Modalità per **inserire JavaScript in pagine web**

Esistono due fondamentali modi per **inserire** uno script in un documento HTML:

primo modo: Con tag <**script**> </script> nell'**intestazione** o nel **corpo**

consigliato <script **Language="JavaScript"**> per versione 1.0
<script Language="JavaScript1.2"> versione 1.2 leggibile da NN 4.0 ed IE 4.0

oppure <script **type="text/javascript"**>

esempio:

```
<head><title>....</title>
<script >
<!--
   document.write("Sono uno script interno")      // istruzione di output
//-->
</script>
</head>
```

secondo modo: Caricandolo **da file esterno** (con estensione .js) con **attributo src** nel tag **script**

<script src= "*nomefile.js*"></script>

esempio:

```
<head>
<title>....</title>
<script src="primo.js">
<!--
//-- >
</script>
</head>
```

con primo.js

```
alert ("Sono uno script esterno")      // istruzione di output con OK
```

oppure primo.js

```
confirm ("Sono uno script esterno")  // istruzione di output con OK e Annulla
```

nb: usare solo ASCII puro infatti i browser - incapaci di *interpretare* caratteri diversi - non visualizzano il risultato voluto e l'utente non ha informazione sul motivo del malfunzionamento.

Finestre modali

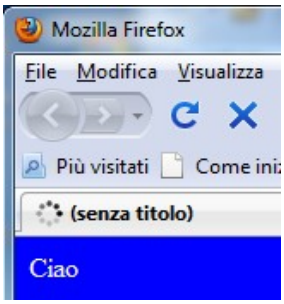
JavaScript è in grado di generare tre differenti tiplogie di "finestre implicite" (dette anche "finestre incorporate", "**finestre modali**", o "finestre di dialogo") ricorrendo a metodi dell'ogetto window:

alert crea una finestra che visualizza un avvertimento,

confirm una finestra che pone una domanda e chiede conferma dando la possibilità di scegliere tra due opzioni ("ok" e "annulla").

prompt una finestra usata per consentire **input** che pone una domanda e consente all'utente di dare la risposta che vuole. Può avere anche un valore di risposta predefinito.

Modalità per **eseguire JavaScript**



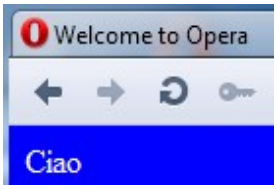
- Si possono **immettere istruzioni** direttamente **nel campo URL** di un Browser (modalità considerata *pericolosa*)

  esempio: per scrivere "Ciao" in bianco, su sfondo blue

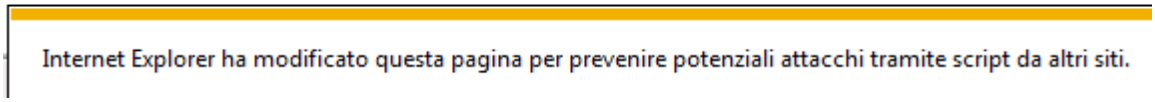
  digitare nel box indirizzo (con Opera o Safari o in *nuova scheda* per versioni meno aggiornate di Firefox poco orientate alla sicurezza):

  javascript: document.write("Ciao"); document.**bgColor**="blue"; document.**fgColor**="white";

  e premere INVIO



nb: altri browser (Google Chrome ed IE) modificano la pagina per prevenire potenziali attacchi ed interpretano le istruzioni come testo da ricercare. IE fornisce anche avviso



Il *comando* **document.write** è in effetti la chiamata del metodo write dell'oggetto document[1].
La stringa fornita come parametro viene trattata come se fosse stata inserita direttamente nella pagina HTML.

- Quindi un modo per ***eseguire*** uno script è **in luogo di URL in un collegamento ipertestuale**

```
<a href="JavaScript: comando">…</a>
```

esempio:

```
<body>….
        <a href= "JavaScript: alert('Hai cliccato')">clicca</a>
                <!-- attenzione: alternare " con '  -->
</body>
```

È caso particolare: impostare come **valore** dell'attributo **action** di un form (quindi come **URL**) prevedendo che alla pressione di INVIO all'interno di una casella di testo sia eseguito tale codice javascript

```
<form action="javascript:document.write('Ciao')" method="get">
   <b>Premi invio</b>
   <input type="text" size="1">
</form>
```

[1] In JavaScript **lato client** l'oggetto ***Window*** serve da oggetto globale: è il padre della gerarchia che contiene ***document*** (che contiene oggetti "riflessi" da HTML quali ***link***, … ***form*** che è contenitore di ***elements*** quali text, button etc…).

