PROGRAMMAZIONE WEB

SCRIPTING CLIENT-SIDE IN JAVASCRIPT
LA LIBRERIA JQUERY

Prof. Ada Bagozi
ada.bagozi@unibs.it

Cos'è JQuery?

JQuery è una libreria Javascript per la programmazione *crossbrowser avanzata* «costruita» attorno al DOM

JQuery si affianca a molte librerie simili, ma presenta caratteristiche notevoli:

- ✓ è molto leggera
- ✓ è sviluppata da una comunità attivissima
- ✓ è supportata da una serie di plug-in in continuo aumento, che realizzano le funzioni più richieste
- ✓ è progettata sulla base di canoni di programmazione molto moderni e per questo risulta estremamente semplice da capire e utilizzare

Iniziare ad usare JQuery

Per iniziare a usare Jquery (**http://jquery.com/**), è sufficiente importarla come script in una pagina HTML (non necessariamente dopo averla scaricata)

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

```
<head>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
</head>
```

JQuery, come i suoi plug-in, viene distribuita in versione "normale", utile per il debug, e "minimizzata", molto più compatta e necessaria per rendere leggere le applicazioni finali

L'oggetto JQuery

L'interazione con JQuery avviene attraverso i metodi e le proprietà esposte dagli *oggetti JQuery*

Per esempio, per applicare un metodo JQuery a un nodo DOM o a un insieme di nodi, è necessario creare un oggetto JQuery che li rappresenti, e su questo chiamare il metodo voluto

Ogni oggetto JQuery contiene un array di elementi, possibilmente vuoto

- ✓ è 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]`**

Costruire un oggetto JQuery

Gli oggetti JQuery vengono creati tramite la speciale funzione `$()`

```
$(selector).action
```

Questa funzione è molto potente e versatile, infatti può essere chiamata

Senza parametri (`$` o `$()`): in questo caso, si costruisce un oggetto JQuery che non rappresenta alcun nodo DOM, ma su cui è possibile chiamare i metodi identificati da `action`

Un elemento DOM (`$(E)`): in questo caso l'oggetto conterrà (come un wrapper) l'elemento dato

Una stringa rappresentante un frammento valido di (X)HTML (`$("<p>Ciao</p>")`): in questo caso, JQuery creerà il DOM corrispondente (senza inserirlo nel documento!)

Un stringa che rappresenta un selettore CSS (`$("a.pippo")` oppure `$("a#pluto")`): in questo caso, JQuery utilizzerà gli elementi identificati dal selettore nella pagina (X)HTML per costruire l'oggetto

Primi esempi

`var body = document.body;` // l'elemento body del documento

`var jqBody = $(body);` // un oggetto JQuery che rappresenta l'elemento body del documento (`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 al sotto-albero che parte da esso), ma non inserito nel DOM

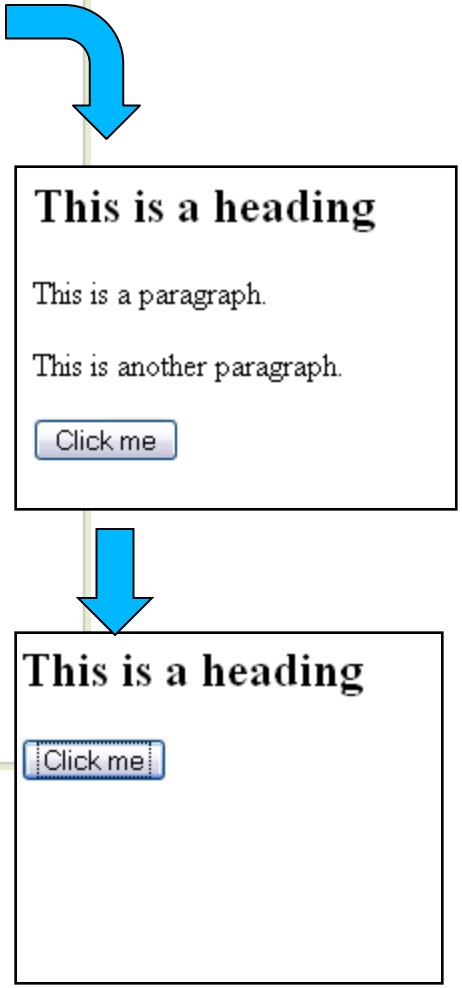
`var elemento_x = $("#x");` // un oggetto JQuery che rappresenta l'elemento con `id="x"` nel documento XHTM a cui lo script è associato (`elemento_x[0] == document.getElementById("x");`)

