Ajax e jQuery

Emiliano Castellina
Dipartimento di Automatica e Informatica
Politecnico di Torino

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, …

XMLHttpRequest (XHR) Metodi

- Oggetto Javascript che permette la comunicazione tra client e server

Metodo	Descrizione
abort()	La richiesta in esecuzione viene cancellata
getAllResponseHeader()	Singola Stringa contenente tutti i nomi e i valori degli header della risposta
getResponseHeader (nomeHeader)	Stringa con il valore dell'header con nomeHeader
open(method, url, async, username,password)	Imposta i parametri della connessione con il client • Method: post o get • url della pagina a cui connettersi(Limitato allo stesso server) • async true se la chimata è asincrona, false sincrona • Username,password dati per connessioni autenticate
send(content)	Avvia la richiesta con il content (opzionale) come body
setRequestHeader(name,request)	Imposta l'header della richiesta

XMLHttpRequest (XHR) Proprietà

Proprietà	Descrizione
readyState	Numero intero che indica lo stato della richiesta: • 0 Non Inizializzato • 1 Connessione Aperta • 2 Dati inviati • 3 Ricezione dati in corso • **4** Ricezione dati completata
onreadystatechange	Permette di registrare una funzione richiamata ogni volta che il readyState cambia
responseText	Il contenuto della risposta
responseXML	Se il contenuto della risposta è XML, questa proprietà restituisce il DOM XML generato a partire dal contenuto
status	Status HTTP della risposta: • 200 OK • 404 NOT FOUND • 301 MOVED PERMANENTLY • ….
statusText	Valore testuale dello status della risposta (OK, NOT FOUND, …)

Esempio Ajax tradizionale

```
var xhr;
if (window.XMLHttpRequest) {
      xhr = new XMLHttpRequest();
} else if (window.ActiveXObject) {
      xhr = new ActiveXObject("Msxml2.XMLHTTP");
} else {
      throw new Error("Ajax non supportato");
}
xhr.onreadystatechange = function() {
 if (xhr.readyState == 4) {
      if (xhr.status == 200){
      var elem=document.getElementById('caricaqui');
      elem.innerHTML =xhr.responseText;
}
xhr.open('GET','/esempio.html');
xhr.send();
```

Completata

OK

Esempio Ajax con jQuery

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

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 del selettori jQuery
- [dati] parametri opzionali inviati nel corpo della richiesta
- [funzione] funzione richiamata quando la richiesta è stata completata

load esempi

```
/*l'elemento con id="ricevuto" carica come contenuto la pagina es.html. A caricamento avvenuto viene mostrata una finesta di dialogo
*/
$("#ricevuto").load("es.html",function(dati){
      alert("dati caricati");
})
/* al click sugli elementi di class="cariqui" 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: • xml • html • json: formato di interscambio dati javascript • jsonp: oggetto json con *padding* • script: risposta processato come un javascript • text (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: • 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)

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>
```

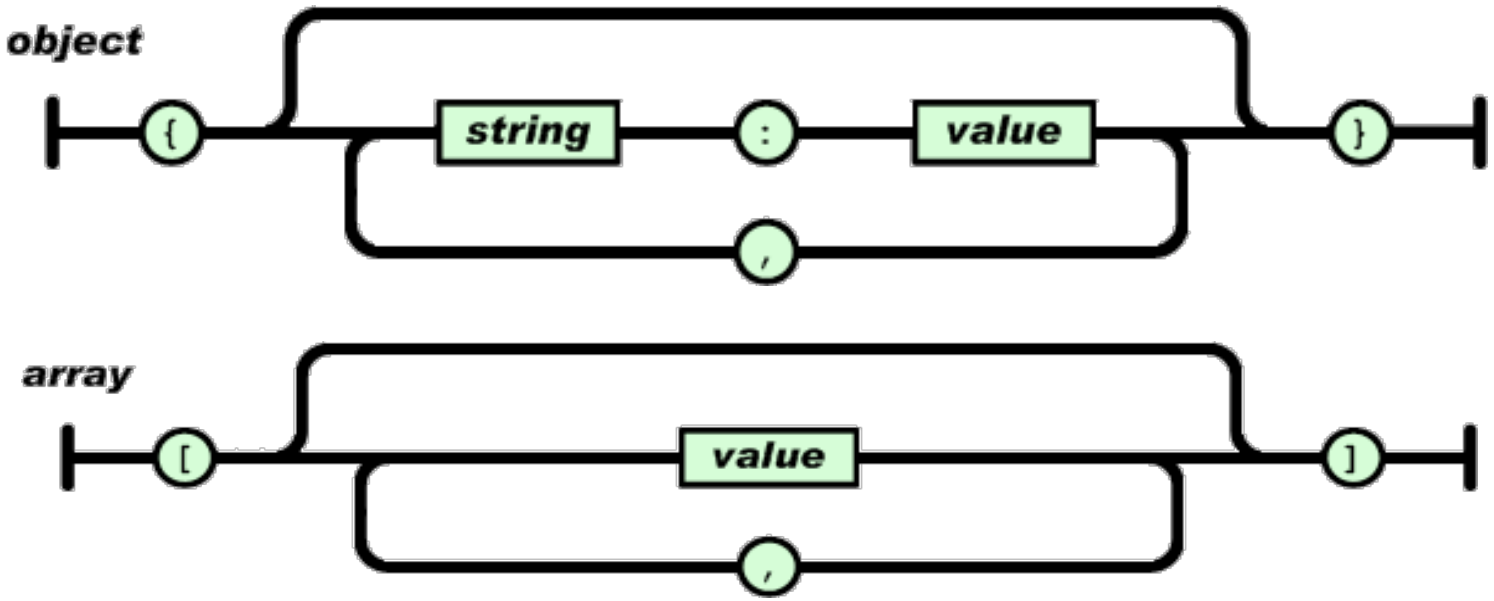
XML esempio

```
$.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

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



http://www.json.org

Esempio JSON

```
[{"regione":"Abruzzo","abitanti":"1338898","uomini":"650752","donne":"688
146"},
{"regione":"Basilicata","abitanti":"588879","uomini":"288274","donne":"30
0605"},
{"regione":"Calabria","abitanti":"2009330","uomini":"979003","donne":"103
0327"},
{"regione":"Campania","abitanti":"5824662","uomini":"2824935","donne":"29
99727"}]
```

```
$.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 sulle 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

jsonp

- Utilizza l'elemento script per ricevere dati da un sito esterno
- La richiesta ha come parametro obbligatorio callback=nomefunzione
  - jQuery lo fa in automatico
- Quando lo script viene caricato, chiama la funzione nomefunzione passando come parametro un oggetto json

Esempio JSONP

```
$.ajax({
    url: 'http://elite.polito.it/files/courses/01NPYPD/ajax/
demografia.php',
    dataType: "jsonp"
 }).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);
         })
 })
```