Cuprins

5. Scripturi client

5.1. Generalităţi

Scripturile sunt mici programe înglobate în fişierul HTML. Un script este de fapt un text mai special, o parte din textul din fişierul HTML. Când browserul încarcă fişierul HTML şi afişează pagina, în acelaşi timp interpretează scriptul (sau scripturile) din pagină, şi le execută.

Scripturile dintr-o pagină de Web pot modifica dinamic pagina (de exemplu, să schimbe mărimea fontului) la apariţia unor evenimente (de exemplu, un clic al mouse-ului), pot verifica dacă datele dintr-un formular sunt corecte etc.

Ca să puneţi un script în pagina HTML, puneţi textul care constituie scriptul între marcaje <script> şi </script>. Puteţi folosi marcajele adiţionale <noscript> şi </noscript> pentru browsere care nu ştiu să lucreze cu scripturi.

În acest capitol, vom trata scripturile client, adică cele care sunt executate de către navigatorul de web, browserul. Scripturile server (ASP, PHP etc.) vor fi tratate în alt capitol.

5.2. JavaScript

JavaScript este un limbaj de script, adică un limbaj de programare simplificat. Se numeşte JavaScript pentru că sintaxa acestui limbaj seamănă foarte mult cu cea a limbajului de programare Java. JavaScript este gratuit, şi browserele majore (IE, Netscape) ştiu să proceseze scripturile de acest tip dintr-o pagină care le conţine.

5.2.1. JavaScript în corpul fişierului HTML

Exemplu:

```
<html>
  <head>
    <title>
        Pagina cu JavaScript
    </title>
  </head>
  <body>
<script type="text/javascript">
document.write("Avem text pe pagina!")
</script>
  </body>
</html>
```

Observaţi că marcajul <script> are atributul „type”, care specifică tipul scriptului (există şi alte tipuri, în afară de JavaScript). Acest script are o singură comandă, care scrie „Avem text pe pagina!” pe pagina Web. Dacă scrieţi mai multe comenzi pe aceeaşi linie, trebuie să puneţi „;” (punct-virgulă) între comenzi.

Exemplu:

```
document.write("<i>"+"Avem text pe pagina!"+"<\/i>")
```

Dacă modificăm comanda ca mai sus, textul va fi afişat cu litere înclinate. Observaţi caracterul „\”; acesta este necesar pentru că în JavaScript, caracterul „/” are semnificaţie specială. Combinaţia „\/” îi spune browserului să trateze „/” normal, fără semnificaţia specială din JavaScript.

În cazul de mai sus, operatorul „+” concatenează două şiruri de caractere (sau mai multe).

Exemplu:

```
<!--
document.write("<i>"+"Avem text pe pagina!"+"<\/i>")
//-->
```

Punând scriptul în marcajul de comentariu, browserele care nu ştiu JavaScript nu vor afişa pe pagină textul scriptului (sursa scriptului). Browserele care ştiu JavaScript vor procesa scriptul, chiar dacă e în comentariu. Şirul „//”, comentariu în JavaScript, spune browserului să nu proceseze linia „-->”. Nu putem pune la început „//<!--”, pentru că un browser care nu ştie JavaScript ar afişa „//”.

5.2.2. JavaScript în partea de antet

Ca să fim siguri că scriptul se execută înainte de afişarea oricărui element pe pagină, îl putem pune în partea de antet a fişierului HTML.

5.2.3. JavaScript în fişier extern

Puteţi pune JavaScript într-un fişier extern, dacă doriţi să-l folosiţi pe mai multe pagini de Web. Astfel, va trebui să-l scrieţi doar odată, nu în fiecare fişier HTML. Fişierul cu script nu poate conţine marcajul <script> sau perechea lui.

Exemplu: putem crea următorul fişier, cu numele „*extern.js”*:

```
document.write("Text din scriptul extern.")
```

În fişierul html, mai adăugăm scriptul următor:

```
<script type="text/javascript" src="extern.js">
</script>
```

Cu atributul „src” al marcajului <script>, puteţi specifica un fişier cu scriptul care să fie executat.

