

Manuale JavaScript

Mosaic, il primo browser, venne rilasciato nel 1992 e permetteva di visualizzare la grafica oltre al testo; nel 1994 parte degli sviluppatori di Mosaic fondarono la Netscape Communications Corporation e il loro browser si rivelò ben presto di qualità superiore. Il 1995, inoltre, resta una pietra miliare nello sviluppo di Internet perché accanto a Netscape anche un'altra società saliva alla ribalta, grazie al previdente investimento sulle potenzialità del Web: la Sun Microsystems Inc., che aveva presentato qualche mese prima Java, il linguaggio evoluto che si proponeva di diventare uno standard nella comunicazione in rete. Il Java è un linguaggio di programmazione fortemente indipendente dalla macchina sulla quale gira, permette quindi l' utilizzo di programmi su qualsiasi tipo di terminale in rete. Il JavaScript nasce da uno sforzo comune della Netscape e della Sun per poter utilizzare un linguaggio con le stesse possibilità di Java (girare su qualsiasi piattaforma) ma con una minore difficoltà di programmazione. Questo sforzo comune diede vita a JavaScript. E' bene precisare che JavaScript è cosa molto diversa da Java. Entrambi i linguaggi sono orientati agli oggetti, ma mentre Java è usato per creare applicazioni autonome o applet, JavaScript viene interpretato insieme al codice HTML (del quale è parte integrante, e senza il quale non può esistere), senza necessità di macchine virtuali o conoscenze approfondite di modelli orientati agli oggetti. Javascript è molto semplice da imparare per chi già conosce linguaggi simili come il C++ o Java, ma non è neanche difficile per chi si avvicina per la prima volta a questo linguaggio data la sua semplicità sintattica e la sua maneggevolezza. Tuttavia ciò può rappresentare un'arma a doppio taglio perché la semplicità si gioca anche su una disponibilità limitata di oggetti per cui alcuni procedimenti, all'apparenza molto semplici, richiedono script abbastanza complessi. La caratteristica principale di Javascript, è quella di essere un linguaggio di scripting, ma soprattutto è il linguaggio di scripting per eccellenza e certamente quello più usato. Questa particolarità comporta una notevole serie di vantaggi e svantaggi secondo l'uso che se ne deve fare e tenendo in considerazione il rapporto che si instaura nel meccanismo client-server. Spiegando in parole molto semplici quest'ultimo rapporto, possiamo dire che il server invia i dati al client e questi dati possono arrivare in due diversi formati: in formato testo (o ASCII) o in formato binario (o codice macchina). Il client sa comprendere solo il formato binario (cioè la sequenza di 1 e 0), per cui se i dati arrivano in questo formato diventano immediatamente eseguibili (e purtroppo senza la possibilità di effettuare controlli), mentre se il formato è diverso devono essere interpretati e tradotti in formato binario, e quindi il client ha bisogno di un filtro o meglio di un interprete che sappia leggere questi dati e li possa tradurre in binario. I dati in formato testo sono visibili all'utente come semplici combinazioni di caratteri e di parole, quindi di facile manipolazione, ma richiedono più tempo per la loro interpretazione a causa dei passaggi e delle trasformazioni che devono subire per essere compresi dal client; i dati in formato binario, invece, sono di difficile comprensione da parte dell'utente, ma immediatamente eseguibili dal client, senza richiedere passaggi intermedi.

Effettuata questa premessa si possono suddividere i linguaggi di solito utilizzati per il Web in quattro tipologie:

- 1. HTML: è in formato testo e non è un linguaggio nel senso tradizionale, ma un impaginatore per consente di posizionare degli oggetti nella pagina con le caratteristiche indicate, naturalmente per la sua peculiarità risulta essere statico e non interagisce con l'utente e non può prendere decisioni se non per i formulari, mentre per la sua interpretazione ha bisogno di un browser;
- 2. linguaggi compilati: sono quei linguaggi abbastanza complessi in cui il sorgente (un file di testo con le operazioni da eseguire) viene compilato in codice macchina e viene impacchettato in un eseguibile utilizzabile solo nella forma e per le operazioni per cui è stato progettato;
- 3. linguaggi semicompilati: in realtà a questa classe appartiene solo Java perché è un linguaggio compilato in un formato intermedio tra il file ASCII e il file binario, tale formato si chiama bytecode e va interpretato sul client da una macchina virtuale chiamata Java Virtual Machine, in tal modo all'atto della ricezione tale macchina completa la compilazione e rende il file eseguibile;
- 4. linguaggi interpretati: sono quei linguaggi che risultano molto simili all'HTML, ma hanno potenzialità maggiori perché consentono di effettuare controlli e operazioni complesse, vengono inviati in file ASCII, quindi con codice in chiaro che viene interpretato ed eseguito riga per riga dal browser in modalità runtime.

PROGRAMMAZIONE AD OGGETTI

Per comprendere la programmazione ad oggetti (OOP) si deve considerare il programma che si sta scrivendo come se questo fosse un "MONDO AESTANTE". Come tale esso è composto da oggetti che possiedono delle loro caratteristiche. Queste caratteristiche possono a loro volta essere diverse all'interno di uno stesso insieme di oggetti. Vediamo un esempio pratico.

Prendiamo "un mondo" come esempio: l' UOMO.

Il mondo uomo è composto da diversi oggetti (che in pratica sono gli uomini).

L' oggetto uomo possiede diverse caratteristiche (capelli, occhi, gambe, etc..).

Ogni caratteristica può avere un diverso stato (i capelli possono essere biondi o neri, gli occhi possono essere aperti o chiusi o azzurri, le gambe possono essere pelose o glabre o in movimento, etc..).

L' insieme degli uomini è definito CLASSE.

Ogni singolo uomo è definito OGGETTO.

Ogni oggetto (uomo) ha delle caratteristiche chiamate PROPIETA'.

Ogni proprietà (es: capelli) può avere un diverso stato definito METODO.

ES: Se noi volessimo, con un programma OO, far camminare un uomo dovremmo scrivere:
uomo.gambe.camminano();

All'interno delle parentesi tonde potremmo mettere il numero dei passi che le gambe devono fare.

RICHIAMO DEGLI SCRIPT

In linea di principio uno script può essere inserito in due modi all'interno di pagina HTML (un'eccezione è rappresentata dagli script del server creati con LiveWire):

- 1. inserendo il codice nel documento (SCRIPT INTERNO);
- 2. caricandolo da un file esterno (SCRIPT ESTERNO).

L'uso del file esterno, infatti, è dettato dal limite di dimensione di 32K che deve rispettare una pagina web.

SCRIPT INTERNO:

Se lo script è all'interno del documento, può essere immesso sia nella sezione di intestazione (tra i tag HEAD /HEAD) sia in quella del corpo del documento (tra i tag BODY /BODY). Bisogna considerare che una variabile o qualsiasi altro elemento di Javascript può essere richiamato solo se caricato in memoria: tutto ciò che si trova nell'intestazione è quindi visibile agli altri script, quello che si trova nella sezione BODY è visibile agli script che lo seguono.

