



AJAX e jQuery

Davide Spano

Università di Cagliari

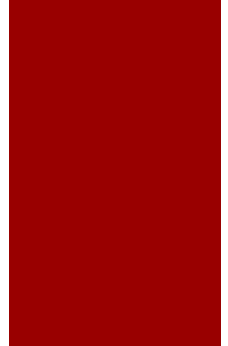
davide.spano@unica.it

Corso di Amministrazione di Sistema

Un framework per javascript



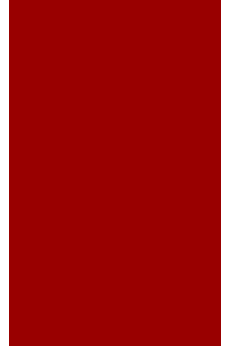
- Come avrete potuto notare dalle scorse lezioni, il meccanismo di gestione della dinamicità delle pagine con javascript è
 - Molto potente
 - Ma altrettanto noioso da programmare
- Soprattutto perché, anche se ne abbiamo accennato poco, diversi browsers hanno alcune differenze nella API
 - Soprattutto il famigerato IE6
- Ma è possibile che nessuno abbia pensato a fare una libreria che risolva questi problemi?
- In realtà ci hanno pensato in molti ma una sta piano diventando uno standard de-facto
- jQuery



jQuery in 2 minuti



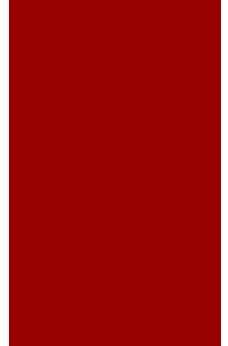
- API pulita per la manipolazione del DOM
- Manipolazione dei CSS
- Gestione degli eventi HTML
- Animazioni ed effetti
- AJAX
- Funzioni di utilità



Altre librerie che lo usano



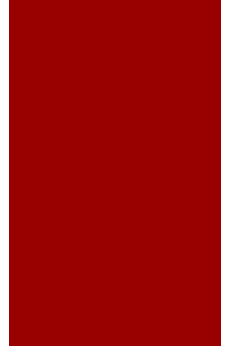
- jQuery UI: Una collezione di estensioni di jQuery che permettono di creare dei widget javascript con poco sforzo
 - La potete utilizzare per il progetto!
 - Tranne che per la variante javascript richiesta
- jQuery mobile: una libreria di widget e funzioni di utilità per lo sviluppo di applicazioni web basate su HTML 5 per dispositivi mobili
- Qunit: una libreria per unit test di codice javascript



Installare jQuery



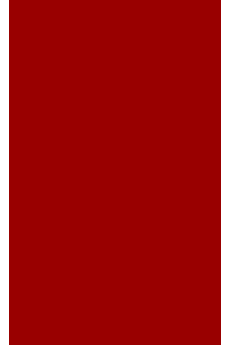
- Potete installare jquery semplicemente scaricando uno script dal sito <http://jquery.com/>
- Dopo di che lo copiate nella cartella del vostro progetto
- E lo collegate come qualsiasi altro script scritto da voi
 - `<script type="text/javascript" src="lib/jquery-1.9.1.min.js"></script>`
- Attenzione alle versioni
 - La 1.9 supporta le versioni di Internet Explorer ≥ 5
 - La 2.0 supporta le versioni di Internet Explorer ≥ 9
 - Per gli altri browser non ci sono problemi particolari



Selezione degli elementi



- La maggior parte del codice che scriviamo in javascript può essere racchiuso nel seguente pattern:
 - Selezionare degli elementi all'interno del DOM
 - Una volta trovato, ci chiamiamo qualche azione sopra
- Gli elementi li selezioniamo in base al nome dell'elemento o ad un id...
- A pensarci bene, all'inizio del corso abbiamo visto un modo molto potente per selezionare gli elementi della pagina HTML: **i selettori CSS!**
- E allora perché reinventare la ruota? Usiamo quelli!
- Ecco il motivo della popolarità di jQuery
 - Ci offre una funzione comoda per selezionare gli elementi con una sintassi pulita e coincisa

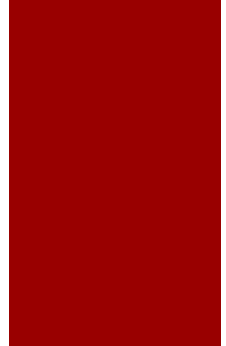


La funzione \$



- I selettori jQuery si utilizzano tramite la funzione \$ e permettono di selezionare elementi HTML in base a
 - Id
 - Classi css
 - Attributi
 - Valore di attributi
 - Ed altro
- Il risultato della selezione è una lista di elementi del DOM, “aumentata” con alcune funzioni di utilità offerte da jQuery, che spesso ci permettono di manipolare l’intera lista con un colpo solo
- Es. Nascondere tutti i paragrafi di un documento HTML
 - `$("p").hide();`
 - Una riga di codice ! 😊

