Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Reggio Emilia

CORSO DI TECNOLOGIE E APPLICAZIONI WEB

JavaScript

Ing. Marco Mamei

Cos’e’ JavaScript?

E’ un linguaggio di scripting, per certi versi ispirato a Java (ma Java c’entra poi poco in verita’…) che permette di associare comportamenti reattivi alle pagine Web, rendendole cosi’ in grado di interagire più efficacemente con l’utente.

Scripting: un linguaggio si dice di scripting quando

- Non definisce precisa struttura di programma – sono pezzi di codice messi qua e la’…al contrario di C, ad esempio, nel cui c’e’ una precisa strutturazione in main e procedure, e di Java, nel quale c’e’ una precisa strutturazione in classi e oggetti.
- Non è compilato, ma è interpretato direttamente riga per riga, e riga per riga vengono eseguite le sue istruzioni. Al contrario ad esempio del C e di Java, dove prima di poter eseguire un programma bisogna compilarlo per tradurlo in un linguaggio macchina o equivalente.

Scopi:

- reagire agli eventi
- cambiare dinamicamente pezzi di HTML
- per validare pezzi di documento

Attenzione: esistono altri linguaggi di scripting che sono praticamente equivalenti, e molto molto simili, p.e., VBScript (Visual Basic, tipico della Microsoft),

Primo Esempio JS

Il seguente documento:

```
<html>
<head></head>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body></html>
```

si limita ad scrivere sul documento (istruzione document.write) una stringa. In questo caso non c'e' reazione agli eventi.

In generale, per inserire un JS in un documento HTML si usa il TAG SCRIPT.
Così:

```
<script type="text/javascript">
//codice Java Script
</script>
```

Oppure cosi':

```
<script language="javascript">
// codice JavaScript
</script>
```

Attenzione: per gestire il fatto che I browser vecchi non gestiscono JS, si usa mettere il codice JS tra commenti HTML (<!-- commento -->). I browser vecchi lo ignorano!

NOTA: document.write() è l'istruzione per scrivere del testo sulla pagina web.

Inserire JS: Alternativa

Non è sempre necessario inserire il codice come parte di un esplicito TAG SCRIPT. E' possibile inserire del codice JavaScript direttamente come attributo di un TAG.

Esempio:

```
<A HREF="nowhere" onMouseover="history.back()"> SE VAI QUI SOPRA TORNI INDIETRO</A>
```

Appena si va sopra il link viene eseguita la istruzione JS history.back(), che fa tornare alla pagina precedente.

Notiamo:

il concetto di ***reazione agli eventi***: **onMouseover** indica l'evento al quale la pagina web deve reagire (appena il mouse ci va sopra. Altri eventi: onClick, onMouseout, etc, etc…

Nota: Sperimentare le altre istruzioni per gestire la storia della navigazione → per tornare indietro, o avanti, a piacere, in reazione ad eventi!

`History.back(), history.forward(), history.go(-3), history.go(3)`

Bottoni in JS

E’ possible interagire con un documento inserrendo esplicitamente dei bottoni da “cliccare” e delle reazioni a questo bottone

I bottoni sono associati a tag di tipo INPUT hanno un nome interno, un valore (il testo sul bottone stesso). e reagiscono con codice JS all’evento **onClick**

```
<INPUT type="button" value="See Some Text"
name="button2"
onClick="window.status='You clicked the button!';
return true">
```

NOTA: window.status scrive nella barra in basso del browser

Altro esempio, per cambiare il colore di sfondo:

```
<INPUT type="button"
value="Sfondo Giallo"
name="button3"
onClick="document.bgColor='yellow'"> <br>
<INPUT type="button"
value="Sfondo Rosso!"
name="button4" onClick="document.bgColor='red'"> <br>
```

NOTA: esistono altri elementi di questo tipo <INPUT…> che si possono inserire in una pagina Web. Radio Button, Text Elements, etc. etc….

Servono per generare eventi, e per accettare input da inviare al server (**programmazione CGI**). **Per ora**, le usiamo **solo per gestire eventi** su una pagina.

