Seminario Introduzione ad Ajax

Linguaggi per il Web

Ingegneria Informatica, Ingegneria dell'Informazione,
Sapienza Università di Roma, sede di Latina

17 Dicembre 2013

Dott. Corrado Di Benedetto

Argomenti

- Introduzione
- Che cosa è Ajax?
- Come funziona Ajax ?
- Dove usare Ajax?
- Esempi di tecnologia Ajax
- Pregi
- Difetti
- Compatibilità dei browser
- Oggetto XMLHttpRequest
  - Interfaccia Standard W3C
  - Creazione dell'Oggetto
  - Invio della Richiesta
  - Risposta del server
  - Proprietà
  - Eventi
- JSON
- Framework Ajax

Introduzione

- AJAX, acronimo di **A**synchronous **J**avaScript **A**nd **X**ML, è stato introdotto nel 2005 da Jesse James Garrett (padre di Ajax) e fu reso popolare sempre nel 2005 da Google.

- Ajax permette di creare una interfaccia web dinamica, come ad **esempio** il completamento del testo della search box di Google, dove il server restituisce una lista di suggerimenti.

Che cosa è Ajax ?

- Ajax non è una singola tecnologia, ma bensì una **collezione di tecnologie** che insieme permettono la realizzazione di applicazioni web interattive.

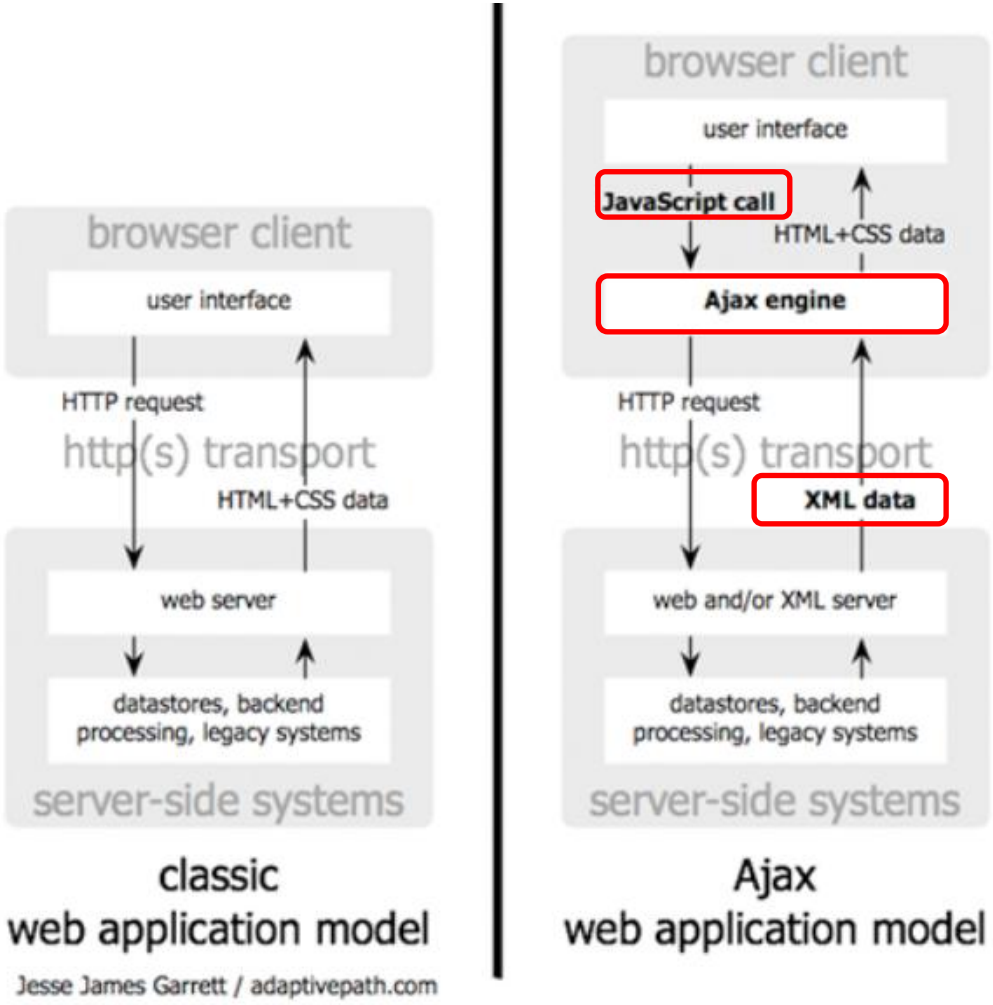
- Oggi il termine Ajax si è **esteso** ad identificare tutte le tecnologie native dei browser che permettono la **comunicazione asincrona** con un server di backend.

