JavaScript

CUPRINS

Conversația 1

JavaScript şi HTML 4.0, XHTML, XML

În această conversație:

Stilul de viață Web

Oamenii şi companiile de pretutindeni se bazează din ce în ce mai mult pe Web pentru a-şi rezolva problemele zilnice. Disponibilitatea crescută şi varietatea funcţiilor şi serviciilor online au dus la înfiinţarea „*stilului de viaţă Web*", iar Web-ul devine încet, încet o parte integrantă a vieţii noastre personale şi profesionale.

Natura *interactivă* a World Web face posibilă existenţa „stilului de viaţă Web".

Cum pot fi făcute paginile Web interactive?

În principal, există trei niveluri de interactivitate:

- ✓ Elementele hiperlink – Cel mai simplu nivel de interactivitate pe care îl oferă Web-ul.
- ✓ Interactivitatea funcţională – Se referă la abilitatea de a executa sarcini specifice şi funcţii într-un site Web.
- ✓ Interactivitatea comunitară – Permite vizitatorilor să se întâlnească şi să interacţioneze unul cu altul. Interactivitatea comunitară este adesea considerată o strategie promoţională online.

Interactivitatea vă obligă să apelaţi la un limbaj de programare sau un limbaj de script.

Limbajele de script destinate Web-ului permit combinarea script-urilor şi a instrucţiunilor (X)HTML pentru a crea pagini Web interactive.

Care este deosebirea între un limbaj de script şi un limbaj de programare?

Totul depinde de cel căruia vă adresaţi: noi vorbim în această carte de limbajul de script JavaScript, dar nimeni nu vă împiedică să menţionaţi în CV-ul dumneavoastră, în lista de limbaje pe care le cunoaşteţi şi limbajul de programare JavaScript, evident după ce aţi terminat de citit această carte!

JavaScript nu este singurul limbaj pentru Web.

`Java`, un limbaj de programare orientat obiect creat de Sun, de exemplu, în anumite situaţii este mult mai bun decât JavaScript. Chiar dacă numele lor şi câteva instrucţiuni se aseamănă, Java şi JavaScript sunt limbaje total diferite.

`VB Script` (`V`isual `B`asic `S`cripting `E`dition) este răspunsul lui Microsoft la JavaScript. Ca şi JavaScript, VB Script este un limbaj de script simplu. El are la bază sintaxa `Visual Basic`, un limbaj de programare Windows foarte răspândit. Script-urile VB Script pot fi direct incluse în documentele (X)HTML.

VB Script prezintă mari avantaje pentru cei care cunosc deja Visual Basic.

VB Script poate fi de asemenea integrat cu `Active X`, standardul Microsoft pentru integrarea aplicaţiilor în paginile Web.

De ce JavaScript şi Java au nume atât de apropiate?

Nu este nimic grav dacă nu ştiţi să răspundeţi la această întrebare! Cu toţii ştim că Java este numele unei insule din Indonezia, dar … JavaScript? JavaScript şi Java sunt două limbaje pentru Web. De ce numele lor sunt atât de apropiate?

Cel puţin trei ar putea fi răspunsurile la această întrebare, pe care vă invităm să le analizaţi şi apoi să decideţi!

> **1.** JavaScript este o versiune „simplificată" a limbajului Java.
> **2.** Sintaxa limbajului JavaScript este parţial inspirată din cea a limbajului Java.
> **3.** Cele două limbaje provin din insula Java.

Răspunsul corect este **2:** `Sintaxa limbajului JavaScript este parţial inspirată din cea a limbajului Java.` Să mă explic.

În ciuda tuturor aparenţelor, JavaScript şi Java sunt diferite. Din cauza asemănării de nume numeroşi sunt cei care cred că JavaScript nu este decât o versiune „simplificată" a limbajului Java (**1**). Aceasta este fals. Java şi JavaScript sunt două tehnologii separate având doar numele şi originea apropiate.

Sintaxa limbajului JavaScript este parţial inspirată din cea a limbajului Java (2).

Diferenţa esenţială între cele două limbaje este aceea că spre deosebire de JavaScript, Java permite crearea de aplicaţii autonome.

Remarcă. Java este numele unei insule din Indonezia, dar el înseamnă de asemenea „cafea" în vorbirea curentă americană. Logo-ul limbajului de programare Java este o ceaşcă de cafea caldă (un detaliu amuzant, nu-i aşa?).

Ce puteți realiza cu JavaScript?

JavaScript este limbajul favorit al creatorilor de site-uri Web deoarece:

> ✓ este un limbaj uşor de învăţat;

- ✓ sunt suficiente câteva linii de cod pentru a mări gradul de interactivitate al paginilor Web;
- ✓ este un limbaj care poate fi interpretat de browser-e;
- ✓ programele JavaScript pot fi incluse direct în documentele (X)HTML.

Limbajul JavaScript poate servi la:

- ✓ generarea paginilor Web personalizate şi modificarea dinamică a prezentării lor;
- ✓ realizarea calculelor matematice;
- ✓ validarea conţinutului unui formular;
- ✓ comunicarea cu applet-urile Java;
- ✓ crearea animaţiilor personalizate;
- ✓ afişarea unor mesaje care defilează în bara de stare a navigatorului;
- ✓ afişarea unor mesaje într-o pagină Web sau într-o casetă de dialog;
- ✓ crearea unor butoane animate;
- ✓ identificarea navigatorului în care se afişează pagina Web;
- ✓ executarea funcţiilor clasice ale unui limbaj de programare.

Evident, această listă nu este exhaustivă; mai există numeroase aplicaţii posibile în JavaScript. Multe dintre acestea le vom descoperi împreună!

Remarcă. Utilizarea limbajului JavaScript nu este limitată doar la navigatoarele Web. Există versiuni server JavaScript (de exemplu, Live Wise sau Server – Side JavaScript pentru navigatorul Netscape) care permit scrierea de programe JavaScript capabile de a executa, de exemplu accesul la o bază de date în vederea afişării informaţiilor într-o pagină Web.

Ce nu puteți realiza cu JavaScript?

Desigur, JavaScript nu este perfect. Cine este perfect! Deşi JavaScript este puternic, el este limitat de restricţii severe impuse de navigatoarele Web, după cum urmează.

Din motive de securitate,

- ✓ JavaScript nu poate citi, scrie, crea şi şterge fişiere de pe hard disc;
- ✓ JavaScript nu poate executa operaţii în reţea;

- ✓ JavaScript nu poate crea aplicaţii autonome. Pentru a scrie astfel de aplicaţii va trebui să utilizaţi unul din limbajele clasice: Java, C sau C++.

Terminologia JavaScript

Pentru a lucra cu limbajul JavaScript, va trebui să vă familiarizaţi cu termenii şi conceptele prezentate în cele ce urmează: `Obiect; Proprietate; Metodă; Instrucţiune; Funcţie; Eveniment; Gestionar de evenimente; Variabilă`.

Să facem cunoştinţă cu fiecare, înarmându-vă cu răbdare şi nu doar atât!

Obiect

Desigur dumneavoastră ştiţi foarte bine ce este un obiect cel puţin din viaţa cotidiană.

Este o entitate cu parte întreagă, ca de exemplu o minge de volei, o maşină etc. În JavaScript, `window` este un obiect, `window` este o fereastră a navigatorului. Pagina Web este de asemenea un obiect. Fiecare element al unui document (X)HTML este la rândul lui un obiect: paragrafe, formulare, tablouri, imagini, link-uri etc. Exemplele pot continua.

În JavaScript există trei tipuri de obiecte:

1. **obiecte interne** furnizate de limbajul JavaScript (vezi Conversaţiile 2, 4, 5): `Arguments, Array, Boolean, Date, Function, Math, Number, Object, RegExp, String, This`.
2. **obiecte personalizate**, create de utilizator în funcţie de cerinţele script-urilor (vezi Conversaţia 12);
3. **obiectele navigatorului** (vezi Conversaţiile 7, 8, 9, 10) sunt exterioare limbajului dar sunt recunoscute de browser-e. Reprezintă diferite componente ale navigatorului şi ale documentului (X)HTML curent.

- Standardul DOM (*Document Object Model*) defineşte obiectele navigatorului în mod ierarhic, după cum urmează: `Window`, `Document`, `Navigator`, `Event`, `Screen`, `History` şi `Location`.
- Lista obiectelor navigatorului (în ordine alfabetică) este următoarea: `button`, `checkbox`, `document`, `event`, `fileUpload`, `form`, `hidden`, `history`, `location`, `MimeType`, `navigator`, `Objects (în general)`, `option`, `Option()`, `password`, `plugins`, `radio`, `reset`, `screen`, `select`, `submit`, `text`, `textarea`, `window`.

Remarcă. În JavaScript obiectele sunt considerate ca „*substantive*".

Proprietate

O proprietate descrie un obiect.

Culoarea reprezintă unul din atributele (proprietăţile) obiectului maşină. Poate avea valoarea „bleu".

În JavaScript `height` este o proprietate a obiectului `window`. De exemplu, ea poate avea ca valoare 200.

O proprietate poate fi de asemenea un obiect care la rândul lui dispune de proprietăţi. De exemplu, farurile sunt proprietăţi ale obiectului maşină. Dar acestea sunt de asemenea obiecte care pot avea diverse proprietăţi: formă, culoare, putere.

În JavaScript, `document` este o proprietate a obiectului `Window` şi este în egală măsură un obiect care are de asemenea proprietăţi.

În figura 1.1 sunt prezentate câteva exemple de proprietăţi ale unor obiecte interne JavaScript (vezi Conversaţiile 2 şi 4).

Denumire obiect intern	***Proprietăţi***
`Array`	`length`
`Math`	`E, LN10, LN2, LOG10E, LOG2E, PI, SQRT1_2, SQRT`
`String`	`length`

Figura 1.1

În figura 1.2 sunt prezentate câteva exemple de proprietăţi ale unor obiecte ale navigatorului (Conversaţia 7).

Denumire obiect navigator	***Proprietăţi***
`Button`	`align, defaultValue, disabled, form, name, size, type, value`
`Checkbox`	`align, checked, defaultchecked, defaultValue, disabled, form, name, size, status, type, value, width`
`Form`	`acceptcharset, action, elements[], encoding, enctype, length, method, name, target`

Figura 1.2

Remarcă. În JavaScript proprietăţile sunt considerate ca „*adjective*".

Metodă

O metodă este o funcţie care defineşte un anumit comportament caracteristic al unui obiect.

Metodele disponibile pentru fiecare obiect descriu ceea ce puteţi face cu acest obiect.

Metoda `deschide` a obiectului `maşina` are ca funcţie deschiderea uşilor.

În JavaScript, metoda `close()` a obiectului `window` are ca funcţie închiderea ferestrei.

Fiecare obiect posedă o colecţie de metode, iar fiecare metodă aparţine cel puţin unui obiect.

În figura 1.3 sunt prezentate câteva exemple de metode ale unor obiecte interne (vezi Conversaţiile 2 şi 4).

Denumire obiect intern	***Metode***
`Array`	`concat(), join(), pop(), push(), reverse(), shift(), slice(), sort(), splice(), toString(), unShift()`
`Math`	`abs(), acos(), asn(), atan(), ceil(), cos(), exp(), floar(), log(), max(), min(), pow(), random(), round(), sin(), sqrt(), tan()`

Figura 1.3

În figura 1.4 sunt prezentate câteva exemple de metode ale unor obiecte ale navigatorului (vezi Conversaţia 7).

Denumire obiect navigator	***Metode***
`Button`	`blur(), click(), focus()`
`Checkbox`	`blur(), click(), focus()`

Figura 1.4

Remarcă. În JavaScript metodele sunt considerate ca „*verbe*".

Instrucţiune

O instrucţiune produce o acţiune (afişarea unui text, modificarea proprietăţii unui obiect, atribuirea unei variabile a unui nou conţinut). Instrucţiunile combină obiecte, proprietăţi şi metode (substantive, adjective şi verbe).

În figura 1.5 este prezentat un exemplu de script care conţine instrucţiuni simple JavaScript.

```
<script language = ”JavaScript” type = ”text/JavaScript”>
        a=4;
        b=13;
        c=a+b;
        document.write (”Suma: ” +c);
</script>
```

Figura 1.5

Remarcă. Instrucţiunile limbajului JavaScript sunt prezentate în Conversaţia 3.

Funcţie

O funcţie este un grup de instrucţiuni tratate ca o singură entitate. Sau, o funcţie este un obiect care conţine cod executabil. Spre deosebire de metode, funcţiile nu sunt asociate nici unui obiect particular. Funcţiile pot fi comparate cu „electronii liberi" care pot fi reutilizaţi în orice circumstanţă. Numeroase funcţii sunt interne: ele sunt incluse în limbajul JavaScript (*Exemple:* `eval(); parseFloat(); parseInt()`).

Nu trebuie să vă surprindă următoarea afirmaţie: *chiar dumneavoastră puteţi să vă creaţi propriile funcţii!*

Ceea ce este nemaipomenit!

În figura 1.6 se prezintă un exemplu de funcţie definită de utilizator: calculul diametrului unui cerc de rază r.

```
<script language = ”JavaScript”>
        function diametru (r)
        {
                temp = 2*r;
                return temp;
        }
</script>
```

Figura 1.6

Remarcă. Funcţiile sunt tratate în detaliu în Conversaţia 2.

Evenimente

Un eveniment este ceva care se întâmplă.

În sens larg, informaţiile pe care le notaţi în agenda dumneavoastră („Oră la dentist”, „Aniversarea lui Droopy”, „Ziua şefului”) sunt `evenimente`.

Diferite evenimente pot fi imprevizibile: contravenţie pentru depăşire viteza legală în localitate; o nuntă cu … dar la rudele nevestii; vizita neaşteptată a soacrei etc.

Dacă aceste evenimente sunt prevăzute sau nu, este problema dumneavoastră de a le gestiona. Gestiunea evenimentului „Aniversarea lui Droopy” va consta probabil în a-i cumpăra un cadou. Cât priveşte vizita soacrei reacţia dumneavoastră poate fi una normală: *stingeţi lumina pentru a demonstra că nu sunteţi acasă!*

Un eveniment JavaScript este o acţiune care se produce în raport cu un element (fereastră, document, un buton).

În JavaScript numeroase evenimente (clic, dublu clic etc.) sunt provocate de utilizator. Alte evenimente nu privesc direct utilizatorul.

Iată o listă cu cele mai frecvente evenimente JavaScript:

- ✓ utilizatorul încarcă o pagină Web în browser;
- ✓ utilizatorul opreşte încărcarea paginii Web în browser;
- ✓ utilizatorul a introdus sau a modificat conţinutul unui câmp de text al unui formular;
- ✓ utilizatorul a executat clic pe o imagine sau pe un buton;
- ✓ utilizatorul a expediat conţinutul unui formular sau l-a reiniţializat.

Fiecare eveniment este asociat unui obiect.

Fiecare eveniment are propriul său nume.

De exemplu, evenimentul `MouseOver` se produce atunci când mouse-ul se poziţionează deasupra unui obiect. Dacă mouse-ul trece pe deasupra unui link (legătură), evenimentul `MouseOver` este expediat gestionarului de evenimente al acestei legături, dacă există.

În figura 1.7 sunt prezentate câteva exemple de evenimente JavaScript dintre cele mai cunoscute.

Denumire eveniment	***Se declanşează atunci când ...***
`click`	`se execută clic pe un element`
`dblclick`	`se execută dublu clic pe un element`
`help`	`se activează tasta F1 în fereastra activă`
`keypress`	`se acţionează o tastă a claviaturii`

Figura 1.7

Gestionarii de evenimente

Gestionarii de evenimente JavaScript indică navigatorului cum să reacţioneze atunci când se produc diferite evenimente.

În general, un gestionar de evenimente ia forma unei funcţii JavaScript pe care o creaţi special pentru a prelucra un anume tip de eveniment, dar el poate fi totodată o instrucţiune sau mai multe instrucţiuni JavaScript, care sunt integrate în codul (X)HTML de definire al evenimentului.

Pentru a defini un gestionar de evenimente, adăugaţi `on` la începutul numelui evenimentului. Astfel, gestionarul de evenimente `onMouseOver` este apelat atunci când mouse-ul trece pe deasupra legăturii (Gestionarul de evenimente este plasat în tag-ul (X)HTML `<a>` al link-ului corespunzător).

Remarci:

- ✓ Poate aţi observat combinaţia de majuscule şi minuscule (*exemplu:* `onMouseOver`, `onMouseOut`). Este vorba de notaţia standard a gestionarilor de evenimente: `on` este scris întotdeauna cu minuscule, iar iniţiala fiecăruia din cuvintele evenimentului, cu majuscule.
- ✓ Evenimentele şi gestionarii de evenimente JavaScript sunt tratate în detaliu în Conversaţia 6.

Variabile

Variabilele reprezintă un element fundamental al limbajului JavaScript. Ele pot conţine: un număr, un şir de caractere sau un obiect.

Fiecare variabilă poartă un nume care respectă anumite reguli de sintaxă.

În figura 1.8 se prezintă câteva exemple de nume de variabile JavaScript (valide).

```
nume_prof
nume_student_olimpic
a
_var13
stoc
```

Figura 1.8

Remarcă. Variabilele JavaScript sunt tratate în detaliu în Conversaţia 2.

Conceptele programării JavaScript

Utilizarea limbajului JavaScript se reduce în principal la două concepte de bază:

- ✓ sintaxa JavaScript;
- ✓ DOM-ul (Document Object Model – modelul obiectelor documentului).

Sintaxa defineşte un ansamblu de reguli care trebuie respectate atunci când scrieţi cod JavaScript. Aceste reguli nu sunt numeroase. Este bine să le cunoaşteţi ca apoi să le aplicaţi. Nu ştiu cum este mai bine: să staţi cu regulile în cap sau cu capul în reguli! Procedaţi cum credeţi!

DOM-ul se referă la componentele paginii Web, `obiectele` pe care le puteţi accesa şi pe care le puteţi manipula cu ajutorul limbajului JavaScript.

Remarcă. Obiectele interne, obiectele navigatorului şi obiectele personalizate sunt tratate în detaliu, în Conversaţia 4, Conversaţia 5, Conversaţia 7, Conversaţia 12.

Trebuie să vă însuşiţi foarte bine DOM-ul JavaScript pentru a putea scrie un script JavaScript.

Cum inserați un script într-un document (X)HTML?

Extinderea paginilor (X)HTML prin folosirea limbajului JavaScript conferă paginilor Web mai multă putere, iar (X)HTML-ului mai multă flexibilitate. JavaScript, prin inserarea unui script în documentele (X)HTML existent permite programatorilor să creeze pagini Web mult mai dinamice.

Elementul script

Script-urile JavaScript sunt inserate într-un document (X)HTML cu elementul `<script> ... </script>`.

Elementul `script` este alcătuit din: tag-ul de început `<script>`, conţinutul propriu-zis şi tag-ul de sfârşit `</script>`.

Tag-ul `<script>` conţine următoarele atribute:

- ✓ `type` – indică tipul limbajului de script. Valoarea sa trebuie să fie un `Internet Media Type` (un tip `MIME`), de exemplu `text/JavaScript` sau `text/vbscript` (figura 1.9).

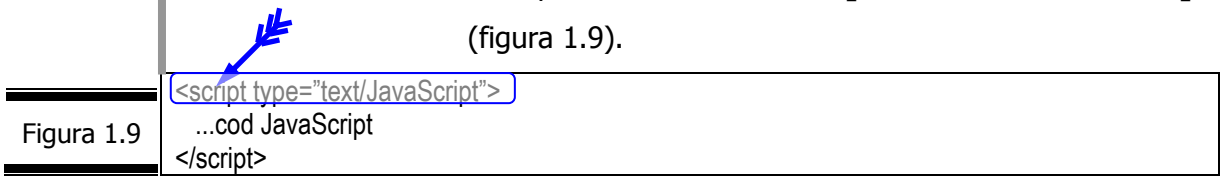


Figura 1.9

- ✓ `language` – identifică limbajul de script şi facultativ versiunea (vezi figura 1.10).

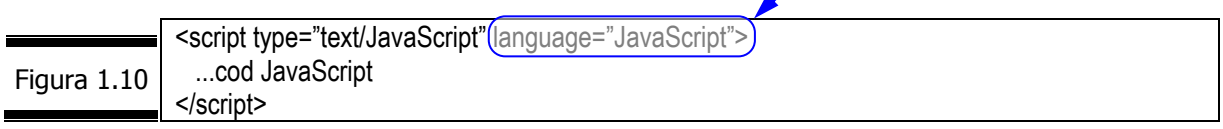


Figura 1.10

- ✓ `src` – facultativ, precizează `URL`-ul fişierului extern care conţine script-ul (vezi figura 1.11).

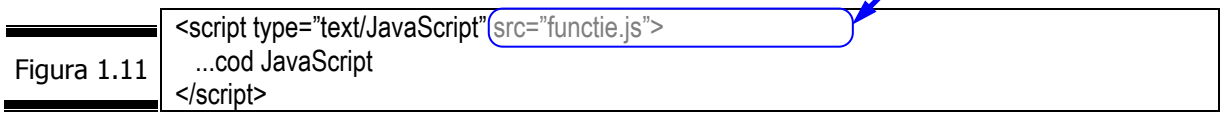


Figura 1.11

Elementul noscript

Incapacitatea navigatorului de a prelucra un script poate avea trei cauze: fie el nu ştie să prelucreze script-urile în general; fie el nu recunoaşte limbajul desemnat prin atributul `type`; fie utilizatorul a dezactivat (nu are încredere!) prelucrarea script-urilor. Din acest motiv se utilizează elementul `<noscript> ... </noscript>` în interiorul căruia puteţi plasa elementele (X)HTML care vor fi ignorate de navigatoarele capabile să exploateze un script (vezi figura 1.12).

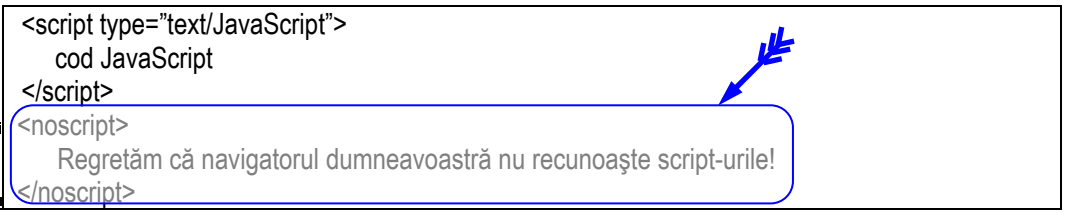


Figura 1.12

Metode de inserare a script-urilor într-un document (X)HTML

Rolul elementului `script` este de a defini script-ul care va fi utilizat în documentul (X)HTML.

Pentru inserarea script-urilor în documentele dumneavoastră (X)HTML folosiţi una din metodele prezentate mai jos:

- ✓ *Metoda 1* – plasaţi script-ul în corpul paginii (între `<body>` şi `</body>`);
- ✓ *Metoda 2* – plasaţi script-ul în antet-ul paginii (între `<head>` şi `</head>`);
- ✓ *Metoda 3* – utilizaţi fişiere sursă externe;
- ✓ *Metoda 4* – creaţi un gestionar de evenimente.

Cum inseraţi un script în documentele XML

Cu JavaScript puteţi crea şi manipula obiectele DOM XML. Pentru a construi pagini Web interactive cu XML folosiţi tehnicile specifice ale DOM-ului XML şi ale limbajului JavaScript.

În lucrarea Liviu Dumitra•cu, XML, Editura Universit••ii din Ploie•ti, în conversaţia 9 se prezintă în detaliu modul de inserare a script-urilor în documentele XML.

Consultaţi de asemenea următoarele resurse:

- ✓ Floarea Năstase, Pavel Năstase, Tehnologia aplicaţiilor Web (XML, DOM, ASP), Editura Economică 2002, Bucureşti
- ✓ http://www.dannyg.com/examples/xmltable/index.html

EXEMPLUL 1 JAVASCRIPT

Scrieți un script care afişează mesajul: „*Îmi place să fiu întotdeauna aşa cum sunt.*” Pentru inserarea script-ului într-un document (X)HTML utilizați toate cele patru metode pe care le-am prezentat anterior.

Metoda 1

Iată cum procedăm pentru a insera (cu Notepad) script-ul în corpul documentului (X)HTML. Mesajul va fi afişat cu italice.

Dacă doriți să inserați un script scurt, reprezentat prin câteva linii de cod JavaScript, cel mai simplu este să-l plasați în corpul paginii.

1. Creați documentul (X)HTML (figura 1.13) cu Notepad.

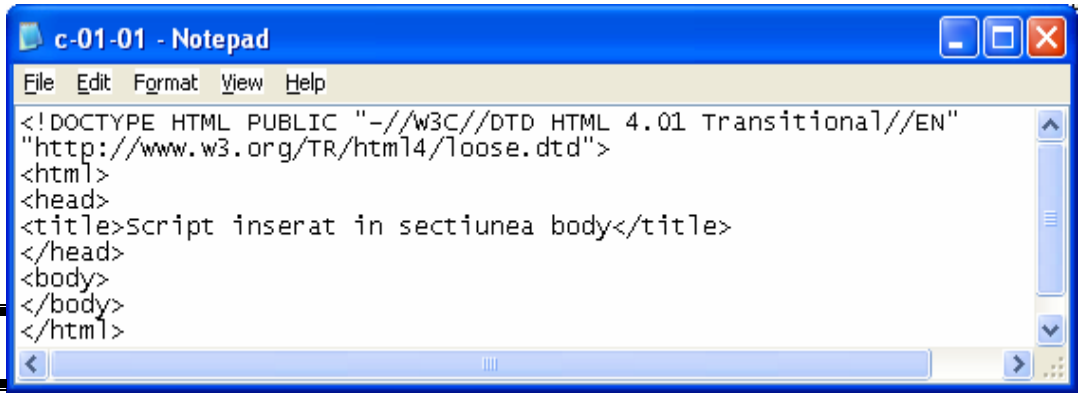


Figura 1.13

2. Plasați elementul script în locul în care doriți să apară script-ul dumneavoastră (figura 1.14).

Figura 1.14

3. Introduceți în tag-ul de deschidere `<script>` atributul `type` pentru a preciza tipul `MIME` al script-ului şi atributul `language` pentru a preciza limbajul de script (JavaScript), figura 1.15.

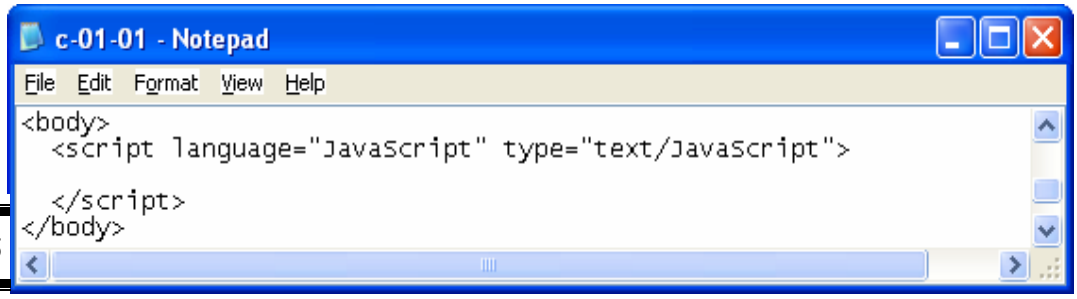


Figura 1.15

4. Inserați tag-ul de comentariu XHTML standard (`<!--` şi `-->` şi plasați „`//`" înaintea tag-ului de sfârşit de comentariu XHTML pentru a ascunde sfârşitul de comentariu de interpretorul JavaScript (figura 1.16).