Finestre di Alert

Si possono fare aprire delle finestre di dialogo (“alert” in reazione agli eventi)

```
<A HREF="nowhere" onMouseover="alert('TI HO DETTO DI
NON CLICCARE!!!')"> NON CLICCARE!</A>
```

la finestra si apre appena il mouse va sopra il link.

Oppure, mettendo una istruzione altert direttamente dentro al BODYhtml si fa in modo che la finestra sia aperta (che lo script sia eseguito) automaticamente mentre si va nella nuova pagina.

```
<HEAD>
<TITLE>Cool JavaScripts</TITLE>
</HEAD>
<BODY>
<SCRIPT language="JavaScript">
<!-- hide from old browsers
alert('Welcome to my Web Site!');
//-->
</SCRIPT>
</BODY>
```

Inoltre: Codice, Variabili e Funzioni

In JavaScript, oltre a poter usare delle funzioni predefinite che vanno ad agire sul documento e sul browser (come quelle viste: alert, document.wirte, history.back, etc.) è possibile:

- **definire delle proprie funzioni con del proprio codice**

Esempio:

```
function confronta (parameter 1,  parameter 2)
{
 JavaScript istruzioni
}
```

- **definire delle variabili, assegnar loro valori, e fare calcoli**

Esempio:

```
var contatore = 1;
contatore = altravariabile + 5;
```

- **usare del codice simile a quello di un linguaggio di programmazione (cicli for, istruzioni if, etc. etc.)**

Esempio:

```
for(i=0; i<1000; i++)
{
 if (contatore < 100) …
}
```

Variabili JavaScript

Come per quasi tutti i linguaggi di Script, le variabili non hanno un tipo esplicito.

```
var contatore = 1
```

definisce implicitamente una variabile numerica

```
var nome = “Franco”
```

definisce implicitamente una variabile stringa.

Altri tipi implicitamente definiti: float, booleani.
Il tipo cambia dinamicamente, a seconda del valore che si assegna a una variabilie…

Le variabili sono **visibili in tutto il documento**, dal punto in cui sono dichiarate in poi. Però, è possibile definire variabili locali in una funzioni.

Si possono definire anche senza scrivere var (in questo caso sono comunque variabili globali):

```
contatore;
nome = “Franco
```

In verità, non esiste una distinzione netta tra assegnamento e definizione: se a una variabile si assegna un valore, viene automaticamente creata. Tipico dei linguaggi di script.

Operare sulle variabili:
sono definiti tutti gli operatori di Java per la variabili numeriche, ed come in Java si possono concatenare con il segno + le variabili stringa

```
contatore+=5;
contatore++;
st1=”ciao” + nome + “amico”
```

Le Funzioni JavaScript

Le funzioni JavaScript sono per la loro definizione molto simili ai metodi Java.

```
function result(a,b){
c=a+b
return c
}
```

Tipicamente si dichiarano nell'head e s'invocano nel body o a seguito di eventi.

Le funzioni al loro interno possono usare tutte le variabili già definite, tutte le istruzioni che vogliono, possono avere un valore di ritorno, e possono definire nuove variabili globali.

Problema: è chiaro che se tutte le variabili sono sempre e solo globali, si possono avere dei forti conflitti nella composizione di diverse funzioni (p.e., due diverse funzioni che usano entrambe una variabile di nome i per due scopi diversi…si può avere confuzione o errori).

Soluzione: una funzione può anche definire delle variabili locali. In questo caso queste variabili saranno definite all'interno delle funzione stessa e facendole precedere da var

Nota: i parametri passati alle funzioni JavaScript, così come i parametri in Java e C, sono passati per copia, e quindi sono a tutti gli effetti delle variabili locali.