Che cosa è Ajax ?

- Ajax permette l'**aggiornamento dinamico** di una pagina web senza l'esplicito ricaricamento; mentre per cambiare il contenuto di una pagina web classica, si deve ricaricarla.

- Ajax è **asincrono,** ovvero i dati scambiati fra browser e server sono caricati in background senza interferire con il comportamento della pagina.

Che cosa è Ajax ?



Che cosa è Ajax ?

- Le **tecnologie base** di Ajax sono JavaScript, XML e XHTML; tuttavia, l'uso di JavaScript e del XML non sono obbligatori.

- Infatti, Ajax permette di scambiare dati con il server usando XML o altre tecnologie come ad esempio JSON (JavaScript Object Notation).

Che cosa è Ajax ?

- E’ basato su i seguenti **Standard Internet**:

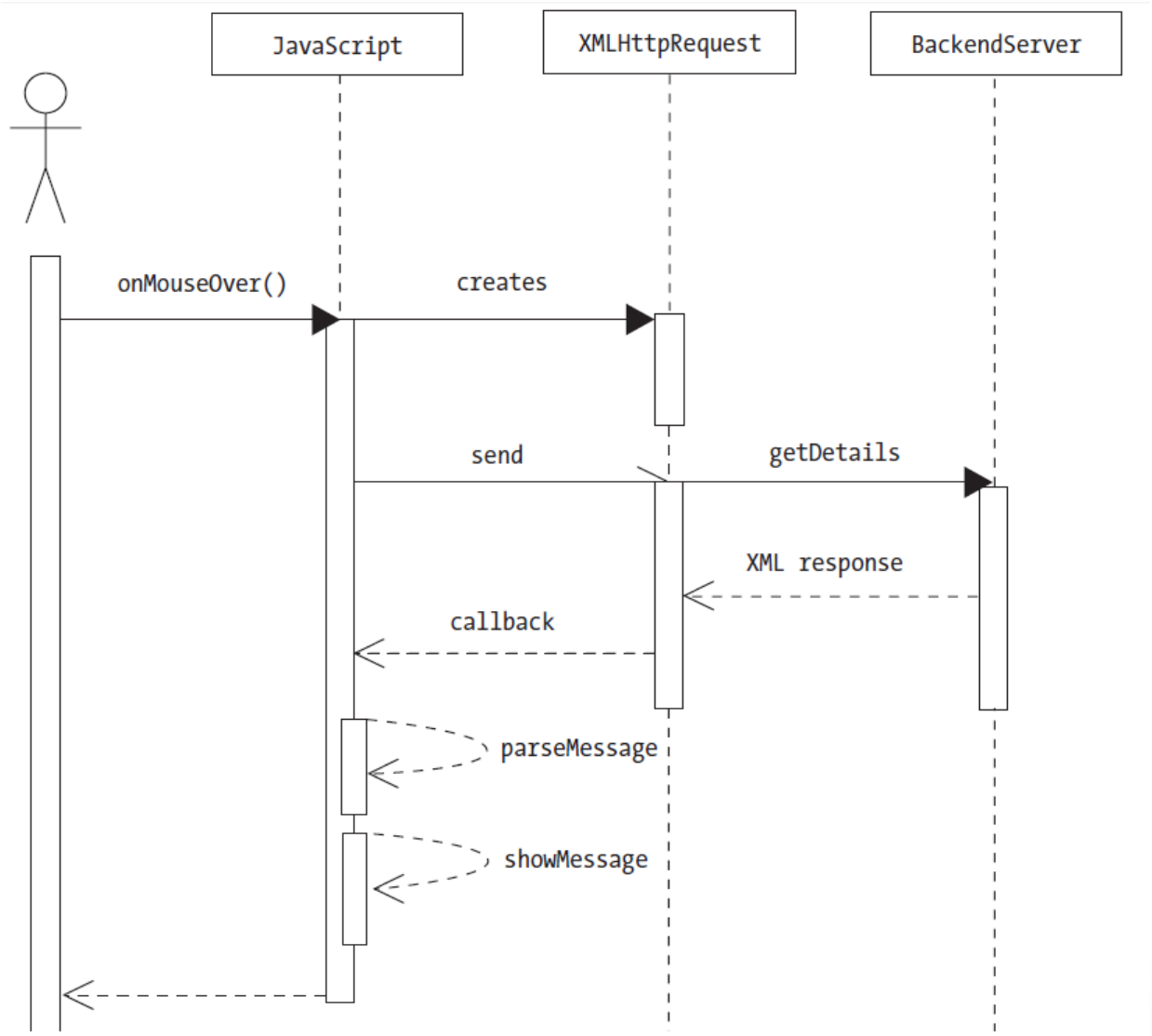
1. **JavaScript:** permette l'interazione con il browser e la risposta agli eventi;
2. **XHTML + CSS:** per la presentazione della pagina web;
3. **DOM:** per accedere e manipolare la struttura XHTML della pagina web;
4. **XML:** rappresenta il formato per scambiare i dati tra il server e il client;
5. **Oggetto XMLHttpRequest:** permette il recupero asincrono dei dati XML tra il server e il client.