Selezione in jQuery: esempi

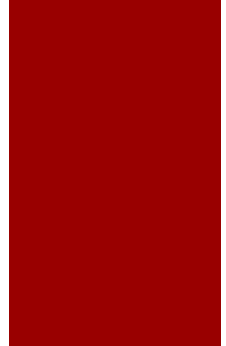


Syntax	Description
<code>\$("*")</code>	Selects all elements
<code>\$(this)</code>	Selects the current HTML element
<code>\$("p.intro")</code>	Selects all <code><p></code> elements with <code>class="intro"</code>
<code>\$("p:first")</code>	Selects the first <code><p></code> element
<code>\$("ul li:first")</code>	Selects the first <code></code> element of the first <code></code>
<code>\$("ul li:first-child")</code>	Selects the first <code></code> element of every <code></code>
<code>\$("[href]")</code>	Selects all elements with an <code>href</code> attribute
<code>\$("a[target='_blank']")</code>	Selects all <code><a></code> elements with a <code>target</code> attribute value equal to <code>"_blank"</code>
<code>\$("a[target!='_blank']")</code>	Selects all <code><a></code> elements with a <code>target</code> attribute value NOT equal to <code>"_blank"</code>
<code>\$(":button")</code>	Selects all <code><button></code> elements and <code><input></code> elements of <code>type="button"</code>
<code>\$("tr:even")</code>	Selects all even <code><tr></code> elements
<code>\$("tr:odd")</code>	Selects all odd <code><tr></code> elements

jQuery manipolazione DOM



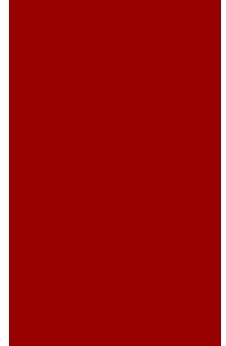
- Una volta che avete selezionato uno o più elementi si possono effettuare varie manipolazioni del DOM
- Attenzione che, per questioni di comodità, praticamente tutte le funzioni che vediamo si possono invocare allo stesso modo sia che il risultato della selezione sia
 - Un **singolo elemento**
 - Una **lista di elementi**
- L'effetto è che la lettura di un valore restituisce tutti i valori di tutti gli elementi (es. uno "stringone" unico)
- La scrittura modifica i valori **di tutti gli elementi della lista**



jQuery: manipolazione DOM (2)



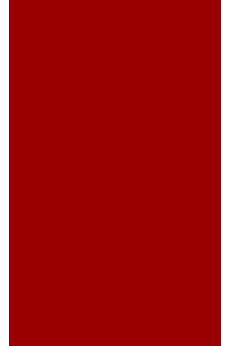
- Di solito le funzioni per fare il get o il set di un valore sono **le stesse**
- Nel caso specifichiate un valore per parametro fa la **set**
- Altrimenti fa la **get**
- Funzioni importanti:
 - **text()** Imposta o restituisce il testo contenuto nell'elemento/elementi selezionati
 - **html()** Imposta o restituisce l'HTML (tutto il markup) per l'elemento/ elementi selezionati
 - **val()** Imposta o restituisce il valore degli elementi di input (qualsiasi sia il loro tipo)
 - **attr(name)** Imposta o restituisce il valore di un attributo (passato per parametro)



jQuery: manipolazione DOM (3)



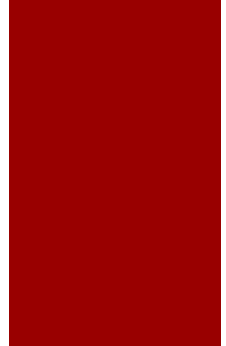
- Cambiamo il testo interno all'elemento con id `hd-news`
`$("#hd-news").text(news.heading);`
- Cambiamo la URL di un'immagine con id `img-news` (N.B. questo ha l'effetto di sostituire l'immagine visualizzata)
`$("#img-news").attr("src", news.image);`
- Salviamo il valore contenuto in un elemento input con id `id-news` all'interno di una variabile
`var input_val = $("#id-news").value(news.id);`
- Sostituiamo la definizione dell'HTML interno ad un elemento con id `txt-news` passando direttamente dell'HTML
`$("#txt-news").html("<h2>Una grandissima notizia!!</h2><p>Cagliari: bla bla</p>");`