Dove Inserire gli Script

Tipicamente:

nell'<HEAD>
si inseriscono tutte le funzioni e le variabili globali.
All'atto del caricamento di una pagina, vengono caricate tutte le funzioni e inizializzate le variabili

Nel <BODY>,
all'interno dei TAG SCRIPT si inseriscono quelle istruzioni che devono essere eseguite automaticamente al momento del caricamento della pagina (p.e., il document.wirte che avevamo visto e l'alert dell'esempio precedente…)

Nei TAG
All'interno dei tag si inseriscono come attributi gli eventi che interessa gestire (p.e., onClick), e poi il valore di questi attributi, tra virgolette, identifica il codice da eseguire in reazione all'evento (p.e., alert('ciao')).
Tipicamente: il codice dentro gli attributi va a richiamare l'esecuzione di funzioni definite nel'HEAD

Su un file esterno .js
E' possibile scrivere dei file con degli script, e poi fare caricare questi file all'interno del nostro documento HTML
`<script src="xxx.js"></script>`
metterli dove si vuole che farli eseguire nell'head o nel body)

Esempio

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

function action1() {
 alert('ciao');
}

function action2(color) {
 action1();
 document.body.bgColor = color;
}
</script>
</head>

<body id="body" bgcolor="#ffffff">
<p id="first_para" onClick="action1();">
Questo è un primo esempio di paragrafo interattivo
</p>

<table border="1">
 <tr>
  <td bgcolor="#ff0000" onClick="action2('#ff0000');"> Rosso </td>
  <td bgcolor="#ffff00" onClick="action2('#ffff00');"> Giallo </td>
 </tr>
 <tr>
  <td bgcolor="#0000ff" onClick="action2('#0000ff');"> Blu </td>
  <td bgcolor="#00ff00" onClick="action2('#00ff00');"> Verde </td>
 </tr>
</table>

</body>
```

Come Accedere agli Elementi nella Pagina HTML?

Uno degli aspetti più complicati di Javascript è l'accesso agli elementi della pagina HTML che esegue lo script. Mentre è relativamente facile far comparire qualcosa di esterno alla pagina (es. alert box), è più complicato accedere agli elementi della pagina stessa (es. **document.body.bgColor='red'**)

Qual è il meccanismo (la sintassi) che sta alla base della parte in grassetto?

Questo meccanismo si chiama **DOM (Document Object Model)** è consiste nel rappresentare un documento HTML nella sua naturale struttura ad albero. È infatti possibile accedere agli elementi del DOM saltando verso elementi specifici o navigando nell'albero.

```
var body = document.getElementById('body');
body.bgColor = ‘red’;
```

```
var body = document.getElementsByTagName('body')[0];
body.bgColor = ‘red’;
```

Attenzione a tutti i dettagli!

È poi possibile navigare nell'albero:

```
body.parentNode;
body.childNodes[0];
```

Separare HTML e SCRIPT

Anche se richiede di scrivere più codice, definire gli script in un file esterno è la soluzione migliore dal punto di vista ingegneristico.

1. Permette di distinguere chiaramente cos'è il **contenuto** del documento e cos'è la **logica applicativa** del documento.
2. Permette di **riutilizzare** gli script tra diversi documenti.

Quindi:

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

function action1() {
 alert('ciao');
}

function action2(color) {
 var body = document.getElementById('body');
 //var body = document.getElementsByTagName('body')[0];
 body.bgColor = color;

 //document.body.bgColor = color;
}
</script>
</head>

<body id="body" bgcolor="#ffffff">
<p id="first_para" onClick="action1();">
Questo è un primo esempio di paragrafo interattivo
</p>

<table border="1">
 <tr>
  <td bgcolor="#ff0000" onClick="action2('#ff0000');"> Rosso </td>
  <td bgcolor="#ffff00" onClick="action2('#ffff00');"> Giallo </td>
 </tr>
 <tr>
  <td bgcolor="#0000ff" onClick="action2('#0000ff');"> Blu </td>
  <td bgcolor="#00ff00" onClick="action2('#00ff00');"> Verde </td>
 </tr>
</table>

</body>
```

Separare HTML e Script

File esempio3.html

```
<html><head>
<script type="text/javascript" src="esempio3.js"></script>
</head>
<body id="body" bgcolor="#ffffff">
<p id="first_para">
Questo è un primo esempio di paragrafo interattivo
</p>
<table border="1">
 <tr>
  <td bgcolor="#ff0000" id="red"> Rosso </td>
  <td bgcolor="#ffff00" id="yellow"> Giallo </td>
 </tr>
 <tr>
  <td bgcolor="#0000ff" id="blue"> Blu </td>
  <td bgcolor="#00ff00" id="green"> Verde </td>
 </tr>
</table></body></html>
```

