Introducere în JavaScript

Ce este JavaScript?

- JavaScript a fost proiectat pentru a adăuga interactivitate paginilor HTML
- JavaScript este un limbaj pentru scripturi. Un limbaj pentru scripturi este un limbaj de programare simplificat
- JavaScript este, în general, înglobat direct în paginile HTML
- JavaScript este un limbaj interpretat (adică scriptul este executat direct, fără compilare prealabilă)
- JavaScript poate fi folosit fără licenţă

Ce poate face JavaScript?

- **JavaScript oferă proiectanţilor HTML un instrument de programare –** în general proiectanţii paginilor HTML nu au cunostinte mari legate de programare, dar JavaScript este un limbaj cu o sintaxă foarte simplă şi multi utilizataori pot insera mici secvenţe de cod în paginile HTML
- **JavaScript poate insera în mod dinamic text într-o pagină HTML –** O instrucţiune JavaScript ca aceasta: **document.write("<h1>" + name + "</h1>")** poate scrie un text variabil în pagina HTML
- **JavaScript poate reacţiona la evenimente –** Un cod JavaScript poate fi proiectat să se execute când se întâmplă ceva, spre exemplu când pagina s-a încărcat complet sau utilizatorul acţionează un element HTML
- **JavaScript poate citi şi scrie elementele HTML –** Un cod JavaScript poate citi şi modifica conţinutul unui element HTML
- **JavaScript poate fi folosit pentru a valida datele –** Un cod JavaScript poate fi folosit pentru a valida datele înainte de a fi trimise către server. În acest fel serverul nu mai face procesări suplimentare.
- **JavaScript poate fi folosit pentru a detecta browserul utilizatorului** – Un cod JavaScript poate detecta tipul browserului şi poate încărca o pagină proiectată special pentru tipul respectiv de browser
- **JavaScript poate fi folosit pentru a crea cookies** – Un cod JavaScript poate fi utilizat pentru a stoca şi extrage informaţii pe calculatorul vizitatorului paginii HTML

Cum se inserează JavaScript într-o pagină HTML

Pentru a insera JavaScript într-o pagină HTML se utilizează tagul **<script>**.
În exemplul următor, JavaScript este utilizat pentru a scrie un text într-o pagină web:

```
<html>
<body>
<h3>Afisarea unui mesaj cu JavaScript</h3> <hr/>
<script type="text/javascript">
document.write("Bine ati venit!");
</script>
</body>
</html>
```

Exemplul următor ilustrează cum pot fi adăugate taguri HTML pentru a formata textul afişat cu JavaScript:

```
<html>
<body>
<h3>Utilizarea tagurilor HTML in mesajul afisat cu
JS</h3> <hr/>
<script type="text/javascript">
document.write("<h1>Bine ati venit!</h1>");
</script>
</body>
</html>
```

Explicaţii:
Pentru a insera JavaScript într-o pagină HTML, folosim tagul **<script>** şi în interiorul acestui tag folosim atributul type pentru a defini limbajul în care este scris scriptul. Astfel, tagurile **<script type="text/javascript">** şi **</script>** marchează locul în care începe, respectiv se sfârşeşte scriptul:

```
<html>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

Comanda **document.write** reprezintă modalitatea JS standard pentru a scrie un text într-o pagină. Deoarece această comandă este inclusă între tagurile **<script>** şi **</script>**, browserul o va recunoaşte drept comandă JS şi va executa respectiva linie de cod. Pentru exemplul considerat, browserul va scrie în pagină textul **Bine ati venit!**

Obs: Dacă comanda **document.write** nu este inclusă între tagurile de script, browserul o va interpreta ca text obişnuit şi va afişa pe ecran linia de cod.

Inserarea scripturilor JS

Dacă scriptul este inclus în secţiunea **body**, el va fi executat cât timp se încarcă pagina. Dacă scriptul este inclus în secţiunea **head**, el va fi executat numai când este apelat.

Scripturi în <head>
Scripturile care trebuie executate când sunt apelate sau când are loc un eveniment, trebuie scrise în secţiunea **head**. În acest fel, scriptul va fi sigur încărcat înainte de a fi utilizat.

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("Aceasta caseta de alertare este apelata si afisata cand are loc evenimentul onload");
}
</script>
</head>
<body onload="message()">
<h3>Casetele de alertare</h3> <hr/>
</body>
</html>
```

Scripturi în <body>
Scripturile care trebuie executate când pagina se încarcă trebuie scrise în secţiunea body şi vor genera conţinutul paginii:

```
<html>
<head>
</head>
<body>
<h3>Afisarea textului cu JavaScript</h3> <hr/>
<script type="text/javascript">
document.write("Acest mesaj este scris cu JavaScript");
</script>
</body>
</html>
```

Scripturi în <head> şi <body>
Puteţi include un număr nelimitat de scripturi JS în document, deci puteţi avea scripturi şi în **head** şi în **body**:

```
<html>
<head>
<script type="text/javascript">
....
```

</script>
</head>
<body>
<script type="text/javascript">
....
</script>
</body>

Folosirea unui script extern
Dacă doriţi să utilizaţi acelaşi script în mai multe pagini web fără a rescrie codul, trebuie să scrieţi scriptul JS într-un fişier extern. Fişierul trebuie să aibă extensia **.js** şi nu poate conţine tagul **<script>**. Pentru a utiliza fişierul extern, trebuie sa il includeţi în atributul **src** al tagului **<script>**:

<html>
<head>
<script type="text/javascript" src="…….js"></script>
</head>
<body>
</body>
</html>

Obs: Scriptul trebuie plasat în locul în care ar fi fost scris în mod normal.

Instrucţiuni JavaScript

JavaScript este o secvenţă de declaraţii, instrucţiuni şi comenzi care vor fi executate de către browser. Spre deosebire de HTML, Java Script este casesensitive, deci aveţi grijă când scrieţi instrucţiuni, declaraţi variabile sau apelaţi funcţii. O instrucţiune JavaScript este o comandă către browser şi are rolul de a spune browserului ce trebuie să facă. Următoarea instrucţiune JS transmite browserului să scrie în pagină textul "Buna ziua":

document.write("Buna ziua");

Fiecare instrucţiune se încheie cu punct şi virgulă (;).

Codul JavaScript este o secvenţă de instrucţiuni JS. Fiecare instrucţiune este executată de browser în ordinea în care a fost scrisă.