`var paragrafi = $("p");` // un oggetto JQuery che rappresenta gli elementi con tag `p` nel documento (la variabile `paragrafi.length` è pari al numero elementi `p` nel documento)

Il primo esempio JQuery


```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $("input[type=button]").click(function(){
    $("p").hide();
  });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<form action="">
<input type="button" value="Click me"/>
</form>
</body>
</html>
```



La funzione `ready`

La funzione **`ready`** fa in modo che le action JQuery non siano eseguite prima che il documento sia completamente caricato

```
$(document).ready(function(){
  $("input[type=button]").click(function(){
    $("p").hide();
  });
});
```

Selettori di base JQuery (I)

I selettori utilizzabili con JQuery sono tutti quelli già visti per i Cascading StyleSheets (CSS), più altri creati ad hoc

Selettori di base

- ✓ `*`
- ✓ `#id`
- ✓ `element`
- ✓ `.class`

Gerarchia

- ✓ `S1 S2` (discendenti)
- ✓ `S1,S2` (unione)
- ✓ `S1>S2` (figli)

Selettori di base JQuery (II)

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`**)

Selettori di base JQuery (III)

Attributi

- ✓ **`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`** il cui valore ha **`V`** come sottostringa)
- ✓ **`S1[A~=V]`** (gli elementi selezionati da **`S1`** che hanno un attributo **`A`** il cui valore contiene **`V`** delimitata da spazi)

Esempi

`$("a[name]");` //oggetto JQuery che rappresenta tutti i tag **`a`** nel documento con attributo **`name`**

`$("div.galleria > img[src$=png]");` //oggetto JQuery che rappresenta tutti i tag **`img`** che sono figli diretti dei **`div`** di classe **`galleria`**, se l'attributo **`src`** termina in **`png`** (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)

I filtri sono pseudo classi CSS (quindi da applicare ad altri selettori, oppure ad un * implicito)

- ✓ **`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`**)

Selettori JQuery: filtri (II)

I filtri sono pseudo classi CSS (quindi da applicare ad altri selettori)

- ✓ **`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`**) analogo a **`S1:lt(n)`**
- ✓ **`S1:not(S2)`** (gli elementi selezionati da **`S1`** che non fanno match con **`S2`**)

Selettori JQuery: filtri (III)

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)

Selettori JQuery: filtri (IV)

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)

Altri esempi

`$("table tr:odd td");` // le celle contenute nelle righe dispari di tutte le tabelle

`$("h2:gt(0)");` // 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

`$("div:contains(Saluti)");` // le `div` che contengono il testo `"Saluti"`

Selettori JQuery: Form (I)

JQuery definisce anche filtri specifici per gli elementi dei moduli

- ✓ **`S1:text`** (gli elementi selezionati da **`S1`** che sono un input di tipo text)
- ✓ **`S1:checked`** (gli elementi selezionati da **`S1`** che sono un campo spuntato)
- ✓ **`S1:disabled`** (gli elementi selezionati da **`S1`** che sono un campo disabilitato)
- ✓ **`S1:enabled`** (gli elementi selezionati da **`S1`** che sono un campo 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)

Selettori JQuery: Form (II)

- ✓ `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, select o button)
- ✓ `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)

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"`

Modifica di contenuto e attributi (I)

- ✓ 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

Modifica di contenuto e attributi (II)

- ✓ **`attr(A)`** restituisce il valore dell'attributo **`A`**
- ✓ **`attr(A,V)`** imposta l'attributo **`A`** al valore **`V`**
- ✓ **`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

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 esempio "Nessun riscontro"