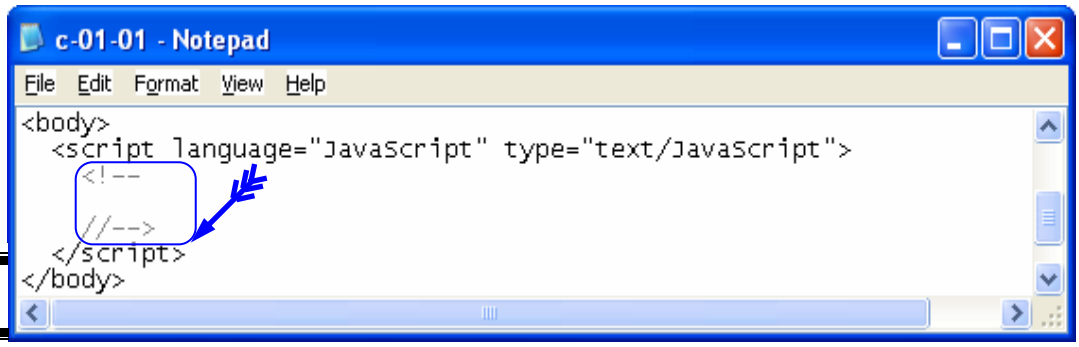


Figura 1.16

Remarcă. Liniile de comentariu au fost inserate pentru a ascunde script-ul de navigatoarele care nu-l recunosc.

5. Inserați instrucțiunea JavaScript `document.write()` care afişează mesajul: „`Îmi place s• fiu întotdeuna a•a cum sunt.`" în pagina Web (atunci când ea este încărcată!), figura 1.17.

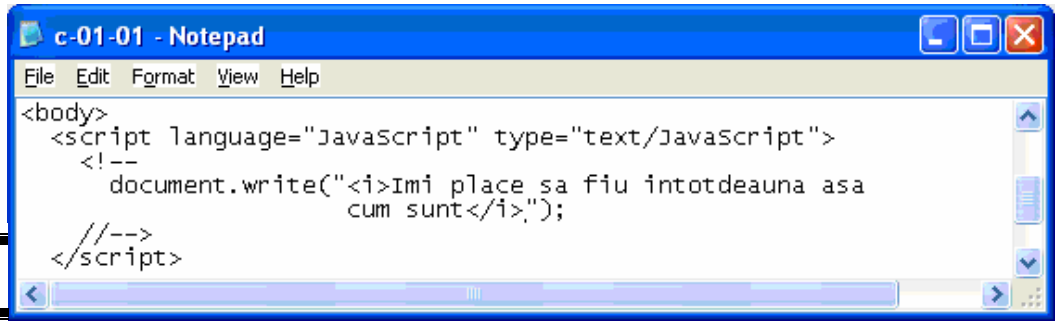


Figura 1.17

Remarci:

- ✓ JavaScript oferă mai multe soluții pentru afişarea informațiilor, una dintre cele mai simple fiind instrucțiunea `document.write`.
- ✓ Instrucțiunea `document.write` (vezi Conversația 7) permite afişarea unui text, a valorilor numerice şi a altor informații. În măsura în care programul dumneavoastră JavaScript va fi inserat într-o pagină Web, rezultatul va fi afişat direct în pagină.
- ✓ Tag-urile (X)HTML, `<i>` de exemplu, sunt plasate între ghilimele. Tag-urile (X)HTML nu sunt interzise în interiorul tag-urilor `<script>`. Plasate între ghilimele, tag-urile (X)HTML sunt interpretate de către navigator ca şi când ar aparține (X)HTML-ului.
- ✓ Facilitatea de combinare a tag-urilor (X)HTML şi a instrucțiunilor JavaScript reprezintă una din caracteristicile cele mai puternice ale unui navigator care recunoaşte JavaScript. În realitate, această facilitate constituie esența Web-ului dinamic şi interactiv pe care dumneavoastră urmează să-l aplicați.

6. Validați documentul XHTML 1.1 cu aplicația `validator` (vezi figura 1.18).

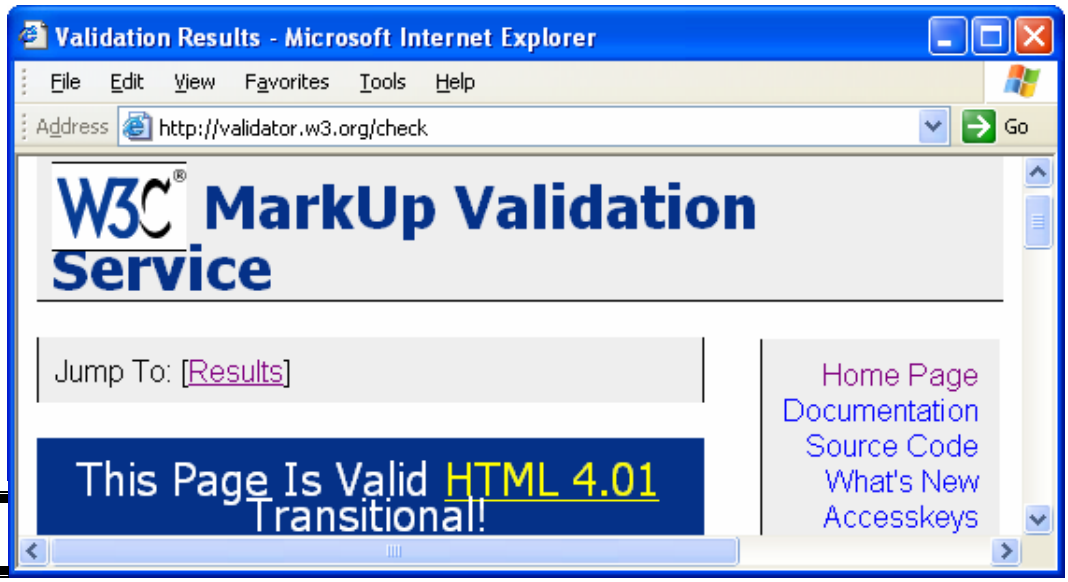


Figura 1.18

7. Inserați codul XHTML care afişează iconul de conformitate (figura 1.19).

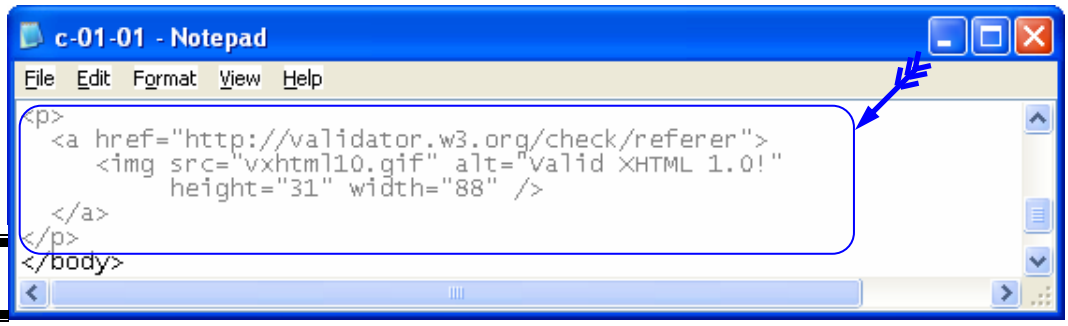


Figura 1.19

8. Afişaţi pagina Web într-un browser (figura 1.20).

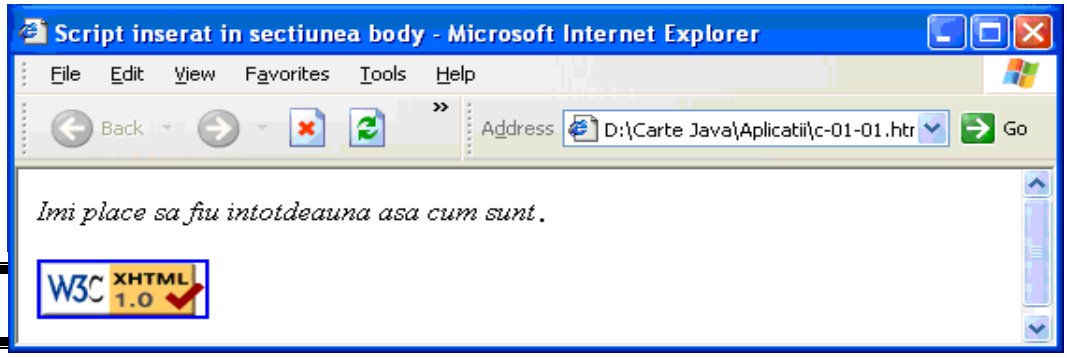


Figura 1.20

Procedura de creare automată a script-ului cu Dreamweaver MX

Iată cum procedăm pentru a crea documentul XHTML pentru EXEMPLUL 1 JAVASCRIPT cu Dreamweaver MX.

1. În grupul de panouri `Insert`, executați clic pe subpanoul `HTML` (figura 1.21).

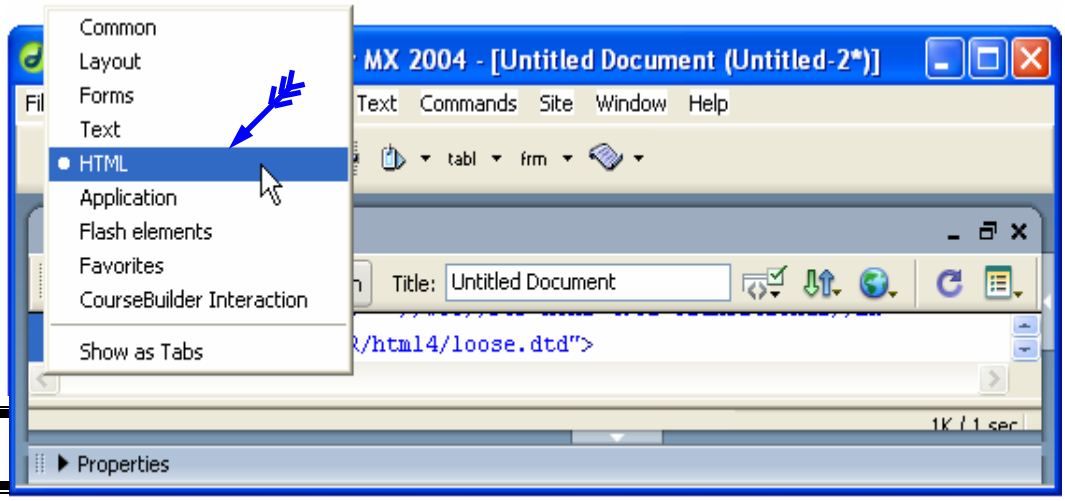


Figura 1.21

2. Executați clic în locul în care doriți să inserați script-ul (figura 1.22).

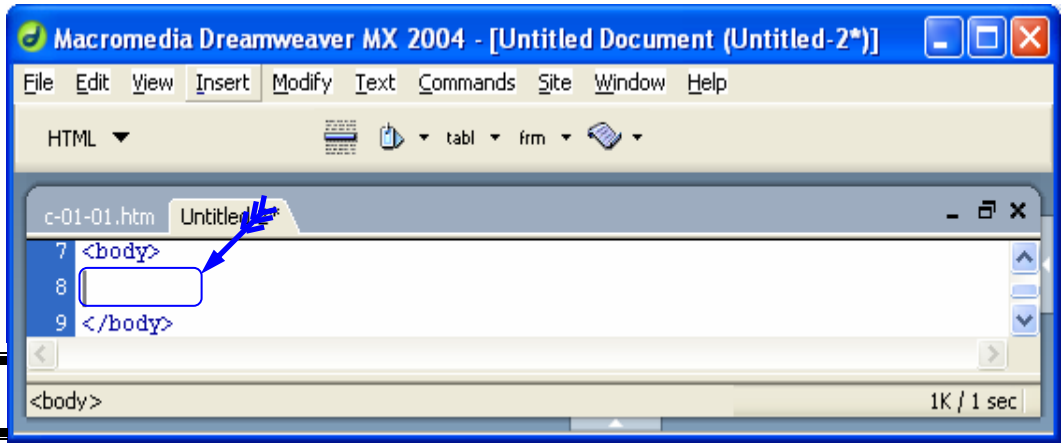


Figura 1.22

3. Executați clic pe butonul (Script), figura 1.23.

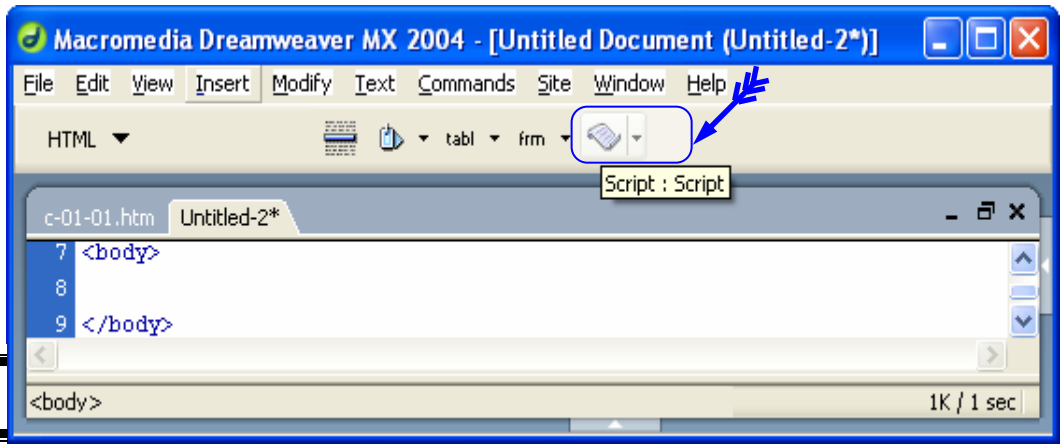


Figura 1.23

Remarcă. Se deschide caseta de dialog `Script` (figura 1.24).

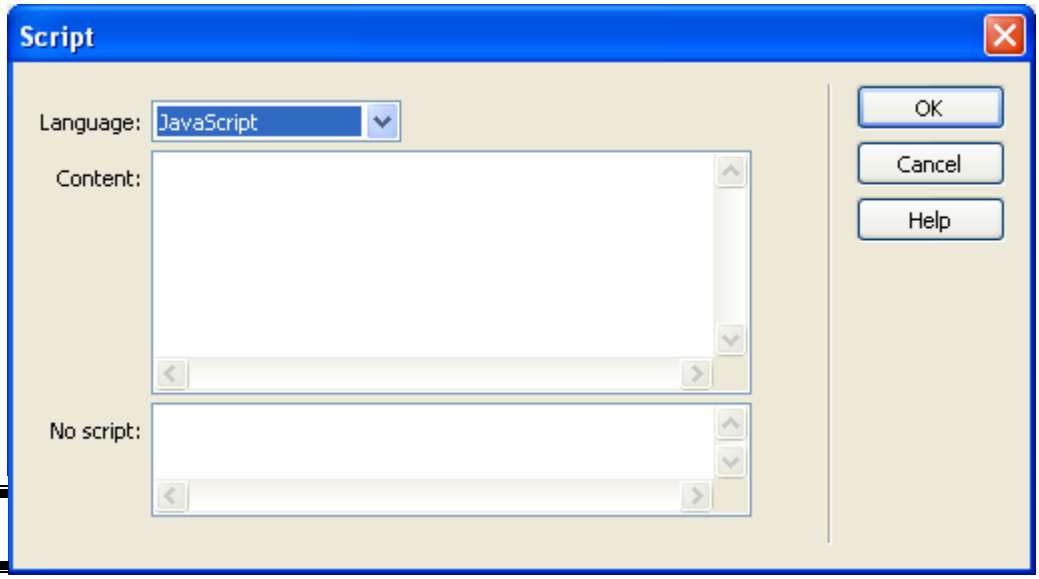


Figura 1.24

4. Introduceţi script-ul (figura 1.25).

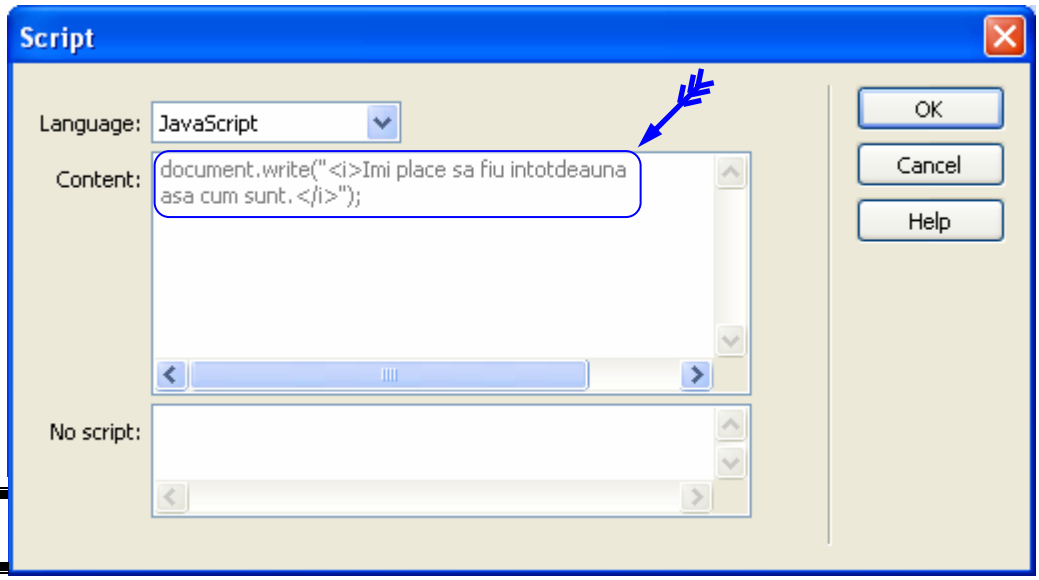


Figura 1.25

5. Executați clic pe butonul OK al ferestrei de dialog `Script` (figura 1.26).

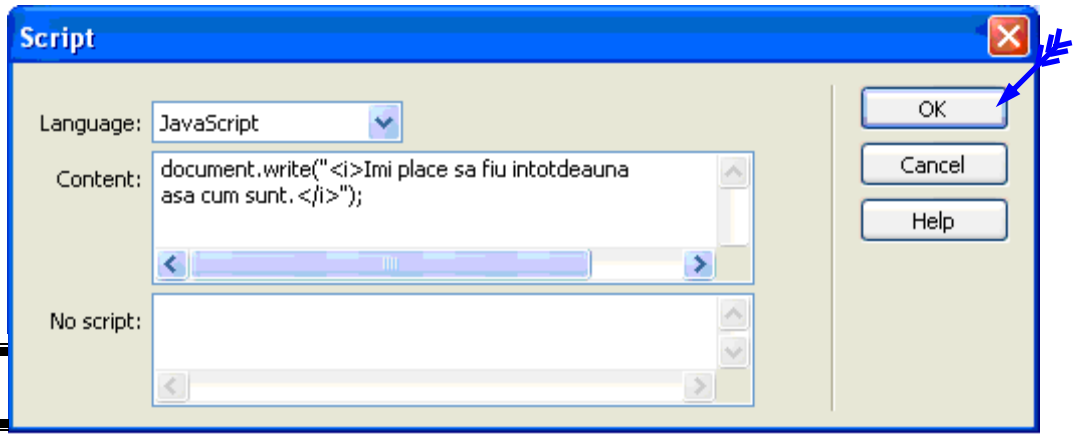


Figura 1.26

Remarcă. Se inserează elementul `<script> ... </script>` (figura 1.27).

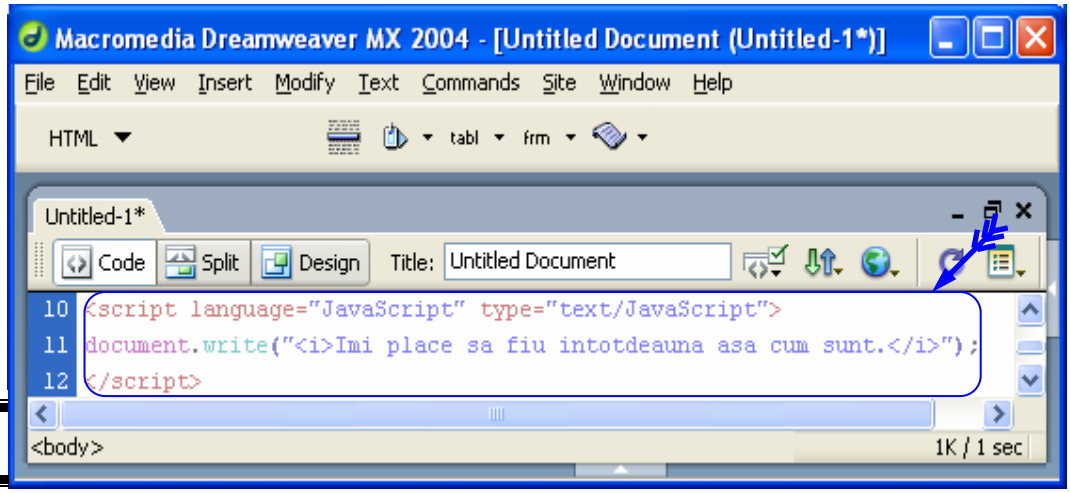


Figura 1.27

6. Vizualizați pagina Web într-un browser (figura 1.28).

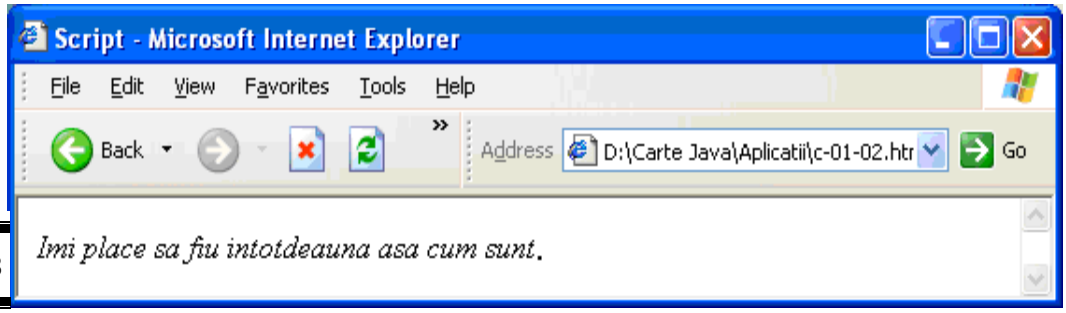


Figura 1.28

Metoda 2

Iată cum procedăm pentru a insera (cu Notepad) script-ul în antet-ul paginii, în secțiunea `<head> ... </head>`.

Secțiunea `<head>` a unei pagini (X)HTML precede secțiunea `<body>`. Tag-urile şi script-urile cuprinse în această secțiune sunt interpretate întotdeauna primele. Din acest motiv în mod frecvent se plasează în această secțiune un script care defineşte variabilele şi funcțiile ce urmează a fi utilizate de către alte script-uri ale paginii.

1. Creați documentul (X)HTML cu Notepad.
2. Plasați tag-urile `script` în elementul `head` al documentului (X)HTML (figura 1.29).

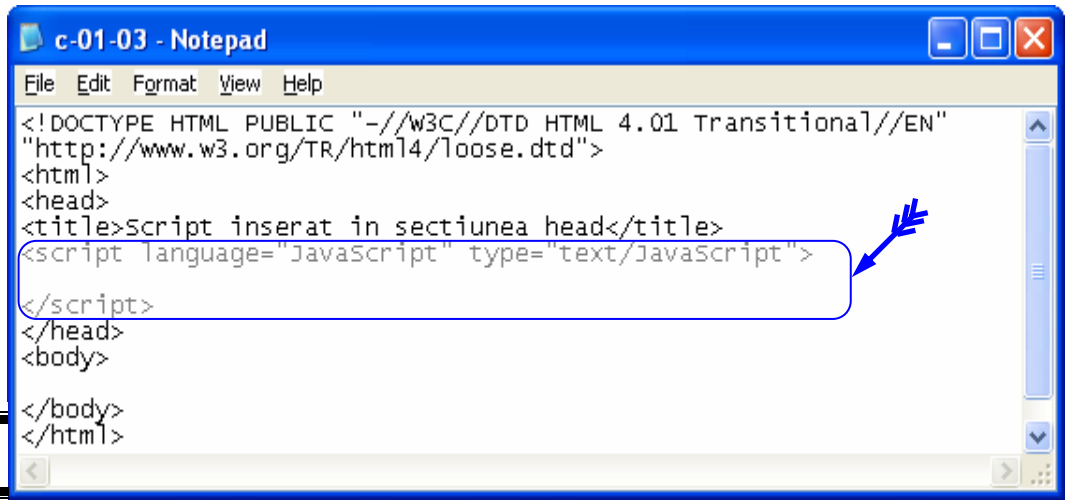


Figura 1.29

3. Introduceți funcția mesaj() (figura 1.30).

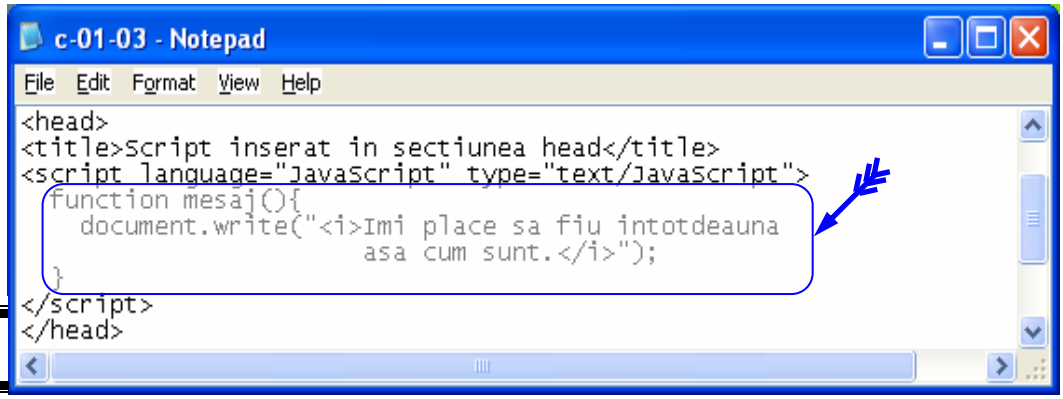


Figura 1.30

Remarci:

- ✓ O funcție este un grup de instrucțiuni tratate ca o singură entitate. Pentru a utiliza o funcție, va trebuie mai întâi să o definiți.
- ✓ Funcția `mesaj` are rolul de a afișa (cu italice) mesajul: „*Îmi place să fiu întotdeauna așa cum sunt.*” în pagina web.
- ✓ Funcția `mesaj` începe cu cuvântul rezervat `function` urmată de numele funcției (mesaj). După numele funcției urmează parantezele vide ”`()`”. După cum veți vedea mai târziu, aceste paranteze nu sunt întotdeauna vide! Acoladele de deschidere (”`{`”) și de închidere (”`}`”) servesc la delimitarea instrucțiunilor JavaScript care alcătuiesc corpul funcției și permit navigatorului să știe unde începe și unde se termină o funcție. În exemplul nostru, o singură linie de cod JavaScript apelează funcția `document.write` (o metodă a obiectului `Document`) care permite afișarea cu italice a mesajului *„Îmi place să fiu întotdeauna așa cum sunt.”*.

4. Apelați funcția `mesaj()`.

Funcția este acum definită și inserată în documentul (X)HTML, dar ... inutilizabilă în acest moment.

Pentru a putea utiliza o funcție, va trebui s-o apelați. Pentru a apela o funcție, va trebui să utilizați numele său într-o instrucțiune a script-ului (vezi figura 1.31).

