Lucrare de laborator nr. 2
JavaScript (I)

1. Scopul lucrării

În această lucrare se studiază realizarea de scripturi JavaScript şi modalitățile de includere ale acestora într-un document HTML. Aplicațiile folosesc instrucțiunile de atribuire, decizie şi repetare din limbajul JavaScript. Se studiază deasemenea despre funcții în JavaScript şi despre vectori (*array)*.

2. Breviar teoretic

JavaScript este un limbaj de programare folosit în dezvoltarea de aplicații web pentru a schimba conținutul sau valorile atributelor unor elemente HTML dintr-un document web, cât şi pentru a face diverse calcule într-o pagină web (validări ale datelor introduse de utilizator, formule, etc.). Programul scris în limbajul JavaScript este executat de către browserul web.

Codul JavaScript poate fi plasat în secțiunea *head* sau în secțiunea *body* a unui document HTML, sau în fişiere externe ce au extensia **.js** . Introducerea codului JavaScript într-un document HTML, se face cu ajutorul tagului *script*. Pentru codul JavaScript care se găseşte într-un fişier extern, trebuie specificat în tagul *script* şi atributul *src*, cu valoarea numelui fişierului (dacă nu se găseşte în acelaşi folder cu documentul HTML, trebuie specificată şi calea). Exemplu:

<script src=”calcul.js”>

Browserele web nu sunt singurele platforme care rulează cod JavaScript. Astfel platforma Node.js foloseşte JavaScript pentru programarea aplicațiilor. De asemenea sunt baze de date ce folosesc JavaScript ca limbaj pentru interogări.

Limbajul JavaScript este diferit de limbajul Java (numele de JavaScript a fost ales din considerente comerciale). În acelaşi timp,

mare parte din sintaxa limbajului JavaScript este asemănătoare cu sintaxa limbajului C (deci şi cu limbajul Java).

Ca şi în limbajul Java, avem acelaşi operator de atribuire = şi avem cei 5 operatori aritmetici (pentru adunare +, scădere -, înmulțire *, împărțire / şi operatorul modulo %.).

Observație:

JavaScript, spre deosebire de limbajul C şi limbajul Java, nu face trunchierea la împărțirea între două numere întregi. Exemplu:

În limbajele C şi Java:

double nr=1/2; //nr=0 (se face trunchiere)

În limbajul JavaScript:

var nr=1/2; //nr=0.5 (nu se face trunchiere)

Comentariile la nivel de linie // şi pentru un bloc /* şi */ , sunt aceleaşi ca în limbajul C.

Pentru a declara variabile, se foloseşte cuvântul cheie *var* .

Numele variabilelor în JavaScript, ca şi în Java este sensibil la litere mici/mari. Spre deosebire de Java, aceeaşi variabilă poate fi folosită pentru a primi tipuri diferite de date.

Tipul unei variabile poate fi aflat folosind operatorul *typeof.*

În JavaScript sunt 6 tipuri de bază de valori: numere, stringuri, Boolean, obiecte, funcții şi valori nedefinite.

Ca şi în limbajul Java o constantă String se introduce folosind ghilimelele duble ” Dar, se poate folosi şi caracterul apostrof ’ .
Exemplu:

“creion”

sau

‘creion’

La fel ca şi în Java se poate folosi operatorul + pentru a concatena şiruri.

var s=”Ion ”+”Ionescu”;

Pentru a afişa date se poate folosi metoda *alert* a obiectului Window. Ea afişează o casetă cu mesajul text transmis ca parametru, şi are un buton OK de confirmare.

Primul exemplu complet de cod JavaScript inclus într-un document HTML, pe care îl dăm în această lucrare de laborator, este pentru afişarea (folosind metoda *alert)* următorului text din Biblie :
Căutați mai întâi Împărăția lui Dumnezeu şi neprihănirea Lui !

```
<!DOCTYPE html>
<html>
```

```
<body>
<h2>Primul program</h2>
<script>
 alert(
"Cautati mai intai Imparatia lui Dumnezeu si neprihanirea Lui !");
</script>
</body>
</html>
```

Putem afişa date şi cu metoda *write* a obiectului *document*, dar doar pentru teste asupra paginii web (*document.write* apelată după ce o pagină web a fost încărcată , va şterge tot ce s-a afişat.)

