JAVASCRIPT

Java este un limbaj de programare orientat obiect, dezvoltat de firma *Sun Microsystems*, în timp ce *JavaScript* are o structură bazată pe obiecte şi a fost dezvoltat de firma *Netscape Communications*. Limbajul *Java* poate fi folosit pentru proiectarea de aplicaţii independente sau pentru proiectarea de microaplicaţii care pot fi incluse în cadrul paginilor Web şi care poartă denumirea de **applet**-uri şi sunt şi ele independente de platformă. Iată câteva deosebiri esenţiale între *Java* şi *JavaScript*:

- aplicaţiile *Java* sunt compilate în fişiere binare care sunt apoi interpretate de către ***VJM*** (***V****irtual* ***J****ava* ***M****achine* - *Maşină Virtuală Java*), în timp ce codul *JavaScript* este transmis ca un text obişnuit şi este interpretat.
- *Java* este orientat obiect în timp ce *JavaScript* este bazat pe obiecte.
- codul *JavaScript* este inclus în cadrul documentului HTML; applet-urile *Java* sunt referite din cadrul unui document HTML, dar codul se află într-un fişier separat.
- script-urile *JavaScript* sunt incluse prin intermediul marcajului <script type=‘‘text/javascript”>, iar applet-urile Java prin intermediul marcajului <**OBJECT**>; există şi marcajul <**APPLET**> dar folosirea lui nu mai este recomandată.
- în cadrul script-urilor *JavaScript* variabilele nu trebuie declarate, în timp ce în cadrul aplicaţiilor *Java* variabilele trebuie declarate înainte de a fi folosite. Mai mult, în cadrul script-urilor *JavaScript*, o variabilă care este considerată ca având un anumit tip la un moment dat poate fi considerată ulterior ca având un alt tip; de exemplu, o variabilă poate fi folosită ca fiind un şir de caractere pentru ca apoi să fie considerată ca fiind un număr real.
- JavaScript foloseşte legarea dinamică, adică referinţele sunt verificate în timpul rulării, în timp ce Java foloseşte legarea statică, adică referinţele trebuie să existe în momentul compilării.
- În general, limbajul *JavaScript* este folosit pentru efectuarea de calcule, citirea unor date dintr-o tabelă, proiectarea de ecrane HTML fără a folosi script-uri CGI.

Marcajul <script type=‘‘text/javascript”>

În interiorul unui document HTML, instrucţiunile *JavaScript* sunt cuprinse, de obicei, în interiorul marcajului <script type=‘‘text/javascript”>. Acest marcaj poate apărea atât în antetul documentului HTML (marcajul <**HEAD**>), cât şi în corpul documentului (marcajul <**BODY**>). Script-urile definite în antet sunt încărcate înaintea încărcării restului paginii, antetul fiind un loc excelent pentru plasarea funcţiilor JavaScript pentru a fi siguri că acestea vor fi disponibile atunci când sunt apelate în alte secţiuni ale documentului HTML. Pentru a insera cod JavaScript într-un document HTML deja existent, este necesara introducerea în fisier a etichetei <script type=‘‘text/javascript”> ..... </script>. Aceasta eticheta are aributul ‘‘type”, sau ‘‘language” (acesta din urma este depreciat in standardul XHTML) care va specifica browserului limbajul folosit pentru interpretarea codului inclus. In interiorul etichetei <script type=‘‘text/javascript”> ... </script> vom scrie codul sursă.

Exemplu de script-ul prin intermediul căruia poate fi afişat mesajul ‘‘Hello World!” în fereastra programului de navigare.

```
<!DOCTYPE html PUBLIC ‘‘-//W3C//DTD XHTML 1.0 Strict//EN”
‘‘http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>
<html xmlns=‘‘http://www.w3.org/1999/xhtml” lang=‘‘en”>
<head>
<meta http-equiv=‘‘Content-Type” content=‘‘text/html; charset=utf-8” />
<title>Test JavaScript</title>
<title>Cod JavaScript</title>
</head>
<body>
   <script type=‘‘text/javascript”>
      document.write(”Hello World! ‘‘);
   </script>
</body>
</html>
```