jQuery: aggiunta di elementi



- jQuery permette di inserire nuovi elementi tramite quattro funzioni, che prendono per parametro diversi oggetti:
 - Plain text
 - HTML
 - Oggetti jQuery
 - Elementi DOM creati con normale javascript
- Sono tutti gestiti in modo trasparente
- `append()` Inserisce il contenuto all'interno di ognuno degli elementi selezionati, in ultima posizione
- `prepend()` Inserisce il contenuto all'interno di ognuno degli elementi selezionati, in prima posizione
- `after()` Inserisce il contenuto dopo ognuno degli elementi selezionati (come fratello)
- `before()` Inserisce il contenuto prima di ognuno degli elementi selezionati (come fratello)



jQuery: aggiunta di elementi (2)





List item 1

List item 2

List item 3





jQuery: rimozione elementi



- La funzione `remove()` elimina dal DOM l'elemento selezionato e tutti i suoi figli
- La funzione `empty()` elimina dal DOM tutti i figli dell'elemento selezionato

- `$("ol").remove();`

``

`——List item 1`

`——List item 2`

`——List item 3`

``

- `$("ol").empty();`

``

`——List item 1`

`——List item 2`

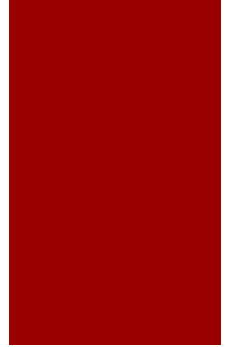
`——List item 3`

``

jQuery: modificare le classi CSS



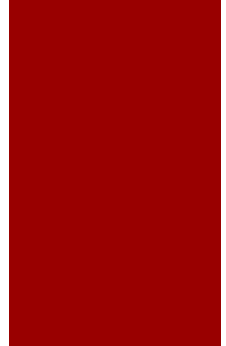
- La scorsa lezione abbiamo visto che il javascript ci permette di aggiungere le classi modificando i valori dell'attributo class
- Però è nostro compito gestire la lista di valori separati da spazi
- jQuery ci mette a disposizione delle funzioni molto comode
 - `addClass()` aggiunge la classe passata alla lista di quelle possedute dall'elemento
 - `removeClass()` rimuove la classe passata per parametro dall'elemento



jQuery: modificare gli stili



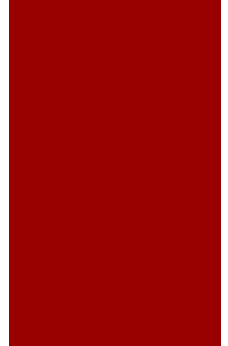
- Oltre ad aggiungere e togliere classi, si possono modificare direttamente anche gli stili di più elementi con una istruzione singola
- La funzione `css()` ci permette appunto di accedere/leggere il valore degli stili
- Leggere il background color degli elementi `p`
 - `$("#p").css("background-color")`
- Impostare il background color di tutti i `p` a giallo
 - `$("#p").css("background-color", "yellow");`
- Impostare più proprietà con una singola istruzione
 - `$("#p").css({`
 - `"background-color": "yellow",`
 - `"font-size": "200%"``});`



