Introduzione a JavaScript   
(Corso di Javascript)

http://plent.altervista.org/inizio.htm

**Premessa**

Nella realizzazione dei siti web, tutti vorrebbero applicare dei codici in Javascript o meglio, trovare delle applicazioni già pronte adatte al caso. Purtroppo sono pochi i siti web che, oltre a fare teoria, presentano degli esempi. Ci siamo sforzati a venire incontro a molti utenti fornendo molta ma molta pratica.

**Che cos'é JavaScript?**

JavaScript é un nuovo linguaggio di scripting. Gli scripts scritti in JavaScript possono essere incapsulati nelle pagine HTML. Con JavaScript é possibile rispondere alle azioni dell'utente ( ad esempio un inserimento in un form). Questo accade senza una transazione sulla rete. In genere quando un utente inserisce un dato in una form é necessario trasmetterlo al server che lo elabora e restituisce una risposta. L'inserimento é controllato dal client e puó essere trasmesso dopo che sia stato controllato. In pratica é il browser che si occupa dell'elaborazione dei JavaScripts e non piú il Server.

Attenzione! JavaScript non é Java. Java é un linguaggio di programmazione molto piú complesso di JavaScript.

**Come eseguire gli scripts ?**

Si possono eseguire con Netscape Navigator 2.0 e versioni successive e con Internet Esplorer 3.0 e con tutti gli altri browser che li supportano.

Per capire come si possono implementare all'interno delle pagine HTML ecco un esempio:

Questo script scrive una riga di testo in un documento HTML.

<html>

<head>

Il mio primo JavaScript!

</head>

<body>

<br>

Questo è un normale documento in HTML.

<br>

<script language="JavaScript">

<!--

document.write("Questo è un JavaScript!")

</script>

<br>

Back in HTML again.

</body>

</html>

... produce:

*Questo è un normale documento in HTML.   
Questo è un JavaScript!   
Back in HTML again.*

Il codice document.write sta ad indicare: scrivi nel documento. Se vogliamo produrre un testo in grassetto, di un certo colore, valgono le stesse considerazioni espresse nel codice HTML. Per es.:  
document.write("<p align='center'><b><font size='6' color='#ff0000'>Questo è un Javascript</font></b></p>")

Vediamo ora di capire ogni punto di questo primo script.  
  
  
**1. <SCRIPT LANGUAGE="JavaScript">**   
  
Ecco il primo nuovo tag, SCRIPT seguito dal parametro LANGUAGE.  
Come siamo abituati con i tag in coppia di HTML (se lo apri chiudilo...), tutto quanto racchiuso tra il tag di apertura (in questo caso SCRIPT e il suo relativo tag di chiusura /SCRIPT) è considerato JavaScript.   
Il parametro LANGUAGE="JavaScript" è necessario per la differenziazione da altri linguaggi di scripting.   
  
  
**2. <!-- Nascondi lo script dai browser non compatibili**  
  
Abbiamo aggiunto questa riga, che è un commento di HTML, per racchiudere tutto lo script in modo non sia mostrato a schermo da un'eventuale browser non compatibile con gli script.   
  
  
**3. {** Con il carattere parentesi graffe aperte '{' racchiudiamo una serie di istruzioni che potranno essere eseguite solo se la condizione impostata nella riga << risulterà vera, quindi unicamente se la variabile 'entra' risultasse 'true'..   
  
  
  
**4. document.writeln("<FONT FACE='Helvetica,Arial' SIZE='5' COLOR='red'>Ciao "  
+ nome + "**</FONT>Grazie per aver inserito il tuo nome.");   
Con la funzione document.writeln possiamo inserire, nella pagina HTML, testo 'on the fly' (al volo...). Notiamo che il testo è racchiuso da parentesi tonde e a sua volta dalle doppie virgolette. All'interno notiamo però l'uso delle virgolette semplici per racchiudere elementi singoli. Possiamo comunque concatenare più parti di testo. Nell'esempio, alla prima parte di testo (racchiusa dalle doppie virgolette) abbiamo aggiunto (con il segno '+') la variabile 'nome' e in seguito (sempre con il segno '+') altro testo. Nel testo racchiuso tra virgolette doppie possiamo inoltre inserire anche tutti i tag di HTML come ad esempio, in questo caso, per contenere gli elementi, utilizziamo le virgolette semplici.

**Oggetto:** button

L'oggetto ***button***, istanza di un oggetto *form*, serve per controllare i pulsanti inseriti in un form HTML.

**Proprietà:**

 ***enabled*** - valore booleano che indica se il pulsante è abilitato.

 ***form*** - riferimento all'oggetto form che contiene il bottone.

 ***name*** - stringa contenente il nome assegnato al bottone.

 ***type*** - stringa contenente l'attributo TYPE della marcatura INPUT

 ***value*** - stringa indicante il valore assegnato al bottone.  
  
**Metodi:**

 ***click()*** - simula l'azione di un clic sul bottone.

 ***focus()*** - attiva il bottone.  
  
**Eventi supportati:**

 ***onMouseDown*** - Specifica il codice da eseguire quando l'utente abbassa il bottone

 ***onMouseUp*** - Specifica il codice da eseguire quando un utente rilascia il pulsante del mouse dopo aver abbassato il bottone.