`$(":input").val();` // restituisce il value del primo controllo input, textarea, select o button 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");` // importa 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 del DOM (I)

JQuery fornisce metodi di manipolazione del DOM

- ✓ **`append(C)`** accoda **`C`** ai figli degli elementi nell'insieme a cui il metodo è applicato; **`C`** può essere un elemento DOM, una stringa HTML e un oggetto JQuery (che rappresenta uno o più elementi)
- ✓ **`appendTo(S)`** accoda gli elementi dell'insieme a cui il metodo è applicato all'insieme dei figli degli elementi 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

Manipolazione del DOM (II)

- ✓ `after(C)`/`before(C)` inseriscono `C` prima/dopo gli elementi dell'insieme a cui il metodo è applicato
- ✓ `insertAfter(S)`/`insertBefore(S)` inseriscono gli elementi dell'insieme a cui il metodo è applicato prima/dopo quelli selezionati da `S`
- ✓ `empty()` elimina il contenuto di tutti gli elementi a cui è applicato
- ✓ `remove()` rimuove tutti gli elementi a cui è applicato
- ✓ `detach()` rimuove tutti gli elementi a cui è applicato, ma non li distrugge, in modo che possano essere reinseriti altrove
- ✓ `clone(B)` esegue una copia completa degli elementi a cui è applicato; se `B` è true, vengono copiati anche gli *event handlers* associati all'elemento da JQuery

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>pippo</p>");` // crea un frammento html

`frammento.appendTo("#a");` // appende il frammento 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

Le espressioni regolari (I)

Le espressioni regolari sono molto utili per eseguire il **parsing e la validazione dei dati** immessi dall'utente

Javascript (e quindi anche jQuery) riconoscono le espressioni regolari scritte nella **sintassi Perl**

Un'espressione regolare **costante** è definita dalla seguente sintassi

```
/pattern/modifiers
```

Un'espressione regolare può anche essere generata a tempo di esecuzione tramite il costruttore **`RegExp`**

```
regularExpr = new RegExp(patter,modifiers)
```

Le espressioni regolari (II)

Modifiers	Descrizione
i	Effettua un matching case-insensitive
g	Trova tutti i match anziché fermarsi alla prima occorrenza

Espressione	Descrizione
[abc]	Caratteri uguali a uno di quelli tra parentesi quadre
[^abc]	Caratteri diversi da quelli tra parentesi quadre
[0-9]	Qualsiasi cifra tra 0 e 9
[A-Z]	Qualsiasi lettera tra A e Z (maiuscole)
[a-z]	Qualsiasi lettera tra a e z (minuscole)
[A-z]	Qualsiasi lettera tra A (maiuscola) e z (minuscola)
(red|green|blue)	Una delle espressioni alternative elencate

Le espressioni regolari (III)

Metacarattere	Descrizione
.	Carattere singolo, eccetto newline o carattere di fine riga
\w	Una lettera qualsiasi
\W	Un carattere che non sia una lettera
\d	Una cifra qualsiasi
\D	Un carattere che non sia una cifra
\s	Un carattere non visibile, come uno spazio, una tabulazione o un fine riga
\S	Un carattere che non sia tra quelli non visibili
\n \r \t	New line, carriage return, tab
\xxx	Carattere specificato dal numero in notazione ottale xxx
\xdd	Carattere specificato dal codice esadecimale dd

Le espressioni regolari (IV)

Quantificatori	Descrizione
n+	Una stringa di uno o più “n”
n*	Una stringa con zero o più “n”
n?	Una stringa con uno o nessun “n”
n{X}	Una stringa che contiene X caratteri “n”
n{X,Y}	Una stringa che contiene da X a Y caratteri “n”
n{X,}	Una stringa che contiene almeno X caratteri “n”
n$	Una stringa che termina con il carattere “n”
^n	Una stringa che inizia con il carattere “n”
?=n	Una stringa seguita dal carattere “n”
?!n	Una stringa non seguita dal carattere “n”

Le espressioni regolari (V)

È possibile utilizzare le espressioni regolari con vari metodi da invocare sull'oggetto stesso che rappresenta l'espressione `r`