- Altra modalità: **creazione dinamica**

  esempio: creo un nuovo documento HTML che esegue alert() definendo la funzione CreaHtml() in file con estensione js

  codice pagina **crea.js**

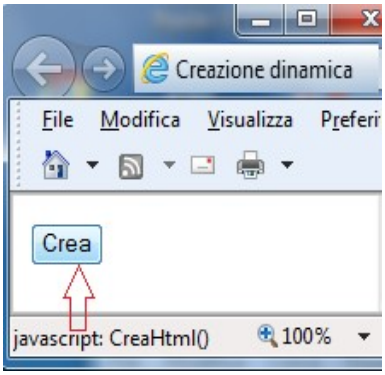
```
function CreaHtml (){
        document.open()
        document.writeln("<script Language= 'JavaScript'>")
        document.writeln(alert("Ciao") + "</script>")
        document.writeln("funzione eseguita con successo")
        document.close()
    }
```

  codice pagina **crea.html**

```
crea.html   crea.js
1  <html>
2  <head>
3      <title>Creazione dinamica</title>
4      <script src="crea.js">
5      </script>
6  </head>
7  <body>
8      <form action ="javascript: CreaHtml()">
9          <input type="submit" value="Crea">
10     </form>
11 </body>
12 </html>
```

La *chiamata* ad una funzione può essere ad esempio all'interno dell'attributo *action* di un tag form[2]:

```
<form action ="javascript: CreaHtml()">
    <input type="submit" value="Crea">
</form>
```



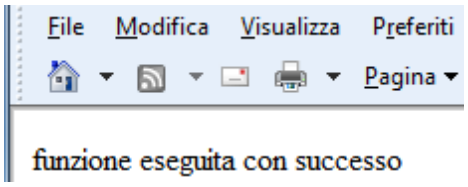
La pressione su pulsante di tipo **submit** ne lancia esecuzione

NB: non si possono definire funzioni all'interno di altre funzioni

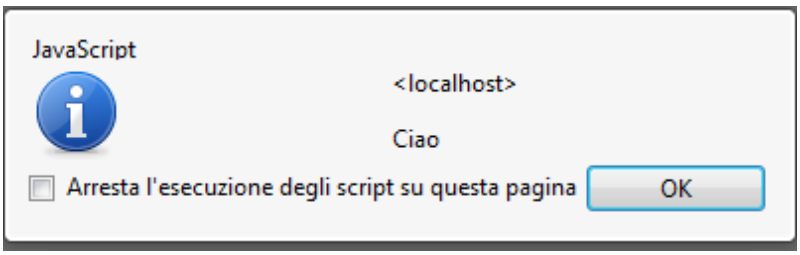
[2] Un form o modulo HTML è una sezione di un documento contenente anche elementi speciali chiamati controlli (caselle di spunta, radiocomandi, menu, ecc.) ed etichette su quei controlli. Gli utenti generalmente "riempiono" un modulo modificando i suoi controlli (introducendo del testo, selezionando elementi di menu, ecc.), prima di inoltrare il modulo ad un programma per il trattamento dei dati (ad es. ad un server Web, ad un server di posta, ecc.)

In IE (oggi browser obsoleto) si doveva *Consentire il contenuto bloccato*

Effetto: nuovo documento costruito



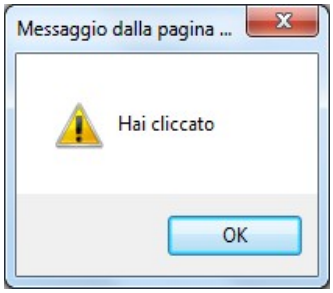
In Opera si prevede di poter "Arrestare l'esecuzione degli script ..." potenzialmente ***pericolosi***



