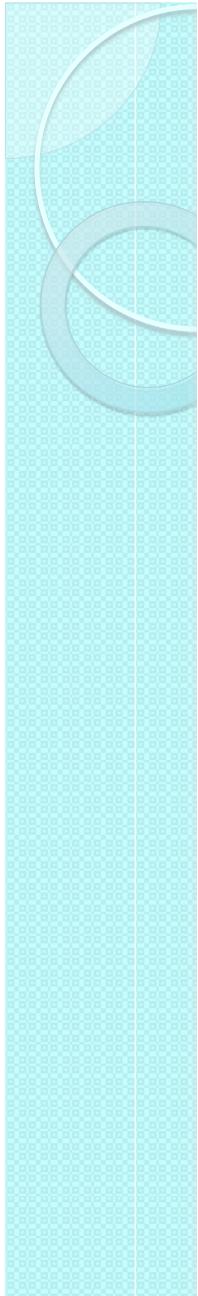
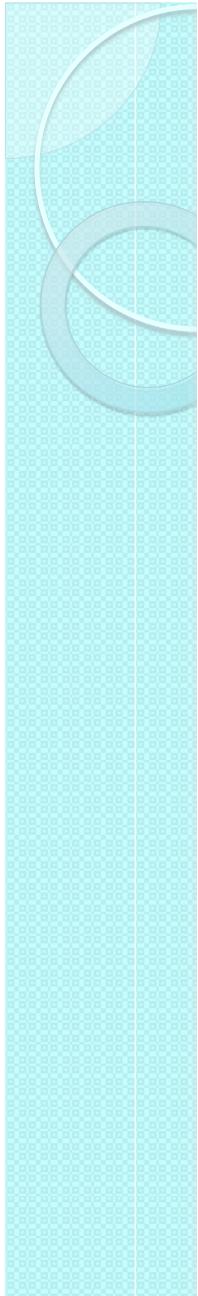


AJAX JQuery



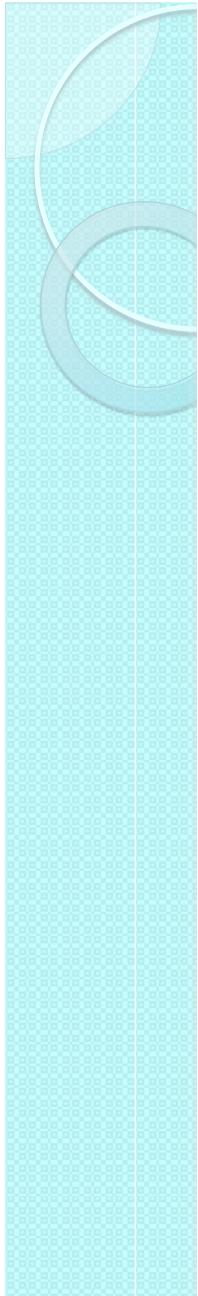
Introduzione

- Jesse James Garrett nel **2005** ideò il termine **Ajax** (*Asynchronous JavaScript and XML*).
- Scambio di dati tra client e server in background
- Aggiornamento dati senza esplicito refresh della pagina
- Rich Internet Application: gmail, google docs, facebook, twitter, ...



Funzioni Ajax di jQuery

- load
- ajax
- get
- post
- getJson



load

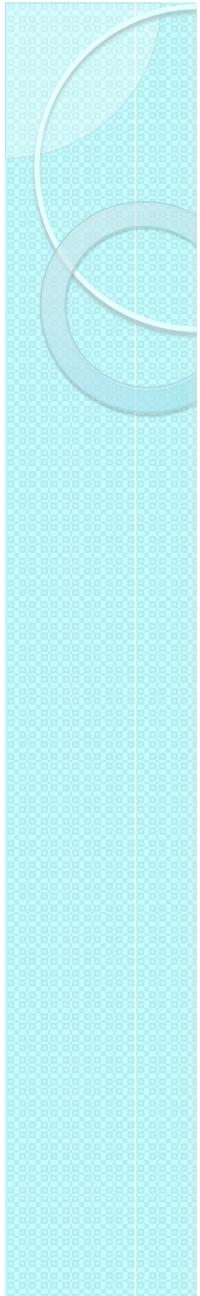
definizione

```
$ (selettore) .load( url [filtro],  
[dati], [funzione(risposta,  
status,xhr) ] )
```

- **url** indirizzo a cui è inviata la richiesta
- **[filtro]** filtro applicato al DOM della risposta ricevuta. Si utilizza la stessa sintassi dei selettori jQuery
- **[dati]** parametri opzionali inviati nel corpo della richiesta
- **[funzione]** funzione richiamata quando la richiesta è stata completata