- Le applicazioni web che usano Ajax richiedono che i browser implementino tali Standard.

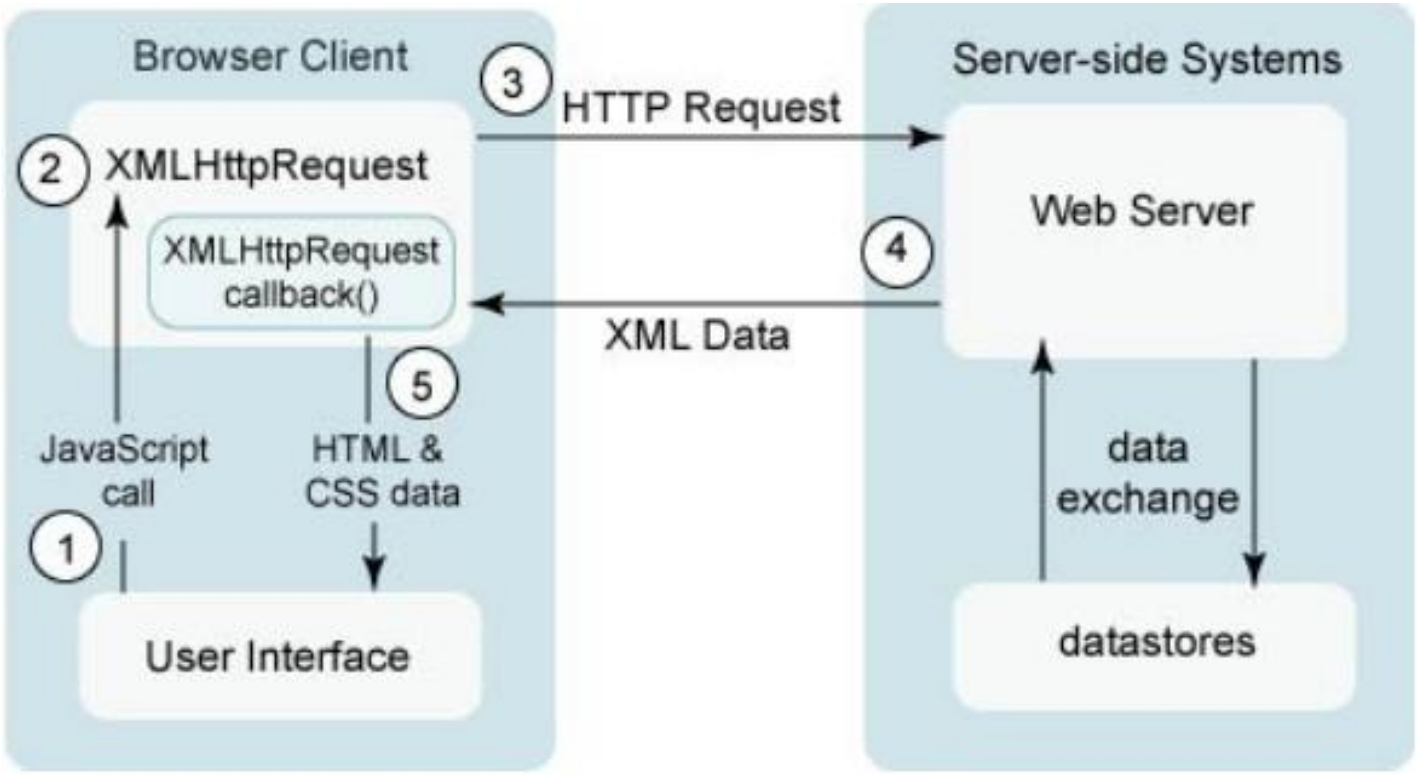
Come funziona Ajax ?

