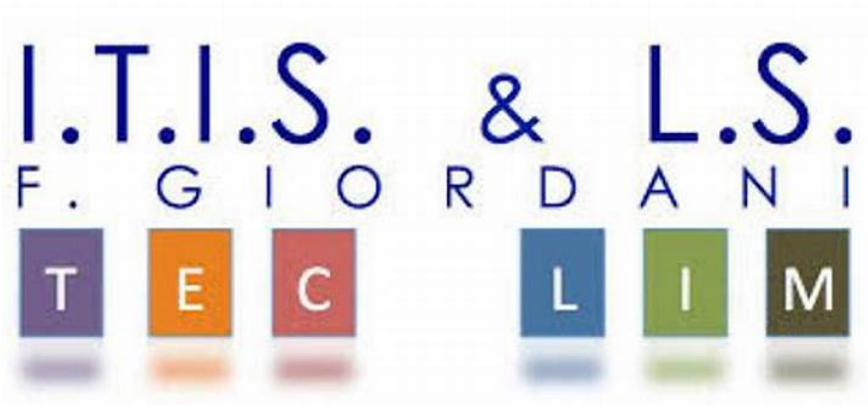

ITIS-LS "Francesco Giordani" Caserta

prof. Ennio Ranucci

a.s. 2019-2020

Semplici esercitazioni AJAX

Esercitazioni JAVASCRIPT AJAX XML JSON

Fonte dei primi 10 esercizi:
Spaghetti Ajax Ovvero Ajax demistificato di Salvatore antirez Sanfilippo, 9 Febbraio 2006

L'acronimo AJAX, che significa esattamente Asynchronous JavaScript And XML (JavaScript asincrono ed XML), è stato enunciato per la prima volta da Jesse Garrett, nel 18 Febbraio 2005, come titolo di un post all'interno del suo blog.
Non si tratta di una nuova tecnologia né di un'invenzione bensì di un concetto utilizzato per sviluppare applicativi avanzati e particolari quali Gmail, Google Maps o Google Suggest. Il concetto è in parte espresso nell'acronimo scelto, un utilizzo asincrono di Javascript che attraverso l'interfacciamento con XML, può permettere ad un client di richiamare informazioni lato server in modo veloce e trasparente, allargando gli orizzonti delle rich internet applications.

Se parliamo di AJAX, oggi, parliamo di un oggetto specifico: XMLHttpRequest. A seconda del browser usato prende nomi differenti o viene richiamato in maniera differente.
Nel caso di Internet Explorer, ad esempio, questo oggetto è restituito da un ActiveXObject mentre nei browsers alternativi più diffusi (Mozilla, Safari, FireFox, Netscape, Opera ed altri) XMLHttpRequest è supportato nativamente, cosa che dovrebbe accadere anche per IE dalla versione 7.
Questo oggetto permette di effettuare la richiesta di una risorsa (con HTTP) ad un server web in modo indipendente dal browser. Nella richiesta è possibile inviare informazioni, ove opportuno, sotto forma di variabili di tipo GET o di tipo POST in maniera simile all'invio dati di un form.
La richiesta è asincrona, il che significa che non bisogna necessariamente attendere che sia stata ultimata per effettuare altre operazioni, stravolgendo sotto diversi punti di vista il flusso dati tipico di una pagina web.
Generalmente infatti il flusso è racchiuso in due passaggi alla volta, richiesta dell'utente (link, form o refresh) e risposta da parte del server per poi passare, eventualmente, alla nuova richiesta da parte dell' utente.
Il terzo ed inevitabile incomodo di questo ciclo è l'attesa che trascorre tra una richiesta dell' utente e la risposta del server. Con l'aggiunta di AJAX si perde questa linearità e mentre l'utente è all'interno della stessa pagina le richieste sul server possono essere numerose e completamente indipendenti.
Nulla infatti vieta, teoricamente, di effettuare decine di richieste simultanee al server per operazioni differenti con o senza controllo da parte del navigatore.
Ajax è dal punto di vista tecnico, fondamentalmente l'unione di due cose.
La capacità di Javascript di aggiornare parte di una pagina HTML senza che questa venga caricata nuovamente.
La capacità di Javascript di fare richieste tramite il protocollo HTTP

