



JAVASCRIPT

Marco Masseroli, PhD
masseroli@biomed.polimi.it

JAVASCRIPT



JAVASCRIPT

Introduzione:

JavaScript è un linguaggio interpretato (come VisualBasic, Perl, ...) e un interprete JavaScript è disponibile all'interno dei browser Web.

E' un linguaggio "orientato agli eventi", ovvero con cui si possono definire funzioni che rispondano a eventi quali: azioni del mouse, selezione tasti, apertura/chiusura finestre, caricamento di una pagina, ...

Ciò permette di rendere "dinamica" la visualizzazione di un documento HTML e avere buona interattività con l'utente.



JAVASCRIPT

JavaScript e CGI:

Come JavaScript anche i CGI permettono dare dinamicità a un documento HTML (cambiarne contenuto e/o visualizzazione in relazione alle scelte dell'utente).

Essendo i CGI eseguiti sul server, ciò avviene mediante successive connessioni al server (una per inviare al client la pagina, una per ricevere le scelte dell'utente, una per restituire al client il risultato dell'elaborazione delle scelte dell'utente).

JavaScript, eseguendosi sul client, richiede solo una connessione al server, per caricare il codice HTML e JavaScript assieme.



JAVASCRIPT

JavaScript lato client e lato server:

Il JavaScript propriamente detto è un linguaggio che si esegue su un client (“client-side”).

Tuttavia esistono anche istruzioni JavaScript che si eseguono sul server (“server-side”). Queste sono spesso utilizzate nella realizzazione di Active Server Pages (ASP).



JAVASCRIPT

Cosa JavaScript non è:

JavaScript non è un linguaggio di scopo generale, ovvero non consente il controllo delle varie risorse di un computer.

Ogni programma JavaScript può agire solo sugli elementi del documento HTML in cui è inserito e sulle finestre del client browser in cui viene eseguito.

Non è un sostituto dei CGI dato che non consente l'accesso a file. Tuttavia, unito a un CGI ne migliora le prestazioni (numero di connessioni) consentendo controllare e depurare subito dagli errori i dati prima di passarli a un CGI.



JAVASCRIPT

JavaScript e Java:

Java è un linguaggio orientato agli oggetti di scopo generale e ha alcune caratteristiche che lo rendono utilizzabile in internet (applet Java, agenti software, ...).

JavaScript è molto più limitato e semplice di Java: per es. gestisce solo istanze (e non classi) di oggetti, e solo dati di tipo numerico, booleano, e testo.

Le principali differenze sono:

- JavaScript è interpretato ed eseguito sul client;
Java è compilato sul server ed eseguito sul client.





JAVASCRIPT

- JavaScript è basato su oggetti; Java è orientato a oggetti. Gli oggetti usati da JavaScript esistono quando si inizia ad eseguire il programma; non esistono classi e quindi nemmeno concetti di ereditarietà e incapsulamento.
- Il codice JavaScript si inserisce nella pagina HTML in cui si esegue.
- JavaScript è un linguaggio debolmente tipificato: non è necessario dichiarare il tipo dei dati e si realizzano automaticamente conversioni da un tipo ad un altro; Java è invece fortemente tipificato. →



JAVASCRIPT

- Sia JavaScript che Java (applet) non permettono scrivere sul disco del client. Con JavaScript non possono nemmeno leggere file.

JavaScript ha similitudini con un altro linguaggio, ActiveX, sviluppato per rendere dinamiche le pagine HTML.

ActiveX è sviluppato da Microsoft e utilizza la stessa sintassi di VisualBasic.



JAVASCRIPT

Uso di JavaScript:

Le istruzioni JavaScript si eseguono in un browser e vengono scritte all'interno del codice HTML di pagine Web. Vi sono vari modi di usare istruzioni JavaScript entro documenti HTML:

1. inserendole entro l'etichetta `<SCRIPT> ... </SCRIPT>`;
2. specificando un file sul server Web che le contenga;
3. usandole come valore di attributi di etichette HTML;
4. usandole dentro alcune etichette HTML per gestire eventi.

Questi modi possono coesistere in una stessa pagina HTML.



JAVASCRIPT

1. L'etichetta `<SCRIPT> ... </SCRIPT>`:

```
<SCRIPT TYPE= "scriptType" LANGUAGE="languageName"  
    SRC="location">.....  
</SCRIPT>
```

LANGUAGE specifica il tipo e la versione del linguaggio in cui sono scritte le istruzioni contenute. Se questa versione del linguaggio non è supportato dal browser le istruzioni non vengono interpretate, ma visualizzate come codice. Per evitarlo si inseriscono le istruzioni come commenti (per l'HTML) e se ne specifica un contenuto alternativo con l'etichetta

```
<NOSCRIPT> ..... </NOSCRIPT>
```





