

Laboratorio di Informatica II

Riccardo Ortale



Javascript

Lezioni 7 - 8

GLI SCRIPTS

- ◆ Il World Wide Web si è sviluppato grazie alla possibilità di poter visualizzare la grafica e la multimedialità in rete.
- ◆ 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, tanto che svilupparono Javascript che venne implementato per la prima volta sulla versione beta di Netscape Navigator 2.0 nel giugno 1995.

GLI SCRIPTS

- ◆ Tale linguaggio apportò un notevole cambiamento alle pagine HTML, per cui alcuni effetti, che erano realizzabili soltanto con l'interfaccia CGI, divennero più facili da effettuarsi.
- ◆ Nel 1995, inoltre, la Sun Microsystems Inc., aveva presentato qualche mese prima Java, il linguaggio evoluto che si proponeva di diventare uno standard nella comunicazione in rete.

GLI SCRIPTS

- ◆ Nel 1996, però, la Microsoft iniziò a mostrare un grande interessamento per il Web, tentando di contrapporre a Javascript una versione ridotta del Visual Basic che prese il nome di VBScript.
- ◆ Microsoft per mantenere la compatibilità con Javascript ripiegò con Internet Explorer 3.0 verso l'adozione di un linguaggio che di fatto era molto simile a Javascript, ma che per esigenze di copyright, non poteva avere lo stesso nome, per cui fu definito JScript.

GLI SCRIPTS

- ◆ Vi sono altri linguaggi script, quali il Perl, ASP, PHP,...

JAVASCRIPT

- ◆ La caratteristica principale di Javascript, è quella di essere un linguaggio di scripting.
- ◆ I FILE vengono inviati in formato ASCII, quindi con codice in chiaro, che viene interpretato ed eseguito riga per riga dal browser in modalità runtime.

JAVASCRIPT

◆ L'HTML prevede un tag apposito per gli script, riportato di seguito:

```
<SCRIPT><!--  
    ...  
    Istruzioni  
    ...  
//--></SCRIPT>
```

JAVASCRIPT

- ◆ I browser ricevono le pagine HTML con tutto il contenuto, quando si incontra il tag `<SCRIPT>` questo viene processato come tutti gli altri tag, dall'alto in basso, ma il suo contenuto è interpretato secondo un codice diverso,
- ◆ in tal modo se il browser comprende il codice, questo viene eseguito, e se si incontra un errore durante l'interpretazione dello stesso i casi sono due:
 1. la pagina viene visualizzata, ma il codice errato non viene interpretato;
 2. se il codice genera un loop (cioè un ciclo infinito) la pagina resta bianca o è visualizzata parzialmente perché l'esecuzione dall'alto in basso del codice HTML è momentaneamente interrotta.

JAVASCRIPT

- ◆ Così come scritto, però, il tag `<SCRIPT>` non è completo perché i linguaggi di scripting sono diversi, allora occorre anche specificare il linguaggio:

<SCRIPT Language="Javascript"><!--

...

Istruzioni

...

//--></SCRIPT>

JAVASCRIPT

- ◆ Ciò potrebbe bastare, ma negli ultimi riferimenti, soprattutto da parte di Netscape, si consiglia vivamente di indicare anche la versione di Javascript che si adopera, soprattutto perché l'evoluzione del linguaggio è continua e non sempre assicura la compatibilità con i vecchi browser.
- ◆ Alla luce di quanto detto, il precedente script può essere considerato valido per la versione 1.0 di Javascript, e quindi per tutti i browser, mentre uno script del genere:

<SCRIPT Language="Javascript1.2"><!--

...

Istruzioni

...

//--></SCRIPT>

- ◆ diventa leggibile solo da Netscape 4.0 e Explorer 4.0 e dalle loro versioni successive.

JAVASCRIPT

- ◆ In linea di principio uno script può essere inserito in due modi all'interno di pagina HTML:
 1. inserendo il codice nel documento;
 2. caricandolo da un file esterno.

SCRIPT ESTERNI

- ◆ In quest caso lo script è salvato in un file con estensione .js. Viene richiamato con l'attributo **SRC** del tag **SCRIPT**:

```
<SCRIPT Language="Javascriptx.y" SRC="nomefile.js"><!--  
//--></SCRIPT>
```

- ◆ dove la specificazione di **Language** è facoltativa, poiché la stessa estensione del file basta a dimostrare il linguaggio adoperato. Tuttavia si consiglia di riportarla, al fine di identificare la versione di Javascript utilizzata. Il nome del file può essere indicato con un URL relativo o assoluto.
- ◆ Sintassi alternativa:

```
<SCRIPT type="text/javascript" SRC="nomefile.js"><!--  
//--></SCRIPT>
```

SCRIPT ESTERNI

- ◆ 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.
- ◆ Per tale motivo il file va salvato come testo ASCII, senza caratteri di controllo e senza tag HTML o elementi di altri linguaggi per non generare errori.
- ◆ Si può adoperare un qualsiasi editor molto semplice (in Windows è consigliato NotePad o Blocco Note).
- ◆ 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. Questo meccanismo funziona solo con Netscape 3.0 ed Explorer 4.0 e nelle versioni successive

SCRIPT ESTERNI

Esempio:

Scrivere con Blocco Note il seguente comando:

```
alert("Sono uno script esterno");
```

e salvarlo con il nome **prova.js**;

Scrivere in un altro file il seguente codice HTML:

```
<HTML><HEAD>  
<SCRIPT type="text/javascript" SRC="prova.js">  
</SCRIPT></HEAD>  
<BODY></BODY>
```

e salvarlo nella stessa directory del file Javascript;

Caricare la pagina HTML in un browser.

