

jQuery

Giuseppe Della Penna
Università degli Studi di L'Aquila

giuseppe.dellapenna@univaq.it
<http://people.disim.univaq.it/dellapenna>

Versione documento: 220124



This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

Cos'è jQuery?



- › jQuery è una libreria Javascript per la programmazione crossbrowser avanzata
- › jQuery si affianca a molte librerie simili, come Prototype o Scriptaculous, ma presenta caratteristiche notevoli:
 - E' molto leggera
 - E' sviluppata da una comunità attivissima
 - E' supportata da una serie di plugin in continuo aumento, che realizzano le funzioni più richieste
 - E' progettata sulla base di canoni di programmazione molto moderni, e per questo risulta estremamente semplice da capire e utilizzare
- › Per iniziare a usare jQuery, è sufficiente scaricarla (<http://jquery.com/>) e importarla come script in una pagina HTML
 - jQuery, come i suoi plugin, viene distribuita in versione “normale”, utile per il debug, e “minimizzata”, molto più compatta e necessaria per rendere leggere le applicazioni finali.
 - L'uso di un debugger Javascript come Firebug è estremamente utile per imparare a usare jQuery

L'Oggetto jQuery



- › L'interazione con jQuery avviene attraverso i metodi e le proprietà esposte dagli **oggetti jQuery**.
- › In particolare, quindi, per **applicare un metodo jQuery a un nodo DOM** è necessario **creare un oggetto jQuery che lo «avvolga»** (*wrapper*), e su questo chiamare il metodo voluto.
- › **In realtà, ogni oggetto jQuery contiene un array di nodi DOM.** Se si applica un metodo a un oggetto jQuery, questo verrà applicato a tutti i nodi in esso contenuti
 - E' possibile usare l'oggetto jQuery come un comune array, verificandone la lunghezza con la proprietà `length` e accedendo ai singoli elementi con la sintassi `[n]`. Tuttavia, va sottolineato che gli elementi così estratti saranno nodi DOM e non nuovi oggetti jQuery.

Costruire un Oggetto jQuery



- › Gli oggetti jQuery vengono creati tramite la speciale funzione **\$()**
- › Questa funzione è molto potente e versatile, infatti può essere chiamata
 - **Senza parametri (\$ o \$())**: l'oggetto jQuery non rappresenterà alcun nodo DOM, ma potrà essere usato per chiamare metodi jQuery che non necessitano di un contesto, come i metodi AJAX.
 - **Su un elemento DOM \$(E)**: l'oggetto conterrà (come un *wrapper*) l'elemento dato
 - **Su una stringa rappresentante un frammento valido di HTML (\$("<p>Ciao</p>"))**: jQuery creerà il DOM corrispondente (senza inserirlo nel documento!) e ne inserirà il nodo radice nell'oggetto creato
 - **Su una stringa rappresentante un selettore CSS3 (\$("a.pippo"))**: l'oggetto jQuery conterrà tutti gli elementi che fanno match col selettore dato

Costruire un Oggetto jQuery: Esempi

```
var body = document.body; //l'elemento body del documento
var jqBody = $(body);
/*un oggetto jQuery che rappresenta l'elemento body del documento*/
//VALE jqBody.length == 1 && jqBody[0] == body

var testo = "Ciao";
var html_fragment = $("<div class='pippo'><p>"+testo+"</p></div>"); /*un oggetto
jQuery che rappresenta l'elemento div, creato (insieme alla struttura interna) ma
non inserito nel documento*/

document.body.appendChild(html_fragment[0]);
/*appende al documento il frammento html appena creato - NON E' IL METODO
CONSIGLIATO PER EFFETTUARE QUESTA OPERAZIONE*/

var elemento_x = $("#x");
/*un oggetto jQuery che rappresenta l'elemento con id="x" nel documento*/
//VALE elemento_x[0] == document.getElementById("x");

var paragrafi = $("p");
/*un oggetto jQuery che rappresenta gli elementi con tag p nel documento*/
//VALE paragrafi.length == numero elementi p nel documento!
```

Selezione di Elementi



- › Un'attività fondamentale nello scripting DOM, semplificata da jQuery, è la selezione di elementi.
- › jQuery permette di cercare e selezionare elementi della pagina web in due modi:
 - **Tramite i selettori CSS3** usati nel costruttore `$()`
 - **Tramite una serie di metodi appositi** esposti dagli oggetti jQuery
- › Le due modalità di selezione possono essere combinate a piacere per facilitare al massimo la selezione
 - La sintassi `$("selettore")` esegue la selezione sull'intero documento
 - E' possibile specificare un particolare elemento come contesto dei selettori, passandolo come secondo parametro: `$("selettore", contesto)`

Selettori jQuery: Base

- › I selettori utilizzabili con jQuery sono tutti quelli di CSS3, più altri creati *ad hoc*. Li riassumiamo qui brevemente.
- › **Selettori di base**
 - *
 - #id
 - elemento
 - .classe
- › **Gerarchia**
 - S1 S2 (*discendenti*)
 - S1, S2 (*unione*)
 - S1 > S2 (*figli*)
 - S1 + S2 (*seguente*)
 - S1 ~ S2 (*fratelli*)