Esempio Ajax con jQuery

```
$('#caricaqui').load("esempio.html")
```



load

esempi

```
/*l'elemento con id="ricevuto" carica come contenuto la  
pagina es.html. A caricamento avvenuto viene mostrata  
una finestra di dialogo  
*/  
$("#ricevuto").load("es.html",function(dati) {  
    alert("dati caricati");  
})  
/* al click sugli elementi di class="caricaqui" viene  
caricato in essi l'elemento della pagina es.html con  
id="speciale"  
*/  
$(".caricaqui").click(function() {  
    $(this).load("es.html #speciale");  
})
```



\$.ajax(configurazione)

1 / 2

Nome	Tipo	Descrizione
url	Stringa	Indirizzo a cui inviare la richiesta
type	Stringa	GET (default) o POST
data	Oggetto	contenuto della richiesta (parametri inviati alla pagina a cui si effettua la richiesta)
dataType	String	Tipologia di dati ottenuti come risposta: <ul style="list-style-type: none">xmlhtmljson: formato di interscambio dati javascriptjsonp: oggetto json con paddingscript: risposta processato come un javascripttext (default): testo semplice
timeout	Numero	Tempo massimo, espresso in millisecondi, per ottenere risposta dal server. In caso non si ottenga risposta viene richiamata la funzione di callback associata all'handler error

\$.ajax(configurazione)

2/2

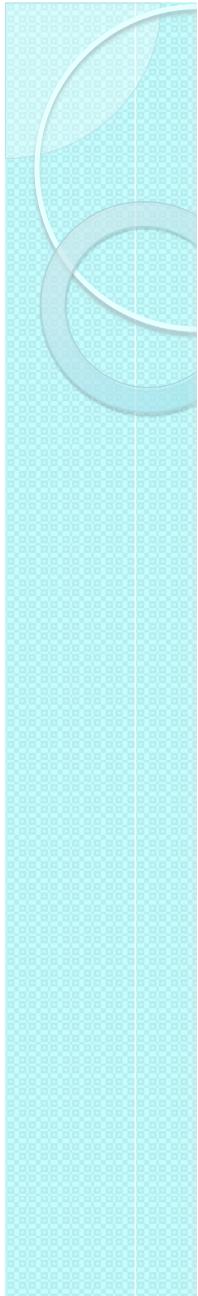
Nome	Tipo	Descrizione
success	Funzione	funzione richiamata quando la richiesta al server ha avuto successo. Il contenuto della risposta è contenuto come primo parametro, mentre lo status come secondo parametro
error	Funzione	funzione richiamata quando si è verificato un errore nella richiesta al server. Questa funzione ha 3 parametri: <ul style="list-style-type: none">• oggetto xhr• status (sempre error)• oggetto exception xhr
complete	Funzione	funzione chiamata quando la richiesta è stata completata. Ha due parametri: xhr e status. Questa funzione è chiamata dopo success o error
beforeSend	Funzione	funzione chiamate prima di iniziare la richiesta al server
async	Booleano	false, effettua una richiesta sincrona. True (default) <small>11</small>



Esempi \$.ajax()

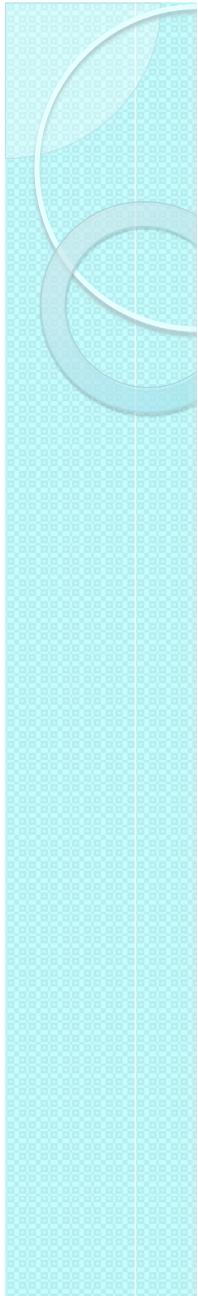
```
$.ajax({  
    url: 'test.html',  
    success: function(data) {  
        $('.result').html(data);  
        alert('caricato');  
    }  
});
```

```
$.ajax({  
    url: 'ajax/test.php',  
    data: {'regioni[]': ['piemonte', "lazio"], colore:'arancione'},  
    success: function(data) {  
        $('.result').html(data);  
        alert('Load was performed.');  
    },  
    dataType: 'xml'  
});
```