Pentru a afişa date (mesaje text şi valorile unor variabile) , fără a mai fi nevoie de fiecare dată să facem un click pe butonul OK, pentru confirmare, ca în cazul metodei *alert*, putem folosi funcţia *console.log* , care scrie rezultatele în consola limbajului JavaScript. Consola se foloseşte şi pentru a vizualiza erorile de sintaxă JavaScript. Această consolă se afişează la majoritatea browserelor, apăsând tasta F12 (implicit, consola este invizibilă). Exemplu:

console.log("O zi buna !");

Pentru a citi date de la tastatură se poate folosi metoda *prompt* a obiectului *window*. Ea afişează o cutie de dialog, în care utilizatorul introduce datele de intrare. Are două argumente: primul este textul care se afişează în cutia de dialog. Al doilea argument este opţional şi specifică textul implicit de intrare.

Metoda *prompt* returnează sub formă de String data de intrare introdusă, dacă utilizatorul face click pe butonul OK, altfel dacă utilizatorul face click pe butonul Cancel, metoda returnează *null*.

Exemplu:

```
var s=prompt("nume universitate","Universitatea ");
var sir=prompt("numar=");
```

În continuare vom da un exemplu complet de cod JavaScript inclus într-un document HTML, pentru afişarea la consolă, folosind operatorul *typeof*, a tipurilor de date pentru anumite valori constante.

```
<!DOCTYPE html>
<html>
<body>
<p>Exemplu pentru afisarea tipului unei date</p>
<script>
```

```
 console.log(typeof -1.5);
 console.log(typeof "abc");
 console.log(typeof true);
</script>
</body>
</html>
```

JavaScript este mult mai liberă în ceea ce priveşte conversia automată de tipuri de date, decât limbajul Java.

Astfel, operatorul binar + folosit pentru a aduna două numere sau pentru a concatena două stringuri, doar dacă primul operand este un string şi al doilea este un număr, va forța conversia automată a operandului numeric la string şi apoi va realiza concatenarea celor două stringuri. Exemplul următor ilustrează această conversie automată.

```
<!DOCTYPE html>
<html>
<body>
<p>Exemplu pentru operatorul + de la Stringuri</p>
<script>
 var s1=2+3+" creioane." //face adunare, deci 5 creioane.
 console.log(s1);
 var s2="creioane : "+2+3+".";
 //Fiind mai intai un string, face concatenare, deci creioane : 23.
 console.log(s2);
</script>
</body>
</html>
```

Ultimul exemplu complet de cod JavaScript inclus într-un document HTML pe care îl dăm, ilustrează sintaxa instrucțiunii *switch*, identică ca în Java.

Citim de la tastatură un număr natural între 1 şi 7 , şi afişăm la ziua corespunzătoare din săptămână.

```
<!DOCTYPE html>
<html>
<body>
<h2>Exemplu instructiunea switch</h2>
```

```
<script>
 var nr=Number(prompt(
          "Introduceti un numar natural intre 1 si 7 :",""));
 switch(nr){
  case 1: alert("Duminica");
          break;
  case 2: alert("Luni");
          break;
  case 3: alert("Marti");
          break;
  case 4: alert("Miercuri");
          break;
  case 5: alert("Joi");
          break;
  case 6: alert("Vineri");
          break;
  case 7: alert("Sambata");
          break;
  default:alert("Ati dat un numar gresit!");
 }
</script>
</body>
</html>
```

În continuare ilustrăm folosirea metodei *prompt* pentru a citi două numere de la tastatură. Pentru cele două numere calculăm şi afişam maximul dintre ele. Vom da doar codul JavaScript.

```
<script>
 var nr1=Number(prompt("nr1=",""));
 //prompt() returneaza un string
 //Folosim functia Number() pentru a converti un string
 // la numar.
 var nr2=Number(prompt("nr2=",""));
 var max;
 if(nr1>nr2)max=nr1;
 else max=nr2;
 alert("maxim dintre "+nr1+" si "+nr2+" = "+max);
</script>
</body>
</html>
```

```
</script>
```

Funcţii

O funcţie se defineşte cu ajutorul cuvântului cheie *function* . Exemplu:

```
function diametru(raza)
{
  return 2*raza;
 }
```

Variabilele declarate în interiorul unei funcţii sunt variabile locale. Parametrii funcţiei sunt ca şi variabile locale ale ei. Variabilele declarate într-un script, în afara oricărei funcţii, sunt variabile globale. În HTML, toate variabilele globale ale unui script JavaScript, aparţin obiectului *window*. Ele pot fi accesate şi din interiorul unei funcţii cu condiţia ca să nu mai fie declarată o variabilă locală, care să aibă acelaşi nume, în acea funcţie.