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es0*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Browser - Javascript*
Obiettivo didattico:
innerHTML
Obiettivo del programma:
Sovrascrivere durante il caricamento della pagina un id del documento

```
<html>

<head>

<title>AJAX, le basi prima dell' utilizzo</title>

<script type="text/javascript">
```

```
function prendiElementoDaId(id_elemento) {
 var elemento;
 if(document.getElementById)
  elemento = document.getElementById(id_elemento);
 else
  elemento = document.all[id_elemento];
 return elemento;
};
</script>
</head>
<body onload="prendiElementoDaId('paragrafo').innerHTML = 'Buon giorno
<strong>JavaScript</strong>';">
<p id="paragrafo">
testo del paragrafo che verrà cambiato al caricamento del documento
</p>
</body>
</html>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es1*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Browser - Javascript*
Obiettivo didattico:
innerHTML
Obiettivo del programma:
Sovrascrivere durante il caricamento della pagina un id del documento

```
<html>
<head>
<script>
function script1() {
   var e = document.getElementById("pluto");
   e.innerHTML = "Benone!";
}
</script>
```

</head>

<body>

* *

Ciao come stai?

* *

<input type="button" value="Esegui esempio" onClick="script1()" />

</body>

</html>

Alla pressione del bottone il contenuto dell'elemento DIV che ha come id pluto viene modificato da "Come stai" a "Benone!". La funzione script1 chiama la funzione document.getElementById passando come argomento l'identificativo del DIV, e quello che viene ritornato è un riferimento all'oggetto che rappresenta il nostro DIV nella pagina HTML. L'oggetto in questione ha una proprietà chiamata innerHTML che può essere letta/scritta. Se viene letta restituisce il codice HTML presente dentro il DIV, se viene scrittta settandola a qualcosa di diverso come abbiamo fatto noi nell'ultima riga della funzione script1 il contenuto del DIV cambia e gli effetti si vedono immediatamente nella pagina.

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es2*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Browser - Javascript*
Obiettivo didattico:
innerHTML
Obiettivo del programma:
nascondere/mostrare un element del documento

```
<html>

<head>

<script>

function script2() {

   var e = document.getElementById("pippo");

   if (e.style.visibility == 'hidden') {

      e.style.visibility = 'visible';

      e.style.display = 'block';

   } else {

      e.style.visibility = 'hidden';

      e.style.display = 'none';

   }

}

</script>
</head>
<body>
<div id="pippo">
Mostra Nascondi
</div>
<input type="button" value="Clicca per far nascondere/mostrare" onClick="script2()" />
</body>
</html>
```

Prendiamo il riferimento dell'oggetto con document.getElementById come al solito e modifichiamo lo stile dell'oggetto a nostro piacimento tamite oggetto.style.nomeProprietà per fare quello che vogliamo. La prima viene settata a visible se si vuole che l'oggetto sia visibile, mentre il valore di hidden la fa scomparire. La seconda invece viene settata a none quando si vuole che l'oggetto non prenda più spazio nella pagina, altrimenti anche se invisibile occuperebbe lo spazio ugualmente. Si noti come la proprietà dispaly venga settata a block quando si vuole far ricomparire l'oggetto: questo valore è infatti il default per un DIV ma varia al variare dei diversi elementi HTML, ad esempio per altri elementi il valore di default è inline. Se volete saperne di più cercate un buon tutorial sui CSS.

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es3*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Browser - Javascript*
Obiettivo didattico:
interagire con il server
Obiettivo del programma:
richiedere l'esecuzione di un file .php

Qual'è la caratteristica principale di una applicazione in Ajax? Il fatto che la pagina si aggiorna senza la necessità di essere ricaricata, o al contrario che un evento generato dall'utente (come ad esempio la pressione di un tasto) causi una operazione sul server, senza la necessita di postare alcun FORM. Abbiamo visto come tramite Javascript sia possibile modificare il contenuto di una porzione della pagina, e di come gli elementi possano scomparire e ricomparire. Questo è sufficiente per creare pagine i cui contenuti si aggiornano in maniera dinamica.
Immaginate un programma di gestione del magazzino scritto in Ajax. Quando l'utente digita il codice di una particolare merce immediatamente appare nella pagina la quantità ancora disponibile in magazzino. Ovviamente non è pensabile che tutti i dati del magazino siano nella memoria di Javascript, dunque è necessario eseguire una richiesta al server "di nascosto", e appena si ottiene la risposta necessaria aggiornare la pagina.

Altro esempio, una web mail scritta in Ajax . Alla pressione del tasto Delete l'email viene cancellata. Eliminare la riga di quella email dalla lista delle email è semplice, abbiamo già visto come fare. Ma chi dice al server di cancellare l'email dal database se non è possibile spedire un FORM?
Quello che ci vuole per risolvere questi problemi è la capacità di fare richieste al server in maniera asincrona, ovvero in background, senza che l'utente si accorga di niente. In questo modo Javascript può contattare il server spedendo e ricevendo informazioni senza che la pagina venga ricaricata.

Per poter fare tale richiesta è necessario utilizzare un oggetto che si chiama XMLHttpRequest in tutti i browser moderni escluso Internet Explorer che necessita di un diverso oggetto. Ciò che serve per fare una richiesta HTTP in Ajax è ovviamente un pò di Javascript, e dall'altra parte (nel server) qualcuno che risponda alla richiesta, ovvero un piccolo script in PHP. Ovviamente l'applicazione server-side può essere scritta in qualunque linguaggio, tutti gli esempi di questo articolo sono scritti in PHP. Funzione che crea un oggetto XMLHttpRequest:

