva quando l'utente passa sopra con il cursore del mouse all'elemento selezionato
- **$(selector).focus(function)** si attiva quando l'elemento è 'focalizzato' (ad esempio quando una pagina del browser è in primo piano)
- **$(selector).dblclick(function)** si attiva quando l'utente fa doppio clic sull'elemento

Effetti

Gli effetti

Una volta imparati a gestire gli eventi si può cominciare ad applicare i primi effetti alla pagina. I comandi jQuery infatti tipicamente prevedono un selettore, un gestore di eventi legato all'elemento selezionato, e un'action che manipola l'elemento. Questo può avvenire sia con un solo comando, concatenando le varie funzioni, che in più comandi o in un blocco di codice.

Gli effetti permettono di modificare la visibilità degli elementi selezionati, creando piccole animazioni che rendono più piacevole l'*user experience* all'interno del nostro sito. Tutto questo senza problemi di compatibilità cross-browser e senza usare strumenti proprietari.

Visibilità

Cominciamo dalle funzioni che servono a rendere visibile o a nascondere:

- **hide()** nasconde l'elemento selezionato, mentre **show()** lo rende di nuovo visibile.

La sintassi per questi due metodi è:

```
$(selector).hide(velocità,callback);
$(selector).show(velocità,callback);
```

Il primo argomento permette di specificare la velocità dell'effetto, e può assumere tre valori possibili: `fast`, `slow`, o in millisecondi. Il secondo argomento invece lo vedremo più avanti. Entrambi sono opzionali.

- **toggle()** alterna la visibilità dell'elemento tra visibile e nascosto

Dissolvenza

Per applicare l'effetto dissolvenza si usano i metodi:

- **fadeIn**(`velocità, callback`) che nasconde l'elemento applicando una dissolvenza
- **fadeOut**(`velocità, callback`) che rende visibile l'elemento
- **fadeToggle**(`velocità, callback`) alterna l'elemento tra fadein() e fadeout()
- **fadeTo**(`velocità, opacità, callback`) applica all'elemento una deterimanata opacità.

La sua sintassi è:

```
$(selector).fadeTo(velocità,opacità,callback)
```

Il primo parametro in questo caso è *required*, e può assumere i seguenti valori: `fast`, `slow`, o in millisecondi. Il secondo parametro è anch'esso obbligatorio e specifica a quale valore di opacità deve fermarsi l'effetto dissolvenza (può assumere valori compresi tra 0 e 1).

Scorrimento

Gli effetti di sliding servono per applicare un morbido effetto 'rullo', gli elementi compaiono e scompaiono scorrendo lungo la loro altezza come in un 'rallenty'.

- **slideDown()** fa scendere l'elemento selezionato
- **slideUp()** fa risalire l'elemento
- **slideToggle()** alterna all'elemento l'effetto comparsa e scomparsa

Animazioni di elementi HTML

- **animate()** permette di creare piccole animazioni personalizzate di elementi HTML, manipolando le sue proprietà CSS.

La sua sintassi è:

```
$(selector).animate({params},velocità,callback);
```

Il primo parametro è obbligatorio.

Possono essere animate anche più di una proprietà dell'elemento allo stesso tempo. Ad esempio:

```
$("button").click(function(){
  $("div").animate({
    left:'150px',
    opacity:'0.5',
    height:'250px',
    width:'300px'
  });
});
```

Si può usare questo metodo per manipolare molte delle proprietà CSS, ma non tutte. Ad esempio per modificare il colore è necessario prima scaricare l'apposito plugin (http://archive.plugins.jquery.com/project/color) dal sito jQuery.com.

E' importante ricordarsi che quando si usa il metodo animate() le proprietà vanno scritte in modo diverso da come si usa nel foglio di stile, in quanto si adotta la grafia camel-cased, quindi margin-left si scrive marginLeft, ecc.

Ci si può riferire alle proprietà CSS anche partendo dal valore corrente ed effettuando delle operazioni algebriche, inserendo += o -= di fronte al valore del parametro:

```
$("div").animate({
    left:'150px',
    height:'+=50px',
    width:'+=50px'
  });
```

Si possono usare anche dei valori predefiniti, come "show", "hide", o "toggle":

```
$("div").animate({
    height:'toggle',
  });
```

Un'altra utile applicazione di animate() è mettere "in coda" i vari effetti, in modo che questi vengano eseguiti in sequenza. Infatti scrivendo più metodi animate() jQuery crea una coda interna eseguendoli uno dopo l'altro. Ad esempio:

```
$("button").click(function(){
    var div=$("div");
    div.animate({left:'150px'},"slow");
    div.animate({fontSize:'2em'},"slow ");
  });
```

sposta prima il div a sinistra e poi aumenta il font del testo.