Instructiunile JavaScript se pot introduce şi într-un alt fisier extern, care va avea extensia ‘‘.js”, iar pentru editarea acestui fisier se poate utiliza un editor simplu de texte. Avantajul este că se poate utiliza acelaşi cod în mai multe pagini HTML.
Dacă codul JavaScript se afla într-un fisier extern, eticheta
<script>
va trebui să contina să atributul **‘‘src”** a carui valoare determina locatia fisierului in care se afla codul JavaScript. In fisierul extern cu extensia ‘‘js” nu trebue să scriem eticheta ‘‘<script type=‘‘text/javascript”>‘‘, se scriu direct instructiunile scriptului. Dacă fişierul extern este info.js, atunci fişierul HML care apelează fişierul extern .JS arată:

```
<!DOCTYPE html PUBLIC ‘‘-//W3C//DTD XHTML 1.0 Strict//EN”
‘‘http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>
<html xmlns=‘‘http://www.w3.org/1999/xhtml” lang=‘‘en”>
<head>
<meta http-equiv=‘‘Content-Type” content=‘‘text/html; charset=utf-8” />
<title>Test JavaScript</title>
<title>Cod JavaScript</title>
<script src=‘‘info.js” type=‘‘text/javascript”>
</script></head>
<body>....
</body>
</html>
```

Fisierele JavaScript externe nu pot conţine decât declaraţii şi funcţii JavaScript, iar extensia trebuie să fie .js.

Comentarii în Javascript

La versiuni mai vechi de browsere, instrucţiunile JavaScript sunt scrise între comentarii. Sintaxa generala este:
<script type=‘‘text/javascript”>
<! - - Inceput comentariu

Instructiuni JavaScript
<! - - Sfarsitul comentariului - - >
</SCRIPT>
Proiectanţii introduc instrucţiunile JavaScript între comentarii HTML astfel încât browserele care nu permit JavaScript vor ignora instrucţiunile, interpretându-le drept comentarii. Comentariile în JavaScript pt fi definite ca în (applet-urile) JAVA, C şi C++ astfel:

```
/* Comentariu pe mai multe linii
*/
// Comentariu pe o singură linie
```

2.3 Elementele limbajului JavaScript

- **Numerele** care pot fi: numere întregi şi numere reale ,
- **Boolean** ce are valorile *true* şi *false*,
- **null** este o valoare nedefinită,
- **undefined**: o proprietate de nivel superior poate avea valoarea undefined,
- **NaN**: Not a Number,
- **String** 'text'. Pentru orice literal de tip string se utilizează metodele clasei **String**.
- **Vectori** şi elemente de tip **Obiect** .

Elementele limbajului pot fi:

JavaScript este un limbaj "dinamic-tipizat" sau "slab-tipizat" (loosely-typed), ceea ce înseamnă că nu trebuie să specific tipul unei variabile când o declar fiindcă ea va fi convertită automat dacă va fi nevoie în timpul execuţiei scriptului. O variabilă se poate declara în două moduri:

- atribuindu-i o valoare: x=124
- cu ajutorul cuvântului rezervat var: var x=124

O variabilă JavaScript căreia nu i-a fost atribuită o valoare va avea valoarea undefined. Rezultatul evaluării unei variabile depinde de modul în care a fost declarată, astfel:
- dacă variabilei nu i-a fost atribuită o valoare şi a fost declarată fără var rezultatul evaluării variabilei va fi o eroare runtime.
- dacă variabilei nu i-a fost atribuită o valoare, dar a fost declarată folosind cuvântul rezervat var rezultatul evaluării variabilei va fi undefined sau NaN în context numeric.

Variabilele, definite în afara funcţiilor, se numesc variabile globale şi ele sunt accesibile oriunde în documentul curent. Variabilele, definite în interiorul funcţiilor, se numesc variabile locale şi ele sunt vizibile numai în interiorul funcţiilor în care sunt declarate.

Şiruri (de caractere) JavaScript

Într-un scenariu JavaScript, şirurile de caractere constante se delimitează între apostrofuri simple sau duble.

Operatorul '+' semnifica concatenarea şirurilor de caractere. Ca şi în C, JAVA, şirurile de caractere pot conţine secvenţe Escape: \n, \t, \f etc.

În JavaScript sunt admise urmatoarele categorii de expresii: aritmetice,şiruri de caractere şi logice.