SCRIPT INTERNI

- ◆ 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>`).**
- ◆ Occorre tener presente che la pagina HTML viene eseguita in ordine sequenziale: dall'alto in basso, per cui la differenza tra le due alternative esiste:
 - **lo script dell'intestazione viene caricato prima degli altri,**
 - **quello nella sezione body, invece, viene eseguito secondo l'ordine di caricamento.**
- ◆ Cosa cambia tutto ciò? 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 a tutti gli script, quello che si trova nella sezione BODY è visibile soltanto agli script che lo seguono.**
 - **La scelta dipende anche da altri fattori (come la creazione della pagina HTML in maniera dinamica), ma sarà poi l'esperienza a suggerirli**

SCRIPT INTERNI

◆ Esempio

```
<HTML><HEAD>
<SCRIPT Language="Javascript">
x=1;alert('TESTA='+x);
</SCRIPT></HEAD>
<BODY>
<SCRIPT Language="Javascript">
x++;alert('CORPO='+x);
</SCRIPT>
<SCRIPT Language="Javascript">
x++;alert('CORPO='+x);
</SCRIPT>
</BODY>
```

BROWSER NON COMPATIBILI

- ◆ Potrebbero essere utilizzati anche browser non compatibili con Javascript oppure quelli in cui Javascript è disabilitato (è possibile con Netscape). In questo caso ci viene in aiuto il tag `<NOSCRIPT></NOSCRIPT>` che può contenere testo e grafica alternativi oppure un reindirizzamento in pagine statiche, che non adoperano Javascript, mediante la sequenza:

```
<NOSCRIPT>  
<META HTTP-EQUIV REFRESH CONTENT="0;  
URL=altrapagina.htm">  
</NOSCRIPT>
```

COMMENTI

- ◆ I commenti sono parti del programma che non vengono prese in considerazione dall'interprete e che, quindi, servono soltanto a chiarire il codice. Questi sono **racchiusi tra barre e asterischi** come nell'esempio sotto riportato:

```
/*commento*/
```

- ◆ Il commento può essere posto su più righe o su una riga singola

COMMENTI

- ◆ Un altro tipo di commento è la **doppia barra**, presa a prestito dal linguaggio C, ma è valida solo per commenti posti su una singola riga, anche se non la occupano per intero:

```
int x: //commento
```

COMMENTI

- ◆ I commenti Javascript non possono essere inseriti al di fuori dei tag che individuano lo script, altrimenti HTML li considererà come parte del testo, e viceversa non si possono utilizzare i tag di commenti HTML all'interno dello script. L'unico commento ammissibile è quello che consente **di racchiudere tutti gli script nei tag di commento di HTML**, facendoli aprire dopo il tag di script e chiudere prima della chiusura del tag:

```
<script language="JavaScript">  
  <!--  
    alert("Welcome!");  
  //-->  
</script>
```

- ◆ in tal modo **si maschera il codice javascript ai vecchi browser** che non lo leggono e si evita che l'HTML lo possa considerare come testo

SPAZI BIANCHI

- ◆ 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.

APICI

- ◆ **Importanti sono gli apici, sia singoli (' ') che doppi (" ").**
- ◆ **I doppi apici si adoperano per racchiudere parti di codice Javascript, e, insieme a quelli singoli, per racchiudere anche le stringhe** (sequenze di caratteri), per cui occorre fare attenzione ad annidare due stringhe racchiuse da apici simili, come ad utilizzare i doppi apici per le stringhe se questi già servono a racchiudere codice Javascript.
- ◆ 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 (\).

APICI

◆ Esempio

```
alert('Questo sito e' in costruzione')
```

La stringa viene
interrotta



- ◆ Produce un errore, interpretato in diverso modo da Netscape e da Internet Explorer. Per correggere l'errore è necessario scrivere:

```
alert('Questo sito e\' in costruzione')
```

LE ISTRUZIONI

- ◆ *Le istruzioni hanno la responsabilità di controllare il flusso di elaborazione del codice.* Esse possono:
 - 1. eseguire iterazioni**, cioè ripetere una parte di codice per un certo numero di volte;
 - 2. eseguire decisioni condizionate**;
 - 3. richiamare funzioni** (si vedrà in seguito cosa sono);
 - 4. consentire al percorso dell'esecuzione di essere modificato dinamicamente.**
- ◆ In Javascript ogni **istruzione inizia ad ogni nuova riga o con il punto e virgola**

LE ISTRUZIONI

◆ Esempio

Individuare le istruzioni nel seguente codice:

```
<SCRIPT Language="Javascript">  
x=1;alert ('PROVA');  
x++  
alert (RIPROVA);y=x  
</SCRIPT>
```

◆ *Le istruzioni sono cinque.*

BLOCCHI DI ISTRUZIONI

- ◆ Un'**istruzione composta**, invece, è formata da un gruppo di due o più istruzioni che formano un gruppo logico, nel senso che l'esecuzione delle stesse è legata, ad esempio, al verificarsi di una condizione: tali istruzioni sono **raggruppate in blocchi individuati dalle parentesi graffe**.

MODALITA' DI ESECUZIONE

- ◆ Le istruzioni in Javascript possono essere eseguite in diverso modo:
 1. **all'interno degli script**, individuati dai tag `<SCRIPT>`, in maniera sequenziale, per cui l'esecuzione è automatica;
 2. **caricandoli da file esterni**;
 3. **in seguito all'attivazione di un evento** (handler) come un click del mouse o la pressione di un tasto (si vedranno in seguito gli eventi);
 4. **in luogo di un link** (a partire da Netscape 3.0) nella forma: ``
 5. **valori Javascript** possono essere richiamati dinamicamente dall'HTML includendoli tra i caratteri `&{` e `};%` ad esempio la larghezza di una tabella può essere ricavata in rapporto ad un valore javascript nella forma `width="&{barWidth};%"`

MODALITA' DI ESECUZIONE

- ◆ Esempio
- ◆ 1. Scrivere il seguente codice e notare come gli script vengano eseguiti e in seguito a quali eventi:
 - ```
<HTML><HEAD></HEAD>
<BODY>
<SCRIPT Language="Javascript"><!--
alert('Script');
//--></SCRIPT>
<a href="#" onmouseover="alert('Hai
passato il mouse')">Passa il
mouse
<a href="javascript:alert('Hai
cliccato')">Clicca qui
</BODY></HTML
```

# EVENTI

- ◆ Gli *eventi* sono utilizzati per richiamare delle istruzioni. Infatti lo script va eseguito in maniera sequenziale, ma **per creare effetti di dinamicità ed interattività 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.

