

## Capitolul 5.

# JavaScript

### Obiective:

- să înțeleagă noțiunea de JavaScript
- să înțeleagă avantajele și dezavantajele utilizării scripturilor JavaScript.

## 5.1. Introducere

Acest capitol este aproape imposibil de asimilat dacă nu ați parcurs capitolele anterioare. Este important să cunoașteți deja noțiunile de Internet, WWW și să știți să creați pagini Web simple în HTML. În afară de acestea, ar fi bine să fiți familiarizați cu un limbaj de programare cum ar fi: C, C++, C#, Visual Basic sau Java, dar nu este o condiție fără de care să nu puteți trece mai departe.

Corporația Netscape Communications a creat limbajul de scriptare "LiveScript" pentru Web designers și dezvoltatori. În decembrie 1995, LiveScript a fost redenumit "JavaScript" și lansat ca parte componentă a browser-ului Netscape Navigator 2.0 de către Netscape Corporation și Sun Microsystems.

JavaScript a fost proiectat ca fiind un limbaj de scriptare simplu, pentru programatori începători care nu cunosc Java și pentru Web designers. Acest limbaj de scriptare este simplu, deoarece la scrierea unui script nu suntem obligați să declarăm variabilele înainte de a le utiliza și nici să utilizăm un compilator. Browser-ul este cel care interpretează scriptul și care ne arată dacă avem greșeli. Nu este nevoie să utilizăm un compilator Java sau C, nu este necesar să instalăm diverse soft-uri pe computerul nostru pentru a putea vizualiza rezultatul. Nu avem nevoie decât de un browser. Browser-ul care a interpretat pentru prima dată scripturile JavaScript este Netscape, după care au urmat Internet Explorer, Opera, Mozilla și altele.

JavaScript este interpretat de către browser-e și poate funcționa indiferent de platformă. Sistemele de operare pe care poate fi interpretat un script JavaScript sunt: Windows, UNIX, Macintosh.

Cu JavaScript putem crea pagini Web dinamice, interactive, pop-up, bare de navigare, putem trimite date pentru verificare prin Internet, putem controla applet-uri Java. JavaScript este un limbaj care poate fi utilizat atât pe parte de client cât și pe parte de server.

JavaScript este un limbaj de scriptare bazat pe obiecte, nu orientat pe obiecte cum este limbajul de programare Java. JavaScript nu este Java, chiar dacă asemănarea numelor poate duce la această confuzie. Cu JavaScript nu putem crea applet, dar îl putem interpreta, după ce îl creăm cu Java.

JavaScript este un limbaj interpretat de către browser și nu are nevoie să fie compilat înainte de a fi rulat. Browser-ul analizează fiecare element al paginii Web pe rând, împărțindu-l în componente mai mici.

În funcție de locul unde este plasat un script, scriptul poate fi pe parte de client sau pe parte de server. Dacă scriptul rulează în browser-ul clientului, atunci îl numim script pe parte de client, așa cum sunt majoritatea script-urilor.

Ca răspuns la provocarea lansată de Netscape, au apărut și alte limbi de scriptare asemănătoare:

- Microsoft a lansat propriul limbaj de scriptare cu numele VBScript, iar în iulie 1996 a lansat Internet Explorer 3.0 cu JScript inclus;
- ECMA a lansat în iunie 1997 o versiune numită ECMAScript, care este de fapt standard internațional pentru JavaScript.

Rezultatul la care trebuie să ajungă orice firmă ce dorește să furnizeze un limbaj de scriptare este că acel limbaj trebuie să ruleze independent de platformă pentru a "trăi" în spațiul

Web. Acest lucru duce la concluzia că, atât utilizatorii cât și dezvoltatorii de soft trebuie să lucreze împreună pentru a găsi soluții utile tuturor.

Bineînțeles că JavaScript are și dezavantaje, cum ar fi faptul că se pot crea bucle infinite cu ajutorul instrucțiunilor repetitive sau atacuri asupra computerelor care permit rularea diverselor scripturi, dar un utilizator Internet cu puțină experiență nu ar trebui să-și facă probleme. Orice limbaj de programare poate fi folosit atât pentru a crea lucruri bune, cât și pentru a crea lucruri rele. JavaScript este doar un limbaj de scriptare, adică un instrument! ☺

## 5.2. Unde scriem script-urile JavaScript

Ca și un stil CSS, un script JavaScript poate fi scris atât în pagina Web, în antet sau în corpul documentului, cât și în exteriorul ei, ca fișier separat. Fișierele care conțin scripturi JavaScript se salvează cu extensia .js.

Pentru scripturile simple, cel mai ușor este să le introducem în interiorul paginii Web, în antet și mai rar în corpul paginii. Scripturile complexe, care este posibil să genereze erori la interpretare sau care se încarcă mai greu este indicat să le salvăm într-un fișier separat și să le apelăm din interiorul paginii Web. Mai este însă un motiv pentru care scripturile ar trebui scrise separat și anume posibilitatea ca acestea să fie modificate individual, fără să modificăm toată pagina Web sau tot site-ul care utilizează acest script.

Ca și la HTML, scriptul JavaScript se introduce între două etichete **<script>** și **</script>** la care ar trebui să adăugăm **language="JavaScript"** pentru ca browser-ul să știe ce fel de script urmează.

**Exemplu** Cum scriem un script JavaScript în *head*.

```
<html>
<head><title>JavaScript in head</title>
<script language="JavaScript">
document.write("Acum avem scriptul in head")
</script>
</head>
<body>
</body>
</html>
```



**Exemplu** Cum scriem un script JavaScript în *body*.

```
<html>
<head><title>Primul exemplu JavaScript </title>
</head>
<body>
<script language="JavaScript">
document.write("Primul meu exemplu JavaScript")
</script>
</body>
</html>
```



**Important!** JavaScript este case-sensitive, adică este important să utilizăm numele variabilelor așa cum le-am declarat, cu litere mari sau mici. Dacă am declarat o variabilă cu numele X1, nu o putem folosi ca x1, ci doar cu numele X1.

## 5.3. Operatori JavaScript

### 5.3.1. Operatori aritmetici

JavaScript conține operatorii aritmetici standard, întâlniți și în limbajele de programare C și C++. Din punctul de vedere al operanților la care se aplică, operatorii sunt de două feluri: unari și binari. Operatorii unari se aplică unui singur operand, iar operatorii binari se aplică între doi operanți.

Operator	Tip de operator	Descriere	Exemplu
+	unar	adunarea a două valori	+x sau a+b
-	unar	diferența a două valori	-x sau a-b
*	binar	înmulțirea a două valori	a*b
/	binar	împărțirea a două valori	a/b
%	binar	restul a două valori	a%b

#### Exemplu de utilizare a operatorilor aritmetici.

```
<html>
<head><title>Operatori aritmetici</title>
</head>
<body><pre>
<script language="JavaScript">
var x=5, y=2;
document.writeln("x=5 y=2");
document.writeln("x+y=", x+y);
document.writeln("x-y=", x-y);
document.writeln("x impartit la y = ", x/y);
document.writeln("Restul impartirii lui x la y =
= ", x%y);
</script>
</pre>
</body></html>
```

The screenshot shows the browser window titled "Operatori aritmetici - Microsoft Internet Explorer". The address bar shows "C:\Operatori aritmetici.html". The page content displays the following output:  
x=5 y=2  
x+y=7  
x-y=3  
x impartit la y = 2.5  
Restul impartirii lui x la y = 1

**OBS** În exemplul de mai sus am declarat variabilele x și y înainte de a le utiliza. Declararea variabilelor însă nu este obligatorie, acest limbaj ne permite să le utilizăm fără să le declarăm anterior.

### 5.3.2. Operatori de incrementare / decrementare

Operatorii se mai clasifică și în funcție de operația pe care o efectuează și anume în:

- **operatori de incrementare** – crește valoarea variabilei cu o unitate
- **operatori de decrementare** – scade valoarea variabilei cu o unitate;  
care la rândul lor sunt de două feluri:
- **prefixată** – operatorul se scrie înaintea operandului. Operația de incrementare (sau decrementare) se efectuează înainte de a se efectua operațiile expresiei în care se află.  
Exemplu dacă inițial a=1 și x=++a; vom avea la final x=2 și a=2, adică se incrementează variabila a cu o unitate înainte de a i se atribui variabilei x valoarea variabilei a.
- **postfixată** – operatorul se scrie după operand. Operația de incrementare (sau decrementare) se efectuează după ce se efectuează operațiile expresiei în care se află.  
Exemplu dacă inițial a=1 și x=a++; vom avea la final x=1 și a=2, adică se incrementează variabila a cu o unitate după ce i se va atribui variabilei x valoarea variabilei a.

Operator	Tip de operator	Descriere	Exemplu
++	incrementare	incrementare prefixată	++x sau x=x+1
		incrementare postfixată	x++ sau x=x+1
--	decrementare	decrementare prefixată	--x sau x=x-1
		decrementare postfixată	x-- sau x=x-1

### **Exemplu de utilizare a operatorilor de incrementare/ decrementare.**

```
<html>
<head><title>Operatori de incrementare/
decrementare </title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=1, y=2;
document.writeln("x=1 y=2");
document.writeln("z=++x-y--=", ++x-y--);
</script>
</pre>
</body></html>
```



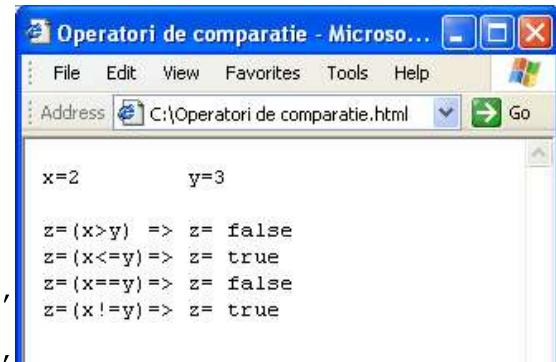
### **5.3.3. Operatori de comparație**

Pentru a compara doi operanzi, utilizăm operatorii binari de comparație, descriși în tabelul de mai jos.

Operator	Descriere	Exemplu pentru $x=2$ și $y=3$	Rezultat
$==$	operator de egalitate	$x==y$	false
$!=$	operator de inegalitate	$x!=y$	true
$>$	mai mare decât...	$x>y$	false
$<$	mai mic decât...	$x<y$	true
$\geq$	mai mare sau egal	$x\geq y$	false
$\leq$	mai mic sau egal	$x\leq y$	true

### **Exemplu de utilizare a operatorilor de comparație.**

```
<html>
<head>
<title>Operatori de comparatie</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=2,
y=3;
document.writeln("x=2      y=3\n");
document.writeln("z=(x>y) => z= ", z=(x>y));
document.writeln("z=(x<=y) => z= ", z=(x<=y));
document.writeln("z=(x==y) => z= ", z=(x==y));
document.writeln("z=(x!=y) => z= ", z=(x!=y));
</script>
</pre>
</body>
</html>
```



**OBS** Codurile \n, \t se utilizează pentru a trece la o linie nouă, respectiv pentru a lăsa un spațiu, similar cu cel lăsat de tasta Tab. Aceste coduri se utilizează în cadrul unui script încadrat de etichetele <pre> și </pre>.

#### 5.3.4. Operatori logici (booleeni)

Acești operatori se utilizează în cadrul unor expresii logice sau se aplică unor variabile, constante. Rezultatul unei expresii logice este tot o valoare logică. Valorile logice sunt true și false. Zero (0) are valoarea logică false, iar orice număr diferit de zero are valoarea logică true.

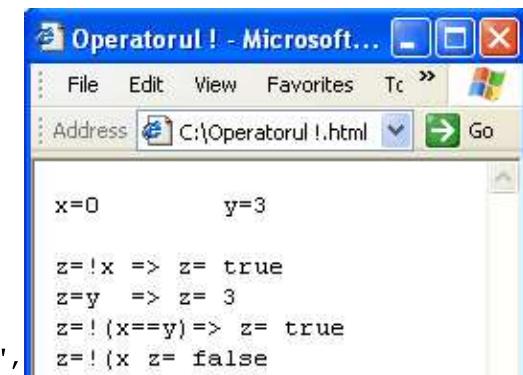
Operator	Descriere	Exemplu pentru x=0 și y=3	Rezultat
!	- operatorul logic NOT (negație logică) - operator unar - returnează o valoare logică diferită	!x !y !(x==y) !(x!=y) !(x>y) !(-25)	true false true false true false
&&	- operatorul logic AND (și logic) - operator binar - returnează <b>true</b> dacă ambii operanzi (sau expresii) au valoarea <b>true</b> , altfel returnează <b>false</b> .	x&&y (!x)&&y 0&&1 (4<=2)&&(5>=4) (x==0)&&(y==3)	0 3 0 false true
	- operatorul logic OR (sau logic) - operator binar - returnează <b>false</b> dacă ambii operanzi (sau expresii) au valoarea <b>false</b> , altfel returnează <b>true</b> .	x  y (!x)  y 0  1 (4<=2)  ((5>=4) (x==0)  (y==3)	3 true 0 true true

#### Exemplu de utilizare a operatorului !

```

<html>
<head>
<title>Operatorul !</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=0, y=3;
document.writeln("x=0      y=3\n");
document.writeln("z=!x => z= ", z=!x);
document.writeln("z=y => z= ", z=y);
document.writeln("z=!(x==y)=>      z=      ", z=!(x==y));
document.writeln("z=!(x<y)=> z= ", z=!(x<y));
</script>
</pre>
</body>
</html>