File: Esempio3.js

```
function action1(e) {
 alert('ciao');
}

function action2(e) {
 var body = document.getElementById('body');
 var cell = window.event.srcElement;
 body.bgColor = cell.bgColor;
}
function addEvent(elm, evType, fn) {
 // cross-browser event handling
 if(elm.addEventListener) {
  elm.addEventListener(evType, fn, false);
  return true;
 }
 else if(elm.attachEvent) {
  var r = elm.attachEvent('on'+evType,fn);
  return r;
 }
 else {
  elm['on'+evType] = fn
 }
}
function addListeners(e) {
 var firstPara = document.getElementById('first_para');
 addEvent(firstPara,'click',action1);

 var tds = document.getElementsByTagName('td');
 for(var i=0; i<tds.length; i++)
  addEvent(tds[i],'click',action2);
}

addEvent(window,'load',addListeners);
```

Un Esmpio più Complesso - HTML

```
<html>

<head>
<script type="text/javascript" src="esempio4.js"></script>
</head>

<body id="body" bgcolor="#ffffff">
<h1>Lista di link che si possono ordinare</h1>
<ol id="list">
 <li><a href="http://www.google.com">Google</a></li>
 <li><a href="http://www.yahoo.com">Yahoo</a></li>
 <li><a href="http://www.repubblica.it">La Repubblica</a></li>
 <li><a href="http://www.meteo.it">Meteo</a></li>
</ol>

<hr/>
Vuoi aggiungere un nuovo link?<br/>

<table>
<tr>
<td>Nome:</td>
<td><input type = "text" id="newname" name= "newname"/></td>
</tr>
<tr>
<td>Indirizzo:</td>
<td><input type = "text" id="newhref" name= "newhref"/></td>
</tr>
</table>

<input type ="button" id="add" name="add" value="add"/>

</body>

</html>
```

Un Esmpio più Complesso - JS

```
function order(e) {
 var list = document.getElementById('list');
 var element = window.event.srcElement;
 list.appendChild(element.parentNode);
 window.event.returnValue = false;
 window.open(element);
 //window.location = element.href;
}

function add(e) {
 var newli = document.createElement('li');
 var newlink = document.createElement('a');
 newlink.href = document.getElementById('newhref').value;
 var linktxt =
document.createTextNode(document.getElementById('newname').value);

 newli.appendChild(newlink);
 newlink.appendChild(linktxt);
 var list = document.getElementById('list');
 list.appendChild(newli);
 addEvent(newli,'click',order);
}


function addListeners(e) {
 var lis = document.getElementsByTagName('li');
 for(var i=0; i<lis.length; i++)
  addEvent(lis[i],'click',order);

 var addbutton = document.getElementById('add');
 addEvent(addbutton,'click',add);
}



function addEvent(elm, evType, fn) {
 // cross-browser event handling
 if(elm.addEventListener) {
  elm.addEventListener(evType, fn, false);
  return true;
 }
 else if(elm.attachEvent) {
  var r = elm.attachEvent('on'+evType,fn);
  return r;
 }
 else {
  elm['on'+evType] = fn
 }
}

addEvent(window,'load',addListeners);
```

Esempio: personalizzare una pagina(*)

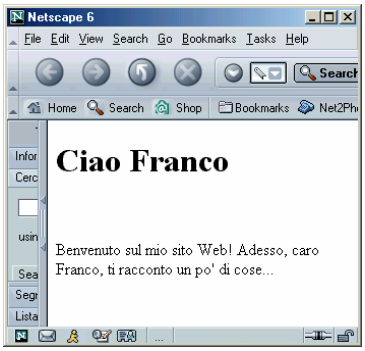
```
<html>
<head>
<script>
var nome;
// dichiarazione della variabile nome
</script>

</head>
<body>
<script>
nome = prompt('dimmi il tuo nome', '');
if (nome == ‘ ’ || nome == null) nome = ‘imbecille’;
// prompt e’ un alert di input
</script>

<h1> Ciao <script> document.write(nome) </script></h1>
<br>

Benvenuto sul mio sito Web! Adesso, caro <script>
document.write(nome) </script> ti racconto un po’ di
cose…

</body>
</html>
```



(*) Nota che in molti degli esercizi che seguono, per rendere più diretta l’esposizione, si è scelto di non dividere il codice HTML da quello JavaScript come sarebbe stato giusto fare. È un esercizio utile quello di modificare gli esempi separando le due parti.