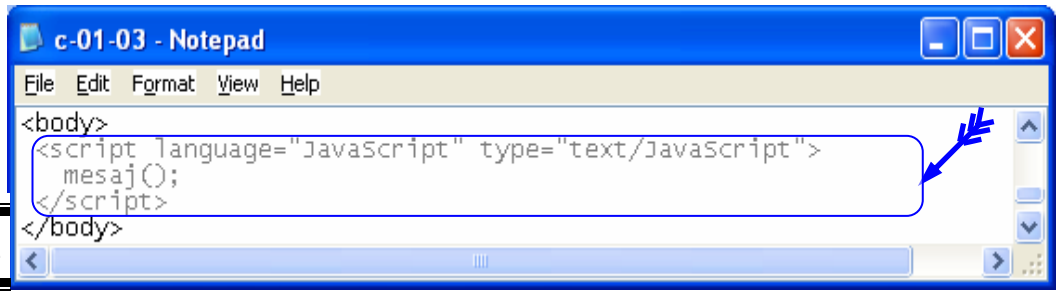


Figura 1.31

Remarcă. Mai multe detalii despre funcții (definire, apel) găsiți în Conversația 2.

5. Validați documentul XHTML 1.1 cu aplicația `validator` (vezi figura 1.32).

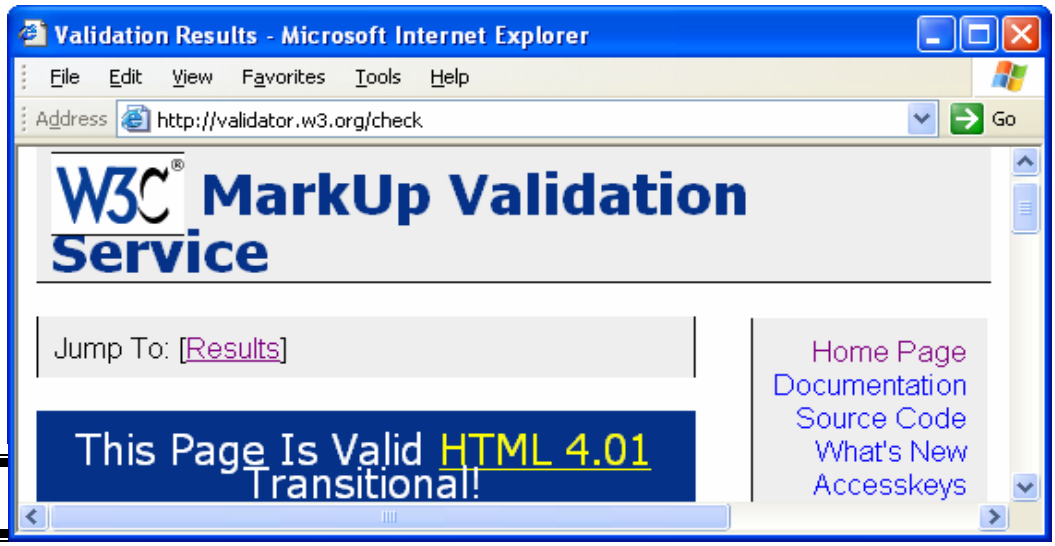


Figura 1.32

6. Inserați codul XHTML care afişează iconul de conformitate (figura 1.33).

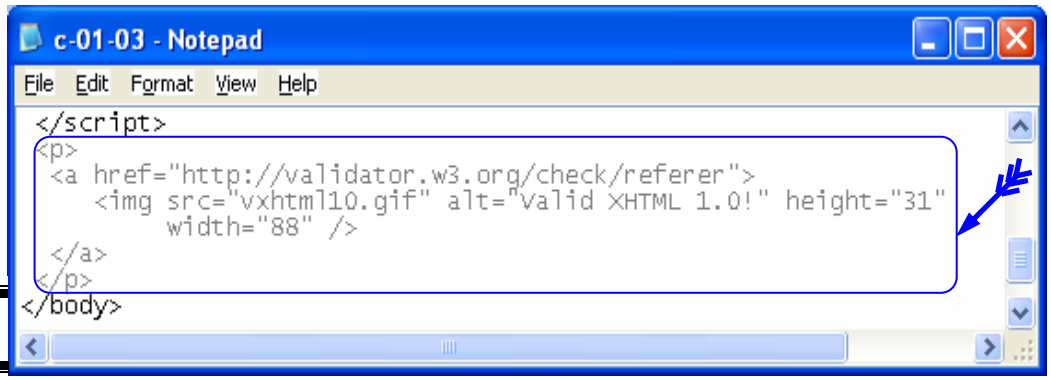


Figura 1.33

7. Afişați pagina într-un browser (figura 1.34).

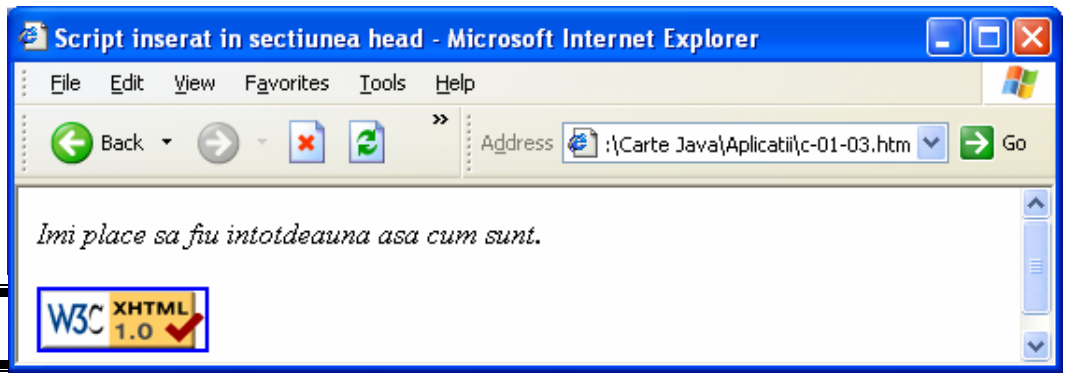


Figura 1.34

Metoda 3

Iată cum procedăm pentru a utiliza un fişier extern, în care plasăm script-ul pentru afişarea mesajului indicat.

Dacă doriți să utilizați funcțiile de script în mai multe documente, atunci plasați-le într-un fişier separat (cu extensia `.js`) pe care-l referiți apoi din documentul dumneavoastră.

Pentru a continua exemplul nostru, creați un script extern, `func•ie1.js` (vezi figura 1.35).

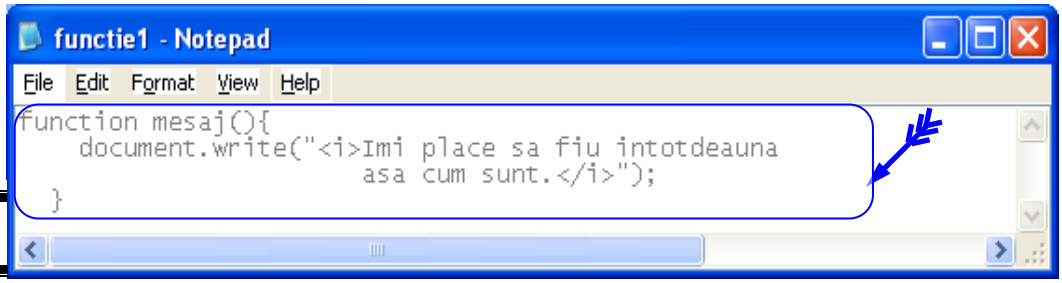


Figura 1.35

Pentru a utiliza acest script extern indicați numele fişierului în atributul `src` al tag-ului `script` (figura 1.36).

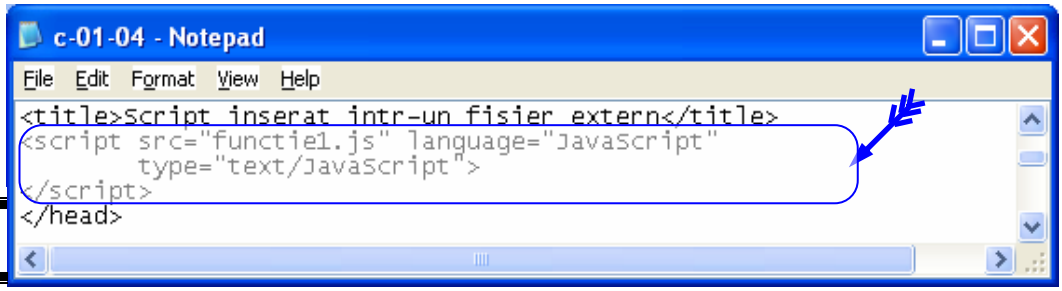


Figura 1.36

Remarcă. Acelaşi script poate fi utilizat în mai multe pagini Web diferite.

În figura 1.37 se prezintă documentul XHTML complet.

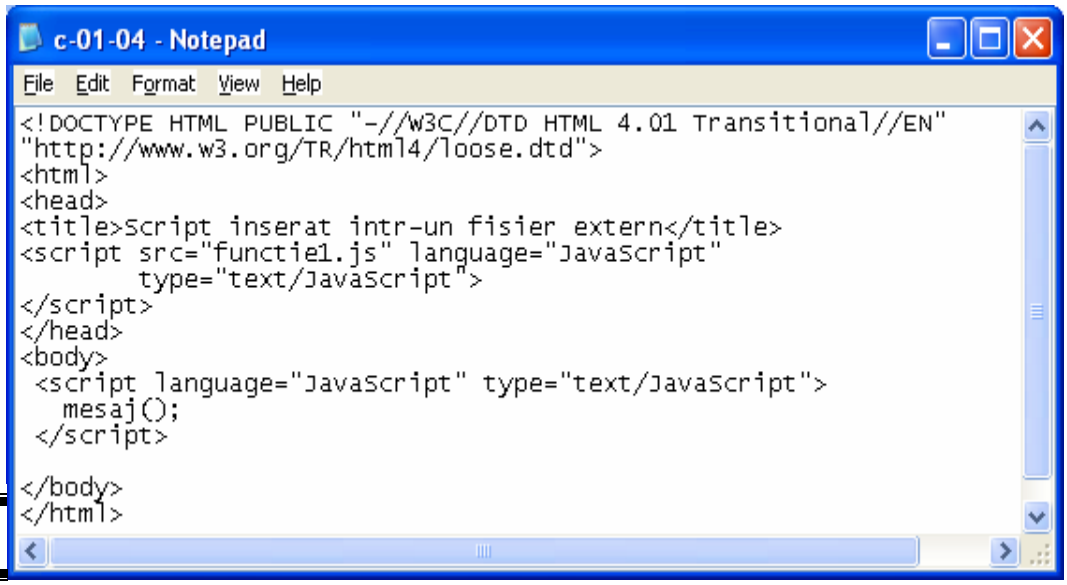


Figura 1.37

Validați documentul XHTML 1.1 cu aplicația `validator`, inserați codul XHTML care afişează icon-ul de conformitate şi afişați pagina Web într-un browser.

Rezultatul vizualizării paginii Web în Internet Explorer este ilustrat în figura 1.38.

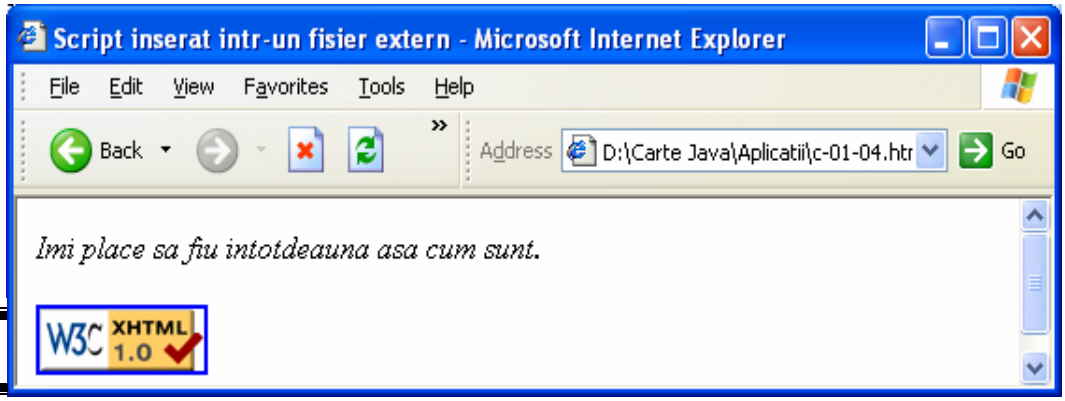


Figura 1.38

Metoda 4

Iată cum procedăm pentru a crea un script care afişează mesajul indicat cu ajutorul gestionarului de evenimente `onClick`.

1. Creați documentul XHTML (figura 1.39).

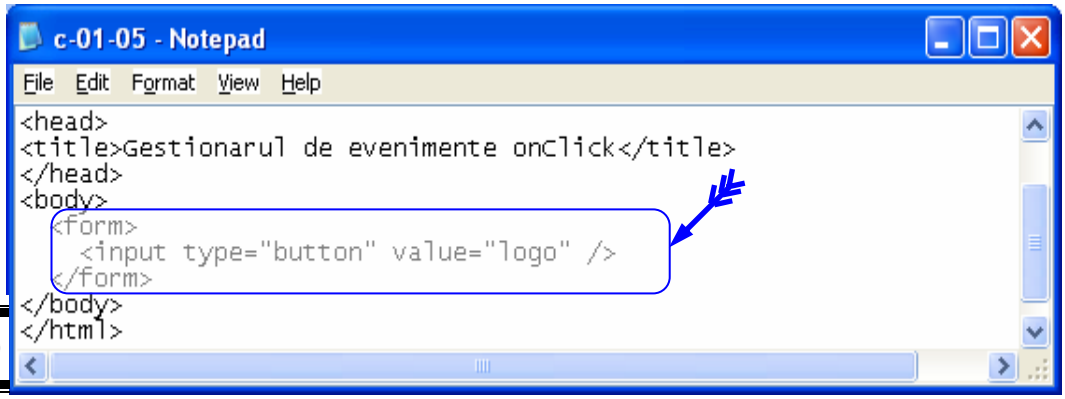


Figura 1.39

2. Introduceți în tag-ul `<input />` gestionarul de evenimente `onClick` (vezi figura 1.40).

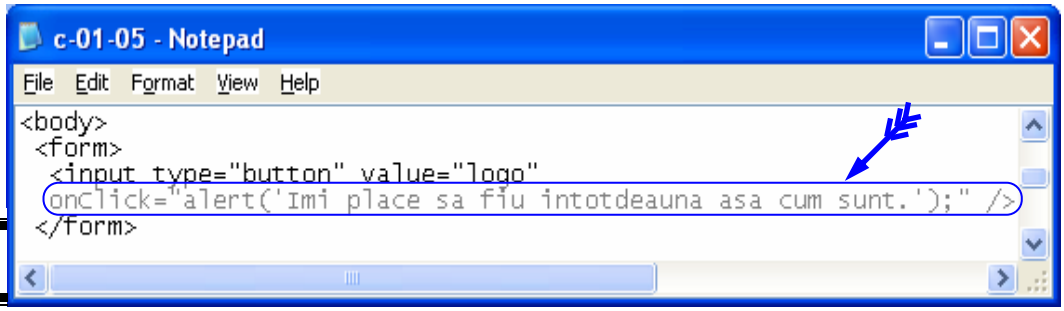


Figura 1.40

Remarci:

- ✓ Nu este obligatoriu ca toate script-urile JavaScript să se găsească în interiorul tag-urilor `<script> ... </script>`. Puteți apela, de asemenea la script-uri sub forma gestionarilor de evenimente, care indică navigatorului cum să reacționeze atunci când se produc anumite evenimente.

- ✓ Gestionarii de evenimente sunt funcții puternice JavaScript. Din fericire, ele sunt uşor de programat. Uneori este suficientă o singură instrucțiune pentru a-l activa.
- ✓ Fiecare eveniment are propriul său nume, de exemplu `click`. Pentru a defini un gestionar de evenimente, adăugați `on` la începutul numelui evenimentului (de exemplu, `onClick`).
- ✓ Gestionarul de evenimente `onClick` este activat atunci când se execută clic pe butonul „logo”. Gestionarul de evenimente `onClick` este plasat în interiorul tag-ului `<input />` al formularului, ca un atribut al acestui tag.
- ✓ Instrucțiunea JavaScript care reacționează la eveniment trebuie să fie plasată între ghilimele. De cele mai multe ori această instrucțiune va fi o funcție, în măsura în care funcțiile au avantajul de-a avea un nume precis, semnificativ şi de a conține mai multe instrucțiuni.
- ✓ Obiectul `Window` conține trei metode (`alert`, `confirm`, `prompt`) practice pentru afişarea mesajelor şi interactivitatea cu utilizatorul. Metoda `alert()` afişează o casetă de dialog de avertizare. Ea serveşte pentru transmiterea informațiilor utilizatorului.
- ✓ Metoda `alert()` este prezentată în detaliu în Conversația 2.

3. Validați documentul (X)HTML 1.1 cu aplicația `validator` (vezi figura 1.41).

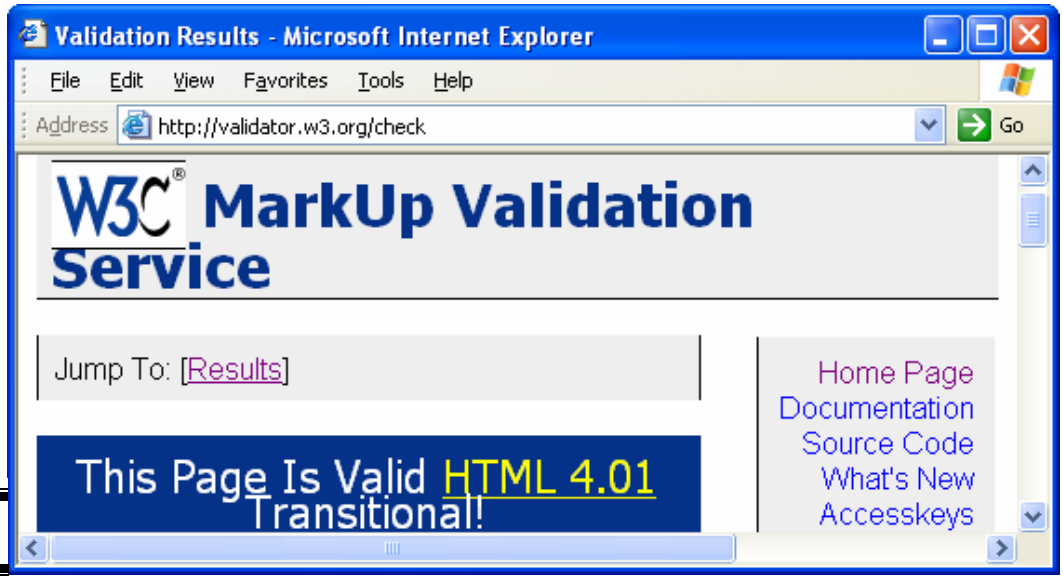


Figura 1.41

4. Inserați codul XHTML care afişează icon-ul de conformitate 

.

5. Afişați pagina Web într-un browser (figura 1.42).

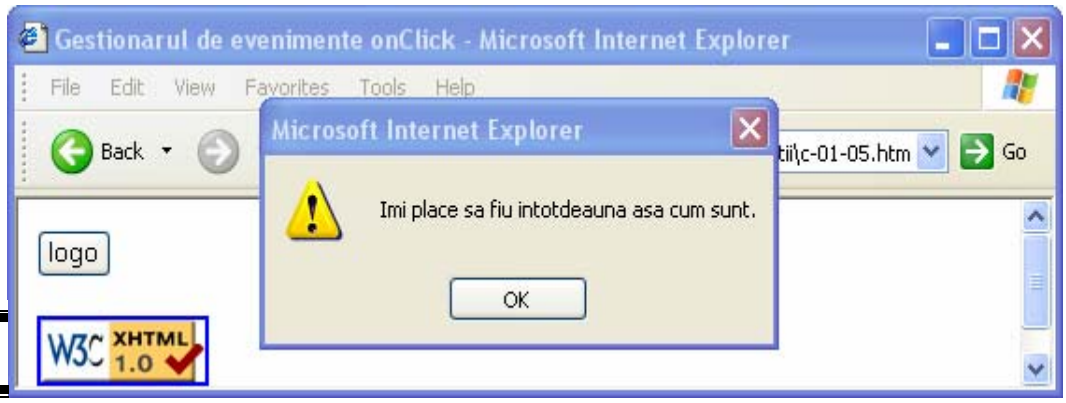


Figura 1.42

Aplicație

❑ Modificați documentul XHTML din figura 1.40 astfel încât funcția apelată din gestionarul de evenimente `onClick` să fie programată într-un script situat în secțiunea `<head>` a documentului (figura 1.43).

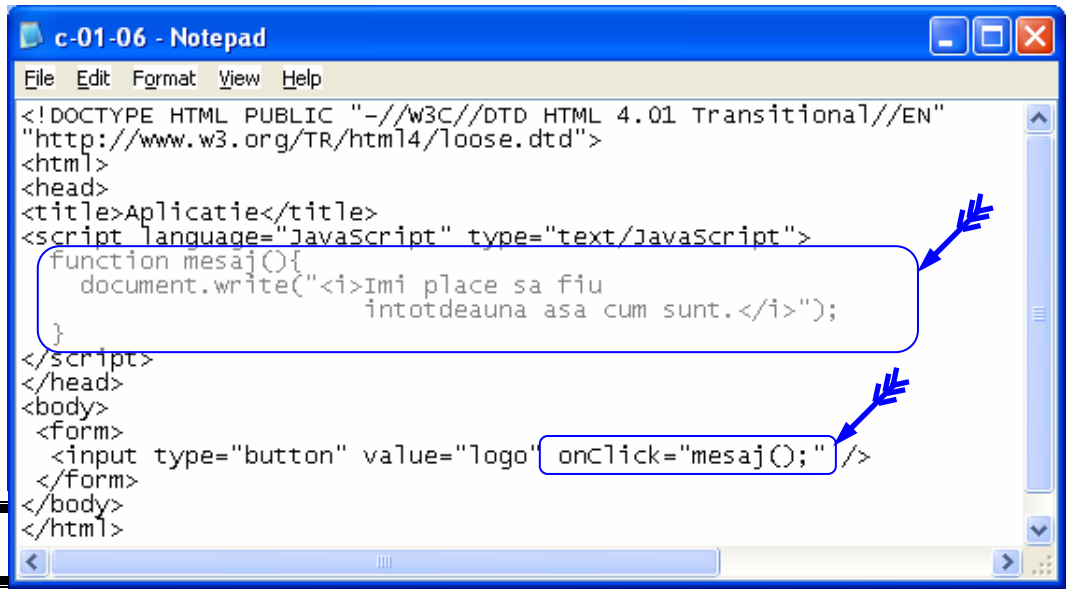


Figura 1.43

Rezultatul execuției script-ului este prezentat în figura 1.44.

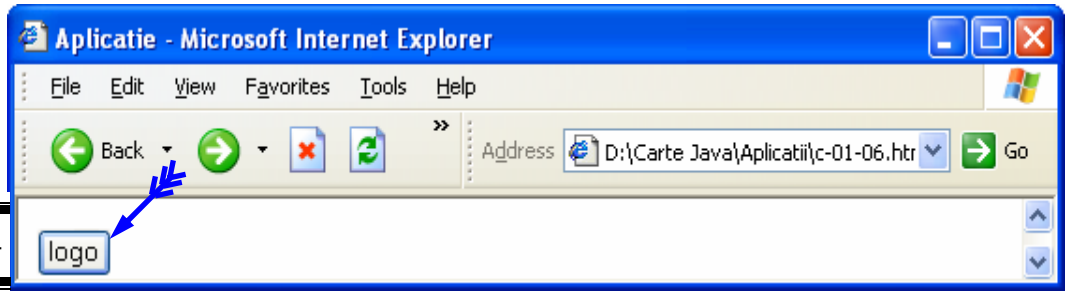


Figura 1.44

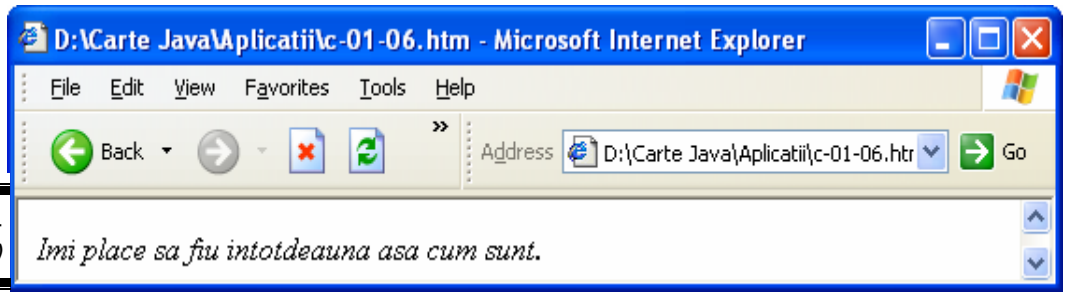


Figura 1.44 *(continuare)*

Script-urile JavaScript sunt salvate ca fişiere de tip text.

În consecinţă, puteţi folosi orice editor de text care generează fişiere text ASCII.

Un mare număr de editoare (X)HTML pot fi folosite cu JavaScript. Aceste editoare conţin, de regulă funcţii specifice JavaScript, ca de exemplu colorarea automată a instrucţiunilor sau crearea automată a script-urilor simple. Iată câteva editoare sub Windows, pe care vi le recomandăm: Homesite; Macromedia Dreamweaver; Adobe GoLive; Microsoft Front Page 2003; Netobjects Script Builder; Text Pad; Web – Expert; Ultra Edit; Edit Plus; Script Edit; Top Style.

Remarcă. Deşi oferta de editoare JavaScript este considerabilă, vă sugerăm să nu renunţaţi la editorul de text standard `Notepad`. Procedaţi cum credeţi.

Resursele JavaScript de care aveți nevoie

Pentru script-urile (JavaScript) pe care urmează să le creaţi nu aveţi nevoie decât de două instrumente de bază:

- ✓ un editor de text (simplu);
- ✓ un navigator.

Remarcă. Înainte de a vă lansa în provocarea … JavaScript pe care v-am lansat-o, vă sugerăm să vă formaţi un … background (X)HTML care să cuprindă: tag-urile (X)HTML, cadrele şi în mod special formularele. Consultaţi următoarea listă de site-uri:

- www.htmlreference.com
- www.w3.org
- www.w3.org/TR/2001/REC-xhtml11-20020531/

JavaScript şi navigatoarele

Ca şi documentele (X)HTML, script-urile JavaScript au nevoie de un navigator pentru a fi afişate. Dar, navigatoarele nu interpretează instrucţiunile în acelaşi mod.

(X)HTML are puţine probleme, dar consecinţele unei incompatibilităţi JavaScript pot fi semnificative.

La ora actuală, două navigatoare domină piaţa: Netscape şi Microsoft Internet Explorer. Cele două navigatoare rămân în continuare; nu uitaţi acest lucru atunci când vă creaţi propriile pagini.

Remarcă. JavaScript 1.5, nu este recunoscut oficial decât de Netscape 6.0 şi de Internet Explorer 5.5 şi următoarele.

După cum am menţionat, JavaScript nu este interptetat în aceeaşi manieră de către toate navigatoarele. Incompatibilitatea se manifestă în următoarele moduri:

- ✓ ignorare – un element de cod nu este „văzut" de navigator, şi-n consecinţă nu este interpretat;
- ✓ eroare – un element de cod provoacă o eroare întrucât sintaxa nu este recunoscută de navigator;
- ✓ interpretare – un element poate fi interpretat diferit de către navigator.

Ideală ar fi o compatibilitate cu toate versiunile tuturor navigatoarelor. Dar, este aproape imposibil. Totuşi, JavaScript poate detecta versiunea navigatorului, ceea ce permite ca anumite blocuri de cod să poată fi prevăzute pentru fiecare navigator. Dar această soluţie nu este întotdeauna realistă.