Exemplul următor va scrie un titlu şi două paragrafe într-o pagină web:

```
<html>
<body>
<h3>Utilizarea tagurilor HTML in mesajele afisate cu JS</h3> <hr/>
```

```
<script type="text/javascript">
document.write("<h1>Acesta este un titlu</h1>");
document.write("<p>Acesta este un paragraf.</p>");
document.write("<p>Acesta este un alt paragraf.</p>");
</script>
</body>
</html>
```

Blocuri JavaScript

Instrucţiunile JavaScript pot fi grupate în blocuri care se scriu între acolade.

Instrucţiunile dintr-un bloc vor fi executate împreună.

În acest exemplu, instrucţiunile care scriu un titlu şi două paragrafe, au fost grupate împreună într-un bloc:

```
<html>
<body>
<script type="text/javascript">
{
document.write("<h1>Acesta este un titlu</h1>");
document.write("<p>Acesta este un paragraf.</p>");
document.write("<p>Acesta este un alt paragraf.</p>");
}
</script>
</body>
</html>
```

În mod normal, un bloc este folosit pentru a grupa un grup de instrucţiuni într-o funcţie sau într-o condiţie (blocul va fi executat dacă o anumită condiţie este satisfăcută).

Variabile JavaScript

În JS, variabilele sunt folosite pentru a păstra valori sau expresii. O variabilă poate avea un nume scurt, de exemplu x, sau mai descriptiv, de exemplu prenume.

Reguli pentru numele variabilelor JavaScript:

- numele este case-sensitive (y şi Y sunt două variabile diferite)
- numele trebuie să înceapă cu o literă sau cu liniuţa de subliniere (underscore)

Exemplu

Valoarea unei variabile se poate modifica în timpul execuţiei scriptului. Puteţi referi variabila prin nume pentru a-i afişa sau modifica conţinutul, ca în exemplul următor:

```
<html>
<body>
<h3>Declararea, initializarea, atribuirea si afisarea unei variabile</h3> <hr/>
<script type="text/javascript">
var prenume;
prenume="Mihai";
document.write("<b>Numele variabilei</b>: prenume");
document.write("<br/>");
document.write("<b>Valoare initiala</b>: "+prenume);
document.write("<br/>");
prenume="Adrian";
document.write("<b>Valoare dupa atribuire</b>: "+prenume);
</script>
</body>
</html>
```

Declararea variabilelor JavaScript

Puteţi crea variabile cu sintaxa:

var *nume_variabila;*

După declarare, variabila nu conţine valori (este vidă). Puteţi să iniţializaţi o variabilă chiar în momentul declarării:

var x=8;

var prenume="Matei";

Obs: Când atribuiţi unei variabile o valoare de tip text, textul trebuie scris între ghilimele.

Dacă atribuiţi valori unei variabile care nu a fost încă declarată, ea va fi declarată automat.

Declaraţiile:

x=8;

prenume="Matei";

au acelaşi efect cu declaraţiile:

var x=8;

var prenume="Matei";

Dacă redeclaraţi o variabilă JavaScript, ea va păstra valoarea iniţială:

var x=7;

var x;

După execuţia instrucţiunilor de mai sus, variabila x are valoarea 7 care nu a fost resetată la redeclarare.

Operatorii JavaScript

Operatorii aritmetici

Sunt folosiţi pentru a efectua operaţii aritmetice cu variabile şi/sau valori.

+ adunare

- scădere

* înmulţire

/ împărţire

% modulo (restul împărţirii întregi)

++ incrementare

-- decrementare

Operatorii de atribuire

Sunt folosiţi pentru a atribui valori variabilelor JavaScript.

=, += , -=, *= , /= , %=

Operatorul + utilizat pentru şiruri de caractere

Acest operator poate fi utilizat şi pentru a concatena variabile tip şir de caractere (string sau text).

Exemplu:

t1="Ce mai";

t2="faci azi?";

t3=t1+t2;

După execuţie, variabila t3 conţine şirul „Ce maifaci azi?”.

Adunarea şirurilor şi a numerelor

Regulă: Dacă se aduna un număr cu un şir de caractere, se va obţine un şir de caractere.

```
<html>
<body>
<h3>Adunarea sirurilor de caractere si a numerelor cu siruri de caractere</h3> <hr/>
<script type="text/javascript">
x=6+7;
document.write("6+7="+x);
document.write("<br />");
x="6"+"7";
document.write('"6"+"7"='+x);
document.write("<br />");
x=6+"7";
document.write('6+"7"='+x);
document.write("<br />");
x="6"+7;
document.write('"6"+7='+x);
document.write("<br />");
</script>
</body>
</html>
```

Operatorii de comparare şi operatorii logici

Operatorii de comparare sunt utilizaţi în construcţii logice pentru a verifica egalitatea sau diferenţa dintre două variabile sau valori.

== egal

=== este egal exact (valoare şi tip). Exemplu: x==="5" este fals

!= Diferit

> Mai mare decat

< Mai mic decât

>= Mai mare sau egal

<= Mai mic sau egal

Operatorii logici sunt utilizaţi pentru a determina relaţia logică dintre variabile sau valori.

&& şi (x < 10 && y > 1) este adevărat

|| sau (x==5 || y==5) este fals

! not !(x==y) este adevărat

Operatorul condiţional

Acest operator atribuie o valoare unei variabile în funcţie de o anumită condiţie.

Sintaxă:

variabila=(conditie)?valoare1:valoare2

Exemplu:

salut=(visitor=="FEM")?"Doamna ":"Domnul";

Instrucţiunile condiţionale

Adesea, când scrieţi cod JS, trebuie să realizaţi operaţii diferite în funcţie de decizii diferite. Pentru a realiza acest lucru, folosiţi în cod instrucţiunile condiţionale.