- Altro modo: in seguito all'attivazione di un **evento**

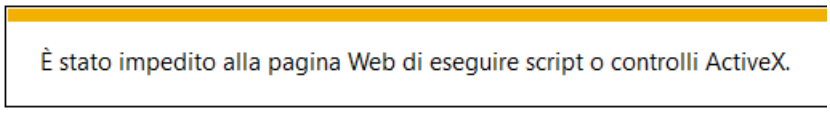
esempi:

```
<body>....
        <a href= "nomerisorsa.htm" onClick = "alert('Hai cliccato')">clicca</a>
                                                <!-- attenzione: alternare " con '  -->
</body>
```



NB: *nomerisorsa*.htm è la pagina che si visualizza nella finestra del browser se si conferma con la pressione di OK

Si noti come in IE (oggi browser obsoleto) si dovesse consentire a sbloccare il contenuto *potenzialmente pericoloso*

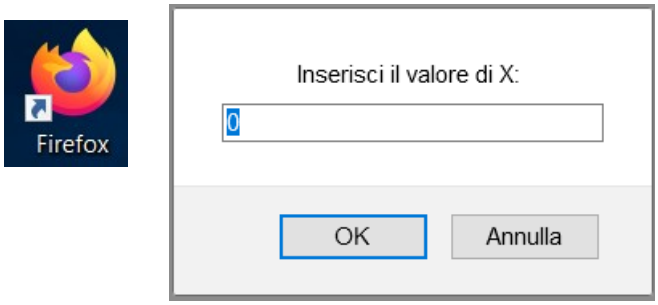


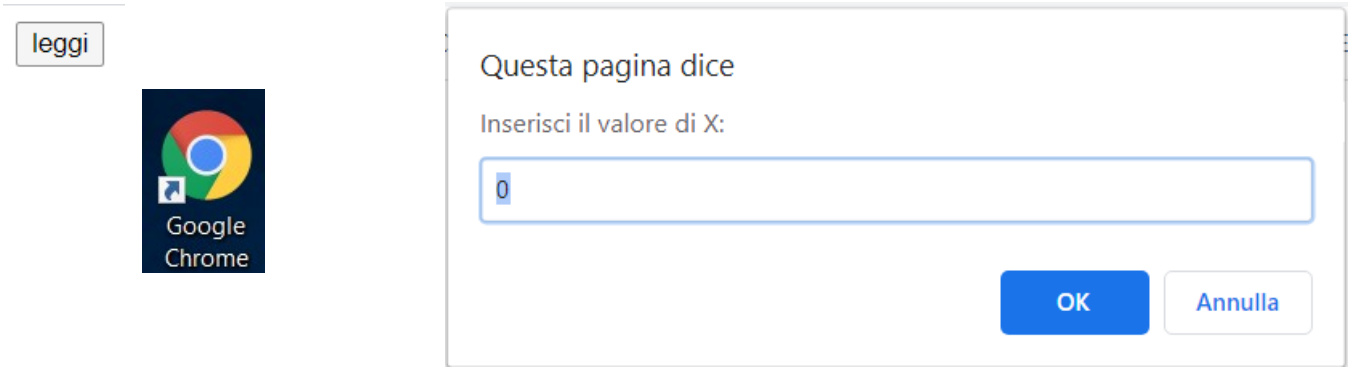
esempio:

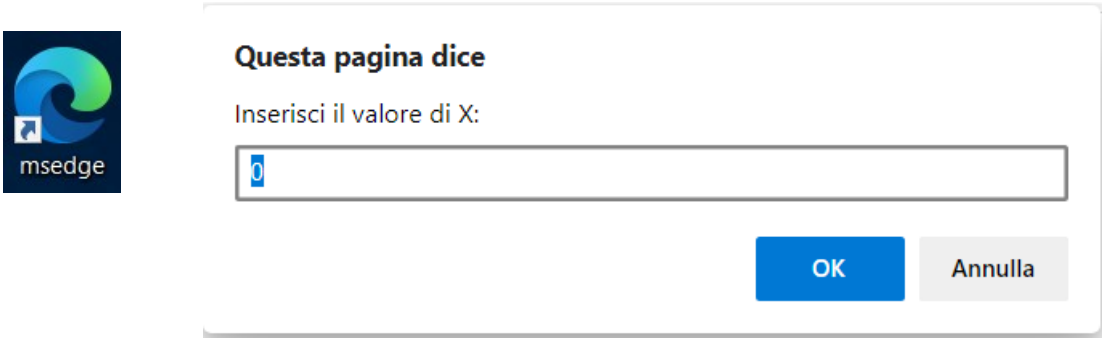
```
<html>
  <head>
  <title>Leggi</title>
  <script><!--
  function leggi() {
      var X = prompt("Inserisci il valore di X: ", 0)   /* Istruzione di Ingresso
                                                            che restituisce stringa
                                                            valore iniziale posto a 0 */
  document.write("Hai inserito " + X)
      }
  //-->></script>
  </head>
```

```
<body>
 <form><input type="reset" value="leggi" onClick = "leggi()"></form>
</body>
</html>
```