JAVASCRIPT

Esempio 1: istruzioni JavaScript in etichetta <SCRIPT>

```
<HTML LANG="it">
<HEAD><TITLE>La mia prima pagina con JavaScript</TITLE></HEAD>
<BODY> <SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript">
<!--
    document.write("<p>Questa è la mia prima pagina in JavaScript :-)</p>");
// -->
</SCRIPT>
<NOSCRIPT> Questo browser non può eseguire istruzioni JavaScript!!
</NOSCRIPT>
</BODY> </HTML>
```

L'interprete JavaScript genera un errore se non riconosce una istruzione; la "//" (o "/ * ... */) indicano commenti in JavaScript. →



JAVASCRIPT

Esempio2: uso etichetta <NOSCRIPT>

```
<HTML LANG="it">
```

```
<HEAD> <TITLE>Esempio di buon uso dell'etichetta <NOSCRIPT>
```

```
</TITLE> </HEAD><BODY>
```

```
<NOSCRIPT>
```

```
<CENTER> <H4>!!Attenzione!! </H4>
```

```
<P>Questa pagina contiene codice JavaScript ma voi non potete eseguirlo o  
perch&eacute; il vostro browser non lo permette (versione troppo antica!),  
o perch&eacute; non ha abilitata la funzione per eseguire JavaScript.
```

```
In tal caso abilitarla selezionando, in Netscape Communicator,
```

```
<EM>Edit</EM>, <EM>Preferences</EM>, <EM>Advanced</EM></P>
```

```
</CENTER> </NOSCRIPT>
```

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





JAVASCRIPT

E' consigliabile inserire le definizioni delle funzioni JavaScript usate in una pagina HTML all'interno dell'intestazione (`<HEAD> </HEAD>`) della pagina.

In tal modo, se la funzione definisce una risposta ad un evento provocato da un utente all'interagire con la pagina HTML, al prodursi l'evento la funzione che ne definisce la risposta già è caricata e disponibile.

Si ricordi che, al contrario dell'HTML, JavaScript distingue tra maiuscole e minuscole! →



JAVASCRIPT

Esempio3: funzioni JavaScript in etichetta <HEAD>

```
<HTML LANG="it">
<HEAD><TITLE>Definendo funzioni in JavaScript</TITLE>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript">
<!--
    function saluta() {
        alert ("Benvenuto alla mia pagina Web");
    }
// -->
</SCRIPT> </HEAD>
<BODY onLoad="saluta();">
</BODY> </HTML>
```



JAVASCRIPT

2. File sul server Web contenenti istruzioni JavaScript:

Le istruzioni JavaScript vengono scritte in un file di testo residente sul server Web nello stesso direttorio della pagina HTML (o in un suo sottodirettorio). Tramite l'attributo *SRC="location"* dell'etichetta `<SCRIPT>` si indica nella pagina HTML la localizzazione del file che contiene le istruzioni JavaScript utilizzate nella pagina.

```
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript"  
        SRC="http://server/direttorio/file.javaScript">  
<!--  
    ... messaggio di errore  
// -->  
</SCRIPT>
```





JAVASCRIPT

Il file indicato in *SRC* verrà scaricato all'interno del client browser assieme al file contenente il codice HTML della pagina HTML alla richiesta della pagina stessa.

E' utile che i file sul server Web contenenti le istruzioni JavaScript abbiano estensione *.javaScript (o *.js) e che sul server tale estensione sia associata al tipo file JavaScript.

In Esplora risorse, selezionare Visualizza\Opzioni cartella... e pannello "Tipi di file"; aggiungere "Nuovo tipo..." con Estensione associata: *.javaScript e Contenuto (MIME): application/x-javascript.



JAVASCRIPT

3. Istruzioni JavaScript valore di attributi di etichette HTML:

Praticamente tutte le etichette HTML hanno attributi che, a seconda dei valori che assumono, fanno sì che l'etichetta si comporti in un dato modo. Tali valori possono essere fissi o essere variabili o funzioni JavaScript.

Nel caso siano variabili JavaScript la sintassi da usare è la seguente: `<ETICHETTA ATTRIBUTO="{variable};">`

Nel caso siano funzioni JavaScript la sintassi è la seguente:

`<ETICHETTA ATTRIBUTO="{funzione();};">`





JAVASCRIPT

Esempio4: variabili JavaScript come valore attributi HTML

```
<HTML LANG="it">  
<HEAD> <TITLE> Usando variabili JavaScript come valore di attributi  
di etichette HTML </TITLE>  
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript">  
<!--  
    var coloreFondo="#FF00FF";  
// -->  
</SCRIPT> </HEAD>  
<BODY BGCOLOR="{coloreFondo};">  
</BODY> </HTML>
```





JAVASCRIPT

Esempio5: funzioni JavaScript come valore attributi HTML

```
<HTML LANG="it">
<HEAD> <TITLE> Usando funzioni JavaScript come valore di attributi
di etichette HTML </TITLE>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript">
<!--
    function doppio(x) {
        return (2*x);
    }
// -->
</SCRIPT> </HEAD> <BODY>
<FORM> <INPUT TYPE="TEXT" SIZE="{doppio(40)};"> </FORM>
</BODY> </HTML>
```





JAVASCRIPT

Questo uso delle variabili JavaScript serve solo per assegnare valori variabili agli attributi di etichette HTML. Quindi a rendere dinamica la visualizzazione di pagine Web, non a rendere variabile (dinamico) il loro contenuto.