În JavaScript o funcţie poate fi apelată într-un punct al programului, chiar dacă ea este definită după codul care a apelat-o.

Ca exemplu de funcţie vom scrie funcţia *estePrim* ce are ca argument un număr şi scoate ca rezultat valoarea *true*, dacă numărul este prim.

```
<!DOCTYPE html>
<html>
<body>
<h2>Numar prim</h2>
<p>Presupunem ca numarul se introduce corect!</p>

<script>
 var nr=Number(prompt("nr="));
 if(estePrim(nr))
   alert("Numarul "+nr+" este prim.");
              else  alert("Numarul "+nr+" nu este prim.");
 function estePrim(x)
 {
  var i;
  for(i=2;i<=Math.floor(Math.sqrt(x));i++)
    if(x%i==0)return false;
  return true;
```

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

Vectori

Un vector (array) este declarat cu ajutorul parantezelor drepte : [ ].

Exemplu:

var nume=["Ion","Maria","George"];

Elementele tabloului se accesează prin indexare, primul element are indexul 0, ca şi în Java.

Dimensiunea vectorului este memorată în proprietatea *length* .

Putem adăuga elemente la sfârşitul unui vector şi prin folosirea metodei *push.* Exemplu:

var a=[];
a.push(7);//echivalent cu: a[0]=7:
a.push(3.4);

Putem scoate ultimul element dintr-un vector, folosind metoda *pop*. Exemplu:

var nr=a.pop();

*Î*n mod similar, pentru a adăuga sau a scoate un element de la **începutul** unui vector, se folosesc metodele *unshift* şi respectiv *shift* .

Metoda *indexOf* are ca parametru de intrare un element şi ea returnează indexul primei apariţii a acestui element, în vector. Dacă elementul căutat nu este prezent, metoda va returna valoarea -1 . Similar, metoda *lastIndexOf* returnează indexul ultimei apariţii a elementului în vector.

O altă metodă de bază pentru vectori este metoda *slice* , ce are doi parametrii: un index de început (inclusiv) şi un index de sfârşit (exclusiv). Metoda returneză un vector ce are elementele vectorului pentru care se apelează metoda, cuprinse între cei doi indecsi. Exemplu:

var b=[2,-10,-20,6].slice(1,3); //[-10,-20]

Al doilea parametru al metodei *slice* , poate să lipsească, şi în acest caz vor fi luate în noul vector toate elementele de la indexul de început până la sfârşitul vectorului.

Pentru a concatena doi vectori se foloseşte metoda *concat.*

Ca exemplu pentru vectori (modul de declarare, proprietatea *length*, şi modul de accesare al valorilor din vector), vom calcula, folosind o funcție separată, numărul de numere pozitive dintr-un vector inițializat cu câteva valori numerice.

```
<!DOCTYPE html>
<html>
<body>
<h2>Exemplu de array</h2>
<script>
 var a=[1,5,-2,-7,10];
 var contorPozitive=calcul(a);
 alert("contor = "+calcul(a));
 function calcul(a)
 {
   var contor=0;
   var i;
   for(i=0;i<a.length;i++)
     if(a[i]>=0)contor++;
   return contor;
 }
</script>
</body>
</html>
```

3. Probleme rezolvate

Problema 1

Se citeşte un număr natural N. Se citesc N numere întregi. Să se calculeze şi afişeze maximul dintre ele.

Soluție

```
<!DOCTYPE html>
<html>
<body>
<h2>Maximul dintre N numere</h2>
<script>
 var N=Number(prompt("N="));
 var nr=Number(prompt("nr="));
 var max=nr;
```

```
 var i;
 for(i=2;i<=N;i++){
  nr=Number(prompt("nr="));
  if(nr>max)max=nr;
 }
 alert("maxim = "+max);
</script>
</body>
</html>
```

Problema 2

Se citeşte un număr natural N. Se citesc N numere întregi. Să se calculeze şi afişeze câte numere pare s-au introdus.