Ogni browser visualizza un'interfaccia diversa





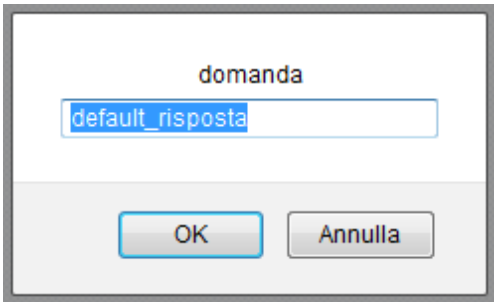


Prompt box javascript

L'istruzione **prompt** box attiva una finestra modale in cui l'utente può digitare liberamente un testo. La sintassi è molto semplice:

prompt ("***domanda***","***default_risposta***");

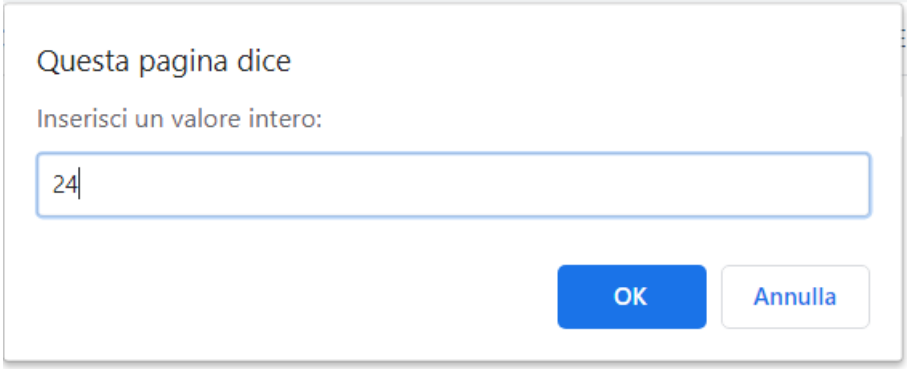
Ogni browser visualizza un'interfaccia diversa ma con stessi componenti: etichetta, casella di testo, pulsante OK ed ANNULLA (in figura: uso Mozilla-Firefox versione 11.0)
ed ugule funzionalità



esempio:

```
<html>
  <head>
    <title>Leggi e somma</title>
     <script >
      <!--
     function leggiInt(){                                  // funzione chiamata
       var X = parseInt (prompt ("Inserisci un valore intero: ", 0))
              // con effetto di conversione da stringa in intero
       X = X+3
       alert("Sommando a 3 si ottiene " + X)   }
      //-->
      </script>
  </head>
  <body>
   <form>
          <input type="submit" value="somma" onClick = " leggiInt()">
   </form>
  </body>
</html>
```