Selettori jQuery: Base

› Attributi

- $S1[A]$ (gli elementi selezionati da $S1$ che hanno un attributo A)
- $S1[A=V]$ (gli elementi selezionati da $S1$ che hanno un attributo A il cui valore è V)
- $S1[A!=V]$ (gli elementi selezionati da $S1$ che hanno un attributo A il cui valore non è V)
- $S1[A^*=V]$ (gli elementi selezionati da $S1$ che hanno un attributo A il cui valore inizia con V)
- $S1[A\$=V]$ (gli elementi selezionati da $S1$ che hanno un attributo A il cui valore termina con V)
- $S1[A|=V]$ (gli elementi selezionati da $S1$ che hanno un attributo A con valore V o iniziante con V -)
- $S1[A*=V]$ (gli elementi selezionati da $S1$ che hanno un attributo A il cui valore ha V come sottostringa)
- $S1[A\sim=V]$ (gli elementi selezionati da $S1$ che hanno un attributo A il cui valore contiene V delimitata da spazi)

Selettori jQuery: Esempi

```
$("a[name]");  
/*oggetto jQuery che rappresenta tutte le ancore nel documento (tag a con attributo  
name)*/  
  
$("div.galleria > img[src$=png]");  
/*oggetto jQuery che rappresenta tutti i tag img (immagini) che sono figli diretti  
dei div di classe galleria, se l'attributo src termina in png (presumibilmente,  
quindi, tutte le immagini di tipo png contenute direttamente in div di classe  
galleria)*/  
  
$("form input[type=checkbox][checked]");  
/*oggetto jQuery che rappresenta tutti i controlli di input inseriti in una form  
che hanno tipo checkbox e sono spuntati (hanno attributo checked). Per il matching  
sulle form esistono però selettori più potenti!*/
```

Selettori jQuery: Filtri

- › I filtri sono pseudo classi CSS (quindi da applicare ad altri selettori, oppure a un * implicito)
- › **Filtri di base**
 - **S1:animated** (*gli elementi selezionati da S1 che sono animati*)
 - **S1:eq(index)** (*l'elemento n-esimo tra quelli selezionati da S1*)
 - **S1:even** (*gli elementi con indice pari tra quelli selezionati da S1*)
 - **S1:odd** (*gli elementi con indice dispari tra quelli selezionati da S1*)
 - **S1:first** (*il primo elemento tra quelli selezionati da S1*)
 - **S1:last** (*l'ultimo elemento tra quelli selezionati da S1*)
 - **S1:gt(n)** (*gli elementi con indice maggiore di n tra quelli selezionati da S1*)
 - **S1:lt(n)** (*gli elementi con indice minore di n tra quelli selezionati da S1*)
 - **S1:not(S2)** (*gli elementi selezionati da S1 che non fanno match con S2*)
 - **S1:header** (*gli elementi selezionati da S1 che sono header*)

Selettori jQuery: Filtri

› Contenuto

- **S1:contains(testo)** *(gli elementi selezionati da S1 che contengono il testo indicato)*
- **S1:empty** *(gli elementi selezionati da S1 che sono vuoti)*
- **S1:has(S2)** *(gli elementi selezionati da S1 che ne contengono almeno uno che soddisfa S2)*
- **S1:parent** *(gli elementi selezionati da S1 che hanno almeno un figlio)*

› Figli

- **S1:first-child** *(gli elementi selezionati da S1 che sono il primo figlio del loro padre)*
- **S1:last-child** *(gli elementi selezionati da S1 che sono l'ultimo figlio del loro padre)*
- **S1:nth-child(n/even/odd)** *(gli elementi selezionati da S1 che sono il figlio n del loro padre, o sono i figli pari/dispari)*
- **S1:only-child** *(gli elementi selezionati da S1 che sono l'unico figlio del loro padre)*

› Visibilità

- **S1:hidden** *(gli elementi selezionati da S1 che sono invisibili)*
- **S1:visible** *(gli elementi selezionati da S1 che sono visibili)*

Selettori jQuery: Esempi

```
$("#table tr:odd td");  
//le celle contenute nelle righe dispari di tutte le tabelle  
  
$("#h1:last + *");  
/*l'elemento che segue immediatamente l'ultima intestazione di livello 1 nel documento*/  
  
$("#h2:gt(1)"); //tutte le intestazioni di livello due tranne la prima  
  
$("#input:not([checked])");  
//i controlli di input senza l'attributo checked impostato  
  
$("#p:has(a)"); //i paragrafi che contengono almeno un link  
$("#p:not(:has(a))"); //i paragrafi che non contengono un link  
$("#p:not(a)"); //i paragrafi che non sono anche link (tutti!)  
  
$("#div:contains(Saluti)"); //le div che contengono il testo "Saluti"  
  
$("#p:empty, p:hidden"); //i paragrafi vuoti o invisibili
```