```



&&	true	false
true	true	false
false	false	false

	true	false
true	true	true
false	true	false

### Exemplu de utilizare a operatorului &&.

```
<html>
<head><title>Operatorul logic &&</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=0, y=3;
document.writeln("x=0           y=3\n");
document.writeln("z=x&&y      => z= ",  

                 z=x&&y);
document.writeln("z=(!x)&&y     => z= ",  

                 z=(!x)&&y);
document.writeln("z=0&&1      => z= ",  

                 z=0&&1);
document.writeln("z=(4<=2)&&(5>=4)=>z=", z=(  

                 4<=2)&&(5>=4));
document.writeln("z=(x==0)&&(y==3)=>z=", z=(  

                 x==0) &&(y==3));
</script>
</pre>
</body>
</html>
```

The screenshot shows the browser window title "Operatorul logic && - Microsoft Internet Explorer". The address bar contains "C:\Operatorul logic &&.html". The page content displays the following output:

x=0	y=3
z=x&&y	=> z= 0
z=(!x)&&y	=> z= 3
z=0&&1	=> z= 0
z=(4<=2)&&(5>=4)	=> z=false
z=(x==0)&&(y==3)	=> z=true

### Exemplu de utilizare a operatorului ||.

```
<html>
<head><title>Operatorul logic ||</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=0, y=3;
document.writeln("x=0           y=3\n");
document.writeln("z=x||y      => z= ",  

                 z=x||y);
document.writeln("z=(!x)||y     => z= ",  

                 z=(!x)||y);
document.writeln("z=0||1      => z= ",  

                 z=0&&1);
document.writeln("z=(4<=2)|| (5>=4)=> z= ",  

                 z=(4<=2)|| (5>=4));
document.writeln("z=(x==0)|| (y==3)=> z= ",  

                 z=(x==0)|| (y==3));
</script>
</pre>
</body>
</html>
```

The screenshot shows the browser window title "Operatorul logic || - Microsoft Internet Explorer". The address bar contains "C:\Operator logic.html". The page content displays the following output:

x=0	y=3
z=x  y	=> z= 3
z=(!x)  y	=> z= true
z=0  1	=> z= 0
z=(4<=2)   (5>=4)	=> z= true
z=(x==0)   (y==3)	=> z= true

### 5.3.5. Operatori logici pe biți (bitwise)

Acești operatori sunt operatori binari și acționează numai asupra operanzilor de tip întreg.

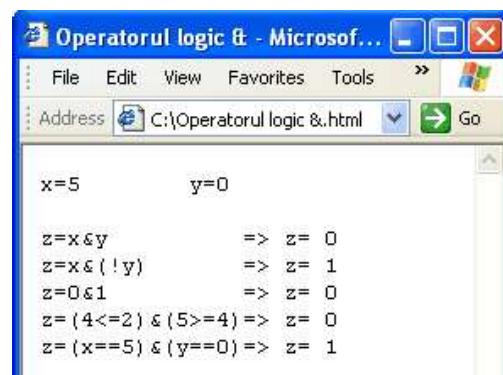
Operator	Descriere	Exemplu	Rezultat
&	<ul style="list-style-type: none"> <li>- operatorul logic și pe biți (bitwise and &amp;)</li> <li>- returnează 1 dacă ambii operanzi sunt 1, altfel returnează 0.</li> </ul>	0&0 1&0 0&1 1&1	0 0 0 1
	<ul style="list-style-type: none"> <li>- operatorul logic sau pe biți (bitwise or 1)</li> <li>- returnează 0 dacă ambii operanzi sunt 0, altfel returnează 1.</li> </ul>	0 0 1 0 0 1 1 1	0 1 1 1
^	<ul style="list-style-type: none"> <li>- operatorul logic sau exclusiv pe biți (bitwise xor ^)</li> <li>- returnează ^ dacă un singur operand este 1, altfel returnează 0.</li> </ul>	0^0 1^0 0^1 1^1	0 1 1 0
<<	<ul style="list-style-type: none"> <li>- operatorul logic de deplasare spre stânga a conținutului tuturor bițiilor operandului din stânga sa, cu un număr de poziții egal cu valoarea reținută de al doilea operand.</li> <li>- Pozițiile rămase libere (în dreapta) vor reține valoarea 0.</li> </ul>	<b>X=1010</b> X<<1 X<<12 <b>Y=1001</b> Y<<0001 Y<<2 Y<<5	2020 4136960 2002 4004 32032
>>	<ul style="list-style-type: none"> <li>- operator logic de deplasare spre dreapta</li> <li>- deplasează spre dreapta conținutul tuturor bițiilor operandului din stânga cu un număr de poziții egal cu valoarea reținută de al doilea operand.</li> </ul>	<b>X=1010</b> X>>1 X>>12 <b>Y=1001</b> Y>>0001 Y>>2 Y>>5	505 0 500 250 31

#### Exemplu de utilizare a operatorului &.

```

<html>
<head>
<title>Operatorul logic &</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=5, y=0;
document.writeln("x=5      y=0\n");
document.writeln("z=x&y      => z= ", 
    z=x&y);
document.writeln("z=x&(!y)      => z= ", 
    z=x&(!y));
document.writeln("z=0&1      => z= ", 
    z=0&1);
document.writeln("z=(4<=2) &(5>=4) => z= ", 
    z=(4<=2) &(5>=4));
document.writeln("z=(x==5) &(y==0) => z= ", 
    z=(x==5) &(y==0));
</script>
</pre>
</body>
</html>

```



### Exemplu de utilizare a operatorului |.

```
<html>
<head>
<title>Operatorul logic |</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=5, y=0;
document.writeln("x=5           y=0\n");
document.writeln("z=x|y          => z= ", z=x|y);
document.writeln("z=x|(!y)        => z= ", z=x|(!y));
document.writeln("z=0|1          => z= ", z=0|1);
document.writeln("z=(4<=2) | (5>=4)=> z=
", z=(4<=2) | (5>=4));
document.writeln("z=(x==5) | (y==0)=> z= ",
z=(x==5) | (y==0));
</script>
</pre>
</body>
</html>
```

The screenshot shows the Microsoft Internet Explorer browser window with the title "Operatorul logic | - Microsoft Internet Explorer". The address bar contains "C:\Operator\_logic.html". The page content displays the variables x=5 and y=0, followed by several writeln statements showing the results of logical OR operations:

Expression	Result
z=x y	=> z= 5
z=x (!y)	=> z= 5
z=0 1	=> z= 1
z=(4<=2)   (5>=4)=>	z= 1
z=(x==5)   (y==0)=>	z= 1

### Exemplu de utilizare a operatorului ^.

```
<html>
<head>
<title>
    Operatorul logic ^
</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=5, y=0;
document.writeln("x=5           y=0\n");
document.writeln("z=true^false   => z= ",
z=true^false);
document.writeln("z=x^y          => z= ", z=x^y);
document.writeln("z=x^(y^2)        => z= ", z=x^(y^2));
document.writeln("z=x^(!y)        => z= ", z=x^(!y));
document.writeln("z=1^1          => z= ", z=1^1);
document.writeln("z=(4<=2)^ (5>=4)=> z=
", z=(4<=2)^ (5>=4));
document.writeln("z=(x==5)^ (y==0)=> z= ",
z=(x==5)^ (y==0));
</script>
</pre>
</body>
</html>
```

The screenshot shows the Microsoft Internet Explorer browser window with the title "Operatorul logic ^ - Microsoft Internet Explorer". The address bar contains "C:\Operatorul logic ^.html". The page content displays the variables x=5 and y=0, followed by several writeln statements showing the results of logical XOR operations:

Expression	Result
z=true^false	=> z= 1
z=x^y	=> z= 5
z=x^(y^2)	=> z= 7
z=x^(!y)	=> z= 4
z=1^1	=> z= 0
z=(4<=2)^ (5>=4)=>	z= 1
z=(x==5)^ (y==0)=>	z= 0

### **Exemplu de utilizare a operatorului <<.**

```
<html>
<head><title>Operatorul logic <<</title></head>
<body>
<pre>
<script language="JavaScript">
var x=1000 , y=1111;
document.writeln("x=1000 (2)      y=1111 (2)\n");
document.writeln("x<<0001 => ", x<<0001);
document.writeln("x<<4     => ", x<<4);
document.writeln("y<<0001 => ", y<<0001);
document.writeln("y<<1     => ", y<<1);
document.writeln("y<<4     => ", y<<4);
</script>
</pre>
</body>
</html>
```

```
x=1000 (2)      y=1111 (2)
x<<0001 => 2000
x<<4     => 16000
y<<0001 => 2222
y<<1     => 2222
y<<4     => 17776
```

### **Exemplu de utilizare a operatorului >>.**

```
<html>
<head><title>Operatorul logic >>></title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=1000 , y=1111;
document.writeln("x=1000 (2)      y=1111
(2)\n");
document.writeln("x>>0001 => ", x>>0001);
document.writeln("x>>4     => ", x>>4);
document.writeln("y>>0001 => ", y>>0001);
document.writeln("y>>1     => ", y>>1);
document.writeln("y>>4     => ", y>>4);
</script>
</pre>
</body>
</html>
```

```
x=1000 (2)      y=1111 (2)
x>>0001 => 500
x>>4     => 62
y>>0001 => 555
y>>1     => 555
y>>4     => 69
```

### **5.3.6. Operator pentru siruri de caractere (string)**

Operatorul de concatenare (+) este un operator binar cu ajutorul căruia se pot uni două siruri de caractere.

### **Exemplu de utilizare a operatorului de concatenare.**

```
<html>
<head><title>Operatorul de
concatenare</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x="Eu nu strivesc ";
    y="corola de minuni";
    z="a lumii";
document.writeln(x+y+" "+z);
</script>
</pre>
</body>
</html>
```

```
Eu nu strivesc corola de minuni a lumii
```

### 5.3.7. Operatori de atribuire

Operatorii de atribuire se utilizează între un operand pe care îl scriem în partea stângă și o constantă, o variabilă sau o expresie în partea dreaptă a operatorului.

Nu se scriu expresii în partea stângă a unui operator de atribuire.

Operator	Exemplu prescurtat	Exemplu
=	x=3 z=2*x+y*x	x=3 z=2*x+y*x
+=	x+=1	x=x+1
-=	x-=1	x=x-1
*=	x*=3	x=x*3
/=	x/=3	x=x/3
%=	x%=3	x=x%3
<<=	x<<=3	x=x<<3
>>=	x>>=2	x=x>>2
>>>=	x>>>=5	x=x>>>5
&=	x&=6	x=x&6
^=	x^=2	x=x^2
=	x =y	x=x y

Exemplu de utilizare a operatorilor de atribuire.

```

<html>
<head>
<title>
    Operatorii de atribuire
</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=2;
    y=3;
document.writeln("x=2      y=3\n");
document.writeln("x+=y      => x=",x+=y);
document.writeln("x-=y+1 => x=",x-=y+1);
document.writeln("y%=2      => y=",y%=2);
document.writeln("y<<=x  => y=",y<<=x);
document.writeln("x>>>=y => y=",x>>>=y);
</script>
</pre>
</body>
</html>

```

```

x=2      y=3

x+=y      => x=5
x-=y+1 => x=1
y%=2      => y=1
y<<=x  => y=2
x>>>=y => y=0

```

### 5.3.8. Operatorul condițional

Operatorul condițional se utilizează în cadrul expresiilor condiționale.

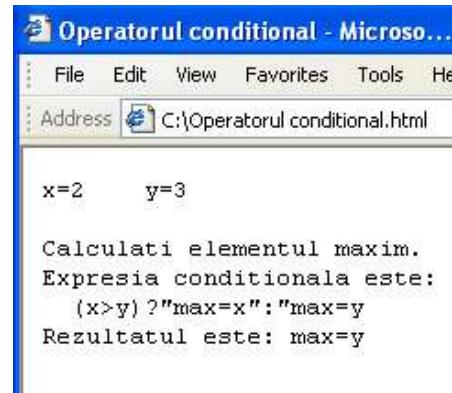
Forma generală:

(cond.)?expr.1:expr.2

Se evaluează condiția **cond.**, iar dacă este adevărată sau diferită de 0, atunci se execută expresia **expr.1**, altfel se execută expresia **expr.2**.

**Exemplu** de utilizare a operatorului condițional.

```
<html>
<head><title>Operatorul conditional</title>
</head>
<body>
<pre>
<script language="JavaScript">
var x=5;      y=7;
document.writeln("x=2      y=3\n");
document.writeln("Calculati elementul maxim.");
document.writeln("Expresia conditională este:");
document.writeln("  (x>y) ? \"max=x\" : \"max=y\"");
document.writeln("Rezultatul este:
", (x>y) ? "max=x" : "max=y");
</script></pre>
</body></html>
```



### 5.3.9. Operatorul typeof

Operatorul **typeof** returnează tipul de date conținut de operandul său.

Tipurile de date pe care le poate returna sunt:

- **string** – pentru siruri de caractere
- **number** – pentru numere
- **function** – pentru functiile JavaScript
- **object** – pentru obiectele JavaScript

**Exemplu** de utilizare a operatorului *typeof*.

```
<html>
<head><title>Operatorul typeof</title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("variabila\t\ttipul de
date\n");
var x1=-33.4;
document.writeln("x1=-33.4\t\t",typeof
x1);
x2=234;
document.writeln("x2=234\t\t",typeof
x2);
y="Lucian Blaga";
document.writeln("y=\\"Lucian
Blaga\\t",typeof y);
z=escape;
document.writeln("z=escape\t\t",typeof z);
t=Image;
document.writeln("t=Image\t\t",typeof
t);
</script>
</pre>
</body></html>
```

variabila	tipul de date
x1=-33.4	number
x2=234	number
y="Lucian Blaga"	string
z=escape	function
t=Image	object

## Evaluare

11. Faceți o analiză comparativă între JavaScript și Java, prezentând avantajele și dezavantajele utilizării lor.
12. Creați un script JavaScript ce afișează dacă variabilele  $x=3.14$  și  $y=44$  sunt diferite sau nu.