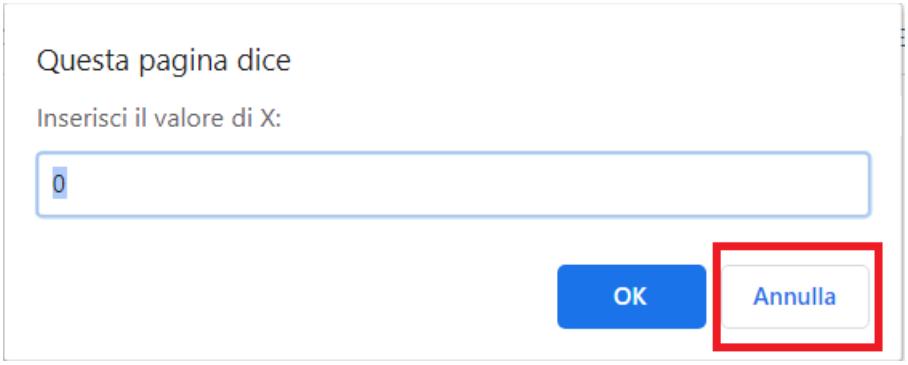
somma



Impostando ad esempio 24 e premendo OK

si ottiene

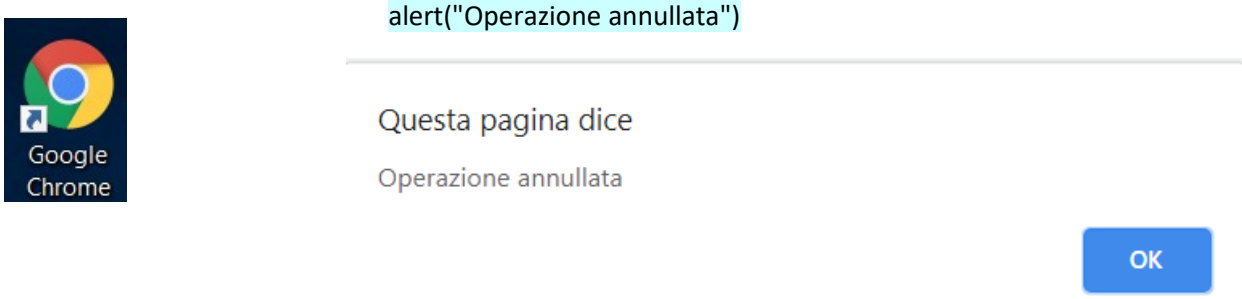
nb: premendo il pulsante Annulla nella finestra del prompt,



invece di una stringa si ottiene **null**
e la conversione individua che non è un numero **NaN** (Not-a-Number).

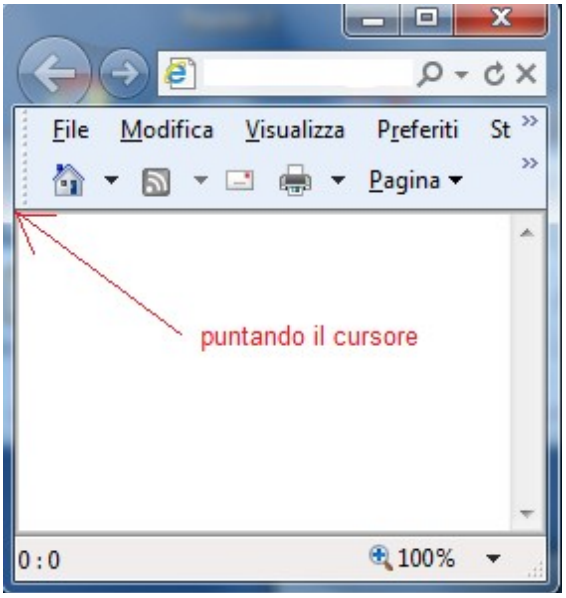
Codice migliore, prevede un controllo:

```
function leggiInt(){                                        // funzione chiamata
        var input = prompt("Inserisci un valore intero: ", 0)
        if (input != null ){
            var X = parseInt(input)  // effetto di conversione da stringa in intero
            X = X+3
            alert("Sommando a 3 si ottiene " + X)
        }
        else
            alert("Operazione annullata")
```



esempio: Script che ***visualizza le coordinate del cursore*** (su barra di stato)

```
<html>
 <head> <title> Stato </title>
   <script language="javascript">
    <!--
    function MouseStatusPosition()  {
                var X = window.event.x + ' : ' + window.event.y;
                window.status = X;
                return true;
            }
   //-->
   </script>
   </head>
  <body onMouseMove="MouseStatusPosition()">
  </body>
  </html>
```



coordinate 0 : 0

esempio:

```
<html>
    <head>
     <title>Modifica del colore del testo e sfondo </title>
    </head>
    <body>
        <div style="border: solid 5"
            onMouseOver="document.fgColor='teal';
                         document.bgColor='lightgreen';">
          testo inserito in rettangolo con bordo spesso
        </div>
    </body>
</html>
```