JavaScript 1.5

După prima sa versiune (JavaScript 1.0) lansată în anul 1995 de către Netscape (Netscape 2.0) JavaScript a evoluat considerabil.

Există în prezent cinci versiuni: 1.0, 1.1, 1.2, 1.3 şi 1.5 fiecare dintre acestea aducând îmbunătăţiri importante în raport cu precedentele.

Remarcă. La început (anul 1995), JavaScript se chema Live Script. El a fost … rebotezat JavaScript pentru a evoca marketingul comun cu limbajul Java. JavaScript este primul limbaj de script dezvoltat pentru Web.

În paralel, Microsoft a dezvoltat propriul său limbaj JavaScript denumit `JScript` în care se regăseşte cea mai mare parte a elementelor limbajului de origine la care a adăugat propriile sale specificaţii.

Rezultatul: `JavaScript nu este un standard.` Navigatoarele Netscape ignoră specificaţiile JScript iar Internet Explorer ignoră pe cele ale lui JavaScript. Opera a introdus JavaScript începând cu versiunea 3 a navigatorului său.

Prezentăm în continuare (vezi figura 1.45 şi figura 1.46) navigatoarele şi versiunile JavaScript sau JS pe care acestea le recunosc.

Versiunea limbajului JavaScript	***Netscape***
JavaScript 1.0	2.x
JavaScript 1.1	3.x
JavaScript 1.2	4.0 – 4.5
JavaScript 1.3	4.6 – 4.7
JavaScript 1.5	6

Figura 1.45

Versiunea limbajului JScript	*Internet Explorer*
JScript 1.0	3.02
JScript 3.0	4.0
JScript 5.0	5.0
JScript 5.5	5.5
JavaScript 5.6	6

Figura 1.46

Pentru a pune capăt acestor divergenţe a fost necesară crearea a două standarde: ECMA şi DOM.

- ✓ ECMA (*European Computer Makers Association*) a publicat norma ECMA – 262, o versiune standardizată a limbajului JavaScript, cunoscută sub numele de ECMA Script. JavaScript 1.3 este conformă cu norma ECMA – 262 iar JavaScript 1.5 este conformă cu ECMA – 262, versiunea 3. Netscape funcţionează de asemenea cu ECMA sub JavaScript 2.0, viitoarea versiune care va corespunde celei de-a patra ediţii a normei ECMA Script. JavaScript 2.0 va constitui o versiune mult îmbunătăţită faţă de precedentele.
- ✓ W3C (*World Wide Web Consortium*) a creat DOM (*Document Object Model*) care descrie elementele paginii Web (obiectele) şi modul în care acestea se relaţionează.

Remarci:

- ✓ La ora actuală două navigatoare domină piaţa: Netscape şi Microsoft Internet Explorer, dar nu trebuie neglijaţi nici utilizatorii altor navigatoare.
- ✓ Acordaţi toată atenţia navigatorului (şi versiunile acestora) care prelucrează JavaScript. Atenţie la incompatibilităţi! Implementarea JavaScript pe Internet Explorer şi Netscape Navigator nu este identică.

JavaScript 2.0

La data redactării acestei lucrări, JavaScript 2.0 era în curs de lansare. Vom trece deci de la versiunea 1.5 la 2.0. Motivul principal al dezvoltării JavaScript este creşterea puterii limbajului, dar şi a capacităţii sale de a se alinia la alte standarde precum: C++ şi Java.

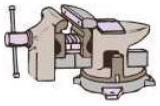
Ca elemente de noutate ale versiunii JavaScript 2.0 amintim:

- ✓ semne de punctuaţie suplimentare:
  - #; &&=; ->; ..; …; @; ^^; ^^=; !=;

- ✓ cuvinte rezervate suplimentare:
  - `abstact; debugger; enum; goto; native; protected; synchronized; throws; transient.`

Remarcă. Dacă doriţi să cunoaşteţi mai mult despre JavaScript 2.0, vizitaţi site-ul http://www.mozilla.org/js/language/js20/.

Testați-vă cunoștințele



1. Ce înţelegeţi prin stilul de viaţă Web?
2. Prin ce se deosebesc limbajele Java şi JavaScript?
3. Care sunt virtuţile limbajului JavaScript?
4. Cum pot fi inserate script-urile în documentele HTML 4.0, XHTML, Dreamweaver MX şi XML?
5. Care sunt obiectele interne ale limbajului JavaScript?
6. Când folosiţi elementul `noscript`?
7. Care este procedura de creare automată a script-urilor cu Dreamweaver MX?
8. Care sunt resursele necesare pentru crearea script-urilor JavaScript?
9. Cum se relaţionează JavaScript cu navigatoarele?

Vizitați site-urile

- ✓ http://JavaScript.Internet.com
- ✓ http://www.pageresource.com/jscript/index4.htm
- ✓ http://www.pageresource.com/jscript/index2.htm
- ✓ http://www.pageresource.com/jscript/index6.htm
- ✓ http://www.webreference.com/js
- ✓ http://www.Javasciptcity.com/tutorials/jltuto1.htm
- ✓ http://www.jscripts.com
- ✓ http://javascript.internet.com/books/
- ✓ http://www.dannyg.com/pubs/index.html

Conversația 2

Variabile şi funcții

În această conversație:

- *Tipurile de date şi valorile speciale JavaScript*
- *Variabile şi funcţii. Aplicaţii*
- *Crearea automată a script-urilor cu Dreamweaver MX*
- *EXEMPLUL 2 JAVASCRIPT*
- *Cuvinte rezervate JavaScript*
- *Temă*

Înainte de a începe să scrieţi cel de-al doilea script – EXEMPLUL 2 JAVASCRIPT va trebui să vă familiarizaţi cu elementele de bază ale limbajului JavaScript. Ele vă vor fi de folos pentru a înţelege cum să scrieţi programul. Următoarele secţiuni vă vor ajuta să înţelegeţi: tipurile de date şi valorile speciale ale limbajului JavaScript; operatorii, variabilele şi funcţiile JavaScript; obiectele matematice (`Math, Number, Boolean`) şi nu doar atât.

Tipurile de date şi valorile speciale JavaScript

De obicei, limbajele de programare cer să definiţi tipul de date pe care-l va reprezenta o variabilă, în plus se generează o eroare atunci când încercaţi să-i atribuiţi variabilei un alt tip de date. Din fericire, aşa ceva nu se întâmplă în JavaScript, care este un limbaj flexibil. Variabilele JavaScript pot accepta oricând un nou tip de date, fapt care duce la modificarea tipului variabilei.

La nivel elementar, în JavaScript nu există decât patru tipuri de date:

- ✓ numerice întregi;
- ✓ numerice în virgulă flotantă;
- ✓ caracter;
- ✓ boolean.

Toate celelalte tipuri de date pe care le veţi întâlni în limbajul JavaScript sunt `obiecte` – combinaţii ale celor patru tipuri de bază.

De exemplu, şirurile de caractere sunt obiecte care reprezintă o colecţie de caractere.

Date numerice

Limbajul JavaScript permite specificarea datelor numerice în patru formate diferite: *întreg, virgulă flotantă, octal* şi *hexazecimal*. Cu siguranţă, cunoaşteţi numerele întregi (*integer*, în limba engleză) şi numerele în virgulă flotantă (*floating – point*, în limba engleză), dar poate cunoaşteţi mai puţin pe cele exprimate în octal şi în hexazecimal.

Prin definiţie, numerele în octal şi în hexazecimal sunt numere întregi care sunt exprimate într-un sistem de numeraţie cu baza 8, respectiv baza 16.

În JavaScript un număr întreg octal este precedat de zero iar un număr hexazecimal este precedat de caracterele ”ox” sau ”OX.”

Remarci:

- ✓ JavaScript recunoaşte numerele întregi (în baza 10; pozitive sau negative) cuprinse între: +/–1.7976931348623157 E 308 şi +/–5 E – 324.
- ✓ Un număr întreg hexazecimal (*hexadecimal*, în limba engleză) începe în mod obligatoriu cu OX sau ox şi este compus din următoarele simboluri: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Puteţi utiliza majuscule sau minuscule (figura 2.1).

```
<script>
   x=0x3F;
   //x ia valoarea 3F (63 în zecimal)
</script>
```

Figura 2.1

- ✓ Un număr întreg octal începe obligatoriu cu 0 (zero) şi este compus din următoarele simboluri: 0, 1, 2, 3, 4, 5, 6, 7 (figura 2.2).

```
<script>
   x=077;
   //x ia valoarea 63 în zecimal
</script>
```

Figura 2.2

- ✓ Dacă sunteţi cumva neliniştiţi, nu vă alarmaţi căci sunt foarte rare cazurile în programele JavaScript care necesită cunoştinţe de sisteme de numeraţie în baza 8 sau în baza 16. Important este să ştiţi că ele există şi le puteţi recunoaşte!
- ✓ Un număr în virgulă flotantă este în baza 10. El poate fi pozitiv sau negativ şi poate conţine zecimale. El poate de asemenea include un exponent pozitiv sau negativ, prin E. Separatorul zecimal este întotdeauna punctul. JavaScript recunoaşte numerele cuprinse între: +/–1.7976931348623157 E 308 şi +/–5 E –324 (vezi figura 2.3).

```
<script>
   x=8.55201;
   x=0.22;
   x=.887;
   x=-.13;
   x=76.885E+9;
</script>
```

Figura 2.3

Şir de caractere

Cu siguranţă că dumneavoastră veţi lucra foarte mult cu tipul de date şir de caractere (alfanumerice).

Un şir de caractere (*string*, în limba engleză) este compus din litere, cifre, simboluri, caractere speciale şi secvenţe de ieşire. Conţinutul unui şir de caractere este considerat tot timpul ca fiind text, chiar dacă el constă din cifre şi simboluri numerice. Un şir de caractere este încadrat de ghilimele simple sau duble. Un şir de caractere încadrat de ghilimele duble poate fi inclus într-un şir încadrat de ghilimele simple şi vice versa.

Aplicaţie

❑ Testaţi şirurile de caractere: ”septembrie”; ”9.80”; ”Salut, Droopy!”; ’a=”01”’; ”anume=’minim’” cu următorul script (figura 2.4).

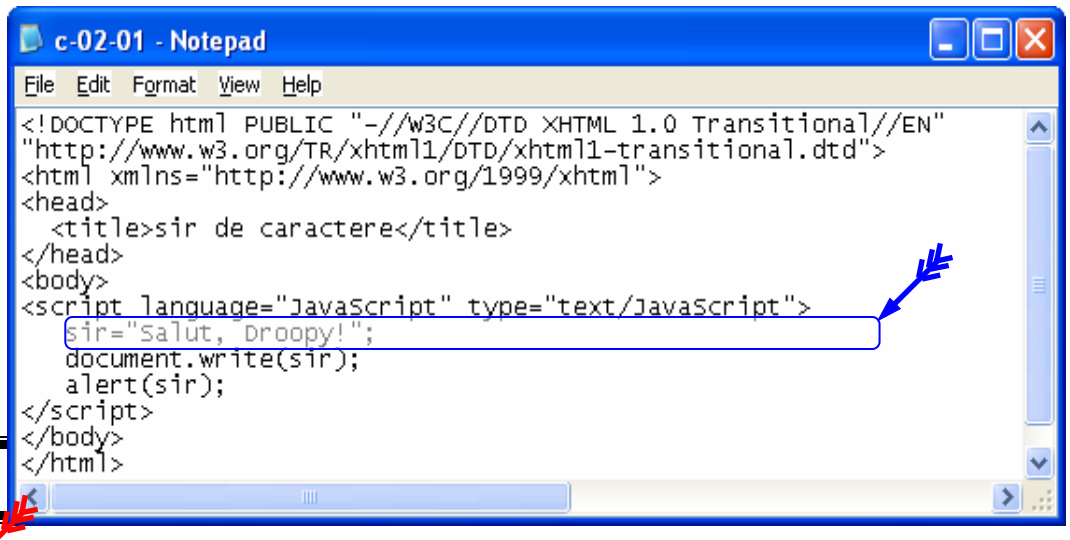


Figura 2.4

Indicaţie. Introduceţi în variabila `sir` datele de test (şirurile de caractere indicate).

Remarcă. Într-un şir de caractere puteţi introduce de asemenea şi caractere care nu se găsesc pe tastatură. În acest caz, utilizaţi o secvenţă de ieşire (*escape sequence*, în limba engleză) care începe întotdeauna cu „\" urmat de un semn sau un cod numeric.

Valori logice sau booleene

Ele sunt în număr de două: `true` (adevărat) şi `false` (fals).

Cele două valori se folosesc pentru a indica dacă rezultatul evaluării unei condiţii este adevărat sau nu.

Nu vă lăsaţi impresionaţi de aceşti termeni!

În figura 2.5 se prezintă un exemplu de utilizare a celor două valori booleene.

Figura 2.5

Remarcă. Cele două valori nu sunt plasate între ghilimele, întrucât *true* şi *false* sunt cuvinte cheie (vezi figura 2.5) JavaScript, având o semnificaţie precisă pentru interpretorul JavaScript.

Valorile speciale JavaScript

În limbajul JavaScript puteţi întâlni de asemenea şi anumite valori speciale pe care vi le semnalăm în cele ce urmează: *Infinity, NaN, null*.

Infinity

Este o valoare numerică specială care se returnează dacă un număr în virgulă flotantă este superior valorii maxime autorizate sau este inferior valorii minime autorizate.

`Infinity` poate fi pozitiv sau negativ.

Remarcă. Folosiţi proprietăţile obiectului `Number`: `Number.POSITIVE_INFINITY` şi `Number.NEGATIVE_INFINITY` pentru a testa dacă `Infinity` este pozitiv sau negativ (vezi Conversaţia 2 (continuare)).

Atunci când o expresie aritmetică conţine o valoare `Infinity`, ea returnează întotdeauna `infinity` (vezi figura 2.6).

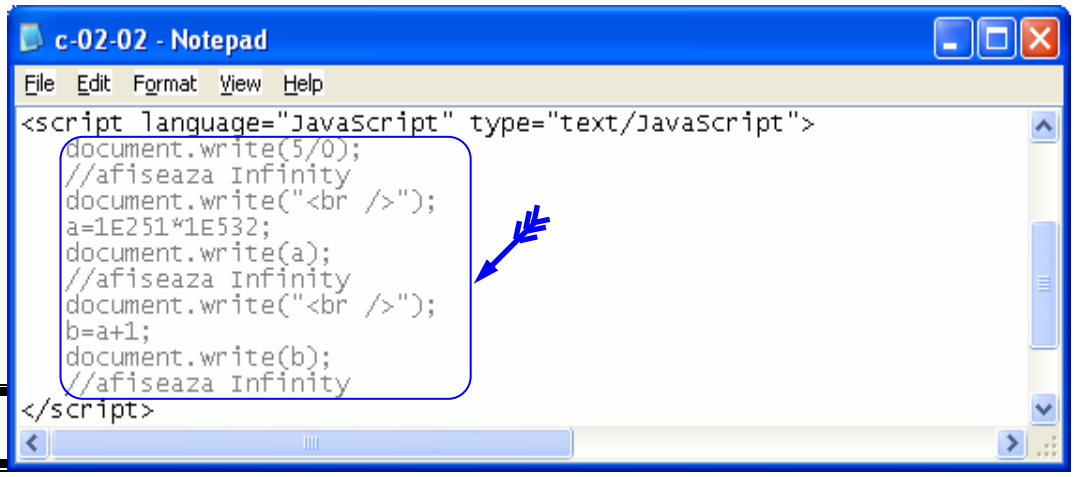


Figura 2.6

NaN

`NaN` (*Not a Number*, în limba engleză) este o valoare specială furnizată ca rezultat de câteva operaţii aritmetice (vezi figura 2.7).

Remarcă. `NaN` semnifică: *„acesta nu este un număr!"*

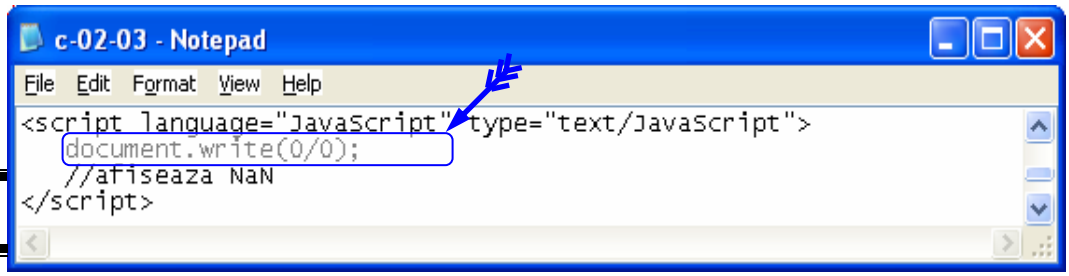


Figura 2.7

null

`null` este o valoare specială care indică absenţa valorii (figura 2.8).

Remarci:

- ✓ `null` nu este egal nici cu zero şi nici cu `undefined`;
- ✓ Valoarea null se poate simula cu operatorul `void: var null=void(0).`

Figura 2.8

undefined

`undefined` este o valoare specială nedefinită.

Remarci:

- ✓ `undefined` nu este egal nici cu `null` nici cu zero.
- ✓ Valoarea `undefined` este returnată în cazul utilizării unei variabile care nu a fost definită sau în cazul unei variabile care a fost declarată dar fără conţinut (figura 2.9).
- ✓ Valoarea `undefined` este des confundată cu `null` de către navigatoare.

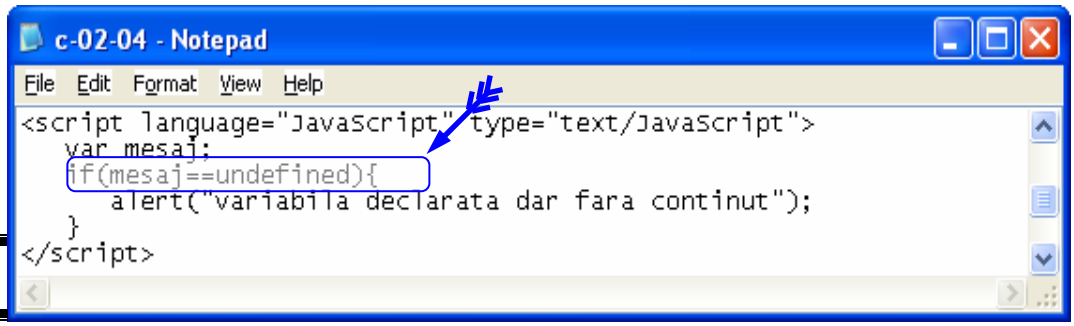


Figura 2.9

Variabile şi funcţii

Variabile JavaScript

Spre deosebire de alte limbaje de programare care impun declararea prealabilă a variabilelor, cu specificarea tipului de date pe care acestea le conţin (întregi, reale, caracter etc.) JavaScript procedează puţin altfel: o variabilă este creată prin simpla afectare (atribuire) a unei valori, din acel moment ea devenind disponibilă. Dacă sunteţi obişnuit cu declararea prealabilă a variabilelor, vă va trebui puţin timp pentru a vă acomoda cu acest mod de lucru rapid şi ... cu puţină practică veţi aprecia cu siguranţă această virtute deosebită a limbajului JavaScript.

Tipuri de variabile

În limbajul JavaScript nu trebuie să specificaţi tipul variabilelor. Atunci când creaţi o variabilă, valoarea pe care o atribuiţi determină tipul variabilei (vezi figura 2.10).

```
<script>
    a=83;
    //a este o variabilă de tip numeric
    mesaj="Au înnebunit salcâmii!";
    //mesaj este o variabilă de tip şir de caractere
</script>
```

Figura 2.10

Remarcă. O variabilă poate să-şi schimbe tipul, în mod dinamic, în interiorul aceluiaşi script (vezi figura 2.11).

```
<script>
    b="nemaipomenit";
    b=10;
</script>
```

Figura 2.11

Numele variabilelor

Fiecare variabilă are un nume. *Regulile* după care se formează numele variabilelor sunt următoarele:

- ✓ numele variabilelor pot conţine toate literele alfabetului (majuscule sau minuscule), cifre (0 la 9) şi caracterul de subliniere (_);
- ✓ numele variabilelor nu trebuie să conţină spaţii şi semne de punctuaţie;
- ✓ numele variabilelor sunt sensibile la majuscule şi minuscule;
- ✓ primul caracter al numelui unei variabile trebuie să fie o literă sau un caracter de subliniere;
- ✓ nu este o limită teoretică pentru numărul de caractere al numelui unei variabile dar nu uitaţi că trebuie să tastaţi corect de două ori numele unei variabile pentru a putea fi utilizată (nu vă complicaţi viaţa!)

În figura 2.12 se prezintă câteva exemple de nume de variabile valide şi invalide.

Nume valide	***Nume invalide***
`nr_persoane`	`π TURCA`
`Num•rPersoane`	`<abc`
`a1`	`a-1`
`b`	`b>1a`
`_var13`	`+a13`

Figura 2.12

Remarci:

- ✓ În afară de variabilele scalare, care nu reprezintă decât o singură entitate mai există şi matrici (vezi Conversaţia 4).
- ✓ În general, variabilele scalare nu sunt declarate, dar în diferite cazuri ele pot fi declarate ca fiind de un tip particular (`boolean, number` sau `string`).
- ✓ JavaScript recunoaşte mai multe tipuri de constante: întregi (pot fi exprimate în sistem zecimal, octal sau hexazecimal); flotante; şiruri de caractere; boolean.

Variabile locale şi globale

În JavaScript, cuvântul cheie `var` poate servi la declararea unei variabile. Totuşi, în foarte multe cazuri, puteţi să-l omiteţi. Variabila va fi în mod automat declarată în momentul în care îi atribuiţi o valoare. Pentru a şti în care situaţie o variabilă trebuie să fie declarată trebuie să înţelegem mai întâi conceptul de `portabilitate` (*scope*, în limba engleză).

Portabilitatea unei variabile (sau funcţii) indică dacă variabila (sau funcţia) definite într-o zonă a programului sunt accesibile dintr-o altă zonă a programului.

Există două tipuri de variabile:

- ✓ variabile globale - sunt definite în exteriorul oricărei funcţii JavaScript (funcţiile sunt întotdeauna globale);
- ✓ variabile locale - sunt definite în interiorul unei funcţii. Utilizaţi cuvântul cheie `var` pentru a evita confuzia între o variabilă locală şi o variabilă globală cu acelaşi nume (omonimă).

Exemplul din figura 2.13 vă va ajuta să înţelegeţi mai bine conceptul de portabilitate (Teoria fără praxă ...).

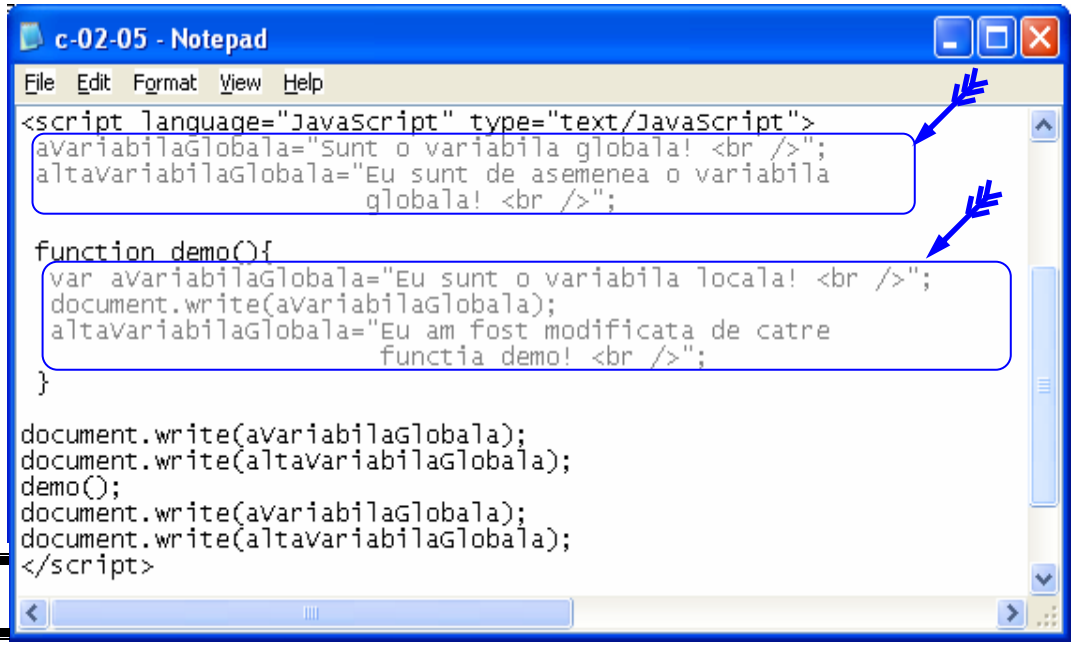


Figura 2.13

În figura 2.14 este prezentat rezultatul execuţiei script-ului.

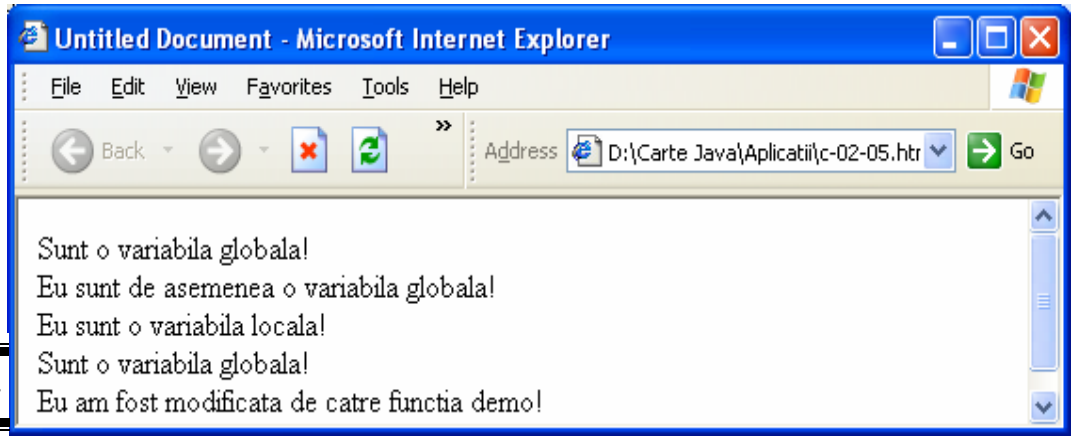


Figura 2.14

Aplicaţie

❑ Simulaţi funcţionarea acestui script (vezi figura 2.13). Dacă şi după acest exerciţiu, conceptul de portabilitate vi se pare şi mai confuz, nu disperaţi! Vom reveni cu explicaţii suplimentare!

Cum atribuiţi valori variabilelor?

Două sunt modurile prin care se atribuie valori variabilelor JavaScript:

- ✓ în mod static - utilizaţi semnul "=" pentru a afecta o valoare unei variabile (instrucţiunea de atribuire).
- ✓ în mod dinamic - utilizaţi metoda `prompt()`.

Remarci:

- ✓ Structura instrucţiunii de atribuire este foarte simplă: variabila din stânga semnului egal (=) ia valoarea indicată în dreapta semnului egal (vezi figura 2.15).

Figura 2.15

```
<script>
    pi=3.14159;
    i=i+1;
    nume="Droopy";
    prenume='Ion';
</script>
```

- ✓ Instrucţiunile JavaScript trebuie să se termine cu punct şi virgulă ... sau nu! Dacă doriţi să plasaţi mai mult de o instrucţiune pe o linie, punct şi virgulă este obligatoriu după fiecare instrucţiune (figura 2.16).