 ***onClick*** - Specifica il codice da eseguire quando un utente clicca.

 ***onFocus*** - Specifica il codice da eseguire quando quando il bottone riceve l'attivazione.

**Esempio:**   
Il seguente codice genera un bottone e mostra un alert quando esso viene cliccato:

*<FORM NAME="modulo">  
<INPUT TYPE="button" NAME="pulsante" ONCLICK="mostra()" VALUE="clicca!">  
</FORM>  
  
<SCRIPT LANGUAGE="JavaScript">  
function mostra() {  
  alert('hai cliccato');  
}  
</SCRIPT>*

Inizio modulo

Fine modulo

**La proprietà onMouseOver**

Potete vedere la funzione di Javascript di cui ci occuperemo ora muovendo il puntatore del mouse su questo [link](http://plent.altervista.org/stupid.htm). Ora osservate la statusbar in fondo alla finestra del browser. E' possibile combinare ciò con altre funzioni Javascript. Se vi posizionate sopra questo [link](http://plent.altervista.org/stupid.htm) comparirà una finestra. Ora vi mostrerò il sorgente di questi due effetti:

<a href="stupid.htm" onMouseOver="window.status='Questo è uno stupido link ...';

return true">

L'unica cosa che dovete fare è aggiungere la proprietà onMouseOver al tag <a>; 'window.status' vi permetterà di scrivere sulla statusbar del browser. Come potete vedere dovete racchiudere il testo tra singoli apici ('); non potete usare i doppi apici ("), altrimenti Javascript non sarà in grado di identificare la stringa che volete visualizzare nella statusbar. Dopo la stringa occorrerà scrivere *;return true*.   
Il secondo esempio usa Javascript invocando la funzione alert. Ecco il codice:

<html>

<head>

<script language="JavaScript">

<!-- nasconde lo script ai vecchi browser

function ciao() {

alert("Ciao!");

}

// -->

</script>

</head>

<body>

<a href="" onMouseOver="ciao()">link</a>

</body>

</html>

E' molto semplice: viene usata la proprietà onMouseOver e la funzione *ciao()* viene invocata quando si verifica il relativo evento.

**Functions**

Le Functions sono dichiarate tra i tags <head> dell' HTML. Le Functions sono chiamate dagli eventi generati dall'utente. É per questo che é ragionevole inserirle tra i tags <head> visto che sono caricati dal browser prima che l'utente possa fare qualcosa che provochi la chiamata ad una funzione. Gli Scripts possono essere inseriti tra i tags di commento <!-- --> in questo modo i vecchi browsers non visualizzeranno il sorgente.

<html>

<head>

<script language="JavaScript">

function pushbutton() {

alert("Hello!");

}

</script>

</head>

<body>

<form>

<input type="button" name="Button1" value="Push me" onclick="pushbutton()">

</form>

</body>

</html>

Se vuoi testare lo Script premi il bottone ...

Inizio modulo

Fine modulo

Questo scripts crea un bottone che se premuto apre un finestra messaggio 'Hello!'

La function é caricata e trattenuta in memoria. Il bottone é stato creato con i tags (HTML) standard dei form. É stato inserito un nuovo elemento nel tag <input> , 'onclick'. Questo indica al browser quale funzione deve essere richiamata quando il bottone viene premuto. La function 'pushbutton()' é dichiarata nell' header. Quando il bottone viene premuto questa function viene eseguita. C'é un altro nuovo elemento in questo script- il metodo 'alert'. Anche questo metodo é dichiarato in JavaScript - occorre solamente richiamarlo. Esistono diversi metodi che si possono richiamare, noi ne mostreremo alcuni, é presente una lista completa nel sito Netscape.

**Come leggere gli input inseriti dall'utente nelle forms...**

<html>

<head>

<script language="JavaScript">

<!-- hide script from old browsers

function getname(str) {

alert("Hi, "+ str+"!");

}

// end hiding contents -->

</script>

</head>

<body>

Please enter your name:

<form>

<input type="text" name="name" onBlur="getname(this.value)" value="">

</form>

</body>

</html>

Ora si puó testare lo script .. *Scrivi il tuo nome :*

Inizio modulo

**

Fine modulo

I commenti cominciano subito dopo il tag <script> e terminano subito prima del tag </script>. I vecchi browser ignorano i due tags <script> e </script> perché non li conoscono e naturalmente il commento in quanto tale. In questo documento quanto l'utente inserisce il nome nel form, il metodo 'onBlur' inserito nel tag <input> indica al client quale funzione richiamare. La function 'getname(str)' sará chiamata quando si 'lascia' il campo della form o quando si preme enter dopo aver inserito qualche cosa. La function riceve la stringa inserita tramite il comando 'getname(this.value)'. 'This.value' indica il valore inserito nel campo della form.

**Last changed - date**

Implementiamo ora la function data nel nostro script. É possibile fare visualizzare dal client le ultime modifiche al documento. In questo modo non sará necessario cambiare la data nel sorgente del documento HTML ogni volta che lo si modifica.

<html>

<body>

This is a simple HTML- page.

<br>

Last changes:

<script language="JavaScript">

<!-- hide script from old browsers

document.write(document.lastModified)

// end hiding contents -->

</script>

</body>

</html>

Con la versione beta2 di Netscape 2.0 occorreva scrivere *lastmodified*. Ora con le nuove versioni il metodo é cambiato in *lastModified*. É bene quindi fare attenzione a come si scrive il codice JavaScript che é ***case sensitive***.