5.2.4. Variabile

Ca în orice limbaj de programare, puteţi avea variabile în JavaScript. În exemplul următor, se defineşte variabila „textul”, şi se şi iniţializează cu valoarea „Textul interesant”. În numele variabilelor JavaScript, se face diferenţă între literele mici şi mari, adică „textul” este altă variabilă decât „Textul”. Similară este situaţia şi pentru funcţii şi obiecte.

Exemplu:

```
<html>
  <head>
    <title>
      Pagina cu JavaScript
    </title>
  </head>
  <body>
<script type="text/javascript">
<!--
var textul = "Textul interesant"
document.write("<i>"+textul+"<\/i>")
//-->
</script>
  </body>
</html>
```

5.2.5. Funcţii

Exemplu:

```
<html>
  <head>
    <title>
      Pagina cu JavaScript
    </title>
  </head>
  <body>
<script type="text/javascript">
<!--
var textul = "Textul interesant"
document.write("<i>"+textul+"<\/i>")
alert(textul)
//-->
</script>
  </body>
</html>
```

Aţi apelat funcţia „alert”, cu parametrul „textul”, ceea ce va determina afişarea unei căsuţe de alertă, cu conţinutul variabilei „textul”. Această funcţie, „alert”, este o funcţie standard a limbajului JavaScript.

Puteţi defini funcţiile dvs. proprii, pe care apoi să le apelaţi. Este indicat să definiţi funcţiile în secţiunea de antet („head”) a fişierului, ca să fie deja încărcate, când le apelaţi. Asta din cauză că browserul începe să proceseze şi să afişeze pagina înainte de a fi adus complet fişierul HTML de pe server.

Exemplu:

```
<html>
  <head>
    <title>
      Pagina cu JavaScript
    </title>
<script type="text/javascript">
<!--
function suma(a,b)
    {
     rezultatul=a+b
     return rezultatul
    }
//-->
</script>
  </head>
  <body>
<script type="text/javascript">
<!--
var textul = "Textul interesant"
document.write("<i>"+textul+"<\/i>")
alert(textul)
alert(suma("O zi ","buna"))
document.write("<br>Suma lui 1+2 este: "+suma(1,2))
//-->
```

```
</script>
  </body>
</html>
```

5.2.6. Execuţie condiţionată

Exemplu:

```
if (navigator.appName.indexOf("Internet Explorer")!=-1)
{
alert("Este Internet Explorer!")
}
else
{
alert("Nu este Internet Explorer!")
}
```

Această bucată de cod afişează o căsuţă de alertă, cu textul în funcţie de tipul browserului în care este afişată pagina. Condiţia pentru IF: se caută textul „Internet Explorer” în numele aplicaţiei browser (navigator). Dacă acest text nu apare în numele browserului, funcţia „indexOf” returnează „-1”. Condiţia pentru IF este adevărată doar dacă această funcţie NU returnează „-1” (operatorul „!=” înseamnă „diferit”). În acest caz, se execută comenzile din blocul IF. Când condiţia este falsă, se execută comenzile din blocul „else”.

Aveţi şi alte comenzi pentru condiţionare în JavaScript, cum ar fi „switch” sau operatorul condiţional „?”.

5.2.7. Bucle

Exemplu:

```
<html>
  <head>
    <title>
      Pagina cu bucle
    </title>
<script type="text/javascript">
</script>
  </head>
  <body>
<script type="text/javascript">
<!--
for (i=0; i<20; i++)
{
    document.write("valoarea lui i: "+i+"<br>")
}
//-->
</script>
  </body>
</html>
```

Exemplul de mai sus scrie în document valorile lui „i”, repetând acest lucru până când „i” ajunge la 19.

Puteţi folosi şi bucle „while” sau „do...while” în JavaScript.

5.2.8. Evenimente

Evenimentele sunt o parte importanţă a JavaScript. De fapt, aproape de fiecare dată când se utilizează JavaScript într-o pagină de web, el procesează şi evenimente.