Selettori jQuery: Form

- › jQuery definisce infine filtri specifici per gli elementi dei moduli
- › **Filtri per controlli form**
 - **S1:text** *(gli elementi selezionati da S1 che sono un input di tipo text)*
 - **S1:checked** *(gli elementi selezionati da S1 che sono un controllo form spuntato)*
 - **S1:disabled** *(gli elementi selezionati da S1 che sono un controllo form disabilitato)*
 - **S1:enabled** *(gli elementi selezionati da S1 che sono un controllo form abilitato)*
 - **S1:selected** *(gli elementi selezionati da S1 che hanno un attributo selected impostato)*
 - **S1:button** *(gli elementi selezionati da S1 che sono un bottone)*
 - **S1:checkbox** *(gli elementi selezionati da S1 che sono una checkbox)*
 - **S1:file** *(gli elementi selezionati da S1 che sono un selettore di file)*
 - **S1:image** *(gli elementi selezionati da S1 che sono un input di tipo immagine)*
 - **S1:input** *(gli elementi selezionati da S1 che sono un input, textarea o select)*
 - **S1:password** *(gli elementi selezionati da S1 che sono un input di tipo password)*
 - **S1:radio** *(gli elementi selezionati da S1 che sono un radio button)*
 - **S1:reset** *(gli elementi selezionati da S1 che sono un bottone di reset)*
 - **S1:submit** *(gli elementi selezionati da S1 che sono un bottone di submit)*

Selettori jQuery: Esempi



```
$(".small:text");  
//gli input di tipo testuale di classe small  
  
$("#selector option:selected");  
//l'opzione selezionata nell'elemento (select) con id="selector"  
  
$("input[type=checkbox]:not(:checked)");  
//gli elementi input di tipo checkbox non spuntati  
  
$("#form1 :submit");  
//i bottoni di submit nell'elemento (form) con id="form1"
```

Funzioni di Attraversamento



- › Molti dei costrutti CSS-like visti finora hanno una loro controparte nelle funzioni di attraversamento del DOM di jQuery
- › Queste funzioni, applicate a un oggetto jQuery, filtrano/manipolano l'insieme di elementi da esso rappresentato
- › Il risultato è sempre un nuovo oggetto jQuery che contiene gli elementi così ottenuti

Funzioni di Attraversamento: Gerarchia

- › **children(S)/find(S)** restituisce i figli/discendenti di ciascun elemento dell'insieme, opzionalmente filtrati dal selettore S
- › **next(S)/prev(S)** restituisce il primo fratello che segue/precede ciascun elemento dell'insieme, opzionalmente filtrato dal selettore S
- › **nextAll(S)/prevAll(S)** restituisce tutti i fratelli che seguono/precedono ciascun elemento dell'insieme, opzionalmente filtrati dal selettore S
- › **nextUntil(S)/prevUntil (S)** restituisce tutti i fratelli che seguono/precedono ciascun elemento dell'insieme, fino al primo che fa match col selettore S
- › **parent(S)/parents(S)** restituisce l'elemento genitore/gli elementi antenati di ciascun elemento dell'insieme, opzionalmente filtrati dal selettore S
- › **siblings(S)** restituisce i nodi fratelli di ciascun elemento dell'insieme, opzionalmente filtrati dal selettore S
- › **contents()** restituisce il contenuto di ciascun elemento dell'insieme, inclusi testo e commenti

Funzioni di Attraversamento: Filtri



- › **first()/last()** restituisce il primo/l'ultimo elemento nell'insieme
- › **eq(n)** restituisce l'n-esimo elemento nell'insieme
- › **filter(S)/has(S)** restituisce gli elementi dell'insieme che fanno match con S/che hanno almeno un discendente che fa match con S
- › **is(S)** restituisce true se almeno un elemento dell'insieme fa match con S. Non è un vero filtro!
- › **not(S)** restituisce gli elementi dell'insieme che non fanno match con S. S può essere anche un array di oggetti DOM o una nodeList DOM.

Funzioni di Attraversamento: Esempi

```
$("h2:eq(14)").nextAll(":header");  
/*tutte le intestazioni che seguono la quattordicesima sezione di livello 2 del  
documento*/  
  
$("img").parents("div"); /*tutte le div che contengono (direttamente o  
indirettamente) in elemento img*/  
  
$("p").filter(":contains(w)")  
/*i paragrafi che contengono la stringa "w" (identico a $("p:contains(w)"))*/  
  
$("#pippo").children("p").next();  
/*gli elementi che seguono immediatamente ciascun paragrafo figlio dell'elemento  
con id="pippo"*/  
  
var e = $("#control");  
If (e.is(":checked")) { /* codice eseguito solo se l'elemento con id="control" è un  
input spuntato */ }
```