Soluţie

```
<!DOCTYPE html>
<html>
<body>
<h2>Cate numere pare sunt dintre N numere</h2>
<p>Presupunem ca se introduc corect, doar numere naturale!</p>
<script>
 var N=Number(prompt("N="));
 var contor=0;
 var i;
 for(i=1;i<=N;i++){
  nr=Number(prompt("nr="));
  if(nr%2==0)contor++;
 }
 alert("contor numere pare = "+contor);
</script>
</body>
</html>
```

Problema 3

Se citeşte un număr natural. Să se calculeze şi afişeze divizorii acestui număr.

Soluţie

```
<!DOCTYPE html>
<html>
```

```
<body>
<h2>Afisare divizori numar</h2>
<p>Presupunem ca numarul se introduce corect!</p>
<script>
 var nr=Number(prompt("nr="));
 var sirDivizori="";
 var i;
 for(i=1;i<=nr;i++)
   if(nr%i==0)sirDivizori=sirDivizori+i+" ";
 alert("Divizorii lui "+nr+" sunt:\n"+
       sirDivizori);
</script>
</body>
</html>
```

Problema 4

Se citesc de la tastatură două numere întregi. Să se calculeze cel mai mare divizor comun al lor şi să se afişeze rezultatul. Se va scrie o funcție separată *cmmdc* .

Soluție

```
<!DOCTYPE html>
<html>
<body>
<h2>CMMDC a doua numere</h2>
<p>Presupunem ca numarele se introduc corect!</p>
<script>
 var a=Number(prompt("a="));
 var b=Number(prompt("b="));
 alert("cmmdc = "+cmmdc(a,b));
 function cmmdc(x,y)
 {
   while(x!=y)
     if(x>y)x=x-y;
     else y=y-x;
   return x;
 }
</script>
</body>
</html>
```

Problema 5

Se citeşte un număr natural N. Să se calculeze termenul de rang N din şirul lui Fibonacci. Se va scrie o funcție **recursivă** care returnează termenul de rang N din şirul Fibonacci.

Apoi, se va scrie şi varianta nerecursivă a acestei funcții şi se vor compara cele două implementări (recursivă şi nerecursivă) pentru N=42 şi N=50.

Soluție

```
<!DOCTYPE html>
<html>
<body>
<h2>Calcul termen din sirul lui Fibonacci</h2>
<script>
 var N=Number(prompt("N="));
 alert("termenul de rang "+N+" = "+fib(N));
 function fib(N)
 {
   //implementare recursiva.
   if(N==0)return 1;
   if(N==1)return 1;
   return fib(N-1)+fib(N-2);
 }
</script>
</body>
</html>

//Varianta nerecursiva:
function fib(N)
 {
   if(N==0)return 1;
   if(N==1)return 1;
   var ultim=1;
   var penultim=1;
   var i;
   var termenCrt;
   for(i=2;i<=N;i++){
     termenCrt=ultim+penultim;
     penultim=ultim;
```

```
    ultim=termenCrt;
   }
   return termenCrt;
  }
```

Problema 6

Să se afişeze maximul dintr-un vector iniţializat cu cîteva valori întregi. Se va scrie o funcţie separată ce are ca parametru vectorul şi care returnează maximul din vector.

Soluţie

```
<!DOCTYPE html>
<html>
<body>

<h2>Exemplu de array</h2>
<script>
 var a=[1,5,-2,-7,10,8];
 var max=maxim(a);
 alert("maxim = "+maxim(a));
 function maxim(a)
 {
   var max=a[0];
   var i;
   for(i=1;i<a.length;i++)
     if(a[i]>max)max=a[i];
   return max;
 }
</script>
</body>
</html>
```

Problema 7

Fie un vector iniţializat cu câteva valori numerice. Mai adăugăm la sfârşitul lui câteva valori folosind metoda *push*. Să afişăm valorile din vector, separate prin spaţii, într-o casetă *alert.*

Soluţie

```
<!DOCTYPE html>
<html>
```

```
<body>
<h2>Exemplu de array</h2>
<script>
 var a=[1,-1,2,-2];
 var s;
 s=afisare(a);
 alert(s);
 a.push(7);
 a.push(100);
 s=afisare(a);
 alert(s);
 function afisare(a)
 {
  var s="";
  var i;
  for(i=0;i<a.length;i++)
   s=s+a[i]+" ";
  return s;
 }
</script>
</body>
</html>
```

Problema 8

Citim un număr natural N. Citim N numere de la tastatură, folosind metoda *prompt* . Să se afişeze dacă cele N numere sunt diferite între ele sau nu.