## 5.4. Instrucțiuni

În JavaScript instrucțiunile se clasifică în: instrucțiuni primitive, instrucțiuni condiționate (de decizie) și instrucțiuni repetitive. Instrucțiunile reprezintă acțiunile ce trebuie executate pentru a putea obține anumite rezultate.

### 5.4.1. Instrucțiuni primitive

Instrucțiunile primitive se clasifică în:

➤ **Instrucțiunea vidă ;**

Atunci când nu este necesară nici o prelucrare a datelor, putem utiliza instrucțiunea; Această instrucțiune nu returnează nici un rezultat, dar este necesară utilizarea ei.

➤ **Instrucțiunea compusă**

Pentru cazul în care trebuie executate mai multe instrucțiuni împreună se utilizează instrucțiunea compusă:

```
{  
    instrucțiune 1;  
    instrucțiune 2;  
    .....  
    instrucțiune n;  
}
```

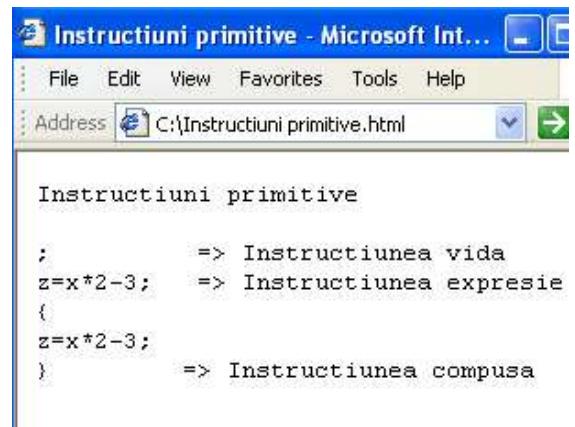
Instrucțiunea compusă se utilizează atunci când dorim să grupăm mai multe instrucțiuni, pentru a putea fi executate în ordine.

➤ **Instrucțiunea expresie**

O instrucțiune expresie poate fi: o expresie, o atribuire sau apelul unei funcții.

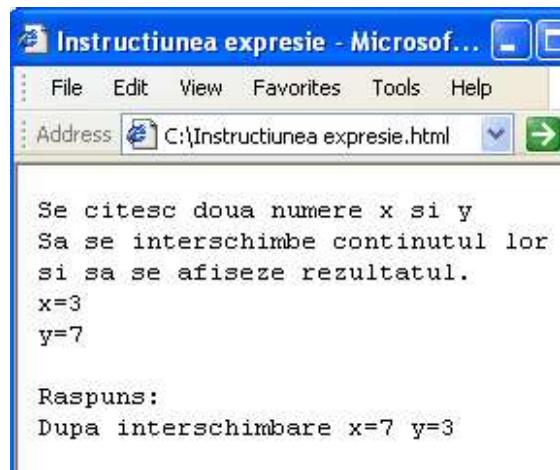
### Exemplu de utilizare a instrucțiunilor primitive.

```
<html>  
<head><title>Instrucțiuni  
primitive</title></head>  
<body>  
<pre><script language="JavaScript">  
document.writeln("Instrucțiuni  
primitive\n");  
{var x=2;  
z=x*2-3;  
;}  
document.writeln(";"           =>  
    Instrucțiunea vida");  
document.writeln("z=x*2-3;"   =>  
    Instrucțiunea expresie");  
document.writeln("{\nz=x*2-3;\n}\t=>  
    Instrucțiunea compusa");  
</script>  
</pre>  
</body></html>
```



**Exemplu** de utilizare a instrucțiunii expresie, în cadrul interschimbării conținutului a două variabile.

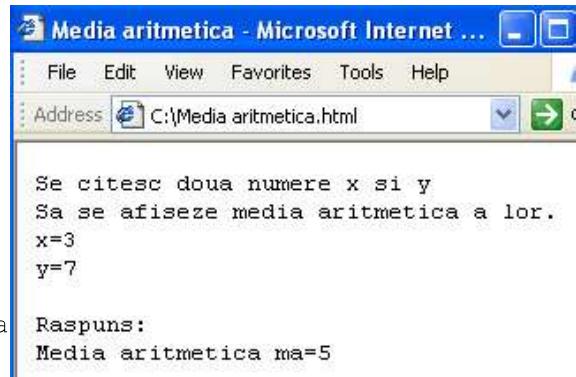
```
<html>
<head>
<title>
    Instructiunea expresie
</title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("Se citesc doua numere x
    si y\nSa se interschimbe continutul lor
    \nSi sa se afiseze rezultatul.");
x=eval(prompt("Dati x:"));
y=eval(prompt("Dati y:"));
document.writeln("x=",x,"ny=",y);
document.writeln("\nRaspuns:");
z=x;
x=y;
y=z;
document.writeln("Dupa interschimbare
    x=",x," y=",y);
</script>
</pre>
</body>
</html>
```



**OBS** În acest exemplu am utilizat funcția **eval**, ce ne permite evaluarea unor date citite de la tastatură prin intermediul unei căsuțe de dialog. Vom studia această funcție mai târziu.

**Exemplu** de utilizare a instrucțiunii expresie, pentru calcularea mediei aritmetice a două numere citite de la tastatură.

```
<html>
<head>
<title>
    Media aritmetica
</title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("Se citesc doua numere x
    si y\nSa se afiseze media aritmetica a lor.");
x=eval(prompt("Dati x:"));
y=eval(prompt("Dati y:"));
document.writeln("x=",x,"ny=",y);
document.writeln("\nRaspuns:");
ma=(x+y)/2;
document.writeln("Media aritmetica ma=",
    ma);
</script>
</pre>
</body>
</html>
```



#### 5.4.2. Instrucțiuni de decizie

Instrucțiunile condiționate (de decizie) se clasifică în:

##### ➤ **Instrucțiunea If**

Se utilizează pentru a lua o decizie în funcție de o condiție dată.

Sintaxa pentru instrucțiunea de decizie simplă:

```
if (cond.)
    { instr. 1; }
else
    { instr. 2; }
```

Principiul de execuție este următorul:

- se evaluează condiția;
- dacă se îndeplinește condiția (cond.), se execută instrucțiunea instr.1, altfel se execută instrucțiunea instr.2.

Sintaxa pentru instrucțiunea de decizie compusă:

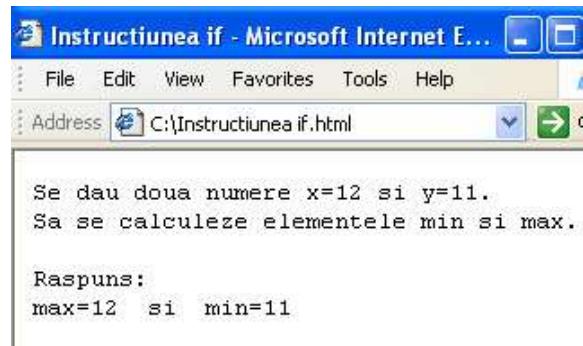
```
if (cond.)
    { instr. 01;
        instr. 02;
        .....
        instr. 0n; }
else
    { instr. 1;
        instr. 2;
        .....
        instr. n; }
```

Principiul de execuție este similar cu cel al instrucțiunii de decizie simplă.

- se evaluează condiția (cond.);
- dacă se îndeplinește condiția (cond.), se execută instrucțiunile de la instr.01 până la instr.0n;
- dacă nu se îndeplinește condiția (cond.), se execută instrucțiunile de la instr.1 până la instr.n.

#### Exemplu de utilizare a instrucțiunii if.

```
<html>
<head>
<title>
    Instructiunea if
</title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("Se dau doua numere x=12
    si y=11.\nSa se calculeze elementele
    min si max.\n");
document.writeln("Raspuns:");
var x=12; y=11;
if (x>y)
{
    document.writeln("max=",x,"tsi
    min=",y);
}
else
{
    document.write("max=",y);
    document.writeln("\tsi  min=",x);
}
</script>
</pre>
</body>
</html>
```

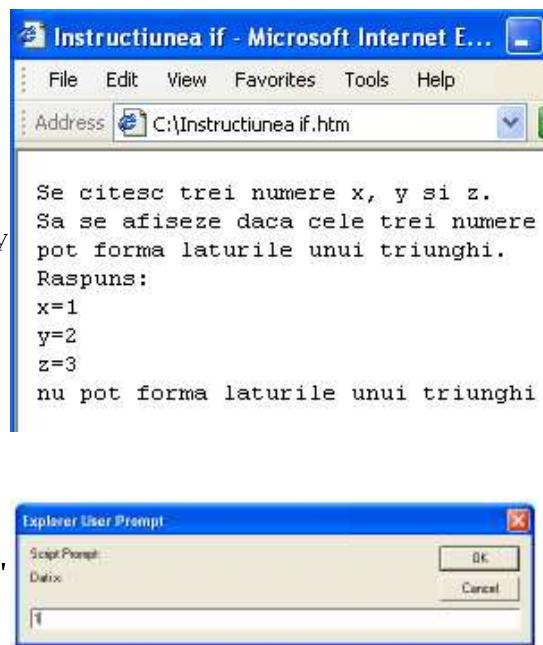


### Exemplu de utilizare a instrucțiunii if.

```

<html>
<head>
<title>Instructiunea if</title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("Se citesc trei numere x, y si z.
si z.\nSa se afiseze daca cele trei numere
pot forma laturile unui triunghi.");
document.writeln("Raspuns:");
x=eval(prompt("Dati x:"));
y=eval(prompt("Dati y:"));
z=eval(prompt("Dati z:"));
if
((x>0)&&(y>0)&&(z>0)&&(x+y>z)&&(x+z>y)&&
(y+z>x))
{document.writeln("x=",x,"ny=",y,"nz=",z,
"\npoate forma laturile unui triunghi");}
else
{document.writeln("x=",x,"ny=",y,"nz=",z,
"\nnu pot forma laturile unui triunghi");}
</script>
</pre>
</body>
</html>

```



### Exemplu de utilizare a instrucțiunii if.

$$F = \begin{cases} x-1 & , x < 1 \\ x & , x = 1 \\ x+5 & , x > 1 \end{cases}$$

```

<html>
<head>
<title>Instructiunea if</title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("Se citeste un numar
x.\nDaca x<1, F=x-1\n  x=1, F=x\n  x>1,
F=x+5");
x=eval(prompt("Dati numarul x:"));
document.writeln("Raspuns:");
if (x<1)
  document.writeln("F= ",x-1);
else
{
  if (x==1)
    document.writeln("F= ",x);
  else
    document.writeln("F= ",x+5);
}
</script>
</pre>
</body></html>

```

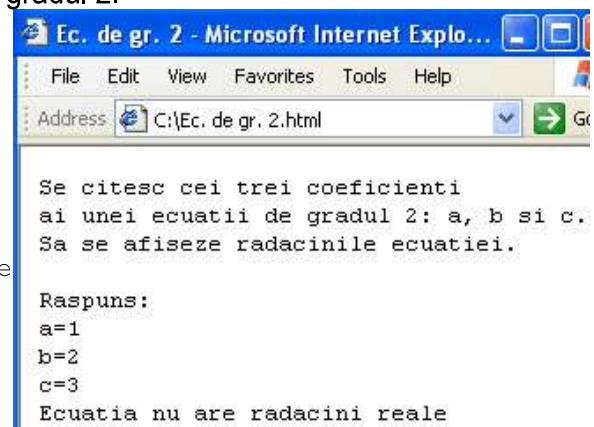


### **Exemplu de utilizare a instrucțiunii if, în ecuația de gradul 2.**

```

<html>
<head><title>Ec. de gr. 2</title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("Se citesc cei trei
coeficienti\nai unei ecuatii de gradul
2: a, b si c.\nSa se afiseze radacinile
ecuatiei.");
a=eval(prompt("Dati a:"));
b=eval(prompt("Dati b:"));
c=eval(prompt("Dati c:"));
document.writeln("\nRaspuns:");
document.writeln("a=",a,"nb=",b,"nc=",c
);
if (a==0)
{x=-c/b;
document.writeln("Ecuatia este de
gradul I\nSolutia ecuatiei este x=",x);}
else
{d=b*b-4*a*c;
if (d<0)
{document.writeln("Ecuatia nu are
radacini reale");}
else
{x1=(-b+Math.sqrt(d))/(2*a);
x2=(-b-Math.sqrt(d))/(2*a);
document.writeln("x1=",x1);
document.writeln("x2=",x2);} }
</script>
</pre>
</body></html>

```



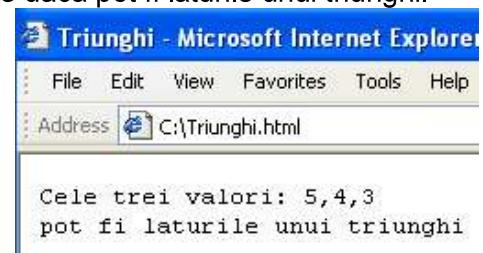
**OBS** În acest exemplu am utilizat metoda `sqrt()` ce returnează rădăcina pătrată a unei valori numerice. Aceasta este o metodă a obiectului *Math*.

### **Exemplu Se citesc trei numere a, b și c. Să se precizeze dacă pot fi laturile unui triunghi.**

```

<html>
<head><title>Triunghi</title></head>
<body>
<pre>
<script language="JavaScript">
a=eval(prompt("Dati a:"));
b=eval(prompt("Dati b:"));
c=eval(prompt("Dati c:"));
if((a>0)&&(b>0)&&(c>0)&&(a+b>c)&&(a+c>b)&&(b+
c>a))
document.writeln("Cele trei valori:
",a,".",b,".",c,"npot fi laturile unui
triunghi");
else
document.writeln("Cele trei valori:
",a,".",b,".",c,"NU pot fi laturile unui
triunghi");
</script></pre>
</body></html>

```



### ➤ **Instrucțiunea switch..case**

Instrucțiunea de decizie multiplă se utilizează atunci când, în urma evaluării unei expresii, trebuie să optăm între mai multe cazuri date. Același lucru îl putem realiza și cu ajutorul instrucțiunii if. Diferența constă în faptul că dacă avem de optat între mai mult de două cazuri, folosirea instrucțiunii **if** devine greoare, caz în care putem apela la instrucțiunea **switch**.