SCRIPT ESTERNO:

lo script è salvato in un file con estensione .js .Viene richiamato con l'attributo SRC del tag SCRIPT:

Il nome del file può essere indicato con un URL relativo o assoluto. Tale file esterno viene eseguito all'interno della pagina HTML, per cui lo script viene solo letto come file di testo, trasferito nell'HTML nella posizione di richiamo e qui eseguito. Il vantaggio di usare file esterni è immenso soprattutto perché apporta la caratteristica della modularità per cui uno script che ricorre di frequente (ad esempio il rollover) può essere scritto una sola volta e richiamato in qualsiasi pagina HTML quando serve, ma tutto ciò ha un prezzo: funziona solo con Netscape 3.0 ed Explorer 4.0 e nelle versioni successive.

Uno SCRIPT può anche essere eseguito:

- In seguito all'attivazione di un'evento
- In luogo di un link, nella forma A HREF="Javascript:comando"
- I valori Javascript possono essere richiamati dinamicamente dall'HTML includendoli tra i caratteri & { e };%.

IDENTIFICATORI

I nomi dei dati sono chiamati identificatori e devono sottostare ad alcune regole:

1. possono contenere solo lettere, numeri e trattino di sottolineatura, per cui sono esclusi gli spazi bianchi;
2. il primo carattere deve essere sempre una lettera. E' utilizzabile come primo carattere anche il trattino di sottolineatura, ma il compilatore tratta quel nome in modo particolare per cui se ne sconsiglia l'uso;
3. Javascript è case sensitive per cui tratta diversamente le lettere in maiuscolo e in minuscolo, per tale motivo è convenzione utilizzare l'iniziale maiuscola per i nomi di costanti e quella minuscola per le variabili;
4. non si possono utilizzare i nomi che rientrano nelle parole chiave.

- abstract
- boolean
- break
- byte
- case
- catch
- char
- class
- const
- continue
- default
- do
- double
- else
- extends
- false
- final
- finally
- float
- for
- function
- goto
- if
- implements
- import
- in
- instanceof
- int
- interface
- long
- native
- new
- null
- package
- private
- protected
- public
- return
- short
- static
- super
- switch
- synchronized
- this
- throw
- throws
- transient
- true
- try
- var
- void
- while
- with

SINTASSI

Innanzitutto, per essere leggibile nelle pagine HTML, il codice in JS deve essere inserito nei tag. Un programma è costituito da blocchi di codice o anche da singoli comandi. In JS i blocchi di codice sono racchiusi tra due parentesi graffe {...}. Le linee dei comandi sono, invece, limitate dal punto e virgola (;).

1) Altri due tipi di segnature, inseriti all'interno dei programmi in JS sono quelle riguardanti i COMMENTI:

```
// questo commento può essere lungo una riga. Non si può andare a capo.
```

```
/* All'interno di questi simboli posso disporre tutto il testo che voglio e posso anche andare a capo */
```

2) Javascript non bada agli SPAZI BIANCHI, tranne che per quelli che si trovano nelle stringhe, per cui si possono omettere o anche aumentare. Il loro uso, tuttavia, con l'indentazione aumenta la leggibilità del programma per cui sono vivamente consigliati.

3) Importanti sono gli APICI, sia singoli (' ') che doppi (" ").

I doppi apici si adoperano per racchiudere parti di codice Javascript, e, insieme a quelli singoli, a racchiudere anche le stringe. Se si desidera che in una stringa appaiano apici doppi o singoli come parte integrante della stringa stessa, si fanno precedere da una barra rovesciata (\).

4) La sintassi di una FUNZIONE è:

```
function nome(argomento)
```

```

{
  comandi
}

```

Una funzione è un blocco di comandi. Questi comandi sono racchiusi tra le parentesi graffe. L'argomento è un valore che può essere dato da analizzare alla funzione.

ES:

```

function esempio(n)
{
  var out = new Array();
  out[0]="uno";
  out[1]="due";
  out[2]="tre";
  alert("Hai schiacciato il pulsante -Argomento " + out[n] + "-");
}

```

5)La sintassi per dichiarare una VARIABILE è:

```
var nomeVariabile = valore/espressione
```

Le variabili sono dei nomi simbolici che servono ad individuare delle locazioni di memoria in cui possono essere posti dei valori. La parte "valore\espressione" è opzionale.). Questo comando non ha altro scopo che dichiarare la creazione di una nuova variabile e, se necessario, di darle come valore una stringa, un numero o una espressione. Le variabili dichiarate al di fuori della funzione (variabili GLOBALI) sono accessibili all'intera applicazione. Le variabili inserite all'interno delle funzioni (variabili LOCALI) saranno visibili solo all'interno della funzione nella quale sono state immesse. I due differenti tipi sono generati da esigenze diverse in quanto le variabili locali hanno una vita brevissima ed esistono finché opera la funzione, alla chiusura della parentesi vengono distrutte liberando memoria, mentre le variabili globali sono dei contenitori che durano per tutta la durata della pagina e servono a veicolare dei valori tra le funzioni e tra gli script, ma anche tra le varie pagine o al server.

ES:

```

var pippo = "esterna";
function es5_2( )
{
  var pippo="interna";
  alert("La funzione stampata è la funzione\n" + pippo);
}

```

Se mandata in esecuzione la funzione stampata sarà "interna"

Se invece avessimo scritto `alert("La funzione stampata è la funzione\n" +this.pippo);` allora la funzione stampata sarebbe stata "esterna", questo perché il suffisso `this` richiama la proprietà dell'oggetto del programma e non della funzione.

I linguaggi tradizionali (come il C) utilizzano solo variabili che siano state precedentemente dichiarate, per due motivi:

1. ottimizzare l'uso della memoria;
2. aumentare l'affidabilità.

Javascript, invece, utilizza un controllo di tipo lasco per cui non esiste una sezione di dichiarazione di variabili e non c'è bisogno di farlo, ma automaticamente viene assegnato il tipo in base alla dichiarazione. Ad esempio `Prova="testo"` e `Prova=59` sono due dichiarazioni pienamente valide, ma nel primo caso `Prova` è considerata oggetto string, nel secondo la stessa è considerata come valore numerico.

Se valori diversi vengono concatenati prevale il valore string, per cui un numero concatenato ad una stringa produce una stringa, tuttavia se la stringa è un numero, il risultato sarà un numero (es.: `temval=365+"10"` darà 375).

Per convertire i valori basta utilizzare:

- 1. gli apici per convertire un valore numerico in stringa o la somma di un numero con uno spazio;
- 2. metodo String() (da Javascript 1.2) per convertire in stringa o con il costruttore String () (es: miastringa=String(num));
- 3. le funzioni eval(), parseInt() e parseFloat() per convertire un valore stringa in numerico.
 - a)parseInt cerca un intero all'inizio di una stringa, scartando le stringhe, e lo visualizza, ignorando le parti decimali e l'eventuale virgola (es.: parseInt("39 gradi")=39), un secondo parametro, facoltativo, è la base numerica(es:parseInt(text,16) per cercare numeri esadecimali)
 - b)parseFloat opera allo stesso modo, ma conserva l'eventuale virgola presente e il segno.
 - c)La funzione eval è abbastanza complessa e cerca qualsiasi espressione Javascript valida per trasformarla in numerico. In Netscape 2.0, però, provoca cadute improvvise: x=10; y=20; z=30; eval ("x+y+z+900")=960
- 4. i valori logici sono ottenuti da valori numerici o stringa ponendo questi uguale a true o false.
In qualsiasi momento, comunque, si può verificare il tipo della variabile tramite l'operatore typeof (es.: typeof 69 restituisce "number") il quale è stato introdotto con Javascript 1.1. I valori restituiti sono: string, boolean, number, function.