Exemplu:

```
<html>
  <head>
    <title>
      Legatura prin buton
    </title>
  </head>
  <body>
    <form>
      <input type="button" value="Apasa-ma!"
       onclick='window.open("http://www.protv.ro",
"fereastra_noua")'>
    </form>
  </body>
</html>
```

Exemplul de mai sus prezintă o bucată de cod JavaScript inline, adică fără marcajele <script> şi </script>, direct într-un marcaj html. La evenimentul „onclick”, adică în momentul când se dă clic pe buton, se apelează funcţia „open” (din grupul de funcţii „window”), cu parametrii aferenţi. Această funcţie face ca la apăsarea pe buton, să se încarce pagina ProTV în fereastra cu numele intern „fereastra_nouă”.

Puteţi găsi multe exemple de JavaScript la http://javascript.internet.com.

5.3. VBScript

Aşa cum JavaScript seamănă cu limbajul Java, şi VBScript seamănă cu limbajul Visual Basic, dezvoltat de Microsoft. Din acest motiv, VBScript funcţionează numai cu browserul Internet Explorer, care este tot un produs Microsoft.

VBScript seamănă oarecum cu JavaScript.

Exemplu:

```
<html>
  <head>
    <title>
      Pagina cu bucle
    </title>
  </head>
  <body>
<script type="text/vbscript">
<!--
document.write("valoarea lui i: ")
//-->
</script>
  </body>
</html>
```

Scripturile VBScript se pot pune în antet, în corp sau în fişier extern, ca şi cele JavaScript. În VBScript, avem subrutine şi funcţii. Subrutinele nu returnează nimic, funcţiile, da. Numele unor funcţii diferă faţă de perechea lor din JavaScript, altele sunt similare; apar şi funcţii noi.

Exemplu:

```
<html>
  <head>
    <title>
      Pagina cu bucle
    </title>
  </head>
  <body>
<script type="text/vbscript">
<!--
msgbox ("Casuta VBScript!")
//-->
</script>
  </body>
</html>
```

5.4. DHTML

DHTML vine de la Dynamic HTML, şi se referă la pagini dinamice, adică pagini de Web a căror conţinut se schimbă dinamic.

Nu există un limbaj DHTML; el este un concept, un mix între HTML, JavaScript şi stiluri CSS. El se bazează pe DOM („Document Object Model”), o teorie care descrie structura obiectuală a paginii de Web şi evenimentele care pot apare.

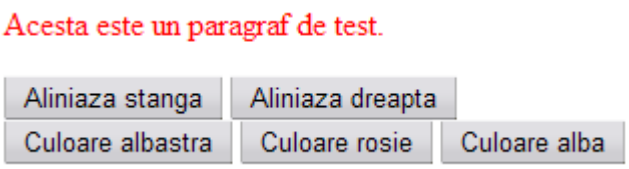
Exemplu:

```
<html>
  <head>
    <title>
      Pagina dinamica
    </title>
  </head>
  <body>
    <!-- Paragraf cu identificatorul "paragraf_de_test" -->
    <p id="paragraf_de_test">
      Acesta este un paragraf de test.
    </p>
    <!-- Un script explicit -->
<script language="JavaScript" type="text/javascript">
paragraf_de_test.style.color="red"
</script>
    <form>
      <!-- JavaScript inline, direct in marcaj -->
       <!-- Alinierea -->
       <input type="button" value="Aliniaza stanga"
       onclick="paragraf_de_test.style.textAlign='left'">
       <input type="button" value="Aliniaza dreapta"
       onclick="paragraf_de_test.style.textAlign='right'"><br>

       <!-- Culoarea -->
       <input type="button" value="Culoare albastra"
       onclick="paragraf_de_test.style.color='blue'">
       <input type="button" value="Culoare rosie"
       onclick="paragraf_de_test.style.color='red'">
```

```
        <input type="button" value="Culoare alba"
        onclick="paragraf_de_test.style.color='white'">
    </form>
  </body>
</html>
```

Rezultat:



Exemplul de mai sus conţine un singur paragraf, cu identificatorul „paragraf_de_test”, şi un formular. Formularul conţine două butoane pentru alinierea paragrafului şi trei pentru schimbarea culorii textului din paragraf.