În JavaScript există următoarele instrucţiuni condiţionale:

if, if...else, switch

Exemplul 1:

```
<html>
<body>
<h3>Scriptul va afisa un mesaj daca ora<10 folosind instructiunea if</h3> <hr/>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time < 10)
{
document.write("<b>Buna dimineata</b>");
}
</script>
</body>
```

```
</html>
```

Exemplul 2:

```
<html>
<body>
<h3>Scriptul va afisa un mesaj sau altul in functie de ora, cu instructiunea if..else</h3> <hr/>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time < 10)
{
document.write("<b>Buna dimineata</b>");
}
else
{
document.write("<b>Buna ziua</b>");
}
</script>
</body>
</html>
</html>
```

Exemplul 3:

```
<html>
<body>
<h3>Scriptul va afisa unul din trei mesaje in functie de ora, cu instructiunea if-else-if-else</h3> <hr/>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time<10)
{
document.write("<b>Buna dimineata</b>");
}
else if (time>=10 && time<17)
{
document.write("<b>Buna ziua</b>");
}
else
{
document.write("<b>Buna seara</b>");
}
</script>
</body>
</html>
```

Exemplul 4:

Link-ul din exemplul următor va deschide Google sau Yahoo.

```
<html>
<body>
<h3>Scriptul afiseaza in mod aleator unul din doua
link-uri, folosind if..else</h3> <hr/>
<script type="text/javascript">
var r=Math.random();
if (r>0.5)
{
document.write("<a href='http://www.google.com'>Google!</a>");
}
else
{
document.write("<a href='http://www.yahoo.com’>Yahoo!</a>");
}
</script>
</body>
</html>
```

Exemplul 5:

```
<html>
<body>
<h3>Scriptul utilizeaza instructiunea switch</h3>
<hr/>
<script type="text/javascript">
var d = new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
document.write("<b>Vineri</b>");
break;
case 6:
document.write("<b>Sambata</b>");
break;
case 0:
document.write("<b>Duminica</b>");
break;
default:
document.write("<b>Astept weekend-ul!</b>");
}
</script>
</body>
```

```
</html>
```

Casete popup

JavaScript are trei tipuri de casete popup: caseta Alert, caseta Confirm şi caseta Prompt.

Caseta Alert

O casetă de alertă se utilizează atunci doriţi să fiţi siguri că o anumită informaţie ajunge în atenţia utilizatorului. Când o casetă de alertă este afişată, utilizatorul va trebui să acţioneze butonul "OK" pentru a putea continua.

Sintaxă:

alert("...un text....");

Exemplu:

```
<html>
<head>
<script type="text/javascript">
function afiseaza_alert()
{
alert("Sunt o caseta de alertare!");
}
</script>
</head>
<body>
<h3>La apasarea butonului va fi apelata o functie care afiseaza caseta alert</h3> <hr/>
<input type="button" onclick="afiseaza_alert()" value="Apasa" />
</body>
</html>
```

Caseta Confirm

O casetă de confirmare se utilizează atunci când doriţi ca utilizatorul să verifice sau să accepte ceva. Când caseta de confirmare este afişată, utilizatorul va trebui să acţioneze butonul "OK" sau butonul "Cancel" pentru a putea continua. Dacă utilizatorul acţionează butonul "OK", caseta returnează valoarea true, dacă acţionează butonul "Cancel", caseta returnează valoarea false.

Sintaxă:

confirm("....un text....");

Exemplu:

```
<html>
<head>
<script type="text/javascript">
function afiseaza_confirm()
{
var r=confirm("Apasati un buton");
if (r==true)
{
document.write("Ati apasat butonul OK!");
}
else
{
document.write("Ati apasat butonul Cancel!");
}
}
</script>
</head>
<body>
<h3>La apasarea butonului va fi apelata o functie care afiseaza caseta confirm si verifica ce buton ati apasat</h3> <hr/>
<input type="button" onclick="afiseaza_confirm()"
value="Apasa" />
</body>
</html>
```

Caseta Prompt

Această casetă se utilizează atunci când doriţi ca utilizatorul să introducă o anumită valoare înainte de a accesa pagina. Când caseta prompt este afişată, utilizatorul va trebui să acţioneze butonul "OK" sau butonul "Cancel" pentru a putea continua după ce introduce valoarea solicitată. Dacă utilizatorul acţionează butonul "OK", caseta returnează valoarea true, dacă acţionează butonul "Cancel", caseta returnează valoarea false.

Sintaxă:

prompt("....un text....","valoare_implicita");

Exemplu:

```
<html>
<head>
<script type="text/javascript">
function afiseaza_prompt()
{
var name=prompt("Va rog sa va introduceti numele","");
if (name!=null && name!="")
{
document.write("Buna ziua " + name + "! Ce mai
faci?");
}
}
</script>
</head>
<body>
<h3>La apasarea butonului va fi apelata o functie care afiseaza caseta prompt</h3> <hr/>
<input type="button" onclick="afiseaza_prompt()"
value="Apasa" />
</body>
</html>
```

Obs. Dacă doriţi ca textul dintr-o casetă să fie afişat pe mai multe linii, trebuie procedat ca în exemplul următor:

```
<html>
<head>
<script type="text/javascript">
function afiseaza_alert()
{
alert("Buna! Asa se adauga" + '\n' + "o intrerupere de linie" + '\n' + "intr-o caseta de alertare!");
}
</script>
</head>
<body>
<h3>Caseta alert cu textul scris pe mai multe linii</h3> <hr/>
<input type="button" onclick="afiseaza_alert()"
value="Apasa" />
</body>
</html>
```

Funcţii

O funcţie va fi executată când are loc un eveniment sau când este apelată. Dacă doriţi ca browserul să nu execute un script atunci când pagina se încarcă, puteţi scrie scriptul într-o

funcţie. O funcţie poate fi apelată din orice punct al paginii, sau chiar din alte pagini, dacă funcţia este scris într-un fişier JS extern.

Funcţiile JS pot fi scrise în secţiunea <head> sau în secţiunea <body> a documentului. Totuşi, pentru a fi siguri că funcţia este încărcată în browser înainte de a fi apelată, este recomandat să o scrieţi în secţiunea <head>.

Definirea unei funcţii

Sintaxă:

function *nume_functie*(*var1,var2,...,varX*)

{

codul functiei

}

Parametrii var1, var2, etc. sunt variabile sau valori transmise funcţiei. Acoladele marchează începutul şi sfârşitul corpului funcţiei.

Exemplu:

```
<html>
<head>
<script type="text/javascript">
function afiseaza_mesaj()
{
alert("Bine ati venit!");
}
</script>
</head>
<body>
<h3>La apasarea butonului este apelata o functie JS care afiseaza caseta alert</h3> <hr/>
<form>
<input type="button" value="Apasati!"
onclick="afiseaza_mesaj()" />
</form>
</body>
</html>
```

Dacă linia de cod alert("Bine ati venit!") din exemplul anterior nu ar fi fost scrisă în corpul unei funcţii, codul ar fi fost executat imediat ce linia respectivă ar fi fost încărcată în browser.

Deoarece codul a fost inclus într-o funcţie, el nu va fi executat decât atunci când utilizatorul acţionează butonul şi este apelată funcţia afiseaza_mesaj().