- ✓ `r.test(s)` restituisce `true` se la stringa `s` è conforme all'espressione regolare `r`
- ✓ `s.match(r)` restituisce un'array con tutte le corrispondenze dell'espressione regolare `r` trovate nella stringa `s`
- ✓ `s.replace(r,sNew)` individua nella stringa `s` tutte le corrispondenze con l'espressione `r` e le sostituisce con `sNew`
- ✓ `s.split(r)` divide la stringa `s` in una serie di segmenti distinti dai separatori specificati con l'espressione `r` e li restituisce come array
- ✓ `s.search(r)` cerca nella stringa `s` la prima corrispondenza con l'espressione regolare `r` e restituisce l'indice della posizione trovata

PROGRAMMAZIONE AD EVENTI

La programmazione ad eventi

- ✓ Javascript è un linguaggio fortemente *orientato agli eventi*
  - ✓ l'utente interagisce con la pagina muovendo il mouse, cliccando, digitando qualcosa in una casella di testo, etc.
  - ✓ la pagina deve “reagire” opportunamente a queste sollecitazioni
  - ✓ il gestore dell'evento (*event handler*) è una funzione che risponde alla “mossa” dell'utente
  - ✓ è possibile catturare gli eventi e gestirli
- ✓ Gli eventi possono essere generati dall'utente attraverso le proprie azioni sul documento o possono essere generati dal browser (e.g., caricamento delle pagine, caricamento del documento)

Eventi in Javascript

- Abort
- Blur
- Change
- Click
- DblClick
- Error
- Focus
- KeyDown
- KeyPress
- KeyUp
- Load
- MouseDown
- MouseMove
- MouseOut
- MouseOver
- MouseUp
- Move
- Reset
- Resize
- Select
- Submit
- Unload

Gestore degli eventi

- ✓ Ad ogni evento viene associato un gestore dell'evento
- ✓ Per ottenere il nome del gestore di un evento basta aggiungere il prefisso **`on`** al nome dell'evento stesso
- ✓ Il gestore di un evento permette di associare ad un evento
  - ✓ La semplice invocazione di una funzione
  - ✓ Una sequenza di comandi complessi separati da un punto e virgola

```
<input type="button" value="Hello" onclick="alert('Hello World!');" />
```

Esempi di gestori degli eventi

Click → onclick

- attivato quando l'utente clicca su un elemento

MouseOver → onmouseover

- attivato quando il cursore del mouse si muove sopra un elemento

MouseOut → onmouseout

- attivato quando il cursore del mouse si sposta fuori un elemento

Gestione degli eventi

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 il metodo **`on`**:

`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

`off(T,F)` rimuove **`F`** dalla lista degli *handlers* per l'evento **`T`** negli elementi dell'insieme

Gestione degli eventi: shortcut

Anche in JQuery esistono dei metodi-scorciatoia per eseguire il binding diretto di eventi, ad esempio `click(F)`

```
$(selector).click(function)
```

Aziona la funzione al click del mouse sull'elemento selezionato

```
$(selector).dblclick(function)
```

Aziona la funzione al doppio click del mouse sull'elemento selezionato

```
$(selector).mouseover(function)
```

Aziona la funzione quando il mouse si sposta sull'elemento selezionato

Esempio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>JQuery - Binding di Event Handlers</title>
        <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>

        <style>
            p {
                background: yellow;
                font-weight: bold;
                cursor: pointer;
                padding: 5px;
            }
            span {
                color: red;
            }
        </style>
    </head>
    <body>
        <p>
            Click or double click here.
        </p>
        <span></span>
        <script>
            $("p").on("click", function(event) {
                var str = "( " + event.pageX + ", " + event.pageY + " )";
                $("span").text("Click happened! " + str);
            });
            $("p").on("dblclick", function() {
                $("span").text("Double-click happened in " + this.nodeName);
            });
        </script>