Esempio: proteggere una pagina con password (molto simile all'esempio di prima)

```
<html>
<head>
<script>
var password = "simsalabim";
var tentative;
// dichiarazione della variabile passwd
</script>

</head>
<body>
<script>
tentativo = prompt('dimmi la password', '');
if (tentativo != password)
  {alert('Mi spiace password errata!');
   history.back();
}
</script>

<h1> Ciao Persona Autorizzata </h1>
<br>

Benvenuto sul mio sito segreto!
</body>
</html>
```

Domanda: questo è un modo efficace di proteggere il contenuto del sito?

No! Esaminando l'HTML è facile risalire alla password. La soluzione finale è fare queste operazioni lato server

Intanto si può migliorare la situazione provando con funzioni difficilmente invertibili…..

Esempio: Guidare la navigazione

```
<HTML>
<HEAD>
<SCRIPT language="JavaScript">
<!--
function go_there()
{var where_to = confirm("Sei sicuro, somaro?");
 if (where_to== true)
 {
   window.location="http://www.jenniferlopez.com";
 }
 else
 {
  window.location="http://www.ingre.unimo.it";
  }
}
//--> </SCRIPT>
</HEAD>
<BODY>
<H1> Ciao Pinocchio </h!>
<br>
Hai due scelte: andare al <A
HREF=http://www.ingre.unimo.it>
sito di Ingre Unimo </A> oppure al sito di
<A HREF= "nowhere" onClick="go_there();return TRUE">
Jennifer Lopez </A>.
```

NOTA: l'uso delle finestre di confirm, che mi danno un valore true o false.
NOTA: il return false fa in modo che l'HREF sia ingnorato! In pratica, quando si ha un return false, si ignorano gli altri possibile effetti standard delle azioni

Alternativa:

```
Hai due scelte: andare al <A
HREF=http://www.ingre.unimo.it> sito di Ingre Unimo
</A> oppure al sito di <A
HREF="javascript:go_there()"> Jennifer Lopez </A>.
```

NOTA: l'uso di Javascript al posto dell'HREF.

Altri Elementi Interattivi: Il Text

Devono essere dentro al tag FORM – tutto ciò che è dentro ai form servirà per mandare input alle CGI – tramite uno speciale elemento di tipo SUBMIT. Questo vale anche per i bottoni, come già visto.

Dentro al un tag FORM, ci possono essere una serie di elementi interattivi.

Quelli che servono per dare input hanno in generale la seguente forma:

<INPUT TYPE=”tipo di elemento” NAME=”nome dell’elemento” >

più eventuali altri attributi tipici dell’elemento.

Esempio: il TEXT FIELD

<INPUT TYPE = “TEXT” NAME=”a piacere” SIZE = “numero di caratteri”>

Permette di inserire del testo dentro alla casella. Tipicamente viene gestito dall’evento **onChange** (che si scatena quanto cambia il testo) oppure dall’evento **onClick** scatenato da un botton associato ad esso.

Altri attributi: SELECTED=”valore” dice che cosa mettere inizialmente prima che l’utente faccia il suo input

Esempio: Controllare l’input di un utente: versione 1

```
<HTML>
<HEAD>

<SCRIPT>
anno_attuale = 2002;

function calcola_eta(anno)
{ var eta = anno_attuale - anno;
  return eta;
}

</SCRIPT>
</HEAD>
<BODY>

<H1> Scrivi il tuo anno di nascita </h1>

<FORM name="eta">
<INPUT TYPE = "TEXT" SIZE = "5" name="ANNONASCITA"
onChange="tua_eta = calcola_eta(this.value); alert('La
tua eta' + tua_eta)">

</FORM>
</BODY>
</HTML>
```