Manipolazione del DOM

- › jQuery affianca ai metodi DOM standard, quali *appendChild*, dei nuovi metodi di manipolazione del DOM
 - **append(C)** accoda C ai figli degli elementi nell'insieme. C può essere un elemento DOM, una stringa HTML o un oggetto jQuery (che rappresenta uno o più elementi).
 - **appendTo(S)** accoda gli elementi dell'insieme ai figli di quelli selezionati tramite S, che può essere un elemento DOM, un selettore CSS, una stringa HTML o un oggetto jQuery
 - **prepend(C)/prependTo(S)** funzionano come append e appendTo, ma inseriscono all'inizio della lista dei figli
 - **after(C)/before(C)** inseriscono C prima/dopo gli elementi dell'insieme
 - **insertAfter(S)/insertBefore(S)** inseriscono gli elementi dell'insieme prima/dopo quelli selezionati da S
 - **empty()** elimina il contenuto di tutti gli elementi dell'insieme
 - **remove()** rimuove tutti gli elementi dell'insieme dal DOM insieme alle informazioni interne che JQuery mantiene su di essi, compresi gli eventi
 - **detach()** rimuove tutti gli elementi dell'insieme dal DOM, ma non li distrugge completamente, in modo che possano essere reinseriti altrove.
 - **clone(B)** esegue una copia completa degli elementi dell'insieme. Se B è true, vengono copiati anche i dati e gli event handlers associati all'elemento da jQuery.
 - **replaceAll(S)** sostituisce gli elementi dell'insieme a quelli selezionati da S
 - **replaceWith(C)** sostituisce agli elementi dell'insieme con C
 - **wrap(C)** avvolge una copia di C (che deve essere una struttura con un solo elemento più interno) attorno a *ciascun* elemento dell'insieme.
 - **wrapAll(C)** avvolge una copia di C attorno a *tutti* gli elementi dell'insieme, come singolo gruppo.
 - **wrapInner(C)** avvolge una copia di C attorno al *contenuto di ciascun* elemento dell'insieme.

Manipolazione del DOM: Esempi

```
$("#div.translation").append("<p>Powered by me</p>");  
/*inserisce il paragrafo dato (dopo averlo creato) alla fine delle div con classe  
translation*/  
  
$("#a > li").appendTo("#b");  
/*sposta (nel DOM inserire un elemento già presente in un'altra locazione corrisponde a  
spostarlo) tutti gli item della lista con id="a" alla fine della lista con id="b"*/  
  
var frammento = $("#<p>pippe</p>"); //crea un frammento html  
frammento.appendTo("#a"); //lo appende all'elemento con id="a"  
frammento.clone().prependTo("div.marked:first");  
/*inserisce una copia del frammento all'inizio della prima div di classe marked*/  
  
var exel = $("p:first").detach();  
//rimuove ma non cancella il primo paragrafo del documento...  
//...  
exel.appendTo(document.body);  
//e lo reinserisce alla fine del documento  
  
$("div:first").replaceWith("<p>pippe</p>");  
/*sostituisce la prima div del documento col paragrafo dato */  
  
$("div.square").wrap("<div><div></div></div>");  
/* inserisce ogni div di classe square all'interno di due ulteriori div */  
$("div.square").wrapAll("<div class='all'></div>")  
/* inserisce tutte div di classe square (insieme) all'interno di una div di classe all */  
$("div.square").wrapInner("<div class='pluto'></div>")  
/* inserisce il contenuto di ogni div di classe square all'interno di una div di classe  
pluto */
```

Modifica di Contenuto e Attributi

- › jQuery mette a disposizione molti metodi standard per manipolare il contenuto degli elementi in maniera sicura e crossbrowser.
- › Va notato che, se applicati su oggetti jQuery che rappresentano più di un elemento, **i metodi di lettura restituiscono il valore estratto dal primo elemento dell'insieme**, mentre quelli di impostazione **agiscono su ciascun elemento dell'insieme stesso**
 - **attr(A)** restituisce il valore dell'attributo A
 - **attr(A,V)** imposta l'attributo A al valore V. Può accettare anche un oggetto, considerando ciascuna proprietà come un attributo da impostare. E' anche possibile usare una funzione come valore (verrà chiamata con l'indice dell'elemento e il precedente valore dell'attributo)
 - **removeAttr(A)** rimuove l'attributo A
 - **html()** restituisce il codice html contenuto nell'elemento
 - **html(T)** imposta il codice html T come contenuto dell'elemento
 - **text()** restituisce il testo contenuto nell'elemento (eliminando l'eventuale markup)
 - **text(T)** imposta il testo T come contenuto dell'elemento
 - **val()** restituisce il valore di un controllo di form
 - **val(V)** imposta a V il valore di un controllo di form

Modifica del Contenuto: Esempi

```
$("#risultato").html("<p>Nessun riscontro</p>");  
/*inserisce il paragrafo dato all'interno dell'elemento con id=risultato*/  
  
$("#risultato").text();  
/*restituisce il solo testo contenuto nell'elemento. In questo caso "Nessun  
riscontro"*/  
  
$(":input").val();  
/*restituisce il value del primo controllo input, textarea o select nel documento*/  
  
$("#pippo").val("pluto");  
/*se l'elemento con id=pippo è un input testuale o una textarea, ne sostituisce il  
contenuto con la stringa "pluto". Se l'elemento è una select, ne seleziona  
l'opzione avente come valore "pluto" (se esiste). Negli altri casi, non ha alcun  
effetto.*/  
  
$(document.body).attr("lang","it");  
/*imposta a "it" l'attributo "lang" sull'elemento body del documento*/  
  
$("[lang]").removeAttr("lang");  
/*rimuove l'attributo lang da tutti gli elementi che lo specificano*/
```