Introduzione a JavaScript

(prima parte)

**Frames e JavaScript.**

Inizio modulo

Fine modulo

Ecco l' HTML :

<HTML>

<HEAD>

<title>Frames</title>

</HEAD>

<FRAMESET ROWS="50%,50%">

<FRAME SRC="frtest1.htm" name="fr1">

<FRAME SRC="frtest2.htm" name="fr2">

</FRAMESET>

</HTML>

Per prima cosa occorre dire al browser quanti frames si vuole visualizzare. Questo é definito nel tag <frameset...> . Le istruzioni da utilizzare sono *cols* e *rows*per creare rispettivamente colonne e righe. Le istruzioni <frameset...> possono essere annidate:

<FRAMESET COLS="50%,50%">

<FRAMESET ROWS="50%,50%">

<FRAME SRC="cell.html">

<FRAME SRC="cell.html">

</FRAMESET>

<FRAMESET ROWS="33%,33%,33%">

<FRAME SRC="cell.html">

<FRAME SRC="cell.html">

<FRAME SRC="cell.html">

</FRAMESET>

</FRAMESET>

Inizio modulo

Fine modulo

Si sono create due colonne e la seconda é stata divisa in 3 parti uguali.  
Quando si utilizza JavaScript é importate utilizzare i nomi dei frames.

Un altro esempio:

Inizio modulo

Fine modulo

Questo mostra come é possibile scrivere alcuni frasi di testo in un determinato frame mediante la pressione di un pulsante.

Ecco il sorgente:

***frames.htm***

<HTML>

<HEAD>

<title>Frames</title>

</HEAD>

<FRAMESET ROWS="50%,50%">

<FRAME SRC="frame1.htm" name="fr1" noresize>

<FRAME SRC="frame2.htm" name="fr2">

</FRAMESET>

</HTML>

***frame1.htm***

<HTML>

<HEAD>

<script language="JavaScript">

<!-- Hiding

function hi() {

document.write("Hi!<br>");

}

function yo() {

document.write("Yo!<br>");

}

function bla() {

document.write("bla bla bla<br>");

}

// -->

</script>

</HEAD>

<BODY>

This is our first frame!

</BODY>

</HTML>

***frame2.htm***

<HTML>

<body>

This is our second frame!

<p>

<FORM NAME="buttonbar">

<INPUT TYPE="button" VALUE="Hi" onClick="parent.fr1.hi()">

<INPUT TYPE="button" VALUE="Yo" onClick="parent.fr1.yo()">

<INPUT TYPE="button" VALUE="Bla" onCLick="parent.fr1.bla()">

</FORM>

</BODY>

</HTML>

Il FORM carica il primo file (frames.htm), questo crea i frames in codice HTML e carica il file frame1.htm nel primo frame ('fr1') e frame2.htm nel secondo frame ('fr2'). Il primo frame1.htm contiene alcune functions di JavaScript, ma non sono chiamate all'interno del file frame1.htm ma da frame2.htm.! I bottoni presenti in frame2.htm lanciano le funzioni definite in frame1.htm; la proprietá onClick ora include : parent.fr1... che cosa significa ? frame1.htm e frame2.htm sono caricati dalle istruzioni del file frames.htm, ***frames.htm*** é chiamato*'parent'* (dgli altri due). Coseguentemente i nuovi 3 frames sono i *'child- frames'* (di frames.htm).

frames.htm parent

/ \

/ \

/ \

fr1(frame1.htm) fr2(frame2.htm) children

Si puó estendere questo concetto creando un *'grandchildren'* in questo modo:

frames.htm parent

/ \

/ \

/ \

fr1(frame1.htm) fr2(frame2.htm) children

/ \

/ \

/ \

gchild1 gchild2 'grandchildren'

Se si vuole far riferimento a funzioni definite in*'parent'*, occorre indicare *parent* prima del nome della function. Se invece si vogliono chiamare functions definite in frame1.htm devo specificare anche *fr1*. (*fr1* é il nome che si era assegnato, all'interno di frames.htm, ad un frame specifico ). Quindi, come si puó notare anche dal grafico, non ci sono connessioni dirette tra i files frame1.htm e frame2.htm, quando chiamo (da frame2.htm) functions definite in frame1.htm, devo per forza indicarle passando attraverso il *'frame parent'* :

***<... onClick="parent.fr1.hi()">  
<... onClick="parent.fr1.yo()">  
<... onClick="parent.fr1.bla()">***

É possibile cancellare i frames una volta creati ?

Certamente:

basta aggiungere *TARGET="\_top"* al tag ***<a href...>*** .

Es.:

<a href="goaway.html" TARGET="\_top">Clicca qui per uscire dai frames</a>

Introduzione a JavaScript  
(seconda parte)

**La statusbar**.  
  
Per statusbar si intende una scritta che compare o scorre nella barra di stato nella parte bassa del Browser. Come é possibile indirizzare del testo nella statusbar ? Lo script che segue mostra come fare:

Inizio modulo

Fine modulo

Ecco il sorgente dello script:

<html>

<head>

<script language="JavaScript">

<!-- Hide

function statbar(txt) {

window.status = txt;

}

// -->

</script>

</head>

<body>

<form>

<input type="button" name="look" value="Write!" onclick="statbar('Hi! This is the statusbar!');">

<input type="button" name="erase" value="Erase!" onclick="statbar('');">

</form>

</body>

</html>

Sono stati creati due pulsanti, i quali eseguono delle chiamate alla function ***statbar(txt)***. Txt nelle parentesi indica la variabile che viene passata alla function. Si puó notare all'interno del tag <form> come eseguire le chiamate alla function ***statbar(txt)***, in realtá non viene passata un variabile vera e propia, ma una stringa di testo, che nel secondo caso (bottone Erase) é vuota e serve per cancellare il contenuto della statusbar.