6)Caratteri speciali

Tra le stringhe occorre indicare i caratteri speciali che consentono di formattare il testo:

Descrizione	Designazione standard	Sequenza
Continuazione	< newlin >	\
Nuova riga	NL (LF)	\n
Tab orizzontale	HT	\t
Backspace	BS	\b
Ritorno carrello	CR	\r
Avanzamento pagina	FF	\f
Backslash	\	\\
Virgolette singole	'	\'
Virgolette doppie	"	\"

7)Escape ed Unescape

Le stringhe possono essere interpretate sia come sequenze di caratteri o come righe di comando. Un campo di applicazione, ad esempio, è nel trattamento dei dati inviati al server, o anche al client, con il metodo GET. In questo caso lo script riceve i dati come una stringa attaccata (appended) all'URL originale in cui alcune sequenze di caratteri indicano vanno interpretati come caratteri particolari come lo spazio vuoto o il separatore tra variabile e valore. Ad esempio se effettuate una ricerca su Altavista vedrete aggiungersi all'URL una serie di caratteri che segue un punto interrogativo, ebbene, quei caratteri opportunamente trattati, formano una stringa che interroga il motore di ricerca. I caratteri scritti in quella notazione si chiamano sequenze escape e utilizzano la codifica URL in cui:

1. le coppie nome=valore sono separate da una &;
2. gli spazi sono sostituiti da +;
3. i caratteri alfanumerici sostituiti dall'equivalente esadecimale preceduto da %.

Parte del lavoro dei programmi CGI che funzionano su server è proprio quello di decifrare la stringa di input in caratteri ISO-Latin-1, e non le sequenze Unicode, ma Javascript può elaborare queste stringhe anche all'interno dei propri script mediante i comandi escape:

```
escape("Ecco qui")="Ecco%20qui"
```

o con unescape per rendere la situazione contraria:

```
unescape("Ecco%20qui")="Ecco qui"
```

Opportunamente utilizzati, queste funzioni si rivelano utilissime e offrono all'utente un grande grado di interattività.

GLI OPERATORI

Gli operatori si dividono in:

- 1. operatori di assegnazione
 - 2. operatori aritmetici
 - 3. operatori relazionali
 - 4. operatori logici
 - 5. operatori sui bit (adoperati in genere solo per generare colori)
 - 6. operatori su stringhe
- 1) Operatori di ASSEGNAZIONE.
 - = assegna alla variabile a sinistra il valore dell'espressione di destra $a=b$
 - += somma al valore della variabile a sinistra il valore dell'espressione di destra $a+=b$ equivale ad $a=a+b$
 - -= sottrae al valore della variabile a sinistra il valore dell'espressione di destra
 - *= assegna alla variabile a sinistra il prodotto del valore della variabile stessa per l'espressione di destra $a*=b$ equivale a $a=a*b$
 - /= assegna alla variabile a sinistra il quoziente del valore della variabile stessa per l'espressione di destra
 - %= assegna alla variabile a sinistra il modulo della divisione della variabile stessa per l'espressione di destra

- 2) Gli operatori ARITMETICI sono binari e unari.

Gli operatori unari matematici possono essere posti prima (prefissi) o dopo (posfissi) dell'operando e il loro valore varia secondo questa posizione in quanto l'operatore prefisso modifica l'operando prima di utilizzarne il valore, mentre l'operatore posfisso modifica l'operando dopo averne utilizzato il valore e sono:

OPERATORE	SIMBOLO	AZIONE
Incremento	++	Incrementa di un'unità
Decremento	--	Decrementa di un'unità
Meno unario	-	Rende negativo un numero

Gli operatori binari matematici non cambiano il valore degli operandi, ma memorizzano il risultato in un terzo operando, comprendono le principali operazioni aritmetiche. Non è possibile, però, utilizzare l'operatore di calcolo del modulo quando gli operandi sono in virgola mobile. L'operatore di divisione, invece, applicato a variabili di tipo intero produce un risultato troncato della parte decimale. Se si divide il modulo per zero, si avrà un'eccezione nell'esecuzione: se l'operazione eccede il limite inferiore (underflow) il risultato sarà zero, se eccede il limite superiore (overflow) si avrà un'approssimazione.

OPERATORE	SIMBOLO	AZIONE
Addizione	+	Somma due operandi
Sottrazione	-	Sottrae il secondo operando dal primo
Moltiplicazione	*	Moltiplica i due operandi
Divisione	/	Divide il primo operando per il secondo
Resto (modulo)	%	Fornisce il resto della divisione tra due operandi

- 3) Con operatore RELAZIONALE (o operatori di confronto) si intende la relazione che un valore ha rispetto ad un altro. Essi si basano sul concetto di verità o di falsità e comunque operano con solo due stati diversi (0/1, acceso/spento, vero/falso). Gli operatori relazionali, come detto, producono 1 se la relazione è vera, e 0 se la relazione è falsa. È importante capire che l'output è dato solo da due valori per evitare confusioni tra

l'operatore di assegnamento e quello di uguaglianza.

OPERATORE	SIMBOLO
>	Maggiore di
>=	Maggiore o uguale
<	Minore di
<=	Minore o uguale
==	Uguale
!=	Diverso

- 4) Gli operatori LOGICI assomigliano tantissimo a quelli relazionali in quanto danno in output solo due valori, 0 se l'espressione logica è vera, 1 se l'espressione è falsa. Gli operatori logici sono (il NOT è un operatore unario):

OPERATORE	SIMBOLO	SIGNIFICATO
AND	&	AND su bit
OR		OR su bit
AND	&&	AND logico
OR		OR logico
XOR	^	XOR su bit
NOT	!	Not logico

Se noi imponiamo due condizioni perchè si avveri un ciclo e le congiungiamo con && (condiz.1 && condiz.2), il ciclo verrà eseguito ma solo se tutte e due le condizioni sono verificate. Nel caso in cui venga utilizzato || (condiz.1 || condiz.2), il ciclo verrà effettuato anche se una solo delle condizioni è vera.

- 5) Gli operatori su STRINGHE sono:

OPERATORE	NOME	SINTASSI
+	Addizione	stringa=stringaA+stringaB
+=	Accoda	stringa=stringaA+="grassa"
==	Uguaglianza	if (stringaA==stringaB)
!=	Disuguaglianza	if (stringaA!=stringaB)

- Precedenza tra gli operatori

Dalla priorità più alta alla più bassa:

- 1) Funzione o indice di matrice () []
- 2) Negazione (!, ~, -), incremento (++), decremento (--)
- 3) Moltiplicazione, divisione e modulo (*, /, %)
- 4) Addizione e sottrazione (+, -)
- 5) Operatore bit (>>, <<, >>>)
- 6) Confronto (<, >, <=, >=)
- 7) Uguaglianza e disuguaglianza (==, !=)
- 8) Operatore relativo ai bit AND (&)
- 9) Operatore relativo ai bit XOR (^)
- 10) Operatore relativo ai bit OR (|)
- 11) Operatore logico AND (&&)
- 12) Operatore logico OR (||)
- 13) Operatori condizionali (?, :)
- 14) Operatori di assegnazione (=)
- 15) Virgola (,)

CICLI

Diversamente dai linguaggi come il Basic, i linguaggi OOP aborriscono i rimandi di tipo GOTO o GOSUB. Nasce all'ora l'esigenza di fornire il programma di cicli. I cicli sono dei blocchi di programma che si ripetono fino a che una determinata condizione non si verifica.