Pertanto la costruzione:

...

```
<TITLE> &{nomeTitolo}; </TITLE>
```

...

non ha senso!!





JAVASCRIPT

Tuttavia però tale uso delle variabili JavaScript non funziona sempre in tutti i client web browser.

Pertanto è preferibile costruire dinamicamente l'intero testo dell'etichetta HTML usando i metodi *write(...)* o *writeln(...)*.

Esempi:

```
document.write("<BODY BGCOLOR=" + coloreFondo + ">");
```

oppure

```
document.write("<INPUT TYPE='TEXT' SIZE=" + doppio(40) + ">");
```

Pertanto, usando cicli, variabili e funzioni, è possibile **generare dinamicamente** con JavaScript tutto il **codice HTML** di una pagina Web.



JAVASCRIPT

4. Istruzioni JavaScript per gestire eventi all'interno di etichette HTML :

Molte etichette HTML che definiscono elementi di pagine Web hanno associati una serie di eventi a cui rispondono. Ad esempio ad un link sono associati eventi quali: muovere il mouse sopra il link, farlo uscire dal link, cliccare il link.

Questi eventi, che si generano automaticamente, possono chiamare funzioni che rispondano e gestiscano tali eventi.

La sintassi generale da usare per gestire eventi è la seguente:

```
<ETICHETTA ATRIBUTO1="valore1" ... ONEVENTO1="istruzione  
JavaScript1" ONEVENTO2="funzioneJavaScript1()" ... >
```





JAVASCRIPT

Esempio6: messaggio in Barra di Stato al passaggio del mouse sopra un link

```
<HTML LANG="it">
<HEAD> <TITLE> Gestire eventi in JavaScript </TITLE> </HEAD>
<BODY>
<P>Visita questa <A HREF="http://meteo.yahoo.it/paese/italia.html"
  onMouseOver="window.status='Se devi viaggiare ti interesserà!';
  return true;"> pagina </A></P>
</BODY> </HTML>
```

Si noti l'uso di virgolette semplici (' ') per racchiudere le stringhe di testo. Così non si confondono con le doppie (" ") che racchiudono le funzioni di risposta all'evento. →



JAVASCRIPT

Esempio7: ... o definendo una funzione ponInStato():

```
<HTML LANG="it">
<HEAD> <TITLE> Gestire eventi in JavaScript </TITLE>
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript">
<!--
    function ponInStato() {
        window.status="Se devi viaggiare ti interesserà!"; return true;
    }
// -->
</SCRIPT> </HEAD> <BODY>
Visita questa <A HREF=" http://meteo.yahoo.it/paese/italia.html"
    onMouseOver="ponInStato();"> pagina </A>
</BODY> </HTML>
```





JAVASCRIPT

Eventi a cui rispondono gli elementi di una pagina HTML:

<u>Eventi</u>	<u>Elementi</u>
Abort:	image
Blur:	window, frame, tutti elementi di form
Click:	button, radio, checkbox, submit, reset, link
Change:	text, textarea, select
Error:	image, window
Focus:	window, frame, tutti elementi di form
Load:	body
MouseOut:	area, link
MouseOver:	link





JAVASCRIPT

Eventi

Elementi

Reset:	form
Select:	text, textarea
Submit:	submit
Unload:	body

Gli attributi delle etichette HTML che si riferiscono a tali eventi si definiscono con il nome dell'evento (con le iniziali maiuscole) preceduto dalla particella “*on*”: es. onAbort, onBlur, onClick, ... onLoad, onMouseOut, onMouseOver, ... onUnload.



JAVASCRIPT

Esempi in JavaScript e utilizzo della funzione *write(...)*:

- Esempio8: un separatore di testo (pagina in cui i paragrafi sono separati da una linea composta da un numero determinato di *; ...);
 - Esempio9: separatore di testo migliorato (inclusione del separatore entro una funzione per evitare la ripetizione di codice);
 - Esempio10: separatore di testo parametrizzato (aggiunta della possibilità di cambiare la lunghezza e il tipo di carattere di cui è composto il separatore, e la distanza di separazione tra i paragrafi);
 - Esempio11: una funzione per dimensione e colore testo;
 - Esempio12: un controesempio dell'uso di *write(...)*. →
-



JAVASCRIPT

Oltre alla funzione *write(...)* esiste anche *writeln(...)*.

Rispetto a *write(...)*, *writeln(...)* aggiunge un “a capo” alla fine della stringa di caratteri scritta. Tuttavia in HTML il ritorno a capo generalmente non ha effetto; al suo posto si usa l’etichetta `
`. Il ritorno a capo risulta rilevante solo all’interno delle etichette `<PRE> ... </PRE>` e `<XMP> ... </XMP>` dove è interpretato e visualizzato.