Sintaxa pentru instrucțiunea de decizie multiplă simplă:

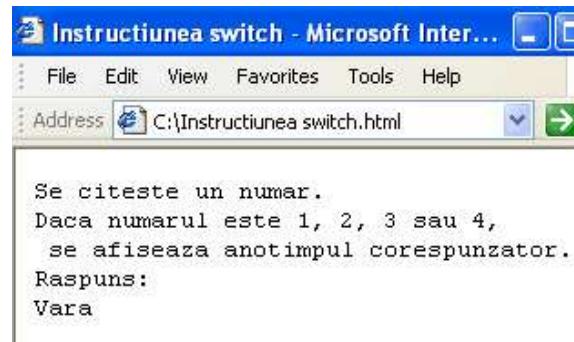
```
switch (expresie)
{
    case valoare_1;
        instr_1; break;
    case valoare_2;
        instr_2; break;
    .....
    case valoare_n;
        instr_n; break;
    [ default: instr_n+1];
}
```

Principiul de execuție este următorul:

- se evaluatează expresia;
- dacă valoarea expresiei este egală cu *valoare\_1*, se execută *instr\_1*
- dacă valoarea expresiei nu este egală cu *valoare\_1*, se verifică dacă este egală cu *valoare\_2*, pentru a se executa *instr\_2*,
- dacă valoarea expresiei nu este egală cu *valoare\_2* se repetă pașii până la *valoare\_n*
- dacă valoarea expresiei nu este egală cu nici o valoare dintre: *valoare\_1...valoare\_n*, se execută *instr\_n+1*

### **Exemplu** de utilizare a instrucțiunii **switch**.

```
<html>
<head>
<title>
    Instructiunea switch
</title>
</head>
<body>
    <pre>
<script language="JavaScript">
document.writeln("Se citeste un
    numar.\nDaca numarul este 1, 2, 3 sau 4,
    se afiseaza anotimpul corespunzator.
Raspuns:
Vara
x=eval(prompt("Dati numarul corespunzator
    anotimpului"));
document.writeln("Raspuns:");
switch(x)
{
    case 1: document.writeln("Primavara");
        break;
    case 2: document.writeln("Vara");
        break;
    case 3: document.writeln("Toamna");
        break;
    case 4: document.writeln("Iarna");
        break;
    default: document.writeln("Dati un numar
        intre 1 si 4!");
}
</script>
</pre>
</body>
</html>
```



### 5.4.3. Instrucțiuni repetitive

Instrucțiunile repetitive se clasifică în:

- instrucțiuni cu număr cunoscut de pași: **for**
- instrucțiuni cu număr necunoscut de pași: **while și do..while**

#### ➤ **Instructiunea for**

Sintaxa pentru instructiunea repetitivă cu număr cunoscut de pași **for**:

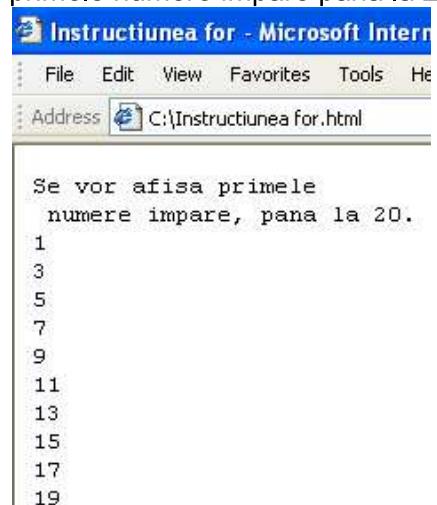
**for (val1; cond.;pas)  
instructiune**

Principiul de execuție este următorul:

- se evaluează condiția **cond.**
- dacă se îndeplinește condiția **cond.**, se incrementează valoarea inițială **val1**, cu valoarea pasului **pas**, și se repetă execuția instructiunii până când nu se mai îndeplinește condiția.

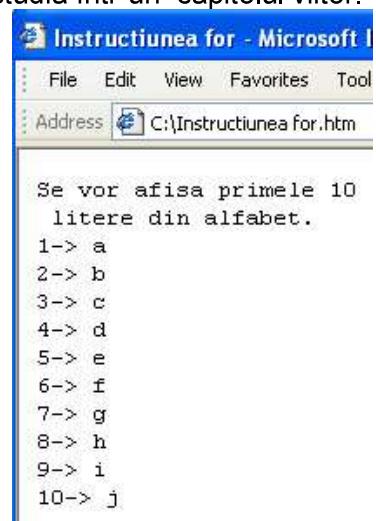
**Exemplu** de utilizare a instructiunii **for**. Se vor afișa primele numere impare până la 20.

```
<html>
<head>
<title>Instructiunea for</title>
</head>
<body>
<pre>
<script language="JavaScript">
var i=1;
document.writeln("Se vor afisa primele\n numere
impare, pana la 20.");
for(i=1; i<=20; i=i+2)
document.writeln(i);
</script>
</pre>
</body></html>
```



**Exemplu** În acest exemplu, se vor afișa literele de la a la j. Pentru aceasta, am inițializat variabila x cu valoarea Unicode a caracterului "a", de la care am dorit să începem afișarea. Condiția este să afișeze până la valoarea Unicode a caracterului "j", cu pasul 1, toate caracterele corespunzătoare acestor valori Unicode. Metodele **charCodeAt()** și **fromCharCode()** le vom studia într-un capitolul viitor.

```
<html>
<head>
<title>Instructiunea for</title>
</head>
<body>
<pre>
<script language="JavaScript">
var y=1;
document.writeln("Se vor afisa primele
10\n litere din alfabet.");
for(x="a".charCodeAt(0);
x<="j".charCodeAt(0); x++)
{document.writeln(y,"->
",String.fromCharCode(x));
y++;}
</script>
</pre>
</body>
</html>
```



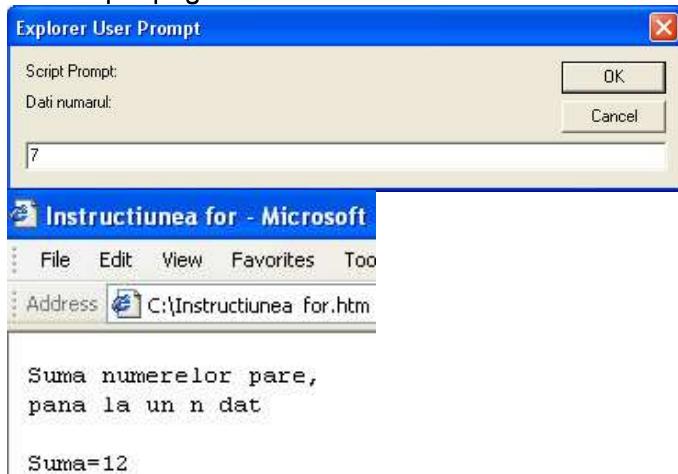
**Exemplu** Vom afișa în ordine descrescătoare numerele cuprinse între 15 și 7.

```
<html>
<head>
<title>Instructiunea for</title>
</head>
<body>
<pre>
<script language="JavaScript">
var i=15;
document.writeln("Se vor afisa in ordine
descrescatoare,\nnumerele de la 15 la 7.");
for(i=15;i>=7;i--)
document.writeln(i);
</script></pre>
</body></html>
```

Se vor afisa in ordine descrescatoare,  
numerele de la 15 la 7.  
15  
14  
13  
12  
11  
10  
9  
8  
7

**Exemplu** Vom afișa suma numerelor pare până la un n citit de la tastură. Pentru a realiza acest lucru, am utilizat metoda *prompt()* ce deschide o fereastră în care se cere numărul n și funcția *eval()* ce evaluează valoarea introdusă de către utilizator. Datele de început și de sfârșit sunt afișate în corpul pagini web.

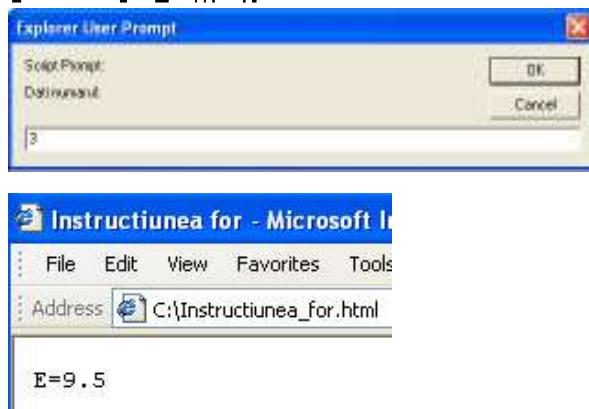
```
<html>
<head>
<title>Instructiunea for </title>
</head>
<body>
<pre>
<script language="JavaScript">
document.writeln("Suma numerelor
pare,\npana la un n dat\n");
n=eval(prompt("Dati numarul:"));
var suma=0;
for (i=0; i<=n; i=i+2)
  suma=suma+i;
document.writeln("Suma=", suma);
</script>
</pre>
</body>
</html>
```



**Exemplu** Să se calculeze și să se afișeze următoarea expresie:

$$E = \frac{1*3}{1} + \frac{2*4}{1*2} + \frac{3*5}{1*2*3} + \dots + \frac{n*(n+2)}{1*2*\dots*n}$$

```
<html>
<head>
<title>Instructiunea for</title>
</head>
<body><pre>
<script language="JavaScript">
n=eval(prompt("Dati numarul:"));
var e=0;p=1;
for (i=1; i<=n; i++)
  {p=p*i;
   e=e+(i*(i+2))/p;}
document.writeln("E=", e);
</script></pre>
```



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

**Exemplu** Să se afișeze minimul și maximul din n numere citite de la tastatură.

```
<html>
<head>
<title>
    Instructiunea for
</title>
</head>
<body>
<pre>
<script language="JavaScript">
n=eval(prompt("Dati numarul n="));
x=eval(prompt("Dati numarul 1"));
    min=x;
    max=x;
for (i=2; i<=n; i++)
{
    x=eval(prompt("Dati numarul ",i));
    if (x<min)
        min=x;
    if (x>max)
        max=x;
}
document.writeln("Elementul
    minim=",min);
document.writeln("Elementul
    maxim=",max);
</script>
</pre>
</body></html>
```

**Exemplu** Să se afișeze dacă un număr citit de la tastatură este prim sau nu. Am utilizat funcția `sqrt()` ce returnează radicalul unui număr.

```
<html>
<head>
<title>Numar prim</title>
</head>
<body>
<pre>
<script language="JavaScript">
n=eval(prompt("Dati un numar:"));
prim=1;
for(i=2;i<=Math.sqrt(n);i++)
    if (n%i==0)
        prim=0;
if (prim==1)
    document.writeln("Numarul ",n," este
        prim.");
else
    document.writeln("Numarul ",n," nu este
        prim.");
</script>
</pre>
</body>
</html>
```

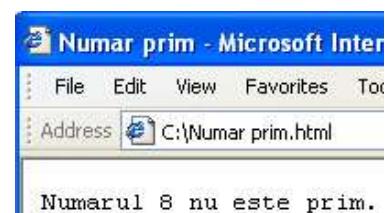
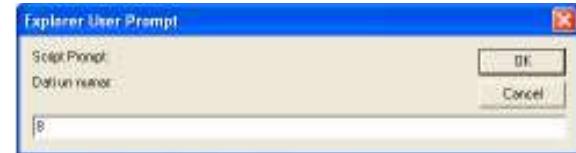
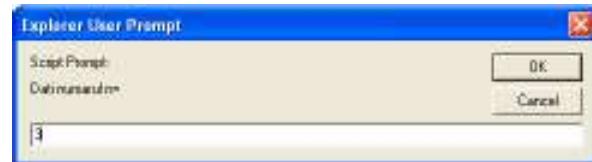
#### ➤ **Instructiunea while**

Sintaxa pentru instructiunea repetitivă cu test inițial și cu număr necunoscut de pași **while**:

**while ( cond.)**  
         **instructiune**

Principiul de execuție este următorul:

- se evaluatează condiția **cond.**;
- cât timp se îndeplinește condiția **cond.**, se execută instructiunea **instructiune**;
- când valoarea condiției devine 0 (zero), adică fals, se trece la următoarea instructiune.



**Exemplu** de utilizare a instrucțiunii **while**. Să se afișeze valoarea corespunzătoare *Unicode* pentru toate numerele introduse de la tastatură, până când introducem cifra zero.