Esempio: Controllare l'input di un utente: versione 2

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE = "JavaScript1.2">
anno_attuale=2002;

function verifica_eta( annonascita, eta)
{ var anni_sommati = 0;

  var stringa_espressione = annonascita + "+" + eta;
  anni_sommati = eval(stringa_espressione);
// per forzare a calcolare una stringa!!!

  if ( anni_sommati != anno_attuale)
    {
     return false;
    }
return true;

}
</SCRIPT>
</HEAD>
<BODY>

<H1> Sinistra anno di nascita e a destra eta
poi spingi il bottone per la verifica</h1>



<FORM name="eta">
<INPUT TYPE = "TEXT" SIZE = "5" name=”annonato”>
<INPUT TYPE = "TEXT" SIZE = "5" name=”miaeta”>
<INPUT TYPE = "BUTTON" VALUE = "VERIFICA"

onClick = "anno=document.eta.annonato.value;
          tua_eta=document.eta.miaeta.value;
//ATTENZIONE QUESTE VARIABILI SONO STRINGHE NON LE
//POTREMO USARE PER FARE I CALCOLI!!! USARE EVAL!
          risultato = verifica_eta(anno, tua_eta);
          alert('You said the ' + risultato)">
</FORM>
</BODY>
</HTML>
```

Altro Esempio: redirezione parametrizzata

```
<HTML>
<HEAD>
<SCRIPT language="JavaScript">
<!--
function go_there(link, i)
{
 if(link!="http://www.ingre.unimo.it")
 {
  where_to = confirm("Sei sicuro, somaro?");
  if (where_to != true)
  {
  document.links[i].href="http://www.ingre.unimo.it";
  }
 }
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Ciao Pinocchio </h!>
<br>
<br>
Da qui ai due scelte: andare al
<A HREF="http://www.ingre.unimo.it"
onClick="go_there(document.links[0].href, 0)">
sito di Ingre Unimo </A>

oppure al sito di
<A         HREF=         "http://www.jenniferlopez.com"
onClick="go_there(document.links[1].href, 1)">
Jennifer Lopez </A>.
Qui invece potresti andare al sito di

<A         HREF=         "http://www.georgeclooney.com"
onClick="go_there(document.links[2].href, 2)">
George Clooney </A>.
```

QUI AI LINK NON ABBIAMO DATO NOME; MA SONO COMUNQUE ACCESSIBILI IN UN ARRAY!!

Altri Elementi Interattivi: Le Liste

Le Liste di Scelta

```
<FORM>
<SELECT name="sport" onChange="istruzioni JavaScript
da Eseguire">
<OPTION SELECTED>--Choose a Sport
<OPTION>Football
<OPTION>Basketball
…
</SELECT>
</FORM>
```

Il codice da eseguire poi potrà fare riferimento all'oggetto opzione selezionato per vedere cosa è stato selezionato.
NOTA: L'evento tipico per le liste è **onChange**
NOTA: che questi elementi hanno sempre un nome (p.e., la lista sopra ha nome "sport", quella sotto ha nome "journal"). Nel caso in un documento ci siano più form, è bene dare il nome anche a form oltre che all'elemento dentro form. (vedi sotto, FORM ha nome "scegli")

Esempio:

```
<FORM NAME = "scegli">

<SELECT name="journal"
onChange="window.location=document.scegli.journal.opti
ons[document.scegli.journal.selectedIndex].value">

<OPTION SELECTED>Scegli un Giornale
<OPTION value="www.repubblica.it">Repubblica
<OPTION value = "www.ilrestodelcarlino.it">Carlino
</SELECT>
</FORM>
```

NOTARE: Il modo con cui ci si riferiscea al valore dell'elemento selezionato.
document.scegli.journal.options[document.scegli.journal.selectedIndex].value

Ancora Altri Elementi Interattivi

TEXTAREA
Simili ai text, ma si possono inserire testi lunghi su diverse righe…
<TEXTAREA NAME=”PIPPO” ROWS=25 COLUMNS=80 …

RADIO BUTTON
Una serie di “bottoncini” dei quail si può selezionarne uno in esclusiva agli altri. Si raggruppano tutti quelli che hanno lo stesso NAME. Uno di questi può essere selzionato (CHECKED) all’inizio.