Instrucţiunea return

Instrucţiunea return este folosită pentru a specifica valoarea returnată de o funcţie şi trebuie inclusă în orice funcţie care returnează o valoare.

În exemplul următor, funcţia suma returnează suma celor doi parametri de intrare:

```
<html>
<head>
<script type="text/javascript">
function suma(a,b)
{
return a+b;
}
</script>
</head>
<body>
<h3>Suma urmatoare este calculata si returnata de o functie</h3> <hr/>
<script type="text/javascript">
document.write("7+9="+suma(7,9));
</script>
</body>
</html>
```

Durata de viaţă a variabilelor JavaScript

Dacă declaraţi o variabilă în interiorul unei funcţii, ea poate fi accesată numai din interiorul funcţiei. Când funcţia se încheie, variabila este distrusă.

Variabilele declarate în corpul unei funcţii se numesc variabile locale. Puteţi avea variabile locale cu acelaşi nume în funcţii diferite, deoarece fiecare variabilă locală este recunoscută numai în interiorul funcţiei în care este declarată. Dacă declaraţi o variabilă în afara tuturor funcţiilor (variabilă globală), ea poate fi accesată de toate funcţiile din pagină. O variabilă global este distrusă numai atunci când pagina este închisă.

Exemplul 1

Ilustrează cum se poate transmite o variabilă unei funcţii şi cum poate fi folosită respectiva variabilă în corpul funcţiei.

```
<html>
```

```
<head>
<script type="text/javascript">
function functia_1(text)
{
alert(text);
}
</script>
</head>
<body>
<h3>Functii JavaScript</h3> <hr/>
<form>
<input type="button" onclick="functia_1('Bune ati venit!')" value="Apasati">
</form>
<p>Cand apasati butonul, va fi apelata o functie cu textul "Bine ati venit!" drept parametru. Functia va
afisa parametrul cu o caseta de alertare.</p>
</body>
</html>
```

Exemplul 2

Ilustrează cum poate fi folosit rezultatul returnat de o funcţie.

```
<html>
<head>
<script type="text/javascript">
function functie_2()
{
return ("Bine ati venit!");
}
</script>
</head>
<body>
<h3>Textul urmator este returnat de o functie apelata direct din document.write()</h3> <hr/>
<script type="text/javascript">
document.write(functie_2())
</script>
</body>
</html>
```

Exemplul 3

```
<html>
<head>
<script type="text/javascript">
function salut(txt)
{
```

```
alert(txt);
}
</script>
</head>
<body>
<h3>Utilizarea functiilor JavaScript</h3> <hr/>
<form>
<input type="button"
onclick="salut('Buna dimineata!')"
value="Dimineata">
<input type="button"
onclick="salut('Buna seara!')"
value="Seara">
</form>
<p>
Cand apasati unul dintre butoane, va fi apelata o functie care afiseaza mesajul primit ca
parametru.</p>
</body>
</html>
```

Instrucţiunea for

Instrucţiunile repetitive sunt utilizate pentru a executa o secvenţă de cod în mod repetat. În JS sunt două tipuri diferite de instrucţiuni repetitive:

for

while si *do ... while*

Sintaxă:

for(var=val_initiala;var<=val_finala;var=var+increment)

{

codul ce trebuie executat

}

Instrucţiunea while

Sintaxă:

while (*propozitielogica*)

{

codul ce trebuie executat

}

Instrucţiunea do...while

Este o variantă a instrucţiunii while. Secvenţa de instrucţiuni va fi executată în mod sigur o dată, apoi în mod repetat, cât timp condiţia specificată este adevărată.

Sintaxă:

do

{

codul ce trebuie executat

}

while (*propozitielogica*);

Instrucţiunea for...in

Această instrucţiune este utilizată pentru a parcurge elementele unui tablou sau a enumera proprietăţile unui obiect.

Sintaxă:

for (*variabila* in *obiect*)

{

cod ce trebuie executat

}

Obs: Codul din corpul instrucţiunii este executat câte o dată pentru fiecare element din tablou sau proprietate.

Obs: Argumentul variabila poate fi o variabilă, un element de tablou sau o proprietate a unui obiect.

Exemplu

Instrucţiunea for..in este utilizată pentru a parcurge elementele unui tablou:

```
<html>
<body>
<h3>Parcurgerea elementelor unui tablou cu instructiunea for..in</h3> <hr/>
<script type="text/javascript">
var x;
var pets = new Array();
pets[0] = "Pisica";
pets[1] = "Caine";
pets[2] = "Papagal";
pets[3] = "Hamster";
document.write("Valorile memorate in tablou sunt:"+"<br/>");
for (x in pets)
{
document.write(pets[x] + "<br />");
}
</script>
</body>
</html>
```

Evenimentele JavaScript

Utilizând JavaScript, putem crea pagini web dinamice. Evenimentele sunt acţiuni ce pot fi detectate de JavaScript. Fiecare element dintr-o pagină web are un anumit număr de evenimente care pot declanşa un script. Spre exemplu, putem utiliza evenimentul onClick al unui buton pentru a indica ce funcţie va fi executată dacă utilizatorul acţionează butonul respectiv. Evenimentele sunt definite în tagurile HTML.

Exemple de evenimente

- Un click de mouse
- Încărcarea unei pagini web sau a unei imagini
- Mişcarea mouse-ului peste o anumită zonă din pagina web
- Selectarea unui câmp de intrare dintr-un formular HTML
- Submiterea unui formular HTML
- Apăsarea unei taste

Obs: Evenimentele sunt în mod normal asociate cu funcţii, care nu vor fi executate înainte de a avea loc evenimentul.

Evenimentele onLoad şi onUnload

Aceste evenimente sunt declanşate când utilizatorul întră într-o pagină web, respectiv când părăseşte pagina.

Evenimentul onLoad este folosit în mod frecvent pentru a detecta tipul şi versiunea browserului utilizatorului şi a încărca varianta de pagină potrivită cu aceste informaţii.

Ambele evenimente sunt folosite frecvent pentru a stabili ce cookies vor fi setate când utilizatorul întră în sau părăseşte pagina. Spre exemplu, puteţi întreba care este numele utilizatorului când acesta vizitează prima dată pagina.

Numele oferit de utilizator este memorat într-un cookie. Data viitoare când utilizatorul vă vizitează pagina, puteţi să-l întâmpinaţi cu un mesaj personalizat cu numele său.

Evenimentele onFocus, onBlur şi onChange

Aceste evenimente sunt utilizate frecvent împreună cu validarea câmpurilor unui formular.