```
function CreateXmlHttpReq(handler) {
  var xmlhttp = null;
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = handler;
  return xmlhttp;
}
```

```
<html>
<head>
<script>
var myRequest = null;

function CreateXmlHttpReq(handler) {
  var xmlhttp = null;
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = handler;
  return xmlhttp;
}

function myHandler() {
    if (myRequest.readyState == 4 && myRequest.status == 200) {
        alert(myRequest.responseText);
    }
}

function esempio3() {
    myRequest = CreateXmlHttpReq(myHandler);
    myRequest.open("GET","localhost/ajax/primo.php");
    myRequest.send(null);
}
</script>
</head>
<body>
<input type="button" value="Clicca per lanciare la richiesta" onClick="esempio3()" /></body>
</html>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es4*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
XML Javascript Ajax
Obiettivo del programma:
Stampare a video il contenuto del file xml con javascript-Ajax

File leggiXml-htm

```
<html>

<head>

</head>

<body>

<div>

<b><span id="cognome0"></span> <span id="nome0"></span></b><br />

<span id="telefono0"></span><br />

<b><span id="cognome1"></span> <span id="nome1"></span></b><br />

<span id="telefono1"></span><br />

</div>

<script type="text/javascript">

if(window.XMLHttpRequest) {

//crea un parser in IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}else{

//crea un parser in IE5, IE6

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.open("GET","rubrica.xml",false);

xmlhttp.send("");

xmlDoc=xmlhttp.responseXML;

var i;

for (i = 0; i < xmlDoc.getElementsByTagName("cognome").length; i++) {

  document.getElementById("cognome"+i).innerHTML=
```

```
xmlDoc.getElementsByTagName("cognome")[i].childNodes[0].nodeValue;
document.getElementById("nome"+i).innerHTML=
xmlDoc.getElementsByTagName("nome")[i].childNodes[0].nodeValue;
document.getElementById("telefono"+i).innerHTML=
xmlDoc.getElementsByTagName("telefono")[i].childNodes[0].nodeValue;

}
</script>
</body>
</html>
```

File rubrica.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE rubrica SYSTEM "rubrica.dtd">

<rubrica>
<contatto lingua="it">
  <cognome>Rossi</cognome>
  <nome>Mario</nome>
  <telefono>123456</telefono>
</contatto>
<contattolingua="it">
  <cognome>Verdi</cognome>
  <nome>Antonella</nome>
  <telefono>223344</telefono>
</contatto>
</rubrica>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es5*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
JSON Javascript Ajax
Obiettivo del programma:
Inviare unstringa JSON ad un file .php

```
<!DOCTYPE html>
<html>
<body>
<h2>Convert a JavaScript object into a JSON string, and send it to the server.</h2>
<script>
var myObj = { name: "John", age: 31, city: "New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
</script>
</body>
</html>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es6*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
JSON Javascript Ajax
Obiettivo del programma:
Convertire una stringa JSON in un oggetto Javascript

```
<!DOCTYPE html>
<html>
<body>
<h2>Convert a string written in JSON format, into a JavaScript object.</h2>

<p id="demo"></p>

<script>

var myJSON = '{"name":"John", "age":31, "city":"New York"}';

var myObj = JSON.parse(myJSON);

document.getElementById("demo").innerHTML = myObj.name;

</script>
</body>
</html>
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es7*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Google Crome Versione 79.0.3945.79*
Obiettivo didattico:
JSON Javascript Ajax
Obiettivo del programma:
Scrivere e leggere una stringa JSON

```
<!DOCTYPE html>

<html>

<body>


<h2>Store and retrieve data from local storage.</h2>


<p id="demo"></p>


<script>

var myObj, myJSON, text, obj;


// Storing data:

myObj = { name: "John", age: 31, city: "New York" };

myJSON = JSON.stringify(myObj);

localStorage.setItem("testJSON", myJSON);


// Retrieving data:

text = localStorage.getItem("testJSON");

obj = JSON.parse(text);

document.getElementById("demo").innerHTML = obj.name;

</script>


</body>

</html>
```