Prima del passaggio del mouse

testo inserito in rettangolo con bordo spesso

e dopo (sfondo verde chiaro ed elementi in primo piano – *foreground* – verde acqua) :

testo inserito in rettangolo con bordo spesso

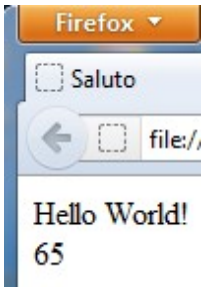
esempio:

```
<html>
<head>  <title>Scritta scorrevole in form al caricamento</title>
<script language="javascript">
        var ScrollString="   Programmazione  docente: Biasotti Paola       ";
        var timer = 0;
        function Scrollon()
            {
         document.box.boxtext.value = ScrollString;
         ScrollString=ScrollString.substring(1,ScrollString.length)
                + ScrollString.charAt(0);
         // diminuire il valore del timeout (100) per maggiore velocita'
         timer = setTimeout("Scrollon()",100) ;
            }
</script>
</head>
<body onLoad = Scrollon()>
<form name = "box" onsubmit = "0">
        <input type="text" name="boxtext" size="35" value="">
</form>
</body>
</html>
```

Esempio di conversioni **da ASCII a intero e viceversa**

```
<html>
  <head> <title>Conversioni: da ASCII a intero e viceversa</title>
    <script>
    <!--
   // Converte un numero intero (valore unicode) a carattere ASCII e visualizza
    function itoa(i){
          document.write(String.fromCharCode(i));
    }

   // Converte un carattere ASCII in un intero (valore unicode) e visualizza
   function atoi(s){
          document.write(s.charCodeAt());          // per default il primo carattere
   }
    // -->
    </script>
  </head>
  <body >
    <form>
  <input type="button" value="65" onClick = "itoa(65)"><p>
          <input type="button" value="A" onClick = "atoi('A')">
    </form>
  </body>
</html>
```

Esempio con **inserimento di script nel body**:

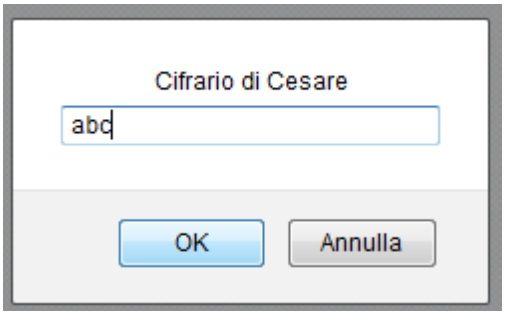



```
<html>
        <head>
                <title>Saluto</title>
        </head>
        <body>
          <script type="text/javascript">
          // outputs: "Hello World!"
          document.write(String.fromCharCode(72,101,108,108,111,32,87,111,114,108,100,33));

          document.write("<br>");

          // outputs: A  cioè il primo carattere della stringa
          document.write("ABC".charCodeAt(0));
          </script>
        </body>
  </html>
```

Si propone una brutale soluzione del "Cifrario di Cesare":

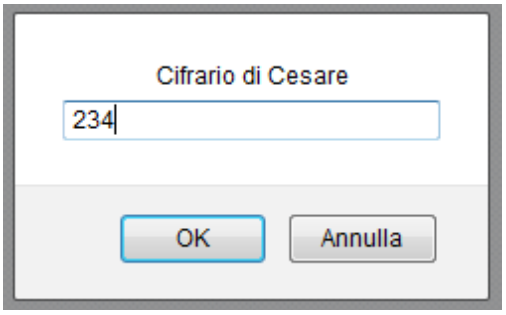
al click su pulsante

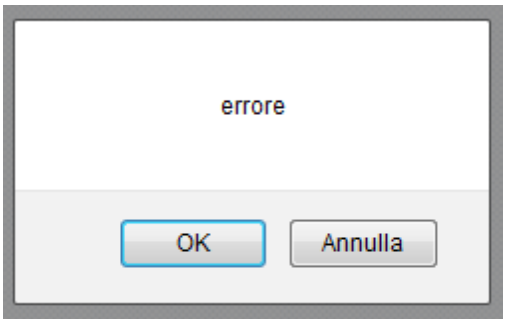
si apre una finestra modale dove inserire il testo in chiaro



alla pressione di OK
si ha conferma del messaggio forzato in maiuscolo
e si apre altra finestra modale con messaggio cifrato

Si prevede **controllo**: nel caso di si digitino caratteri che, forzati in maiuscolo, non siano compresi tra A e Z (ad esempio numeri o segni di interpunzione)
si avverte di errore e si forza l'uscita dallo script.