Soluţie

```
<!DOCTYPE html>
<html>
<body>
<h2>Sunt N numere diferite ?</h2>
<script>
 var N=Number(prompt("N="));
 var a=[];//nu merge var a[];
 var i;
 for(i=0;i<N;i++){
  var nr=Number(prompt("nr="));
  a.push(nr);
```

```
 }
 if(suntDiferite(a))alert("Sunt toate diferite.");
 else alert("Nu sunt toate diferite.");
 function suntDiferite(a)
 {
  var i,j;
  for(i=0;i<a.length-1;i++)
    for(j=i+1;j<a.length;j++)
      if(a[i]==a[j])return false;
  return true;
 }
</script>
</body>
</html>
```

4. Probleme propuse

Problema 1

Se citesc trei numere naturale folosind metoda *prompt* . Afişaţi la consolă câte numere pare au fost introduse.

Problema 2

Se citesc trei numere folosind metoda *prompt* . Afişaţi la consolă maximul dintre cele 3 numere.

Problema 3

Se citeşte un număr natural. Să se afişeze câţi divizori are.

Problema 4

Se citeşte un număr natural N. Se citesc N numere. Să se afişeze suma lor.

Problema 5

Se citeşte un număr natural N. Se citesc N numere. Să se afişeze pentru fiecare număr, dacă este posibil, valoarea radicalului.
(Pentru a putea extrage radicalul trebuie ca numărul citit de la tastatură să fie un număr pozitiv).

Problema 6

Se citesc două numere naturale. Să se afişeze dacă sunt prime între ele sau nu. Se va scrie o funcție separată ce returnează *true* dacă cele două numere (argumentele funcției) sunt prime între ele.

Problema 7

Se citesc N numere naturale. Să se calculeze şi afişeze câte numere prime s-au introdus. Se va scrie o funcție separată pentru număr prim.

Problema 8

Se citesc N numere naturale. Să se calculeze şi afişeze cel mai mare divizor comun al lor.

Problema 9

Se citeşte un număr natural. Să se afişeze dacă este termen în şirul lui Fibonacci.

Problema 10

Se citeşte un număr natural N. Se citesc N valori numerice într-un vector. Se citeşte un număr x. Să se afişeze de câte ori apare x în vector.

Problema 11

Se citeşte un număr natural N. Se citesc N valori numerice într-un vector. Să se afişeze dacă cele N valori sunt în ordine crescătoare.

Problema 12

Se citeşte un număr natural N. Se citesc N valori numerice naturale într-un vector. Să se afişeze câte numere prime sunt în vector.

Problema 13

Se citeşte un număr natural na. Se citesc na valori numerice diferite între ele, în vectorul a. Se citeşte un număr natural nb. Se citesc nb valori numerice diferite între ele, în vectorul b. Să se calculeze vectorul c, intersecția mulțimilor a şi b. Să se afişeze acest vector.

Problema 14

Se citeşte un număr natural na. Se citesc na valori numerice diferite între ele, în vectorul a. Se citeşte un număr natural nb. Se citesc nb valori numerice diferite între ele, în vectorul b. Să se calculeze vectorul c, reuniunea mulțimilor a şi b. Să se afişeze acest vector.

Problema 15

Se citeşte un număr natural N. Să se copieze primii N termeni ai şirului lui Fibonacci într-un vector şi să se afişeze acest vector.

Problema 16

Se citeşte un număr natural N. Se citesc N numere într-un vector a. Se citeşte un număr x. Să se afişeze dacă numărul x este prezent în vector sau nu. Se va scrie o funcție separată ce returnează *true* dacă x este în vector. În această funcție se va utiliza metoda *indexOf* .

Problema 17

Se citeşte un număr natural N. Se citesc N numere într-un vector a. Să se calculeze ante-maximul din vector (al doilea număr ca mărime din vector, după maxim). Exemplu:
a=[-1,-1,10,5,7,6] , ante-maximul este 7 (maximul este 10).

După citirea vectorului, pentru a afla ante-maximul, se va face o singură parcurgere a vectorului.

Indicație:

Se va inițializa maximul cu maximul dintre primele două numere din vector şi ante-maximul cu minimul dintre primele două numere din vector.