# EVENTI

- ◆ Gli eventi, per poter interfacciare HTML con Javascript, **non vengono definiti nel tag <SCRIPT>** (tranne che in qualche caso), **ma sono inseriti all'interno dei tag HTML.**
- ◆ Il browser compatibile con Javascript, incontrando la definizione di un evento, la interpreta ed attiva la gestione dello stesso.
- ◆ Ecco un esempio:

```
<A HREF="pagina.html"
onclick="alert('ciao') ">Link
```

# ATTIVARE GLI EVENTI ALL'INTERNO DEGLI SCRIPT

- ◆ Gli eventi, tuttavia, si possono anche attivare direttamente all'interno degli Script, richiamabili come se fossero una proprietà dell'oggetto. La sintassi è:

*Oggetto.evento=handler;*

- ◆ *handler* denota la funzione (eventualmente composta) con cui gestire l'evento.
- ◆ Esempio:

```
document.onload=alert("documento caricato");
```

# ATTIVARE GLI EVENTI ALL'INTERNO DEGLI SCRIPT

- ◆ Con Explorer, invece, si può utilizzare uno Script apposito per un oggetto e per un evento tramite la sintassi:

```
<SCRIPT FOR=Object
EVENT=evento>...</SCRIPT>
```

# RAGGRUPPARE GLI EVENTI

## 1. EVENTI DEL MOUSE O DELLA TASTIERA

1. Eventi attivabili dai tasti del mouse
2. Eventi attivabili dai movimenti del mouse
3. Eventi attivabili dal trascinamento del mouse (drag and drop)
4. Eventi attivabili dall'utente con la tastiera

## 2. EVENTI LEGATI AGLI OGGETTI

1. Eventi attivabili dalle modifiche dell'utente
2. Eventi legati al "fuoco"
3. Eventi attivabili dal caricamento degli oggetti
4. Eventi attivabili dai movimenti delle finestre
5. Eventi legati a particolari bottoni
6. Altri e nuovi tipi di eventi

# SINTASSI

- ◆ Prima della descrizione anticipiamo che tutti gli eventi hanno la propria sintassi, composta sintatticamente dal loro nome col prefisso **on**.
- ◆ Ad esempio l'evento **click** è richiamato con l'handler **onclick**.
- ◆ Il nome deve essere specificato interamente in minuscolo

## EVENTI LEGATI AI TASTI DEL MOUSE

1. **onclick**: attivato quando si clicca su un oggetto;
2. **ondblclick**: attivato con un doppio click;
3. **onmousedown**: attivato quando si schiaccia il tasto sinistro del mouse;
4. **onmouseup**: attivato quando si alza il tasto sinistro del mouse precedentemente schiacciato;

## VALORI TRUE E FALSE

- ◆ L'evento onclick con JavaScript 1.1 ha aggiunto la **possibilità di ricevere un valore true e false**, con cui può essere attivato o meno. Tale possibilità è valida anche per gli eventi onmousedown e onmouseup
- ◆ Esempio

```
<html>
<body>
<A HREF="http://www.itcgfontana.tn.it"
onclick="return(confirm('Sei sicuro?'))">
</body>
</html>
```

- Se si risponde *ok* il link si attiva.

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
onClick	Questo gestore è usato con i pulsanti di invio (submit), pulsanti di reset (reset), caselle di controllo (checkbox e radio), bottoni, tag <code>&lt;INPUT&gt;</code> di tipo OPTION e tag <code>&lt;A&gt;</code> .
onDbIclick	Usato con i tag <code>&lt;BODY&gt;</code> e <code>&lt;A&gt;</code>
onMouseDown	Usato con i bottoni e i tag <code>&lt;BODY&gt;</code> e <code>&lt;A&gt;</code>
onMouseUp	Usato con i bottoni e i tag <code>&lt;BODY&gt;</code> e <code>&lt;A&gt;</code>

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

# EVENTI LEGATI AL MOVIMENTO DEL MOUSE

1. **onmouseover**: attivato quando il mouse si muove su un oggetto;
2. **onmouseout**: attivato quando il mouse si sposta da un oggetto;
3. **onmousemove**: si muove il puntatore del mouse, ma poiché questo evento ricorre spesso (l'utilizzo del mouse è frequente), non è disponibile per default, ma solo abbinato con la cattura degli eventi.

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
onmouseover	Questo gestore è usato con i pulsanti di invio (submit), pulsanti di reset (reset), caselle di controllo (checkbox e radio), bottoni, tag <code>&lt;INPUT&gt;</code> di tipo OPTION e tag <code>&lt;A&gt;</code> .
onmouseout	Usato con i tag <code>&lt;BODY&gt;</code> e <code>&lt;A&gt;</code>
onmousemove	Usato con i bottoni e i tag <code>&lt;BODY&gt;</code> e <code>&lt;A&gt;</code>

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

# ROLLOVER

- ◆ 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'"
```

dove il cancelletto sostituisce qualsiasi altro link, mentre `num` è il numero di indice dell'immagine nella pagina HTML.

- ◆ Prima di Flash, il rollover era utilizzato per rendere interattive le pagine HTML, movimentandone gli elementi statici (menù e barre di navigazione).

# EVENTI LEGATI AL TRASCINAMENTO DEL MOUSE

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;
- ◆ Esistono altri eventi in questa categoria, che tuttavia vengono riconosciuti solo da Internet Explorer 5 e versioni successive.

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
<b>ondragdrop</b>	Questo gestore è usato con Window.
<b>onmove</b>	Questo gestore è usato con Window.

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

## EVENTI LEGATI ALLA TASTIERA

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;
3. **onkeyup**: evento attivato quando un tasto, che era stato premuto, viene rilasciato;
4. **onhelp**: attivato quando il mouse schiaccia il tasto F1 (riconosciuto solo da Internet Explorer 5 e versioni successive);

# EVENTI LEGATI ALLA TASTIERA

## ◆ Esempio