```
<h1> Dai un voto alla pagina</h1>

<INPUT type="RADIO" name="VOTO" value="1" checked>1
<INPUT type="RADIO" name="VOTO" value="2">2
<INPUT type="RADIO" name="VOTO" value="3">3
```

Reagiscono tipicamente all’**onChange** o all’**onClick**

CHECK BOX
Tipo I radio button, ma non sono in esclusiva – sono una serie di pulsantini indipendenti l’uno dall’altro.

RESET - un button speciale
Annulla tutte le azioni e gli input effettuati nel FORM contenente. Riporta il FORM allo stato iniziale.

SUBMIT – un button speciale
Servirà per le CGI – scatena l’invio di tutti i valori (**VALUE**) verso il server, da usarsi come input
<INPUT TYPE=”SUBMIT” value=”Invia i dati”>

Altri Esempi: Cambiare una Immagine

```
<HTML>
<HEAD>
</HEAD>

<BODY>

<img name = "fotofamiglia" src="TuttieTre.JPG"
width="300" height="400">


<input type="button" value="Tutti e Tre"
   onClick="document.fotofamiglia.src='TuttieTre.JPG';
      document.fotofamiglia.width=300;
      document.fotofamiglia.height=400">

<input type="button" value="Anna e Veronica"
   onClick="document.fotofamiglia.src='AnnaeVero.JPG';
   document.fotofamiglia.width=400;
   document.fotofamiglia.height=300">

<input type="button" value="Anna e Veronica"
   onClick="document.fotofamiglia.src='PrimoPianoVeron
ica.JPG';
   document.fotofamiglia.width=400;
   document.fotofamiglia.height=300">

</BODY>
</HTML>
```

NOTA: è anche possibile fare il cosiddetto “preload” delle immagini. Si caricano tutte le immagini che serviranno durante il caricamento della pagina. Così, quando serve visualizzarle, esse sono più veloci.

Si fa semplicemente (nell’header o nel body):
immagine = new Image();
immagine.src = “PrimoPianoVeronica.JPG”;

Altri Esempi: Animare una Immagine

NOTA: per cambiare una immagine senza l'intervento dell'utente (creando delle specie di animazioni) si può usare una funzione predefinita set interval setInterval che permette di eseguire ripetutamente una funzione JavaScript a intervalli determinati di tempo.

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript1.2">

var quale_delle_due=1;

function cambia_immagine()
{if (quale_delle_due == 1)
 { document.fotofamiglia.src ="AnnaeVero.jpg";
   quale_delle_due = 2;
 }
 else
 {document.fotofamiglia.src ="PrimoPianoVeronica.jpg";
   quale_delle_due=1;
  }
}
</SCRIPT>
</HEAD>
<BODY>
<img   name   =   "fotofamiglia"   src="TuttieTre.JPG"
width="400" height="300"
  onLoad="setInterval('cambia_immagine()',1000)">
</BODY>
</HTML>
```

Ulteriori Caratteristiche di JS (1)

Oggetti

JavaScript, ispirandosi a Java, permette di definire oggetti e di usare i loro attributi secondo una notazione ad oggetti. Le classi non esistono, ma si può creare un oggetto direttamente attraverso una funzione "costruttore"

Esempio:

```
function Animale(tipo, verso)
{this.tipo = tipo;
 this.verso = verso;
```

e poi posso creare oggetti animali:

```
gatto = new Animale("felino", "miao");
```

e usarli

```
if(gatto.tipo == "felino")…
```

Classi di Oggetti Predefinite

Esistono in JS classi di oggetti predefiniti che possono già usare (p.e., la classe data e la classe Array)

```
data = new Date();
data.anno = 2002;

vettore = new Array(10);
for (var i = 0; i < 10; i++)
       vettore[i] = 23;
```

Le stringhe

Come in Java, le variabili di tipo stringa sono considerate oggetti della classe stringa, che mette a disposizione molte funzioni utili per usarle

```
Var nome = FRanCO;
nomeminuscolo = nome.toLowerCase();
```

Ulteriori Caratteristiche di JS (2)

Oggetti Predefiniti
Oltre alle classi predefinite, esistono oggetti predefiniti di cui si possono variare gli attributi. Molti li abbiamo gà visti.