**Timeout function**

Abbiamo giá visto la proprietá onMouseOver nella seconda parte del tutorial:

*<a href="tpage.htm" onMouseOver="window.status='ecco un'altro link...'; return true">*

É anche possibile impostare un timer in modo da cancellare la scritta sulla statusbar dopo un determinato tempo:

Prova a spostare il puntatore del mouse [qui ma non cliccare !](http://www.unimo.it/corsi/html/jsintro/dontclck.htm)

Ecco il sorgente:

<html>

<head>

<script language="JavaScript">

<!-- Hide

function moveover(txt) {

window.status = txt;

setTimeout("erase()",1000);

}

function erase() {

window.status="";

}

// -->

</script>

</head>

<body>

<a href="dontclck.htm" onMouseOver="moveover('Disappearing!');return true;">

link</a>

</body>

</html>

In questo secondo script é stata aggiunta la function erase(), nell'HTML é stato creato un link utilizzando la propietá onMouseOver. La function moveover(txt) viene chiamata passando la stringa 'Disappearing!'. Alla variabile **'window.status'** viene assegnato il contenuto di txt. La stessa cosa accadeva per la function statbar(txt). La function setTimeout(...) invece setta il tempo durante il quale la stringa deve stare visualizzata sulla statusbar e indica che cosa deve accadere allo scadere di questo tempo. Nell'esempio la function erase() viene chiamata dopo 1000 millisecondi (1 secondo).

**Testo scorrevole**

Inizio modulo

Fine modulo

Ecco lo script:

<html>

<head>

<script language="JavaScript">

<!-- Hide

var scrtxt="Questo é il messaggio che scorre sulla statusbar";

var lentxt=scrtxt.length;

var width=100;

var pos=1-width;

function scroll() {

pos++;

var scroller="";

if (pos==lentxt) {

pos=1-width;

}

if (pos<0) {

for (var i=1; i<=Math.abs(pos); i++)

{ scroller=scroller+" ";}

scroller=scroller+scrtxt.substring(0,width-i+1); }

else { scroller=scroller+scrtxt.substring(pos,width+pos); }

window.status=scroller; setTimeout("scroll()",150); } //-->

</script>

</head>

<body onLoad="scroll();return true;">

Inserire qui la vostra pagina

</body>

</html>

Questo scripts utilizza functions o parti di esse che abbiamo gia visto. Le operazioni che vengono fatte all'inizio servono per calcolare le posizioni di partenza, ma non sono importanti per capire come lo script lavora.

Introduzione a JavaScript

(terza parte)

**Arrays**

In ogni linguaggio di programmazione, gli array svolgono un compito molto importante. Se si esaminano i documenti messi a disposizione da Netscape si ci accorge che gli arrays non sono presi in considerazione. Il fatto é che gli arrays non esistono in JavaScript! Quindi...  
Se ad esempio si vuole assegnare il numero 17 ai primi 10 elementi dell'array *MyArray[],*per non ripetere 10 volte la stessa istruzione di assegnamento, occorrerebbe scrivere un ciclo simile a questo:

*for (var i=1;i<11;i++) MyArray[i]="17"*

In javaScript per inizializzare un array si deve scrivere una function come questa:

*function initArray() {  
this.length = initArray.arguments.length  
for (var i = 0; i < this.length; i++) this[i+1]="initArray.arguments[i]" }*

Non é molto importante capire come lavora questa function, ma é importante capire come utilizzarla. Se si vuole creare un array di tre elementi bisogna scrivere:

*var MyArray= new initArray(17,18,19)*.

17,18,19 vengono assegnati agli elementi 1,2,3. Se invece si vuole memorizzare delle stringhe negli elementi dell'array la sintassi é la seguente:

*var heyho= new initArray("This","is","cool")*.

É anche possibile fare degli assegnamenti **'misti'**:

*var Mixedup= new initArray(17,"yo",103)*.

Inizializzamo ora l'array 'Mixedup' ed osserviamo l'output di questo script:

Elemento N° 1: Rossi Mario  
Elemento N° 2: Bianchi Roberto  
Elemento N° 3: Narcisio Franco

Ecco lo script per l'output:

<script language="JavaScript">

<!-- Hide

var Mixedup= new initArray("<font color='blue'>Rossi Mario</font>","<font color='blue'>Bianchi Roberto</font>","<font color='blue'>Narcisio Franco</font>");

document.write("<font color='red'>Elemento N° 1:</font> "+Mixedup[1]+"<br>");

document.write("<font color='red'>Elemento N° 2:</font> "+Mixedup[2]+"<br>");

document.write("<font color='red'>Elemento N° 3:</font> "+Mixedup[3]+"<br>");

// -->

</script>

**Netscape Navigator 3.0**

Netscape 3.0 implementa new array:

Il codice che segue crea un array di 25 elementi e assegna valori ai primi tre:

musicTypes = new Array(25)

musicTypes[0] = "R&B"

musicTypes[1] = "Blues"

musicTypes[2] = "Jazz"

**Come cancellare un documento**

Se si vuole cancellare una finestra o un frame la documentazione di Netscape suggerisce di utilizzare la function **'document.clear()'**. Ma se si prova ad implementare questa function non accade nulla !

A tale proposito Gordon McComb propone queste righe di codice:

document.close();  
document.open();  
document.write("<P>");

Non é importante scrivere *document.write("<P>");* É importante inviare qualsiasi cosa alla finestra.

**Bottoni di navigazione**

Se inserisco un link in un documento che mi porta alla pagina precedente non é la stessa cosa del tasto back del browser. Si pensi ad esempio ad una pagina HTML che viene puntata da piú punti di pagine diverse, in questo caso dovrei inserire tanti link di back quante sono le pagine chiamanti. Con JavaScript é possibile accedere alla*'history list'* del browser

Prova a [cliccare qui](http://plent.altervista.org/backtest.htm).