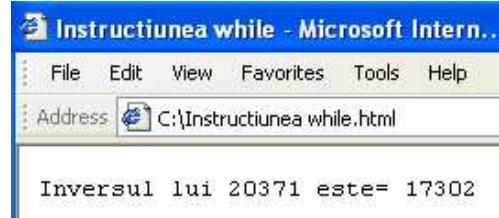
```
<html>
<head>
<title>Instructiunea while </title>
</head>
<body>
<pre>
<script language="JavaScript">
n=prompt("Dati un numar:","");
while (n!=0)
{
  document.writeln("nr=",
    n.charCodeAt(0));
  n=prompt("Dati un numar:","");
}
</script>
</pre>
</body></html>
```



nr=65

**Exemplu** Să se afișeze inversul unui număr citit de la tastatură.

```
<html>
<head>
<title>Instructiunea while</title>
</head>
<body>
<pre>
<script language="JavaScript">
n=prompt("Dati un numar:");
x=n;
inv=0;
while (n!=0)
{inv=inv*10+n%10;
 n=Math.floor(n/10);}
document.writeln("Inversul lui ",x," este=
",inv);
</script>
</pre>
</body></html>
```



Inversul lui 20371 este= 17302

**Exemplu** Să se calculeze și să se afișeze produsul primelor n numere naturale (n factorial).

```
<html>
<head>
<title>n!</title>
</head>
<body>
<pre>
<script language="JavaScript">
n=eval(prompt("Dati un numar:"));
i=1;p=1;
while(i<=n)
{
  p=p*i;
  i=i+1;
}
document.writeln("n!= ",p);
</script></pre>
</body></html>
```



n!= 6

**Exemplu** Se citesc numere până la întâlnirea cifrei zero (0). Să se afișeze cel mai mic și cel mai mare număr citit.

```
<html>
<head>
<title>
    Min si max
</title>
</head>
<body>
<pre>
<script language="JavaScript">
n=eval(prompt("Dati un numar:"));
min=n;
max=n;
while(n!=0)
{
    n=eval(prompt("Dati un numar:"));
    if (n<min)
        min=n;
    if (n>max)
        max=n;
}
document.writeln("Minimul= ",min);
document.writeln("Maximul= ",max);
</script>
</pre>
</body>
</html>
```



**Exemplu** Se citesc pe rând cifrele unui număr. Să se afișeze numărul obținut prin aranjarea cifrelor în ordinea citirii lor.

```
<html>
<head>
<title>
    Cifre
</title>
</head>
<body>
<pre>
<script language="JavaScript">
var n=0;
x=eval(prompt("Dati prima cifra:"));
while ((x>=0)&&(x<=9))
{
    n=n*10+x;
    x=eval(prompt("Dati urmatoarea cifra:"));
}
document.writeln("Numarul obtinut este: ",
    n);
</script>
</pre>
</body>
</html>
```

Numarul obținut este: 46203021

**Exemplu** Se citește un număr în baza 2. Să se afișeze acel număr în baza 10. De exemplu, dacă se citește  $111_2$ , se va afișa  $7_{10}$ .

```
<html>
<head>
<title>Baza zece</title>
</head>
<body>
<pre>
<script language="JavaScript">
var putere=1;
var bzece=0;
var k=0;
bdoi=eval(prompt("Dati numarul in baza
2:"));
x=bdoi;
while (x!=0)
{
    r=x%10;
    if ((r!=0)&&(r!=1))
        k=1;
    bzece=bzece+r*putere;
    putere=putere*2;
    x=Math.floor(x/10);
}
if (k==0)
    document.writeln("Numarul ",bdoi," in baza
10 este: ",bzece);
else
    document.writeln("Numarul nu este in baza
2");
</script>
</pre>
</body>
</html>
```

#### ➤ **Instrucțiunea do...while**

Sintaxa pentru instrucțiunea repetitivă cu test final și cu număr necunoscut de pași **do...while**:

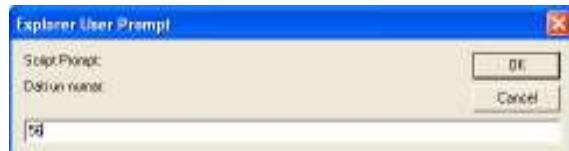
```
do
{
    instrucțiune 1;
    instrucțiune 2;
    .....
    instrucțiune n;
}
while ( cond.)
```

Principiul de execuție este următorul:

- se execută instrucțiunile;
- se evaluatează condiția **cond.**;
- dacă valoarea rezultată este diferită de zero (adevărat), se repetă execuția instrucțiunii până când aceasta devine zero (fals);
- se trece la instrucțiunea următoare

**Exemplu** Se citește un număr natural pozitiv. Să se descompună în factori primi și să se afișeze rezultatul.

```
<html>
<head>
<title>Factori primi</title>
</head>
<body>
<pre>
<script language="JavaScript">
var i=2;
```



```

n=eval(prompt("Dati un număr:"));
do
{
    putere=0;
    while (n%i==0)
    {
        putere=putere+1;
        n=Math.floor(n/i);
    }
    if (putere!=0)
        document.writeln(i," la puterea",
",putere);
    i++;
}
while (n!=1)
</script></pre>
</body></html>

```



**Exemplu** Se citește un număr natural pozitiv. Să se afișeze câte cifre conține numărul.

```

<html>
<head>
    <title>Nr. cifre</title>
</head>
<body>
<pre>
<script language="JavaScript">
var s=0;
n=eval(prompt("Dati un număr:"));
x=n;
if (n>=0)
{
    do
        { n=Math.floor(n/10);
            s++; }
    while (n!=0)
        document.writeln("Numarul ",x," are
",s," cifre.");
}
else
    document.writeln("Numarul ",x," nu
este pozitiv.");
</script>
</pre>
</body>
</html>

```



Numarul -23 nu este pozitiv.

sau

Numarul 40801200 are 8 cifre.

## Evaluare

1. Care sunt operatorii aritmetici?
2. Unde se pot scrie script-urile JavaScript?
3. Ce fel de operator este + (plus)?
4. Care sunt instrucțiunile repetitive?
5. Realizați un script ce permite calcularea suma numerelor pare până la un număr n citit de la tastatură.
6. În interiorul căror etichete se încadrează codul sursă JavaScript?
  - a) <js> și </js>
  - b) <script> și </script>

- c) <JavaScript> și <JavaScript>
  - d) <java> și <java>
7. Unde este corect să poziționăm codul JavaScript?
- a) În secțiunile: < head > sau < body >
  - b) În secțiunea < title >
  - c) În secțiunea < head >
  - d) În secțiunea < body >
8. Operatorii unari sunt:
- a) + / %
  - b) + -
  - c) \* /
  - d) + - %
9. Cum afișăm pe ecran: "Salutari!"?
- a) WriteLn ("Salutari!")
  - b) document.writeLn (Salutari!)
  - c) alert ("Salutari!")
  - d) document.writeLn ("Salutari!")
10. Cum afișăm într-un buton alert: "Salutari!"?
- a) writeln . alert ("Salutari!")
  - b) msgBox ("Salutari!")
  - c) alert ("Salutari!")
  - d) alert . writeln ("Salutari!")
11. Cum scriem: "Dacă x este mai mic decât y, atunci minim=x, altfel minim=y"?
- a) if (x < y) then minim=x else minim=y
  - b) (x < y)?minim=x:minim=y
  - c) minim=(x < y)?x:y
  - d) if x< y minim=x else minim=y

## Capitolul 6.

# Funcții JavaScript

### Obiective:

- să înțeleagă noțiunea de funcție JavaScript
- să poată utiliza o funcție JavaScript acolo unde este necesar.
- să înțeleagă noțiunile de apelare și parametrii ai funcțiilor
- să poată crea un script ce utilizează funcții predefinite JavaScript

### 6.1. Introducere

Atunci când dorim ca pagina Web să îndeplinească o anumită cerință, creăm o funcție JavaScript ce va avea ca scop executarea acelei cerințe. Funcțiile JavaScript le putem crea noi sau putem utiliza funcții predefinite ale limbajului, cum ar fi: `escape()`, `eval()`, pe care o să le studiem în acest capitol. Apelarea unei funcții se face prin numele ei, atunci când este necesar. O funcție poate fi poziționată în partea de sus a paginii, adică în antet sau în exterior, într-un alt fișier.

### 6.2. Definirea funcțiilor

Definirea unei funcții constă în cuvântul cheie **function**, urmat de: numele funcției, o listă cu parametri separați prin virgulă și instrucțiunile scrise între acolade. O funcție poate fi apelată de către o altă funcție sau de către ea însăși (se autoapelează).

Sintaxa unei funcții ce conține parametri:

Partea de declarare **function Nume (variabila1, variabila2,..., variabilan)**  
a funcției.  
{  
    **instrucțiuni** ;  
}

---

Partea de apelare a **Nume (variabila1, variabila2,..., variabilan)**  
funcției.

Funcțiile pot fi declarate și apelate chiar dacă nu conțin parametri.

Partea de declarare **function Nume ()**  
a funcției.  
{  
    **instrucțiuni** ;  
}

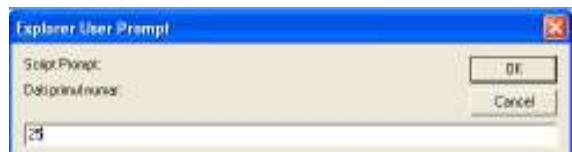
---

Partea de apelare a **Nume ()**  
funcției.

Pentru a rezolva problema din exemplul de mai jos, am apelat funcția `cmmdc`, care, pentru a returna cel mai mare divizor comun dintre două valori `x` și `y` primite ca parametri, se autoapelează până când se îndeplinește condiția `y=0`. Autoapelarea unei funcții poartă numele de recursivitate.

**Exemplu** de utilizare a unei funcții JavaScript în `head`. Se citesc două numere naturale de la tastatură. Să se afișeze cel mai mare divizor comun al celor două numere, utilizând algoritmul lui Euclid.

```
<html>
<head>
<title>
    Functie JavaScript in head
</title>
<script language="JavaScript">
```

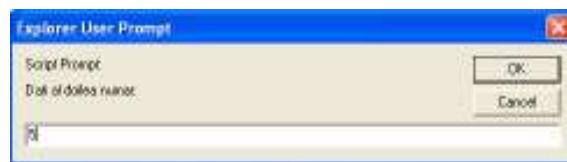


```

function cmmdc(x,y)
    if (y==0)
        return x;
    else
        return cmmdc(y,x%y);    }
</script>
</head>
<body><pre>
<script language="JavaScript">
x=eval(prompt("Dati primul numar:"));
y=eval(prompt("Dati al doilea numar:"));
    document.writeln("cmmdc= ",cmmdc(x,y));
</script>
</pre>
</body></html>

```

**OBS** În exemplul de mai sus am utilizat **return** pentru a întoarce un rezultat prin numele funcției.



cmmdc= 5

### 6.3. Apelarea funcțiilor

Definirea unei funcții nu înseamnă neapărat că acea funcție se va apela. Putem defini funcții care nu se vor executa niciodată, dar și funcții care pot fi apelate de una sau mai multe funcții. Apelarea unei funcții presupune executarea unei acțiuni specifice, pentru anumiți parametri.

În exemplul de mai sus, se citesc doi parametri x și y, după care se efectuează apelul funcției pentru parametri citiți.

Mai jos, putem vedea o funcție fără parametri, care va fi apelată doar dacă executăm click pe un buton. În acest exemplu, nu au fost necesari parametri, de aceea nu există.

**Exemplu** de utilizare a unei funcții JavaScript în *head* fără parametri. Se apasă pe butonul pe care scrie "Apasati!" și va apărea o fereastră în care scrie "Salut!".

```

<html>
<head>
<title>Functie fara parametri </title>
<script type="text/JavaScript">
function f()
{
    alert("Salut!")
}
</script>
</head>
<body>
<input type="button" onclick="f()"
       value="Apasati!">
</body>
</html>

```



**Exemplu** de utilizare a unei funcții JavaScript exterioară. Pentru aceasta, se crează o funcție simplă, ca cea de mai jos, care se salvează cu extensia **.js**. În cazul nostru, am salvat fișierul cu numele **ff.js**.

```

function f()
{
alert('Salutari      dintr-un      script
exterior!');}

```



Odată salvat, fișierul poate fi apelat din interiorul unui script, ca în exemplul de mai jos. După ce este apelat fișierul ff.js, putem utiliza funcția sau funcțiile pe care le conține fișierul respectiv. În cazul nostru, apelăm funcția f().

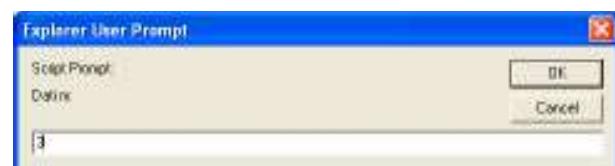
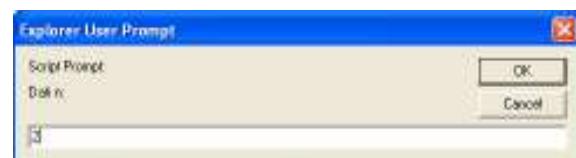
```
<html>
<head>
<title>Functie externa</title>
<SCRIPT language="JavaScript" SRC="ff.js">
</SCRIPT>
</head>
<body>
<A HREF="javascript:f()">Dati click pentru
un mesaj!</A>
</body></html>
```

Dati click pentru un mesaj!