1. L'utente genera un evento, ad esempio attraverso l'inserimento di un testo in un controllo input di una pagina web, che l'applicazione gestisce attraverso JavaScript.
2. Viene poi creato, usando JavaScript, un oggetto XMLHttpRequest.
3. L'oggetto XMLHttpRequest interagisce attraverso una richiesta asincrona con il server di backend.
4. Un modulo software lato server (es. PHP) riceve ed elabora la risposta e poi risponde in XML.
5. L'oggetto XMLHttpRequest riceve ed elabora la risposta XML, e poi aggiorna di consequenza il DOM che rappresenta la pagina web.

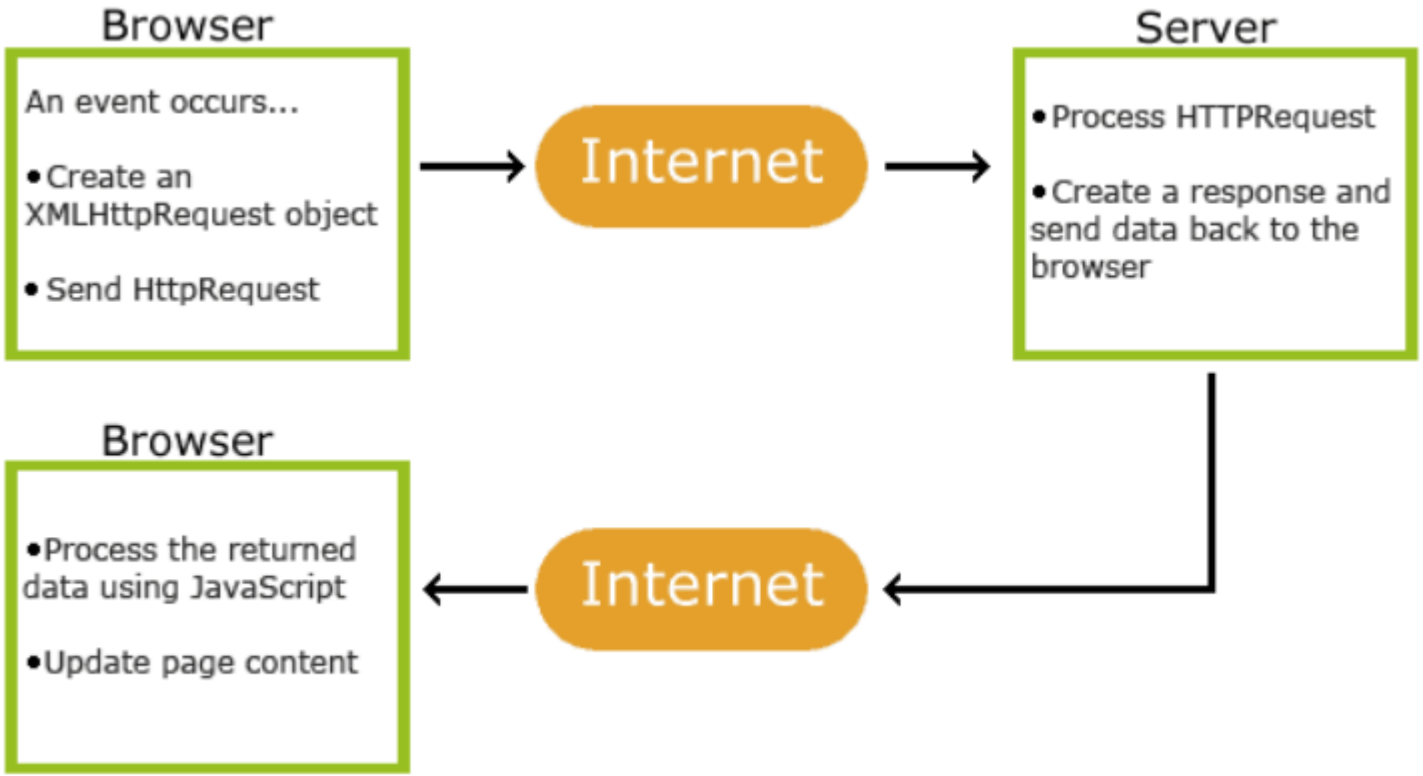
Come funziona Ajax ?



Come funziona Ajax ?



Come funziona Ajax ?



Dove usare Ajax ?

- **Form**: Aumenta sensibilmente le prestazioni di una form HTML.
- **Comunicazione degli utenti**: Utile nella progettazione di componenti software per la comunicazione, ad esempio chat, bottoni di voto, messaggi thread, rating, ecc.
- **News**: RSS feeds può essere gestito con tecnologia Ajax (es. Google News).
- **Manipolazione dei dati**: Un esempio è l'ordinamento o il filtraggio dei dati attraverso le colonne di una tabella. Un altro esempio è il completamento di un campo di una form HTML con i suggerimenti.