Exemplul următor ilustrează utilizarea evenimentului onChange. Funcţia verificaEmail() va fi apelată ori de câte ori utilizatorul modifică conţinutul unui câmp:

<input type="text" size="30" id="email" onchange="verificaEmail()">

Evenimentul onSubmit

Acest eveniment este utilizat pentru a valida toate câmpurile unui formular înainte de trimiterea lui către server.

Exemplul următor ilustrează utilizarea evenimentului onSubmit. Funcţia verificaFormular() va fi apelată atunci când utilizatorul acţionează butonul submit din formular. Dacă valorile introduse în câmpuri nu sunt valide, trimiterea formularului va fi anulată. Funcţia verificaFormular() returnează true sau false. Dacă funcţia va returna valoarea false, trimiterea formularului va fi anulată:

<form method="post" action="xxx.htm" onsubmit="return verificaFormular()">

Evenimentele onMouseOver şi onMouseOut

Aceste evenimente sunt utilizate frecvent pentru a crea butoane „animate”.

În exemplul următor va fi afişată o casetă de alertă când este detectat un eveniment onMouseOver:

```
<a href=http://www.google.com onmouseover="alert('Un eveniment onMouseOver detectat');return false"> <img src="flowers.gif" alt="Flowers" /></a>
```

Instrucţiunea try...catch

Când navigăm prin paginile web de pe internet, pot să apară mesaje de eroare la încărcarea unei pagini. În acest caz, uzual, apare o casetă de alertare JavaScript care ne anunţă că s-a detectat o eroare de execuţie (runtime error) şi ne întreabă dacă dorim să depanăm codul paginii. Aceste mesaje sunt utile pentru proiectanţii paginilor web, nu şi pentru vizitatori care, de obicei, părăsesc pagina respectivă. În acest capitol veţi învăţa cum să gestionaţi mesajele de eroare JavaScript, astfel încât să nu vă pierdeţi audienţa.

Instrucţiunea try...catch vă permite să testaţi blocurile de cod pentru a depista erorile. Blocul try conţine codul ce trebuie executat, iar blocul catch conţine codul ce va fi executat dacă apare o eroare.

Sintaxă:

```
try
{
codul ce trebuie executat
}
catch(err)
{
gestionarea erorilor
}
```

Exemple

În exemplul următor ar trebui afişată o casetă de alertare cu mesajul "Bine ati venit!" când butonul este acţionat. Totuşi, în corpul funcţiei mesaj() există o eroare, cuvântul rezervat alert este scris greşit. Această eroare va fi detectată de JS. Blocul catch sesizează eroarea şi execută un cod special pentru a o rezolva. Acest cod afişează un mesaj de eroare pentru a informa utilizatorul ce se întâmplă. Dacă utilizatorul apasă butonul OK, încărcarea paginii va continua fără probleme:

```
<html>
```

```
<head>
<script type="text/javascript">
var txt="";
function mesaj()
{
try
{
adddlert("Bine ati venit!");
}
catch(err)
{
text="In aceasta pagina este o eroare.\n\n";
text+="Descrierea erorii: " + err.description +
"\n\n";
text+="Pentru a continua apasati OK.\n\n";
alert(text);
}
}
</script>
</head>
<body>
<h3>Utilizarea instructiunii try..catch pentru sesizarea erorilor</h3> <hr/>
<input type="button" value="Vedeti mesajul" onclick="mesaj()" />
</body>
</html>
```

În exemplul următor alert este de asemenea scris greşit. Blocul catch utilizează o casetă de confirmare pentru a afişa un mesaj care informează utilizatorii că pot apăsa OK pentru a continua să viziteze pagina în care a fost depistată eroarea sau pot apăsa Cancel dacă doresc să se întoarcă la pagina principală (homepage). Dacă metoda confirm returnează false (utilizatorul a acţionat butonul Cancel), atunci utilizatorul este redirectat. Dacă confirm returnează true, codul din blocul catch nu are nici-un efect:

```
<html>
<head>
<script type="text/javascript">
var txt="";
function mesaj()
{
try
{
adddlert("Bine ati venit!");
}
catch(err)
{
text="In aceasta pagina este o eroare.\n\n";
```

```
text+="Apasati OK daca doriti sa continuati
vizualizarea paginii,\n";
text+="sau Cancel pentru a va intoarce la pagina principala.\n\n";
if(!confirm(text))
{
document.location.href="http:
//carpenmanuela.wik.is/";
}
}
}
</script>
</head>
<body>
<h3>Un alt exemplu de utilizare a instructiunii  try..catch</h3> <hr/>
<input type="button" value="Vedeti mesajul"
onclick="mesaj()" />
</body>
</html>
```

Instrucţiunea throw

Această instrucţiune vă permite să creaţi o excepţie. Dacă o utilizaţi împreună cu instrucţiunea try...catch, puteţi controla execuţia programului şi afişa mesaje de eroare adecvate.

Sintaxă:

throw(exceptie)

Argumentul exceptie poate fi un şir de caractere, un număr întreg, o valoare booleană sau un obiect.

Exemplu:

Exemplul următor testează valoarea variabilei x. Dacă valoarea este mai mare decât 10, mai mică decât 10 sau nu este un număr, blocul throw aruncă o eroare. Această eroare este prinsă de blocul catch care afişează un mesaj corespunzător:

```
<html>
<body>
<h3>Utilizarea instructiunii throw pentru tratarea corespunzatoare a erorilor</h3> <hr/>
<script type="text/javascript">
var x=prompt("Introduceti un numar cuprins intre 0 si 10:","");
try
{
if(x>10)
```

```
{
throw "Err1";
}
else if(x<0)
{
throw "Err2";
}
else if(isNaN(x))
{
throw "Err3";
}
}
catch(er)
{
if(er=="Err1")
{
alert("Eroare! Valoarea este prea mare");
}
if(er=="Err2")
{
alert("Eroare! Valoarea este prea mică");
}
if(er=="Err3")
{
alert("Eroare! Valoarea nu este un numar");
}
}
</script>
</body>
</html>
```

Exemplu:

Exemplul următor ilustrează utilizarea evenimentului onerror.

```
<html>
<head>
<script type="text/javascript">
onerror=handleErr;
var txt="";
function handleErr(msg,url,l)
{
txt="In aceasta pagina este o eroare.\n\n";
txt+="Eroare: " + msg + "\n";
txt+="URL: " + url + "\n";
txt+="Linie: " + l + "\n\n";
txt+="Pentru a continua apasati OK.\n\n";
```

```
alert(txt);
return true;
}
function message()
{
adddlert("Bine ai venit!");
}
</script>
</head>
<body>
<h3>Exemplu de utilizare a evenimentului onerror</h3>
<hr/>
<input type="button" value="Afiseaza mesajul"
onclick="message()" />
</body>
</html>
```

Obiectele JavaScript

JavaScript este un limbaj de programare orientat pe obiecte (POO). Un limbaj POO vă permite să vă definiţi propriile obiecte şi propriile tipuri de variabile.