Operatorii utilizaţi de limaj pot fi:

Matematici:

+, -, *, /, ^, %(modulo)

- unari: se aplică la un singur operand: a++, a—
- binari: se aplică la 2 operanzi:

  a+b, a-b

  a+=2 echivalent cu a=a+2

De comparaţie, folosiţi pentru scrierea unor expresii logice. La evaluarea acestora se poate obţine **true** sau **false**: <, >, <=, >=, ==, !=

Există în JavaScript o serie de funcţii predefinite cum ar fi:

- *eval*(expr): evaluează o expresie data ca şi parametru (string);
- *isFinite*(number) : determină dacă argumentul este un număr finit;
- *isNaN*(testValue) : determina dacă testValue este NaN;
- *parseFloat*(str) : transformă stringul str în valoare float şi o returnează;
- *parseInt*(str [, radix]) : transformă *str* într-o valoare de tip întreg şi o returnează; radix este baza de numeraţie;
- *Number*(obj) : converteşte obj la number;
- *String*(obj) : converteşte obj la string;
- *escape/unescape*(str) : folosite pentru codificare/decodificare stringuri.

Preluarea de intrări tip text de la utilizator

În cazul când este nevoie de citirea unei linii text de la utilizator, se poate utiliza funcţia **prompt** pentru afişarea unei casete de dialog care oferă utilizatorului un prompter (cursor) pentru a introduce date şi pentru preluarea ulterioara a intrării utilizatorului. Funcţia arată **prompt (text, [valoare iniţiala])** şi provoacă apariţia unei casete în care utilizatorul va putea introduce un şir de caractere. Caseta conţine şi un buton 'OK' . Controlul de editare este iniţializat cu *valoare iniţială*, dacă acest argument este prezent, altfel este iniţializat cu textul (undefined)/

Funcţia EVAL se utilizează pentru a converti un şir de caractere la o valoare întreagă.

Funcţia confirm() are rolul de a crea o fereastră prin intermediul căreia se cere confirmarea utilizatorului pentru efectuarea unei acţiuni. Funcţia returnează o valoare logică (true sau false), iar sintaxa ei este **confirm(mesaj).**
Fereastra creată are două butoane: ***Ok*** şi ***Cancel***, nu poate fi minimizată sau redimensionată şi are un buton special pentru închidere. În cazul apăsării butonului ***Ok*** sau a tastei ENTER valoarea returnată este true, iar în cazul apăsării butonului ***Cancel***, a butonului de închidere sau a tastei ESC valoarea returnată este false .

Obiectul document
În JavaScript, obiectul **document** corespunde documentului HTML curent. Daca un script utilizează metodă **write** pentru obiectul **document** în vederea afişării ieşirii, browserul va reda ieşirea în documentul curent.Exemple de utilizare:

document.write(”text”)
document.write(variabilă)

Putem intercala cod JavaScript cu cod HTML. În **document.write** se pot utiliza etichetele HTML cu condiţia de a fi scrise între ghilimele. Dacă în cadrl ghilimelelor amen nevoie de alte ghilimele atunci pentru cele din urmă se utilizeazo apostrof. De exemplu, putem afişa texte scrise cu fonturi bold, respectiv italic ca în exemplu alăturat.

```
<html> <head>
<title> document.write </title>
</head>
<body>
<script type=‘‘text/javascript”>
document.write(”<b>bold</b> <i>italic</i>‘‘);
</script>
</body> </html>
```

Instrucţiuni JavaScript
Instrucţiunile JavaScript pot fi terminate să nu cu caracterul ;. Acest caracter este obligatoriu doar atunci când sunt mai multe instrucţiuni pe acelaşi rând.
Instrucţiunea de atribuire a fost utilizată şi în exemple anterioare şi este cea cunoscută din C şi Java. *Instrucţiunea IF este o* instrucţiunea condiţională şi foloseşte cuvintele cheie *if şi else* având următoarea sintaxă:

```
if (condiţie)
  {
instrucţiuni pentru condiţie adevărată
}
else
  {
instrucţiuni pentru condiţie falsă
}
```

Să se sciteasca numele şi parola unui utilizator iar în funcţie de acestea să se afişeze un anumit mesaj (figura 2.1).

```
<script type=‘‘text/javascript”>
nume=prompt(”Introduceti numele”,””);
parola=prompt(”Introduceti parola”,””) ;
n_valid=‘‘student” ;
p_valid=‘‘utm” ;
if((nume==n_valid)&&(parola==p_valid))
{
alert(”Date corecte!”) ;
document.write(”Bun venit pe pagina studentilor de la facultatea...”) ;
}
else
{
alert(”Date incorecte!”) ;
document.write(”Apasati F5 şi mai incercati !!!...”) ;
}
</script>
```