Dove usare Ajax ?

Osservazioni

- Ajax non è la soluzione ad ogni problema.
- La gestione di grandi quantità di dati con Ajax può portare a problemi di prestazioni e di altro tipo.
- Usare Ajax solo quando i tradizionali widget JavaScript non sono sufficienti e quando si deve fare la manipolazioni dati che comportano scambio con il server.
- Nel blog *Ajax Mistakes* è possibile trovare alcune linee guida riguardo l'uso di Ajax:

  http://alexbosworth.backpackit.com/pub/6768

Esempi di tecnologia Ajax

Alcuni esempi di applicazioni web che implementano Ajax sono:

Google

- Gmail e Mappe, Calendario, Home Page Personalizzate e Search Box Google.

Yahoo

- Home Page di Yahoo, un gran numero di personalizzazioni e di caratteristiche, come le anteprime veloci delle e-mail.

Altri Esempi

- Youtube, Facebook tabs, …

Pregi

- **Usabilità**
  - Interattività
- **Velocità**
  - Minore quantità di dati scambiati
  - Parte della computazione sul client
- **Portabilità**
  - Supportato dai maggiori browser
  - Indipendente dalla piattaforma
  - Non richiede plug-in

Difetti

- **Usabilità**
  - Non funzionano il pulsante “back” e i segnalibri
  - I motori di ricerca non indicizzano i contenuti dinamici
- **Accessibilità**
  - Non supportato dai browser non-visuali
  - Richiede meccanismi di accesso alternativi
- **Configurazione**
  - JavaScript abilitato
  - Oggetti ActiveX abilitati in Internet Explorer
- **Compatibilità**
  - Test sui diversi browser
  - Richiede funzionalità alternative per i browser che non supportano JavaScript

Compatibilità dei browser

- **Compatibili con Ajax:**
  - Internet Explorer dalla versione 5;
  - Mozilla Firefox dalla versione 7.1;
  - Konqueror dalla versione 3.2;
  - Safari dalla versione 1.2;
  - Opera dalla versione 8.0;
  - Chrome.

- **Non compatibili con Ajax:**
  - Versioni precedenti a quelle viste sopra;
  - Tutti i browser testuali;
  - Tutti i browser per disabili visivi (screen-reader, browser vocali, …);
  - Tutti i browser precedenti al 1997.

Oggetto XMLHttpRequest

- E’ di fondamentale importanza nell’uso di Ajax, tutti i browser moderni lo supportano come oggetto nativo (IE5 e IE6 usano un ActiveX).

- E’ usato per scambiare i dati in background con un server ovvero aggiornare parti di pagina senza ricaricarla completamente.

- Il W3C ha in corso uno studio, oggi in Working Draft, per renderlo Standard Internet:

  http://www.w3.org/TR/XMLHttpRequest/

Interfaccia Standard W3C

```
[NoInterfaceObject]
interface XMLHttpRequestEventTarget : EventTarget {// for future use};

[Constructor]
interface XMLHttpRequest : XMLHttpRequestEventTarget {
  // event handler attributes
  attribute Function onreadystatechange;
  // states
  const unsigned short UNSENT = 0;
  const unsigned short OPENED = 1;
  const unsigned short HEADERS_RECEIVED = 2;
  const unsigned short LOADING = 3;
  const unsigned short DONE = 4;
  readonly attribute unsigned short readyState;
  // request
  void open(DOMString method, DOMString url);
  void open(DOMString method, DOMString url, boolean async);
  void open(DOMString method, DOMString url, boolean async, DOMString? user);
  void open(DOMString method, DOMString url, boolean async, DOMString? user, DOMString? password);
  void setRequestHeader(DOMString header, DOMString value);
  void send();
  void send(Document data);
  void send([AllowAny] DOMString? data);
  void abort();
  // response
  readonly attribute unsigned short status;
  readonly attribute DOMString statusText;
  DOMString getResponseHeader(DOMString header);
  DOMString getAllResponseHeaders();
  readonly attribute DOMString responseText;
  readonly attribute Document responseXML;};
```

Creazione dell'Oggetto

- Tutti i browser moderni (IE7+, Firefox, Chrome, Safari e Opera) implementano l'interfaccia W3C con un oggetto nativo XMLHttpRequest.