Figura 2.16

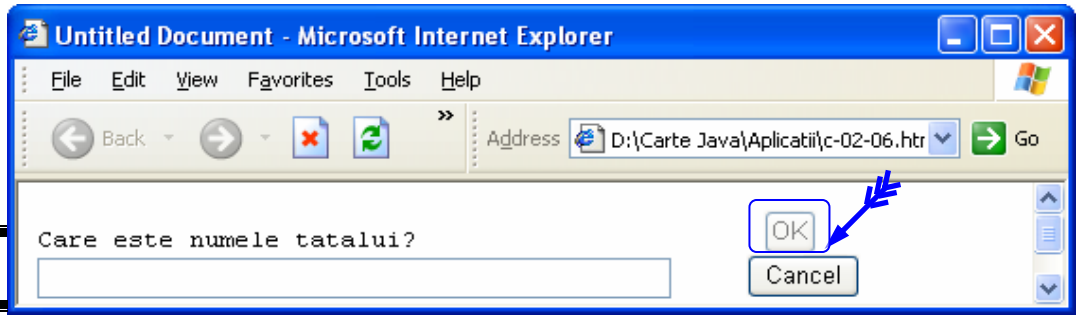
```
<script>
    pi=3.14159; i=i+1; nume="Droopy"; prenume='Ion';
</script>
```

- ✓ Metoda `prompt()` (vezi obiectul `Window`, Conversaţia 7) deschide o casetă de dialog care conţine: mesajul indicat (primul argument), o zonă de text ce urmează a fi completată de utilizator (al doilea argument), butonul OK şi butonul Cancel. Cel de-al doilea argument este facultativ. Dacă el este absent, zona de text afişează implicit `undefined`. Utilizatorul poate să introducă o altă valoare sau să execute clic pe butonul OK pentru a accepta răspunsul implicit. Dacă utilizatorul execută clic pe butonul OK, valoarea conţinută în zona de text este returnată script-ului; dacă execută clic pe butonul Cancel, valoarea `null` este returnată script-ului. Instrucţiunea următoare:

```
var raspuns=prompt("Care este numele tatalui?","");
```

deschide caseta de dialog prezentată în figura 2.17.

Figura 2.17



Conversia tipurilor de date

Ori de câte ori este posibil JavaScript converteşte în mod automat tipul de date întâlnit.

Pentru conversia tipurilor de date, JavaScript propune următoarele două funcţii:

- ✓ `parseInt()` - converteşte un şir de caractere într-un număr întreg.
- ✓ `parseFloat()` - converteşte un şir de caractere într-un număr în virgulă mobilă.

Remarcă. Cele două funcţii detectează numerele la începutul şirului de caractere. Dacă nici unul din numere nu este găsit la începutul şirului de caractere, funcţiile returnează valoarea `NaN` (Not a Number).

Aplicaţie

❑ Simulaţi funcţionarea următorului script (vezi figura 2.18).

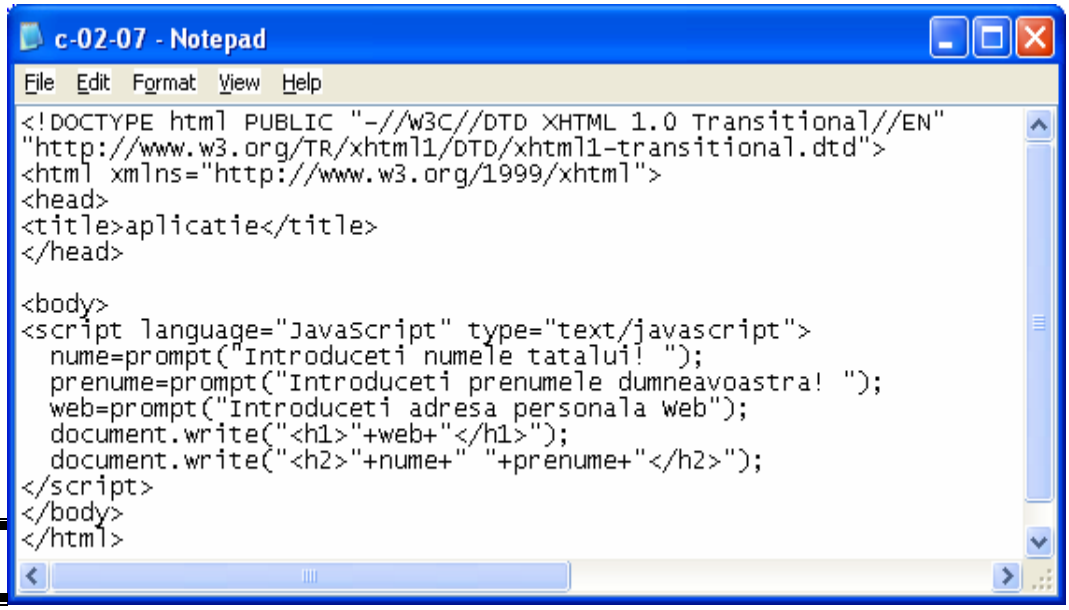


Figura 2.18

Funcţii JavaScript

Definiţi o funcţie

O funcţie este un grup de instrucţiuni tratate ca o singură entitate. Pentru a utiliza o funcţie va trebui mai întâi s-o definiţi.

Iată cum definim o funcţie (`logo()`) care afişează mesajul: *"Îmi place să fiu întotdeauna aşa cum sunt"*.

În figura 2.19 este prezentată funcţia pe care ne-am propus s-o construim.

Figura 2.19

```
function logo(){
  alert("Imi place sa fiu intotdeauna asa cum sunt.");
}
```

Funcţia este foarte simplă, dar vom analiza rapid fiecare din elementele sale.

- ✓ Primul element este cuvântul cheie `function` (figura 2.20) care indică interpretorului JavaScript că blocul de cod care urmează trebuie să fie tratat ca pe o funcţie şi numai după aceea el este apelat în mod explicit.

Figura 2.20

```
function logo (){

}
```

- ✓ Al doilea element este numele funcţiei (figura 2.21), aici `logo`. Regulile de formare a numelui funcţiei sunt identice cu cele ale numelui variabilelor (vezi paragraful Variabile).

Figura 2.21

```
function logo () {

}
```

- ✓ Al treilea element este setul de paranteze (figura 2.22) care conţine lista parametrilor funcţiei (în exemplul nostru, nu este cazul!)

Figura 2.22

```
function logo ( ) {

}
```

- ✓ Al patrulea element este acolada de deschidere, care marchează debutul corpului funcţiei (vezi figura 2.22).
- ✓ Al cincilea element, pe linia următoare (figura 2.23) este metoda `alert`, care afişează într-o casetă de dialog textul indicat în interiorul parantezelor.

Figura 2.23

```
function logo(){
  alert("Imi place sa fiu intotdeauna asa cum sunt.");
}
```

- ✓ Ultimul element este acolada de închidere (figura 2.24), care marchează sfârşitul funcţiei.

Figura 2.24

```
function logo(){
}
```

Să revenim la paranteze. Exemplul nostru (`func•ia logo()`) face întotdeauna acelaşi lucru: ea va afişa acelaşi mesaj ori de câte ori este apelată. Ceea ce nu este nemaipomenit!

Pentru a exploata din plin o funcţie, se recomandă utilizarea parametrilor, pe care îi vom numi *argumente*.

Este vorba de variabile utilizate de către funcţie ori de câte ori aceasta este apelată. Puteţi, de exemplu să utilizaţi un parametru `mesaj` ce reprezintă logo-ul pe care doriţi să-l afişaţi.

Funcţia `logo()` modificată este prezentată în figura 2.25.

Figura 2.25

```
function logo(mesaj){
  alert(mesaj);
}
```

De bună seamă, pentru a putea utiliza o astfel de funcţie, va trebui s-o includeţi într-un document (X)HTML.

În general, cea mai bună zonă pentru definirea unei funcţii este antet-ul `<head>` al documentului.

Deoarece instrucţiunile antet-ului sunt executate primele, veţi avea certitudinea că funcţia este definită înainte de a fi utilizată.

În figura 2.26 este prezentat documentul (X)HTML incomplet în care s-a inclus, în secţiunea `<head>` ... `</head>` script-ul ce conţine funcţia `logo()` modificată.

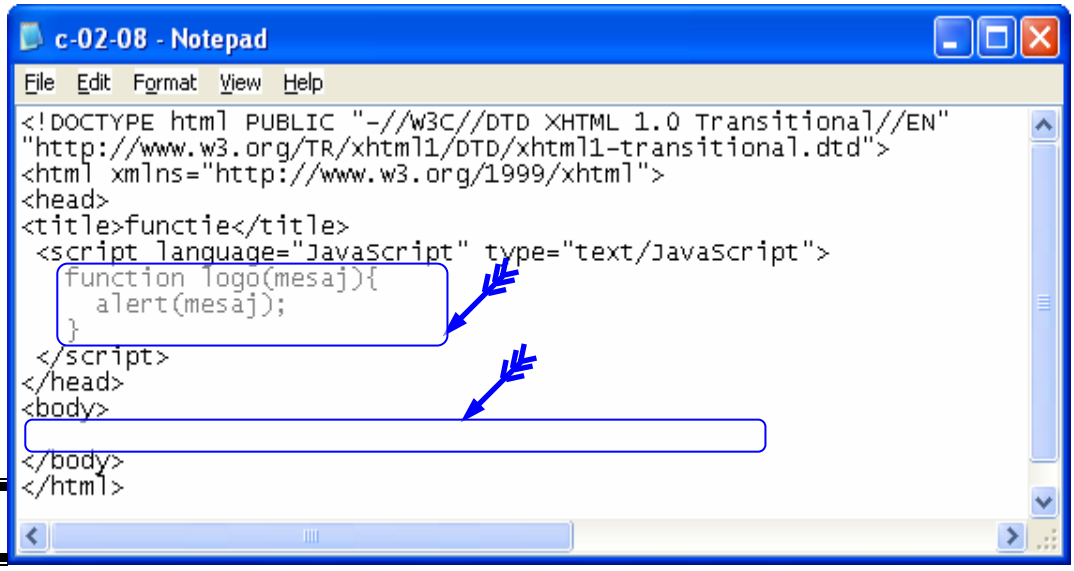


Figura 2.26

Apelaţi funcţia

În acest moment funcţia (`logo(mesaj)`) este definită şi inserată într-un document (X)HTML.

Dacă afişaţi documentul (X)HTML (vezi figura 2.26) într-un browser, nu se întâmplă absolut nimic. Funcţia este pregătită, dar … inutilizabilă!

Pentru a putea utiliza o funcţie va trebui s-o apelaţi. Pentru a apela o funcţie, va trebui să utilizaţi numele său într-o instrucţiune a script-ului şi să includeţi parantezele şi parametrii.

În figura 2.27 este prezentat documentul (X)HTML complet, cu definiţia funcţiei şi apelul funcţiei în corpul documentului (X)HTML.

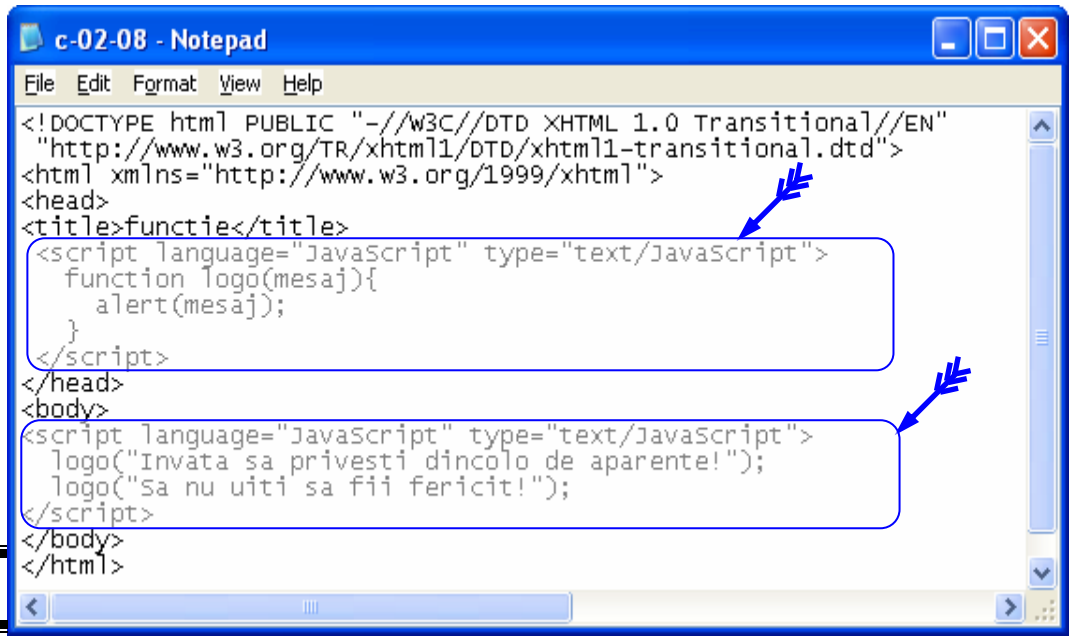


Figura 2.27

Pentru a înţelege mai bine cum … lucrează funcţia `logo(mesaj)`, apelaţi funcţia de două ori, pentru a afişa două logo-uri diferite (vezi figura 2.27).

Remarci:

- ✓ Documentul XHTML (vezi figura 2.27) conţine al doilea set de tag-uri `<script>` inserat în corpul paginii.
- ✓ În cel de-al doilea script veţi găsi două apeluri ale funcţiei `logo()`, fiecare cu un mesaj diferit (vezi figura 2.28).

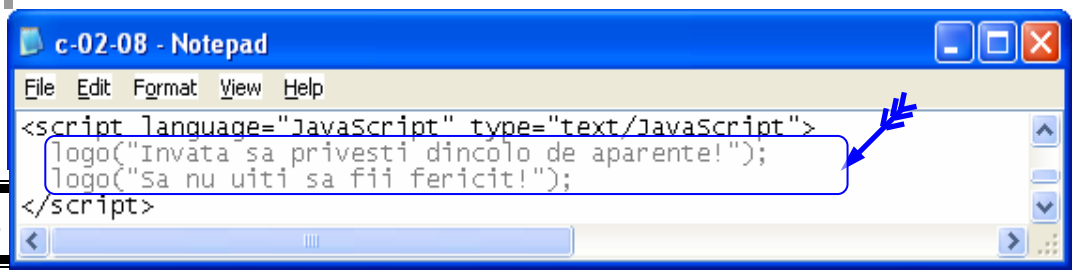


Figura 2.28

În figura 2.29 se prezintă rezultatul execuţiei funcţiei `logo()`. Pentru a afişa cel de-al doilea logo executaţi clic pe butonul OK.

Figura 2.29

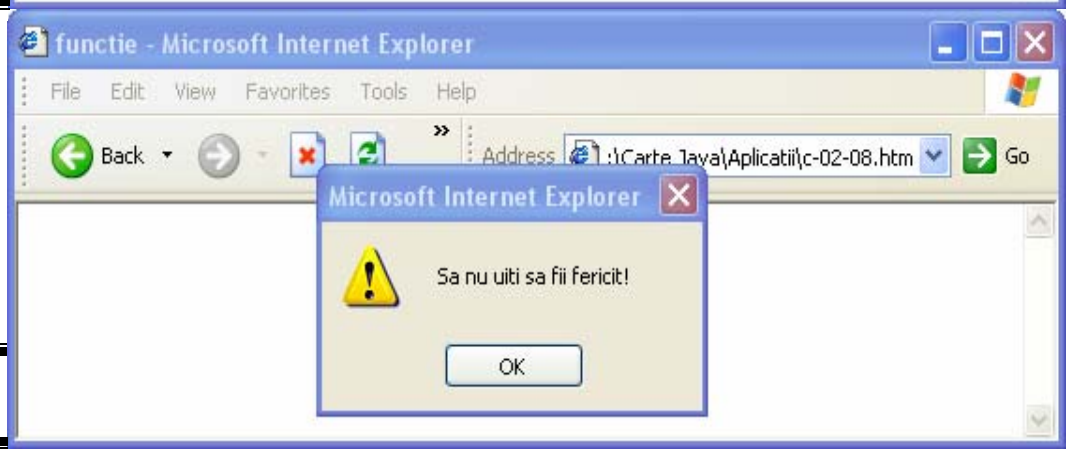


Figura 2.29 *(continuare)*

Returnarea unei valori

Funcţia (`logo()`) pe care am creat-o în aplicaţia precedentă afişează un mesaj. O funcţie poate, de asemenea returna o valoare script-ului apelant.

Iată cum procedăm pentru a defini şi apela o funcţie care calculează media aritmetică a trei valori.

În figura 2.30 este prezentată funcţia pe care ne-am propus s-o construim.

```
function media(a,b,c){
   rezultat=(a+b+c)/3.0;
   return rezultat;
}
```

Figura 2.30

Remarci:

- ✓ Funcţia începe cu `function` – cuvânt rezervat, urmat de numele funcţiei – media şi de trei parametrii – a, b, c pentru cele trei valori (numerice).
- ✓ Acolada de deschidere a fost plasată la sfârşitul primei linii, dar puteţi foarte bine să o izolaţi pe o singură linie, sau să o plasaţi la începutul liniei următoare.
- ✓ Variabila (locală) `rezultat` stochează media aritmetică a celor 3 valori numerice.
- ✓ Pentru a returna rezultatul (`rezultat`) script-ului, utilizăm cuvântul rezervat `return` (vezi figura 2.31).

```
...
   return rezultat;
}
```

Figura 2.31

În figura 2.32 se prezintă documentul complet (X)HTML. Acest document conţine de asemenea, în secţiunea `<body>` un mic script care apelează funcţia `media()` şi afişează rezultatul.

Figura 2.32

c-02-09.htm - Notepad

File Edit Format View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>function</title>
 <script language="JavaScript" type="text/JavaScript">
   function media(a,b,c){
     rezultat=(a+b+c)/3.0;
         return rezultat;
   }
 </script>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
  mediaaritmetica=media(7,8,9);
  document.write("Media aritmetica este: "+mediaaritmetica);
</script>
</body>
</html>
```

Remarci:

- ✓ Pentru apelul funcţiei puteţi folosi o variabilă de stare (`mediaaritmetic•`, în exemplul nostru).
- ✓ Puteţi plasa apelul unei funcţii într-o expresie.
- ✓ În paralel cu variabilele şi valorile constantelor, puteţi utiliza apelul funcţiilor care returnează rezultatele în interiorul unei expresii (vezi figura 2.33).

Figura 2.33