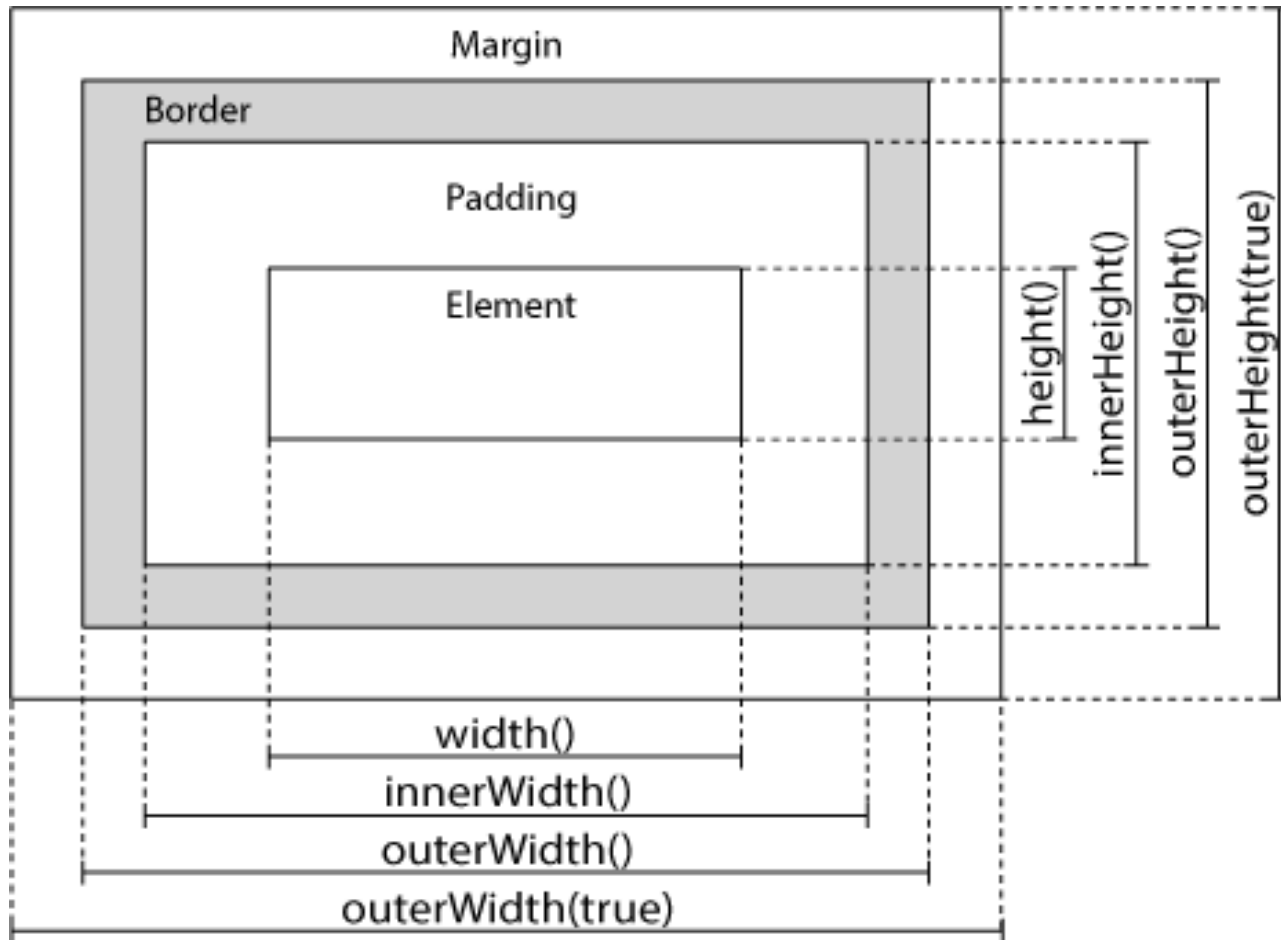
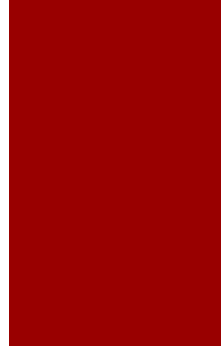
jQuery: dimensioni



- Nella scorsa lezione, avevamo visto che le dimensioni ottenute dall'attributo **style** sono quelle impostate nell'attributo
 - E non quelle dell'elemento quando viene disegnato
- jQuery ci offre una serie di funzioni che restituiscono le dimensioni *reali* dei box
- Molto utili quando si deve fare qualche lavoro di grafica avanzata



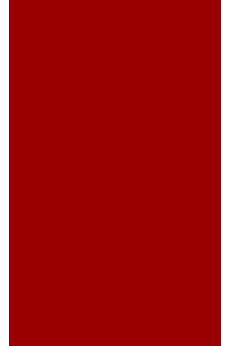
jQuery: dimensioni (2)



jQuery: event handlers



- Gli event handlers si agganciano sempre utilizzando delle funzioni particolari sugli elementi selezionati
- Il primo è utilizzare la funzione **on**, specificando il nome dell'evento (senza «on» ...) e la funzione da agganciare
 - `$('#foo').on('click', function () {
 alert('User clicked on "foo."');
});`
 - Si possono anche specificare più eventi in un colpo solo
`$('#foo').on('mouseenter mouseleave', function
() {
 $(this).toggleClass('entered');
});`
- jQuery mette anche a disposizione degli funzioni alias per la **on**, che si chiamano come il nome dell'evento
 - Es. il click
`$('#foo').click(function () {
 alert('User clicked on "foo."');
});`



jQuery: struttura codice



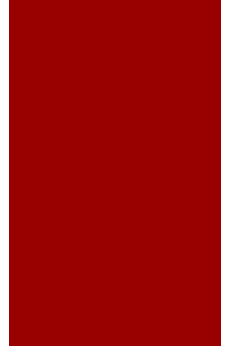
- Le istruzioni per agganciare gli event handlers si effettuano solitamente al termine del caricamento del documento.
- In jQuery basta specificare una funzione (anche senza nome) per l'evento **ready** del **document**
- In gran parte dei siti che utilizzano jQuery troverete questo schema:

```
$(document).ready(function () {  
  // istruzioni per agganciare event handlers  
    $("#next-news").click(function () {  
      // ...  
    });  
    // ...  
  
  // altre funzioni  
  function changeNews(news) {  
    // ...  
  }  
});
```