    </body>
</html>
```

ANIMAZIONI

Animazioni di base

JQuery dispone di diverse funzioni per animare gli elementi

- ✓ **`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; se **`C`** è fornito, è una funzione che viene chiamata appena gli elementi sono scomparsi/apparsi (*funzione di callback*)
- ✓ **`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
- ✓ **`animate(style,T,C)`** modifica le proprietà CSS specificate in **`style`**, nel tempo **`T`** (se specificato, può assumere anche i valori **`fast`**, **`slow`** e **`normal`**); **`C`** è la funzione di callback
- ✓ Il filtro **`S1:animated`** consente di filtrare gli elementi selezionati da S1 che sono animati

Esempio

// aggiorna in maniera “dolce” il contenuto dell’elemento con **`id=display`**:
prima fa scomparire 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

```
var dis = $("#display");
dis.fadeOut(1000,function() {
   dis.html("<b>Sorpresa!</b>").fadeIn(1000);
});
```

Esempio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>JQuery - Animazioni di base</title>
        <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>
        <script type="text/javascript">
            $(document).ready(function() {
                $("button").click(function() {
                    $("div").animate({
                        height : 300
                    }, "slow");
                    $("div").animate({
                        width : 300
                    }, "slow");
                    $("div").animate({
                        height : 100
                    }, "slow");
                    $("div").animate({
                        width : 100
                    }, "slow");
                });
            });
        </script>
    </head>

    <body>
        <button>
            Start Animation
        </button>
        <br />
        <br />
        <div style="background: #98bf21;height:100px;width:100px;position:relative"> </div>
    </body>
</html>
```

Manipolazione dello stile

JQuery dispone di tre metodi per manipolare l'attributo di classe degli elementi:

- ✓ `addClass(C)` aggiunge la classe `C` (attributo `class` html)
- ✓ `removeClass(C)` rimuove la classe `C`
- ✓ `hasClass(C)` vero se almeno un elemento dell'insieme ha classe `C`

È inoltre possibile manipolare direttamente gli stili di ogni elemento utilizzando le funzioni

- ✓ `css(P)` restituisce il valore corrente (calcolato) della proprietà CSS, specificata secondo lo standard W3C
- ✓ `css(P,V)` imposta la proprietà CSS `P` al valore `V`

Esempio


```
<html>
<head>
  <style>
  p { margin: 8px; font-size:16px; }
  .selected { color:red; }
  .highlight { background:yellow; }
  </style>
  <script src="http://code.jquery.com/jquery-1.5.js"></script>
</head>
<body>
  <p>Hello</p>
  <p>and</p>
  <p>Goodbye</p>
<script>
  $("p:last").addClass("selected highlight");
  </script>

</body>
</html>
```

Hello

and

Goodbye

Esempio

```
<html>
    <head>
        <title>JQuery - Animazioni di base</title>
        <script type="text/javascript" src="https://code.jquery.com/jquery.js"></script>
        <style>
            div {
                background: yellow;
                border: 1px solid #AAA;
                width: 80px;
                height: 80px;
                margin: 0 5px;
                float: left;
            }
            div.colored {
                background: green;
            }
        </style>
    </head>
    <body>
        <form>
            <input type="button" id="run" value="Run">
        </form>
        <div> </div>
        <div id="mover"> </div>
        <div> </div>
        <script>
            $("#run").click(function() {
                $("div:animated").toggleClass("colored");
            });
            function animateIt() {
                $("#mover").slideToggle("slow", animateIt);
            }

            animateIt();
        </script>

    </body>
</html>
```

Iterazione su un oggetto JQuery

Per eseguire operazioni complesse, è 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

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);
    });
});
```

// 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"));
});
```

Altri esempi

Sulla pagina Moodle del corso:

- ✓ Orologio digitale
- ✓ Conto alla rovescia
- ✓ Timer
- ✓ Validazione input utente
- ✓ Interazione con i controlli di una form (e.g., focus, submit)
- ✓ Interazioni tramite il mouse

Tutorial W3School

- ✓ https://www.w3schools.com/jquery/default.asp

PROGRAMMAZIONE WEB

SCRIPTING CLIENT-SIDE IN JAVASCRIPT
LA LIBRERIA JQUERY

Prof. Ada Bagozi
ada.bagozi@unibs.it