- La sintassi per la creazione è:

```
variable = new XMLHttpRequest();
```

- Mentre le vecchie versioni di Internet Explorer (IE5 e IE6) usano un oggetto ActiveX:

```
variable = new
      ActiveXObject("Microsoft.XMLHTTP");
```

Creazione dell'Oggetto

Per gestire tutti i browser bisogna controllare se il browser supporta l'oggetto XMLHttpRequest :

```
var xmlhttp;

if (window.XMLHttpRequest)
  {// IE7+, Firefox, Chrome, Opera, Safari
   xmlhttp = new XMLHttpRequest();
  }
else
  {// IE6, IE5
   xmlhttp = new
      ActiveXObject("Microsoft.XMLHTTP");
  }
```

Invio della Richiesta

- Per inviare una richiesta al server bisogna usare i metodi `open()` e `send()` dell'oggetto XMLHttpRequest:

```
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

- `send(string)`: invia la richiesta al server
  - `string`: solo POST

- `open(method, url, async)`: specifica il tipo di richiesta

1. `method`: GET o POST
2. `url`: indirizzo server
3. `async`: true (asincrona) o false (sincrona)

1. Method: GET o POST

- GET è più semplice di POST, e può essere usato nella maggior parte dei casi.

Andrebbe usata la richiesta POST quando:

a) Non è possibile utilizzare un file memorizzato nella cache (bisogna aggiornare un file o un database sul server);
b) Bisogna inviare al server una grande quantità di dati (POST non ha limitazioni di dimensione);
c) Bisogna inviare l'input utente (POST è più robusto di GET).

Richieste GET

a. Una semplice richiesta GET:

```
xmlhttp.open("GET","demo_get.php",true);
xmlhttp.send();
```

b. Nel precedente esempio potremo ottenere un risultato presente nella cache, per ovviare si può aggiungere un ID univoco all'URL:

```
xmlhttp.open("GET",
             "demo_get.php?t=" + Math.random(),
             true);
xmlhttp.send();
```

Richieste GET

c. Se si vuole inviare informazioni con il metodo GET, bisogna aggiungere informazioni all'URL:

```
xmlhttp.open("GET",
        "demo_get2.php?fname=Henry&lname=Ford",
        true);
xmlhttp.send();
```

Richieste POST

a. Una semplice richiesta POST:

```
xmlhttp.open("POST","demo_post.php",true);
xmlhttp.send();
```

b. Per fare una POST dei dati, come una form HTML, si deve aggiunge con il metodo setRequestHeader() un HTTP header e specificare nel metodo send() il dato che vuole inviare:

```
xmlhttp.open("POST","ajax_test.php",true);
xmlhttp.setRequestHeader("Content-type",
      "application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

Richieste POST

- `setRequestHeader(header, value)`: aggiunge alla richiesta gli HTTP header.

  `header`: nome del header

  `value`: valore del header

2. Url: indirizzo file sul server

- Il parametro `url` specifica l'indirizzo di un file sul server:

```
xmlhttp.open("GET", "ajax_test.php", true);
```

- Il file può essere di tipo testuale (.txt e .xml), oppure un componente software lato server (.asp, .php, .jsp, ...) che elabora la richiesta prima di rispondere.