Esiste anche un metodo `queue()` che serve a mettere in coda gli effetti creati senza usare animate().

Interrompere un effetto

Il metodo `stop()` serve a fermare un effetto o un'animazione prima che sia terminata.

La sua sintassi è:

```
$(selector).stop(stopAll,goToEnd)
```

I parametri sono opzionali. Il primo indica se la coda delle animazioni deve essere cancellata; di default il valore è *false*. Il secondo fa terminare ma allo stesso tempo completare immediatamente l'animazione; di default l'effetto viene terminato senza essere completato.

La funzione di Callback

Il parametro `callback` che abbiamo visto nella firma dei metodi precedenti, serve a eseguire una funzione dopo che l'effetto è terminato. Infatti di norma jQuery esegue i comandi linea per linea, senza attendere che gli effetti siano finiti e senza creare una coda interna. Questo può creare un risultato diverso da quello voluto.

Inserendo il paramatro callback jQuery aspetterà che l'effetto sia terminato prima di eseguire il callback e gli alti metodi.

La sintassi è:

```
$(selector).method(speed,callback)
```

Ad esempio:

```
$("button").click(function(){
  $("p").hide("slow",function(){
    alert("Il paragrafo ora è nascosto");
  });
});
```

L'alert comparirà solo quando il paragrafo sarà completamente nascosto. Senza il callback l'alert

comparirebbe immediatamente.

Modificare il DOM

Una potente caratteristica di jQuery (e quindi anche di Javascript) è quella di fornire numerosi metodi per modificare, dal lato client, gli elementi HTML, i loro attributi e la struttura stessa del document.

Le modifiche inserite nel codice jQuery vengono eseguite quando l'utente attiva un elemento al quale è legato un gestore di eventi, così le pagine vengono rese dinamiche senza interazione con il server.

Manipolare il contenuto dei tag

Per recuperare e impostare il contenuto di tag HTML si usano i metodi:

- **text()** recupera e imposta il contenuto testuale dell'elemento selezionato
- **html()** recupera e imposta il contenuto dell'elemento compresi i tag HTML
- **val()** recupera e imposta i valori inseriti nei campi dei form

Come si può vedere in questo caso, jQuery spesso utilizza la stessa sintassi sia per recuperare il contenuto di un elemento che per settarlo.

Ad esempio:

```
$("#bottone").click(function(){
    alert("Value: " + $("#input1").val());
  });
```

mostra un alert con il testo inserito nel campo con id input1 quando l'utente invia il form.

Modificare gli attributi

Per modificare gli attributi dei tag HTML si usa il metodo:

- **attr()** recupera e imposta uno o più attributi dell'elemento selezionato

La sintassi è:

```
attr(attributo : valore)
```

Ad esempio:

```
$("button").click(function(){
    $("#link1").attr({
       "href" : "http://it.wikibooks.org/wiki/JQuery",
       "title" : "Wikibooks jQuery Tutorial"
     });
  });
```

imposta un nuovo valore per gli attributi `href` e `title` del link con id link1.

Per rimuovere un attributo dall'elemento selezionato si usa `removeAttr()`.

Aggiungere elementi al DOM

Prima abbiamo visto metodi che sostituiscono per intero il contenuto dei tag. Ora vedremo alcuni metodi che permettono di aggiungere contenuto in una determinata posizione:

- **append()** inserisce contenuto alla fine dell'elemento selezionato
- **prepend()** inserisce contenuto all'inizio dell'elemento
- **after()** inserisce contenuto dopo l'elemento
- **before()** inserisce contenuto prima dell'elemento
- **wrap()** avvolge l'elemento selezionato con gli elementi passati come parametro

Questi metodi possono essere usati sia per inserire del semplice testo, ma anche per aggiungere elementi al DOM, attraverso tag HTML, codice jQuery, o Javascript.

Ad esempio:

```
function afterText()  {
   var txt1="<b>I </b>";   // elemento creato con HTML
   var txt2=$("<i></i>").text("love ");   // con jQuery
   var txt3=document.createElement("big");  // con Javascript
   txt3.innerHTML="jQuery!";   // con DOM
   $("img").after(txt1,txt2,txt3);
   // inserisce nuovi elementi dopo img
}
```

la funzione afterText() crea un blocco HTML e poi lo inserisce con after() dopo l'immagine.