Ajax



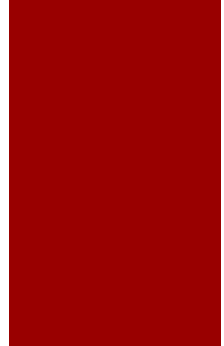
- Significa *Asynchronous Javascript and XML*
- Termine coniato nel 2005 circa
- In realtà più che il vero significato dell'acronimo, questo termine viene utilizzato per le tecniche di sviluppo javascript che consentono di scambiare dati con il server ***senza ricaricare la pagina***
- Un esempio **Google Maps**
- Vantaggi:
 - Una migliore interfaccia utente
 - In moltissimi casi riduce il traffico di rete



Come funziona...



- Il tutto è nato dall'introduzione dell'oggetto **XMLHttpRequest**
 - Prima come plugin aggiuntivo del browser (IE 5)
 - Poi come oggetto incorporato direttamente nell'interprete
- Questo oggetto consente di inviare delle richieste HTTP al server
 - Quando la pagina web è stata già caricata
 - Direttamente da codice javascript
 - In modo asincrono
- Cosa vuol dire?
 - Vuol dire che si possono richiedere dei dati solo quando sono necessari, senza per forza caricarli da subito
 - La pagina viene modificata in base a cosa fa l'utente e cosa viene inviato dal server, senza ricaricare
 - L'applicazione web diventa dinamica



Esempi:



■ Google Maps

- La mappa è un insieme di quadrati
- Vengono caricati solo quando serve
- Diversi livelli di dettaglio

■ Completamento automatico e suggerimenti

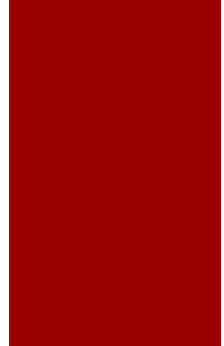
- Si prende quello che l'utente scrive
- E si crea una lista di suggerimenti
- Oppure si suggerisce un unico completamento per l'input
- Ovviamente i dati dei suggerimenti stanno sul server

■ Validazione

- Si prende quello che scrive l'utente e si manda al server
- Che restituisce la lista degli errori

■ Fotogallery

- Si caricano le immagini solo quando l'utente ci clicca sopra
- Non si ricarica la pagina...



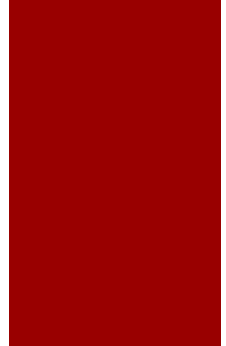


Demo...

XmlHttpRequest



- Non è uniforme su tutti i browser
 - In alcune versioni di IE si crea in modo differente
- Permette di ricevere dati
 - Testuali
 - In formato XML
- Noi useremo dei dati testuali
 - Ma codificati in modo furbo
- Permette di inviare delle richieste asincrone al server
 - Cioè l'esecuzione del vostro codice non si blocca in attesa della risposta del server
 - Si registra una funzione che l'interprete richiama non appena la risposta arriva
- Ha un'interfaccia per il programmatore un po' confusionaria
- Perciò noi utilizziamo una sintassi unificata offerta da jQuery...