La finestra del browser
`window.status('stringa nello status')`
`window.open('www.dismi.unimo.it')`
apre una nuova finestra di browser, e si può anche dire con che dimensioni e magari senza strumenti e senza barre di scorrimento, etc….tipico di finestre pubblicitarie che si aprono quando si va su un sito…
`window.location('new URL')` ridirige la navigazione

Il browser
`Navigator.appName` → dice se Explorer o Netscape

Ogni pezzo di documento
`Document.nomepezzo.nomepezzo.interno.nomeattributo`
Come già detto, ogni pezzo di documento si può considerare come un oggetto con i suoi attributi, e si può specificare con la notazione gerarchica giàò vista.

I Frames
Una pagina divisa in frame da' luogo ad un array di frame. Da un frame all'altro, si interviene non più parlando di "**document**" (visto che sono multipli) ma di frames. Esempio, un frame può riferirsi a un frame ad esso affiancato in questo modo:
`Parent.frame[i]`

Ulteriori Caratteristiche di JS (3)

I Cookies

E' un oggetto particolare che può essere creato in un documento. La caratteristica principale e' che viene memorizzato stabilmente nel file system del client. Permette a una pagina Web di ritrovare informazioni sul sito del client.
Un cookie ha attributi che possono essere definiti a piacere. Sono espressi in termini di coppie NOMEATTRIBUTO=VALORE.

Esempio uso di cookie:

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript1.2">
var nome;
function estraicookie()
{ if(!document.cookie)
   { newname = prompt("dimmi il tuo nome");
    document.cookie = "PIPPO=" + newname + ";";
   }

start = document.cookie.indexOf('=');
nome = document.cookie.substring(start+1);
return nome;
}
</SCRIPT>
</HEAD>

<BODY>
<h1> Ciao <SCRIPT> document.write(estraicookie()) </SCRIPT> </h1>
</BODY>
</HTML>
```

È importante notare che grazie ai cookie il parametro immesso resta valido anche una volta che si ricarichi la pagina.

Altri Linguaggi di Scripting

Piccole differenze con JavaScript, e a volte obiettivi leggermente diversi. Non ci sono problemi a impararli…

DHTML (dynamic HTML)

Non è un liguaggio di script. Si parla di DHTML se di arricchisce HTML con la capacità di fare piccoli script direttamente sui tag, da usarsi anche insieme a CSS, per cambiare anche gli stili CSS. Esempio:

```
<html><body>
<h1 onmouseover="style.color='red'"
onmouseout="style.color='black'">
Mouse over this text</h1>
</body></html>
```

Anche gli ultimi esempi visti si possono definire DHTML…

VBScripts (Visual Basic Scripts)

Risposta Microsoft a JavaScripts. Non dà niente di più e niente di meno rispetto a JavaScripts. Cambia solo un poco la sintassi delle istruzioni. Abbastanza usato a livello industriale.

WMLScript

Specializzato per applicazioni WAP, non ha un grande futuro…(verrà sostituito da appositi stylesheet XSL). Sintassi comunque molto simile a quella di JS.

Un caso a parte è: **Macromedia Flash**

Specializzato per rendere dinamiche le pagine HTML tramite animazioni interattive. Non difficile da imparare, ma è ovvio che ha scopi e caratteristiche completamente diverse da quelle di JS. Noi vedremo come fare animazioni con le applet (è più difficile, ma più generale!)

In Conclusione

Javascript è una tecnologia molto potente per offrire funzionalità applicative **lato client**.

Occorre prestare molta attenzione a creare codice javascript che sia pulito e riutilizzabile.

Sarà opportuno riesaminare le funzionalità offerte da javascript una volta che avremo imparato a gestire le funzionalità applicative lato server.

Le applicazioni che congiungono in modo ottimale funzionalità lato client e funzionalità lato server sono ad oggi lo stato dell'arte.

Alcuni Link Interessanti

http://www.w3schools.com

http://www.pageresource.com/jscript/