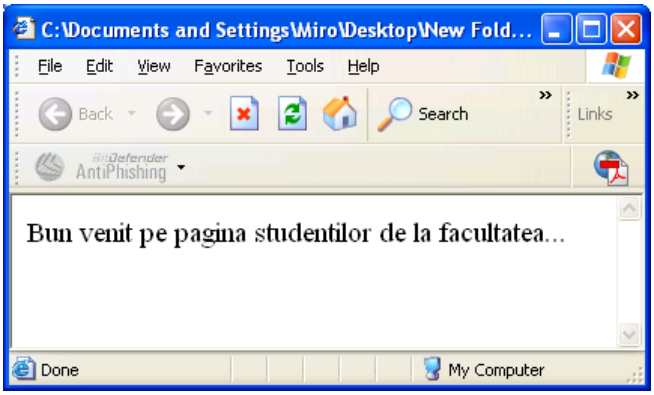

Figura 2.1

Instrucţiuni de ciclare

Adeseori se doreşte executarea repetată a mai multor instrucţiuni. Acest lucru este posibil, cu ajutorul instrucţiunilor de ciclare. În *JavaScript* există instrucţiunile de ciclare: **for**, **while şi do...while**. Sintaxa instrucţiunii **for** este:

```
for (iniţializare; condiţie; incrementare)
{
corpul ciclului;
}
```

Oricare dintre cele patru secţiuni (de iniţializare, condiţia, de incrementare sau corpul) poate lipsi.

Sintaxa instrucţiunii **while** este:

```
while (condiţie)
{
corpul ciclului
}
```

- În cazul instrucţiunii **while** poate lipsi corpul ciclului, prezenţa condiţiei fiind obligatorie. Exemplu:

```
<script type=‘‘text/javascript”>
i=””;
while(i!==‘‘student”)
{
i=prompt(”Introduceti parola”,””);
}
document.write(”Password accepted”);
</script>
```

Instrucţiunea SWITCH

Poate fi folosită pentru alegerea unei opţiuni din mai multe opţiuni.
Sintaxa instrucţiunii este:

```
switch (expresie)
{
case eticheta : instrucţiune; break;
case eticheta: instrucţiune; break;
...
default : instrucţiune;
}
```

```
Exemplu:  <script type=‘‘text/javascript”>
opt=eval(prompt(”Introduceti nr zilei”,”” ))
switch (opt)
{
case 1: zi=‘‘luni”;break
case 2: zi=‘‘marti”;break
case 3: zi=‘‘miercuri”;break
case 4: zi=‘‘joi”;break
case 5: zi=‘‘vineri”;break
case 6: zi=‘‘sambata”;break
case 7: zi=‘‘duminica”;break
default: zi=‘‘Introduceti un nr între 1 şi 7”
}
document.write(zi)
</script>
```

Instrucţiunile break şi continue

Instrucţiunea **break** permite ieşirea forţată dintr-o instrucţiune de ciclare. În momentul în care interpretorul *JavaScript* întâlneşte o astfel de instrucţiune, el întrerupe execuţia ciclului în interiorul căruia se află aceasta.

Instrucţiunea **continue** permite saltul peste anumite instrucţiuni din corpul ciclului care nu mai trebuie executate în anumite condiţii. În cazul unui ciclu for, se trece în mod automat, la ultimul pas al execuţiei ciclului, în timp ce în cazul unui ciclu while se trece la primul pas.

Folosirea tablourilor de elemente în JS
Un tablou de elemente (şir) este o structură complexă care înglobează mai multe variabile de acelaşi tip sub un nume. Un şir de elemente are o anumită dimensiune, are un anumit număr de elemente, fiecare element având o valoare şi este identificat prin poziţia în cadrul şirului.
Un şir se defineşte prin: **numeşir = new Array()**.

Funcţii şi evenimente în Java Script

O funcţie este văzută ca un bloc de instrucţiuni identificat printr-un nume care poate primi anumite argumente şi întoarce o valoare. Sintaxa unei funcţii este:

```
function numefuncţie (listă parametri formali)
{
instrucţiuni;
return (valoare)
}
```

Apelul funcţiei se realizează **numefuncţie(parametri efectivi).** Funcţiile sunt definite de obicei în zona HEAD şi sunt apelate apoi, ori de câte ori este nevoie, în BODY.