Per rimuovere elementi invece si usano i metodi:

- **remove()** rimuove l'elemento selezionato e i suoi elementi figli. remove() accetta come parametro un selettore per filtrare ulteriormente la selezione
- **empty()** rimuove gli elementi figli dell'elemento selezionato
- **unwrap()** rimuove l'elemento genitore dell'elemento selezionato

Manipolare le proprietà CSS degli elementi

jQuery fornisce molti metodi per modificare le proprietà CSS associate agli elementi HTML. Il metodo fondamentale è `css()`, che recupera e imposta gli attributi di stile dell'elemento selezionato.

```
css("proprietà_css", "valore")
```

Questo metodo utilizza gli stessi nomi delle proprietà CSS dei fogli di stile.

Ad esempio:

```
$("p").css({"background-color":"green","font-size":"150%"});
```

Esistono anche metodi per proprietà specifiche.

- **offset()** recupera e imposta la posizione di un elemento rispetto al document, specificando le coordinate dall'alto e da sinistra

- **position()** recupera e imposta le distanze dall'alto e da sinistra rispetto all'elemento selezionato: `$(selettore).position()`

Altri metodi servono per modificare le dimensioni degli elementi.

Dimensioni degli elementi e della window

Le dimensioni di un elemento del DOM sono deteriminate dalla sua `width`/`height`, più il `padding`, i bordi e il margine. I seguenti metodi agiscono su queste specifiche proprietà:

- **width()** recupera e imposta la lunghezza di un elemento
- **height()** agisce sulla proprietà css `height`
- **innerWidth()** e **innerHeight()** agiscono sulla lunghezza/altezza interna includendo il padding
- **outerWidth()** e **outerHeight()** agiscono sulle dimensioni esterne, comprendendo quindi anche padding, bordi e margini

Ad esempio:

```
$("bottone").click(function(){
  $("#div1").width(700).height(400);
});
```

imposta le dimensioni dell'elemento selezionato.

jQuery per AJAX

AJAX sta per Asynchronous JavaScript and XML, ed è una tecnologia per interagire con il server e aggiornare parti della pagina web senza caricare di nuovo tutta la pagina (in modo più simile a quanto avviene nelle applicazioni desktop). Utilizzano AJAX ad esempio Gmail, Google Maps, Youtube, Facebook.

Ajax effettua una chiamata HTTP che poi viene gestita lato client mediante il linguaggio Javascript: il web server invia soltanto i dati richiesti e non l'intera pagina come avviene utilizzando esclusivamente i linguaggi lato server.

jQuery anche in questo caso permette di ridurre e semplificare notevolmente il codice da scrivere, garantendo sempre una alta compatibilità cross-browser.

Con i metodi jQuery per Ajax è possibile caricare dati dal server direttamente nell'elemento HTML selezionato.

- **load()** carica i dati da un server e li inserisce all'interno dell'elemento selezionato. Il metodo richiede un parametro con l'url della risorsa
- **get()** e **post()** caricano i dati dalla risorsa specificata come parametro ma senza inserirli nell'elemento. I dati ottenuti devono essere quindi elaborati attraverso altri comandi. I due metodi corrispondono alle chiamate get e post in http.
- **ajax()** è il metodo più generico per effettuare una chiamata Ajax, e che viene richiamato dagli altri metodi jQuery per Ajax. Accetta come parametri l'url della risorsa, coppie di nomi/valori da passare alla risorsa, e diversi altri parametri opzionali.[1]
- **serialize()** estrae tutti i valori di un form quando questo viene inviato. Questo metodo può essere usato per processare più rapidamente i dati di un form:

```
$(“bottone”).post("ajax/form1.php", ("#form1").serialize());
 });
```

questo codice recupera i valori del form con id form1, e li passa alla risorsa richiamata.

Note

1. W3Schools - jQuery ajax() Method (http://www.w3schools.com/jquery/ajax_ajax.asp)

Crediti

Grazie a tutti quelli che hanno contribuito.

Se sei uno di questi scrivi il tuo nome qui:

- Luca Polpettini (disc.)

Categorie: JQuery | Moduli 100% | Migrazione licenza candidati | Immagini trasferibili su Commons | Immagini GFDL

- Questa pagina è stata modificata per l'ultima volta il 21 ago 2016 alle 21:50.
-