Setarea alinierii se face prin intermediul identificatorului paragrafului şi prin atributul de stil „*style.textAlign*”.

Schimbarea culorii se face tot cu identificatorul paragrafului şi cu atributul de stil „*style.color*”.

5.5. Exerciţii scripturi client

1. Încercaţi exemplele din acest capitol.
2. Creaţi un script care să genereze o matrice de 20x20, cu elementele numere întregi pozitive, de la 1 la 400, sub formă de tabel cu chenarul de grosime 1.

6. JavaScript avansat

6.1. JavaScript în pagina de Internet

Părţile de cod javascript sunt incluse în pagina HTML prin marcajul SCRIPT. Numărul de astfel de elemente de cod nu este limitat. Atributul LANGUAGE este optional dar recomandat: ajută la excluderea unor versiuni nedorite de script.

Atributul SRC este folosit pentru a include un fişier extern cantinând cod javascript.

Exemplu:

```
<script language="JavaScript"
src="corefunctions.js">
</script>
```

Atributele posibile ale marcajului script sunt următoarele:

Atributul	Valoare	Efect	DTD
type	text/ecmascript text/javascript application/ecmascript application/javascript text/vbscript	Specifică tipul MIME al porţiunii de cod care va urma.	STF
charset	charset	Specifică tipul de caractere utilizat în script.	STF
defer	defer	Indică faptul că script+ul nu va	STF

		genera conţinut, browser-ul putând continua generarea paginii de internet.	
language	javascript livescript vbscript xml etc.	Specifică tipul de limbaj folosit. Scos din uz.	TF
src	URL	Defineşte adresa la care este stocat fisierul script care va fi inclus.	STF

6.2. Variabile JavaScript

Variabilele javascript sunt referite, ca în orice limbaj de programare, prin nume şi pot stoca o valoare. Javascript sesizează diferenţa dintre litere mari şi mici. Variabilele sunt globale sau locale. O variabilă globală poate fi accesată de orice cod javascript din pagină.

O variabilă locală este recunoscută doar de funcţia în interiorul căreia a fost declarată.

Exemplu: newVariable=5;

Tipurile de date javascript sunt: întreg(în decimal sau hexazecimal), real sau cu virgulă mobilă, boolean (cu valori doar de true sau false), şirur de caractere scris între ghilimele "", obiect. Tipul variabilei este stabilit în momentul în care variabila primeşte o valoare. Imediat după ce o variabilă a fost creată ea are un tip nedefinit.

6.3. Operatori

Tabel tipuri de operatori javascript

Operatori aritmetici:	Numele operaţiei	Operatori logici	Numele operaţiei
+	adunare	==	egalitate
-	scădere	!=	diferit
*	înmulţire	>	mai mare
/	împărţire	>=	mai mare sau egal
%	modul	<	mai mic
		<=	mai mic sau egal
++	incrementare cu 1		
--	decrementare cu 1		
-	negaţie		
Operatori boolean			
&&	şi		
||	sau		
!	negaţie		

6.4. Funcţii

Sintaxa unei funcţii este:

```
function functie(var1, var2,etc..)
  {
        ...instrucţiuni...
  return valoare;
  }
```

Apelul unei funcţii se realizează astfel:

new a=funcţie(parametrii);

6.5. Instrucţiuni condiţionale şi de ciclare

Instrucţiunile condiţionale de bază au acelaşi comportament ca în cazul multor limbaje, de exemplu C.