Esistono diversi tipi di cicli:

- 1) Il ciclo IF....ELSE
La dicitura per questo tipo di ciclo è:

```

if (condizione)
{
    comandi da applicare se la condizione è vera
}
else
{
    comandi da applicare se la condizione è falsa
}

```

L' else non è sempre necessario; se non ci sono soluzioni rimanenti infatti si può omettere o, se per esempio il numero delle soluzioni è definito e piccolo, ognuna di queste può essere coperta da un semplice if. Vedi ESEMPIO Funzione saluto1().

Come si può notare non ci sono nemmeno le graffe;. Queste, infatti, non sono necessarie, in questo caso, perchè se la condizione risulta vera, deve essere eseguito un solo comando e non un blocco di comandi. In questo secondo caso sarebbero allora necessarie le "{ }" .

- 2)Il ciclo con l' OPERATORE TERNARIO ?

Il simbolo ? viene utilizzato come un if.

La sua sintassi è la seguente:

```
(Espressione1) ? (Espressione2) : (Espressione3)
```

Per cui se è vera la prima espressione viene eseguita la seconda, se falsa viene eseguita la terza.

- 3)Il ciclo WHILE

La sintassi di questo comando è:

```

while (condizione)
{
    comandi
}

```

Ovvero finchè condizione è vera esegui i comandi.Quando la condizione diventa falsa allora passa al comando successivo.

- 4)Il ciclo DO...WHILE

Se occorre eseguire il blocco di istruzioni almeno una volta.

La sua sintassi è:

```

do
{
    istruzioni
}
while (condizione)

```

- 5)Il ciclo FOR

La sintassi di questo ciclo è la seguente:

```

for(inizializzazione; condizione; operazione-fase-ciclo)
{
    comandi
}

```

Esegue una serie di comandi fino a che non è stato raggiunto il limite indicato da una

condizione. La prima espressione dice da dove inizializzare la variabile contatore, la seconda espressione pone il limite condizionale entro il quale l'istruzione deve essere reiterata, mentre l'ultima espressione dice al loop di quanto deve incrementare o decrementare la variabile. Ad esempio molto utile e' il ciclo che serve ad inizializzare un array (nel nostro caso prendiamo un array di 10 elementi):

```
for (i=0; i<10; i++)
{
    matrice[i]=0;
}
```

ricordiamo che le matrici si iniziano a contare da 0 per cui quando il contatore assume valore 10 il ciclo non è ripetuto.

- 6) Il ciclo SWITCH.....CASE

La sua sintassi è:

```
switch (espressione)
{
    case costante1:
        istruzioni;
        break;

    case costante2:
        istruzioni;
        break;

    ....

    default:
        istruzioni
}
```

Il valore dell'espressione viene confrontato con i diversi valori dei case e quando viene trovata corrispondenza, si eseguono le sequenze di istruzioni associate, anche se al case è associato uno statement vuoto oppure un'ulteriore switch. L'istruzione di default è opzionale e viene eseguita solo se non è trovata corrispondenza con nessuno dei case. L'istruzione break consente al programma di uscire dal ciclo di switch; se fosse mancante, il programma continuerebbe a confrontare il valore.

EVENTI

Gli eventi sono utilizzati per richiamare delle istruzioni. Infatti lo script è eseguito in maniera sequenziale, ma per consentire la DINAMICITA' e l' ITERATTIVITA' occorre che questo resti caricato in memoria e venga attivato o richiamato solo quando si verificano particolari situazioni (come il passaggio del mouse, il caricamento di un documento, ecc). Ad un evento può essere associata una sola istruzione o un blocco di istruzioni, racchiusi in una funzione. Tale funzione sarà poi richiamata da parole chiave dette HANDLER (o gestore di evento) composte dal nome dell' evento preceduto dal suffisso - on -. (l'handler che richiama l' evento click è onClick) Gli eventi, per poter interfacciare HTML con Javascript, non vengono definiti nel tag SCRIPT (tranne che in qualche caso), ma sono richiamati all'interno dei tag HTML (ogni tag supporta determinati eventi).

Vediamo di effettuare un raggruppamento:

- 1) Eventi attivabili dai TASTI DEL MOUSE

A questo gruppo si possono ricondurre i seguenti eventi:

- 1. onClick: attivato quando si clicca su un oggetto
- 2. onDbClick: attivato con un doppio click (NON SUPPORTATO IN MAC)

- 3. `onMouseDown`: attivato quando si schiaccia il tasto sinistro del mouse
 - 4. `onMouseUp`: attivato quando si alza il tasto sinistro del mouse precedentemente schiacciato
 - 5. `onContextMenu`: attivato quando si clicca il tasto destro del mouse aprendo il `ContextMenu`
- 2) Eventi attivati dai MOVIMENTI DEL MOUSE
A questo gruppo si possono ricondurre i seguenti eventi:
 - 1. `onMouseOver`: attivato quando il mouse si muove su un oggetto
 - 2. `onMouseOut`: attivato quando il mouse si sposta da un oggetto
 - 3. `onMouseMove`: attivato quando si muove il puntatore del mouse

Gli eventi `onMouseOver` e `onMouseOut` sono complementari in quanto il primo è attivato nel momento in cui il puntatore è posto nell'area dell'oggetto il cui tag contiene l'evento e il secondo quando ne esce. Importantissimo l'evento `onMouseOver` abbinato ad `onMouseOut` per creare l'effetto `RollOver`. La sintassi è molto semplice:

```
< A HREF="#" onmouseover="document.images[num].src='immagine.gif'"
onmouseout=document.images[num].src='immagine1.gif'"
```

- 3) Eventi attivati dal TRASCINAMENTO DEL MOUSE
A questo gruppo si possono ricondurre i seguenti eventi:
 - 1. `onDragDrop`: evento attivato quando un utente trascina un oggetto sulla finestra del browser o quando rilascia un file sulla stessa
 - 2. `onMove`: attivato quando un oggetto muove una finestra o un frame
 - 3. `onDragStart`: evento attivato appena l'utente inizia a trascinare un oggetto
 - 4. `onDrag`: attivato quando il mouse trascina un oggetto o una selezione di testo nella finestra dello stesso browser o anche di un altro o anche sul Desktop;
 - 5. `onDragEnter`: attivato appena l'utente trascina un oggetto su un obiettivo valido dello stesso o di un altro browser
 - 6. `onDragOver`: attivato quando l'utente l'utente trascina un oggetto su un obiettivo valido ad ospitarlo, ed è simile all'evento precedente, ma viene attivato dopo quello
 - 7. `onDragLeave`: attivato quando l'utente trascina un oggetto su un obiettivo adatto per ospitarlo, ma non vi viene rilasciato
 - 8. `onDragEnd`: attivato quando l'utente rilascia l'oggetto al termine del trascinamento
 - 9. `onDrop`: attivato quando il mouse si alza il tasto del mouse in seguito ad un'operazione di trascinamento

Dal 3 al 9 funzionano solo con IE5 e seguenti.