**OBS** Apelarea fișierului se face din antet (head).

**Exemplu** Se citesc două numere naturale m și n. Scriptul următor va calcula funcția lui Ackermann. La încărcarea paginii, se apelează o funcție f() care citește două valori n și m, după care apelează funcția recursivă Ackermann, de parametrii n și m citiți. Rezultatul se va afișa pe ecran.

```
<html>
<head>
<title>Functie JavaScript</title>
</head>
<body onLoad="f()">
<script language="JavaScript">
function f()
{ n=eval(prompt("Dati n: "));
  m=eval(prompt("Dati m: "));
  document.writeln(Ackermann(m, n));
}
function Ackermann(m,n)
{
  if (m==0)
    return n+1;
  else
    if (n==0)
      return Ackermann(m-1, 1);
    else
      return
          Ackermann(m-1,
Ackermann(m, n-1));
}
</script>
</body></html>
```



**Exemplu** În acest exemplu avem funcția **culori**, care va putea fi apelată de mai multe ori, pentru valori diferite. Dacă dăm click pe butonul pe care scrie **Rosu**, se va transmite funcției prin variabila x, valoarea **Red**. Această valoare va defini culoarea de fundal a unei pagini Web.

```
<html>
<head>
<title>Functii</title>
<script type="text/JavaScript">
function culori(x)
{
  document.writeln("<body bgcolor='"+x+">");
  document.writeln("Pagina cu fundal de culoare");
  document.writeln("diferite.");
}
</script>
</head>
<body>
<form>
Dati click pe un buton pentru a afisa
<br />
```



```

culoare diferita a fundalului.<br /><br />
<input type="button" onclick="culori('Red')"
       value="Rosu">
<input type="button" onclick=
       "culori('Yellow')" value="Galben">
<input type="button" onclick="culori('Blue')"
       value="Albastru">
</form>
</body>
</html>

```

**Exemplu** Un exemplu similar cu cel de mai sus, dar în care avem doi parametri transmiși la apelul realizat de către evenimentul *onClick*. Cei doi parametri vor modifica valorile implicate ale culorii fundalului și textului.

```

<html>
<head>
  <title>Functii</title>
<script type="text/JavaScript">
function culori(x,y)
{
  document.writeln("<body bgcolor='"+x+"' text ="
    "+y+"");
  document.writeln("<b> Pagina cu fundal si text"
    "<br /> de culori diferite.");
}
</script>
</head>
<body>
<form>
Dati click pe un buton pentru a afisa <br /> o
culoare diferita a fundalului si a <br />
textului<br /><br />
<input type="button" onclick=
       "culori('Red','blue')" value="Rosu">
<input type="button" onclick=
       "culori('black','Yellow')" value="Galben">
<input type="button" onclick="culori('green')"
       value="Albastru">
</form>
</body>
</html>

```



## 6.4. Parametri funcțiilor

Parametri pe care-i utilizăm atunci când apelăm o funcție poartă numele de parametri efectivi sau actuali. Ei specifică valorile care vor fi prelucrate în cadrul funcției.

Parametri aflați în partea de declarare a funcției poartă numele de parametri formali. Atât parametri formalii cât și parametri efectivi pot lipsi atunci când utilizăm o funcție. Numărul parametrilor formalii trebuie să fie egal cu numărul parametrilor efectivi.

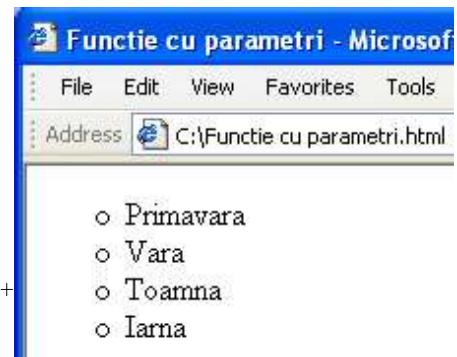
**OBS** Excepție de la această regulă face proprietatea *arguments* ce poate fi utilizată cu un număr diferit de parametri la apelul funcției. Pentru a determina numărul de parametri din apel, utilizăm în cadrul funcției, *arguments.length*. *arguments* este un masiv (vector unidimensional) ce are ca primă valoare *arguments[0]*.

**Exemplu** În acest exemplu, se apelează funcția *lista* din interiorul unei instrucțiuni de afișare, prin *lista("U", "Primavara", "Vara", "Toamna", "Iarna")*. La execuția funcției, se parcurg parametri cu ajutorul *arguments.length*. Parametrul "U" de la apel se introduce în funcție pentru a afișa o listă neordonată (Unordered Lists) UL. Pentru a afișa o listă ordonată, trebuie să înlocuim U cu O, de la Ordered lists.

```

<html>
<head>
<title>
    Functie cu parametri
</title>
<script type="text/JavaScript">
function lista(x)
{
    document.write("<" + x + "L type=circle>")
    for (var i=1; i<lista.arguments.length; i++)
        document.write("<LI>" +
            lista.arguments[i] + "</LI>")
    document.write("</" + x + "L>")
}
</script>
</head>
<body>
<script type="text/JavaScript">
write(lista("U", "Primavara", "Vara", "Toamna",
    "Iarna"));
</script>
</body>
</html>

```

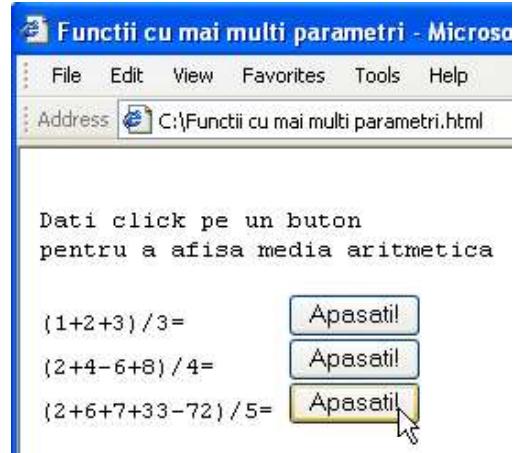


**Exemplu** Un exemplu asemănător cu cel de mai sus, dar în care aceeași funcție va putea fi apelată de mai multe ori, pentru a realiza calcularea mediei cu mai mulți sau mai puțini parametri.

```

<html>
<head>
<title>
    Functii cu mai multi parametri
</title>
<script type="text/JavaScript">
function medie()
{
    var s = 0
    for(var i=0; i<arguments.length; i++)
        s+=arguments[i] ;
    var ma=s/arguments.length
    alert(ma);
}
</script>
</head>
<body>
<pre>
Dati click pe un buton pentru a afisa media aritmetica
<br /><br />
(1+2+3)/3= <input type="button" onclick=
    "medie(1,2,3)" value="Apasati!">
(2+4-6+8)/4= <input type="button" onclick=
    "medie(2,4,-6,8)" value="Apasati!">
(2+6+7+33-72)/5= <input type="button" onclick=
    "medie(2,6,7,33,-72)"
value="Apasati!">
</pre>
</body>
</html>

```



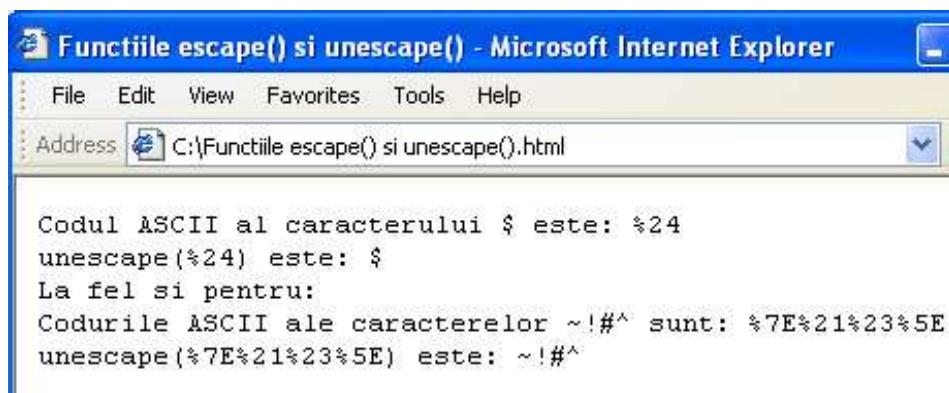
## 6.5. Funcții predefinite

În acest capitol vom studia cele mai importante funcții predefinite JavaScript. Printre cele mai utilizate sunt: eval, isFinite, isNaN, parseInt, parseFloat, Number, String, encodeURI, decodeURI, encodeURIComponent, decodeURIComponent

**Functiile escape() și unescape()** transformă seturi de caractere admise de ISO Latin-1 în coduri ASCII (scrise în hexazecimal) și invers.

Pentru început vom face un exemplu simplu de transformare a sirurilor de caractere în codurile ASCII corespunzătoare și invers.

**Exemplu** Se transformă mai întâi caracterul \$ în codul ASCII corespunzător și înapoi din codul ASCII în caracterul initial. După aceasta, se transformă un sir de caractere în codul ASCII corespunzător și invers.



```
<html>
<head>
    <title>Functiile escape() si unescape() </title>
</head>
<body>
<pre>
<script language="JavaScript">
    x=escape("$");
    document.writeln("Codul ASCII al caracterului $ este: "+x);
    y=unescape(x);
    document.writeln("unescape("+x+") este: "+y);
    document.writeln("La fel și pentru: ");
    x=escape("~!#^");
    document.writeln("Codurile ASCII ale caracterelor ~!#^ sunt: "+x);
    y=unescape(x);
    document.writeln("unescape("+x+") este: "+y);
</script>
</pre>
</body>
</html>
```

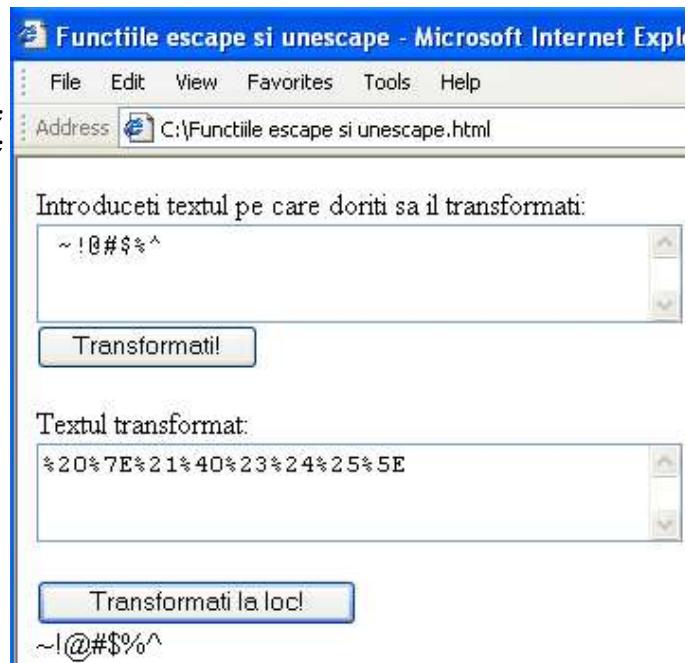
Chiar dacă încă nu am parcurs toate elementele necesare pentru exemplul următor, vă puteți face o impresie despre modul de transformare interactiv pe care-l puteți realiza cu ajutorul funcțiilor JavaScript.

**Exemplu** În corpul paginii introducem un formular care apelează două funcții din interiorul unui script. Scriptul este apelat la începutul paginii, în antet (head), prin src="FTR.js".

```
<html>
<head>
<title>
    Functiile escape si unescape
</title>
<script type="text/javascript" src="FTR.js"> </script>
</head>
<body>
<form name="x" action="">
    Introduceti textul pe care doriti sa il transformati:<br />
    <textarea name="tr" cols="40" rows="3">
    </textarea><br />
    <input type="button" name="action" value="Transformati!" onClick="Transform()"/>
<br/><br />
    Textul transformat:<br />
    <textarea name="re_tr" cols="40" rows="3">
    </textarea><br /><br />
    <input type="button" name="action" value="Transformati la loc!" onClick="ReTransform()"/> <br />
    <div id="RTr">
    </div>
</form>
</body>
</html>
```

Scriptul "FTR.js" este cel de mai jos. În acest script avem două funcții: Transform(), care transformă caracterele în coduri ASCII și ReTransform() care convertește codurile ASCII în caracterele introduse inițial.

```
function Transform()
{
    cod = escape(x.tr.value);
    cod = cod.replace(/\\/g,"%2F");
    cod = cod.replace(/\?/g,"%3F");
    cod = cod.replace(/=/g,"%3D");
    cod = cod.replace(/\&/g,"%26");
    cod = cod.replace(/\@/g,"%40");
    x.re_tr.value = cod;
}
function ReTransform()
{
    RTr.innerHTML =
        unescape(x.re_tr.value);
}
```



**Functia eval()** evaluează un sir de caractere, o expresie, o variabilă sau anumite proprietăți ale unor obiecte existente. Sintaxa funcției este: **eval(string)**  
Dacă **string** reprezintă o expresie, atunci **eval** va evalua acea expresie și va returna rezultatul.

**Exemplu** de utilizare a funcției **eval()**. Se citesc numerele naturale n și k. Să se calculeze  $C_n^k$ , utilizând relația de recurență:

$$C_n^k = \begin{cases} 1 & \text{pentru } k = 0 \text{ sau } k = n \\ C_{n-1}^{k-1} + C_{n-1}^k & \text{pentru } k = 1, 2, \dots, n-1 \end{cases}$$

În variabilele n și k se depune rezultatul evaluării funcției **eval()**. Cele două valori sunt transmise funcției *combinări* ce va calcula recursiv numărul de combinări cerute.

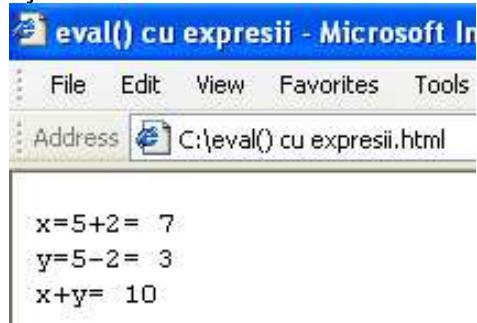
```
<html>
<head>
<title>Functia eval()</title>
</head>
<body>
<pre>
<script type="text/javascript">
var n=eval(prompt("Dati numerele naturale
\nn:"));
var k=eval(prompt("si k:"));
document.writeln("Combinarile
pentru
numerele date: ",combinari(n,k));
function combinari(n,k)
{
    if ((k==0) || (k==n))
        return 1;
    else
        return combinari(n-1,k-1)+combinari(n-
1,k);
}
</script>
</pre>
</body></html>
```



Combinarile pentru numerele date: 21

**Exemplu** de utilizare a expresiilor aritmetice. Se evaluează pe rând două expresii aritmetice, după care se combină rezultatele și se afișează rezultatul.