Tabel instrucţiuni condiţionale şi de ciclare

Instrucţiunea condiţională **if… else**	if (condiţie)
	{ instrucţiuni1; }
	else
	{ instrucşiuni2; }

Instrucţiunea condiţională **case**
Dacă este îndeplinită condiţia se execută intrucţiunea care corespunzatoare etichetei. Break părăseşte corpul instrucţiunii condiţionale.
	switch (expresie){
	case etichetă :
	instrucţiuni;
	break;
	case etichetă :
	instrucţiuni;
	break;
	...
	default : statements;
	}

Instrucţiunea de repetare for Se execută instrucţiune1 necondiţionat. Se verifică valaorea de adevăr a testullui. În caz de adevăr, se execută instrucţiune. La final se incrementează.	for (intrucţiune1; test; increment) { instrucţiune; }
Instrucţiune de ciclare cu test final Setul de intrucţiuni se execută cel puţin o dată. Cât timp candiţia are valaore de adevăr adevărat atunci este execută setul de intrucţiuni.	do { intrucţiune1;} while (condiţie)
Instrucţiune de repetare cu test iniţial Dacă valoarea de adevăr a condiţiei este adevărată atunci se execută setul de instrucţiuni.	while (condiţie) { intrucţiune1; }
Instrucţiune for… in Pentru fiecare variabilă a obiectului se execută setul respective de instrucţiuni	for (variabilă in obiect) { intrucţiune1; }
Instrucţiunea with Pentru fiecare element corespunzător obiectului object, se execută instrucţiunea.	with (object) { instrucţiunea; }

6.6. Funcţii pentru atenţionare

alert("text")- se creează o fereastră de dimensiuni reduse în care va fi afişat textul introdus ca parametru

Exemplu:

```
<html>
<head>
<title>Alert</title>
</head>
<body onLoad="window.alert('Si pagina se incarca...')">
</body>
</html>
```

confirm("text")- se creează o fereastră în care se solicită confirmarea utilizatorului

prompt("text")-se solicită utilizatorului să introducă o anumită valoare

Exemplu:

```
<html>
<head>
<title>Prompt si confirm</title>
</head>
<body onLoad="window.confirm('Ciao '+ y+', s-a incarcat
pagina?')">
<script type="text/javascript">
var y=window.prompt("Numele, te rog",'no name')
</script>
</body>
</html>
```

6.7. Obiecte

Un obiect este o colecţie de date, proprietăţi(variabile) şi metode(funcţii). În javascript obiectele sunt de două tipuri: create de programator sau predefinite. Aceste obiecte predefinite sunt redate în figura următoare:

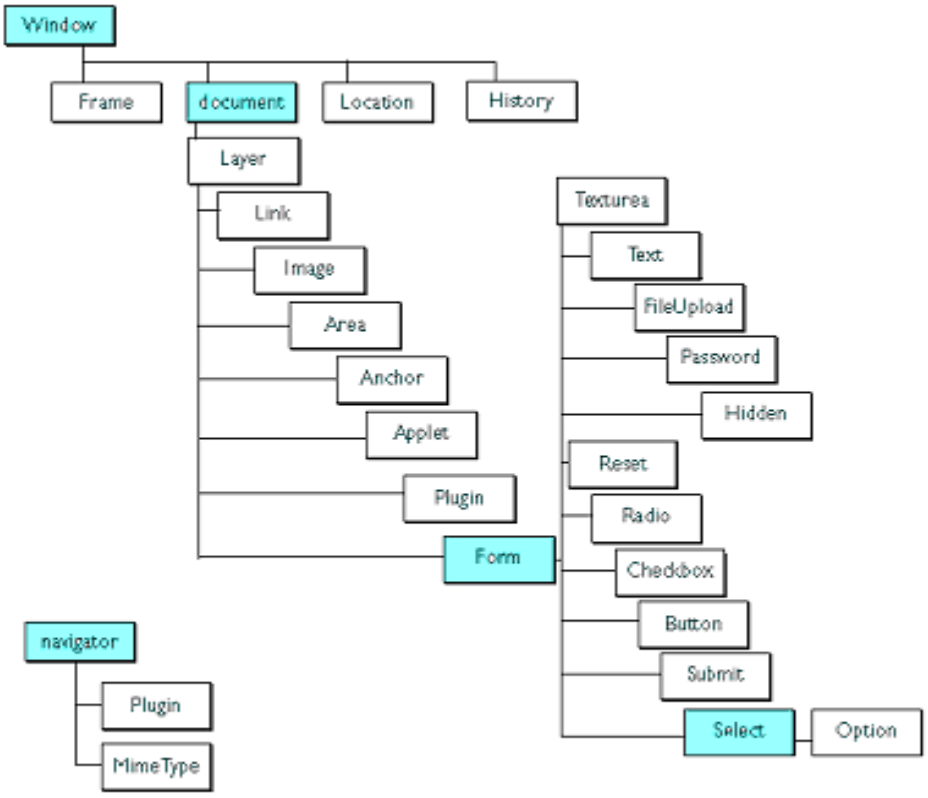


Figura 6.7.1. Modelul obiectelor de tip window, document, form, navigator.

Majoritatea acestor obiecte sunt corelate cu cele mai importante caracteristici ale unei pagini web.

Fiecăruia dintre aceste elemente îi sunt asociate funcţii specifice.

Un exemplu elocvent este window.close() care determină închiderea ferestrei curente a navigatorului de internet.

6.7.1. Obiectul string

Nume metodă sau atribut	Efect
+	Concatenarea mai multor siruri
indexOf(caracterul de căutat)	Returnează indexul unui anumit element din şirul de caractere
length	
charAt(indexul care ne interesează)	Returnează caracterul aflat la indexul furnizat ca şi parametru.
lastIndexOf(caracterul dorit)	Returnează indexul ultimei apariţii a caracterului furnizat ca şi parametru
split(characterul de delimitare)	Împarte şirul în mai multe elemente în funcţie de un anumit caracter introdus ca şi parametru.
substring(primul index, al doilea index)	Permite selectarea unei anumite porţiuni din şirul iniţial intre doi indecşi specificaţi.
substr(primul index, lungime subsir)	Acelaşi efect, cu deosebirea ca se furnizează ca parametri primul index şi al doilea, lungimea dorită a subşirului
toLowerCase()	Elementele sirului vor fi prezentate cu

	litera mica.
toUpperCase()	Elementele sirului vor fi prezentate cu litera mare.
replace(sirul care va fi inlocuit , sirul care inlocuieste)	Inlocuieste sirul furnizat ca prim parametru cu al doilea sir dat ca si parametru.

6.7.2. Obiectul Date

Nume metode sau atribute	Efect
Date()	Constructor- creează un obiect de tip data care redă data şi timpul current la momentul apelării sale
Date(un şir de caractere care reprezintă o dată)	Utilizează formatul introdus ca parametru pentru a crea un obiect de tip dată; Şirul are următorul format: "luna zi,an ora:minute:secunde"
Data(an,luna,ziua,ora,minutul,secunda)	Creează un obiect de tip dată pe bază parametrilor introduşi la apelul funcţiei.
getYear()	Returnează un întreg care reprezintă anul
getMonth()	Returnează un întreg reprezentând luna din an; valoarea va aparţine intervalului: $[1,12]\cap N$.
getDate()	Returnează un întreg reprezentând ziua din lună; valoarea va aparţine intervalului: $[1,31]\cap N$.
getDay()	Returnează un întreg reprezentând ziua din săptămână; valoarea va aparţine intervalului: $[1,6]\cap N$.
getHour(), getMinutes(), getSeconds()	Similar ca mai sus.
getTime()	Redă numărul se milisecunde care au trecut începând cu data de 1 ianuarie 1970
parse(o data de referinţă)	Calculează numărul de milisecunde care s-au scurs începând cu 1 ianuarie 1970 şi până la data furnizată ca argument.
setDate(un întreg ce reprezintă noua valoare pentru ziua curentă) **setHours**(un întreg ce reprezintă noua valoare pentru ora curentă), s**etMinutes**(un intreg ce reprezintă noua valoare pentu minutul curent), **setMonth**(un întreg ce reprezintă noua valoare pentru luna curentă),	Permit setarea valorilor pentru zi, lună, an, ora, minut, secunda

setYear(un întreg ce reprezintă noua valoare pentru anul current), **setSeconds**(un întreg ce reprezintă noua valoare pentru secunda curentă) **setTime**(numărul de milisecunde care va produce modificarea tipului calculat in milisecunde începând cu 1 ianuarie 1970)	