Manipolazione dello Stile



- › jQuery dispone di tre metodi per manipolare l'attributo di classe (*class*) degli elementi:
 - **addClass(C)** aggiunge la classe C
 - **removeClass(C)** rimuove la classe C
 - **toggleClass(C)** aggiunge/rimuove la classe C
 - **hasClass(C)** vero se almeno un elemento dell'insieme ha classe C
- › E' inoltre possibile manipolare direttamente gli stili di ogni elemento utilizzando le funzioni
 - **css(P)** restituisce il valore corrente (calcolato) della proprietà CSS P, specificata secondo lo standard W3C
 - **css(P,V)** imposta la proprietà CSS P al valore V. Può accettare anche un oggetto, considerando ciascuna proprietà come un proprietà CSS da impostare.

Manipolazione Stile: Dimensioni e Posizione

- › Con i propri metodi jQuery annulla i problemi derivanti dal diverso modo di leggere o impostare le dimensioni e la posizione degli elementi.
- › Anche in questo caso, se applicati a degli insiemi, i metodi di lettura restituiscono i dati estratti dal solo primo elemento.
 - **height()/height(V)** restituisce o imposta (usando una stringa CSS o un numero di pixel) l'altezza (che comprende solo lo spazio interno all'elemento, escludendo padding, bordo e margini)
 - **width()/width(V)** restituisce o imposta l'ampiezza (che comprende solo lo spazio interno all'elemento, escludendo padding, bordo e margini)
 - **outerWidth(b)/outerHeight(b), outerWidth(V)/outerHeight(V)** restituisce l'ampiezza/altezza totale, comprensiva di padding interno, spessore del bordo e anche margini esterni se b è true, oppure imposta i corrispondenti valori
 - **innerWidth()/innerHeight(), innerWidth(V)/innerHeight(V)** restituisce/imposta l'ampiezza/altezza interna, comprensiva di padding.
 - **position** restituisce la posizione (oggetto con attributi *left* e *top*) *relativa all'offsetParent()*
 - **offset()/offset({top:Y, left:X})** restituisce o imposta la posizione (oggetto con attributi *left* e *top*) *relativa al documento*

Manipolazione dello Stile: Esempi

```
$(".generic-input:text").addClass("text-input").removeClass("generic-input");  
/*rimuove la classe generic-input ed aggiunge la classe text-input a tutti gli  
elementi input di classe generic-input*/  
  
$("small").css("font-size","8pt");  
/*forza una dimensione carattere di otto punti su tutti gli elementi small  
(ovviamente sarebbe meglio fare la stessa cosa con una regola di stile!)*/  
  
$("#pippo").css("margin-top");  
/*fornisce le informazioni (calcolate) sul margine superiore dell'elemento con  
id=pippo. Attenzione: jQuery non supporta molte delle proprietà aggregate, come  
"margin"*/  
  
$("p.collapsible").height(10);  
/*imposta a 10 pixel l'altezza di tutti i paragrafi di classe collapsible*/  
  
var pos = $("#moveable").offset();  
$("#moveable").offset({left: pos.left+10, top: pos.top});  
/*sposta l'elemento con id= moveable di 10 pixel in basso*/
```

Associazione di Dati



- › jQuery permette di associare dati arbitrari, sotto forma di coppie (*chiave, valore*) a tutti gli elementi DOM. Questi dati non appaiono nel codice HTML né nella sua rappresentazione visuale.
 - **data(K)/data(K,V)** restituisce o imposta a V il valore associato alla chiave K per tutti gli elementi dell'insieme
 - **removeData(K)** rimuove il valore K da tutti gli elementi dell'insieme

Associazione di Dati: Esempi



```
var x = $("#pippo");  
/*individua l'elemento con id=pippo*/  
  
x.data("colore",x.css("color"));  
/*memorizza il colore css dell'elemento all'interno dei suoi dati*/  
  
x.css("color","red");  
/*cambia il colore dell'elemento*/  
  
//...  
  
x.css("color",x.data("colore"));  
/*ripristina il colore dell'elemento a quello salvato nell'elemento stesso*/
```