Ecco il sorgente:

<html>

<body>

<FORM NAME="buttonbar">

<INPUT TYPE="button" VALUE="Back" onClick="history.back()">

<INPUT TYPE="button" VALUE="JS- Home" onClick="location='script.htm'">

<INPUT TYPE="button" VALUE="Next" onCLick="history.forward()">

</FORM>

</body>

</html>

É anche possibile scrivere *history.go(-1)* e *history.go(1)*.

Introduzione a JavaScript  
Parte 4

**Altre funzioni relative alla data**

Voglio mostrarvi ora un altro esempio di uso dei metodi relativi all'ora e alla data. Abbiamo visto come funziona la proprietà *lastModified*. Ora visualizzeremo l'ora nel nostro documento. Questo metodo utilizza l'ora e la data della vostra macchina, quindi se avete la data della macchina impostata al 17/5/1983 otterrete ovviamente la data sbagliata. Non si tratta quindi di un'ora e una data fornita da Internet (o qualcosa del genere).   
  
L'ora attuale è: 20:16  
La data di oggi è: 19/1/115

Ecco il codice:

<script language="JavaScript">

<!--

oggi = new Date()

document.write("L'ora attuale è: ",oggi.getHours(),":",oggi.getMinutes())

document.write("  
La data di oggi è: ", oggi.getDate(),"/",oggi.getMonth() + 1,"/",oggi.getYear());

// -->

</script>

Come prima cosa creiamo una variabile contenente la data di oggi. Questo è fatto con *oggi=new Date()*. Se non specifichiamo una certa ora e una data, il browser utilizza l'ora locale e l'assegna alla variabile *oggi*. Notate che non occorre dichiarare la variabile *oggi*. Questa è una differenza nei confronti di Java e della maggior parte degli altri linguaggi di programmazione in cui è necessario specificare il tipo di una variabile prima del suo uso.   
Abbiamo quindi creato un oggetto contenente l'ora e la data locali. Ora possiamo scrivere il suo contenuto nel documento. Bisogna scrivere *oggi* prima di ciascun metodo get, altrimenti il browser non sa a quale oggetto ci si riferisce. Il metodo getMonth() restituisce un numero compreso tra 0 e 11 (0 sta per gennaio e 11 per dicembre), quindi dobbiamo aggiungere 1 al numero restituito per ottenere il mese corretto.   
Una cosa interessante a cui si potrebbe pensare è creare una data, per esempio la data di creazione di un documento, a partire dalla quale è possibile calcolare dopo quanti giorni qualcuno legge il vostro documento e , se sono passati più di 10 giorni, si potrebbe dire qualcosa come: Ehi, questo documento è molto vecchio, non leggerlo!   
Per fare ciò avrete bisogno della data di oggi, come mostrato nell'esempio precedente, e la data di creazione. E' possibile impostare una data durante la creazione di un oggetto data, come nel seguente esempio: *docStarted= new Date(96,0,13)*   
Bisogna specificare per prima cosa l'anno, poi il mese (ricordatevi di decrementare il mese di uno!) e infine il giorno. E' anche possibile specificare l'ora: *docStarted= new Date(96,0,13,10,50,0)*   
I primi numeri definiscono sempre la data; essi sono seguiti dall'ora, i minuti e i secondi.

E' necessario che sappiate che Javascript non ha un vero e proprio tipo data, ma come vedete può lavorare con le date molto bene. Ciò funziona perché le date vengono rappresentate con il numero di millisecondi trascorsi dalle ore 0:0 del 1/1/1970. Può sembrare un po' complicato, ma è un metodo comune per rappresentare le date sui computer. Ma non preoccupatevi di questo, avrete a che fare soltanto con le relative funzioni e non è per niente difficile. Ho voluto dirlo soltanto per completezza.

**Numeri casuali**

Un problema comune riguarda l'uso di numeri casuali nei linguaggi di programmazione. Per il momento la funzione di generazione di numeri casuali in Javascript non funziona (credo comunque che verrà presto implementata), quindi dovremo sfruttare qualche trucco. Quella che vedremo è una tecnica che quasi tutti i compilatori che conosco utilizzano per generare numeri casuali. Sì, i numeri casuali vengono *calcolati* in base all'ora e alla data della tua macchina. Penso che la corrispondente funzione Javascript userà anch'essa questo metodo o qualcosa del genere. Come vi ho già detto prima, l'ora è rappresentata da un numero molto grande; è possibile utilizzare questo numero per fare qualche calcolo. Per esempio, potremmo calcolarne il seno e prendere il suo valore assoluto. Otterremo così un numero compreso tra 0 e 1. Poiché il valore dell'ora cambia ogni millisecondo, non correremo il rischio di ottenere lo stesso numero due volte di seguito. Se però vogliamo calcolare più numeri casuali in una sequenza temporale molto breve, non dobbiamo usare soltanto il seno: otterremmo una serie di numeri che segue la curva del seno! Ciò non è esattamente casuale. Ma se vogliamo calcolare un numero casuale e dopo 20 secondi ne vogliamo calcolare un altro, la funzione che vi ho proposto è certamente adatta.