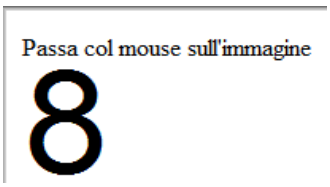


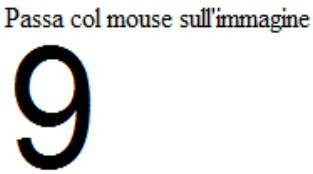
Pagina **cifra.html**

```
<html>
  <head> <title>Cesare</title>
    <script>
    <!--
    // Converte un numero intero (valore unicode) a carattere ASCII
    function itoa(i){
        return String.fromCharCode(i);
    }

    // Converte un carattere ASCII in un intero (valore unicode)
    function atoi(a){
        return a.charCodeAt();
    }

    function cifra() {
        var k = 3;                          // chiave
        var cifrato = "";
        var originale = prompt("Cifrario di Cesare", "Digita parola");
        var chiaro = originale.toUpperCase();
        document.write("Hai inserito " + chiaro);                    // per test

        for(var conta = 0; conta < chiaro.length; conta++){
          ch = chiaro.charAt(conta);
          if (ch >= 'A' & ch <= 'Z') {
              num = atoi(ch);            // converto in intero singola lettera
              num = num + k;             // cifro
              lettera = itoa(num);       // converto in carattere
              cifrato += lettera;
           }
           else {
                          confirm ("errore");
                          exit;
           }
        }
        confirm (cifrato);
    }
    // -->
    </script>
  </head>
  <body >
    <form>
            <input type="button" value="cifra" onClick = "cifra()">
    </form>
  </body>
</html>
```

Esempi: ***Immagini rollover***

```
<html>
    <head> <title>Script con immagini rollover</title></head>
     <body >
      <div>Passa col mouse sull'immagine</div>
      <img src="8.gif" onMouseOver=" document.images[0].src= '9.gif' ;"
                        onMouseOut=" document.images[0].src ='8.gif';">
       </body>
 </html>
```

Prima del passaggio del mouse 

e dopo: 

NB: si può sostituire a **document.images[0]**.src semplicemente **this**.src infatti l'immagine può essere referenziata con **this** cioè un particolare puntatore che fa riferimento all'oggetto con cui si sta lavorando.

Si può anche impostare un *nome* per l'immagine e referenziarla come mostrato:

esempio:

```
<html>
    <head><title> Script con immagini rollover e link fittizio</title></head>
    <body>
          <a href="#"
          onMouseOver="document.img1.src = '8.gif';"
          onMouseOut=" document.img1.src = '9.gif';">
          <img src="9.gif" name="img1" border="0" alt="numero">
           </a>
    </body>
</html>
```

esempio:

```
<html>
  <head><title> Script con immagini rollover e link per autotest</title></head>
  <body>
   <a href="http://www.w3schools.com/quiztest/quiztest.asp?qtest=JavaScript">

   <img src="9.gif" border="0" alt="numero"
          onMouseOver="this.src = '8.gif';"
          onMouseOut="this.src = '9.gif';">
   </a>
  </body>
 </html>
```

Esempio con **array di immagini** ed **effetto rollover**:

```
<html>
  <head> <title>Script con immagini rollover e array</title>
  <script language="javascript">  <!--
      var immagini = new Array();                    // creo oggetto Array
      var indice = 0;
      immagini[indice++] = "8.gif";
      immagini[indice++] = "9.gif";
      var scarica = new Array();                     // creo altro oggetto Array
          if (document.images) {
             for (i=0; i<scarica.length; i++) {
                ok[i] = new Image();                 // creo oggetto Image
                ok[i].src = scarica[i]; }
          }
     // -->
     </script>
     </head>
  <body >
   <div>Passa col mouse sull'immagine</div>
  <img src="8.gif" onMouseOver="this.src='9.gif'"
                   onMouseOut="this.src='8.gif'" border="0">
  </body>
</html>
```

