Oggetti built - in
(vedere Javascript Object Road Map)

L'oggetto Window
È l'oggetto principale della gerarchia degli oggetti e descrive la finestra correntemente aperta. Opera come oggetto globale per i programmi JavaScript lato client. Ogni finestra di un browser (ogni frame) viene rappresentata da un oggetto Window che definisce proprietà e metodi per la programmazione lato client.

La barra di stato
Mediante le proprietà **window.status** e **window.defaultStatus** si può controllare il contenuto nella barra di stato del browser. Il browser per default visualizza nella barra di stato il nome del link su cui è posizionato il mouse, ma si può controllare questo comportamento

```
<a href="http://www.disi.unige.it"
onMouseOver="window.status = 'visita il sito del DISI' ; return true;"
onMouseOut="window.status = ' '; ">www.disi.unige.it</a>
```

L'istruzione **return true;** nel gestore di eventi **onMouseOver** disabilita il comportamento di default del browser impedendogli di sovrascrivere il messaggio scelto dall'utente.

Timeout e intervalli
Il metodo **window.setTimeout**(codice,ritardo) ritarda l'esecuzione del codice JavaScript incluso come primo parametro (di solito si scrive il nome di una funzione) di una quantità di tempo pari a quella specificata nel secondo parametro (in millisecondi).

Oggetto Location
Fornisce informazioni sul documento correntemente visualizzato.

Esempio

```
<script>
window.alert("href: "     + window.location.href);
window.alert("protocol: " + window.location.protocol);
window.alert("host: "     + window.location.host);
window.alert("pathname: " + window.location.pathname);
</script>
```

Oggetto Navigator
L'oggetto Navigator permette di conoscere alcune proprietà del browser da cui sta arrivando una richesta; può essere usato per aprire pagine progettate specificamente per un particolare tipo di browser.
Bisogna fare attenzione perché quando c'è la ridirezione automatica il pulsante Back del browser non sempre funziona correttamente.

```
<script>
window.alert("appName: "     + window.navigator.appName);
window.alert("appCodeName: " + window.navigator.appCodeName);
window.alert("appVersion: "  + window.navigator.appVersion);
window.alert("userAgent: "   + window.navigator.userAgent);
window.alert("platform: "    + window.navigator.platform);
window.alert("language: "    + window.navigator.language);
</script>
```

Oggetto History
L'oggetto History è un array di indirizzi relativi alle pagine visitate dall'utente. Si tratta di informazioni private e quindi non leggibili. Questo oggetto supporta però tre metodi che permettono di navigare tra le pagine, simulando il comportamento dei pulsanti Back, Forward, e Go del browser.

```
window.history.back()
window.history.forward()
window.history.go(n) / window.history.go(-n)
```

NB: in molti casi quando si usano proprietà e metodi si può **omettere la parola** `window` (l'importante è che non ci siano ambiguità nei nomi, cioè che non ci siano oggetti diversi per i quali sono definite proprietà - o metodi - con lo stesso nome).

Il DOM – Document Object Model

L'oggetto Document
Ogni oggetto Window ha una proprietà `document` che si riferisce al documento correntemente visualizzato nella finestra del browser. L'oggetto Document è caratterizzato da molte proprietà e metodi.
Alcune proprietà dell'oggetto Document sono degli array le cui componenti rappresentano a loro volta degli oggetti contenuti nel documento stesso. Ad esempio, l'array `forms[]` contiene oggetti Form che descrivono i moduli contenuti nel documento HTML. Allo stesso modo, l'array `links[]` descrive tutti i link, l'array `applets[]` descrive tutte le applet all'interno di un documento, etc. etc.

Inoltre, se agli elementi di un documento HTML viene associato un **nome** mediante l'attributo `name="xxx"`, viene automaticamente creata una proprietà dell'oggetto Document di nome xxx.

Esempio
Se la prima immagine nel file HTML è

```
<img src="pippo.gif" name="pippo">
```

le proprietà `document.images[0]` e `document.pippo` si riferiscono entrambe ad essa.

Conviene dare dei **nomi significativi** agli elementi presenti in un file HTML. È più facile far riferimento ad essi grazie al loro nome piuttosto che usando la notazione ad array (che li indentifica rispetto alla loro posizione nel documento stesso).

Alcune proprietà dell'oggetto Document sono le seguenti:

```
document.bgColor
document.fgColor
document.lastModified
document.linkColor
document.location
document.referrer
document.title
document.URL
```

Per i metodi si ha:

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

Questi metodi permettono ai programmi JavaScript di inserire del testo in modo dinamico all'interno dei file HTML. **Attenzione**: si può inserire testo in un documento HTML in modo dinamico solo durante la parsificazione

del documento stesso. Se si usa il metodo **`document.write()`** all'interno di una funzione o all'interno di un gestore di eventi si sovrascrive il documento corrente.