JAVASCRIPT

Metodi e caratteri speciali delle stringhe di testo:

string.length: ritorna la lunghezza della stringa di caratteri *string*.

string.substring(numFirstChar, numLastChar): estrae dalla stringa di caratteri *string* una sottostringa che contiene dal *numFirstChar* al *numLastChar* carattere della stringa *string*.

<u>Carattere</u>	<u>Descrizione</u>
<i>\n</i>	newline
<i>\t</i>	tab
<i>\r</i>	carriage return
<i>\f</i>	form feed
<i>\b</i>	backspace



JAVASCRIPT

Cicli e variabili in JavaScript:

Valgono tutti i cicli, condizionali e operatori del C e di Java.

In JavaScript le variabili possono (ma non è necessario) essere dichiarate usando *var* (es. *var testo="Ciao";*)

Nelle funzioni le variabili sono globali; per renderle locali le si devono dichiarare (es. *for (var i=0; i<7; i++) {...}*).

Gli argomenti che si passano a una funzione possono essere in numero diverso in chiamate successive dato che la funzione ogni volta crea una matrice con gli argomenti che le si passano.



JAVASCRIPT

Creazione di matrici in JavaScript:

Creazione

a=new Array(7);

i=new Array(10); oppure i[0]; i[1]; ... i[9];

Creazione e inizializzazione

a=new Array{"Luca", "Paolo", ... "Ciao"};

a[78]="Ciao"; Così si crea una matrice di 79 elementi dove però i primi 78 sono vuoti (quindi si possono usare solo per assegnazioni) e il 79-novesimo è inizializzato a *"Ciao"*.



JAVASCRIPT

Oggetti di una pagina HTML:

Ogni volta che si carica una pagina HTML in un browser, l'interprete JavaScript crea automaticamente una serie di oggetti che si possono usare per programmare in JavaScript.

Ogni oggetto possiede una serie di variabili di istanza e di metodi. In JavaScript le prime sono denominate proprietà e i secondi funzioni o metodi.

Le proprietà definiscono le caratteristiche di un oggetto, ovvero come questo appare; due oggetti con le stesse proprietà differiscono per il valore che queste assumono. →



JAVASCRIPT

Le funzioni definiscono le azioni che un oggetto può compiere.

Oltre a proprietà e funzioni gli oggetti di una pagina HTML hanno associati degli eventi a cui possono rispondere.

Gran parte della programmazione in JavaScript consiste nel programmare risposte a tali eventi, nelle quali si assegnano valori di proprietà di altri oggetti, o li si modificano tramite loro funzioni. Questo da dinamicità visuale alla pagina HTML.

Attenzione che in JavaScript tutti i nomi di oggetti, proprietà, funzioni, parole chiave, operatori e variabili, sono “case sensitive”.



JAVASCRIPT

Oggetti creati automaticamente da un browser al caricare una pagina HTML:

window: è l'oggetto principale, a cui appartengono tutti gli altri come se fossero sue proprietà. Si crea un oggetto di questo tipo ogni volta che si apre una nuova finestra del browser.

navigator: contiene informazioni riguardo al browser utilizzato (nome, versione, tipi MIME supportati e plug-in installati).

location: URL della pagina caricata.

history: lista degli URL visitati.

document: la pagina HTML; è il contenitore di tutti gli oggetti presenti nella pagina: frame, form, immagini, link, ...





JAVASCRIPT

Le proprietà dell'oggetto document dipendono dagli oggetti presenti nella pagina HTML. Ogni oggetto della pagina è una proprietà di document, e a sua volta può essere composto da altri oggetti che ne costituiscono le sue proprietà.