Questo è un numero casuale: 0.8747264641424469

Ecco il codice di questo esempio:

<html>

<head>

<script language="JavaScript">

function RandomNumber() {

oggi = new Date();

num= Math.abs(Math.sin(oggigetTime()));

return num;

}

</script>

</head>

<body>

<script language="JavaScript">

<!--

document.write("Questo è un numero casuale:", RandomNumber());

// -->

</script>

</body>

</html>

Naturalmente la funzione di generazione di numeri casuali appena mostrata non è adatta per tutti gli scopi; serve soltanto a dare un'idea di come può essere realizzata una funzione di questo tipo. Vi presento ora una funzione che ho avuto da Maynard Demmon: dovete semplicemente impostare la variabile limits al valore che desiderate, per esempio 100, e otterrete un 'buon' numero casuale tra 0 e 99. Ecco il codice:

function random() {

today = new Date();

num = today.getTime();

num = Math.round(Math.abs(Math.sin (num)\*1000000)) % limits;

return num;

}

**Creazione di finestre**

La creazione di finestre è una caratteristica notevole di Javascript. Potete generare nuove finestre, caricare un documento HTML, navigare in Internet, tutto con Javascript. Ora vi mostrerò come è possibile aprire una finestra e scrivere qualcosa al suo interno. Se cliccate su questo pulsante vedrete quello che vado a spiegarvi.

Inizio modulo

Fine modulo

Rompendo con la tradizione non ho scritto *Ciao mondo!* nella pagina ...   
Ecco il sorgente:

<html>

<head>

<script language="JavaScript">

function WinOpen() {

msg=open("","DisplayWindow","toolbar=no,directories=no,menubar=no width=300 height=300");

msg.document.write("<HEAD><TITLE>Ehi!</TITLE></HEAD>");

msg.document.write("<CENTER><h1><B>E' veramente fantastico!</B></h1></CENTER>  
");

}

</script>

</head>

<body>

<form>

<input type="button" name="Button1" value="Clicca qui" onclick="WinOpen()">

</form>

</body>

</html>

Come al solito, al pulsante è associata una funzione, WinOpen(), che crea una nuova finestra invocando il metodo open.   
Il primo argomento contiene l'URL della pagina: qui potete inserire l'indirizzo di un documento HTML che volete caricare. Se lo lasciate vuoto ("") non verrà caricata nessuna pagina e potrete scriverci con Javascript!

Infatti, proviamo ora ad inserire una url per vedere cosa succede:  
l'URL inserita si riferisce alla pagina principale di questo sito, ovvero:  
http://plent.altervista.org/index.htm che andremo a scrivere nella function window open() tra le doppie virgolette di msg.

Poichè in questa pagina sussistono du pulsanti di apertura finestre in modalità completamente diverse, verranno scritte due funzioni corrispondenti a ciascun pulsante.

//function WinOpen1() {  
msg=open("","DisplayWindow","toolbar=no,directories=no,menubar=no width=300, height=300");  
msg.document.write("<HEAD><TITLE>il caimano</TITLE></HEAD>");  
msg.document.write("E' veramente fantastico!");  
}

e l'altra è:  
//function WinOpen2() {  
msg=open("http://plent.altervista.org/index.htm","DisplayWindow","toolbar=no,directories=no,menubar=no width=300, height=300");  
}

Inizio modulo

Fine modulo

L'argomento successivo specifica il nome della finestra: qui potete scrivere quasi tutto, ma non ha nessun effetto nel nostro esempio. Avrete comunque un messaggio d'errore se scrivete *Display Window* (con uno spazio tra le due parole - Netscape vi darà un'informazione strana, infatti ho impiegato una buona mezz'ora per riuscire a trovare l'errore in base alle sue indicazioni!).   
L'ultimo argomento specifica le proprietà della finestra ed è veramente interessante. Potete dire se volete una toolbar, delle scrollbar ... Se scrivete *toolbar=yes* otterrete una toolbar nella vostra finestra. Nella lista seguente ci sono alcune proprietà che potete modificare nel modo che vi ho appena descritto e senza **nessuno spazio** tra di loro! Ecco ciò che potete modificare nella vostra pagina:

toolbar  
location  
directories  
status  
menubar  
scrollbars  
resizable  
copyhistory  
width=*pixels*  
height=*pixels*

Al posto di *pixels* dovete scrivere il numero di pixel, così potete specificare al browser quanto deve essere grande la vostra finestra.   
Dopo aver aperto la vostra finestra e l'avete associata alla variabile msg, potrete scrivere al suo interno. Potete utilizzare normale codice HTML! Potreste quindi costruire un documento HTML utilizzando l'input immesso da un utente in una form; ad esempio, potreste costruire una pagina con una form in cui l'utente scrive il suo nome e poi creare un nuovo documento HTML contenente il suo nome! Fino ad alcuni mesi fa ciò era possibile soltanto con script CGI!

**Nota:** Quando scrivete qualcosa all'interno di una finestra, dovete sempre inserire un <br> alla fine del testo, altrimenti potreste non vederne l'ultima riga. Ciò accade perché il browser scrive soltanto linee complete e se non trova la fine di una linea rimane in attesa di altro testo.   
**Un'altra cosa importante:** Se volete inserire un'immagine in una nuova finestra, assicuratevi di impostare le proprietà *height* e *width* del tag <img> altrimenti non vedrete alcuna immagine o la visualizzazione della pagina darà qualche errore. Possono accadere infatti problemi molto strani e non vi aspettereste che la vera 'responsabile' è la vostra immagine. Quindi specificate sempre le immagini come nel seguente esempio e non avrete problemi:

<img src="mycool.gif" height=100 width=100>

Introduzione a JavaScript  
(Parte 5)

L'inserimento di dati in una form è molto importante per certe pagine Web. L'input delle form spesso viene inviato al server per le opportune elaborazioni. Javascript consente di controllare l'input delle form prima del loro invio al server. Come prima cosa voglio mostrarvi come è possibile controllare l'input di una form, poi vedremo come è possibile ottenere l'invio di informazioni con Javascript o HTML.