- 4) Eventi legati alla TASTIERA
A questo gruppo si possono ricondurre i seguenti eventi:
 - 1. `onKeyPress`: evento attivato quando un utente preme e rilascia un tasto o anche quando lo tiene premuto
 - 2. `onKeyDown`: attivato quando viene premuto il tasto (con alcuni tasti non funziona)
 - 3. `onKeyUp`: evento attivato quando un tasto, che era stato premuto, viene rilasciato
 - 4. `onHelp`: attivato quando si schiacci il pulsante F1
- 5) Eventi legati alle MODIFICHE DELL'UTENTE
A questo gruppo si possono ricondurre i seguenti eventi:
 - 1. `onChange`: attivato quando il contenuto di un campo di un form o modulo è modificato o non è più selezionato. Questo evento, infatti, è attivato quando viene selezionato un altro elemento da una lista o quando si modifica un campo di testo,

- per cui oltre all'attivazione, occorre anche operare un'azione
- 2. `onCellChange`: attivato quando si modifica un elemento in un Database, per questa sua caratteristica ha un uso non propriamente legato a Javascript
 - 3. `onPropertyChange`: evento attivato quando cambia la proprietà di un elemento
 - 4. `onReadyStateChange`: evento attivato quando lo stato del caricamento di un elemento cambia, l'evento è utile, ad esempio, per verificare che un elemento sia stato caricato.
- 6)Eventi legati al FUOCO

A questo gruppo si possono ricondurre i seguenti eventi:

 - 1. `onFocus`: Questo handler è l'opposto di `onBlur` per cui si attiva quando l'utente entra in un campo
 - 2. `onBlur`: attivato quando il puntatore del mouse o il cursore esce dalla finestra corrente utilizzando il mouse o il carattere TAB. Applicato ai moduli, invece, tale handler si avvia se si esce dal campo il cui tag contiene il controllo
 - 3. `onSelect`: attivabile quando si seleziona del testo all'interno di una casella di testo sia col mouse sia tenendo premuto SHIFT e selezionando con i tasti Freccia. Questo gestore è usato con il tag TEXTAREA e INPUT di tipo TEXT
 - 4. `onSelectStart`: si attiva quando si inizia a selezionare un evento
 - 5. `onbeforeEditFocus`: si attiva con un doppio click o con un click su un oggetto che ha già la selezione, quando questo è in DesignMode
 - 6. `onLoseCapture`: si attiva quando un oggetto perde la cattura del mouse

Gli ultimi tre solo in IE5 e seguenti.
 - 7)Eventi attivati dal CARICAMENTO DEGLI OGGETTI

A questo gruppo si possono ricondurre i seguenti eventi:

 - 1. `onLoad`: Questo handler funziona nel caricamento di oggetti, per lo più finestre e immagini
 - 2. `onUnload`: è l'opposto del precedente e funziona quando si lascia una finestra per caricarne un'altra o anche per ricaricare la stessa (col tasto refresh)
 - 3. `onAbort`: funziona quando l'utente ferma il caricamento di un oggetto cliccando su un altro link o premendo il tasto stop del browser
 - 4. `onError`: si attiva quando il caricamento di un oggetto causa un errore, ma solo se questo è dovuto ad un errore di sintassi del codice e non del browser così funziona su un link errato di un'immagine della pagina, ma non su un link errato di caricamento di una pagina intera
 - 5. `onBeforeUnload`: questo handler funziona allo stesso modo di `onUnload` ma si carica in un momento prima
 - 6. `onStop`: questo handler funziona quando si ferma il caricamento della pagina con il tasto stop del browser e dovrebbe funzionare anche allo stesso modo di `onUnload` caricandosi prima di questo ma dopo `onBeforeUnload`
 - 8)Eventi attivati dal MOVIMENTO DELLE FINESTRE

A questo gruppo si possono ricondurre due soli eventi:

 - 1. `onResize`: Questo handler si attiva quando l'utente rimpicciolisce o ingrandisce una finestra o un frame o, in caso particolare per Explorer, un oggetto a cui siano stati fissati l'altezza e la larghezza o anche la posizione, come ad esempio un layer
 - 2. `onScroll`: attivato quando si effettua lo scrolling della pagina sia col mouse con i tasti PGUP e PGDOWN o anche con il metodo `doScroll`

CLASSIFICHIAMO GLI EVENTI IN BASE ALL'OGGETTO

Gli eventi fanno riferimento solo ed esclusivamente all'oggetto WINDOW e i suoi due sotto

oggetti DOCUMENT e FRAME. Non esistono eventi che fanno riferimento agli altri oggetti navigator - date - math - string e history - location.

- 1) Vediamo gli eventi che si riferiscono direttamente all'oggetto WINDOW:

WINDOW

- onBeforeUnload=""
- onBlur=""
- onError=""
- onFocus=""
- onHelp=""
- onLoad=""
- onResize=""
- onScroll=""
- onUnload=""

- 1a) Vediamo gli eventi che si riferiscono direttamente all'oggetto DOCUMENT (sottoggetto di window):

DOCUMENT

- onAfterUpdate=""
- onBeforeUpdate=""
- onClick=""
- onDbClick=""
- onDragStart=""
- onErrorUpdate=""
- onHelp=""
- onKeyDown=""
- onKeyPress=""
- onKeyUp=""
- onMouseDown=""
- onMouseMove=""
- onMouseOut=""
- onMouseOver=""
- onMouseUp=""
- onReadyStateChange=""
- onRowEnter=""
- onRowExit=""
- onSelectStart=""

- 2a) Vediamo gli eventi che si riferiscono all'oggetto FORM (sottoggetto di document):

FORM

- onAfterUpdate=""
- onBeforeUpdate=""
- onClick=""
- onDataAvailable=""
- onDataSetChanged=""
- onDataSetComplete=""
- onDbClick=""
- onDragStart=""
- onErrorUpdate=""
- onFilterChange=""
- onHelp=""
- onKeyDown=""
- onKeyPress=""

- onKeyUp=""
- onMouseDown=""
- onMouseMove=""
- onMouseOut=""
- onMouseOver=""
- onMouseUp=""
- onReset=""
- onRowEnter=""
- onRowExit=""
- onSelectStart=""
- onSubmit=""

- 2b) Vediamo gli eventi che si riferiscono all'oggetto IMAGE (sottogetto di document):

IMAGE

- onAbort=""
- onAfterUpdate=""
- onBeforeUpdate=""
- onBlur=""
- onClick=""
- onDataAvailable=""
- onDataSetChanged=""
- onDataSetComplete=""
- onDbClick=""
- onDragStart=""
- onErrorUpdate=""
- onFilterChange=""
- onFocus=""
- onHelp=""
- onKeyDown=""
- onKeyPress=""
- onKeyUp=""
- onLoad=""
- onMouseDown=""
- onMouseMove=""
- onMouseOut=""
- onMouseOver=""
- onMouseUp=""
- onReadyStateChange=""
- onResize=""
- onRowEnter=""
- onRowExit=""
- onSelectStart=""