6.7.3. Obiectul Array

Nume atribut sau metodă	Efectul
new Array(a1,a2,a3,..) sau **new Array**(lungime)	Construieşte un tablou de elemente prin specificarea fiecărui element în parte sau a numărului total de elemente pe care îl va avea tabloul.
concat(mai multe tablouri)	Ca rezultat, tablourile vor fi concatenate într-un singur şir.
join(caracter de separare)	Returnează ca rezultat un singur şir de caractere în care elementele componenete vor fi separate de caracterul furnizat ca parametru
pop()	Îndepărtează ultimul element din tablou; rezultatul returnat este ultimul element din tablou.
shift()	Îndepărtează primul element din şir; returnează primul element din şir
push(a1,a2,…)	Adaugă unul sau mai multe elemente; returnează noua lungime a tabloului.
slice(index-ul de inceput, indexul final)	Creează un nou tablou dintr-un tablou existent, de la un index specificat până la final sau pâna la indexul final, dacă a fost furnizat ca parametru
splice(indexul, câte elemente[, a1,a2…])	Metoda permite îndepărtarea sau adaugarea unor elemente începând cu un anumit index, un anumit număr de elemente.

6.7.4. Obiectul Math

Nume atribut sau metodă	Efectul
E	Constanta lui Euler
PI	Valoarea lui PI=3.1416
SQRT1_2	$1/\sqrt{2}=0,7071$
SQRT2	$\sqrt{2}=1,41$
random(număr)	Generează un număr aleator
floor(număr)	
round(număr)	
min(număr1, numar2,...) **max(număr1, număr2,...)**	Identifică numărul cel mai mic dintre numerele furcnizate ca arguent
sqrt(număr)	Extrage rădăcina pătrată.
pow(x,y) **log(număr)**	Ridică pe x la puterea y. Returnează logaritmul natural (în baza e) a

	numărului.
sin(număr) **cos(număr)** **asin(număr)** **acos (număr)**	Returnează sinusul, cosinusul, arcsinusul sau arc-cosinusul numărului.

6.8. Evenimente

6.8.1. Generalităţi

Evenimentele sunt acţiuni care pot fi detectate de javascript. De cele mai multe ori, evenimentele sunt folosite în combinaţie cu funcţii: funcţia este apelată la declaşarea unui eveniment.

Principalele evenimente sunt:

- onfocus="" ; primeşte focalizare, poate fi folosit în combinaţie cu select, text, textarea,
- onblur=""; pierde focalizarea, poate fi folosit în combinaţie cu select, text, textarea,
- onchange=""; conţinutul unui câmp se modifică(select, text, textarea),
- onselect=""; textul este selectat (text, textarea),
- onmouseover=""; mouse-ul se mută pe o legătură <a>,
- onmouseout=""; mouse-ul iese de pe suprfaţa unei legături <a>,
- onclick="" ; click cu mouse-ul pe un obiect(a, buton,checkbox, radio, reset, submit),
- onload=""; când utilizatorul a accesat pagina şi aceasta s-a afişat în întregime (body, frameset),
- onunload=""; când utilizatorul părăseşte pagina sau browser-ul deschide un nou document (body, frameset),
- onSubmit=""; butonul de submit a fost apăsat(form).

Evenimentele sunt captate cu două scopuri: să execute o funcţie sau să afişeze un

element popup .

OnFocus şi onBlur sunt folosite pentru validarea câmpurilor unui formular.