Prima di tutto creeremo una piccola form. La pagina HTML conterrà due caselle di testo. L'utente dovrà scrivere il suo nome nella prima casella e l'indirizzo di e-mail nella seconda. Potete inserire qualsiasi cosa negli elementi della form, poi cliccate sul relativo pulsante. Provate anche a non inserire nulla e cliccate sul pulsante.

Inizio modulo

Inserisci il tuo nome:  


Inserisci il tuo indirizzo di e-mail:  


Per quanto riguarda la prima casella di testo, vedrete un messaggio di errore quando viene lasciata in bianco, mentre un qualsiasi input viene considerato valido. Naturalmente ciò non previene l'inserimento di un nome sbagliato. Il browser accetta anche numeri, quindi se inserite '17' otterrete un 'Ciao 17!'.   
Il controllo della seconda casella è leggermente più sofisticato. Provate ad inserire una semplice stringa, il vostro nome per esempio. Non verrà accettato (a meno che non abbiate un @ nel vostro nome...). Il criterio di accettazione dell'input come un indirizzo di e-mail valido consiste nel verificare la presenza del simbolo @. Viene accettato anche il solo simbolo @, che non è certamente significativo. Per semplicità abbiamo deciso di controllare soltanto la presenza di @, dal momento che ogni indirizzo di e-mail in Internet contiene questo simbolo.

Come è fatto lo script che crea questa form e ne controlla l'input? Eccolo:

**<html>**

**<head>**

**<script language="JavaScript">**

**<!--**

**function test1(form) {**

**if (form.text1.value == "")**

**alert("Inserisci una stringa, per favore!")**

**else {**

**alert("Ciao "+form.text1.value+"! Inserimento dati ok!");**

**}**

**}**

**function test2(form) {**

**if (form.text2.value == "" ||**

**form.text2.value.indexOf('@', 0) == -1)**

**alert("Indirizzo di e-mail non valido!");**

**else alert("OK!");**

**}**

**// -->**

**</script>**

**</head>**

**<body>**

**<form name="first">**

**Inserisci il tuo nome:<br>**

**<input type="text" name="text1">**

**<input type="button" name="button1" value="Controlla" onClick="test1(this.form)">**

**<P>**

**Inserisci il tuo indirizzo di e-mail:<br>**

**<input type="text" name="text2">**

**<input type="button" name="button2" value="Controlla" onClick="test2(this.form)">**

**</body>**

Diamo prima uno sguardo al codice HTML nel corpo del documento. Abbiamo creato due caselle di testo e due pulsanti. I pulsanti invocano le funzioni test1(...) o test2(...) in base a quale di loro viene selezionato e viene passato come argomento *this.form* per permetterne l'accesso ai suoi elementi.   
La funzione test1(form) controlla se la stringa è vuota; questo è fatto con *if (form.text1.value == "")...*. 'form' è la variabile a cui viene assegnato il valore 'this.form' nella chiamata di funzione. Possiamo ottenere il valore del dato inserito nella casella di testo usando 'value' in combinazione con *form.text1*. Per controllare se la stringa è vuota la confrontiamo con "". Se la stringa inserita è uguale a "", verrà segnalato all'utente un messaggio di errore, altrimenti verrà segnalato un messaggio di ok.   
Notate che se l'utente inserisce solo spazi, verrà considerato come input valido! Potete naturalmente controllare anche questa possibilità ed escluderla. Penso che non dovrebbe essere troppo difficile con le informazioni che avete fin qui acquisito.   
Vediamo ora la funzione *test2(form)*. Anche questa funzione controlla che sia stato inserito qualcosa nella relativa casella facendo un confronto con la stringa vuota, ma ho aggiunto qualcosa all'*if*. Il simbolo || indica l'operatore OR, di cui abbiamo parlato nella parte 6 di questo tutorial.   
L'istruzione *if* controlla se il primo o il secondo confronto sono veri; se almeno uno lo è viene eseguito il comando successivo. Ciò vuol dire che otterrete un messaggio di errore se la vostra stringa è vuota o non contiene un @. La seconda operazione dell'istruzione *if* controlla se la stringa inserita contiene un @.

Quali diverse possibilità esistono per inviare il contenuto di una form? Il modo più semplice consiste nell'invio tramite e-mail. Questo è il metodo che ci accingiamo ad analizzare. Se volete ottenere un feedback senza e-mail e volete che il server gestisca l'input automaticamente, dovete usare CGI, per il momento. Dovete utilizzare CGI, per esempio, se volete creare un motore di ricerca come Yahoo, in cui l'utente ottiene un risultato velocemente dopo l'inserimento dei dati nella form. L'utente non deve attendere che il gestore del server legga l'input, vada alla ricerca dei dati richiesti e fornisca il relativo output: il tutto viene fatto automaticamente dal server. Javascript non è in grado di fare cose del genere. Anche se volete creare un guestbook (registro dei visitatori), non potete fare in modo con Javascript che il server aggiunga automaticamente informazioni ad una pagina HTML esistente. Soltanto con CGI potete ottenere questo al momento. Potete comunque creare un guestbook utilizzando l'e-mail: dovrete però inserire le informazioni manualmente. Può andar bene se non vi aspettate 1000 mail al giorno!   
Il codice che segue è in puro HTML, quindi non abbiamo bisogno di Javascript! Naturalmente se vogliamo controllare l'input prima di inviare la form, dovremo scrivere qualche funzione Javascript. Devo dire che il comando mailto non funziona con tutti i browser, ma i browser più recenti lo supportano.