Per aumentare l'efficienza si può pre-caricare la seconda immagine nella cache del browser al caricamento della pagina inserendo la seguente funzione nel tag <script>

```
function preLoadImages()   {
        swap1 = new Image();       // creo oggetto Image
        swap1.src = "9.gif";
       }
```

e introducendo nel tag body il ***gestore di evento*** : <body ***onLoad***= "preLoadImages()">

Linkografia

Guide

http://www.lucagalli.net/ita/didattica/corsoJS/summary.htm semplice corso scaricabile

http://javascript.html.it/guide/leggi/25/guida-javascript-di-base/
http://javascript.html.it/guide/leggi/26/guida-javascript-per-esempi/
http://javascript.html.it/guide/leggi/175/guida-javascript-tecniche-avanzate/

http://it.wikipedia.org/wiki/JavaScript

funzioni per stringhe http://www.morpheusweb.it/html/manuali/javascript/javascript_stringhe.asp

In IE (old version) non erano gestite ulteriori modalità:

- **Creazione dinamica in una finestra.** Produce infatti errore l'apertura con

open(" ", "NomeFinistra", "toolbar= no, menubar=no")

- uso caratteri per richiamare da HTML valori JavaScript
  Le espressioni, infatti, possono essere calcolate anche all'interno di tag HTML permettendo di costruire valori "al volo": la sintassi è:

&{espressione};

esempio:

<input type= "txt" size=10 value="&{*nomevariabile*};" > </input>

creando, ad esempio, uno script che preleva l'ora d'inizio del caricamento della pagina sul client e la conserva nella variabile resa visibile all' utente anche dopo molto tempo dall'inizio del caricamento della pagina.

Ricordiamo che è possibile inserire nella pagina HTML delle entità, ovvero caratteri speciali, attraverso un particolare nome simbolico preceduto da "&" e seguito da ";". Le entità JavaScript seguono la stessa regola ma tra & e ; va inserita un'espressione racchiusa tra parentesi graffe. Supponiamo di aver definito una variabile *barWidth*. E` possibile creare una barra orizzontale della lunghezza percentuale specificata da essa con

<HR WIDTH="&{barWidth};%" ALIGN="LEFT">

Una volta che la pagina è stata visualizzata, il layout può cambiare solo se essa viene ricaricata. Si noti che a differenza delle entità standard che possono apparire ovunque in una pagina HTML, le entità JavaScript sono ammesse solo come valori per gli attributi. Ad esempio

<H4>&{myTitle};</H4>

visualizzerebbe myTitle e non il contenuto della variabile myTitle.

Modalità di inserimento in pagine web ed esecuzione: **sintesi**

Le istruzioni JavaScript possono essere inserite in modi diversi:

1. all'interno degli script (tra i tag <**SCRIPT**>);

2. caricandole **da un file esterno** specificando un file come sorgente JavaScript ; può essere utile quando questo script debba essere utilizzato su più pagine.

Le istruzioni JavaScript possono essere eseguite in modi diversi:

1. In seguito all'attivazione di un **evento**, solitamente come gestore di eventi all'interno di certi tag HTML .Gli eventi si possono attivare, forzandoli, anche all'interno degli script, come se fossero delle proprietà dell'oggetto:

Oggetto.*gestore_evento* = *handler_code*;

Ad un evento può essere associata un'unica istruzione, ma di solito l'associazione viene fatta con un **blocco di istruzioni**: tale funzione, attivata dal gestore di evento (event handler del tipo ***on****Nome_Evento*) è detta *handler code* facendo riferimento alla funzione implementata dall'utente che realizza la risposta all'evento.

2. in luogo di un **link** nella forma <A HREF= "javascript: *comando*">, inserendo un'espressione JavaScript come valore di un attributo HTML

3. valori JavaScript possono essere richiamati dall'HTML "*al volo*" includendoli tra i caratteri &{ e };%. Occorre osservare che ***sotto Internet Explorer*** queste espressioni ***non sono gestite***.