Gestione degli Eventi: Binding



- › jQuery dispone di una gestione degli eventi che permette di superare molte delle incompatibilità tra i browser.
- › Le funzionalità di event handling sono accessibili tramite i metodi
 - **on(T,F)** aggancia, negli elementi dell'insieme, all'evento specificato da T l'*event handler* F.
 - › T è una stringa contenente il nome di un evento Javascript (ad es. "click").
 - › F è una funzione che verrà chiamata assegnando al suo *contesto* (*this*) l'elemento che ha scatenato l'evento e passando l'oggetto evento come unico argomento. L'oggetto evento è normalizzato, come vedremo più avanti
 - › E' possibile passare come argomento un oggetto le cui proprietà saranno interpretate come nomi di eventi e i relativi valori come gli handlers da associarvi.
 - **off(T,F)** rimuove F dalla lista degli handlers per l'evento T negli elementi dell'insieme. Omettendo F si rimuovono tutti gli handlers per l'evento T.
 - **one(T,F)** come *on*, ma l'handler viene rimosso dopo la prima attivazione
 - **trigger(T)** simula l'evento T su tutti gli elementi dell'insieme, attivandone gli handlers
- › Anche in jQuery esistono dei metodi-scorciatoia per eseguire il binding diretto di eventi, ad esempio **click(F)**, **dblclick(F)**, **hover(F)**, **focusin(F)**, **focusout(F)**, **keypress(F)**, **keyup(F)**, **mouseenter(F)**, **mousedown(F)**, **mousemove(F)**, **mouseout(F)**, **mouseover(F)**.
 - Chiamati senza argomenti, questi helper simulano l'evento corrispondente (attivandone gli handlers).

Gestione degli Eventi: Eventi Normalizzati

- › L'oggetto *Event* passato da jQuery agli handler agganciati con il metodo *on* è un **evento normalizzato**, che assicura la presenza e la semantica dei seguenti campi e metodi principali:
 - **type** descrive il tipo di evento, ad esempio “click”
 - **pageX/pageY** è la posizione dell'evento relativa alla pagina
 - **which** è il bottone (per gli eventi mouse) o tasto (per gli eventi tastiera) che ha scatenato l'evento
 - **target** è l'elemento DOM che ha attivato l'evento
 - **relatedTarget** è l'elemento definito per questo evento nella *specificazione degli eventi DOM*
 - **currentTarget** è l'elemento del DOM a cui è attualmente passato l'evento (in fase di bubbling): ha lo stesso valore di *this* nella funzione
 - **result** è il valore ritornato dall'ultimo handler per questo evento
 - **preventDefault()** evita che il browser esegua l'azione di default per questo evento
 - **stopPropagation()** blocca il bubbling dell'evento

Gestione degli Eventi: Delegazione

- › Tramite la **event delegation** è possibile scrivere un *handler* che gestirà un particolare evento per tutti i discendenti dell'elemento su cui è impostato.
- › Si utilizza il metodo *on* con una sintassi leggermente diversa:
 - **on(T,S,F)** aggancia, per tutti i discendenti degli elementi dell'insieme che fanno match con S (un selettore), all'evento specificato da T l'*event handler* F.
- › L'handler sarà valido **anche per discendenti inseriti successivamente negli stessi elementi**, per questo l'espressione appena vista è diversa dall'assegnare lo stesso handler a tutti i discendenti usando un'espressione del tipo `$("E S").on(T,F)`

Gestione degli Eventi: Esempi



```
$("#p").on("click",function(e) { $(this).text(e.pageX+", "+e.pageY); });  
/*con questo handler, ogni paragrafo cliccato sarà riempito con le coordinate del  
mouse al momento del click*/  
  
$(" :text").on("keyup",function(e) {  
    if (e.which == 32) alert("hai premuto lo spazio"); });  
/*con questo handler, assegnato a tutti gli input testuali, verrà visualizzata una  
notifica ogni volta che si inserisce uno spazio nel controllo*/  
  
$("div.test").mouseenter(function() {  
    $(this).css("background-color","red");  
}).mouseleave(function() {  
    $(this).css("background-color","inherit"); });  
/*con questi handler tutte le div di classe test cambieranno colore al passaggio  
del mouse*/  
  
$("body").on("click","a", function(e) {  
    e.preventDefault(); alert(this.href);});  
/*con questo handler delegato, tutte le volte che si clicca un link nella pagina  
verrà visualizzata la notifica col corrispondente indirizzo web, ma la pagina non  
cambierà*/
```

Binding degli Eventi Load e Ready

- › L'evento *load* del documento è essenziale, come sappiamo, per eseguire codice solo quando il documento è pronto per la manipolazione.
- › Tuttavia, *load* viene generato quando il documento è renderizzato (compreso download di tutte le dipendenze, come le immagini), mentre è quasi sempre possibile avviare gli script quando il DOM è stato creato in memoria.
- › Per questo jQuery, oltre a permettere il binding di handler all'evento *load*, mette a disposizione un evento *ready*, e una sintassi estremamente compatta per associare codice a questo evento:
 - **$\$(F)$** chiama la funzione *F* quando il DOM è in stato *ready*
- › Si possono così dichiarare, anche a più riprese (cioè con più istanze della chiamata $\$(F)$), tutti i blocchi di codice da eseguire quando il DOM è pronto per la manipolazione.