- 2c) Vediamo gli eventi che si riferiscono all'oggetto LINK (sottogetto di document):

LINK

- onAfterUpdate=""
- onBeforeUpdate=""
- onClick=""
- onDataAvailable=""
- onDataSetChanged=""
- onDataSetComplete=""
- onDbClick=""
- onDragStart=""

- onError=""
- onErrorUpdate=""
- onFilterChange=""
- onHelp=""
- onKeyDown=""
- onKeyPress=""
- onKeyUp=""
- onLoad=""
- onMouseDown=""
- onMouseMove=""
- onMouseOut=""
- onMouseOver=""
- onMouseUp=""
- onReadyStateChange=""
- onRowEnter=""
- onRowExit=""
- onSelectStart=""

- 1b) Vediamo gli eventi che si riferiscono all'oggetto FRAME (sottogetto di window):

FRAME

- onAfterUpdate=""
- onBeforeUpdate=""
- onBlur=""
- onClick=""
- onDataAvailable=""
- onDataSetChanged=""
- onDataSetComplete=""
- onDbClick=""
- onDragStart=""
- onErrorUpdate=""
- onFilterChange=""
- onFocus=""
- onHelp=""
- onKeyDown=""
- onKeyPress=""
- onKeyUp=""
- onLoad=""
- onMouseDown=""
- onMouseMove=""
- onMouseOut=""
- onMouseOver=""
- onMouseUp=""
- onResize=""
- onRowEnter=""
- onRowExit=""
- onSelectStart=""

OGGETTI - PROPRIETA' - METODI

- 1) OGGETTO **WINDOW**

L'oggetto window è l'oggetto di grado più alto, visto che ogni documento per esistere deve essere inserito in una finestra (Diciamo che anche l' oggetto NAVIGATOR è

considerabile di grado pari a WINDOW, poi vedremo in che senso). All'interno dell'oggetto WINDOW troviamo altri 4 oggetti che sono: HISTORY - LOCATION - DOCUMENT - FRAME . Iniziamo col vedere le proprietà e i metodi direttamente legati all'oggetto window.

PROPRIETA'

window.defaultStatus
 window.length
 window.name
 window.opener
 window.parent
 window.self
 window.status
 window.top
 window.frames[]

METODI

window.alert("")
 window.blur()
 window.clearTimeout()
 window.close()
 window.confirm("")
 window.focus()
 window.open("", "")
 window.prompt("", "")
 window.scroll()
 window.setTimeout()

■ 1a)OGGETTO **HISTORY**(sottoggetto di window)

E' sostanzialmente un array i cui elementi contengono le URL visitate in quella finestra del browser. Gli elementi sono creati nell'ordine di accesso, per cui history [0] contiene la prima URL visitata e history[history.lenght] contiene l'ultima. Vediamo le proprietà e i metodi dell'oggetto history.

PROPRIETA'

history.length

METODI

history.back()
 history.forward()
 history.go()

■ 1b)OGGETTO **LOCATION**(sottoggetto di window)

Contiene informazioni sulla corrente URL completa. E' un oggetto predefinito che costituisce una proprietà dell'oggetto window cui si riferisce, ed è accessibile tramite tale proprietà della window. Se non è referenziata la window di appartenenza, location si riferisce alla corrente finestra. Tuttavia nei gestori di eventi occorre referenziarlo con la finestra relativa, altrimenti si riferirà alla proprietà URL dell'oggetto document corrente. Vediamo le proprietà e i metodi dell'oggetto location.

PROPRIETA'

location.hash
 location.host
 location.hostname
 location.href
 location.pathname
 location.port
 location.protocol
 location.search

METODI

```
location.reload()
window.location.replace("")
```

- 1c) OGGETTO **DOCUMENT** (sottoggetto di window)

Contiene informazioni sul documento, cioè su quella parte della finestra o del frame che contiene il documento HTML. La maggior parte delle proprietà di questo oggetto sono dei riflessi del linguaggio HTML. All'interno dell'oggetto DOCUMENT troviamo altri 3 oggetti che sono: FORM - LINK - IMAGE. Vediamo le proprietà e i metodi direttamente legati all'oggetto document.

PROPRIETA'

```
document.alinkColor
document.linkColor
document.vlinkColor
document.bgColor
document.fgColor
document.cookie
document.embeds
document.lastModified
document.location
document.referrer
document.title
document.anchors[]
document.forms[]
document.links[]
```

METODI

```
document.clear()
document.close()
document.open()
document.write("")
document.writeln("")
```

- 2a) OGGETTO **FORM** (sottoggetto di document)

Consente all'utente di immettere testo, effettuare scelte tramite checkbox, bottoni radio e listbox ed inviarli per l'elaborazione ad uno javascript ovvero ad un server remoto. L'oggetto è creato dal TAG html FORM. Gli oggetti form sono accessibili dall'oggetto *document* o tramite la proprietà *forms* dell'oggetto *document* o tramite le proprietà dell'oggetto *document* che hanno lo stesso nome delle *forms* stesse. Ogni oggetto form ha i propri elementi, creati con il codice html. A tali elementi si può accedere o con il relativo nome, o usando la proprietà *elements* di form (un array in cui gli elementi prendono posto secondo l'ordine di dichiarazione).

Vediamo le proprietà e i metodi dell'oggetto form.

```
document.form.action
document.form.encoding
document.form.length
document.form.method
document.form.target
document.form.elements[]
document.form.reset()
document.form.submit()
```

- 2b) OGGETTO **LINK** (sottoggetto di document)

Parte di testo, immagine o area identificati come link di ipertesto. In html è creato con i TAG A HREF o AREA. In javascript può essere creato con il metodo `link` dell'oggetto `string` `miastringa.link(attributoHREF)`.

Ogni oggetto link è un oggetto location. Dal momento che il contenuto è un

riferimento ad una URL. Per cui il link, oltre che ad una URL del WEB può puntare ad un'azione javascript, se l'attributo HREF ha per destinazione un'entità javascript. Se si desidera creare un link che non punta a nulla dare come attributo HREF una funzione javascript vuota: < A HREF="javascript:void(0)" >.

Vediamo le proprietà e i metodi dell'oggetto link.

```
document.link.hash
document.link.host
document.link.hostname
document.link.href
document.link.pathname
document.link.port
document.link.protocol
document.link.search
document.link.target
```

○ 2c) OGGETTO **IMAGE**(sottoggetto di document)

E' un'immagine sul documento HTML. L'oggetto è creato con il TAG html IMG. Ma è possibile creare un oggetto Image con il costruttore

```
miaFigura = new Image( [larghezza] [, altezza])
```

dove larghezza e altezza sono opzionali valori in pixel. La creazione di un oggetto con javascript è utile per caricare e decodificare un'immagine prima che debba essere visualizzata. Quindi occorre settare la proprietà src dell'immagine posizionata con la corrispondente dell'oggetto creato (miodocumento.images[indice].src = miaFigura.src).

In tale modo l'immagine vienesi ricaricata, ma dalla cache del browser. E' anche possibile creare gestori di eventi per ciascun evento dell'oggetto, assegnando alla proprietà correlata all'evento una funzione di gestione degli eventi (abort, error, keydown, keypress, keyup, onload).