Crearea propriilor obiecte va fi explicată mai târziu, în secţiunea JavaScript avansat. Vom începe prin a examina obiectele încorporate în JS şi cum sunt ele utilizate. Reţineţi că un obiect este de fapt, un tip special de date care are proprietăţi şi metode.

Proprietăţi

Proprietăţile sunt valori asociate cu un obiect.

În exemplul următor, utilizăm proprietatea length a obiectului String (şir de caractere) pentru a determina numărul de caractere memorate într-un şir:

```
<script type="text/javascript">
var txt="Bine ati venit!";
document.write(txt.length);
</script>
```

Codul de mai sus va afişa valoarea: 15

Metode

Metodele sunt acţiuni ce pot fi realizate cu un obiect.

În exemplul următor, utilizăm metoda UpperCase() a obiectului String pentru a afişa un text cu litere mari:

```
<script type="text/javascript">
var txt="Bine ati venit!";
document.write(txt.toUpperCase());
</script>
```

Codul de mai sus va afişa şirul: BINE ATI VENIT!

Obiectul String

Obiectul String este folosit pentru a manipula secvenţe de caractere (text). Un obiect String este creat cu instrucţiunea new String().

Sintaxa:

var txt = new String(string);

sau mai simplu:

var txt = string;

Proprietăţile obiectului String

constructor

Returnează funcţia care a creat prototipul obiectului String

length

Returnează lungimea şirului

prototype

Permite adăugarea de proprietăţi şi metode unui obiect

Metodele obiectului String

charAt() Returnează caracterul cu indexul specificat

charCodeAt()

Returnează codul Unicode al caracterului cu indexul

specificat

concat()

Concatenează două sau mai multe şiruri şi returnează

şirul obţinut

fromCharCode()

Converteşte valori Unicode în caractere

indexOf()

Returnează poziţia primei apariţii a unui subşir într-un şir

lastIndexOf()

Returnează poziţia ultimei apariţii a unui subşir într-un şir

match()

Caută potrivirile dintre un subşir şi un string şi returnează subşirul sau null (dacă subşirul nu este găsit)

replace()

Caută toate apariţiile unui subşir într-un şir şi le înlocuieşte cu un nou subşir

search()

Caută potrivirea dintre un subşir şi un şir şi returnează poziţia în care apare potrivirea

slice()

Elimină o porţiune din şir şi returnează şirul extras

split()

Împarte un şir în subşiruri pe baza unui caracter separator

substr()

Extrage dintr-un şir secvenţa de caractere care începe într-o anumită poziţie şi are o anumită lungime

substring()

Extrage dintr-un şir caracterele situate între două poziţii

toLowerCase()

Converteşte un şir în litere mici

toUpperCase()

Converteşte un şir în litere mari

valueOf()

Returnează valoarea primară a unui obiect String

Metode împachetate în taguri HTML

Aceste metode returnează şirul împachetat în tagurile HTML potrivite.

anchor()

Creează o ancoră

big()

Afişează şirul cu font mare

blink()

Afişează un şir care clipeşte

bold()

Afişează şirul cu font bold

fixed()

Afişează şirul cu un font cu pas fix

fontcolor()

Afişează şirul folosind o anumită culoare

fontsize()

Afişează şirul cu o anumită dimensiune a fontului

italics()

Afişează şirul cu font italic

link()

Afişează şirul ca hiperlegătură

small()

Afişează şirul cu font mic

strike()

Afişează şirul ca tăiat

sub()

Afişează şirul ca subscript (indice)

sup()

Afişează şirul ca superscript (exponent)

Exemplul 1

Ilustrează utilizarea proprietăţii length pentru a determina lungimea unui şir.

```
<html>
<body>
<h3>Obiectul String. Determinarea lungimii unui sir</h3> <hr/>
<script type="text/javascript">
var txt="Bine ati venit!";
document.write("Sirul este: "+txt+"<br/>");
document.write("Are lungimea "+txt.length);
</script>
<p><b>Obs.</b>Sirul nu se modifica.</p>
```

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

Exemplul 2

Ilustrează cum se utilizează tagurile HTML pentru a stiliza un şir.

```
<html>
<body>
<h3>Obiectul String. Utilizarea tagurilor HTML pentru stilizarea unui sir.</h3> <hr/>
<script type="text/javascript">
var txt="Bine ati venit!";
document.write("<p>Big: " + txt.big() + "</p>");
document.write("<p>Small: " + txt.small() + "</p>");
document.write("<p>Bold: " + txt.bold() + "</p>");
document.write("<p>Italic: " + txt.italics() + "</p>");
document.write("<p>Blink: " + txt.blink() + " (nu functioneaza in IE, Chrome, Safari)</p>");
document.write("<p>Fixed: " + txt.fixed() + "</p>");
document.write("<p>Strike: " + txt.strike() + "</p>");
document.write("<p>Fontcolor: " + txt.fontcolor("Blue") + "</p>");
document.write("<p>Fontsize: " + txt.fontsize(14) + "</p>");
document.write("<p>Subscript: " + txt.sub() + "</p>");
document.write("<p>Superscript: " + txt.sup() + "</p>");
document.write("<p>Link: " + txt.link("http://www.google.com") + "</p>");
</script>
<br/> <br/>
<p><b>Obs.</b>Sirul stilizat nu se modifica!</p>
```

```
</body>

</html>
```

Exemplul 3

Ilustrează cum se utilizează metoda concat() pentru a concatena şiruri.