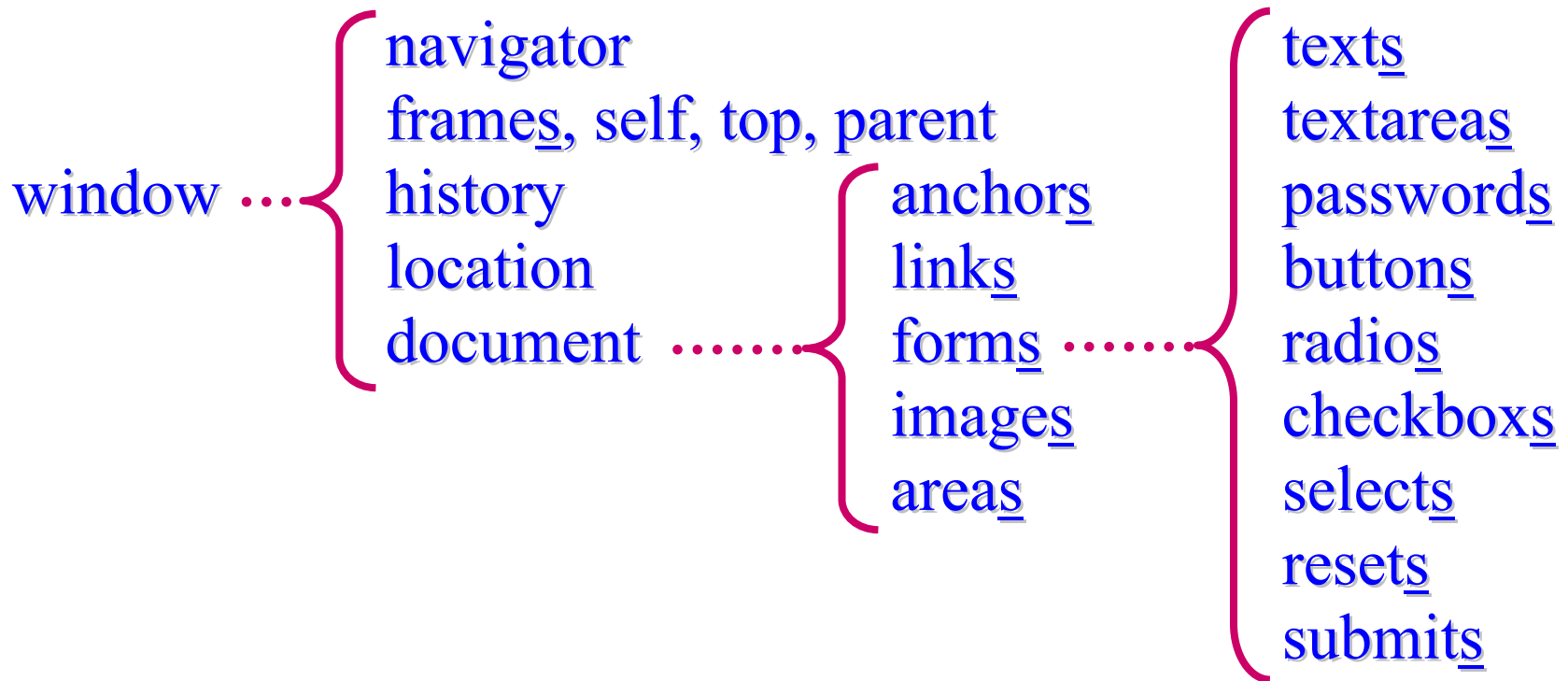
Pertanto gli oggetti di una pagina HTML sono legati da una gerarchia.

Ogni volta che si voglia accedere a una proprietà o a una funzione di un oggetto si devono citare tutti i suoi oggetti padri, separati da un punto (es. la proprietà di un bottone di un form si chiama “*window.document.form.button.proprietà*”). Solo il padre window può essere tralasciato. →



JAVASCRIPT

Gerarchia degli oggetti di una pagina HTML:



In plurale sono gli oggetti di cui possono esistere più istanze in uno stesso documento.





JAVASCRIPT

Uso degli oggetti di una pagina HTML in JavaScript:

Un browser crea gli oggetti di una pagina HTML mentre carica la pagina (che viene caricata dall'alto al basso).

Pertanto bisogna porre attenzione che il codice JavaScript non usi gli oggetti di una pagina HTML prima della loro creazione!

Tuttavia nelle definizioni di funzioni JavaScript (poste nell'intestazione di una pagina HTML) si può fare riferimento ad oggetti non ancora creati (es. tutti quelli di document).

Infatti la definizione non comporta l'uso, che avviene quando la funzione viene chiamata (cosa che deve quindi avvenire dopo la creazione degli oggetti cui la funzione si riferisce).



JAVASCRIPT

L'oggetto window:

E' l'oggetto principale a partire dal quale si possono creare altri oggetti *window* che si possono controllare dalla finestra padre.

L'oggetto *window* ha funzioni oltre che per aprire finestre, anche per chiuderle, agire sui frame, e inviare "dialog box" (finestrelle per messaggi di errore, conferma, o inserimento dati).

Inoltre, mantiene una matrice per ogni tipo di oggetto che può essere presente in un documento (es. frame, immagine, link, form, ...).



JAVASCRIPT

Apertura di una finestra:

Si usa la funzione *open()* con l'URL della pagina da caricare nella nuova finestra: *nuovaFinestra=window.open("URL");*

La nuova finestra deve essere assegnata a una variabile, *nuovaFinestra*, (che costituisce una proprietà della finestra da cui si apre la nuova finestra) per poter poi operare su di lei (per es. per chiuderla).

Tale variabile rappresenta l'oggetto finestra, non il suo nome. Quest'ultimo può essere definito all'apertura della finestra assieme ad altri parametri; per cui la sintassi completa risulta:





JAVASCRIPT

variabile=window.open("URL", "nomeFinestra", "parametriApertura");

dove:

URL: indirizzo della pagina da caricare nella nuova finestra;

nomeFinestra: nome della finestra (che per es. si può usare come valore dell'attributo *TARGET* dei link);

parametriApertura: stringa di caratteri che contiene i valori degli attributi della finestra: *toolbar*, *location*, *directories*, *status*, *menubar*, *scrollbars*, *resizable*, *copyhistory* (tutti con valori YES o NO), *width*, *height* (dimensioni in pixel della finestra):

newFinestra=window.open("http://www.medinfopoli.polimi.it/default.htm", "Fin2", "toolbar=yes,location=no,resizable=no,width=400,height=400");



JAVASCRIPT

Chiusura di una finestra:

Si usa la funzione *close()*:

nuovaFinestra.close();

dove *nuovaFinestra* è la variabile assegnata alla finestra all'atto della sua apertura.