Esempio: miaFigura.onabort =funzioneGestore

Gli eventi click e MouseOut e MouseOver non sono gestiti dall'oggetto Image.

Ma è possibile identificare l'immagine come link (con i TAG html A HREF) o definire un oggetto area per l'immagine. Posizione e dimensione dell'immagine sono determinate dai TAG html e non possono essere modificate con javascript (le proprietà height e width sono di sola lettura).

Vediamo le proprietà e i metodi dell'oggetto image.

```
border
complete
height
hspace
lowsrc
name
prototype
src
vspace
width
```

■ 1d) OGGETTO **FRAME**(sottoggetto di window)

L'oggetto frame viene trattato da javascript come un oggetto window dipendente dalla finestra che lo contiene. Pertanto ha tutte le proprietà e i metodi dell'oggetto window. Vediamo le proprietà e i metodi direttamente legati all'oggetto frame.

PROPRIETA'
METODI

---FINE OGGETTO WINDOW---

---INIZIO OGGETTO NAVIGATOR---

• 2)OGGETTO **NAVIGATOR**

E' un oggetto creato dal motore javascript che dà informazioni sulla versione del browser in uso. Le proprietà di navigator sono di sola lettura e consentono di conoscere la versione del browser, i tipi MIME che possono essere trattati, i plug-in installati. Vediamo le proprietà e i metodi legati all'oggetto navigator.

PROPRIETA' E METODI

```
navigator.appCodeName
navigator.appName
navigator.appVersion
navigator.mimeTypes
navigator.plugins
navigator.userAgent
navigator.javaEnabled()
```

PLUGINS

```
description
filename
length
name
```

---FINE OGGETTO NAVIGATOR---

OGGETTI INTERNI

Gli oggetti interni del NN sono utilizzati la maggior parte delle volte per creare delle istanze dell' oggetto stesso.(Appartengono a questa categoria anche: Array() e Object())

---INIZIO OGGETTO DATE---

• 3)OGGETTO **DATE**

Consente di lavorare con date e orari.

Sintassi

```
var miaData = new Date()
```

crea un oggetto con la data e orario attuali ricavati dall'orologio del computer locale

```
var miaData = new Date(millisecondi)
```

crea un oggetto con la data e tempo espressi in millisecondi a partire dal 1° gennaio 1970 (tempo UTC)

```
var miaData = new Date(stringa)
```

crea un oggetto secondo le indicazioni contenute in stringa. La stringa dev'essere in un formato riconosciuto dal metodo date.parse()

```
var miaData = new Date(anno, mese, giorno [, ora [, minuti [, secondi ]]])
```

crea un oggetto con data e tempo secondo i parametri numerici passati. Anno deve essere un numero completo (1999 e non 99): mese è un numero tra zero ed 11; giorno un numero tra 1 e 31. Ora, minuti, secondi sono numeri rispettivamente tra 0 e 23, 0 e 59, 0 e 59; se omessi vengono impostati a zero.

L'oggetto Date è internamente rappresentato con un numero intero che rappresenta i millisecondi rispetto a mezzanotte del 1° gennaio 1970. Il range di anni esprimibili dall'oggetto è compreso circa 274.000 anni dal 1° genn. 1970. Nelle versioni javascript dalla 1.3 il numero, se negativo, può esprimere una data precedente al 1 gennaio 1970. In quelle precedenti non è possibile.

PROPRIETA' E METODI

```
prototype
getDate()
getDay()
getHours()
getMinutes()
getMonth()
getSeconds()
```

```

getTime()
getTimezoneOffset()
getFullYear()
parse("")
setDate()
setHours()
setMinutes()
setMonth()
setSeconds()
setTime()
setYear()
toGMTString()
toLocaleString()
UTC() ---FINE OGGETTO DATE---

```

```
---INIZIO OGGETTO MATH---
```

- 4)OGGETTO **MATH**

È un oggetto che fornisce costanti e funzioni matematiche e può essere solo usato. Non è possibile creare nuovi oggetti. Va chiamato direttamente, senza referenza ad oggetti. Ad es. :Math.abs(100).

```

PROPRIETA'
Math.E
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
Math.PI
Math.SQRT1_2
Math.SQRT2

```

```

METODI
Math.abs()
Math.acos()
Math.asin()
Math.atan()
Math.ceil()
Math.cos()
Math.exp()
Math.floor()
Math.log()
Math.max()
Math.min()
Math.pow()
Math.random()
Math.round()
Math.sin()
Math.sqrt()
Math.tan() ---FINE OGGETTO MATH---

```

```
---INIZIO OGGETTO STRING---
```

- 5)OGGETTO **STRING**

Fornisce una serie di metodi per trattare stringhe di testo. Sintassi:

```
var miastringa = new String(valoreStringa)
```

L'oggetto String è concettualmente diverso dalla stringa creata direttamente con un valore letterale, come ad esempio var miastringa ="Ma che bella giornata". Tuttavia i metodi e la proprietà dell'oggetto String possono essere applicati ad una stringa creata con notazione letterale, in quanto questa viene trasformata in un oggetto temporaneo di

tipo String che poi viene scartato una volta applicato il metodo. Per cui difficilmente è necessario creare specificamente un oggetto di tipo String; verosimilmente ciò potrebbe esser necessario se si vuole aggiungere una particolare proprietà o metodo a tutti gli oggetti String, usando la relativa proprietà prototype. Di contro, se si vuole usare il metodo statico eval, l'argomento di questo dev'essere una stringa creata con notazione letterale; se si utilizza un oggetto String il risultato di ritorno sarà la stringa stessa.

PROPRIETA'

length

prototype

METODI

anchor("")

big()

blink()

bold()

fixed()

fontcolor("")

fontsize()

italics()

link("")

small()

split()

strike()

sub()

sup()

toLowerCase()

toUpperCase()

charAt()

indexOf()

lastIndexOf()

substring()

---FINE OGGETTO STRING---

UTILIZZO DELLE FINESTRE

L' oggetto centrale di questa lezione è window. I metodi fondamentali dell' oggetto window sono:

- open()
- close()

A questi possono essere aggiunti anche i metodi focus() e blur() (che risultano molto utili per decidere quale finestra deve stare in primo piano nel caso vengano aperte diverse finestre.

1) Vediamo il metodo open().

Questo accetta diversi parametri. Vediamo come il metodo open() potrebbe apparire:

```
open("nomeUrl", "nomeFinestra", "altreVariabili")
```

- nomeUrl Contiene l' indirizzo dei dati che compariranno nella finestra
- nomeFinestra Rappresenta, ovviamente, il nome di riferimento della finestra
- altreVariabili Sono delle variabili utili per definire l' aspetto della finestra

Quella che segue è la lista delle variabili della finestra:

VARIABLE	VALORE	DESCRIZIONE
copyhistory	yes/no 1/0	Copia history da finestra madre
directories	come sopra	Attiva o disattiva i bottoni del browser
height	altezza finestra in pixel	Altezza finestra
location	yes/no 1/0	Attiva o disattiva la barra dell'URL del

browser		
menubar	come sopra	Attiva o disattiva la barra dei menu del browser
resizable	come sopra	Specifica se la finestra può essere ridimensionabile
scrollbars	come sopra	Attiva o disattiva le barre di scorrimento del browser
status	come sopra	Attiva o disattiva la barra di stato del browser
toolbar	come sopra	Attiva o disattiva i pulsanti della barra degli strumenti
width	larghezza in pixel	Larghezza della finestra

È bene notare che per poter agire sulle finestre e sulle loro variabili bisogna assegnare ad una variabile il metodo open. Se volessimo aprire una finestra questo sarebbe, più o meno, il nostro codice:

```
var nuovaFinestra = window.open("prova.html" , "_blank" , "copyhistory=yes, toolbar=yes, status=no, resizable=yes, scrollbars=no, menubar=no, location=no");
```

2) Vediamo la proprietà opener().