Eventi Load e Ready: Esempi



```
$(function() {  
    $(document.body).css("background-color","red");  
});  
/*cambia in rosso il colore di fondo del body appena il DOM è stato caricato in  
memoria*/  
  
$(document).ready(function() {  
    $(document.body).css("background-color","red");  
});  
/*metodo alternativo*/
```

Animazione



- › jQuery dispone di potenti funzioni per animare gli elementi. Qui vediamo quelle di uso più comune.
 - **hide(T,C)/show(T,C)** nasconde/mostra ciascun elemento dell'insieme. Se T è fornito, gli elementi vengono animati fino a scomparire/apparire in T millisecondi (T può essere anche una costante come "slow" e "fast"). Se C è fornito, è una funzione che viene chiamata per ciascun elemento, appena la sua animazione è terminata, e al suo interno *this* punta all'elemento appena animato.
 - **fadeOut(T,C)/fadeIn(T,C)** agiscono come *hide* e *show*, ma con un effetto sfumato.
 - **slideUp(T,C)/slideDown(T,C)** agiscono come *hide* e *show*, ma con un effetto di scorrimento.
 - **toggle(T,C), slideToggle(T,C), fadeToggle(T,C)** applicano l'effetto corrispondente invertendo lo stato di visibilità attuale dell'elemento
 - **animate(P,T,C)** anima le proprietà indicate nell'oggetto P fino ai corrispondenti valori. Non si possono animare i colori.
 - **stop()** interrompe l'animazione corrente su tutti gli elementi dell'insieme.

Animazione: Esempi



```
$("#display").fadeOut(1000,function() {
    $(this).html("<b>Sorpresa!</b>").fadeIn("fast"); });
/*aggiorna in maniera "dolce" il contenuto dell'elemento con id=display: prima fa
scompare in maniera sfumata l'elemento, quindi appena quest'ultimo è invisibile,
ne modifica il contenuto html e lo rende di nuovo visibile, sempre in modo
sfumato*/

$("body").on("click","div.toggle-button",function(){
    $(this).parent().children(".toggle-area").toggle(); });
/*questo event handler delegato fa in modo che tutte le volte che si clicca su una
div di classe toggle-button, le div di classe toggle-area all'interno dello stesso
contenitore (parent) appaiano o scompaiano. */

$("body").animate({fontSize: "+=20", fontWeight: 700},"slow")
/*anima lentamente le proprietà del body fino a portare la font-size a un valore
maggiore di quello attuale (di 20) e la font-weight a 700*/
```

jQuery e AJAX

- › jQuery comprende molte funzioni di accesso ad AJAX di livello diverso. Mostreremo prima la funzione di basso livello **\$.ajax** e poi i più convenienti metodi helper di alto livello.
- › E' possibile specificare a jQuery il tipo di dati che ci si aspetta nella risposta AJAX, così da poterla validare e, se necessario, interpretare:
 - **text** (testo generico), **html** (html da inserire nella pagina), **script** (uno script da aggiungere alla pagina), **json** (dati in formati json da interpretare come strutture javascript), **jsonp** (json da un diverso dominio), **xml** (dati generici in XML)
 - Se non si specifica il tipo, jQuery lo dedurrà dal tipo MIME specificato dal server
 - Per aggirare la **same-origin-policy**, la modalità jsonp scarica i dati sotto forma di script (aggiungendo un tag script al documento) e poi li interpreta come json.
- › Tutti i parametri dei metodi AJAX possono essere omessi tranne la URL a cui effettuare la richiesta. I default sono metodo **GET** e chiamata **asincrona**.

jQuery e AJAX: \$.ajax

- › **\$.ajax(O)** esegue una chiamata AJAX configurata in base ai parametri dell'oggetto O. Questi sono i più comuni:
 - **async**: permette di rendere la chiamata asincrona (default) o sincrona (bloccante)
 - **success**: funzione *callback* chiamata se la richiesta ha successo. La funzione riceve come parametri
 - › I dati ricevuti dal server, opportunamente decodificati (stringa, elemento XML, file Javascript, oggetto JSON decodificato, ecc.)
 - › Il messaggio testuale di stato restituito dal server
 - › L'oggetto XMLHttpRequest usato per la chiamata
 - **error**: funzione callback chiamata se la richiesta *non* ha successo. La funzione riceve come parametri l'oggetto XMLHttpRequest usato per la chiamata e il messaggio testuale di stato restituito dal server.
 - **complete**: funzione callback chiamata quando la richiesta è terminata (con o senza successo). La funzione riceve gli stessi parametri della *error*.
 - **data**: i dati da inviare al server, sotto forma di oggetto o query string. I dati verranno codificati come parametri della richiesta.
 - **dataType**: il tipo di dati della risposta. Si veda la slide precedente.
 - **timeout**: timeout della richiesta in millisecondi.
 - **method**: tipo della richiesta, "POST", "GET", ...
 - **url**: indirizzo a cui dirigere la richiesta

jQuery e AJAX: helper di alto livello

- › **`$.get(U,D,C,T)`** esegue una chiamata asincrona alla url U, passando opzionalmente come dati GET le coppie (*chiave, valore*) contenute nell'oggetto D. Se la chiamata ha successo, i dati ricevuti vengono decodificati in base al tipo MIME specificato dal server o al tipo T dichiarato nella chiamata e passati alla funzione *callback* C.
 - La funzione callback di successo riceve come parametri
 - › I dati ricevuti dal server, opportunamente decodificati (stringa, elemento XML, file Javascript, oggetto JSON decodificato, ecc.)
 - › Il messaggio testuale di stato restituito dal server
 - › L'oggetto XMLHttpRequest usato per la chiamata
 - In caso di errore, non viene eseguita alcuna azione!