```
<html>
<body>
<SCRIPT LANGUAGE="JavaScript"><!--
function prova() {
var ieKey=event.keyCode
alert("Hai premuto il tasto: " + ieKey)
}
document.onkeydown = prova
//-->
</SCRIPT>
<p>Prova a premere un tasto</p>
</body>
</html>
```

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
<b>onkeypress</b>	Questo gestore è usato con i tag <BODY>, <IMG>, <A> e input TEXTAREA.
<b>onkeydown</b>	Questo gestore è usato con i tag <BODY>, <IMG>, <A> e input TEXTAREA.
<b>onkeyup</b>	Questo gestore è usato con i tag <BODY>, <IMG>, <A> e input TEXTAREA.

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

# TASTI INTERCETTABILI

- ◆ In Internet Explorer 4.0, l'evento **onkeydown** funziona con i tasti:
  - DELETE, INSERT
  - Tasti funzione da F1 - F12
  - Lettere: a - z
  - Tasti di spostamento: HOME, END, Left Arrow, Right Arrow, Up Arrow, Down Arrow
  - Numeri: 0 - 9
  - Simboli: ! @ # \$ % ^ & \* ( ) \_ - + = < > [ ] { } , . / ? \ | ' ` " ~
  - Tasti di sistema: ESCAPE, SPACE, SHIFT, TAB
- ◆ In Internet Explorer 5 funzionano anche i tasti:
  - BACKSPACE
  - PAGE UP, PAGE DOWN
  - SHIFT+TAB

# EVENTI LEGATI ALLE MODIFICHE

1. **onchange**: attivato quando il contenuto di un campo di un form o modulo è modificato o non è più selezionato;
- ◆ Internet Explorer 5 riconosce anche:
1. **onCellChange**: attivato quando si modifica un elemento in un Database, per questa sua caratteristica ha un uso non propriamente legato a Javascript;
  2. **onPropertyChange**: evento attivato quando cambia la proprietà di un elemento;
  3. **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.

# EVENTI LEGATI ALLE MODIFICHE

## ◆ Esempio

```
<html>
<body>
<form method="post" action="" name="prova">
<select name="prova"
onchange="if(this.options[1].selected) alert('Hai
selezionato il secondo');
else if(this.options[2].selected) alert('Hai
selezionato il terzo') ">
<option selected>primo</option>
<option>secondo</option>
<option>terzo</option>
</select>
</form>
</body>
</html>
```

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
onchange	Questo gestore è usato con i tag <SELECT> e <TEXTAREA>, e il tag <INPUT> di tipo TEXT. Con la versione Javascript 1.1 si aggiunge anche il tag <FILEUPLOAD>, ma non funziona con Internet Explorer.

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

## EVENTI LEGATI AL FUOCO

1. **onfocus**: Questo handler è l'opposto di **onblur**, per cui si attiva quando l'utente entra in un campo;
2. **onblur**: attivato quando un elemento perde il focus che aveva in precedenza. 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;

# TAG SENSIBILI

Evento	Tag associati in Netscape e Jscript
<b>onfocus</b>	Questo gestore è usato con i tag <code>&lt;SELECT&gt;</code> e <code>&lt;TEXTAREA&gt;</code> e con il tag <code>&lt;INPUT&gt;</code> di tipo <code>TEXT</code> . Con Javascript 1.1, cioè da Netscape Navigator 3, questo handler è stato associato anche con i tag <code>&lt;BODY&gt;</code> e <code>&lt;FRAMESET&gt;</code> e con il resto dei tag di form come <code>&lt;BUTTON&gt;</code> , <code>&lt;CHECKBOX&gt;</code> , <code>&lt;FILEUPLOAD&gt;</code> , <code>&lt;PASSWORD&gt;</code> , <code>&lt;RADIO&gt;</code> , <code>&lt;RESET&gt;</code> , <code>&lt;SUBMIT&gt;</code> . In Javascript 1.2 si aggiunge anche il tag <code>&lt;LAYER&gt;</code> .
<b>onblur</b>	Questo gestore è usato con i tag <code>&lt;SELECT&gt;</code> e <code>&lt;TEXTAREA&gt;</code> e con il tag <code>&lt;INPUT&gt;</code> di tipo <code>TEXT</code> . Con Javascript 1.1, cioè da Netscape Navigator 3, questo handler è stato associato anche con i tag <code>&lt;BODY&gt;</code> e <code>&lt;FRAMESET&gt;</code> e con il resto dei tag di form come <code>&lt;BUTTON&gt;</code> , <code>&lt;CHECKBOX&gt;</code> , <code>&lt;FILEUPLOAD&gt;</code> , <code>&lt;PASSWORD&gt;</code> , <code>&lt;RADIO&gt;</code> , <code>&lt;RESET&gt;</code> , <code>&lt;SUBMIT&gt;</code> . In Javascript 1.2 si aggiunge anche il tag <code>&lt;LAYER&gt;</code> .
<b>onselect</b>	Questo gestore è usato con il tag <code>&lt;TEXTAREA&gt;</code> e <code>&lt;INPUT&gt;</code> di tipo <code>TEXT</code> , anche per Internet Explorer.

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

# EVENTI LEGATI AL CARICAMENTO DEGLI OGGETTI

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: funziona su un link errato di un'immagine della pagina, ma non su un link errato di caricamento di una pagina intera;

# EVENTI LEGATI AL CARICAMENTO DEGLI OGGETTI

## ◆ Esempio