3. Async: true o false

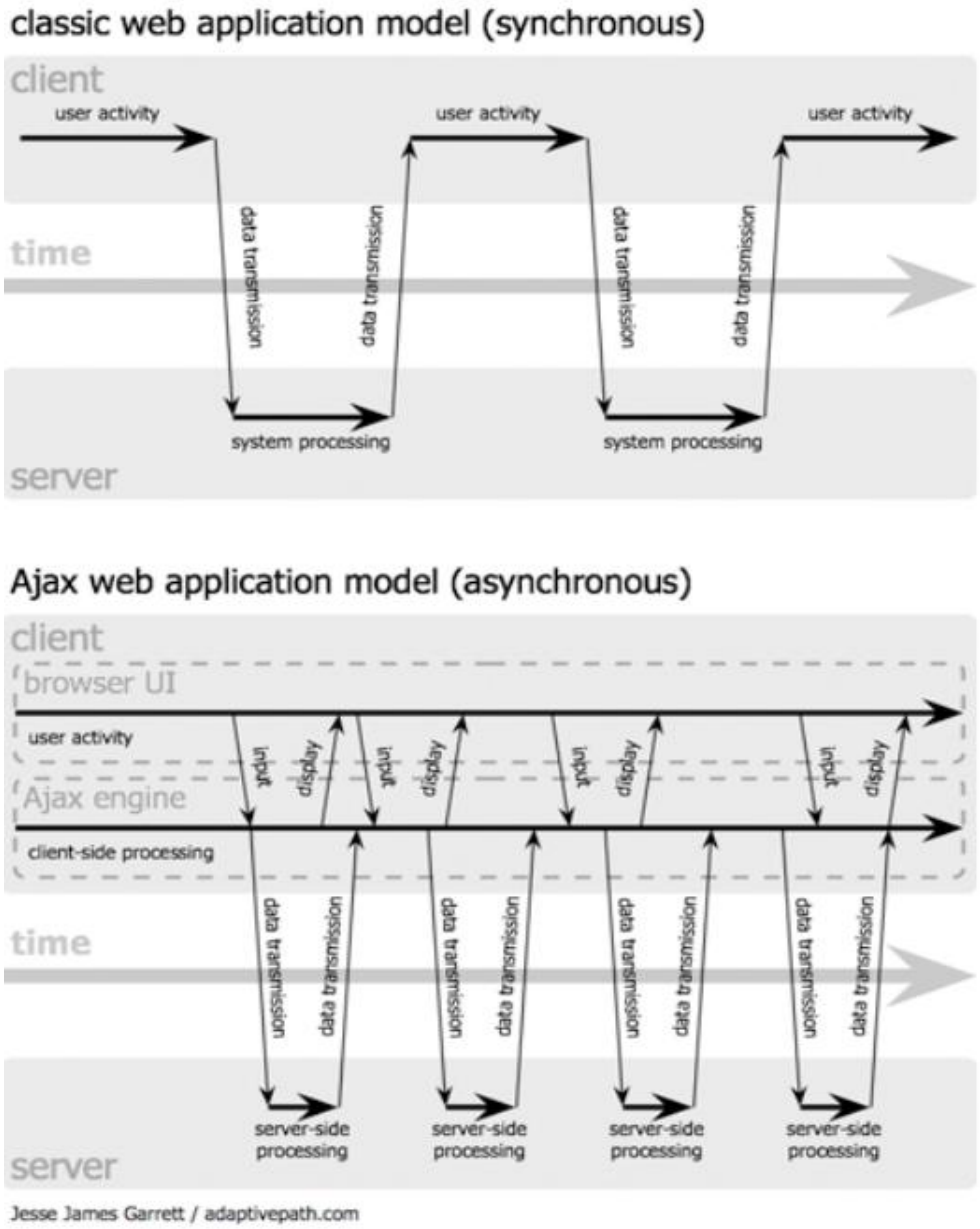
- Per parlare di Ajax il parametro `async` del metodo `open()` deve essere impostato a true:

```
xmlhttp.open("GET", "ajax_test.php", true);
```

- L'invio di richieste **asincrone** offre un miglioramento per le prestazioni delle applicazioni web. Gli script non devono aspettare la risposta del server, ma possono eseguire altro e poi proseguire quando essa è pronta.

- Le richieste **sincrone** possono causare rallentamenti o fermi della applicazione quando il server è sovraccarico e risponde con ritardo.

Richieste Sincrone e Asincrone



Richiesta Sincrona

- L'uso di `async = false` non è raccomandato, ma può andar bene per piccole quantità di dati.

- Quando si usa `async = false`, non si deve scrivere una funzione in `onreadystatechange`, ma basta inserire del codice dopo l'istruzione `send()`, ad esempio:

```
xmlhttp.open("GET", "ajax_info.txt", false);
xmlhttp.send();

document.getElementById("myDiv").innerHTML
                              = xmlhttp.responseText;
```

Richiesta Asincrona

- Quando si usa `async = true`, specificare nell'evento `onreadystatechange` la funzione da eseguire quando la risposta è pronta.

Esempio

```
xmlhttp.onreadystatechange = function()
  {
  if(xmlhttp.readyState == 4 && xmlhttp.status == 200)
    {
      document.getElementById("myDiv").innerHTML
                          = xmlhttp.responseText;
    }
  }

xmlhttp.open("GET", "ajax_info.txt", true);
xmlhttp.send();
```

Risposta del server

- Le proprietà per ottenere la risposta del server sono:
  1. `responseText;`
  2. `responseXML.`

Se la risposta non è XML, usare la proprietà `responseText` che restituisce una stringa, ad esempio:

```
document.getElementById("myDiv").innerHTML
                          = xmlhttp.responseText;
```

Risposta del server

- Se la risposta del server è XML e si vuole fare il parse, usare la proprietà `responseXML`, ad esempio:

```
xmlDoc = xmlhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");

for (i=0; i < x.length; i++)
{
  txt = txt + x[i].childNodes[0].nodeValue
            + "<br />";
}

document.getElementById("myDiv").innerHTML = txt;
```

Proprietà

- `readyState`: contiene lo stato di XMLHttpRequest.