Se da una finestra principale aprissimo altre n finestre, ad avere il fuoco sarebbe l'ultima di queste. A noi interessa invece che sia la finestra principale ad avere il fuoco di modo da poter controllare l'apparizione e la sparizione delle finestre "figlie" dalla finestra "madre". Ma c'è il problema che gli oggetti window che stiamo per aprire sono tutti ugualmente di classe alta; non c'è una effettiva ereditarietà perciò non si possono utilizzare gli attributi top, parent, etc.. La versione 3 del NN ci viene incontro con la proprietà opener.

Questa indica la finestra "apertrice" delle altre finestre.

Per riferirci alla finestra opener utilizziamo quindi window.opener.

FORMATTAZIONE DEGLI STILI

• I COLORI IN JAVASCRIPT

The string literals in this table can be used to specify colors in the JavaScript **alinkColor**, **bgColor**, **fgColor**, **linkColor**, and **vlinkColor** properties and the **fontcolor** method.

The following red, green, and blue values are in hexadecimal.

Color	Red	Green	Blue
aliceblue	F0	F8	FF
antiquewhite	FA	EB	D7
aqua	00	FF	FF
aquamarine	7F	FF	D4
azure	F0	FF	FF
beige	F5	F5	DC
bisque	FF	E4	C4
black	00	00	00
blanchedalmond	FF	EB	CD
blue	00	00	FF
blueviolet	8A	2B	E2
brown	A5	2A	2A
burlywood	DE	B8	87
cadetblue	5F	9E	A0

chartreuse	7F	FF	00
chocolate	D2	69	1E
coral	FF	7F	50
cornflowerblue	64	95	ED
cornsilk	FF	F8	DC
crimson	DC	14	3C
cyan	00	FF	FF
darkblue	00	00	8B
darkcyan	00	8B	8B
darkgoldenrod	B8	86	0B
darkgray	A9	A9	A9
darkgreen	00	64	00
darkkhaki	BD	B7	6B
darkmagenta	8B	00	8B
darkolivegreen	55	6B	2F
darkorange	FF	8C	00
darkorchid	99	32	CC
darkred	8B	00	00
darksalmon	E9	96	7A
darkseagreen	8F	BC	8F
darkslateblue	48	3D	8B
darkslategray	2F	4F	4F
darkturquoise	00	CE	D1
darkviolet	94	00	D3
deeppink	FF	14	93
deepskyblue	00	BF	FF
dimgray	69	69	69
dodgerblue	1E	90	FF
firebrick	B2	22	22
floralwhite	FF	FA	F0
forestgreen	22	8B	22
fuchsia	FF	00	FF
gainsboro	DC	DC	DC
ghostwhite	F8	F8	FF
gold	FF	D7	00
goldenrod	DA	A5	20
gray	80	80	80
green	00	80	00
greenyellow	AD	FF	2F
honeydew	F0	FF	F0
hotpink	FF	69	B4
indianred	CD	5C	5C
indigo	4B	00	82
ivory	FF	FF	F0
khaki	F0	E6	8C
lavender	E6	E6	FA
lavenderblush	FF	F0	F5
lawngreen	7C	FC	00
lemonchiffon	FF	FA	CD

lightblue	AD	D8	E6
lightcoral	F0	80	80
lightcyan	E0	FF	FF
lightgoldenrodyellow	FA	FA	D2
lightgreen	90	EE	90
lightgrey	D3	D3	D3
lightpink	FF	B6	C1
lightsalmon	FF	A0	7A
lightseagreen	20	B2	AA
lightskyblue	87	CE	FA
lightslategray	77	88	99
lightsteelblue	B0	C4	DE
lightyellow	FF	FF	E0
lime	00	FF	00
limegreen	32	CD	32
linen	FA	F0	E6
magenta	FF	00	FF
maroon	80	00	00
mediumaquamarine	66	CD	AA
mediumblue	00	00	CD
mediumorchid	BA	55	D3
mediumpurple	93	70	DB
mediumseagreen	3C	B3	71
mediumslateblue	7B	68	EE
mediumspringgreen	00	FA	9A
mediumturquoise	48	D1	CC
mediumvioletred	C7	15	85
midnightblue	19	19	70
mintcream	F5	FF	FA
mistyrose	FF	E4	E1
moccasin	FF	E4	B5
navajowhite	FF	DE	AD
navy	00	00	80
oldlace	FD	F5	E6
olive	80	80	00
olivedrab	6B	8E	23
orange	FF	A5	00
orangered	FF	45	00
orchid	DA	70	D6
palegoldenrod	EE	E8	AA
palegreen	98	FB	98
paleturquoise	AF	EE	EE
palevioletred	DB	70	93
papayawhip	FF	EF	D5
peachpuff	FF	DA	B9
peru	CD	85	3F
pink	FF	C0	CB
plum	DD	A0	DD
powderblue	B0	E0	E6

purple	80	00	80
red	FF	00	00
rosybrown	BC	8F	8F
royalblue	41	69	E1
saddlebrown	8B	45	13
salmon	FA	80	72
sandybrown	F4	A4	60
seagreen	2E	8B	57
seashell	FF	F5	EE
sienna	A0	52	2D
silver	C0	C0	C0
skyblue	87	CE	EB
slateblue	6A	5A	CD
slategray	70	80	90
snow	FF	FA	FA
springgreen	00	FF	7F
steelblue	46	82	B4
tan	D2	B4	8C
teal	00	80	80
thistle	D8	BF	D8
tomato	FF	63	47
turquoise	40	E0	D0
violet	EE	82	EE
wheat	F5	DE	B3
white	FF	FF	FF
whitesmoke	F5	F5	F5
yellow	FF	FF	00
yellowgreen	9A	CD	32

IL PASSAGGIO DEI DATI NEL FRAMESET

Affrontiamo ora un tema di notevole importanza, senza la cui conoscenza l'utilizzo del javascript risulta notevolmente limitato. Avendo un "set di frame" risulta necessario padroneggiare i metodi per interagire tra finestre contenute in frame diversi. Analizziamo le diverse esigenze.

1)SITUAZIONE:

Ho un setframe diviso in n frame.

In "n1" ho la pagina "pagina1" che contiene il pulsante "btn1", in "n2" ho la pagina "pagina2" che contiene il campo nascosto "nascosto1" a cui voglio passare il valore "20".

Il codice per mettere in comunicazione i due elementi è il seguente:

```
top.frames["n2"].document.forms["nome form in n2"].elements.nascosto1.value = 20;
```

Al posto di "top" può essere utilizzato "parent"; ma ci sono delle differenze e non sempre è la stessa cosa, più avanti vedremo perchè.