**<FORM METHOD=POST ACTION="mailto:nome@domi.nio">**

**<H3>Ti piace questa pagina?</H3>**

**<INPUT NAME="choice" TYPE="RADIO" VALUE="1">No per niente.<BR>**

**<INPUT NAME="choice" TYPE="RADIO" VALUE="2" CHECKED>E' una perdita di tempo.<BR>**

**<INPUT NAME="choice" TYPE="RADIO" VALUE="3">E' il peggior sito della Rete.<BR>**

**<INPUT NAME="submit" TYPE="SUBMIT" VALUE="Send">**

**</FORM>**

Con questa form avrete un feedback via e-mail. L'unico problema è che riceverete una mail che può sembrare incomprensibile a prima vista. Talvolta tutti gli spazi vengono sostituiti con '+' o con '%20'. Le+mail+possono+avere+questo+aspetto. Esistono dei parser sulla Rete che formattano la mail in un aspetto più gradevole.

C'è un'altra cosa che può rendere gli elementi della vostra form un po' più user-friendly. Potete stabilire su quale elemento impostare inizialmente il focus oppure potete indicare al browser di posizionare il focus sulla form in cui l'utente ha inserito dei dati sbagliati. Ciò vuol dire che il browser posizionerà il cursore nell'elemento della form specificato in modo tale che l'utente non deve cliccarvi sopra per inserire i dati. Potete fare ciò con la seguente funzione:

function setfocus() {

document.first.text1.focus();

return;

}

Questo script imposta il focus sulla prima casella di testo della form dell'esempio illustrato prima. Occorre specificare il nome dell'intera form, nel nostro esempio *first*, e il nome del singolo elemento, *text1*. Se volete impostare il focus su questo elemento quando la pagina viene caricata, dovete aggiungere la proprietà onLoad al tag <body>; per esempio:

<body onLoad="setfocus()">

## Esempi in JavaScript

Questa riga e' stata scritta in javaScript

### (Esercitazioni)

### Modalità di inserimento dei dati

<INPUT attributi> ..... scritta ....... </INPUT>

*Gli attributi* possono essere:

**NAME**

identificatore del campo, per esempio, nell'operazione matematica si scrive: operando1, in generale può essere un nome qualsiasi.

**VALUE**

valore iniziale. Per esempio, volendo far comparire zero, si scrive: value=0. Nel caso di un tasto può essere la stringa del tasto, per es. value=submit.

**type**

tipo del campo, nel caso di un tasto si scrive: type=button

Questo è un modo per fornire un messaggio attraverso una finestra.

Ecco il sorgente di questo Form:

|  |
| --- |
| < form > < input name=tasto value=Controlla onclick=allerta() type=button > < /form > In questo caso, onclick richiama la funzione "allerta". |

Inizio modulo

Fine modulo

Premendo il tasto, si apre una finestra con un testo.

|  |
| --- |
| < form > < INPUT name=tasto value=Apri onclick=WinOpen() type=button >  < /form >  In questo caso, onclick richiama la funzione "WinOpen". |

Inizio modulo

Fine modulo

Somma di due numeri

In questo form la programmazione è ad oggetti: la form deve avere un nome e ogni campo di INPUT ha come nome la variabile e la larghezza del campo stesso. Il campo dove verrà scritto il risultato, gli viene attribuito un risultato iniziale uguale a zero.

Il campo d'uscita, oltre ai soliti attributi, ha onclick che richiama la form che contiene l'operazione risultato.

Ogni operando è un oggetto contenuto nella parola chiave "Number".

|  |
| --- |
| < FORM name=somma>> < INPUT name=operando1 size=5 > + < INPUT name=operando2 size=5 > = < INPUT name=risultato size=5 value=0 >  < INPUT name=button value=calcola onclick="somma.risultato.value = Number(somma.operando1.value)+ Number(somma.operando2.value)" type=button >  < INPUT TYPE=RESET VALUE="Reset" >  < /FORM > |

Come si può notare, abbiamo introdotto un altro tipo di Input, ovvero, il RESET.

Inizio modulo

 +  = 



Fine modulo

Disione tra due numeri

Inizio modulo

 /  = 



Fine modulo

## Somma di più numeri

Introdurre un numero per ogni campo.   
L'operazione matematica è eseguita sfruttando l'evento "*OnChange*".

|  |  |
| --- | --- |
| < FORM name=sommamult > 1^ cifra:       < INPUT name=numero1 onchange=EseguiAddizione() size=10 > 2° cifra:       < INPUT name=numero2 onchange=EseguiAddizione() size=10 > 3° cifra:      < INPUT name=numero3 onchange=EseguiAddizione() size=10 >    Totale:   < INPUT name=totale onfocus=Foc() readOnly size=17 >  < /FORM > | Dal sorgente possiamo notare che ogni INPUT ha come name: numero1, numero2, ecc., le stesse variabili usate dalla funzione. Per richiamare la predetta funzione, si usa l'evento OnChange. Mentre OnClick serve a gestire i tasti, On change è più adatto a gestire l'evento semplicemente con la pressione del tasto sinistro del mouse . In ogni caso, l'istruzione OnChange richiama una funzione: EseguiAddizione(). Ogni volta che inseriamo un dato in un campo e passiamo all'altro, viene eseguita la somma come in una calcolatrice. |

Inizio modulo

1^ cifra:      

2° cifra:      

3° cifra:      

Totale:  

Fine modulo

## Moltiplicazione

Moltiplicazione dei due valori contenuti nelle seguenti caselle di testo.  
L'operazione matematica è eseguita sfruttando l'evento "*OnClick*".

Inizio modulo

 \*  = 



Fine modulo