Scorciatoie

- `$.get(url, parameters, callback)`
- `$.post(url, parameters, callback)`
- `$.ajax(...).success(fn)`
- `$.ajax(...).error(fn)`
- `$.ajax(...).complete(fn)`

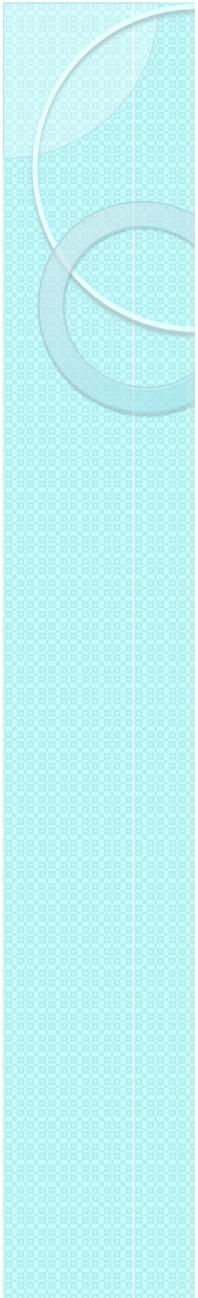


Funzioni Globali

`$(selettore).funzioneGlobale()`

- `ajaxStart(callback)`
- `ajaxSend(callback)`
- `ajaxSuccess(callback)`
- `ajaxError(callback)`
- `ajaxComplete(callback)`
- `ajaxStop(callback)`

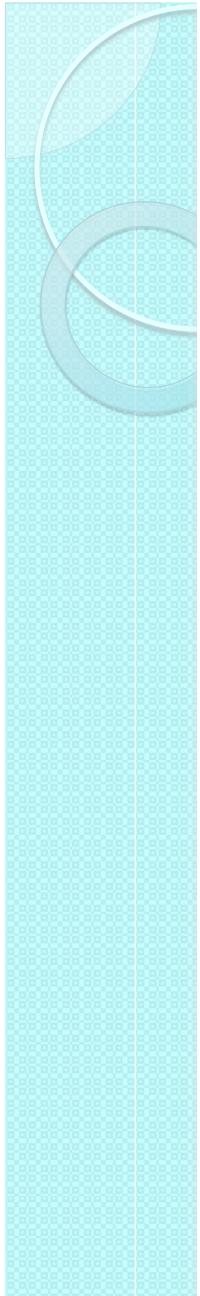
PROCESSAMENTO RISPOSTA



HTML

- Il server restituisce direttamente codice html nella risposta

```
$ .ajax ({  
    dataType:'html',  
    url:'aggiungi.html',  
} ) .success (function (data) {  
    //modifica html ricevuto  
    $(data).children("p") .addClass (" .ricevuto " );  
    $("#ricevitore") .append (data);  
})
```



XML

- Viene fornita una risposta in formato XML
- Il DOM XML può essere gestito con i *classici* selettori di jQuery (`.find` `.children` `.each` `.attr`)
- Esempio

```
<?xml version="1.0" encoding="UTF-8"?>
<regioni>
    <regione name="Abruzzo">
        <abitanti totale="1338898">
            <uomini>650752</uomini>
            <donne>688146</donne>
        </abitanti>
    </regione>
</regioni>
```



XML esempio

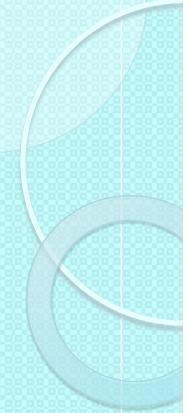
```
$ .ajax( {
    url:'ajax/regioni.xml',
    dataType:"xml",
    success(function(data) {
        $(data).find("regione").each(function() {
            var nome=$ (this).attr("name");
            var divR=$ ("<div>").append($ ("<h1/>")
                .text(nome));
            var abitanti=$ (this).find("abitanti").get(0);
            $(divR).append($ ("<h2>")
                .text("abitanti "+$ (abitanti).attr
                    ('totale')));
        });
        $("#regioni").append(divR);
    })
})
```

JSON 1/3

```
var JSON = {  
    proprietA1: 'Valore',  
    proprietA2: 'Valore',  
    proprietAN: 'Valore'  
}
```