```
...
alert(media(7,8,9));
...
```

Crearea automată a script-urilor cu Macromedia DREAMWEAVER

După cum am precizat în Conversaţia 1 (vezi paragraful: Oferta de editoare JavaScript), Macromedia Dreamweaver MX poate fi folosit şi ca editor JavaScript. Funcţiile sale specifice, modul simplu de utilizare îl recomandă ca pe un excelent instrument de editare a script-urilor.

În figura 2.34 sunt prezentate cele mai utilizate funcţii JavaScript puse la dispoziţie de Dreamweaver.

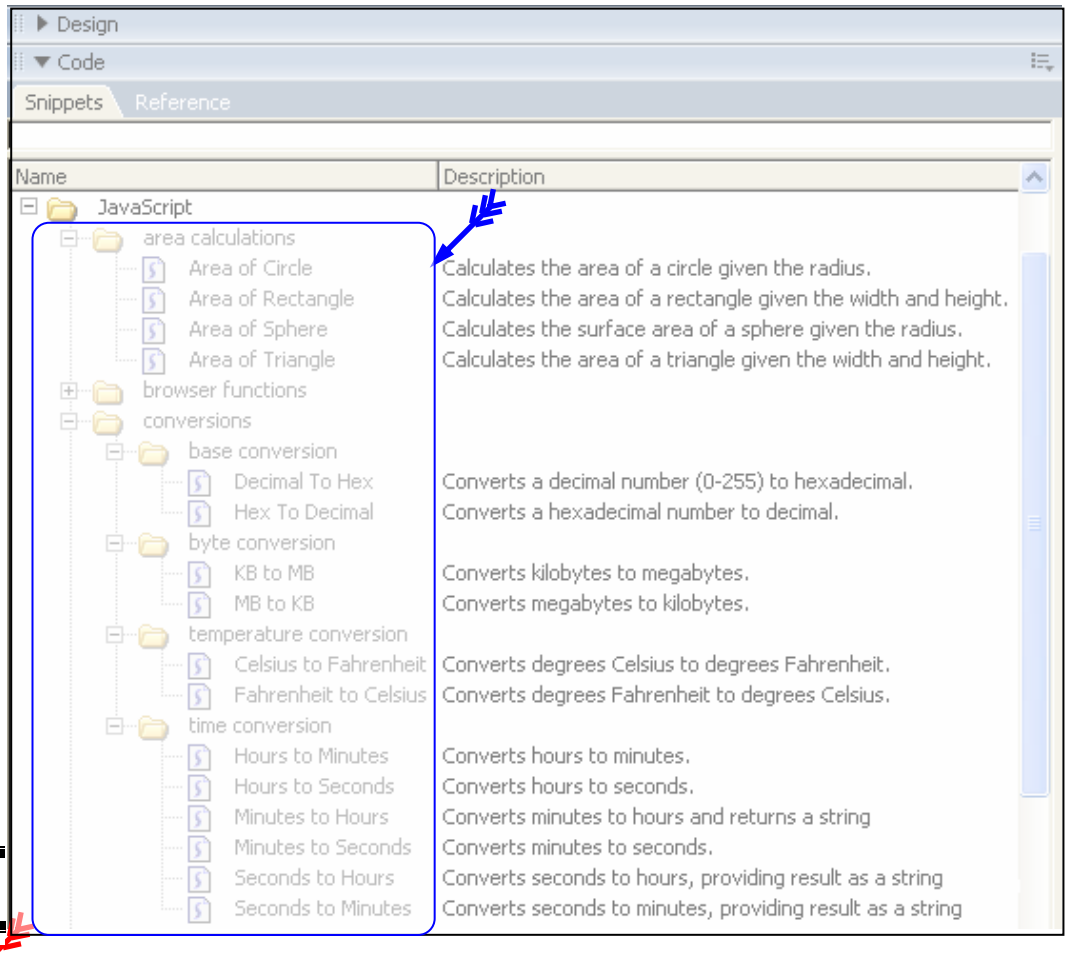


Figura 2.34

Iată cum procedăm pentru a crea un script care calculează aria unei sfere (*Area of Sphere*).

1. În grupul de panouri `Insert`, executaţi clic pe subpanoul `HTML` (figura 2.35).

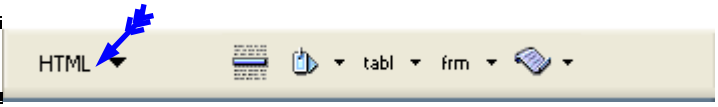


Figura 2.35

2. Executaţi clic în locul în care doriţi să inseraţi script-ul.

3. Executaţi clic pe butonul (Script), figura 2.36.

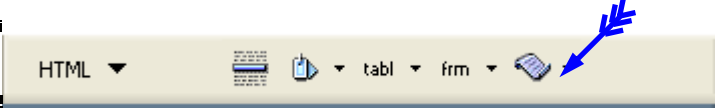


Figura 2.36

Remarcă. Se deschide caseta de dialog Script (figura 2.37).

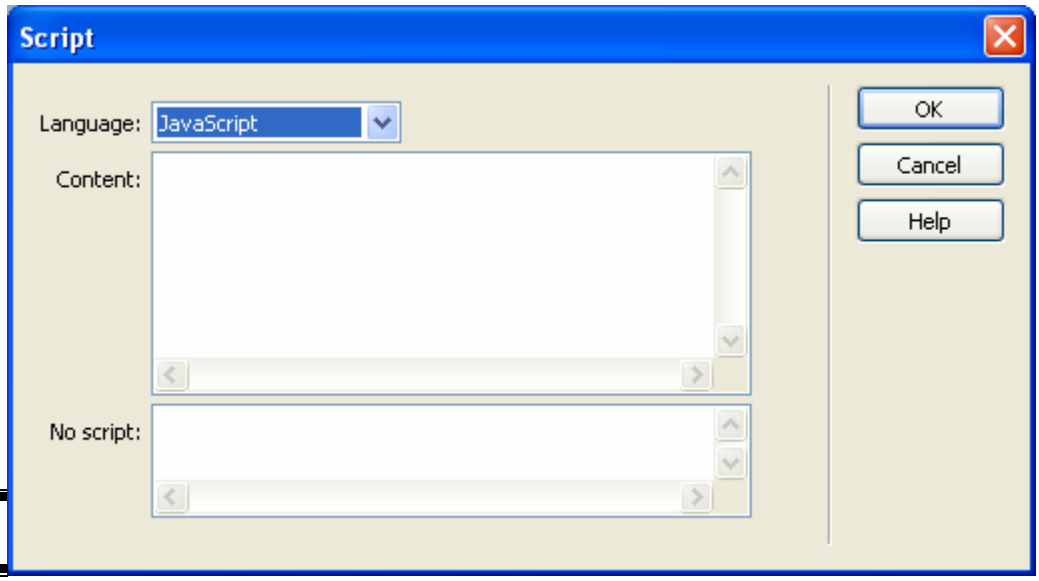


Figura 2.37

4. Executaţi clic pe butonul OK al ferestrei de dialog `Script` (figura 2.38).

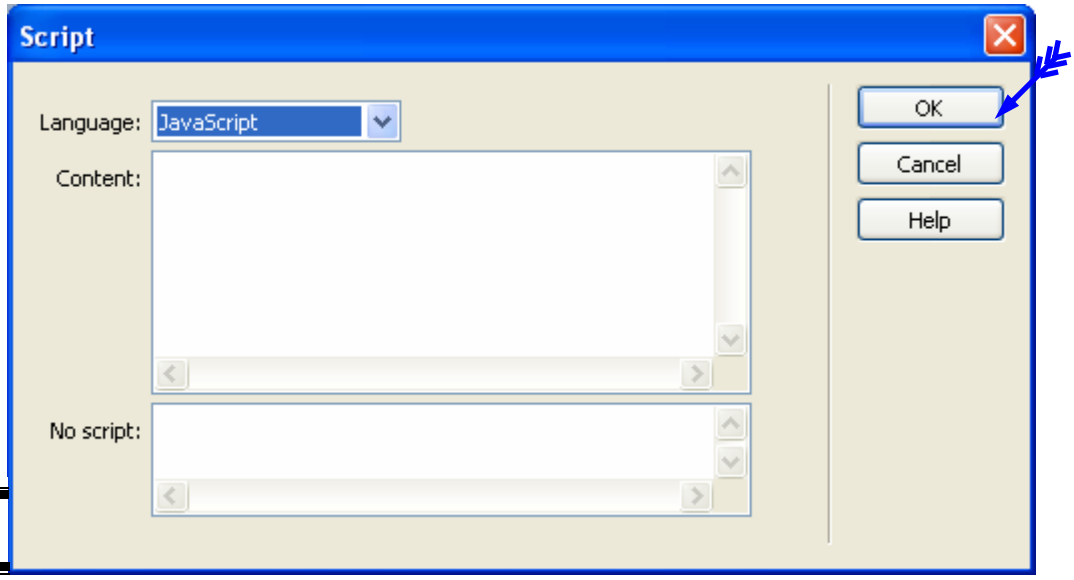


Figura 2.38

Remarcă. Se inserează elementul `<script> ... </script>` (figura 2.39).

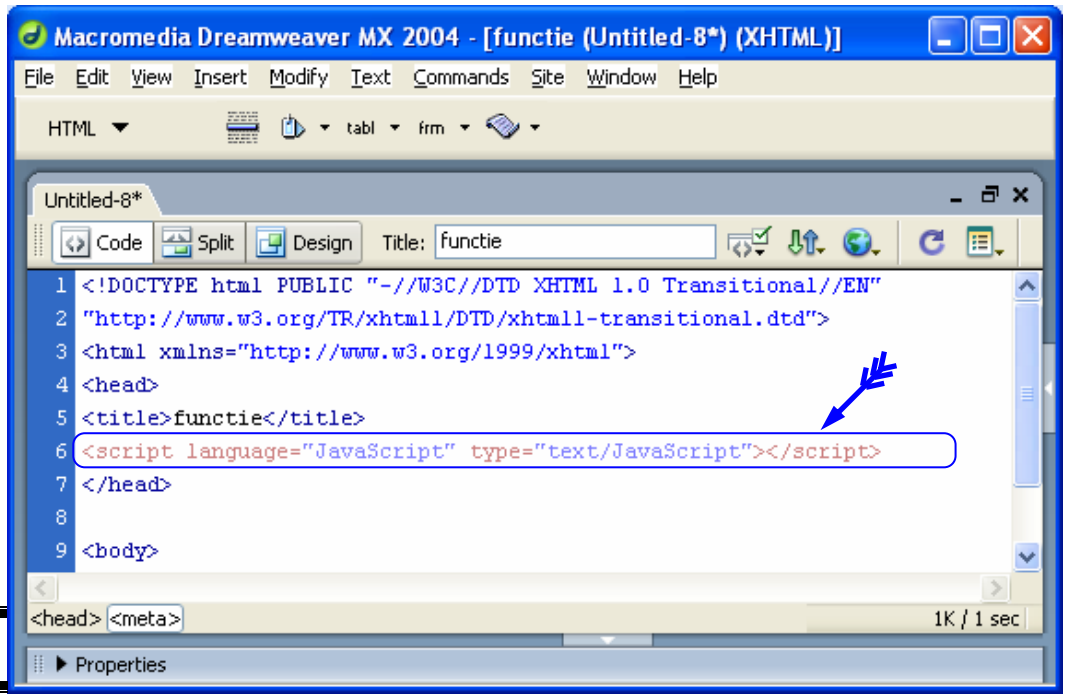


Figura 2.39

5. Între tag-urile `<script>` şi `</script>` inseraţi funcţia `areaOfSphere` (figura 2.40, figura 2.41).

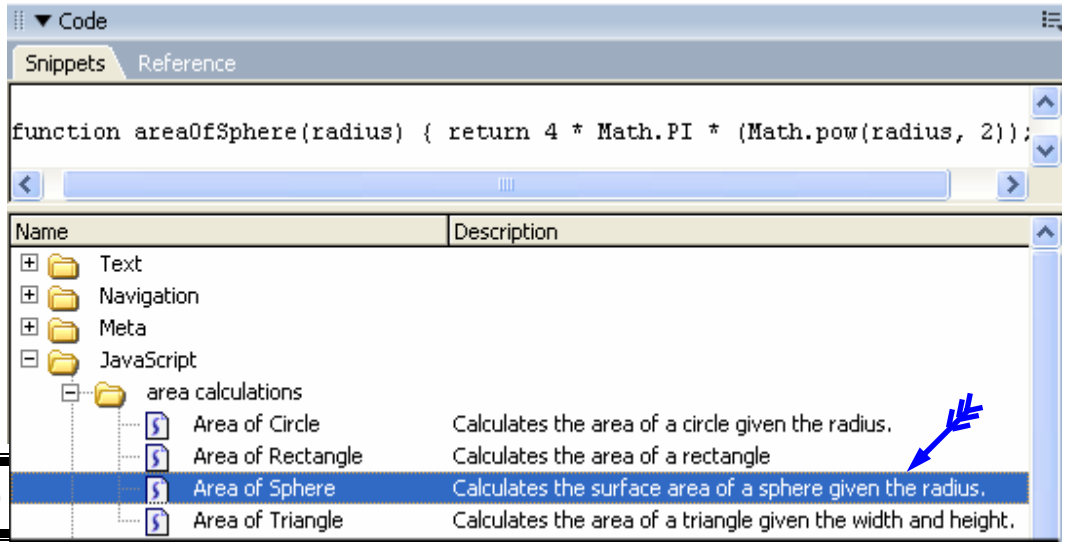


Figura 2.40

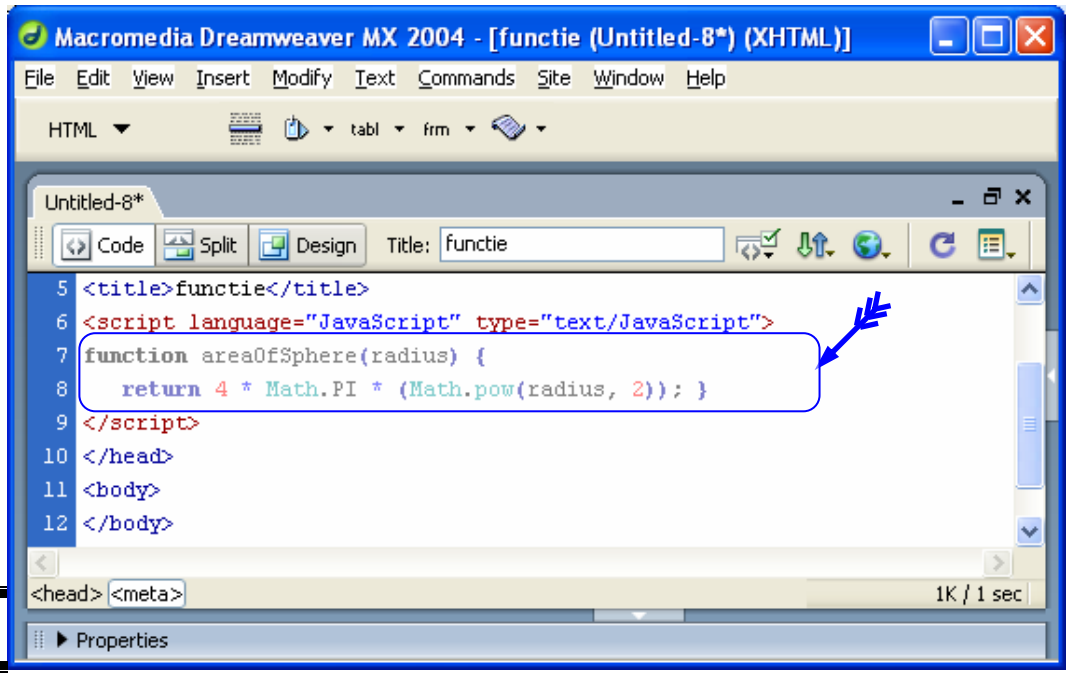


Figura 2.41

6. Apelaţi funcţia areaOfSphere (radius), pentru radius=3 (figura 2.42), în corpul documentului.

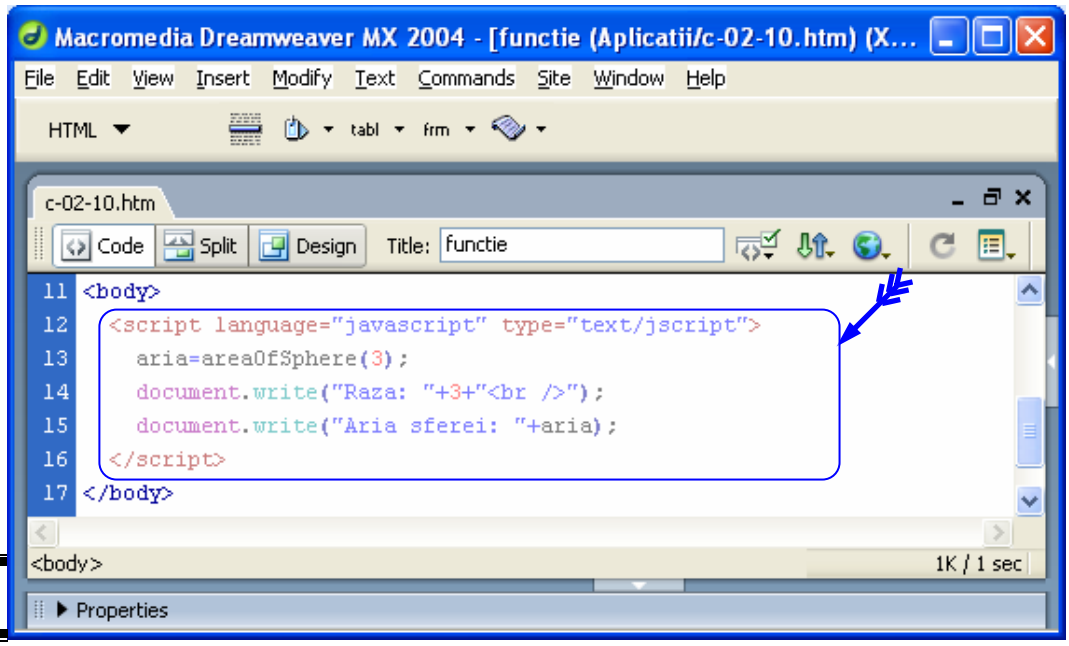


Figura 2.42

7. Vizualizaţi pagina Web într-un browser (figura 2.43).

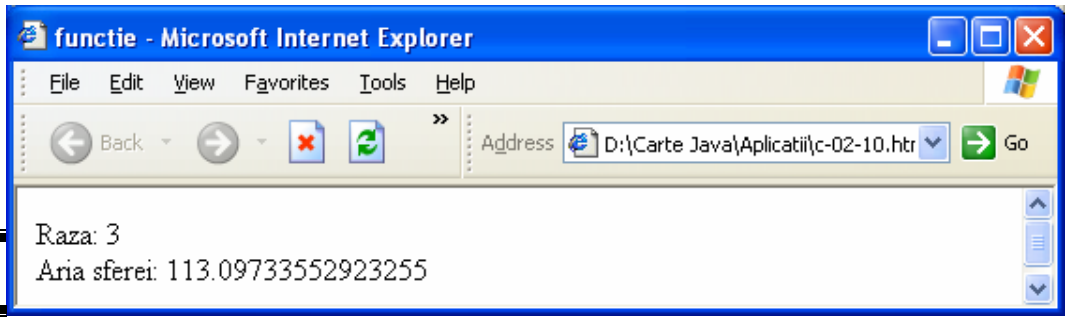


Figura 2.43

Remarcă. Pentru afişarea rezultatului cu două zecimale consultaţi Conversaţia 8.

EXEMPLUL 2 JAVASCRIPT

❑ De la problemă la script

Prima etapă pe care trebuie s-o parcurgeți în lungul drum al problemei către script o constituie definirea obiectivului script-ului. În consecință, luați o foaie de hârtie şi preț de câteva minute adunați cuvintele ... potrivite pentru a descrie pe scurt modul de rezolvare informatică a problemei. Altfel spus, redactați cu cuvinte obişnuite acest mic „caiet de sarcini". Succes!

Iată cum formulăm pe scurt problema pe care dorim să o rezolvăm cu JavaScript: creați o pagină Web care să afişeze aria unui rezervor sferic cu raza de 3m.

Ce problemă simplă, veți exclama! Într-adevăr, problema nu este dificilă, dar trebuie să recunoaştem că ceea ce pare simplu pentru unii este foarte complicat pentru alții.

Desigur, ea poate fi rezolvată foarte bine şi de către un elev de şcoală elementară sau chiar de un adult (părinții elevului!). Obiectivul nostru, însă este rezolvarea informatică a problemei utilizând limbajul JavaScript.

Remarci:

- ✓ Procesul de alcătuire a unui script, pe care il vom urma pe tot parcursul lucrării, constă din următoarele faze:
  - analiza problemei;
  - proiectarea programului (script-ului);
  - codificarea în limbajul JavaScript;
  - testare şi depanare.
- ✓ În cadrul fazei de analiză se definesc:
  - formatul datelor de ieşire;
  - tabela de variabile;
  - specificațiile de programare (descriere script, intrări, ieşiri, listă funcțiuni).
- ✓ Asigurându-ne că nu s-au strecurat greşeli în faza de analiză a problemei, putem aborda în continuare faza de proiectare a script-ului, utilizând ca instrumente de proiectare: pseudocodul, diagrama de structură a prelucrărilor etc.
- ✓ Pseudocodul este un limbaj independent de limbajul de programare (nu există un standard!). Limbajul pseudocod pe care vi-l propunem este alcătuit din câteva instrucțiuni standard ce definesc structurile algoritmice fundamentale (secvența, iterația, selecția) la care se adaugă instrucțiuni, în bună parte aflate la latitudinea celui care realizează documentația de analiză şi proiectare a programului.

❑ Analiza problemei

Înainte de a scrie un program (JavaScript) trebuie să cunoaştem şi să înțelegem foarte bine problema (Analiza problemei). În cazul nostru, problema este una dintre cele mai simple: *calculul ariei unui rezervor sferic atunci când se cunoaşte raza.*

În rezolvarea acestei probleme se urmăreşte crearea unei pagini Web, în care să se afişeze simplu, într-o formă deloc pretențioasă, cele două valori pentru rază (3) şi arie (vezi figura 2.44).

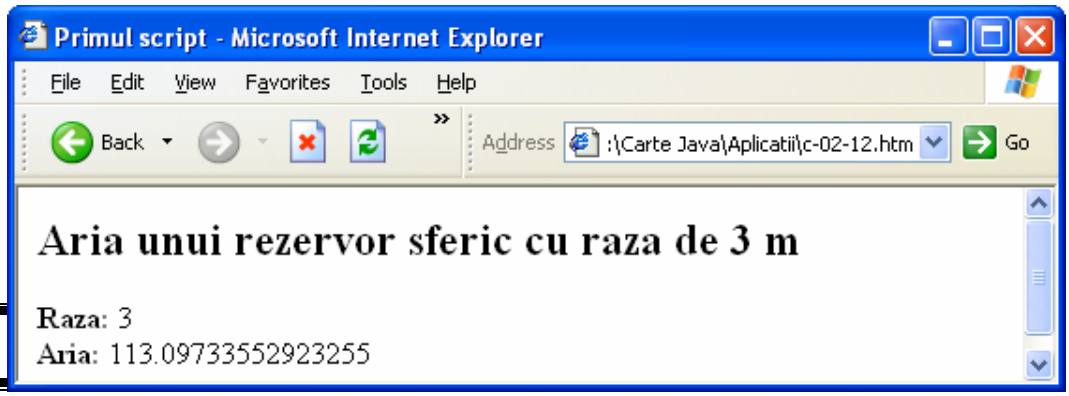


Figura 2.44

Remarcă. Ecranul din figura 2.44 reprezintă formatul datelor de ieşire.

Vă mai amintiţi cum se calculează aria unei sfere, căci despre ea este vorba. Se înmulţesc: patru cu trei şi paisprezece (Π) şi cu pătratul razei. Cam dificilă această exprimare, nu vi se pare? Să alegem pentru arie şi rază câte un nume de **variabilă** semnificativ, urmând apoi să codificăm în JavaScript formula de calcul prezentată ($A=4\Pi R^2$).

Tabela de variabile

În figura 2.45 sunt prezentate: variabilele de ieşire, variabilele de intrare şi variabilele de stare, structurate într-o **tabelă de variabile**, document care se realizează în faza de analiză structurată a problemei.

Tabela de variabile

Variabile de intrare	***Variabile de stare***	***Variabile de iesire***
	r: raza cercului	aria: aria rezervorului sferic

Figura 2.45

Remarcă. În situația în care variabilele de ieşire şi variabilele de intrare nu sunt suficiente, definiți un al treilea tip de variabilă: variabile de stare, care nu sunt nici de intrare şi nici de ieşire.

Descrierea script-ului JavaScript, precum şi lista principalelor funcțiuni sunt prezentate în figura 2.46.

Specificații de programare

Descrierea programului

Script-ul calculează şi afişează aria unui rezervor sferic cu raza de 3 m.

Intrări: -

Ieşiri: Aria rezervorului sferic cu raza de 3 m.

Lista de funcțiuni ale script-ului

1. Atribuie variabilei r (de stare) valoarea 3.
2. Calculează aria rezervorului sferic.
3. Afişează raza şi aria rezervorului sferic (aria).
4. Stop.

Figura 2.46

Asigurându-ne că nu s-au strecurat greşeli în faza de `analiz•` a problemei, putem aborda în continuare faza de `proiectare` a script-ului.

❑ **Proiectarea script-ului**

Pentru proiectarea script-ului vom folosi ca instrument de proiectare, pseudocodul, cu una din variantele prezentate mai jos:

- ✓ *Varianta 1* – scrierea în limbaj natural structurat;
- ✓ *Varianta 2* – scriere formalizată (apropiată de limbajul JavaScript).

Iată cum procedăm pentru a construi pseudocodul în ambele variante.

Varianta 1

În figura 2.47 se prezintă pseudocodul în limbaj natural structurat.

Pseudocodul (*Varianta 1*)

1. Atribuiți razei (r) valoarea 3.
2. Calculați aria (aria) rezervorului sferic.
3. Afişați raza (r)
4. Afişați rezultatul (aria).
5. Stop.

Figura 2.47

Remarcă. Prezentarea algoritmului în limbaj natural structurat are, după cum ați putut constata şi unele dezavantaje: lizibilitate redusă, exprimări lungi, greoaie etc.

Varianta 2

Varianta prezentării algoritmului de o manieră formalizată este de preferat celei narative (*Varianta 1*). Se preferă o formalizare a acțiunilor primitive (atribuie, calculează, afişează) care să pună în evidență atât variabilele de intrare/ieşire cât şi variabilele de stare.

În consecință, s-o folosim cât este cu putință pe aceasta.

În figura 2.48 se prezintă pseudocodul în manieră formalizată.

Pseudocodul (*Varianta 2*)

```
REZERVOR    BEGIN
                    r=3
                    aria=4π r^2
                    WRITE(r)
                    WRITE(aria)
REZERVOR    END
```

Figura 2.48

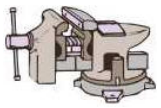
Cuvintele rezervate JavaScript

Anumite cuvinte nu pot fi utilizate ca nume de variabile, funcţii, obiecte şi metode. În figura 2.49 se prezintă lista cuvintelor rezervate.

break	in	false	this	void
continue	for	new	true	while
delete	function	null	var	typeof
with	else	return	if	abstract
default	implements	private	throw	boolean
do	import	protected	throws	byte
double	instanceof	public	translent	case
extends	int	short	try	catch
final	interface	static	char	finally
long	class	float	native	switch
goto	package	super	synchronized	const
enum	debugger	volatile	export	

Figura 2.49

Testați-vă cunoștințele



1. Identificaţi erorile de sintaxă din următoarele script-uri:
   - Figura 11.1 (Conversaţia 11);
   - Figura 11.2 (Conversaţia 11);
   - Figura 11.3 (Conversaţia 11).
2. Care sunt valorile speciale JavaScript?
3. Ce tip de date puteţi atribui unei variabile JavaScript?
4. Care este deosebirea dintre o variabilă locală şi o variabilă globală?
5. Ce execută prima dată un navigator?
   - un script din antet-ul paginii;
   - un script din corpul paginii;
   - un gestionar de evenimente plasat în tag-ul `<body>`.
6. O funcţie JavaScript:
   - acceptă parametrii;
   - returnează o valoare;
   - ambele variante.
7. Care sunt cele două constante booleene?
8. Cum definiţi o funcţie JavaScript?
9. Cum atribuiţi valori variabilelor?
10. Ce realizează funcţia `parseInt()`?
11. Care este semnificaţia cuvântului rezervat `return`?

Vizitați site-urile

- ✓ www.glscript.com
- ✓ www.toutJavascript.com
- ✓ www.javanett.com
- ✓ www.webcoder.com
- ✓ www.script-masters.com
- ✓ www.javascript.internet.com

- ✓ www.javascripts.com
- ✓ www.bratta.com
- ✓ www.javascript.com

Conversația 2
(continuare)

Operatorii JavaScript. Obiecte matematice

În această conversație:

Operatorii JavaScript

Pentru a crea programe utile JavaScript, trebuie să evaluaţi sau chiar să modificaţi datele pe care le prelucrează script-urile dumneavoastră. Instrumentele necesare pentru realizarea acestor operaţii se numesc *operatori*.

Operatorii [1] sunt simbolurile şi identificatorii care reprezintă fie felul în care sunt modificate datele, fie felul în care este evaluată o combinaţie de expresii. JavaScript recunoaşte atât operatorii binari (necesită existenţa a doi operanzi în expresie) cât şi operatorii unari (necesită existenţa unui singur operand în expresie).

Cei mai mulţi dintre operatorii JavaScript sunt recunoscuţi de către toate navigatoarele, dar ... mai există şi divergenţe!

În continuare, vom proceda la o clasificare a operatorilor din limbajul JavaScript urmând ca apoi să-i examinăm pe fiecare în parte (vezi figura 2.50).

Tipuri de operatori	***Operatorii JavaScript***
aritmetici	% (modulo); ++ (increment); – – (decrement); + (adunare); – (scădere); * (înmulţire); / (împărţire)
atribuire plus combinaţii (între operatorul de atribuire şi operatorii aritmetici)	= (atribuire simplă); += (atribuire cu adunare); –= (atribuire cu scădere); *= (atribuire cu înmulţire); /= (atribuire cu împărţire); %= (atribuire cu modulo)
relaţionali (de comparaţie)	< (mai mic); > (mai mare); <= (mai mic sau egal); >= (mai mare sau egal); == (egalitate); === (identitate); != (non egal); !== (non identic)
booleeni (logici)	&& (AND); || (OR); ^ (SAU EXCLUSIV); ! (NOT)
operatori pentru funcţii	(); , (virgula)
operatori pentru structuri de date	. (punctul); []
operatori condiţionali	?:
operatori pentru şiruri	toţi operatorii relaţionali (de comparaţie) şi operatorul de concatenare (+)
operatorul `typeof` operatorul `delete` operatorul `new` operatorul `void`	

Figura 2.50

Operatorii JavaScript sunt prezentaţi în detaliu în figura 2.51.

Operator	***Sintaxă***
`+`	`operand1 + operand2`

Adunare sau concatenare.

1. Tip de date: numeric.
   - ✓ Operator aritmetic.

Remarcă. Dacă unul din operatori este `NaN`, rezultatul este `NaN`.

2. Tip de date: alfanumeric (şir de caractere).
   - ✓ Operator de concatenare.

Remarcă. Operatorul + serveşte în general pentru efectuarea convertirii tuturor tipurilor de date în şiruri de caractere.

Exemplu:

```
<script>
   a=8;
   b=7;
   z=3+8;
   c=a+b;
   alert(z); // afişează 11
   alert(c); // afişează 15
</script>
```

```
<script>
   prenume="Vasile";
   nume="Adam";
   numecomplet="Domnul "+
               prenume+" "+nume;
   alert(numecomplet);
// afişează Domnul Vasile Adam
</script>
```

Figura 2.51

+=	operand1 += operand2

Adunare sau concatenare asociată unei atribuiri.

1. Tip de date: numeric.
   - ✓ Operator aritmetic şi de atribuire.

Remarcă. Echivalent cu: operand1= operand1 + operand2.

2. Tip de date: alfanumeric.
   - ✓ Operator de concatenare şi de atribuire.

Remarcă. Conţinutul variabilei situate la stânga operatorului este completat cu data alfanumerică din dreapta operatorului.

Exemplu:

```
<script>
  a=3;
  a+=6;
  alert(a);
   //afişează 9
</script>
```

```
<script>
  numecomplet="Dumitrescu";
  nume="Vasile";
  numecomplet+=nume;
  alert(numecomplet);
 // afişează Dumitrescu Vasile
</script>
```

++	++ operand operand++

Incrementare.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic.

Remarci:
- ✓ Pre-incrementare (++ operand): valoarea variabilei este mărită cu 1, apoi este utilizată.
- ✓ Post-incrementare (operand ++): valoarea variabilei este utilizată apoi este mărită cu 1.
- ✓ Este imposibil de a combina post-incrementarea cu pre-incrementarea.
- ✓ Acest operator se aplică numai variabilelor. Este foarte des utilizat în bucle.
- ✓ Este echivalent cu: operand1= operand1 + operand2.
- ✓ ++x++ returnează un mesaj de eroare.
- ✓ x++ este echivalent cu: x=x+1.

Exemplu:

```
<script>
 //post-incrementare
   a=5;
   alert(a++);
 // afişează 5
  alert(a);
 //afişează 6
</script>
```

```
<script>
 //pre-incrementare
   a=5;
   alert(++a);
 // afişează 6
  alert(a);
 //afişează 6
</script>
```

```
<script>
 //post-incrementare  într-o buclă

   for(i=0;i<5;i++){
   document.write(i);
 // afişează 0 1 2 3 4
 }
</script>
```

–	operand1 – operand2

Scădere sau negare.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic.

2. Operator de negare.

Converteşte o valoare pozitivă într-o valoare negativă.

Figura 2.51
(continuare)

Exemplu:

```
<script>
  a=20;
  b=13;
  c=a-b;
  d=8-2;
  alert(d); // afişează 6
  alert(c); // afişează 7
</script>
```

```
<script>
  a=-7;
  a=-a;
  alert(a);
// afişează 7
</script>
```

`-=` `operand1 -= operand2`

Tipul de date: numeric.
- ✓ Operator aritmetic şi de afectare.

Remarcă. Este echivalent cu: operand 1= operand 1 – operand 2.

Exemplu:

```
<script>
  a=4;
  a-=6;
  alert(a); // afişează -2
</script>
```

`- -` `operand - -`
`- - operand`

Decrementare.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic.

Remarci:
- ✓ Pre-decrementare (-- operand): valoarea variabilei este diminuată cu 1 şi apoi este evaluată.
- ✓ Post-decrementare (operand --): valoarea variabilei este evaluată apoi este diminuată cu 1.
- ✓ Acest operator se aplică numai variabilelor. Este des utilizat în bucle.
- ✓ Este echivalent cu: operand1= operand1 – 1.
- ✓ x– – este echivalent cu x=x–1.

Exemplu:

```
<script>
//post-decrementare
 a=5;
 alert(a--);//afişează 5
 alert(a);// afişează 4
</script>
```

```
<script>
//pre-decrementare
 a=5;
 alert(--a);  //afişează 4
 alert(a)// afişează 4
</script>
```

```
<script>
//post-decrementare  în
bucla for
for(a=7;a<=1; a--){
  document.write(a);
// afişează 6 5 4 3 2 1
</script>
```

`*` `operand1 * operand2`

Înmulţire.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic.

Exemplu:

```
<script>
  a=7;
  b=8;
  c=3*6;
  d=a*b;
  alert(c); // afişează 18
  alert(d); // afişează 56
</script>
```

Figura 2.51
(continuare)

`*=` `operand1 *= operand2`

Înmulţire şi atribuire.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic şi de atribuire.

Remarcă. Este echivalent cu: operand1= operand1 * operand2.

Exemplu:

```
<script>
  a=2;
  a*=5;
  alert(a); // afişează 10
</script>
```

`/` `operand1 / operand2`

Împărţire.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic.

Remarci:
- ✓ Dacă unul din operanzi este `NaN`, rezultatul este `NaN`.
- ✓ Împărţirea 0/0 dă `NaN`.
- ✓ Împărţirea 0/valoare_finită dă 0.

Exemplu:

```
<script>
  a=12; b=3;
  c=5/2; d=a/b;
  alert(c); // afişează 2.5
  alert(d); // afişează 4
</script>
```

`/=` `operand1 /= operand2`

Împărţire şi atribuire.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic şi de atribuire.

Remarcă. Este echivalent cu: operand1= operand1 / operand2.

Exemplu:

```
<script>
  a=5;
  a/=2;
  alert(a);
// afişează 2.5
</script>
```

← echivalent cu →