JSON



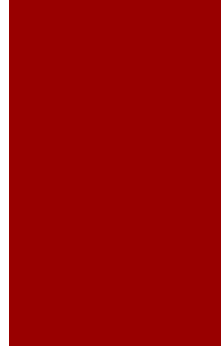
- Il formato testuale che si utilizza di solito per scambiare informazioni con il server tramite ajax si chiama **JSON**
- Che sta per *JavaScript Object Notation*
- L'abbiamo già incontrata, è la short object notation con qualche restrizione in più (di poco conto)...
- Ci permette di non dover decodificare i valori quando arrivano al javascript
- E siccome è supportata la codifica degli array in JSON anche dal PHP, non dobbiamo sforzarci molto neanche lato server


```
{  
    "id" : 1,  
    "heading": "Il governo Letta...",  
    "img": "http://ilmioserver/miaimg.jpg",  
    "text": "ROMA: bla bla..."  
}
```

Ajax in azione: scorrimento news




- Lo trovate nella cartella ajax sulla mia cartella SVN
- <http://spano.sc.unica.it/davide/ajax/index.php>
- Se si clicca sul bottone *Avanti* o *Indietro* la notizia cambia
- Ma la pagina non viene ricaricata




**AMM News**


Tutte le ultime notizie con ajax

Home

 **Navigazione**
[Home](#)


 **Link esterni**
[La Repubblica](#)
[L'Unione Sarda](#)
[La nuova Sardegna](#)

Letta: "L'Europa non sia una gabbia ma un aiuto per la crescita"



"L'Europa non deve tradursi in una gabbia di vincoli regole e procedure che finiscono spesso per limitare l'azione di tutti, cittadini, famiglie e imprese". E soprattutto: non basta l'unione monetaria, ma servono "gli Stati Uniti d'Europa". Ne è convinto il premier Enrico Letta, intervenuto questa mattina nell'Aula del Senato in occasione delle comunicazioni sul vertice straordinario dell'Ue in programma per domani, ricordando come il decreto approvato all'ultimo Cdm vada in questa direzione.

[Indietro](#) [Avanti](#)

 **Istruzioni**

Usare i bottoni Avanti/Indietro per cambiare la notizia



**AMM News**

Tutte le ultime notizie con ajax

Home

 **Navigazione**
[Home](#)

 **Link esterni**
[La Repubblica](#)
[L'Unione Sarda](#)
[La nuova Sardegna](#)

Ineleggibilità Berlusconi, oggi al Senato la partita decisiva sul presidente della Giunta



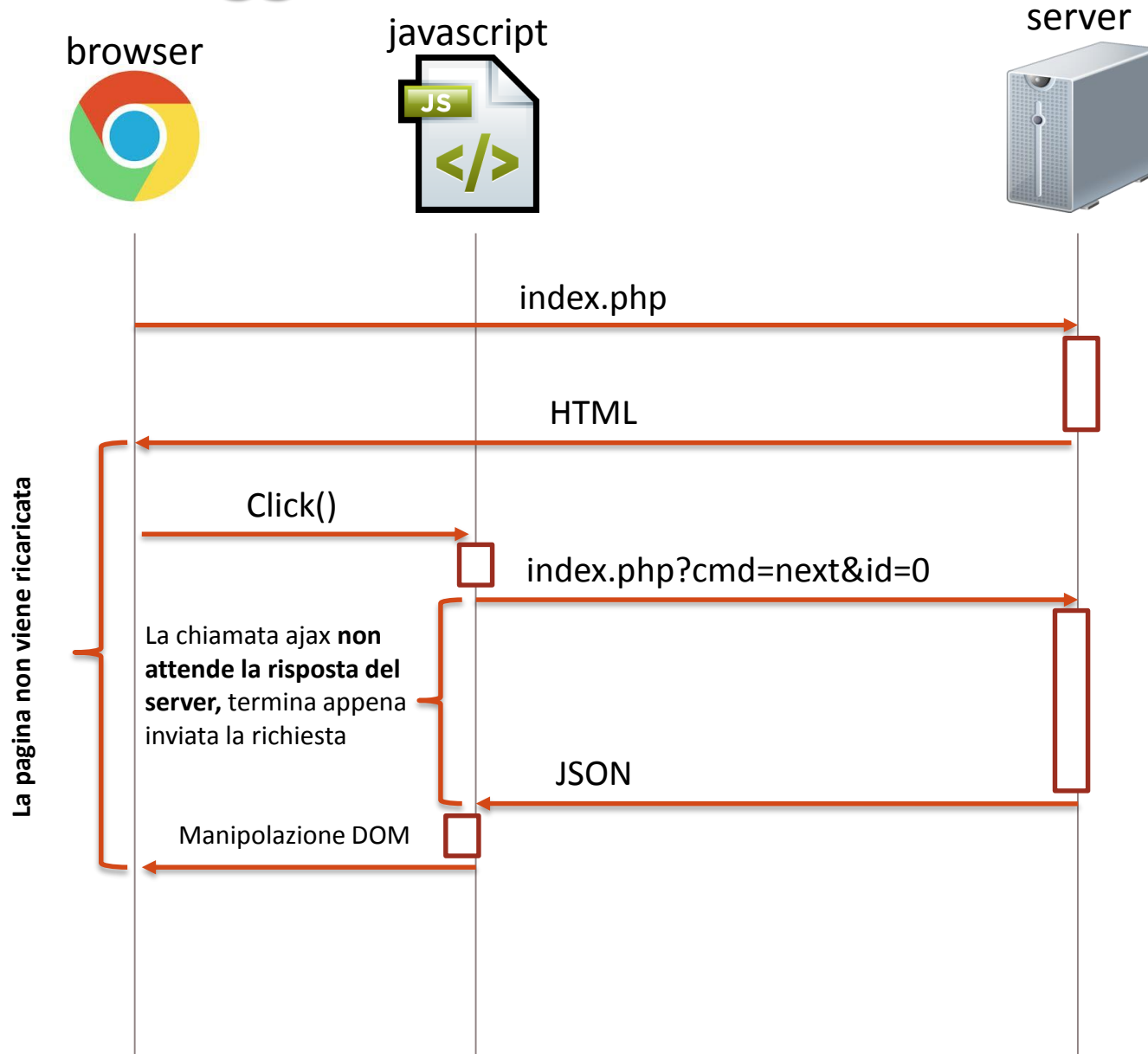
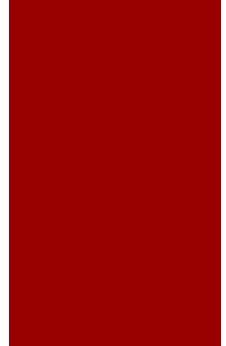
E' un passaggio decisivo per le sorti del governo quello in programma oggi pomeriggio al Senato. Nella sede di Sant'ivo alla Sapienza si riunisce la Giunta per le immunità e le elezioni. All'ordine del giorno c'è l'elezione del presidente e a breve la proposta per le ineleggibilità di Silvio Berlusconi, che il Movimento Cinque Stelle ha già detto di voler depositare. Gli equilibri politici in giunta si decidono oggi. E' cruciale sarà la compattezza del Pd. Il centrodestra con Lega e Gal ha infatti otto senatori. Otto ne ha il Pd, quattro M5S, e uno Sel. Se tiene l'asse tra il Pdl, la Lega e il Pd potrebbe essere eletto il leghista Raffaele Volpi. Ma se una parte del Pd (ne bastano quattro) seguirà le indicazioni espresse nei giorni scorsi da Felice Casson (contrario a votare un leghista presidente), allora la Giunta avrà un altro presidente e vincerà verso posizioni ostili a Silvio Berlusconi, che potrebbe essere dichiarato ineleggibile in quanto concessionario pubblico, ai sensi della legge 361 del 1957.