OnLoad şi onUnload sunt utilizate la afişarea elementelor popup când utilizatorul accesează sau părăseşte o pagină.

OnSubmit are ca rol important: să valideze datele unui formular înainte de a fi expediate.

OnMouseOver şi onMouseOut sunt folosite pentru crearea butoanelor animate, pentru deschiderea unor ferestre de alertă.

6.8.2. Gestiunea evenimentelor

Codul Javascript este capabil să gestioneze evenimente adesea prin apelul unor funcţii javascript. Evenimentele sunt acţiuni care apar în paginile de web ca rezultat la o acţiune a utilizatorului. De cele mai multe ori, codul javascript sau apelul funcţiei javascript de executat în cazul producerii unor evenimente este plasat în interiorul corpului unei pagini HTML, chiar în componenţa unui marcaj.

Exemplu:<tag attribute1 attribute2 onEventName="javascript code;">

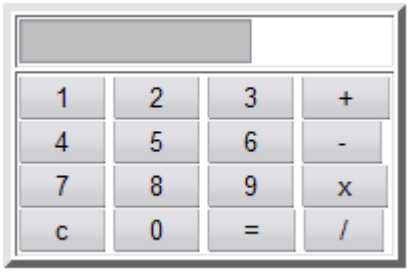
6.9. Puterea JavaScript

Puterea limbajului JavaScript şi a scripturilor client în general nu constă în complexitatea limbajului, ci în faptul că instrucţiunile au acces la componentele paginii de web, astfel că pot obţine informaţii despre ele şi pot să le modifice. Un exemplu interesant este mai jos, preluat de pe pagina http://javascript.internet.com:

```
<html>
<head>
<!-- Luat de pe javascript.internet.com -->
<title>Calculator</title>
</head>
<body>
<form name="Calc" id="Calc">
  <table border="4">
    <tr>
      <td><input type="text" name="Input" size="16">
        <br>
      </td>
    </tr>
    <tr>
      <td>
        <input type="button" name="one"   value="  1  "
onClick="Calc.Input.value += '1'">
        <input type="button" name="two"   value="  2  "
onClick="Calc.Input.value += '2'">
        <input type="button" name="three" value="  3  "
onClick="Calc.Input.value += '3'">
        <input type="button" name="plus"  value="  +  "
onClick="Calc.Input.value += ' + '">
        <br>
        <input type="button" name="four"  value="  4  "
onClick="Calc.Input.value += '4'">
        <input type="button" name="five"  value="  5  "
onClick="Calc.Input.value += '5'">
        <input type="button" name="six"   value="  6  "
onClick="Calc.Input.value += '6'">
        <input type="button" name="minus" value="  -  "
onClick="Calc.Input.value += ' - '">
        <br>
        <input type="button" name="seven" value="  7  "
onClick="Calc.Input.value += '7'">
        <input type="button" name="eight" value="  8  "
onClick="Calc.Input.value += '8'">
        <input type="button" name="nine"  value="  9  "
onClick="Calc.Input.value += '9'">
        <input type="button" name="times" value="  x  "
onClick="Calc.Input.value += ' * '">
        <br>
        <input type="button" name="clear" value="  c  "
onClick="Calc.Input.value = ''">
        <input type="button" name="zero"  value="  0  "
onClick="Calc.Input.value += '0'">
```

```
        <input type="button" name="DoIt"  value="  =  "
onClick="Calc.Input.value = eval(Calc.Input.value)">
        <input type="button" name="div"   value="  /  "
onClick="Calc.Input.value += ' / '">
        <br>
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

Rezultatul:



Scriptul, imbricat printre elementele de formular şi tabel html, compune formula de calculat direct în căsuţa de text; când se apasă butonul „=”, textul (formula) din căsuţă este procesat de instrucţiunea JavaScript „eval(Calc.Input.value)”, care transformă textul cu cifre si semne algebrice într-o formulă şi o calculează, rezultatul formulei ajungând înapoi în aceeaşi căsuţă.