```
<html>
<body onload="alert('Evento Onload') "
onunload="alert('Evento Unload') ">
<script language="javascript">
document.onstop="alert('Evento onStop')";
</script>
<p>Prova: cliccare sul tasto Refresh.</p>
<p> Al termine chiudere la finestra con la x.
 </p>
</body>
</html>
```

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
<b>onLoad</b>	Questo gestore è usato con i tag <BODY> e <FRAMESET> e da Javascript 1.1 anche con <IMG> mentre in Explorer occorre aggiungere anche i tag <SCRIPT>, <LINK>, <EMBED>, <APPLET>. In Javascript 1.2 in Netscape si aggiunge anche il tag <LAYER>.
<b>onUnload</b>	Questo gestore è usato con i tag <BODY> e <FRAMESET> anche in Internet Explorer.
<b>onAbort</b>	Questo gestore è usato solo con il tag <IMG> anche in Internet Explorer.
<b>onError</b>	Questo gestore è usato solo con il tag <IMG> e con Window mentre in Internet Explorer anche con <OBJECT> e <STYLE>.

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

# EVENTI LEGATI AL MOVIMENTO DELLE FINESTRE

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, sia con i tasti PGUP e PGDOWN, o anche con il metodo `doscroll` (riconosciuto solo da Internet Explorer 5).

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
onResize	Questo gestore è usato con il tag <DOCUMENT>.
onScroll	Questo gestore è usato solo in Explorer e con i tag <APPLET>, <BDO>, <BODY>, <DIV>, <EMBED>, <MAP>, <MARQUEE>, <OBJECT>, <SELECT>, <TABLE>, <TEXTAREA>.

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

## EVENTI LEGATI A TASTI PARTICOLARI

1. **onsubmit**: Questo handler è attivato dal click su tasto di Invio di un form;
2. **onreset**: questo handler è attivato dal click su tasto di Annulla di un form.

# TAG SENSIBILI

<b>Evento</b>	<b>Tag associati in Netscape e Jscript</b>
<b>onsubmit</b>	Questo gestore è usato col tag <FORM> sia in Netscape che in Explorer.
<b>onreset</b>	Questo gestore è usato col tag <FORM> sia in Netscape che in Explorer.

## Tag associati Explorer

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

# VALORI LETTERALI

- ◆ **valori letterali sono quantità esplicite** per cui non vanno dichiarati e servono a fornire informazioni ad espressioni o a funzioni (es.: il numero 45 è esplicitamente considerato numerico).

# VALORI LETTERALI

- ◆ In Javascript esistono diversi tipi di valori che sono:
  1. **Numerici**, ulteriormente suddivisi in:
    1. Interi
    2. decimali o, con linguaggio tecnico, a virgola mobile (in notazione scientifica o in quella standard) (Valori non precisi in Netscape 2.0)
  2. **logici** o booleani: che possono assumere soltanto due stati (vero o falso);
  3. **stringhe** (o sequenza di caratteri)
  4. **valore nullo**
  5. **caratteri speciali** (es.: `\f` per l'avanzamento pagina)
  6. **oggetti**
- ◆ Importante è sottolineare che ad una variabile può essere assegnato anche un oggetto e si può anche creare un **oggetto generico** tramite l'enunciato:

```
nomeoggetto=new Object();
```

# CARATTERI SPECIALI

- ◆ Tra le stringhe occorre indicare i **caratteri speciali** che costituiscono un mezzo per formattare il testo:

Descrizione	Designazione standard	Sequenza
Continuazione	<newline>	\
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	"	\"

# CARATTERI SPECIALI

## ◆ Esempio

```
<html>
<body>
<script language="javascript">
document.write('L\'accento viene
 inserito come carattere speciale')
</script>
</body>
</html>
```

# DICHIARAZIONE VARIABILI

- ◆ Le variabili sono dei nomi simbolici che servono ad individuare delle locazioni di memoria, in cui possono essere posti dei valori. Metaforicamente bisogna considerare queste locazioni come delle scatole con un nome, in cui si possono inserire i suddetti valori.
- ◆ In Javascript le variabili vengono create con la **dichiarazione var** (assegnando loro eventualmente nessun valore), ma anche semplicemente con l'*atto di assegnazione di un valore* (ad esempio `x=15` crea automaticamente una variabile numerica). La dichiarazione **var** su più variabili va ripetuta per ognuna, oppure va fatta con un'interruzione di linea:

```
var miocarattere,
 miavariabile;
```

# DICHIARAZIONE VARIABILI

- ◆ Per le variabili dichiarata e non inicializzate assumono valore **null**.

```
var Verifica=null;
```

# VARIABILI GLOBALI E VARIABILI LOCALI

- ◆ Dove dichiarare le variabili? Dipende dall'utilizzo e dalla distinzione che c'è tra **variabili globali** e **variabili locali**. La distinzione non è poca cosa, anzi è alla base della programmazione orientata agli oggetti:
  1. le **variabili globali** hanno valore per tutto il documento HTML e vanno dichiarate all'inizio dello script e fuori da ogni funzione: il posto preferibile è nei tag `<SCRIPT>` della sezione `<HEAD>`, in modo tale da creare i contenitori dei valori prima di ogni loro utilizzo;
  2. le **variabili locali** hanno valore solo all'interno della funzione in cui sono dichiarate, cioè all'interno del blocco di codice compreso tra `function(){` e la chiusura della parentesi `}` e vanno dichiarate entro questi blocchi.

Da ciò si ricava che le *variabili dichiarate all'interno di parentesi graffe possono essere utilizzate solo in quel blocco*, perchè sono **variabili locali**.

# 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 l'interprete 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: 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*.

# PAROLE CHIAVE

- ◆ In Javascript ci sono delle parole chiave che non si possono utilizzare come identificatori di dati, eccone l'elenco:

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

# LIFETIME DEI DATI

- ◆ Per tenere conto di tali variabili nel passare da una pagina ad un'altra le uniche soluzioni sono:
  - utilizzo di cookie nel lato client;
  - passaggio tramite URL;
  - uso di campi nascosti;
  - passaggio utilizzando i frame e la forma **parent.frameName.variabile** o **parent.frameName.nomefunzione**;
  - passaggio utilizzando le finestre e la forma **nomefinestra.variabile** o **nomefinestra.nomefunzione**.

# ARRAY

- ◆ Gli array sono liste numerate di oggetti, che in Javascript possono anche essere di tipo diverso.
- ◆ Javascript non supporta gli array come variabili primitive, ma come oggetti. Pertanto, si crea un array **solo se lo si dichiara** e tale dichiarazione crea una **istanza dell'oggetto** Array:

*nome\_array=new Array(dimensione\_array)*

- ◆ Per accedere ad un elemento dell' array si utilizza l'indice che indica la posizione dell'elemento all'interno della stessa.
- ◆ Tale posizione si inizia a conteggiare da 0 e non da 1.

# ARRAY

## ◆ Esempio