JAVASCRIPT

“dialog box” per messaggi di errore, conferma, inserimento dati:

Vi sono diversi tipi di “dialog box”; per ognuno esiste una diversa funzione JavaScript: *alert("messaggio")*, *confirm("messaggio")*, e *prompt("messaggio")*.

alert("messaggio"): visualizza una finestrella con “messaggio” e un bottone “OK”. Fino a che non si clicca il bottone non prosegue l’interpretazione dello script e del codice HTML che lo contiene. E’ usato per attirare l’attenzione dell’utente su messaggi di errore, aiuto, informazioni.





JAVASCRIPT

confirm("messaggio"): visualizza una finestra con "messaggio", un bottone "OK" e uno "Annulla". Ritorna un valore booleano (*true* se viene cliccato "OK", *false* se cliccato "Annulla") e deve quindi usarsi in una frase di assegnazione o in una espressione logica. E' usato per messaggi che richiedono la conferma di una opzione (si/no).

prompt("messaggio"): visualizza una finestra con "messaggio" e fornisce un campo per l'inserimento dati con un valore di default che può essere fornito (es. *window.prompt("Scrivi il tuo nome ... ", "Marco")*);). Usato per messaggi con inserimento di dati, è il modo più semplice per interagire con l'utente.



JAVASCRIPT

Esempi di utilizzo delle funzioni dell'oggetto *window*:

- Esempio 13: apertura e chiusura di una finestra (*open()*, *close()*);
- Esempio 14: invio finestrella per messaggi di errore, aiuto, informazioni (*alert("messaggio")*);
- Esempio 15: invio finestrella per messaggi di conferma (*confirm("messaggio")*);
- Esempio 16: invio finestrella per messaggi di inserimento dati (*prompt("messaggio")*).



JAVASCRIPT

I temporizzatori:

Vi sono due funzioni JavaScript che gestiscono l'uso di temporizzatori: *setTimeout("azione", tempo)* e *clearTimeout(temporizzatore)*.

setTimeout("azione", tempo): definisce un temporizzatore; trascorso il *tempo* (in millisecondi) viene eseguita l'*azione* specificata. Un temporizzatore si esegue solo una volta. Per continuarne l'esecuzione, l'*azione* deve contenere un altro *setTimeout(...)* (si vedano gli Esempi 17 e 18).

clearTimeout(temporizzatore): interrompe il temporizzatore specificato (Esempio 19).





JAVASCRIPT

La barra di stato di una finestra:

Si può accedere alla barra di stato della finestra di un browser tramite le proprietà *status* o *defaultStatus* di *window*.

Pertanto si può scrivere un messaggio nella barra di stato assegnando un testo a una di tali proprietà (es. `window.status="Messaggio"`).

Usando i temporizzatori si può dare movimento ai messaggi (Esempio20).



JAVASCRIPT

L'oggetto navigator:

Raccoglie l'informazione relativa al browser, la quale varia a seconda della piattaforma e del browser considerati, nonché delle installazioni fatte (plugins e tipi MIME).

Le principali informazioni memorizzate in Netscape sono:

- `userAgent` Mozilla/4.7 [en] (Win98; I)
- `appName` Mozilla
- `appVersion` 4.7 [en] (Win98; I)
- `appName` Netscape
- `language` en





JAVASCRIPT

- `platform` Win32
- `securityPolicy` Export policy
- `plugins` [object PluginArray]
- `mimeTypes` [object MimeTypeArray]

Le ultime due informazioni sono in realtà delle matrici che contengono tutti i dati dei plugins e dei tipi MIME installati (vedi Esempio21).



JAVASCRIPT

L'oggetto frame:

E' un oggetto che appartiene a *window*, ma che possiede molte sue proprietà e metodi.

Infatti l'unica differenza tra un *frame* e una *window* è la loro visualizzazione, mentre le funzionalità e le caratteristiche sono le stesse.



JAVASCRIPT

Riferimenti a window o a frame:

Per riferirsi a un oggetto *window* (o a uno *frame*) si può usare:

- la variabile alla quale è stato assegnato l'oggetto all'atto della sua apertura;
- *self* o *window* che si riferiscono alla finestra (o frame) attuale;
- *parent* e *top* che si riferiscono alla finestra (o frame) da cui discende l'attuale e alla prima finestra (o frame) della gerarchia, rispettivamente.



JAVASCRIPT

L'oggetto *history*:

Ogni browser memorizza nell'oggetto *history* la storia e gli URL delle pagine visitate in ogni sessione, in modo da consentire poi di muoversi tra di esse con i bottoni “indietro” e “avanti”.

Usando il metodo *go()* di *history* si possono quindi creare dei bottoni di navigazione personalizzati (es. *history.go(-1)*); per accedere alla pagina anteriore).