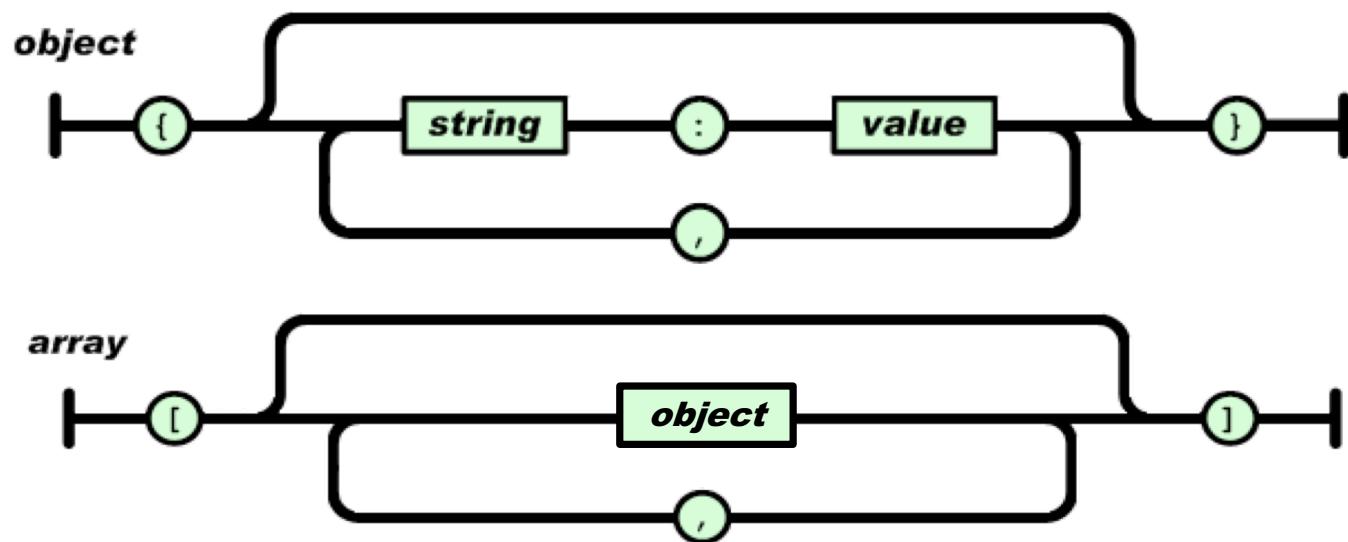
JSON 2/3

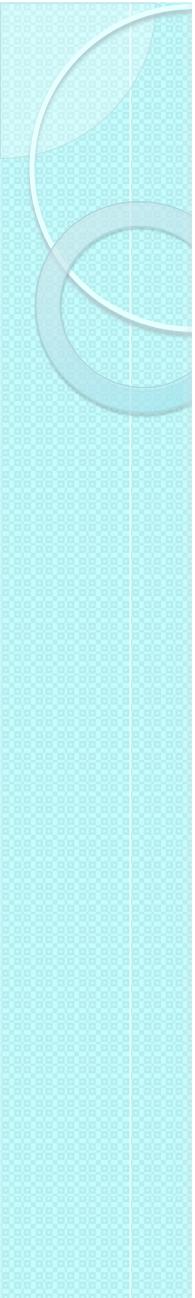
```
{  
    "home": "Home.it",  
    "link": "http://www....",  
    "argomento": "Standard del web",  
    "aree": [  
        {  
            "area": "CSS",  
            "url": "http://css...."  
        },  
        {  
            "area": "Basic",  
            "url": "http://...."  
        }  
    ]  
}
```



JSON 2/3

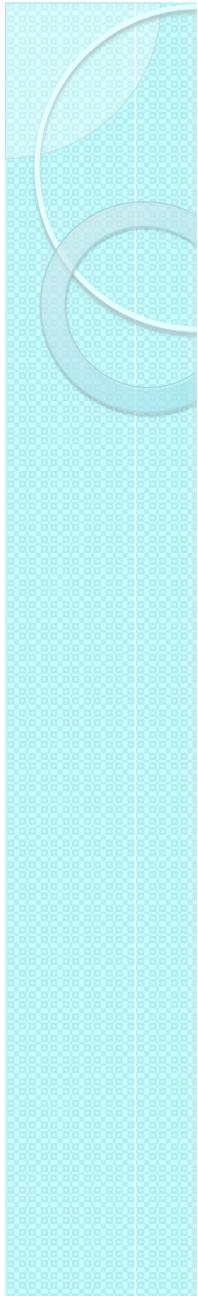
- Javascript Object Notation
- Formato di interscambio dati
- Due elementi principali