```
<html>
<body>
<script language="javascript">
vettore = new Array(3)
vettore[0] = "1° elemento"
vettore[1] = "2° elemento"
vettore[2] = "3° elemento"
document.write(vettore[0])
document.write(vettore[1])
document.write(vettore[2])
</script>
</body>
</html>
```

# OPERATORI

- ◆ Gli operatori si dividono in:
  1. operatori di assegnamento;
  2. operatori aritmetici;
  3. operatori logici;
  4. operatori sui bit, adoperati in genere solo per generare colori;
  5. operatori su stringhe.
- ◆ possono essere **unari** o **binari**: i primi richiedono un unico operando, i secondi due.

# OPERATORI

- ◆ Il primo operatore che occorre conoscere è l'**operatore di assegnamento**, *il cui segno è l'uguale (=)*.
- ◆ Tuttavia il suo uso in Javascript è diverso da quello della matematica perchè **serve ad assegnare un valore e non ad effettuare confronti**, per i quali esiste un altro operatore (`= =`).
- ◆ Adoperando l'operatore di assegnamento si crea come una fotocopia del valore dell'espressione a destra dell'operatore sul dato a sinistra dell'operatore.

# OPERATORI UNARI

- ◆ Gli operatori unari **modificano il valore a cui sono applicati** e sono:

Operatore	Simbolo	Azione
Incremento un'unità	++	Incrementa di
Decremento	--	Decrementa di un'unità
Meno unario numero	-	Rende negativo un

# OPERATORI UNARI

- ◆ Gli operatori unari matematici *possono essere posti prima (prefissi) o dopo (posfissi)* dell'operando.
- ◆ 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.*
- ◆ Esempio:

	<code>x=10;</code>	<code>x=10;</code>
	<code>y=x++;</code>	<code>y=++x;</code>
per cui	<code>y=10</code>	<code>y=11</code>
e	<code>x=11</code>	<code>x=11</code>

# OPERATORI BINARI

- ◆ Gli operatori binari matematici **non cambiano il valore degli operandi, ma memorizzano il risultato in un terzo operando.**
- ◆ Comprendono le principali operazioni aritmetiche:

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

# OPERATORI BINARI

- ◆ L'operatore di assegnamento può anche essere **compattato**, cioè **abbinato ad un operatore aritmetico**. In genere quando le espressioni sono del tipo:

*variabile=variabile operatore espressione*

- ◆ possono essere cambiati in:

*variabile operatore = espressione*

- ◆ cioè si ha la seguente tabella:

## Scrittura compatta

$x += y$

$x -= y$

$x *= y$

$x /= y$

$x \% = y$

## Scrittura equivalente

$x = x + y$

$x = x - y$

$x = x * y$

$x = x / y$

$x = x \% y$

# OPERATORI RELAZIONALI

- ◆ Con **operatore relazionale** 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 sono:

## Operatore

## Azione

>

Maggiore di

>=

Maggiore o uguale

<

Minore di

<=

Minore o uguale

==

Uguale

!=

Diverso

- ◆ Gli operatori relazionali, come detto, producono 1 se la relazione è vera, e 0 se la relazione è falsa.

# OPERATORI LOGICI

- ◆ Gli **operatori logici** sono (il NOT è un *operatore unario*):

Operatore	Simbolo	Significato
AND valutazione	&&	AND
OR valutazione		OR
NOT	!	Negazione

- ◆ Esempio:

```
if (Nome=="Valeria") && (Cognome=="Rossi")
```

# OPERATORI SU STRINGHE

◆ Gli operatori su stringhe hanno gli operatori:

Operatore	Nome	Sintassi
+	Addizione	<code>stringa=stringaA+stringaB</code>
+=	Accoda	<code>stringaA+=stringaB</code> (ris. <code>stringaA=stringaA+stringaB</code> )
==	Uguaglianza	<code>if (stringaA==stringaB)</code>
!=	Disuguaglianza	<code>if (stringaA!=stringaB)</code>

# OPERATORI AL VOLO

- ◆ Le espressioni possono essere calcolate anche all'interno di tag HTML, attraverso il meccanismo di costruzione “al volo” di valori.
- ◆ La sintassi è:

*&{espressione};*

- ◆ Un esempio é:

```
<script>
<!--
 var tabWidth=50;
//-->
</script>
<table width="&{tabWidth};%">
```

- ◆ Tali espressioni sono importanti se vengono legate ad un valore che varia secondo opportune modalità.
- ◆ Internet Explorer 4.0 non gestisce questo tipo di espressione

# ESPRESSIONI CONDIZIONALI

- ◆ Il cuore di queste istruzioni è nella condizione, che ne influenza l'esecuzione. Per tale motivo l'istruzione deve essere formulata con attenzione.
- ◆ Ecco un esempio di espressione condizionale:

```
if (valore<10 && valore>0)
```

## IF..ELSE

- ◆ L'espressione base del controllo di flusso è:

*if(espressione) istruzione  
[else istruzione]...*

- ◆ dove l'espressione è booleana, cioè può assumere i valori true o false. L'istruzione principale può essere seguita dall'else, che indica un'istruzione alternativa da eseguire quando la prima non è verificata.

## IF..ELSE..ELSE

- ◆ Se occorre effettuare una serie di test si può iterare l'*else* in questo modo:

```
if (espressione) {istruzione}
else (espressione) {istruzione}
else (espressione) {istruzione}
else (espressione) {istruzione}.
else istruzione
```

- ◆ per cui si valuta l'espressione accanto all'*else* e viene eseguita se l'espressione è vera, altrimenti si esegue l'*else* finale.

# OPERATORE TERNARIO

- ◆ Molto utile è anche l'espressione condizionale ternaria che fonde l'if...else in un unico comando che è il **?**, conosciuto anche come **operatore ternario**.
- ◆ La sua forma è (fare attenzione ai due punti):

*Espressione1 ? Espressione2 :Espressione3*

*Esempio:*

*$x==1?y=1:y=2$*

# FOR

- ◆ Un altro potente strumento di iterazione di istruzioni è il loop **for**, che esegue una serie di istruzioni fino a che non è stato raggiunto il limite indicato da una condizione.
- ◆ La variabile assume spesso il nome di contatore e va inizializzata. Ad ogni passaggio questa viene incrementata e confrontata con la condizione: se falsa il ciclo avvia una nuova iterazione, se vera il ciclo termina. L'istruzione è:

```
for (inizializzazione; condizione;
incremento) istruzione;
```

- ◆ dove la prima espressione è un'inizializzazione della variabile contatore, la seconda espressione pone il limite massimo entro cui l'istruzione deve essere reiterata, mentre l'ultima espressione indica quanto incrementare o decrementare la variabile ad ogni iterazione.

# FOR

## ◆ Esempio:

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

WHILE

- ◆ Altro controllo di flusso si può avere con:

while (condizione) espressione

che esegue l'espressione finchè la condizione è vera.

DO..WHILE

- ◆ Se occorre eseguire il blocco di istruzioni almeno una volta ci viene in aiuto un altro controllo di flusso:

do {istruzioni} while (condizione)

- ◆ che esegue l'espressione finchè la condizione è vera.

FUNCTION

- ◆ Una funzione è un blocco di comandi. Questi comandi sono racchiusi tra le parentesi graffe.

- ◆ Sintassi:

```
function nomefunzione  
  (argomenti) {  
  comandi  
}
```

- ◆ In **argomenti** possono essere inseriti parametri da utilizzare nella funzione (se si hanno più argomenti devono essere separati da ,).

FUNZIONI

◆ Esempio:

```
<html>
<head>
<script language="JAVASCRIPT">
function controllo(numero) {
    vettore = new Array();
    vettore[0]="Uno";
    vettore[1]="Due";
    vettore[2]="Tre";
    alert("Hai schiacciato il pulsante n. " + vettore[numero]);
}
</script>
</head>
<body>
<form>
<input type="button" value="Pulsante uno" onclick="controllo(0)"><br>
<input type="button" value="Pulsante due" onclick="controllo(1)"><br>
<input type="button" value="Pulsante tre" onclick="controllo(2)"><br>
</form>
</body>
</html>
```

OBJECT ORIENTED PROGRAMMING

- ◆ Possiamo ora cominciare ad argomentare sulla OOP. Per la OOP il programma che si sta scrivendo è un mondo. 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 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 **PROPRIETA'**.
Ogni uomo può eseguire diverse operazioni (es. camminare) definite **METODI**.

OGGETTI

- ◆ Ricordando che il nostro mondo è il **browser**, bisogna ora scoprire da cosa è formato il nostro browser.
Gli oggetti del browser si dividono in tre grandi famiglie:
 - Gli oggetti del **Navigator**
 - Gli oggetti **interni**
 - Gli oggetti **riflessi dall' HTML**

OGGETTI NAVIGATOR

◆ **Gli oggetti Navigator**

sono quelli propri del browser:

- **window**, ovvero la finestra in cui compaiono i documenti;
- **document**, che sono i documenti veri e propri;
- **location**, cioè l'URL di un documento;
- **navigator**, che corrisponde al browser;
- **history**, che è la lista dei file visitati durante la sessione di lavoro.

PROPRIETA' DELL'OGGETTO WINDOW

L'oggetto window è l'oggetto gerarchicamente più rilevante, poiché ogni documento, per esistere, deve essere inserito in una finestra. Le proprietà dell'oggetto window sono elencate di seguito.

parent	A partire da NN2.0 vengono inserite le finestre multiple. Per poter fare un riferimento univoco al <i>frame</i> o finestra che ci interessa bisogna, quindi, utilizzare l'oggetto parent accompagnato dalla proprietà frame
frame	Frame è un array che contiene tutti i frame nel documento. Si può scrivere quindi <code>parent.frames[0]</code> per fare riferimento al primo frame che compare nel documento attivo.
self	frame corrente
top	frame principale

PROPRIETA' DELL'OGGETTO WINDOW

defaultStatus	Fa riferimento al messaggio di default che compare nella barra di stato del Navigator
status	Contiene il messaggio della barra di stato
length	Contiene il numero di frames presente nel documento
name	Contiene il titolo della finestra corrente

METODI DELL'OGGETTO WINDOW

confirm()	apre una finestra di conferma
alert()	apre una finestra di messaggio
prompt()	apre una finestra di input
open()	apre una nuova finestra
close()	chiude una finestra
setTimeout	ottiene un effetto a tempo in una finestra
clearTimeout	rimuove il timeout

METODI DELL'OGGETTO WINDOW

◆ Esempio:

```
<html>
<body>
<form>
<input type="button" value="alert()"
  onClick="alert('Attenzione\n mi hai
  schiacciato!!!') "></input>
<input type="button" value="confirm()"
  onClick="confirm('Vuoi continuare con gli
  esempi?') "></input>
<input type="button" value="prompt()"
  onClick="domanda = prompt('Come ti
  chiami?', ''); conf = confirm('Sei sicuro di
  chiamarti ' + domanda + '
  ?'); if(conf) alert('Benvenuto\n' +
  domanda); "></input>
</form>
</body>
</html>
```

PROPRIETA' DELL'OGGETTO DOCUMENT

- ◆ La maggior parte delle proprietà di questo oggetto sono dei riflessi del linguaggio HTML.

title	Contiene il titolo del documento
alinkColor	Visualizza il colore dei link attivi (quelli selezionati)
linkColor	Visualizza il colore dei link presenti nel testo
vlinkColor	Visualizza il colore dei link visitati
fgColor	Visualizza il colore del testo
bgColor	Visualizza il colore di background del documento

PROPRIETA' DELL'OGGETTO DOCUMENT

links	Array di tutti i link contenuti nel documento
anchors	Array di ancore del documento
forms	Array dei forms presenti nel documento
images	Array delle immagini presenti nel documento
location	Contiene l'URL del documento
lastModified	Contiene la data in cui il documento è stato modificato
referer	Memorizza l'URL del documento contenente il link al documento

METODI DELL'OGGETTO DOCUMENT

◆ I metodi di **document** sono:

open()	Aprire il documento
close()	Chiudere il documento
write()	Consente di " <i>scrivere</i> " dinamicamente i documenti
writeln()	Consente di " <i>scrivere</i> " dinamicamente i documenti

METODI DELL'OGGETTO DOCUMENT

◆ Esempio

```
<html>
<head>
<title>Proprietà del documento</title>
</head>
<body>
<script language="javascript">
document.write("Prova metodi document <br>")
document.write("Il documento ha indirizzo : " +
    document.location + "<br>")
document.write("Il documento ha titolo: " +
    document.title + "<br>")
</script>
</head>
</body>
</html>
```

PROPRIETA' DELL'OGGETTO LOCATION

- ◆ Esso contiene le specificazioni della location del Navigator:

host	Computer al quale si è collegati
hostname	Computer al quale si è collegati
href	Indirizzo della pagina che si sta visualizzando
port	Porta usata per il trasferimento dei dati da parte del server
protocol	Protocollo usato per la trasmissione
target	Contiene l' attributo target di <i><a href></i>

PROPRIETA' DELL'OGGETTO HISTORY

length

contiene il numero di documenti visitati durante la sessione di lavoro.

METODI DELL'OGGETTO HISTORY

back()	Permette di caricare il documento visitato precedentemente
forward()	Permette di caricare il documento prossimo (pulsante forward)
go()	Permette di andare allo <i>n-esimo</i> documento visitato

METODI DELL'OGGETTO LOCATION

◆ Esempio