JAVASCRIPT

L'oggetto location:

Contiene informazioni riguardo alle proprietà della pagina caricata, tra cui il suo nome.

In particolare ha le seguenti proprietà: *protocol*, *host*, *port*, *pathname*, *hash* (contiene la parte dell'URL introdotta da # e che indica il punto da visualizzare nella pagina, es. #marker1) e *search* (contiene la parte dell'URL introdotta da ? e che indica una query fatta alla pagina richiesta, es. ?name=luca), oltre a *href* (contenente i valori concatenati di tutte le precedenti) e *hostname* (con il nome completo dell'host).





JAVASCRIPT

Inoltre vi è la proprietà *window.location* (contenente l'URL corrente) da non confondersi con la *document.location* (contenente la locazione del documento), che può avere valore diverso, e che è stata sostituita da *document.URL*. Quest'ultima può essere usata per cambiare l'URL di una finestra.

L'oggetto *location* ha due metodi:

- *reload()*, che ricarica la pagina attuale;
- *replace()*, che carica una nuova pagina che si aggiunge all'*history* sostituendovi la pagina attuale.



JAVASCRIPT

L'oggetto *document*:

Ogni pagina HTML ha un oggetto *document* che contiene tutti gli elementi della pagina, ognuno con le sue proprietà e funzioni.

Le principali proprietà di *document* sono:

- i colori della pagina HTML: *bgColor* (colore di sfondo), *fgColor* (colore di primo piano), *linkColor* (colore dei link), *alinkColor* (colore assunto dai link mentre si sta caricando il documento corrispondete, prima di assumere il colore *vlinkColor*), *vlinkColor* (colore dei link visitati).

ATTENZIONE alle maiuscole/minuscole!!!





JAVASCRIPT

- le matrici degli elementi che può contenere *document*: anchors (i punti, o markers, posti nella pagina), applets, embeds (i plugins), forms, images, links;
- altre proprietà: *cookie* (contiene i cookie associati al documento), *domain* (contiene il nome del dominio del server che ha fornito la pagina), *lastModified*, *referrer* (l'URL della pagina dalla quale è stata chiamata la pagina corrente), *title*, *URL*.

I principali metodi di *document* sono:

- *close()*, *open()*, *write()*, *writeln()*.

L'oggetto *document* non risponde a nessun evento.





JAVASCRIPT

Data di ultima modifica di una pagina HTML:

In ogni pagina HTML è consigliabile inserire sempre la data in cui è stata eseguita l'ultima modifica della pagina.

Per fare ciò si può usare la proprietà *lastModified* dell'oggetto *document* corrispondente:

```
var lastModi=document.lastModified;
```

```
document.write("Ultima modifica di questa pagina il: "+lastModi);
```

(Si veda Esempio22).



JAVASCRIPT

Gli oggetti di document:

Gli oggetti che può contenere un *document* sono:

- *links*
- *anchors*
- *images*
- *areas*
- *forms*

Agendo su questi elementi con JavaScript si può rendere dinamica e personalizzata la visualizzazione di una pagina HTML, nonché migliorare e verificare l'introduzione di dati nei form.





JAVASCRIPT

Gli oggetti links:

In ogni *document* vi è una matrice *links* con tutti il link, e rispettive proprietà, presenti nel documento.

Ogni link ha le stesse proprietà di un oggetto *location* ed in più la proprietà *target*.

Ciò permette cambiare dinamicamente, anche dopo il caricamento della pagina in cui si trova il link, il documento a cui il link si riferisce, nonché la finestra o il frame in cui il documento viene caricato. →



JAVASCRIPT

Un oggetto *link* può rispondere a tre eventi: *onClick*, *onmouseover*, *onmouseout* (Esempio23).

Interessante sapere che l'oggetto "stringa di testo" ha un metodo (*link("URL")*). Questo metodo permette creare un link a un documento, passando come parametro l'URL del documento, dove la stringa di testo costituisce il testo sottolineato del link
(es. *"Politecnico di Milano".link("http://www.polimi.it");*).



JAVASCRIPT

Gli oggetti anchors:

Tali oggetti non hanno nulla a che vedere con i link, anche se appaiono a loro simili.

Un oggetto *anchor* non ha ne proprietà ne risponde a eventi.

Pertanto in JavaScript si può solo cambiare o definirne il nome di un *anchor*, mentre la posizione che identifica dipende dalla posizione nel documento HTML in cui viene inserito.



JAVASCRIPT

Gli oggetti images:

Una delle proprietà più interessanti di un oggetto immagine di un documento HTML è *src* che indica l'URL dell'immagine caricata nel documento.

Questo URL può essere modificato dinamicamente, anche dopo il caricamento della pagina in cui si trova l'immagine. Ciò consente creare facilmente immagini animate, caricando una sequenza di immagini variabili di volta in volta (Esempio24).

Gli eventi a cui risponde un oggetto immagine sono tre: *onError*, *onAbort*, *onLoad*.