Il metodo **`document.close()`** indica la conclusione dell'operazione di scrittura (è buona norma usare questo metodo). Il metodo **`document.open()`** è invece facoltativo: se non si usa, non appena il browser incontra il metodo **`document.write()`** in un documento già chiuso, assume che si debba aprire un nuovo documento.

Esempio

```
<html>
<head>
<title>Prova oggetto Document</title>
</head>

<script>
var x=window.prompt("dammi una stringa in formato RGB per il colore del testo","");
var y=window.prompt("dammi una stringa in formato RGB per il colore dello
sfondo","");
document.write("<body bgcolor=#" + x + " text=#" + y + ">");
document.close();
</script>

<h1>Oggetto Document</h1>
<h3>prova colori</h3>
<h3>prova colori</h3>
<h3>prova colori</h3>
<h3>prova colori</h3>
<br>
<br>
Titolo:
<script> document.write(document.title);</script>
<br>
URL:
<script> document.write(document.URL);</script>
<br>
Last Update:
<script> document.write(document.lastModified);</script>
<br>
Referrer:
<script> document.write(document.referrer);</script>
</body>
</html>
```

Immagini
La proprietà **`document.images[]`** definisce un array di elementi di tipo immagine (Image), uno per ogni immagine presente nel documento HTML. L'oggetto Image permette di manipolare dinamicamente le immagini presenti in un documento.

Esempio

```
<script>
msg="";
for (prop in document.images[0])
msg = msg + "\n" + prop + ": " + document.images[0][prop];
window.alert(msg)
</script>
```

La proprietà **src** dell'oggetto Image è accessibile in lettura e scrittura e permette di cambiare l'immagine correntemente visualizzata nel browser per ottenere oggetti grafici sofisticati. Si usa per esempio nel caso del roll over delle immagini. **Attenzione**: perché il roll over funzioni correttamente la nuova immagine deve avere le stesse dimensioni di quella che viene rimpiazzata.

Esempio

Roll over dell'immagine correntemente sotto il puntatore del mouse (guardate l'esempio completo in rete)

```
<a href=""
 onMouseOver="document.im1.src='images/rossob.gif';"
 onMouseOut="document.im1.src='images/rosso.gif';"
 onClick="return false;"><img border="0" width="150" height="150"
 name="im1" src="images/rosso.gif"></a>
```

Attenzione: in alcuni browser di vecchia generazione le immagini non "sentono" gli eventi onMouseOver, onMouseOut, onClick e quindi per ottenere questi effetti esse vanno inserite in link fittizi. Per evitare che un clic sull'immagine possa causare una navigazione non voluta si deve usare l'istruzione **onClick="return false;"** in modo da disabilitare il comportamento di default del browser.

Per non avere troppi ritardi si può pensare di pre-caricare le immagini nella memoria cache del browser creando immagini "fuori schermo".
Si creano degli oggetti di tipo immagine con il costruttore **Image()** cui si associano le immagini che vengono caricate sul client senza essere visualizzate.

```
<html>
<head>

<script>
var temp = new Array(9);            // creo un array di 9 celle
temp[0] = new Image();              // ad ogni cella associo un oggetto Image()
temp[0].src = "images/rosso.gif" ;  // alla proprietà src associo l'immagine
temp[1] = new Image();
temp[1].src = "images/rossob.gif" ;
temp[2] = new Image();
temp[2].src = "images/giallo.gif" ;
temp[3] = new Image();
temp[3].src = "images/giallob.gif" ;
temp[4] = new Image();
temp[4].src = "images/verde.gif" ;
temp[5] = new Image();
temp[5].src = "images/verdeb.gif" ;
temp[6] = new Image();
temp[6].src = "images/blu.gif" ;
temp[7] = new Image();
temp[7].src = "images/blub.gif" ;
temp[8] = new Image();
temp[8].src = "images/qbianco.gif" ;
</script>

</head>

 <body>
...
...
</body>
</html>
```

Gli oggetti di tipo immagine hanno la proprietà **src** che definisce il file sorgente che rappresenta l'immagine e la proprietà **complete** che ha valore **false** fino a quando l'immagine è in fase di caricamento. Inoltre, questi oggetti reagiscono agli eventi:

onLoad, che viene richiamato quando termina il caricamento;
onAbort, che viene richiamato quando si interrompe il caricamento;
onError, che viene richiamato quando si verifica un errore durante il caricamento.