[Indietro](#) [Avanti](#)

 **Istruzioni**

Usare i bottoni Avanti/Indietro per cambiare la notizia

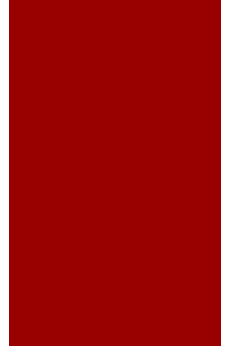
Messaggi



La funzione \$.ajax()



- jQuery ci mette a disposizione una funzione molto comoda che ci permette di gestire le differenti implementazioni in modo trasparente
- La funzione prende per parametro un solo oggetto
- Al suo interno si possono specificare un insieme di campi che permettono configurare la richiesta
- I parametri di base sono:
 - **url**: indirizzo al quale inviare la chiamata
 - **success**: la funzione da chiamare nel caso la richiesta vada a buon fine
 - **error**: la funzione da chiamare nel caso la richiesta fallisca
 - **data**: i dati da passare al server (POST o GET)
 - **dataType**: la codifica dei dati inviati dal server (per noi JSON)
 - **synch** (false): se impostato a true, si attende la risposta del server prima di eseguire l'istruzione javascript dopo la chiamata ajax
- Gli altri potete trovarli qui <http://api.jquery.com/jQuery.ajax/>



Esempio \$.ajax(): next news



```
$("#next-news").click(function () {  
    $.ajax({  
        url: "index.php",  
        data: {  
            cmd: "next",  
            id: $("#id-news").attr("value")  
        },  
        dataType: 'json',  
        success: function (data, state) {  
            changeNews(data);  
        },  
        error: function (data, state) {  
        }  
    });  
});
```