F(x)=x+1 unde x – parametru formal,
F(2)=3 unde 2 – parametru efectiv.

```
function suma (a,b) { s=a+b;return s;  }
```

Se poate apela **x=suma (1,1).**

Exemplu: funcţia **titlu** (parametru), să afişeze parametru aceasteia cu font albastru de mărime 5 (figura 2.2).

```
<script type=‘‘text/javascript”>
function titlu (a)
{document.write(“<font color=red size=7>‘‘+a+”</font>‘‘)}
x=prompt(“Introduceti titlul”,””);titlu(x);
</script>
```

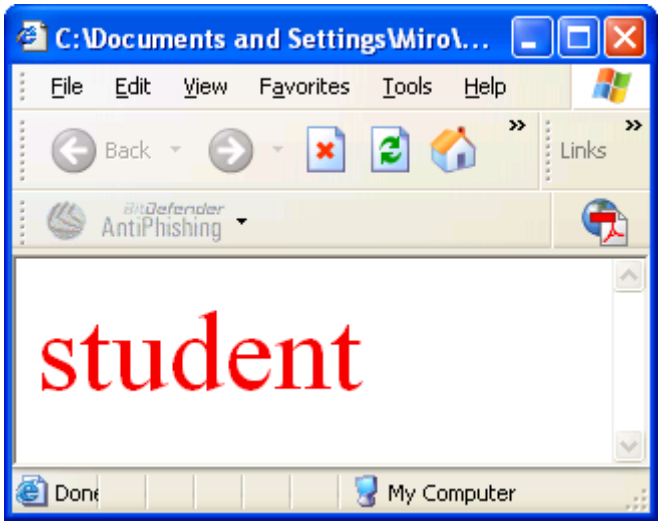

Figura 2.2 Text scris prin aplelul unei funcţii

Funcţiile pot fi definite şi fără parametri şi pot returna sau nu o valoare.

```
<script type=‘‘text/javascript”>
function titlu ()
{document.write(”<font color=red size=7>‘‘+a+”</font>‘‘);}
a=prompt(”Introduceti titlul”,””);
titlu();</script>
```

Utilizarea casetelor de validate

Conţinutul enei casete de validare se preia cu propietatea value care se aplica casetei astfel: *numecaseta.value*. Daca caseta face parte dintr-un formular atunci continutul casetei se accesează: *numeformular.numecaseta.value*.

Să se preia dintr-un formular numele şi parola, să se verifice şi să se afişeze un mesaj într-o altă casetă a formularului (figura 2.3).

```
<html>
<head>
<title> Formular butoane java – casete text </title>
<script>
function afisare(){
// pot să pun varibile formale
a=form1.t1.value;
b=form1.t2.value;
if (a==”123” && b==”abc”) form1.t3.value=”Corect!”;
else
if(a==”123” && b!=”abc”) form1.t3.value=”parola incorecta!”;
else
if(a!=”123” && b==”abc”) form1.t3.value=”user incorrect!”;
else
form1.t3.value=”user incorrect! şi parola incorecta”;
}
function sterge(x)
{x.value=””}
// “\n” trece la randul urmator
// sintaxa if: if(conditie) executa o secventa s1; else executa secventa s2
// observatie: daca conditia are 2 elem se grupeaza inte {}
```

```
// secventa s2 poate să lipseasca şi atunci se reduce la if(conditie) S1;
</script>
</head>
<body><form name="form1">
<h1><center> casete de text</center></h1>
User: <input type="text" name="t1" value="" onclick="sterge(t1)" size="40" >
<br><br><br>
parola: <input type="password" name="t2" value="" onclick="sterge(t2)">
<br><br>
<br>mesaj: <input type="text" name="t3" value="" onclick="sterge(t3)" size="40">
<input type="button" name="b1" value="testare" onclick="afisare()">
</form></BODY>
</html>
```

Figura 2.3. Testare user şi parola

Utilizarea casetelor de tip checkbox

Pentru o casetă de tip checkbox starea acesteia se preia cu proprietate checked.

Utilizarea butoanelor de tip radio

Dacă butoanele de tip radio fac parte din acelaşi grup, atunci ele sunt gestionate într-un vector, astfel încât primul element este pe poziţia zero. Proprietatea utilizată pentru a testa starea elementului (dacă a fost selectat sau nu) este tot checked, asemănător ca la casetele de tip ckeckbox.