```
<html>
<body>
<form><input type="button" value="Host" onClick="alert('host
    al quale sei collegato\n'
+document.location.host) "></input><br>
<input type="button" value="hostName"
    onClick="alert('hostname al quale sei
collegato\n'+document.location.hostname) "></input><br>
<input type="button" value="href" onClick="alert('href del
    documento
attuale\n'+document.location.href) "></input><br>
<input type="button" value="Port" onClick="alert('Porta di
    comunicazione del
server\n'+document.location.port) "></input><br>
<input type="button" value="Protocol"
    onClick="alert('Protocollo del
documento\n'+document.location.protocol) "></input></form>
</body>
</html>
```

OGGETTI INTERNI

- ◆ **Array** Oggetto che contiene la struttura degli Array
- ◆ **Object** Oggetto generico per molti versi simile a Array
- ◆ **String** Oggetto di tipo stringa
- ◆ **Math** Oggetto con proprietà e metodi matematici
- ◆ **Date** Oggetto di tipo data

String

- ◆ Proprietà:
 - **length**: Numero di caratteri contenuti nella stringa
- ◆ Metodi:

toLowerCase()	trasforma i caratteri di una stringa in minuscoli
toUpperCase()	trasforma i caratteri di una stringa in maiuscoli
charAt(n)	restituisce il carattere in <i>n-esima</i> posizione
indexOf(carattere)	restituisce la posizione del primo carattere incontrato nella stringa
lastIndexOf(carattere)	restituisce la posizione dell'ultimo carattere incontrato nella stringa
substring(inizio , fine)	restituisce la sottostringa che va dal carattere nella posizione inizio al carattere nella posizione fine

String

◆ Esempio:

```
<html>
<script language="JAVASCRIPT">
parola=prompt("Inserisci una parola in
    minuscolo", "")
alert("Parola in maiuscolo: " +
    parola.toUpperCase())
</script>
</html>
```

Math

◆ Proprietà:

E	Contiene il valore della costante e
PI	Contiene il valore della costante <i>pi greco</i>
LN2	Contiene il valore della costante del logaritmo naturale di 2
LN10	Contiene il valore della costante del logaritmo naturale di 10
LOG2_E	Contiene il valore del logaritmo di e in base 2
LOG10_E	Contiene il valore del logaritmo di e in base 10

Math

◆ Metodi:

abs()	Restituisce il valore assoluto
acos()	Valore dell' arco coseno
asin()	Valore del seno dell' arco
atan()	Valore della tangente all' arco
ceil()	Restituisce il numero intero successivo (Se numero interno = intero, ceil() darà lo stesso numero)
cos()	Coseno
sin()	Seno
tan()	Tangente
exp()	Esponenziale
floor()	Numero intero precedente
log()	Logaritmo naturale
max(valore1, valore2)	Numero più grande tra i due valori
min(valore1, valore2)	Numero più piccolo tra i due valori
pow(valore1, valore2)	Valore1 elevato valore2
random()	Numero casuale tra 0 e 1
round()	Arrotonda al più vicino numero intero
sqrt()	Radice quadrata

Math

◆ Esempio:

```
<html>
<script language="JAVASCRIPT">
a=prompt("Inserisci il primo numero","")
b=prompt("Inserisci il secondo
numero","")
alert("Il maggiore dei due è: " +
Math.max(a,b) )
</script>
</html>
```

Date

- ◆ Un oggetto Date viene creato con

nomeQualsiasi = new Date ().

Se le parentesi vengono lasciate vuote *nomeQualsiasi* è istanziato con i dati riguardanti Mese, Giorno, Anno, Ore, Minuti, Secondi correnti.

- ◆ Alternativamente, all' interno delle parentesi, si può specificare una data. Tale data deve avere il seguente formato:

"November 13, 1973 08:30:00"

"73,11,13,08,30,00"

Date

◆ Metodi:

getDate()	Restituisce la data
getDay()	Restituisce il giorno (0 = domenica)
getMonth()	Restituisce il mese (0 = gennaio)
getFullYear()	Restituisce l'anno
getHours()	Restituisce l'ora
getMinutes()	Restituisce i minuti
getSeconds()	Restituisce i secondi

Date

◆ Metodi:

setDate()	Imposta la data
setDay()	Imposta il giorno (0 = domenica)
setMonth()	Imposta il mese (0 = gennaio)
setYear()	Imposta l'anno
setHours()	Imposta l'ora
setMinutes()	Imposta i minuti
setSeconds()	Imposta i secondi

Date

◆ Metodi:

toGMTString	per convertire una data in stringa, secondo le convenzioni GMT
toLocaleString	per convertire una data in stringa, secondo le convenzioni locali

Date

◆ Esempio

```
<html>
<script language="JAVASCRIPT">
Oggi = new Date()
alert("Giorno: " + Oggi.getDay() + "\n" +
      "Data: " + Oggi.getDate() + "\n" +
      "Mese: " + Oggi.getMonth() + "\n" +
      "Anno: " + Oggi.getYear() + "\n" +
      "Ora: " + Oggi.getHours() + "\n" +
      "Minuto: " + Oggi.getMinutes() + "\n" +
      "Secondo: " + Oggi.getSeconds() + "\n")
</script>
</html>
```

Date

◆ Esempio:

```
<html>
<script language="JAVASCRIPT">
Oggi = new Date("1/1/2001 0:00:00")
alert("Giorno: " + Oggi.getDay() + "\n" +
>Data: " + Oggi.getDate() + "\n" +
>Mese: " + Oggi.getMonth() + "\n" +
>Anno: " + Oggi.getYear() + "\n" +
>Ora: " + Oggi.getHours() + "\n" +
>Minuto: " + Oggi.getMinutes() + "\n" +
>Secondo: " + Oggi.getSeconds() + "\n")
</script>
</html>
```