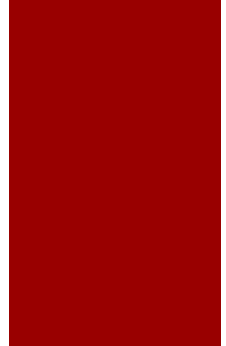
JSON, è un
oggetto javascript
direttamente
utilizzabile

```
function changeNews(news) {  
    $("#hd-news").text(news.heading);  
    $("#img-news").attr("src", news.image);  
    $("#id-news").attr("value", news.id);  
    $("#txt-news").text(news.text);  
}
```

Ajax ed MVC



- Ma come vengono gestite le richieste ajax sul server?
- Sono sempre delle richieste HTTP, non cambia nulla rispetto a quelle inviate direttamente del browser
- Solo che questa volta dovete restituire del JSON e non HTML
- Quindi che cosa cambia rispetto a prima?
 - **Il tipo di vista!**
- Il controller deve smistare le richieste ajax e creare delle viste che riempiano del JSON
 - Il modo più semplice per discriminarle dalle altre richieste è utilizzare una parametro (es. ajax)
 - Ma potete pure utilizzare solo il comando...
- Non dovete quindi caricare la master page in questo caso
 - È un if...



Creare JSON da PHP



- Il modo più semplice è quello di creare degli array associativi della stessa struttura del JSON
- Dopo di che si può invocare il metodo **json_encode**
- Segue un esempio di «vista» JSON

```
<?php
header('Cache-Control: no-cache, must-revalidate');
header('Expires: Mon, 26 Jul 1997 05:00:00 GMT');
header('Content-type: application/json');

$json = array();

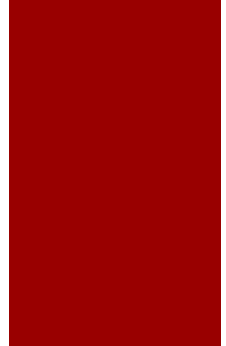
$json["id"] = $news->getId();
$json["heading"] = $news->getHeading();
$json["image"] = $news->getImage();
$json["text"] = $news->getText();

echo json_encode($json);
?>
```


Same Origin Policy



- Per motivi di sicurezza, l'interprete javascript non permette di effettuare richieste ajax verso **origini diverse rispetto a quella della pagina attuale**
- Cosa vuol dire origine?
 - Stesso dominio
 - Stesso protocollo
 - Stessa porta TCP
- Esempio <http://www.example.com/dir/page.html>



URL controllato	Risultato	Motivo
http://www.example.com/dir/page.html	Successo	Stesso protocollo e host
http://www.example.com/dir2/other.html	Successo	Stesso protocollo e host
http://www.example.com:81/dir/other.html	Fallimento	Stesso protocollo e host ma porta diversa
https://www.example.com/dir/other.html	Fallimento	Protocollo diverso
http://en.example.com/dir/other.html	Fallimento	Host diverso
http://example.com/dir/other.html	Fallimento	Host diverso (è necessario che siano esattamente uguali)
http://v2.www.example.com/dir/other.html	Fallimento	Host diverso (è necessario che siano esattamente uguali)

Ajax nel progetto

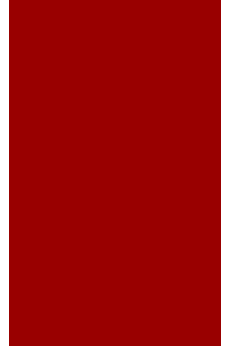


■ Validazione

- Prima di effettuare la submit, si fa una richiesta ajax che verifica **sul server** l'ammissibilità dei valori inseriti nei form
- A seconda del risultato della validazione, vanno mostrati dei messaggi appositi **senza ricaricare la pagina**

■ Suggerimenti/autocompletion

- Ogni volta che viene premuto un tasto, inviare il valore di un text field al server, che restituirà una lista di valori
- Questa lista va mostrata sotto il text-field che mostra i valori ricevuti dal server
- L'utente può cliccare su uno di questi valori per completare il valore scritto parzialmente sul text field.

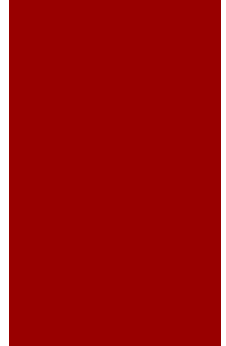


Ajax nel progetto (2)



■ Fotogallery

- È simile all'esempio delle news
- Un pulsante avanti ed uno indietro per cambiare la foto correntemente visualizzate
- **Non caricate tutte le foto all'inizio**



Riferimenti

- David Flanagan: Javascript, the definitive guide
 - Cap 19
- jQuery reference
 - <http://jquery.com/>