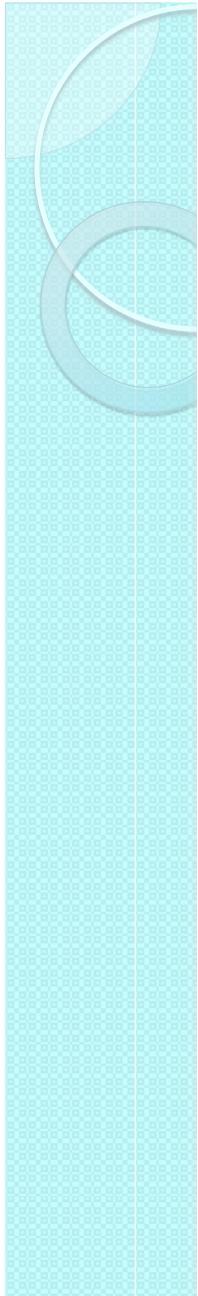
Esempio JSON

```
[{"regione":"Abruzzo","abitanti":1338898,"uomini":650752,"donne":688146},  
 {"regione":"Basilicata","abitanti":588879,"uomini":288274,"donne":300605},  
 {"regione":"Calabria","abitanti":2009330,"uomini":979003,"donne":1030327},  
 {"regione":"Campania","abitanti":5824662,"uomini":2824935,"donne":299727}]  
  
$.ajax({  
    url: 'ajax/regioni.json',  
    dataType: "json",  
    success: function(data) {  
        $.each(data, function(chiave, elemento) {  
            var divR = $("<div>").append($("h1").text(elemento.regione));  
            $(divR).append($("h2").text("abitanti " + elemento.abitanti));  
            $("#regioni").append(divR);  
        })  
    })  
})
```



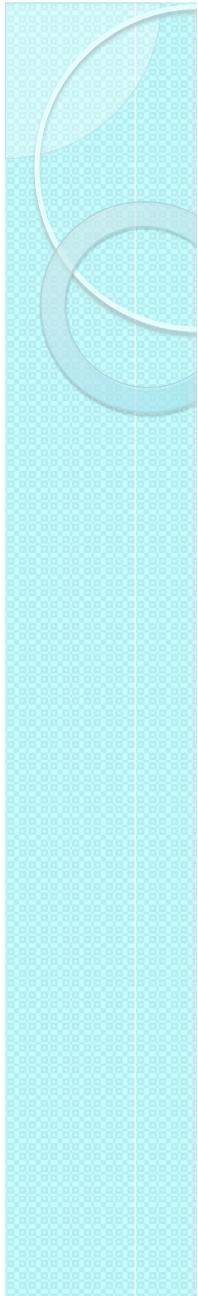
Limitazioni di Sicurezza

- È possibile effettuare richieste solo sullo stesso sito della pagina che contiene lo script
- Come si superano?
 - W3C Cross-Origin Resource Sharing
 - Pagine proxy
 - <script> jsonp



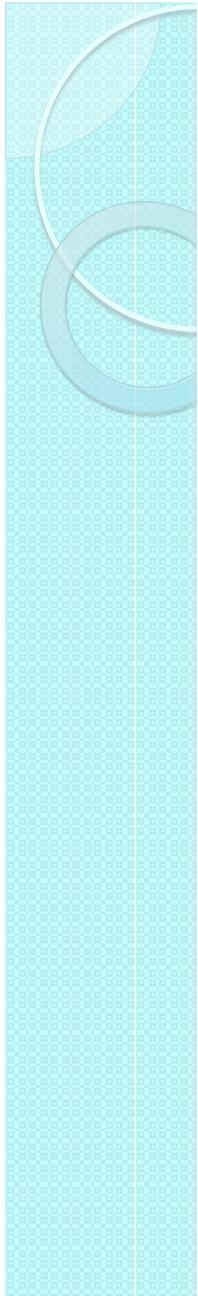
Cross-Origin Resource Sharing

- <http://www.w3.org/TR/cors/>
- Proposta del 2010
- Specifica negli header della risposta dei client che possono accedere ai dati
 - Access-Control-Allow-Origin: *



Proxy

- La richiesta ajax non viene effettuata direttamente al sito da cui si vogliono ricevere dati
- la richiesta viene inviata a una pagina web dinamica (proxy) presente sullo stesso sito in cui è ospitato lo script.
- Il proxy (php, jsp, asp, ..) accede direttamente ai dati contenuti su un altro sito.



<script>

- L'elemento script nell'attributo src può contenere riferimenti a siti esterni
- Permette il caricamento di script da altri server
- Oltra agli script i file caricati possono contenere altri dati
- Gli oggetti definiti negli script caricati possono accedere ai dati residenti sul server d'origine
- Esempi:
 - Google maps, Google calendar, youtube