Concatenarea a două şiruri:

```
<html>

<body>

<h3>Obiectul String. Concatenarea a doua siruri.</h3>

<hr/>

<script type="text/javascript">

var txt1 = "Buna ";

var txt2 = "ziua!";

document.write("Primul sir este: "+txt1+"<br/>");

document.write("Al doilea sir este: "+txt2+"<br/>");

document.write("Sirul concatenat este: "+txt1.concat(txt2)+"<br/>");

</script>

<p><b>Obs.</b>Sirurile concatenate nu se modifica. Rezultatul concatenarii poate fi pastrat intr-un nou sir.</p>

</body>

</html>
```

Concatenarea a trei şiruri:

```
<html>

<body>

<h3>Obiectul String. Concatenarea a trei siruri.</h3>
```

```
<hr/>
<script type="text/javascript">
var txt1="Buna ";
var txt2="ziua!";
var txt3=" Bine ati venit!";
document.write("Primul sir este: "+txt1+"<br/>");
document.write("Al doilea sir este: "+txt2+"<br/>");
document.write("Al treilea sir este: "+txt3+"<br/>");
document.write("Sirul concatenat este: "+txt1.concat(txt2,txt3)+"<br/>");
</script>
<p><b>Obs.</b>Sirurile concatenate nu se modifica. Rezultatul concatenarii poate fi pastrat intr-un nou sir.</p>
</body>
</html>
```

Exemplul 4

Ilustrează cum se utilizează metoda indexOf() pentru a determina poziţia primei apariţii a unei valori într-un şir.

```
<html>
<body>
<h3>Obiectul String. Cautarea primei aparitii a unei valori in sir cu indexof().</h3> <hr/>
<script type="text/javascript">
var str="Buna ziua!";
document.write("Sirul in care se cauta este: "+str+"<br/>");
document.write("Sirul \"Buna\" apare in sir in pozitia "+str.indexOf("Buna") + "<br />");
document.write("Sirul \"ZIUA\" apare in sir in pozitia "+str.indexOf("ZIUA") + "<br />");
```

```
document.write("Sirul \"ziua\" apare in sir in pozitia "+str.indexOf("ziua"));

</script>

<p><b>Obs.</b>Sirul nu se modifica in urma cautarii!</p>

</body>

</html>
```

Valorile afişate sunt: 0 -1 5.

Obs. Dacă valoarea nu apare în şir, valoarea returnată este -1. Şirurile sunt indexate de la 0.

Exemplul 5

Ilustrează cum se utilizează metoda match() pentru a căuta un subşir într-un şir. Metoda returnează subşirul, dacă este găsit, sau valoarea null, dacă subşirul nu este găsit în şir.

```
<html>

<body>

<h3>Obiectul String. Cauta unui subsir intr-un sir cu match().</h3> <hr/>

<script type="text/javascript">

var str="Hello world!";

document.write("Sirul in care se cauta este: "+str+"<br/>");

document.write("Sirul cautat: "+"world"+". ");

document.write("Valoarea returnata: "+str.match("world") + "<br />");

document.write("Sirul cautat: "+"World"+". ");

document.write("Valoarea returnata: "+str.match("World") + "<br />");

document.write("Sirul cautat: "+"worlld"+". ");

document.write("Valoarea returnata: "+str.match("worlld") + "<br />");
```

</script>

 Obs.Sirul nu se modifica in urma cautarii. Rezultatul poate fi memorat intr-o variabila.

</body>

</html>

Obiectul Date

Obiectul Date este utilizat pentru a lucra cu date calendaristice şi ore. Un obiect de tip Date este creat cu instrucţiunea new Date(). Sunt patru metode de a instanţia un obiect Date:

var d = new Date();

var d = new Date(milisecunde);

var d = new Date(dataString);

var d = new Date(an, luna, zi, ore, minute, secunde, milisecunde);

Setarea datei

Putem manevra uşor datele calendaristice folosind metodele obiectului Date.

În exemplul următor, data este setată la 18 aprilie 2016:

var myDate=new Date();

myDate.setFullYear(2016,4,18);

În exemplul următor, data este setată la şapte zile în viitor:

var myDate=new Date();

myDate.setDate(myDate.getDate()+7);

Compararea a două date calendaristice

Exemplul următor compară data curentă cu 18 aprilie 2016:

```
var myDate=new Date();
myDate.setFullYear(2016,4,18);
var today = new Date();
if (myDate>today)
{
alert("Astazi este inainte de 18 aprilie 2016");
}
else
{
alert("Astazi este dupa 18 aprilie 2016");
}
```

Metodele obiectului Date

getDate() Returnează ziua din lună (între 1 şi 31)

getDay() Returnează ziua din săptămână (0-6)

getFullYear() Returnează anul (patru cifre)

getHours() Returnează ora (0-23)

getMilliseconds() Returnează milisecundele (0-999)

getMinutes() Returnează minutele (0-59)

getMonth() Returnează luna (0-11)

getSeconds() Returnează secundele (0-59)

getTime() Returnează numărul de milisecunde scurse de la 1.01.1970

getTimezoneOffset() Returnează diferenţa dintre GMT şi timpul local, în minute

parse() Analizează(parsează) o dată ca şir de caractere şi returnează numărul de milisecunde scurse de la 1.01.1970

setDate() Setează data din lună (1-31)

setFullYear() Setează anul (patru cifre)

setHours() Setează ora (0-23)

setMilliseconds() Setează milisecundele (0-999)

setMinutes() Setează minutele (0-59)

setMonth() Setează lunile (0-11)

setSeconds() Setează secundele (0-59)

setTime() Setează o dată şi o oră adunând sau scăzând un anumit număr de milisecunde la/din 1.01.1970

toDateString() Converteşte porţiunea corespunzătoare datei calendaristice dintr-un obiect Date intr-un şir de caractere

toString() Converteşte un obiect Date într-un şir de caractere

toTimeString() Converteşte porţiunea corespunzătoarea timpului

dintr-un obiect Date într-un şir de caractere

valueOf() Returnează valoarea primară a unui obiect Date

Exemplul 1

Ilustrează utilizarea metodei Date() pentru a obţine data curentă.

```
<html>

<body>

<h3>Obiectul Date. Obtinerea datei curente cu

Date().</h3> <hr/>

<script type="text/javascript">

document.write("Astazi este: "+Date());

</script>

</body>
```

```
</html>
```

Exemplul 2

Ilustrează utilizarea metodei getTime() pentru a calcula anii scurşi din 1970 până în prezent.

```
<html>
<body>
<h3>Obiectul Date. Utilizarea metodei getTime().</h3>
<hr/>
<script type="text/javascript">
var d=new Date();
document.write("Au trecut "+d.getTime() + " milisecunde din 01.01.1970 si pana acum.");
</script>
</body>
</html>
```

Exemplul 3

Ilustrează utilizarea metodei setFullYear() pentru a seta o dată specifică.

```
<html>
<body>
<h3>Obiectul Date. Setarea datei cu setFullYear().</h3> <hr/>
<script type="text/javascript">
var d = new Date();
d.setFullYear(2010,1,19);
document.write("Data a fost setata la "+d);
</script>
```

```
</body>

</html>
```