Altre proprietà dell'oggetto Image sono le seguenti:

```
document.images[0].width
document.images[0].height
document.images[0].border
document.images[0].hspace
document.images[0].vspace
document.images[0].lowsrc
```

Link
L'array **links[]** dell'oggetto **document** contiene oggetti di tipo **Link** che rappresentano i collegamenti ipertestuali presenti nel documento stesso. L'oggetto Link ha molte proprietà simili a quelle dell'oggetto Location. In un caso viene descritta la URL correntemente visualizzata nella barra di navigazione, nell'altro vengono descritte le destinazioni dei link presenti nel documento HTML.

L'oggetto Link permette di scrivere programmi che scoprono tutte le pagine raggiungibili a partire da una determinata pagina (web crawler)

L'oggetto Link supporta parecchi gestori di eventi, tra cui:

onClick (se si restituisce il valore false - **return false** - si inibisce il comportamento standard dei link)
onMouseOver
onMouseOut

Ancore
L'array **anchors[]** dell'oggetto **document** contiene informazioni sulle ancore interne. Ha la proprietà **name** che contiene il valore dell'attributo HTML **name** e le proprietà **x** e **y** che specificano la posizione dell'ancora all'interno del documento.

Applet
L'array **applets[]** dell'oggetto **document** contiene oggetti che rappresentano le applet all'interno del documento.

Eventi
I programmi JavaScript utilizzano un modello di programmazione **guidato dagli eventi**.
JavaScript lato client supporta un numero abbastanza limitato di tipi di evento; inoltre, oggetti diversi in circostanze simili possono generare eventi diversi.

In molti casi i gestori di eventi restituiscono dei valori che possono essere modificati per impedire il comportamento di default del browser.

I gestori di eventi sono espressi come attributi HTML i cui valori sono istruzioni JavaScript che vengono eseguite quando si verifica l'evento stesso. Quando le istruzioni sono troppe, si preferisce definire una funzione che viene richiamata nel tag HTML.

Si possono definire esplicitamente i gestori di eventi come funzioni e assegnare tali funzioni a proprietà JavaScript. Ogni oggetto JavaScript lato client ha delle proprietà che rappresentano i gestori di eventi: se si

assegna una funzione a queste proprietà, questa verrà richiamata ogni volta che si verifica l'evento. **Attenzione**: le proprietà devono essere scritte in minuscolo:

```
document.b1.onclick
document.b1.onmouseover
document.b1.onmouseout
```

L'oggetto Form

E' uno degli oggetti più complessi e importanti del DOM: permette all'utente di inviare delle informazioni ad un server remoto (per es. per registrarsi ad un forum oppure per effettuare un acquisto on line). In JavaScript **l'enfasi non è posta sull'invio dei dati ma piuttosto sulla loro validazione:** è meglio "far partire" un modulo solo quando i dati in esso contenuti sono nel formato corretto previsto dal programma remoto che su questi dati dovrà lavorare.

Quando si inserisce un modulo in un documento HTML

```
<form name="frm1" enctype=" … " action=" … " method=" … ">

…

</form>
```

vengono create le proprietà

```
document.frm1.name
document.frm.encoding
document.frm1.action
document.frm1.method
document.frm.length
```

Esempio

```
<html>
<head>


<script>
function visprop(){
      window.alert("Nome: " + document.frm1.name);
      window.alert("Tipo di codifica: " + document.frm1.encoding);
      window.alert("Numero elementi: " + document.frm1.length);
      window.alert("Metodo: " + document.frm1.method);
}
</script>

<body>
<center>
<b>Proprietà dell'oggetto form</b>
<br>
<br>
<form name="frm1" action="xxx" method="post">
<table>
<tr>
<td>Nome</td>
<td><input type="text" name="nome"></td>
```

```
</tr>
<tr>
<td>Cognome</td>
<td><input type="text" name="cognome"></td>
</tr>
<tr>
<td>Sesso</td>
<td>
<input type="radio" name="sesso" value="m">M
<input type="radio" name="sesso" value="f">F
</td>
</tr>
<tr>
<td colspan="2">
<center>
<input type="button" value="Proprietà form" onClick="visprop();">
</center>
</td>
</tr>
</table>

</form>
</center>
</body>
</html>
```

Quando un browser legge il codice HTML di un modulo di nome **`frm1`** crea anche un array **`document.frm1.elements[]`** che contiene una cella per ogni elemento inserito nel modulo. Se agli elementi è stato dato un nome (ricordate che si **DEVE** dare sempre un nome agli elementi di un modulo) vengono anche create delle proprietà aggiuntive. Nell'esempio precedente avremo:

```
document.frm1.nome
document.frm1.cognome
document.frm1.sesso
```

Possiamo scrivere una funzione come quella seguente che legge il tipo, il nome e il valore di ogni elemento di un modulo

```
<script>
function visprop()
{
 var dummy=document.frm1.elements;  // serve solo per semplificare i nomi nel for
 for (i=0; i<document.frm1.length; i++)
      window.alert(dummy[i].type + ", " + dummy[i].name + ", " +  dummy[i].value);
 }
</script>
```

A questo punto bisogna considerare tutti gli elementi di un modulo uno alla volta per vedere quali proprietà e metodi supportano. Distinguiamo varie classi di elementi: elementi che permettono di inserire del testo (TEXT, TEXTAREA, PASSWORD, FILE), elementi che permettono di scegliere tra valori prefissati (RADIO, CHECKBOX), pulsanti che attivano delle azioni (BUTTON, RESET, SUBMIT).
I menu a tendina (SELECT + OPTION) sono più complicati e meritano un discorso a parte.