Exemplu

```
<html>
<head>
<title>  butoane radio </title>
<script>
function test(x){
if(x[0].checked) alert ("Ati selectat culoarea "+ x[0].value);
if(x[1].checked) alert ("Ati selectat culoarea "+ x[1].value);
if(x[2].checked) alert ("Ati selectat culoarea "+ x[2].value);
}
```

```
</script>
</head><body>
```

Utilizarea listelor de selecţie

Propietatea utilizată pentru a testa dacă un element dintr-o lista de selecţie a fost selectat *este selected*. Valoarea ataşată atributului value este preluată cu .value, la fel ca la casetele de tip text. Exemplu:

```
<html>
<head>
<title> </title>
<script>
function lista(x)
{
for(i=0;i<6;i++)
if (x[i].selected) alert("ati selectat produsul "+ x[i].value)
}
</script>
</head>
<body>
<h1><center> liste selectie </center></h1>
<form>
<SELECT name="s" size="3">
<option value="test 11">test 11</option>
<option value="test 12">test 12</option>
<option value="test 13">test 13</option>
<option value="test 14">test 14</option>
<option value="test 15">test 15</option>
<option value="test 16">test 16</option>
</select>
<br><br><br>
<input type="button" name="b1" value="afiseaza" onclick="lista(s)">
</form></BODY></html
```

Evenimente

Limbajul JavaScript este bazat pe evenimente, acestea fiind folosite pentru a controla interacţiunea dintre utilizator şi aplicaţie. Programele convenţionale operează într-o manieră diferită, codul acestora fiind executat secvenţial. Pentru a specifica instrucţiunile JavaScript care trebuie executate la apariţia unui eveniment, limbajul HTML pune la dispoziţie anumite atribute pentru diferite marcaje, atribute care au ca valori instrucţiuni JavaScript (de obicei apeluri de funcţii). Cele mai multe evenimente sunt legate de acţiuni ale mouse-ului ca de exemplu: un clic (poate fi tratat cu atributul onclick), mouse deasupra cu atributul (onmouseover), mouse în afară cu atributul (onmouseout) etc. Pentru fiecare astfel de eveniment se pot defini handlere de evenimente care sunt functii javascript sau secvente de instructiuni care se vor executa atunci cand evenimentul respectiv are loc.

Evenimentele şi handlerele de evenimente care se pot folosi sunt urmatoarele:

Eveniment	Handler de ev.	are loc pentru	se aplică
Abort	OnAbort	oprirea incarcarii unei imagini	Image
Blur	OnBlur	pierderea focusului	Window şi elem. ale obiectului Form
Change	OnChange	schimbarea valorii unui element	Text, TextArea, Select
Click	OnClick	utilizatorul face click	Button, Checkbox, Link, Radio, Submit, Reset
DragDrop	OnDragDrop	plasarea unui obiect în fereastra browserului	Window
Error	OnErrorFocus	eroare la încarcarea obiectului	Image, Window
Focus	OnFocus	elementul capata focus	Window şi elem. ale obiectului Form
KeyDown	OnKeyDown	apasarea unei taste	Document, Image, Link, TextArea
KeyPress	OnKeyPress	apasarea sau mentinerea apasata a unei taste	Document, Image, Link, TextArea
KeyUp	OnKeyUp	eliberarea unei taste	Document, Image, Link, TextArea
Load	OnLoad	incarcarea paginii in navigator	Document
MouseDown	OnMouseDown	apasarea butonului de mouse	Document, Button, Link
MouseMove	OnMouseMove	mutarea cursorului	Nici unui obiect
MouseOut	OnMouseOut	mutarea cursorului de mouse in afara obiectului	Area, Link
MouseOver	OnMouseOver	mutarea cursorului peste un link	Link
MouseUp	OnMouseUp	eliberarea butonului de mouse	Document, Button, Link
Move	OnMove	deplasarea ferestrei	Window
Reset	OnReset	click pe butonul reset	Form
Resize	OnResize	redimensionarea ferestrei	Window
Select	onSelect	selectarea unui elem. al campului	Text, TextArea
Submit	onSubmit	apasarea pe submit	Form
Unload	onUnload	parasirea paginii	Document

În afara evenimentelor generate de utilizator pot fi tratate şi evenimente ce se produc automat. Apar evenimente ca:

- **load** – încărcarea paginii – poate fi gestionat cu atributul **onload**
- **unload** – închiderea documentului - poate fi gestionat cu atributul **onunload**

Ambele atribute aparţin marcajului BODY.

```
<script type=‘‘text/javascript’’>
a=‘‘Bun venit pe pagina mea’’; b=‘‘Multumesc!...Good Bye !’’
function f(x)
{
alert(x)
}
</script>
<body onload=‘‘f(a)’’ onunload=‘‘f(b)’’>
```