jQuery e AJAX: helper di alto livello

- › **\$.post(U,D,C,T)** agisce come \$.get ma esegue una richiesta POST. *Post* andrebbe usato al posto di *get* per invocare operazioni che modificano i dati sul server.
- › **\$.getJSON(U,D,C)** agisce come \$.get ma sottintende il tipo JSON come risposta. In questo modo la funzione callback riceverà un oggetto Javascript decodificato.
- › **\$.getScript(U,C)** agisce come \$.get ma sottintende il tipo Script come risposta. JQuery valuterà il testo ricevuto come uno script Javascript e lo eseguirà nel contesto globale. Utile per caricare script dinamicamente.
- › **.load(U,D,C)** se applicato a un oggetto JQuery, sostituisce il contenuto degli elementi da esso rappresentati con il testo html della risposta. Il *callback* C, se specificato, viene eseguito dopo la sostituzione.

jQuery e AJAX: helper per le form

- › Per trasformare rapidamente i campi di una form in oggetti passabili come *data* (D) in una chiamata AJAX, jQuery mette a disposizione due metodi:
 - **serialize()** applicato a una form, produce la *query string* corrispondente ai suoi campi (la stessa che genererebbe il browser per una normale sottomissione della form!)
 - **serializeArray()** applicato a una form, produce un array di oggetti contenenti coppie (*nome_campo*, *valore_campo*) corrispondenti ai suoi campi.

AJAX: Esempi



```
$("#latest").load("http://server/latest-news.php");  
/*il modo più semplice per ricaricare dinamicamente il contenuto di una parte di  
documento (in questo caso l'elemento con id=latest)*/  
  
$.get("http://server/latest-news.php",{ },updateNews,"html");  
function updateNews(data) {  
    $("#latest").fadeOut(1000,function() {$(this).html(data).fadeIn(1000); });  
}  
/*come nell'esempio precedente, ma in caso di successo aggiorna il contenuto  
dell'elemento con un effetto sfumato. Se la chiamata fallisce, nessuna funzione  
viene chiamata. Se è necessario gestire anche i casi di fallimento, usare la  
funzione a basso livello $.ajax()*/
```

AJAX: Esempi



```
/*la funzione che segue può essere utilizzata per caricare dinamicamente degli
script solo quando sono necessari, alleggerendo il tempo iniziale di caricamento
dei documenti*/
var loaded = {} //globale
function requireScript(url) {
    if (! (url in loaded) || loaded[url]==false) {
        $.getScript(url, function() {loaded[url] = true; alert("ora è possibile usare le
funzioni in "+url)});
    }
}

$(".sbutton").on("click", function() {
    var formdata = $(this).parents("form").first().serialize();
    $.ajax({url: "http://server/update.php",
        success: resultHandler,
        error: errorHandler,
        dataType: "json",
        data: formdata,
        type: "POST"
    })
});
/*quando si clicca su un elemento di classe sbutton, invia i dati della form che lo
contiene con una chiamata POST AJAX, gestendo anche i possibili errori con un
apposito handler e aspettandosi dati JSON come risposta*/
```

Iterazione su un oggetto jQuery

- › E' possibile iterare l'applicazione di una funzione su tutti gli elementi dell'insieme contenuto in un oggetto jQuery usando il metodo **each**
 - **each(F)**, dove F è una funzione, richiama F una volta per ogni elemento dell'insieme, impostandone il *contesto* (*this*) all'elemento corrente e passando opzionalmente come argomento *l'indice dell'elemento* nell'insieme e l'elemento stesso. **Restituendo false**, l'iterazione viene arrestata.
 - Questa funzione può essere chiamata anche direttamente sull'oggetto jQuery passando un array come primo parametro: **\$.each(A,F)**. In questo caso l'iterazione avverrà all'interno dell'array A.

Iterazione: Esempi



```
/*il codice che segue imposta su ciascuna div del documento un handler per il click
che visualizza un messaggio contenente il numero d'ordine della div nel documento*/
$("div").each(function(i){
    $(this).on("click",function(e) {
        alert("Questa è la DIV numero "+i);
        e.stopPropagation(); //evitiamo che le div più esterne intercettino lo stesso
        evento
    });
});

/*il frammento che segue popola l'array colors con i colori di tutti i paragrafi
nel documento*/
var colors = [];
$("p").each(function(){
    colors.push($(this).css("color"));
});
```

Riferimenti



- › **Sito di jQuery**
<http://jquery.com>