TEXT, TEXTAREA, PASSWORD, FILE
La proprietà **`value`** in questo caso assume il valore che l'utente digita quando interagisce con l'oggetto.
La lunghezza dell'input si legge nella proprietà **`length.`**

Reagiscono all'evento **onChange** che si verifica quando l'utente sposta il focus al di fuori dell'elemento stesso. Reagiscono anche gli eventi generati dai tasti della tastiera (**onKeyDown, onKeyPress**, …) e agli eventi **onFocus, onBlur** che vengono generati quando gli elementi acquisiscono/perdono il focus. Nel campo di tipo password l'eco a video è costituito da una sequenza di *****.

RADIO, CHECKBOX
Sono caratterizzati da uno stato che è rappresentato dalla proprietà **checked**: questa proprietà vale **true** nel caso di pulsante selezionato, **false** altrimenti.
Il valore associato a questi pulsanti non è scritto dall'utente ma è codificato nel codice HTML come valore dell'attributo **value**. Reagiscono all'evento **onClick** che viene generato quando l'utente cambia la selezione.

BUTTON, RESET, SUBMIT
Non sono caratterizzati da uno stato e il valore dell'attributo **value** è l'etichetta che compare sul bottone, che in questo caso non corrisponde al valore che viene inviato quando si spedisce il modulo. Reagiscono all'evento **onClick** che viene usato principalmente per attivare delle funzioni.

Quando si preme su un pulsante di tipo **submit** il modulo viene inviato all'indirizzo specificato come valore dell'attributo **action**. Si può modificare il comportamento di questo punsante restituendo il valore **false**. Analogamente per il pulsante di tipo **reset**.

```
<input type="submit" value="Invia"
onClick="temp = window.confirm('Sei sicuro di voler inviare i dati?'); return
temp;">

<input type="reset" value="Cancella"
onClick="temp = window.confirm('Sei sicuro di voler cancellare i dati?'); return
temp;">
```

Oppure si può scrivere in modo più compatto:

```
<input type="submit" value="Invia"
      onClick="return window.confirm('Sei sicuro di voler inviare i dati?');">
```

SELECT e OPTION
L'oggetto di tipo **Select** permette di definire dei menu a tendina ed ha le proprietà **length**, **type**, **selectedIndex**, più la proprietà **options[]** che definisce un array di oggetti di tipo **Option**, uno per ogni elemento **<option>** inserito all'interno di un tag di tipo **<select>**.

Questo oggetto reagisce all'evento **onChange** che si verifica quando l'utente cambia la selezione corrente nel menu. Il valore della proprietà **selectedIndex** è un numero intero che specifica l'indice dell'opzione che è stata scelta.

Per quanto riguarda le opzioni, ogni volta che il browser incontra un tag del tipo

```
<option value="xx">bla bla bla</option>
```

crea le proprietà **value** e **text.**

Si possono creare nuove opzioni oppure modificarle dinamicamente usando il costruttore **Option()** (ovviamente, se non sono salvate su un server le nuove opzioni verranno perse quando si lascia la pagina).

Esempio

```
<html>
<head>

<script>

function add()
{
  var temp = new Option();
  var nuova=window.prompt("dammi una città","");

  if ((nuova != "") && (nuova != null))
      {
        temp.value=nuova;
        temp.text=nuova;
        document.frm1.citta[document.frm1.citta.length] = temp;
      }
}

function rem()
{
      document.frm1.citta[document.frm1.citta.selectedIndex] = null;
}

</script>
</head>

<body>
<center>

<h3>Aggiunta dinamica di opzioni in un menu</h3>
<br>
<br>
<form name="frm1">
<select name="citta" SIZE="10">
<option value="GE">Genova</option>
<option value="MI">Milano</option>
<option value="NA">Napoli</option>
<option value="PA">Palermo</option>
<option value="TO">Torino</option>
<option value="VE">Venezia</option>
</select>
<br><br>
<input type="button" value="Aggiungi una città" onClick="add();">
<input type="button" value="Cancella una città" onClick="rem();">
</form>
</center>
</body>
</html>
```