  0: richiesta non inizializzata (UNSENT)

  1: connessione al server stabilita (OPENED)

  2: richiesta ricevuta (HEADERS_RECEIVED)

  3: elaborazione richiesta in corso (LOADING)

  4: richiesta completa e risposta pronta (DONE)

- `status`: contiene il codice di stato HTTP

  0: se lo stato è UNSENT o OPENED

  200: ok

  404: pagina non trovata

Eventi

- Quando è inviata una richiesta ad un server e si vuole fare una qualche azione basata sulla risposta, si utilizza l'evento `onreadystatechange` dell'oggetto XMLHttpRequest.

- Nell'evento `onreadystatechange`, grazie alla funzione (o al suo nome) chiamata automaticamente ad ogni cambiamento della proprietà `readyState`, è specificato cosa accadrà quando la risposta del server è pronta per essere processata.

Esempio

```
xmlhttp.onreadystatechange = function()
  {
  if (xmlhttp.readyState == 4
      &&
      xmlhttp.status == 200)
    {
      document.getElementById("myDiv").innerHTML
                    = xmlhttp.responseText;
    }
  }
```

L'evento `onreadystatechange` è scatenato quattro volte, una volta per ogni cambio di `readyState`.

JSON

- JavaScript Object Notation è un semplice formato per lo scambio dei dati, insieme all'XML, è il formato più comune per la trasmissione dei dati su canale HTTP da parte delle applicazioni Ajax. Come per l'XML, JSON contiene sia dato sia etichetta permettendo la creazione di strutture dati autodescrittive.

- Si basa su un sottoinsieme del linguaggio di programmazione JavaScript (Standard ECMA-262 terza edizione).

JSON

- E’ un formato testuale completamente indipendente dal linguaggio di programmazione, esistono librerie JSON per la maggior parte dei linguaggi.

- Definisce un piccolo insieme di regole di formattazione per la rappresentazione di strutture dati portabili, ed è basato sul concetto di oggetto JavaScript e di array associativo (array i cui indici sono parole).

Esempio di Oggetto JSON

```
{
  "Image": {
     "Width":  800,
     "Height": 600,
     "Title":  "View from 15th Floor",
     "Thumbnail": {
        "Url":    "http://www.example.com/image/481989943",
        "Height": 125,
        "Width":  "100"
     },
     "IDs": [116, 943, 234, 38793]
   }
}
```

Esempio di Array JSON

```
[
   {
     "precision": "zip",
     "Latitude":  37.7668,
     "Longitude": -122.3959,
     "Address":   "",
     "City":      "SAN FRANCISCO",
     "State":     "CA",
     "Zip":       "94107",
     "Country":   "US"
   },
   {
     "precision": "zip",
     "Latitude":  37.371991,
     "Longitude": -122.026020,
     "Address":   "",
     "City":      "SUNNYVALE",
     "State":     "CA",
     "Zip":       "94085",
     "Country":   "US"
   }
 ]
```

Framework Ajax

- Sono librerie JavaScript che semplificano la creazione di applicazioni che implementano Ajax, i cui scopi fondamentali sono:

1. **Astrazione**: gestiscono le differenze tra un browser e l'altro e forniscono un modello unico di programmazione compatibile con molti browser;

2. **Struttura**: forniscono un modello omogeneo di progetto dell'applicazione, indicando con esattezza dove e come inserire le caratteristiche specifiche dell'applicazione;

3. **Libreria di widget**: forniscono una ricca collezione di componenti di presentazione assemblabili per creare velocemente interfacce sofisticate e modulari.

Framework Ajax

Librerie JavaScript

- jQuery (http://jquery.com/)
- Prototype (http://www.prototypejs.org/)

Rich Internet Application Framework

- Dojo (http://www.dojotoolkit.org/)
- Ext (http://www.extjs.com/)
- GWT (http://code.google.com/webtoolkit/)
- YUI (http://developer.yahoo.com/yui/)

Risorse

- http://it.wikipedia.org/wiki/AJAX
- http://www.w3schools.com/ajax/default.asp
- http://www.ietf.org/rfc/rfc4627.txt?number=4627
- http://www.w3schools.com/json/default.asp
- http://alexbosworth.backpackit.com/pub/67688
- http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications
- https://developer.mozilla.org/en/AJAX/Getting_Started
- http://www.w3.org/TR/XMLHttpRequest/
- http://www.oracle.com/technetwork/java/javaee/tutorial-jsp-140089.html
- **Pro Apache Struts with Ajax** by John Carnell, Rob Harrop, Kunal Mittal (2006) (CHAPTER 12 - Struts and Ajax)