JAVASCRIPT

Gli oggetti areas:

Un'area è una zona cliccabile di una immagine e collegata ad un URL (vedi image maps).

Comportandosi quindi come un link, un oggetto *area* ha le stesse proprietà, e metodi e risponde agli stessi eventi di un *link*.



JAVASCRIPT

Gli oggetti forms:

Questi oggetti sono tra i più utilizzati in JavaScript dato che includono elementi che permettono l'interazione con l'utente di una pagina HTML e il cui contenuto e caratteristiche possono essere modificati anche dopo il caricamento della pagina in cui si trovano.

Inoltre JavaScript è spesso applicato a *form* per controllare completezza e “correttezza” dei dati introdotti dall'utente prima di inviarli (per es. a un CGI o a una pagina ASP). Si veda l'Esempio25.



JAVASCRIPT

Gli oggetti di form:

Gli oggetti che può contenere un *form* sono:

- *texts*
- *textareas*
- *passwords*
- *buttons*
- *radios*
- *checkboxs*
- *selects*
- *resets*
- *submits*





JAVASCRIPT

Gli oggetti `text`:

Sono delle zone riquadrate in una pagina HTML dove può essere inserito del testo, dall'utente o automaticamente (per es. tramite JavaScript).

Tra le proprietà di un oggetto `text`, `value` consente modificarne dinamicamente, anche dopo il caricamento della pagina in cui si trova, il testo contenuto (Esempi 26 e 27).

Quindi si dovranno usare tali oggetti (e non la funzione `write(...)`) per scrivere un testo in una pagina HTML dopo il suo caricamento.





JAVASCRIPT

I principali metodi di un oggetto *text* sono:

- *focus()*: fa sì che l'oggetto *text* cui si riferisce sia l'oggetto selezionato della pagina HTML, ovvero che il cursore sia posizionato al suo interno e l'utente possa direttamente iniziare a scrivere (es. *document.forms[0].text1.focus()*);
- *select()*: risalta il testo contenuto nell'oggetto *text* cui si riferisce, come se fosse stato selezionato con il mouse. In tal modo l'utente iniziando a scrivere sostituisce tutto il testo precedentemente presente nell'oggetto.



JAVASCRIPT

Gli oggetti *textareas*:

Sono simili agli oggetti *text* e si applica loro tutto quanto relativo agli oggetti *text*.

In più gli oggetti *textarea* possono contenere varie linee di testo e dispongono di barre di scorrimento (“scrollbar”) per visualizzarle completamente.

Tuttavia non hanno metodi per gestirne lo scorrimento in modo automatico.



JAVASCRIPT

Gli oggetti passwords:

Sono simili agli oggetti *text* tranne che il testo che vi si inserisce risulta nascosto da dei caratteri particolari.

Si applica agli oggetti *password* tutto quanto relativo agli oggetti *text*.

Gli oggetti buttons:

Sono bottoni con proprietà *name* e *value* e che rispondono all'evento *onClick* quando cliccati.



JAVASCRIPT

Gli oggetti radios e checkboxes:

Sono bottoni di scelta che hanno le stesse proprietà degli oggetti *button* e con in più la proprietà *checked* che permette selezionarli/deselezionarli automaticamente.

Più oggetti *radio* possono essere raggruppati assieme, assegnando loro lo stesso nome, per definire scelte alternative. In tal caso i vari oggetti *radio*, raggruppati assieme con lo stesso nome, costituiscono gli elementi di una matrice con quel nome (es. `<FORM Name="form"> <INPUT TYPE="radio" NAME="rad"> <INPUT TYPE="radio" NAME="rad"> </FORM>` `document.form.rad[1].checked=true;`).



JAVASCRIPT

Gli oggetti selects:

Sono menù con liste di selezione.

Hanno proprietà *name*, *size*, *value*, *selected* e *multiple*.

Le ultime due permettono selezionare/deselezionare automaticamente anche più voci della lista.

Rispondono agli eventi *onClick*, *onChange*, *onBlur* e *onFocus*.



JAVASCRIPT

Gli oggetti *reset* e *submit*:

Sono bottoni con le stesse proprietà degli oggetti *button*.

Gli oggetti *reset* rispondono all'evento *onClick* quando cliccati, mentre gli oggetti *submit* solo inviano il contenuto del form da cui dipendono, invocandone il metodo *submit()*.



JAVASCRIPT

Riferimenti agli oggetti di un documento o di un form:

Per riferirsi a un oggetto di un documento o di un form si può usare il suo nome o la sua posizione all'interno della corrispondente matrice dell'oggetto padre che contiene quel tipo di oggetti: ovvero *elements[]* per gli elementi di un form (es. *document.forms[0].elements[0].value=""*). Vedi confronto Esempi 26 e 27.



JAVASCRIPT

Riferimenti bibliografici:

<http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/>

<http://docs.iplanet.com/docs/manuals/javascript.html>

<http://www.html.it/javascript/tutorial/>

<http://script.aculo.us/> (librerie Javascript basate su AJAX)

<http://www.swishzone.com/> (clip personalizzate in formato Flash)