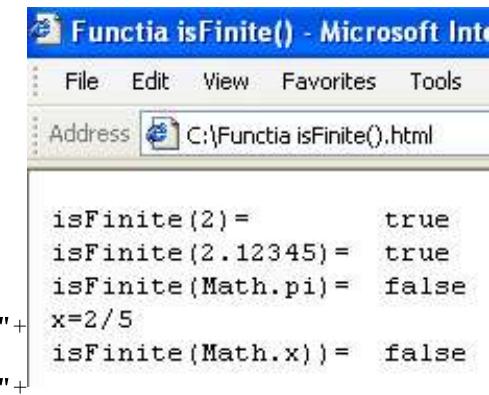
```
<html>
<head>
<title>eval() cu expresii</title>
</head>
<body>
<pre>
<script type="text/javascript">
    var x=eval("5+2");
    var y=eval("5-2");
    document.writeln("x=5+2= ",x);
    document.writeln("y=5-2= ",y);
    document.writeln("x+y= ", eval(x+y));
</script>
</pre>
</body></html>
```



**Functia isFinite()** evaluează un parametru pentru a determina dacă este număr finit sau nu. Dacă parametrul primit tinde către  $+\infty$  sau către  $-\infty$ , dacă este *Nan* sau nu are limite finite, funcția va returna false, altfel va returna true.

**Exemplu** Am utilizat pentru verificarea funcției *isFinite()*, atât numere ce au zecimale finite, cât și numere ale căror zecimale nu sunt într-un număr finit.

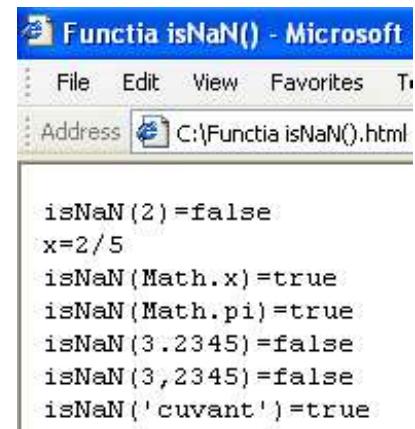
```
<html>
<head>
<title>
    Functia isFinite()
</title>
</head>
<body>
<pre>
<script type="text/JavaScript">
document.writeln("isFinite(2)="+isFinite(2));
document.writeln("isFinite(2.12345)=
    "+isFinite(2.12345));
document.writeln("isFinite(Math.pi)=
    "+isFinite(Math.pi));
var x=2/5;
document.writeln("x=2/5 <br />
    "+isFinite(Math.x));
</script>
</pre>
</body>
</html>
```



**Functia isNaN()** evaluează argumentul pentru a determina dacă este număr sau nu. În cazul în care argumentul este număr, returnează **false**, altfel returnează **true**.

**Exemplu** de utilizare a funcției *isNaN()*.

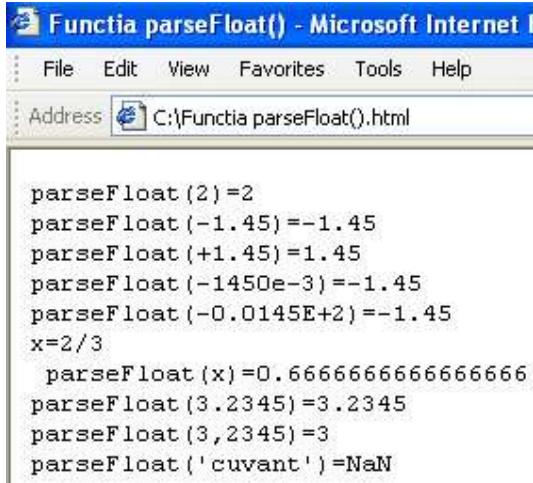
```
<html>
<head>
<title>
    Functia isNaN()
</title>
</head>
<body>
<pre>
<script type="text/JavaScript">
document.writeln("isNaN(2)="+isNaN(2));
document.writeln("x=2/5 <br />
    "+isNaN(Math.x)+"isNaN(Math.x));
document.writeln("isNaN(Math.pi)="+isNaN(Math.pi));
document.writeln("isNaN(3.2345)="+isNaN(3.2345));
document.writeln("isNaN(3,2345)="+isNaN(3,2345));
document.writeln("isNaN('cuvant')='"+isNaN('cuvant'))
    );
</script>
</pre>
</body>
</html>
```



**Functia parseFloat()** transformă un sir de caractere într-un număr de tip Float. Dacă primul caracter din sir nu este număr, funcția va returna *NaN* (adică Not a Number).

Sintaxa: **parseFloat** (*sir de caractere*)

**Exemplu** În exemplul următor am utilizat ca sir de caractere, numere pozitive și negative, numere cu zecimale și cu exponent. La cel de-al patrulea exemplu, am utilizat -1450e-3, unde e reprezentă 10 la puterea dată de numărul care urmează, respectiv -3. Deci -1450e-3 este  $-1450 \cdot 10^{-3}$  adică -1.45.



The screenshot shows a Microsoft Internet Explorer window with the title "Functia parseFloat() - Microsoft Internet Explorer". The address bar contains "C:\Functia parseFloat().html". The main content area displays the following output:

```
parseFloat(2)=2
parseFloat(-1.45)=-1.45
parseFloat(+1.45)=1.45
parseFloat(-1450e-3)=-1.45
parseFloat(-0.0145E+2)=-1.45
x=2/3
parseFloat(x)=0.6666666666666666
parseFloat(3.2345)=3.2345
parseFloat(3,2345)=3
parseFloat('cuvant')=NaN
```

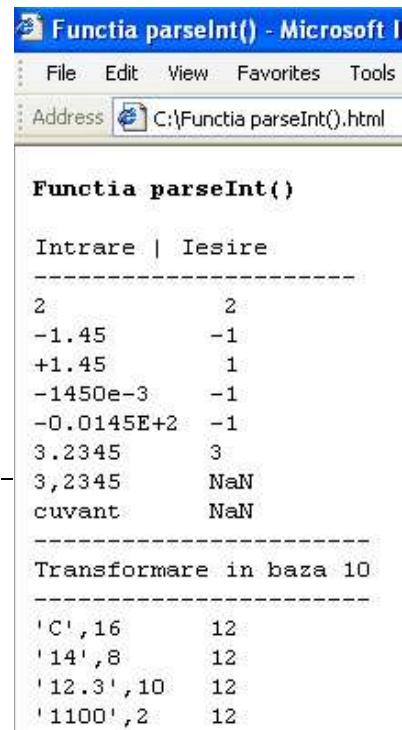
```
<html>
<head>
  <title>Functia parseFloat()</title>
</head>
<body>
<pre>
<script type="text/JavaScript">
  document.writeln("parseFloat(2)="+parseFloat(2));
  document.writeln("parseFloat(-1.45)="+parseFloat(-1.45));
  document.writeln("parseFloat(+1.45)="+parseFloat(+1.45));
  document.writeln("parseFloat(-1450e-3)="+parseFloat(-1450e-3));
  document.writeln("parseFloat(-0.0145E+2)="+parseFloat(-0.0145E+2));
  var x=2/3;
  document.writeln("x=2/3    <br /> parseFloat(x)="+parseFloat(x));
  document.writeln("parseFloat(3.2345)="+parseFloat(3.2345));
  document.writeln("parseFloat(3,2345)="+parseFloat(3,2345));
  document.writeln("parseFloat('cuvant')="+parseFloat('cuvant'));
</script>
</pre>
</body>
</html>
```

**Functia parseInt()** transformă un sir de caractere într-un număr de tip Int. Lângă sirul de caractere, poate fi specificată baza de numerație din care va fi transformat în baza 10. Cu ajutorul funcției *parseInt()*, putem transforma un număr din bazele de numerație 2, 8, 10, 16, în baza 10, dacă va fi specificată baza din care va fi transformat numărul. Dacă primul caracter din sir nu este număr, sau dacă nu va fi specificată corect baza de numerație, funcția va returna *NaN* (adică Not a Number).

Sintaxa: **parseInt** (*sir de caractere[,baza de numerație]*)

**Exemplu** În prima parte a exemplului de mai jos, am transformat diferite numere în numere întregi, iar în a doua parte am trasformat numere scrise în diverse baze de numerație, în baza 10.

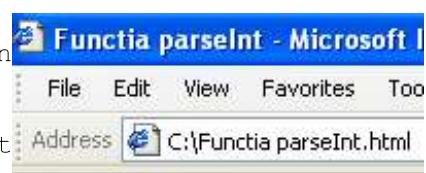
```
<html>
<head>
    <title>Functia parseInt()</title>
</head>
<body>
<pre>
<script type="text/JavaScript">
document.writeln("<b>Functia parseInt()</b>");
document.writeln("Intrare | Iesire");
document.writeln("-----");
document.writeln("2           "+parseInt(2));
document.writeln("-1.45      "+parseInt(-1.45));
document.writeln("+1.45      "+parseInt(+1.45));
document.writeln("-1450e-3   "+parseInt(-1450e-3));
document.writeln("-0.0145E+2  "+parseInt(-0.0145E+2));
document.writeln("3.2345      "+parseInt(3.2345));
document.writeln("3,2345      "+parseInt(3,2345));
document.writeln("cuvant      "+parseInt('cuvant'));
document.writeln("-----");
document.writeln("Transformare in baza 10");
document.writeln("-----");
document.writeln("'C',16      "+parseInt("C",16));
document.writeln("'14',8      "+parseInt("14",8));
document.writeln("'12.3',10   "+parseInt("12.3",10));
document.writeln("'1100',2     "+parseInt("1100",2));
</script></pre>
</body></html>
```



**Exemplu** În următorul exemplu transformăm un număr citit de la tastatură, din baza 2 în baza 10. Pentru aceasta, am transformat numărul cu **parseInt** și am verificat cu **isNaN**, dacă ceea ce a tastat utilizatorul este sau nu număr.

```
<html>
<head>
    <title>Functia parseInt</title>
<script language="JavaScript">
function t(x)
{
    y=parseInt(x,2);
    if (isNaN(y))
        document.writeln("Nu ati tastat un
                        numar in baza 2!");
    else
        document.writeln("Numarul      transformat
                        \nin baza 10 este:<b> ",y,"</b>");

}
</script></head>
<body><pre>
<script language="JavaScript">
x=eval(prompt("Dati un numar in baza
2:"));
t(x);
</script></pre>
</body></html>
```



**Functia Number()** transformă parametrul unui obiect specificat într-un număr, reprezentând valoarea obiectului respectiv. Dacă valoarea respectivă nu este un număr, va fi returnat *Nan*. Dacă obiectul specificat este *Date*, funcția va returna valoarea în milisecunde, măsurată din 1 ianuarie 1979 UTC (GMT), pozitivă după această dată și negativă înainte.

Sintaxa: **Number (obiect )**

**Exemplu** Funcția returnează zero dacă data specificată ca parametru, va fi cea din enunț, +2, pentru București. Dacă scriem ora 00:00:00, ne va returna o dată negativă, reprezentând cele două ore transformate în milisecunde, deoarece pentru București avem GMT+2.

```
<html>
<head><title>Functia Number()</title>
</head>
<body><pre>
<script language="JavaScript">
document.writeln("Functia Number()");
document.writeln("Intrare   |   Iesire")
document.writeln("-----");
document.writeln("-234.53      -234.53");
document.writeln("true        1");
document.writeln("false       0");
document.writeln("cuvant     NaN");
document.writeln("new Date(January 01, 1970 00:00:00) 0");
document.writeln("new Date(January 01, 1970 00:00:00) -7200000");
document.writeln("new Date(January 01, 1970 00:00:00) "+Number(new
Date("January 01,
1970 02:00:00")));
document.writeln("new Date(January 01, 1970 00:00:00) "+Number(new
Date("January 01,
1970 00:00:00")));
</script></pre>
</body></html>
```

Functia Number() - Microsoft Internet Explorer		
File	Edit	
View Favorites Tools Help		
Address	C:\Functia Number().html	
Functia Number()		
Intrare		Iesire
-----		
-234.53		-234.53
true		1
false		0
cuvant		NaN
new Date(January 01, 1970 00:00:00)	0	
new Date(January 01, 1970 00:00:00)	-7200000	
new Date(January 01, 1970 00:00:00)	"+Number(new	
Date("January 01,		
1970 02:00:00"));		
document.writeln("new		
Date(January		
01, 1970 00:00:00) "+Number(new		
Date("January 01,		
1970 02:00:00")));		
document.writeln("new		
Date(January		
01, 1970 00:00:00) "+Number(new		
Date("January 01,		
1970 00:00:00")));		
</script></pre>		
</body></html>		

**Functia String()** transformă un obiect specificat într-un String (șir de caractere). Dacă obiectul este *Date*, funcția va returna un șir de caractere ce reprezintă data calendaristică, după cum putem vedea în exemplul de mai jos.

Sintaxa: **String (obiect )**

**Exemplu** Funcția transformă parametrul introdus, în șir de caractere.

Functia String() - Microsoft Internet Explorer		
File	Edit	
View Favorites Tools Help		
Address	C:\Functia String().html	
Functia String()		
Intrare		Iesire
-----		
-234.53		-234.53
true		true
cuvant oarecare		cuvant
0		0
new Date(1234567890)	Thu Jan 15 08:56:07 UTC+0200 1970	

```

<html>
<head>
    <title>Functia String()</title>
</head>
<body>
<pre>
<script language="JavaScript">
    document.writeln("Functia String()");
    document.writeln("Intrare           | Iesire");
    document.writeln("-----");
    document.writeln("-234.53          "+String(-234.53));
    document.writeln("true            "+String(true));
    document.writeln("cuvant oarecare "+String("cuvant"));
    document.writeln("0              "+String(0));
    document.writeln("new Date(1234567890) "+String(new Date(1234567890)));
</script>
</pre>
</body>
</html>

```

## 6.6. Funcții și stiluri

Nu este suficient să știm să utilizăm funcții și stiluri. Important este ca tot ceea ce știm să putem utiliza pentru a realiza diverse aplicații mai complexe. Tocmai acest lucru voi încerca să realizez în exemplul de mai jos.