Exemplul 4

Ilustrează utilizarea metodei toString() pentru a converti data curentă într-un şir de caractere.

```
<html>

<body>

<script type="text/javascript">

var d=new Date();

document.write(d.toString());

</script>

</body>

</html>
```

Exemplul 5

Ilustrează utilizarea metodei getDay() şi a unui tablou pentru a scrie denumirea zilei din săptămână curente.

```
<html>

<body>

<h3>Obiectul Date. Utilizarea metodei getDay() pentru a determina ziua din saptamana.</h3>
<hr/>

<script type="text/javascript">

var d=new Date();

var weekday=new Array(7);

weekday[0]="Duminica";

weekday[1]="Luni";
```

```
weekday[2]="Marti";
weekday[3]="Miercuri";
weekday[4]="Joi";
weekday[5]="Vineri";
weekday[6]="Sambata";
document.write("Astazi este " + weekday[d.getDay()]);
</script>
</body>
</html>
```

Exemplul 6

Ilustrează cum se poate afişa un ceas într-o pagină web.

```
<html>
<head>
<script type="text/javascript">
function ceas()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
//functia urmatoare adauga un zero in fata
//numerelor<10
m=verifica(m);
s=verifica(s);
```

```
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('ceas()',500);
}
function verifica(i)
{
if (i<10)
{
i="0" + i;
}
return i;
}
</script>
</head>
<body onload="ceas()">
<h3>Obiectul String. Afisarea unui ceas.</h3> <hr/>
<div id="txt"></div>
</body>
</html>
```

Obiectul Array

Un tablou este o variabilă specială care poate păstra la un moment dat mai multe valori de un anumit tip. Dacă aveți o listă de elemente, animale de companie de exemplu, aţi putea păstra valorile în variabile simple, ca în exemplul următor:

```
pet1="Caine";
pet2="Pisica";
pet3="Papagal";
```

Desigur, problema se complică dacă aveţi de memorat zeci, sau sute de valori. Cea mai bună soluţie este să folosiţi tablouri. Un tablou poate reţine toate valorile sub un singur nume şi puteţi accesa fiecare valoare stocată în tablou folosind numele tabloului şi indexul valorii.

Crearea unui tablou

Un tablou poate fi definit în trei moduri:

1:

```
var pets=new Array();

//tablou obisnuit

pets[0]="Caine";

pets[1]="Pisica";

pets[2]="Papagal";
```

2:

```
var pets=new Array("Caine","Pisica","Papagal");

//tablou condensat
```

3:

```
var pets=["Caine","Pisica","Papagal"];

//tablou literal
```

Obs: Dacă în tablou stocaţi valori numerice sau logice, tipul tabloului va fi Number sau Boolean, în loc de String.

Accesarea elementelor dintr-un tablou

Puteţi accesa un element dintr-un tablou precizând numele tabloului şi indexul elementului. Primul element din tablou are indexul 0.

Următoarea linie de cod

```
document.write(pets[0]);
```

va afişa şirul: Caine

Modificarea valorilor dintr-un tablou

Pentru a modifica o valoare dintr-un tablou, este suficient să atribuiţi o nouă valoare elementului respectiv, ca în exemplul următor:

pets[0]="Iguana";

Proprietăţile obiectului Array

constructor Returnează funcţia care a creat prototipul obiectului Array

length Setează sau returnează numărul elementelor stocate în tablou

prototype Permite adăugare de proprietăţi şi metode unui obiect

Metodele obiectului Array

concat() Concatenează două sau mai multe tablouri şi returnează tabloul obţinut

join() Concatenează toate elementele unui tablou într-un şir de caractere

pop() Înlătură ultimul element dintr-un tablou şi returnează respectivul element

push() Adaugă noi elemente la sfârşitul unui tablou şi returnează noua lungime a tabloului

reverse() Răstoarnă ordinea elementelor dintr-un tablou

shift() Înlătură primul element dintr-un tablou şi returnează respectivul element

slice() Selectează o parte dintr-un tablou şi returnează elementele selectate

sort() Sortează elementele unui tablou

splice() Adaugă/Înlătură elemente dintr-un tablou.

toString() Converteşte un tablou în şir de caractere şi returnează rezultatul

unshift() Adaugă noi elemente la începutul unui tablou şi returnează noua lungime a tabloului

valueOf() Returnează valoarea primară a unui tablou

Exemplul 1

Ilustrează crearea unui tablou, atribuirea de valori şi afişarea elementelor tabloului.

```
<html>
<body>
<h3>Obiectul Array. Crearea unui tablou, initializarea si afisarea elementelor.</h3> <hr/>
<script type="text/javascript">
var pets = new Array();
pets[0] = "Pisica";
pets[1] = "Caine";
pets[2] = "Perus";
document.write("Elementele memorate in tablou sunt:"+"<br/>");
for (i=0;i<pets.length;i++)
{
document.write(pets[i] + "<br />");
}
</script>
</body>
</html>
```

Exemplul 2

Ilustrează utilizarea instrucţiunii for...in pentru a parcurge elementele unui tablou.

```
<html>
<body>
<h3>Obiectul Array. Afisarea elementelor unui tablou cu instructiunea for..in.</h3> <hr/>
<script type="text/javascript">
```

```
var x;
var pets = new Array();
pets[0] = "Pisica";
pets[1] = "Caine";
pets[2] = "Perus";
document.write("Elementele memorate in tablou sunt:"+"<br/>");
for (x in pets)
{
document.write(pets[x] + "<br />");
}
</script>
</body>
</html>
```

Exemplul 3

Ilustrează utilizarea metodei concat() pentru a concatena trei tablouri.

```
<html>
<body>
<h3>Obiectul Array. Concatenarea a trei tablouri cu concat().</h3> <hr/>
<script type="text/javascript">
var parinti = ["Maria", "George"];
var copii = ["Elena", "Mihai"];
var frati = ["Paul", "Dan"];
var familie = parinti.concat(copii,frati);
document.write("Parinti: "+parinti+"<br/>");
```

```
document.write("Copii: "+copii+"<br/>");

document.write("Frati: "+frati+"<br/>");

document.write("Familia: "+familie);

</script>

<p><b>Obs.</b>Tablourile concatenate nu se modifica. Rezultatul concatenarii este un nou tablou.</p>

</body>

</html>
```