### Stiluri pentru câmpul textarea

În acest exemplu am introdus un text într-un câmp de tip textarea pe care îl pot afișa în trei moduri, utilizând stiluri diferite. Pentru aceasta, am definit cele trei stiluri s1, s2 și s3, în antetul paginii Web. Stilurile sunt apelate prin intermediul funcțiilor: f2, f3 și f4, la declanșarea evenimentului onClick prin apăsarea butoanelor: Stil 1, Stil 2 și Stil 3. Funcțiile f2, f3 și f4 utilizează masivul a, care pentru prima valoare (t=0), va utiliza stilul s1, pentru a doua valoare (t=1), va utiliza stilul s2 și pentru a treia valoare (t=2), va utiliza stilul s3.

În concluzie, scriem un text în câmpul textarea, apăsăm butonul pe care scrie **Apasati!**, alegem unul din stiluri prin apăsarea butoanelor: Stil 1, Stil 2 sau Stil 3, după care apăsăm butonul pe care scrie **click**. Implicit, textul afișat este cel scris în body și anume o strofă dintr-o poezie scrisă de Ienăchiță Văcărescu.

```

<html>
<head>
    <title>functii si stiluri</title>
<style>
.s1
{
    font-family: Arial, Helvetica, sans-serif;
    font-size: 18px;
    color: #006633;
    background-color: #91FFC8;
}
.s2
{
    font-family: Georgia, Times New Roman, Times, serif;
    font-size: 16px;
    color: #990000;
    background-color: #FF8282;
}
.s3
{
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 18px;
    color: #3333FF;
}

```

```

        background-color: #9B9BFF;
    }
</style>
<script>
function f1()
{
    var reg=/\n/g;
    var getElementById=document.formular.ta.value;
    y=getElementById.replace(reg,"<BR>");
    document.getElementById("target").innerHTML=y;
}
var t=0;
var a=new Array("s1","s2","s3");
function f2()
{
    t=0;
    document.getElementById("abc").className=a[t];
}
function f3()
{
    t=1;
    document.getElementById("abc").className=a[t];
}
function f4()
{
    t=2;
    document.getElementById("abc").className=a[t];
}
function f()
{
    document.getElementById("target").className=document.
        getElementById("abc").className;
}
</script>
</head>
<body>
    <table>
        <tr>
            <td>
                <div id=target class=s2>Scrieti un text si dati click!
                </div>
            </td>
        </tr>
    </table>
<form name=formular>
    <textarea name=ta class=s1 id=abc>
        Spune, inimioar&#259;

        Spune, inimioar&#259;, spune
        Ce dureaza r&#259;pune?
        Arata&#259; ce te muncui&#351;te,
        Ce boala&#259; te chinuie&#351;te?

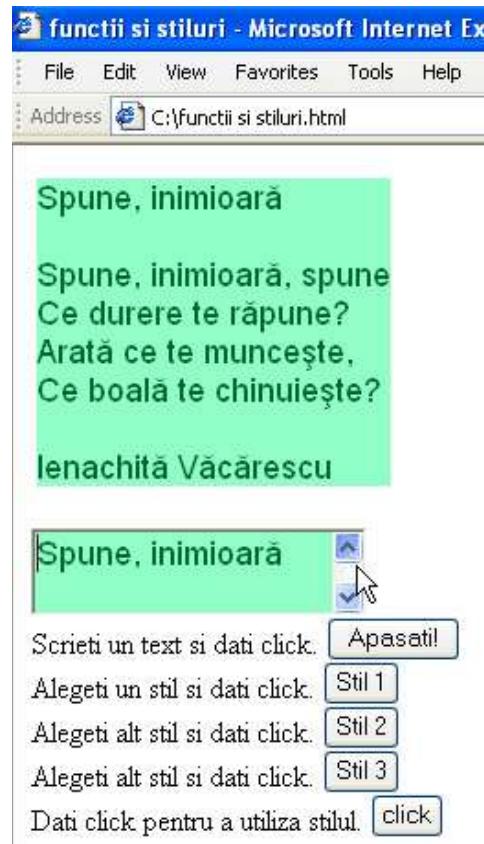
        Ienachit&#259; V&#259;c&#259;rescu
    </textarea>
    <br />
    Scrieti un text si dati click.
    <input type=button value="Apasati!" onclick="f1()"><br />
    Alegeti un stil si dati click.

```

```

<input type=button value="Stil 1" onclick="f2()"><br />
Alegeti alt stil si dati click.
<input type=button value="Stil 2" onclick="f3()"><br />
Alegeti alt stil si dati click.
<input type=button value="Stil 3" onclick="f4()"><br />
Dati click pentru a utiliza stilul.
<input type=button value="click" onclick="f()"><br />
</form>
</body>
</html>

```



### Stil diferit pentru o pagină

Atunci când vizităm o pagină, de cele mai multe ori avem impresia că are importanță numai conținutul, nu și aspectul paginii. Mai jos, am să încerc să demonstreze faptul că puțină atenție acordată aspectului, nu strică niciodată. Deci, efectuând click pe legătura din partea de jos a paginii, se va apela funcția f() care va da atribute diferite stilurilor utilizate. Tot funcția f() ne va atenționa în cazul în care utilizăm alt browser, în afară de Internet Explorer, în care se poate folosi acest exemplu.

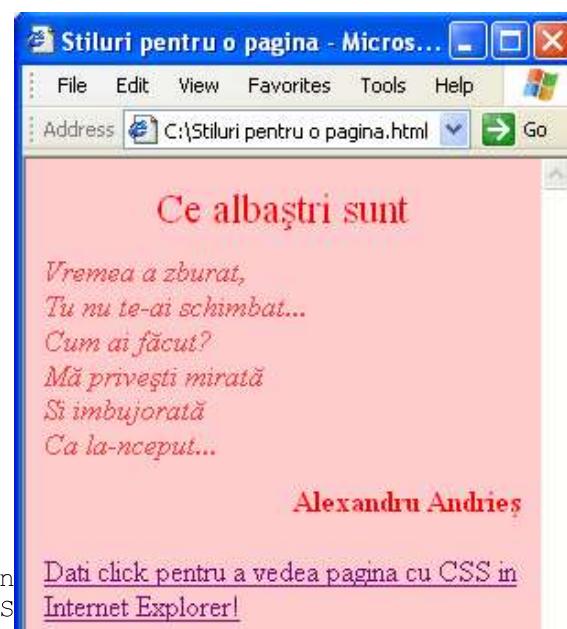
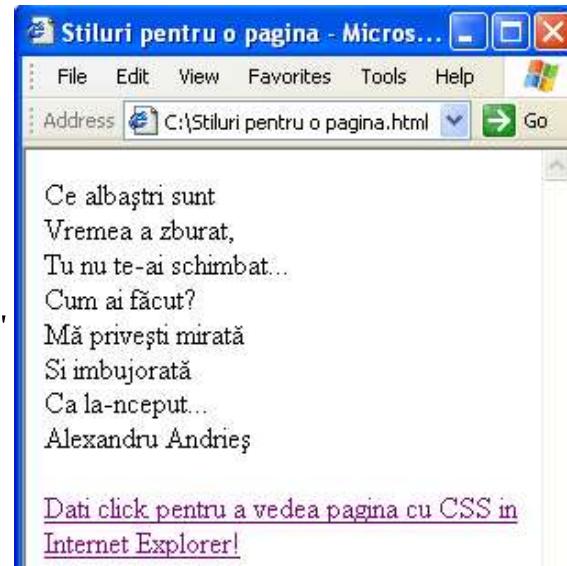
```

<html>
<head>
<title>Stiluri pentru o pagina</title>
<script type="text/javascript">
function f()
{
    if(!document.all)
        { alert("Nu folositi Internet Explorer!")
          return
        }
    document.body.style.backgroundColor="#FFCCCC"
    document.all.a1.style.color="red"
    document.all.a1.style.fontSize="22px"
    document.all.a1.style.textAlign="center"

    document.all.a2.style.color="#FF3C3C"
    document.all.a2.style.fontStyle="oblique"
    document.all.a2.style.textAlign="left"
    document.all.a2.style.marginTop="12px"

    document.all.a3.style.color="red"
    document.all.a3.style.fontWeight="bold"
    document.all.a3.style.fontStyle="normal"
    document.all.a3.style.textAlign="right"
    document.all.a3.style.marginTop="12px"
}
</script>
</head>
<body>
<div id=a1>Ce alba&#351;tri sunt</div>
<div id=a2>
Vremea a zburat,<br />
Tu nu te-ai schimbat...<br />
Cum ai f&#259;cut?<br />
M&#259; prive&#351;ti mirat&#259;<br />
Si imbujorat&#259;<br />
Ca la-nceput...<br /></div>
<div id=a3>Alexandru Andrie&#351;</div>
<br><a href="#" onClick="f();return false">Dati click pentru a vedea pagina cu CSS in Internet Explorer!</a>
</body>
</html>

```



## Butonul alert decorat

Cu ajutorul funcțiilor și a stilurilor, putem decora butonul alert, după cum dorim noi. Am utilizat stiluri pentru toate elementele butonului alert: fundal, margini, imagine pentru fundal, poziția butonului de inchidere, distanța dintre litere, distanță pentru margini.

```

<html>
<head>
<title>Butonul alert</title>
<style type="text/css">
#dreptunghi
{ position:absolute;
  width:auto;
  height:auto;
  top:0px;
  left:0px;
  z-index:10000;  }

```

```

#drA
{
    position: relative;
    width: 300px;
    min-height: 100px;
    margin-top: 50px;
    border: 2px solid #FFFF00;
    background-repeat: no-repeat;
    background-position: 20px 30px;
    visibility: hidden;
    background-image: url(1trandafiri_albi.jpg);
    background-color: #FFFFCC; }

#dreptunghi > #drA
{
    position: fixed; }

#drA h1
{
    margin: 0;
    font: bold 0.9em verdana, arial;
    background-color: #FFFF6C;
    color: #000000;
    padding: 2px 0 2px 5px;
    text-align: center;
    border-top-width: thin;
    border-right-width: thin;
    border-bottom-width: thin;
    border-left-width: thin;
    border-top-style: none;
    border-right-style: none;
    border-bottom-style: none;
    border-left-style: none; }

#drA p
{
    font: 0.7em verdana, arial;
    height: 50px;
    padding-left: 5px;
    margin-left: 55px; }

#drA #Inchide_Buton
{
    display: block;
    position: relative;
    padding: 3px;
    border: 2px solid #FFFF6C;
    width: 100px;
    text-transform: none;
    text-align: center;
    color: #000000;
    background-color: #FFFF6C;
    font-family: Geneva, Arial, Helvetica, sans-serif;
    font-size: 14px;
    margin-top: 5px;
    margin-right: auto;
    margin-bottom: 5px;
    margin-left: auto;
    text-decoration: none;
    font-weight: bold;
    letter-spacing: 0.3em;
    left: 100px;
    bottom: 20px; }

h1
{
    margin: 0;
    padding: 4px;
    font: bold 1.5em verdana;
    border-bottom: 1px solid #000; }

.important
{
    background-color: #F5FCC8;
}

```

```

        padding:2px;    }
</style>
<script type="text/javascript">
var titlul_alert = "Mesajul transmis";
var text_button = "Inchideti!!";
if(document.getElementById)
{
  window.alert = function(x)
  { f(x);  }
}
function f(x)
{ d = document;
  if(d.getElementById("dreptunghi"))
    return;
  obiect      =      d.getElementsByTagName("body") [0].appendChild
(d.createElement("div"));
  obiect.id = "dreptunghi";
  obiect.style.height = d.documentElement.scrollHeight + "px";
  obiectA = obiect.appendChild(d.createElement("div"));
  obiectA.id = "dra";
  if(d.all      && !window.opera)      obiectA.style.top      =
document.documentElement.scrollTop + "px";
  obiectA.style.left      =      (d.documentElement.scrollWidth -
obiectA.offsetWidth)/2 + "px";
  obiectA.style.visibility="visible";
  h1 = obiectA.appendChild(d.createElement("h1"));
  h1.appendChild(d.createTextNode(titlul_alert));
  msg = obiectA.appendChild(d.createElement("p"));
  msg.appendChild(d.createTextNode(x));
  Buton = obiectA.appendChild(d.createElement("a"));
  Buton.id = "Inchide_Buton";
  Buton.appendChild(d.createTextNode(text_button));
  Buton.href = "#";
  Buton.onclick = function()
  {
    ra();
    return false;
  }
}
function ra()
{document.getElementsByTagName("body") [0].removeChild(document.
  getElementById ("dreptunghi"));
}
</script>
</head>
<body>
<p align="center">
<input type="button" value ="Dati click pentru a vedea mesajul"
onClick="alert('Mesajul pe care l-ati dorit :');" /></p>
</body>
</html>
```

Dati click pentru a vedea mesajul



## Evaluare

10. Cum se definește o funcție?
11. Cum se apelează o funcție?
12. Ce funcție transformă un sir de caractere într-un număr de tip întreg?
13. Scrieți un script ce permite citirea conținutului a două variabile și o funcție ce calculează elementele minim și maxim dintre cele două numere citite.
14. Cum apelăm funcția cu numele f, de parametrii x și y?
  - a) f(x,y)
  - b) f parametri x, y
  - c) function f(x,y)
  - d) f(x and y)
15. Cum creăm o funcție cu numele f de parametri x și y?
  - a) function:x,y
  - b) function=f(x, y)
  - c) function f(var x, var y)
  - d) function f(x, y)