```
<script>
  a=5;
  a=a/2;
  alert(a);
// afişează 2.5
</script>
```

`%` `operand1 % operand2`

Modulo.

1. Tipul de date: numeric.
   - ✓ Operator aritmetic.

Remarcă. Calculează restul întreg al împărţirii întregi a primului număr prin cel de-al doilea număr.

Exemplu:

```
<script>
  a=18;
  a=a%5;
  alert(a); // afişează 3
</script>
```

Figura 2.51
(continuare)

	`%=` `operand1 %= operand2`
	Modulo şi afectare. 1. Tipul de date: numeric. ✓ Operator aritmetic şi de afectare. **Remarcă**. Este echivalent cu: operand1= operand1 % operand2.
Exemplu:	<script> a=18; a%=5; alert(a);// afişează 3 </script> ← echivalent cu → <script> a=18; a=a%5; alert(a); // afişează 3 </script>
	`=` `Variabila = Valoare`
	Atribuire. 1. Tipul de date: numeric, alfanumeric şi logic. **Remarcă**. A nu se confunda cu operatorul ==.
Exemplu:	<script> a=20; nume="Droopy"; b=a+3; //b are valoarea 23 </script> <script> //atribuire multiplă var a=b=c=13; // cele trei variabile vor avea aceeaşi valoare: 13 </script>
	`<` `operand1 < operand2`
	Strict mai mic. 1. Tipul de date: numeric, alfanumeric şi logic. ✓ Operator relaţional (de comparare). **Remarci**: ✓ Se returnează `TRUE`, dacă rezultatul evaluării condiţiei este adevărat. ✓ Se returnează `undefined` dacă una din valorile care se compară este `NaN`. ✓ În cazul comparării valorilor alfanumerice se ia în considerare codul ISO.
	`<=` `operand1 <= operand2`
	Mai mic sau egal. 1. Tipul de date: numeric, alfanumeric şi logic. ✓ Operator relaţional (de comparare). **Remarci**: ✓ Se returnează `TRUE` dacă rezultatul evaluării condiţiei (operand1 <= operand2) este adevărat. ✓ Se returnează `undefined` dacă una din valorile care se compară este `NaN`. ✓ În cazul comparării valorilor alfanumerice se ia în considerare codul ISO.
Exemplu:	<script> a=6; b=6; document.write(a<=b); //afişează true nume1= "MARK"; nume2="SPENCER"; document.write(" "+nume1<=nume2); // afişează true </script>

Figura 2.51
(continuare)

	`>` `operand1 > operand2`
	Strict mai mare. 1. Tipul de date: numeric, alfanumeric şi logic. ✓ Operator relaţional (de comparare). **Remarci**: ✓ Se returnează `TRUE` dacă rezultatul evaluării condiţiei (operand1>operand2) este adevărat. ✓ Se returnează `undefined` dacă una din valorile care se compară este `NaN`. ✓ În cazul comparării valorilor alfanumerice se ia în considerare codul ISO.
Exemplu:	<script> a=6; b=6; document.write(a>b); //afişează false nume1= “MICIURIN”; nume2=”POLEVOI”; document.write(“ ”+nume1>nume2); // afişează false; </script>
	`>=` `operand1 >= operand2`
	Mai mare sau egal. 1. Tipul de date: numeric, alfanumeric şi logic. ✓ Operator relaţional (de comparare). **Remarci**: ✓ Se returnează `TRUE` dacă rezultatul evaluării condiţiei (operand1 >= operand2) este adevărat. ✓ Se returnează `undefined` dacă una din valorile care se compară este `NaN`. ✓ În cazul comparării valorilor alfanumerice se ia în considerare codul ISO.
Exemplu:	<script> a=7; document.write(a>=7); //afişează true nume1= “M1”; nume2=”P1”; document.write(“ ”+nume1>=nume2); // afişează false </script>
	`==` `operand1 == operand2`
	Egal cu. 1. Tipul de date: numeric, alfanumeric şi logic. ✓ Operator relaţional (de comparare). **Remarci**: ✓ Se returnează `TRUE` dacă rezultatul evaluării condiţiei (operand1 este egal cu operand2) este adevărat. ✓ Se returnează `undefined` dacă una din valorile care se compară este `NaN`. ✓ În cazul comparării valorilor alfanumerice se ia în considerare codul ISO. ✓ Compararea unei valori numerice (de exemplu, 7) cu o aceeaşi valoare alfanumerică (de exemplu, ”7”) returnează TRUE.
Exemplu:	<script> a=6; b=7; nume=”MIRCEA”; document.write(nume==a); // afişează false; document.write(“ ”); document.write(a==’6’); // afişează true; document.write(“ ”); document.write(a==b’); // afişează false; </script>
	`===` `operand1 === operand2`

Figura 2.51
(continuare)

Identic cu.

1. Tipul de date: numeric, alfanumeric şi logic.
   - ✓ Operator relaţional (de comparare): egalitate de valori şi de tip de date.

Remarcă. Se returnează `TRUE` dacă rezultatul evaluării condiţiei (operand1 este egal cu operand2 şi de acelaşi tip de date) este adevărat.

Exemplu:

```
<script>
  a="7"; b=7;
  nume1="MAXITAXI";
  document.write(nume1==a); // afişează false;
  document.write("<br />"+(a===b)); // afişează false
  document.write("<br />"+(nume1==="maxitaxi"); // afişează false;
  document.write("<br />"+(nume1==="MAXITAXI")); // afişează true;
</script>
```

`!=` `operand1 != operand2`

Diferit de.

1. Tipul de date: numeric, alfanumeric şi logic.
   - ✓ Operator relaţional (de comparare): non-egalitate de valori.

Remarcă. Se returnează `TRUE` dacă rezultatul evaluării condiţiei (operand1 este diferit de operand2) este adevărat.

Exemplu:

```
<script>
  a="7"; b=7; c=8; nume1="Maxim";
  document.write(nume1!=a); // afişează true;
  document.write("<br />");
  document.write(a!=b); // afişează false;
  document.write("<br />");
  document.write(nume1!="maxim"); // afişează true;
  document.write("<br />"+(nume1!="Maxim")); // afişează false
  document.write("<br />"+(c!=a)); // afişează true
</script>
```

`!==` `operand1 !== operand2`

Non identic cu.

1. Tipul de date: numeric, alfanumeric şi logic.
   - ✓ Operator relaţional (de comparare): non-egalitate de valoare şi de tip de date.

Remarcă. Se returnează `TRUE` dacă rezultatul evaluării condiţiei (`operand1` nu este egal şi nici de acelaşi tip cu `operand2`) este adevărat.

Exemplu:

```
<script>
  a="7"; b=7; nume1="Maxim";
  document.write(nume1!==a); // afişează true;
  document.write("<br />");
  document.write(a!==b); // afişează true;
  document.write("<br />");
  document.write(nume1!=="maxim"); // afişează true;
  document.write("<br />"+(nume1!=="Maxim")); // afişează false
</script>
```

Figura 2.51 *(continuare)*

`&&` `operand1 && operand2`

ŞI.

Remarci:
- ✓ Operator logic ŞI (`AND`).
- ✓ Returnează `TRUE` dacă cele două expresii returnează valoarea `TRUE`.
- ✓ Acest operator este foarte des utilizat în bucle (`for, while`) şi în teste (`if ... else`).

Exemplu:
```
<script>
  a=4; b=6; nume1="Maxim";
  document.write((a>2)&&(b==6)); // afişează true;
  document.write("<br />");
  document.write((a>2)&&(b==nume1)); // afişează false;
</script>
```

Exemplu:
```
<script>
  a=4;b=6;nume1="Maxim";
  if((a>2)&&(b==6)){
     document.write("a este mai mare ca 2");
     document.write(" şi b este egal cu 6");
   }
</script>
```

`||` `operand1 || operand2`

SAU (OR).

Remarci:
- ✓ Operator logic SAU (`OR`).
- ✓ Returnează `TRUE` dacă una din cele două expresii returnează valoarea `TRUE`.
- ✓ Acest operator este foarte des utilizat în bucle (`for, while`) şi teste (`if ... else`).

Exemplu:
```
<script>
  a=4; b=6; nume1="Maxim";
  document.write((a>2)||(b==6)); // afişează true
  document.write("<br />");
  document.write((a>2)||(b==nume1)); // afişează false
</script>
```

Exemplu:
```
<script>
  a=4;b=6;nume1="Maxim";
  if((a>2)||(b==6)){
     document.write("a este mai mare ca 2");
     document.write(" sau b este egal cu 6");
   }
</script>
```

`,` `(virgulă)`

Operator care permite mai multor instrucţiuni de a fi executate ca şi când ar fi o singură instrucţiune.

Remarci:
- ✓ Valoarea returnată este cea care aparţine ultimei instrucţiuni.
- ✓ Acest operator este utilizat de asemenea pentru transferarea mai multor parametrii unei funcţii.

Figura 2.51 *(continuare)*

Exemplu:
```
<script>
```

```
var a,b,c,d;
 a=(b=5,c=6,d=7);
 document.write("a="+a); // afişează a=7;
 document.write("<br />");
 document.write("b="+b); // afişează b=5;
 document.write("<br />");
 document.write("c="+c); // afişează c=6;
 document.write("<br />");
 document.write("d="+d); // afişează d=7;
</script>
```

?:	condiţie1 ? cod1:cod2

Condiţie.

1. Tipul de date: numeric, alfanumeric şi logic.
   - ✓ Operator de test (condiţie).

Remarci:

- ✓ Este utilizat pentru a crea un test rapid sub forma: expresielogică ? DacăDa: DacăNu.
- ✓ Expresielogică este o expresie care returnează `TRUE` sau `FALSE`.
- ✓ DacăDa: instrucţiune care va fi executată dacă rezultatul evaluării expresielogică este `TRUE`.
- ✓ DacăNu: instrucţiune care va fi executată dacă rezultatul evaluării expresielogică este `FALSE`.

Exemplu:

```
<script>
 a=2;
 (a>10)?alert("adevarat"):alert("false");
</script>
```

← echivalent cu →

```
<script>
 a=2;
 if(a>10)
   alert("adevarat");
 else
   alert("false");
</script>
```

delete	delete obiect delete obiect.Proprietate delete Array[*Index*]

Ştergere.

1. Tipul de date: obiect.
   - ✓ Operator de ştergere a proprietăţii obiectului.

Remarci:

- ✓ Şterge o proprietate a unui obiect creat cu `new`.
- ✓ Şterge un element al unei matrici (`Array`) indicând numărul său de ordine (primul element este de rang zero).

Exemplu:

```
<script>
 //ştergerea unui element al unei matrici
 var v=new Array(1,2,3,7);
 document.write(v);
 //afişează 1,2,3,7
 delete v[2];
 //este şters al 3-lea element
 document.write(v);
 //afişează 1,2,7
</script>
```

```
<script>
 azi=new Date();
 //ştergerea unui obiect
 x=delete azi;
 // x ia valoarea true şi azi este undefined
</script>
```

Figura 2.51 *(continuare)*

Exemplu:

```
<script>
 //Ştergerea unei proprietăţi a unui obiect
```

```
a=new Number();
a.valoare=13;
x=delete a.valoare;
//x ia valoarea true şi a.valoare este nedefinit
</script>
```

`new`	`obiect= new Constructor`

Nou.

1. Tipul de date: obiect.
   - ✓ Operator de creare a unui nou obiect.

Remarci:

- ✓ Creează o nouă instanţă a unui obiect cu ajutorul unui constructor.
- ✓ `new` este utilizat pentru crearea instanţelor următoarelor obiecte: `Array; Boolean; Date; Function, Number; Object; RegExp; String`.

Exemplu:

```
<script>
  //crearea obiectului astăzi
  var astazi=new Date();
  //crearea obiectului matrice
  var matrice=new Array();
  //crearea obiectului Object
  var obiect=new Object();
  obiect.prenume="stejărel"
</script>
```

`. (punct)`	`obiect.Proprietate` `obiect.Metoda` `obiect.Eveniment`

Punctul (.) este utilizat pentru a putea accesa proprietăţi, metode sau evenimente ale unui obiect.

1. Tipul de date: obiect.
   - ✓ Operator de acces.

Exemplu:

```
<script>
  //acces la metoda write() a obiectului document
  document.write("Au înnebunit salcâmii!");
</script>
```

`typeof`	`typeof(operand)` `typeof operand`

Tipul de date.

Remarci:

- ✓ Returnează sub formă de şir de caractere tipul de date al obiectului sau al variabilei indicate.
- ✓ În tabelul 1 se prezintă pe două coloane: tipul de dată şi şirul de caractere returnat de Typeof.

Figura 2.51
(continuare)

Tabelul 1

Tabelul 1

Tipul de dată	*Şirul de caractere returnat de typeof*
logic	boolean
număr	number
infinit	number
şir de caractere	string
obiect	object
funcţie	function
nedefinit	undefined
null	object

Exemplu:
```
<script>
  a=7; nume="salcam";
  document.write(typeof a); //afişează number
  document.write("<br />"+typeof(nume)); // afişează string
</script>
```

`void` `void Expresie` `void(Expresie)`

Nedefinit. Evaluează o expresie şi returnează ca rezultat `undefined`.

Exemplu:
```
<body>
  <a href="JavaScript: void(alert('Au innebunit salcamii!'))">
  Iar au innebunit salcamii</a>
</body>
```

`( )` `Function(Argumente)`

Utilizat pentru invocarea metodei unui obiect sau parametrii unei funcţii.

1. Tipul de date: obiect.
   - ✓ Operator de invocare.

Exemplu:
```
<script>
  /*Argumentul „Bine ati venit!" este transmis metodei write() a obiectului
    Document*/
  document.write("Bine ati venit!");
  /*argumentele 8,10 sunt transmise functiei media()
  media(8,10);
</script>
```

`[]` `matrice[]` `obiect["Proprietate"]`

Utilizat pentru accesarea unei proprietăţi a unui obiect sau a unui element al unei matrici.

1. Tipul de date: obiect.
   - ✓ Operator de matrice.

Exemplu:
```
<script>
  //acces la elementele unei matrici
  var matrice=[7,5,3,1];
  document.write(matrice[0]);
  document.write("<br />");
  //afişează 7
  document.write(matrice[2]);
  document.write("<br />");  //afişează 3
</script>
```

Figura 2.51 *(continuare)*

	! !Operand
	Operator logic (NOT). Returnează TRUE dacă data este FALSE şi invers.
Exemplu:	<script> a=true; alert(!a); //afişează FALSE <script>
Exemplu:	<script> a=5;b=6; alert(!(a>b)); //afişează TRUE </script>
	^ operand1 ^ operand2
	Operator logic SAU exclusiv. Returnează TRUE dacă una şi numai una din expresii returnează TRUE.
Exemplu:	<script> a=4; b=6; if((a>2)^(b==6)){ document.write("sau a>2 sau b=6"); } </script>

Figura 2.51 *(continuare)*

Asociativitatea şi prioritatea operatorilor

Asociativitatea indică sensul (→ sau ←) în care expresia care conţine operatorul este evaluată.

Prioritatea (precedenţa) indică ordinea în care expresiile sunt evaluate.

În figura 2.52 ([2]) sunt prezentaţi toţi operatorii JavaScript cu precizarea asociativităţii şi priorităţii acestora.

Rang de prioritate	***Operatori***	***Asociativitate***	***Semnificaţie***
1	.	→	acces la proprietăţi
2	[]	→	acces la matrici (arrays)
3	()	→	regrupare sau funcţie
4	++	←	incrementare
5	– –	←	decrementare
6	–	←	negare aritmetică

Figura 2.52

7	!	←	NOT logic
8	delete	←	ştergerea unei proprietăţi sau a valorii unei matrici
9	new	←	crearea unui obiect
10	typeof	←	determinarea tipului de date
11	void	←	evaluarea fără returnarea valorii
12	* / %	→	înmulţire, împărţire, modulo
13	+ −	→	adunare, scădere
14	+	→	concatenare
15	< <=	→	mai mic, mai mic sau egal
16	> >=	→	mai mare, mai mare sau egal
17	==	→	egal
18	!=	→	diferit
19	===	→	identic (egalitate de valori şi de tip)
20	!==	→	non-identic
21	&&	→	ŞI
22	||	→	SAU
23	?:	←	condiţie
24	=	→	afectare (atribuire/asignare)
25	*= /= %= +=−=	←	afectare cu calcul
26	,	→	evaluare multiplă

Figura 2.52 *(continuare)*

Remarci:

- ✓ Expresiile sunt evaluate în funcţie de prioritatea operatorilor.
- ✓ Parantezele au o prioritate foarte ridicată. Expresiile din paranteze sunt evaluate primele. Dacă există mai multe nivele de paranteze, expresiile sunt evaluate începând cu parantezele aflate cel mai în interior.
- ✓ Asociativitatea joacă un rol, de exemplu, în cazul combinării unei adunări şi unei concatenări (vezi figura 2.53).

```
<script>
   document.write((7+9+"abc")); //afişează 16abc
   document.write("<br />");
   document.write(("abc"+7+9)); //afişează abc79
   document.write("<br />"));
</script>
```

Figura 2.53

Cu siguranţă că veţi avea nevoie în programele dumneavoastră de formule matematice care să nu folosească doar simple adunări şi înmulţiri. În acest caz, obiectele matematice: `Math`, `Number`, `Boolean` vă pot fi de un real folos.

Ele vă permit să accesaţi constante (π, true, false) şi să executaţi diferite funcţii matematice (sqrt(); exp() etc.).

`Math` este un obiect predefinit al limbajului JavaScript care conţine numeroase constante (PI, E, LN10, LN2, SQRT2 etc.) şi funcţii (abs(); floor(); pow(); sqrt() etc.).

Nu este nevoie să creaţi obiectul `Math`: el se creează în mod automat de către navigator (există în toate programele JavaScript). Proprietăţile obiectului `Math` sunt constante matematice iar metodele sale sunt funcţii matematice.

`Number`, cel de-al doilea obiect matematic, este un obiect predefinit al limbajului JavaScript pe care îl veţi găsi util atunci când trebuie să accesaţi anumite constante: cel mai mare şi cel mai mic număr care poate fi reprezentat; plus şi minus infinit; NaN (Not a Number). JavaScript reprezintă aceste valori ca proprietăţi ale obiectului `Number`.

Pentru a crea un nou obiect `Number`, utilizaţi funcţia specială numită constructor, `Number()` şi cuvântul cheie `new` (figura 2.54).

Figura 2.54

```
<script>
   var numar_clienti=1234; //declarare implicita
   var numar_clienti=new Number(1234); //declarare explicita
</script>
```

`Boolean` este un alt obiect predefinit al limbajului JavaScript pe care îl veţi găsi util atunci când doriţi să transformaţi o valoare non-booleană într-o valoare booleană (`true` sau `false`).

Pentru a crea un nou obiect `Boolean` utilizaţi constructorul `Boolean()` şi cuvântul cheie `new`.

Obiectul Math

Fişa obiectului `Math` este prezentată în figura 2.55.

Fişa obiectului	***Math***
Cum creăm obiectul?	`Se creează automat de către navigator.`
Proprietăţi:	`E, LN10, LN2, LOG10E, LOG2E, PI, SQRT1_2, SQRT2`
Metode:	`abs(), acos(), asin(), atan(), ceil(), cos(), exp(), floor(), log(), max(), min(), pow(), random(), round(), sin(), sqrt(), tan()`
Gestionarii de evenimente:	–

Figura 2.55

Proprietăţile obiectului Math

Proprietăţile obiectului `Math` sunt prezentate în detaliu în figura 2.56.

	Proprietate	***Sintaxa***
	`E`	`Math.E`
	Constanta lui Euler.	
Exemplu:	<script> document.write(Math.E); //afişează 2.71821828459045 </script>	
	`LN10`	`Math.LN10`
	Logaritm natural de 10.	
Exemplu:	<script> document.write(Math.LN10); //afişează 2.30258509294046 </script>	
	`LN2`	`Math.LN2`
	Logaritm natural de 2.	
Exemplu:	<script> document.write(Math.LN2); //afişează 0.6931471805599453 </script>	
	`LOG10E`	`Math.LOG10E`
	Logaritm în bază 10 din e.	
Exemplu:	<script> document.write(Math.LOG10E); //afişează 0.4342944819032518 </script>	
	`LOG2E`	`Math.LOG2E`

Figura 2.56

Logaritm în baza 2 din e.

Exemplu:
```
<script>
 document.write(Math.LOG2E);
  //afişează 1.4426950408889633
</script>
```

PI	Math.PI

Constanta PI.

Exemplu:
```
<script>
 document.write(Math.PI);
  //afişează 3.141592653589793
</script>
```

SQRT1_2	Math.SQRT1_2

$\sqrt{1/2}$.

Exemplu:
```
<script>
 document.write(Math.SQRT1_2);
  //afişează 0.7071067811865476
</script>
```

SQRT2	Math.SQRT2

$\sqrt{2}$.

Exemplu:
```
<script>
 document.write(Math.SQRT2);
  //afişează 1.4142135623730951
</script>
```

Figura 2.56 *(continuare)*

Metodele obiectului Math

Metodele cele mai frecvent folosite ale obiectului `Math` sunt prezentate în detaliu în figura 2.57.

Metodă	***Sintaxă***
`abs()`	`Math.abs(`*`ValoareNumerică`*`)`

Valoare absolută.

Remarcă. În Tabelul 2 se prezintă rezultatele funcţiei `abs()` pentru câteva argumente speciale.

Tabelul 2

Argument	***Rezultat***
abs(0)	0
abs(NaN)	NaN
abs(-Infinity)	+Infinity

Exemplu:
```
<script>
  a=-57; b=85;
  rez1=Math.abs(a);
  rez2=Math.abs(b);
  document.write("abs("+a+")="+rez1);
  //afişează abs(-57)=57
```

Figura 2.57

```
document.write("<br />");
document.write("abs("+b+")="+rez2);
//afişează abs(85)=85
</script>
```

`acos()` `Math.acos(ValoareNumerică)`

Arc cosinus.

Remarcă. Pentru valori ale argumentului: >1; < –1; `NaN` rezultatul este `NaN`.

Exemplu:

```
<script>
a=0.9965;
rezultat=Math.acos(a);
document.write("acos("+a+")="+rezultat);
</script>
```

`asin()` `Math.asin(ValoareNumerică)`

Arc sinus.

Remarcă. În Tabelul 3 se prezintă rezultatele funcţiei `asin()` pentru câteva argumente speciale.

Tabelul 3

Argument	***Rezultat***
0	0
>1	NaN
<-1	NaN
NaN	NaN

Exemplu:

```
<script>
a=0.9965;
rezultat=Math.asin(a);
document.write("asin("+a+")="+rezultat);
</script>
```

`atan()` `Math.atan(ValoareNumerică)`

Arc tangentă.

Exemplu:

```
<script>
a=0.9965;
rezultat=Math.atan(a);
document.write("atan("+a+")="+rezultat);
</script>
```

`ceil()` `Math.ceil(ValoareNumerică)`

Număr întreg superior cel mai apropiat.

Remarcă. În Tabelul 4 se prezintă câteva argumente speciale şi rezultatul corespunzător.

Tabelul 4

Argument	***Rezultat***
+Infinity	+Infinity
Între -1 şi 0	0
-Infinity	-Infinity
0	0
NaN	NaN

Figura 2.57
(continuare)

Exemplu:

```
<script>
a=-4.223;
rezultat=Math.ceil(a);
document.write("ceil("+a+")="+rezultat);
//afişează ceil(-4.223)=-4
</script>
```

`cos()` `Math.cos(ValoareNumerică)`

Cosinus.

Exemplu:
```
<script>
  a=0.9965;
  rezultat=Math.cos(a);
  document.write("cos("+a+")="+rezultat);
</script>
```

`exp()` `Math.exp(ValoareNumerică)`

Exponenţiala.

Exemplu:
```
<script>
  a=5;
  rezultat=Math.exp(a);
  document.write("exp("+a+")="+rezultat);
</script>
```

`floor()` `Math.floor(ValoareNumerică)`

Număr întreg inferior cel mai apropiat.

Exemplu:
```
<script>
  a=-4.223;
  rezultat=Math.floor(a);
  document.write("floor("+a+")="+rezultat);
 //afişează floor(-4.223)=-5
</script>
```

`log()` `Math.log(ValoareNumerică)`

Logaritm.

Exemplu:
```
<script>
  a=5;
  rezultat=Math.log(a);
  document.write("log("+a+")="+rezultat);
  //afişează 1.609437912431002
</script>
```

`max()` `Math.max(ValoareNumerică1, ValoareNumerică2)`

Valoare maximă.

Remarcă. În Tabelul 5 se prezintă câteva argumente speciale ale metodei şi rezultatele corespunzătoare.

Tabelul 5

Argument 1	***Argument 2***	***Rezultat***
Orice valoare	NaN	NaN
NaN	Orice valoare	NaN

Figura 2.57 *(continuare)*

Exemplu:
```
<script>
  a=7; b=13.336;
  rezultat=Math.max(a,b);
  document.write("max("+a+","+b+")="+rezultat);
  //afişează max(7,13.336)=13.336
</script>
```

`min()` `Math.min(ValoareNumerică1, ValoareNumerică2)`

Valoarea minimă.

Exemplu:

```
<script>
  a=7; b=13.336;
 rezultat=Math.min(a,b);
  document.write("min("+a+","+b+")="+rezultat);
  //afişează min(7,13.336)=7
</script>
```

`pow()` `Math.pow(ValoareNumerică1, valoareNumerică2)`

Returnează ValoareNumerică1 la puterea ValoareNumerică2**.**

Remarcă. În Tabelul 6 se prezintă câteva argumente speciale ale metodei şi rezultatele corespunzătoare.

Tabelul 6

Argument 1	***Argument 2***	***Rezultat***
+0	<0	+Infinity
+Infinity	<0	0
+Infinity	>0	+Infinity
-Infinity	<0	0
-Infinity	întreg par>0	+Infinity
-Infinity	întreg impar>0	-Infinity
0	>0	0
1	orice valoare	1
Orice valoare	NaN	NaN
NaN	Diferit de zero	NaN

Exemplu:

```
<script>
  a=3; b=2;
 rezultat=Math.pow(a,b);
  document.write("pow("+a+","+b+")="+rezultat);
  //afişează pow(3,2)=9
</script>
```

`random()` `Math.random()`

Numere aleatoare. Returnează un număr aleator cuprins între 0 şi 1.

Exemplu:

```
<script>
  /*generează 3 numere aleatoare
cuprinse între 0 şi 1*/
for(i=0;i<3;i++){
 a=Math.random();
 document.write(a+"<br />");
}
</script>
```

```
<script>
  /*generează 4 numere aleatoare
cuprinse între 0 şi 25*/
for(i=0;i<5;i++){
 a=Math.round(Math.random()*25);
 document.write(a+"<br />");
}
</script>
```

Figura 2.57 *(continuare)*

Exemplu:

```
<script>
  /*generează 5 numere aleatoare cuprinse între 12 şi 48*/
var n=12;  var m=48;
for(i=0;i<5;i++){
 a=Math.round(Math.random()*(m-n)+n);
 document.write(a+"<br />");
```

```
}
</script>
```

`round()` `Math.round(ValoareNumerică)`

Returnează valoarea numerică rotunjită la întregul inferior.

Remarcă. În Tabelul 7 se prezintă câteva argumente speciale ale metodei şi rezultatele corespunzătoare.

Tabelul 7

Argument	***Rezultat***
+Infinity	+Infinity
0	0
NaN	NaN

Exemplu:

```
<script>
 x=-5.88; y=8.336;
 rezultat1=Math.round(x);
 rezultat2=Math.round(y);
 rezultat3=Math.round(z);
 document.write("round("+x+")="+rezultat1);
 //afişează round(-5.88)=-6
 document.write("round("+y+")="+rezultat2);
 //afişează round(8.336)=8
</script>
```

`sin()` `Math.sin(ValoareNumerică)`

Sinus.

Exemplu:

```
<script>
  a=0.9965;
  rezultat=Math.sin(a);
  document.write("sin("+a+")="+rezultat);
  //afişează sin(0.9965)=0.8395747765937453
</script>
```

`sqrt()` `Math.sqrt(ValoareNumerică)`

Rădăcina pătrată.

Remarcă. În Tabelul 8 se prezintă câteva argumente speciale ale metodei şi rezultatele corespunzătoare.

Tabelul 8

Argument	***Rezultat***
+Infinity	+Infinity
0	0
<0	NaN
NaN	NaN

Exemplu:

```
<script>
  a=25;
  rezultat=Math.sqrt(a);
  document.write("sqrt("+a+")="+rezultat);  //afişează sqrt(25)=5
</script>
```

Figura 2.57 *(continuare)*

`tan()` `Math.tan(ValoareNumerică)`

Tangentă

Exemplu:

```
<script>
  a=0.9965;
```

Figura 2.57 *(continuare)*

```
rezultat=Math.tan(a);
  document.write("tan("+a+")="+rezultat);
  //afişează tan(0.9965)=1.545483358938326
</script>
```

Mai multe despre obiectul matematic Math

Există două inconveniente majore ale limbajului JavaScript, pe care le puteţi întâlni în efectuarea calculelor matematice care conţin numere zecimale:

- ✓ timpul de execuţie (JavaScript este relativ lent!);
- ✓ precizia.

Aplicaţie

❑ Scrieţi un program JavaScript care calculează şi afişează cât fac:

- ✓ 0.121*100;
- ✓ 0.119*100.

În figura 2.58 se prezintă documentul (X)HTML complet în care s-a inserat script-ul aplicaţiei [3].

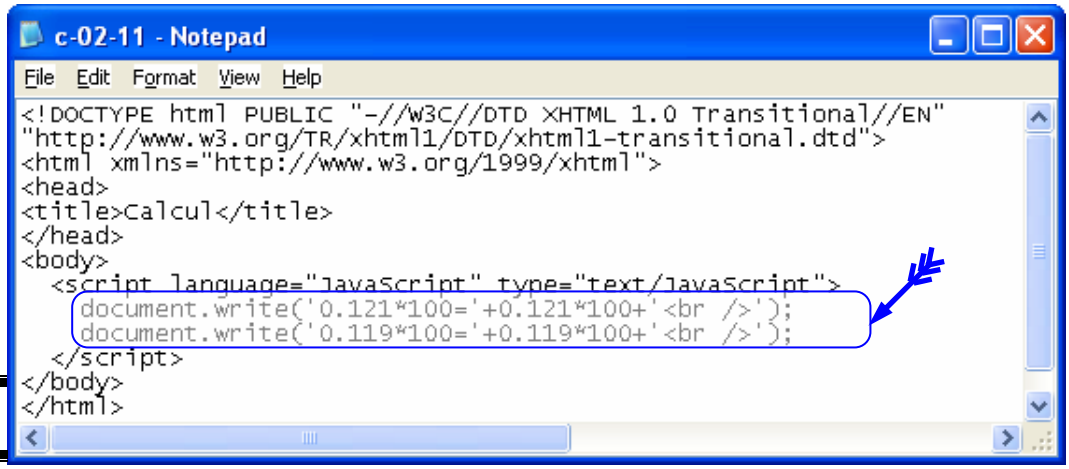


Figura 2.58

Rezultatele execuţiei script-ului sunt afişate în figura 2.59.

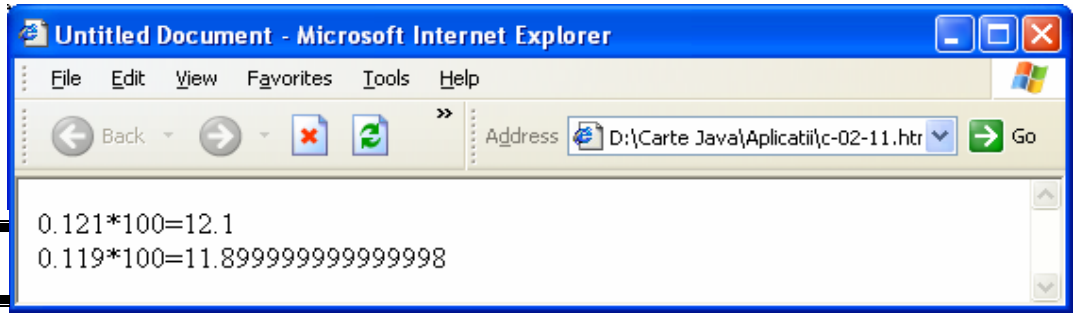


Figura 2.59

Remarci:

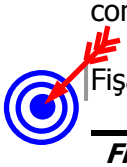
- ✓ Rutinele interne ale limbajului JavaScript nu sunt atât de precise pe cât ne aşteptam. Script-ul dumneavoastră este corect dar rezultatul este diferit faţă de cel scontat.

Aşteptaţi-vă la astfel de surprize (neplăcute!) mai ales atunci când folosiţi numere care conţin multe zecimale!

- ✓ Nici precizia funcţiilor trigonometrice nu este infailibilă!
- ✓ Pentru a rezolva problema afişării rezultatelor cu multe zecimale scrieţi o funcţie personalizată care trunchiază numărul zecimal la un număr de zecimale dorit, înainte de a-l afişa (vezi Conversaţia 8).

Obiectul Number

Obiectul `Number` facilitează gestiunea numerelor. Proprietăţile sale sunt constante care permit definirea valorilor de bază indiferent de navigator.



Fişa obiectului `Number` este prezentată în figura 2.60.

Fişa obiectului	***Number***
Cum creăm obiectul?	`Constructorul Number()`
Proprietăţi	`MAX_VALUE, MIN_VALUE, NaN, NEGATIVE_INFINITY, POSITIVE_INFINITY`
Metode	`toExponential(), toFixed(), toPrecision(), toString()`
Gestionarii de evenimente	–

Figura 2.60

Constructorul Number()

Nu este obligatoriu să creaţi explicit obiecte `Number`. Dacă totuşi ... insistaţi folosiţi constructorul `Number()` care este prezentat în detaliu, în figura 2.61.

Constructor	***Sintaxă***
`Number()`	`Variabila=new Number(`*`Num•r`*`)`

```
Variabila=new Number()
Variabila=Num•r
```

Constructorul `Number()` poate fi folosit în egală măsură ca funcţie. În acest caz, ea returnează valoarea argumentului transmis ca parametru (Tabelul 9).

Tabelul 9

Argument	***Rezultat***
null	0
false	0
true	1
număr	număr
şir de caractere alfanumerice	NaN
şir de caractere numerice	număr

Exemplu:

```
<script>
  x="14.253";y=Number(x);
  document.write(x+"-"+y);
</script>
```

Figura 2.61

Proprietăţile obiectului Number

Proprietăţile obiectului `Number()` sunt prezentate în detaliu în figura 2.62.

Proprietate	***Sintaxă***
`MAX_VALUE`	`Number.MAX_VALUE`

Cea mai mare valoare pozitivă a obiectelor `Number`.

Exemplu:

```
<script>
  document.write(Number.MAX_VALUE);
  //afişează 1.79769931348623157e+308
</script>
```

`MIN_VALUE`	`Number.MIN_VALUE`

Cea mai mică valoare pozitivă a obiectelor `Number`.

Exemplu:

```
<script>
  document.write(Number.MIN_VALUE);
  //afişează 5e.-324
</script>
```

`NaN`	`Number.NaN`

Valoare numerică non validă. Ea este returnată printr-o metodă precum `parseFloat()` sau `parseInt()` atunci când o valoare matematică nu poate fi returnată.

Figura 2.62

`NEGATIVE_INFINITY`	`Number.NEGATIVE_INFINITY`

Valoarea minus infinit. Identic cu `-Infinity`.

`POSITIVE_INFINITY`	`Number.POSITIVE_INFINITY`

Figura 2.62 *(continuare)*

	Valoarea plus infinit. Identic cu `+Infinity`.

Metodele obiectului Number

Metodele obiectului `Number` sunt prezentate în detaliu în figura 2.63.

	Metodă	***Sintaxă***
	`toExponential()`	`Number.toExponential(`*`NumărZecimal`*`)`
	Afişează un număr în format exponenţial cu rotunjire.	
Exemplu:	<script> a=4.445845558555588574; b=a.toExponential(2); document.write(b); //afişează .45e+0 </script>	
	`toFixed()`	`Number.toFixed()`
	Rotunjeşte un număr la întregul cel mai apropiat superior sau inferior.	
Exemplu:	<script> a=8.5; b=a.toFixed(); document.write(b); //afişează 8 </script>	
	`toPrecision()`	`Number.toPrecision(`*`Număr`*`)`
	Rotunjeşte un număr cu precizia indicată ca parametru. Dacă acest parametru este egal cu 1, numărul este rotunjit la întreg.	
Exemplu:	<script> a=4.445845558555588574; b=a.toPrecision(4); document.write(b); //afişează 4.446 </script>	
	`toString()`	`Number.toString()`
	Converteşte un număr într-un şir de caractere. Metoda este foarte puţin utilizată întrucât valorile numerice sunt imediat şi în mod automat convertite în caractere atunci când ele sunt introduse într-un şir de caractere.	
Exemplu:	<script> a=88.596; b=a.toString(); document.write(b); </script>	

Figura 2.63

Remarci:

- ✓ Obiectul `Number` este folosit de foarte puţine ori. Este util atunci când trebuie să accesaţi anumite constante (vezi proprietăţile obiectului `Number`).
- ✓ Crearea obiectelor `Number` este implicită în cele mai multe cazuri.
- ✓ Puteţi crea obiecte `Number` atunci când trebuie să le adăugaţi proprietăţi.

Obiectul Boolean

Obiectul `Boolean` este folosit pentru a transforma o valoare non-booleană într-o valoare booleană.

Obiectul `Boolean` nu poate conţine decât două valori: `TRUE` sau `FALSE`.

Fişa obiectului Boolean este prezentată în figura 2.64.

Fişa obiectului	***Number***
Cum creăm obiectul?	`Constructorul Boolean()`
Proprietăţi:	`prototype`
Metode:	`toString()`
Gestionarii de evenimente:	–

Figura 2.64

Remarcă. prototype permite adăugarea de proprietăţi şi metode obiectului Boolean.

Constructorul Boolean()

Constructorul `Boolean()` este prezentat în detaliu în figura 2.65.

	Constructor	***Sintaxă***
	`Boolean()`	`Variabila=new Boolean()` `Variabila=new Boolean(`*`Valoare`*`)` `Variabila=true` `Variabila=false`
	Creează un nou obiect `Boolean`. El poate fi creat cu operatorul `new` sau implicit, atribuind o valoare logică (`true/false`) unei variabile.	
Exemplu:	<script> //crearea unui obiect boolean vid a=new Boolean(); a=true; </script>	
Exemplu:	<script> //crearea şi iniţializarea simultană a unui obiect boolean a=new Boolean(true); </script>	
Exemplu:	<script>	

Figura 2.65

Figura 2.65 *(continuare)*

```
//crearea implicită a unui obiect boolean
a=true;
</script>
```

Mai multe despre obiectul Boolean

- ✓ Obiectele booleene care conţin una din următoarele valori:
  - `0, false, null, NaN, undefined, ""`

  sunt evaluate `FALSE`.
- ✓ Obiectele booleene care conţin una din următoarele valori:
  - true, "text oarecare", 58, –96

  sunt evaluate `TRUE`.

EXEMPLUL 2 JAVASCRIPT (continuare)

❑ **Codificarea în limbajul JavaScript** *(continuare din Conversația 2)*

Pentru a codifica în limbajul JavaScript algoritmul prezentat în conversația precedentă, creați documentul (X)HTML (utilizați Notepad sau Macromedia Dreamweaver) în care inserați script-ul, cu una din metodele prezentate mai jos:

- ✓ *Metoda 1* – inserați script-ul în corpul paginii;
- ✓ *Metoda 2* – plasați script-ul în antet-ul paginii;
- ✓ *Metoda 3* – utilizați fişiere sursă externe;
- ✓ *Metoda 4* – creați un gestionar de evenimente.

Folosiți unul din editoarele prezentate mai jos:

- ✓ Notepad;
- ✓ Macromedia Dreamweaver MX.

Metoda 1

1. Creați documentul (X)HTML (figura 2.66).

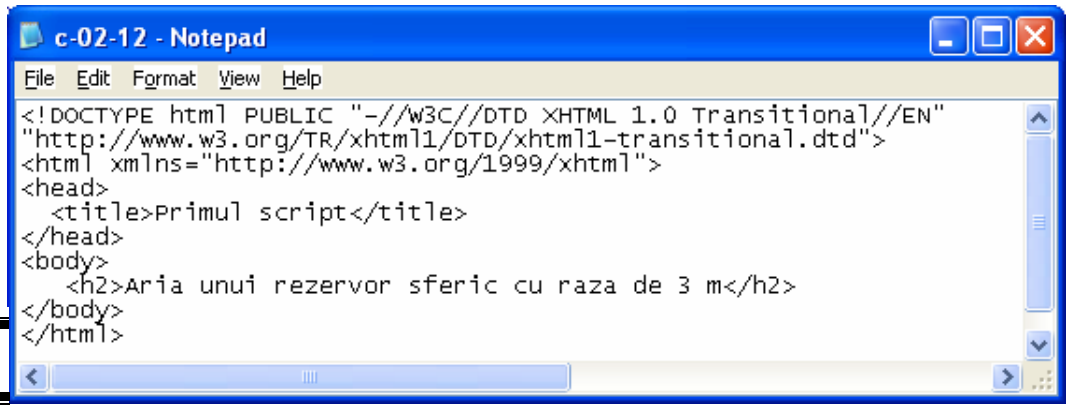


Figura 2.66

2. Inserați elementul `<script> ... </script>` în corpul documentului (figura 2.67).

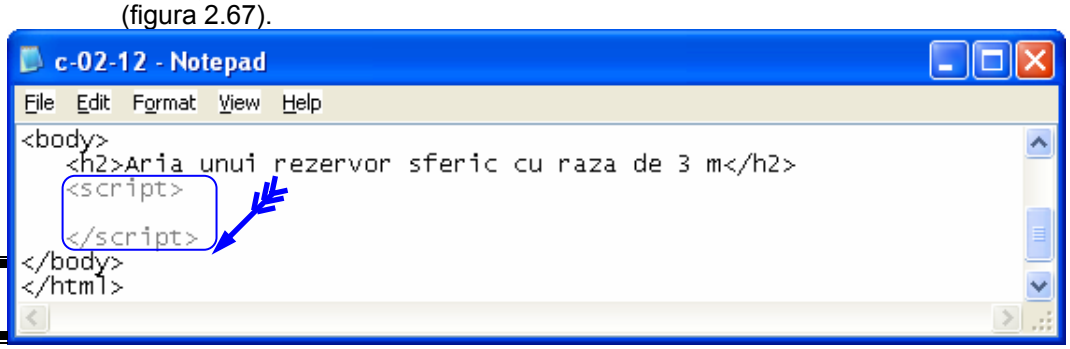


Figura 2.67

3. Atribuiți variabilei r valoarea 3 (figura 2.68).

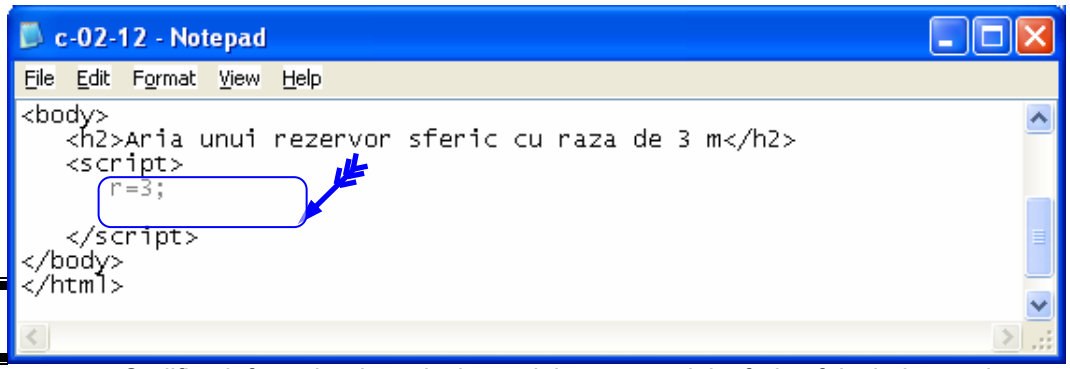


Figura 2.68

4. Codificați formula de calcul a ariei rezervorului sferic, folosind proprietatea `Math.PI` (pentru constanta π) şi metoda `Math.pow()` (pentru codificarea lui r^2, (vezi figura 2.69).

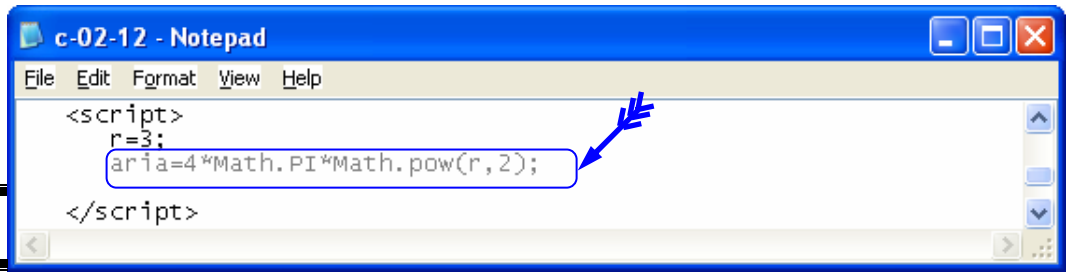


Figura 2.69

Remarci:

- ✓ Proprietatea `Math.PI` a obiectului `Math` este prezentată în detaliu în figura 2.57.
- ✓ Metoda `pow()` a obiectului `Math` este prezentată în detaliu în figura 2.57.

5. Afişați rezultatele (figura 2.70).

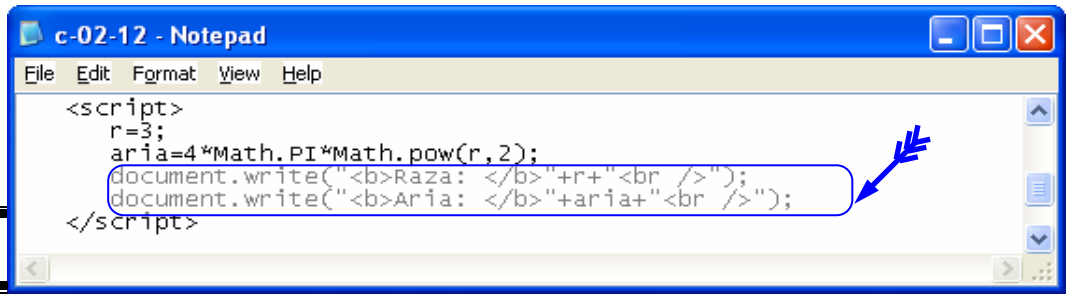


Figura 2.70

Remarci:

- ✓ Posibilitatea de combinare a tag-urilor (X)HTML şi a instrucțiunilor JavaScript este una din caracteristicile cele mai importante ale unui navigator care suportă JavaScript. În realitate, această facilitate constituie esențialul Web-ului dinamic şi interactiv.
- ✓ În figura 2.71 este prezentat documentul (X)HTML complet, editat cu Macromedia Dreamweaver.

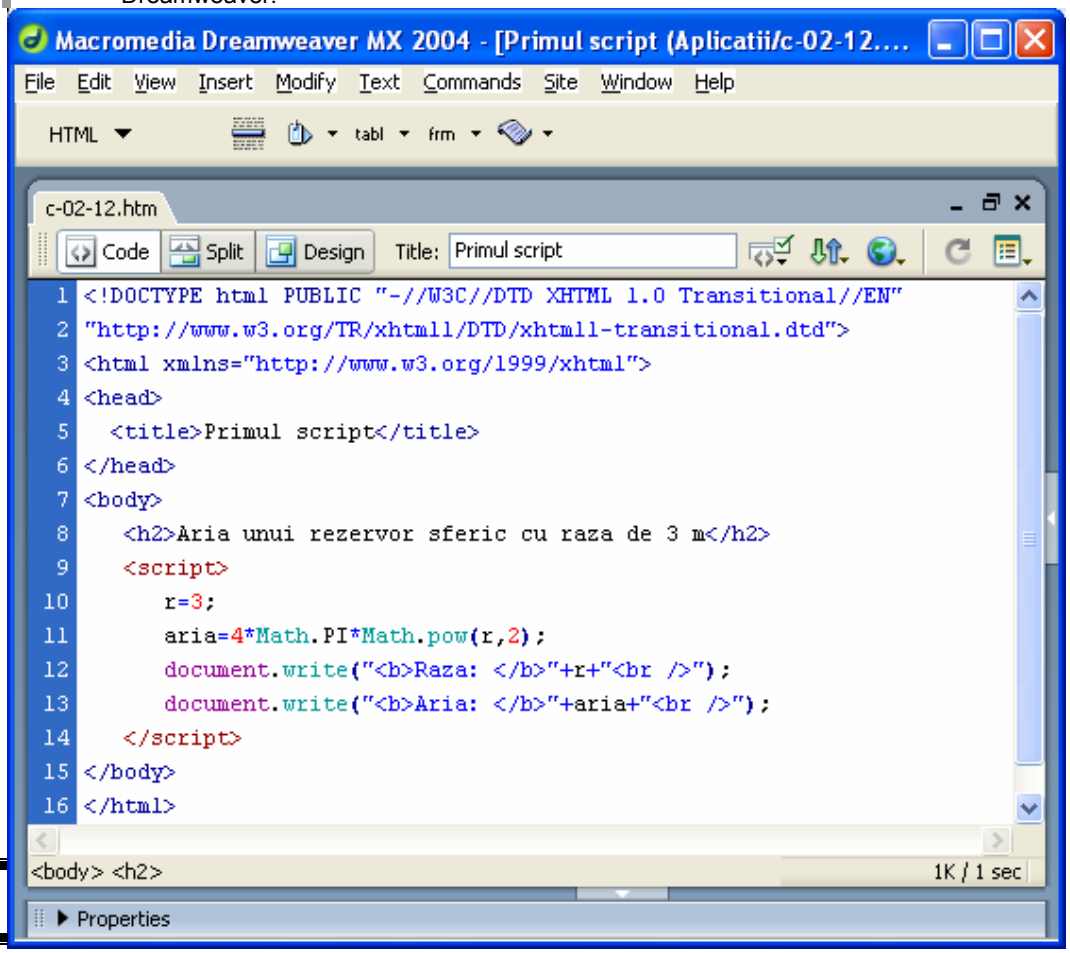


Figura 2.71

6. Afişați pagina Web într-un navigator (figura 2.72).

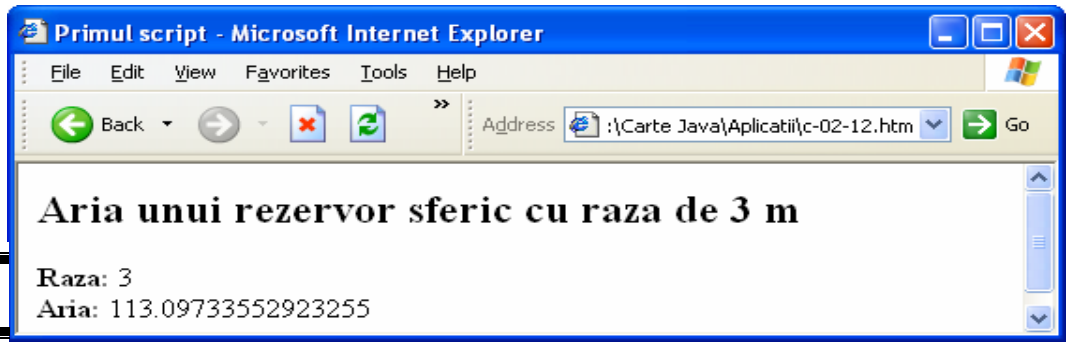


Figura 2.72

Metoda 2

1. Creați documentul (X)HTML.
2. Introduceți elementul `<script> ... </script>` în secțiunea `<head> ... </head>` a documentului (vezi figura 2.73).

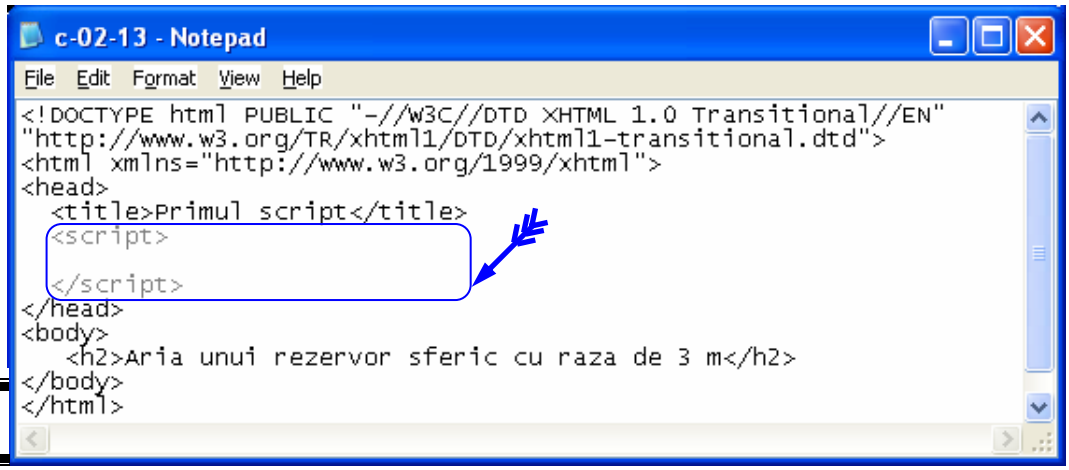


Figura 2.73

3. Definiți funcția `calcul()` în care introduceți formula de calcul precedată de cuvântul cheie `return` (vezi figura 2.74).

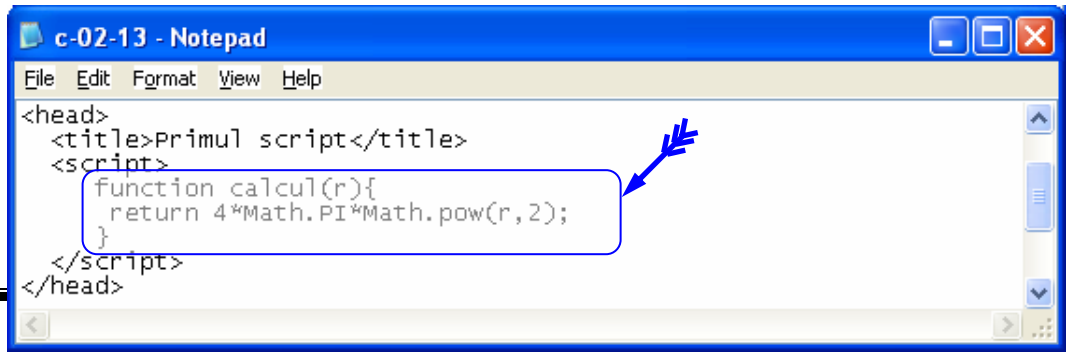


Figura 2.74

Remarci:

- ✓ Revedeți paragraful „Definiți o funcție” (Conversația 2).
- ✓ Revedeți paragraful „Returnarea unei valori” (Conversația 2).
- ✓ De regulă, cea mai bună locație pentru definirea unei funcții este antet-ul (`<head> ... </head>`) documentului. Întrucât instrucțiunile din antet sunt executate primele, aveți certitudinea că funcția este definită înainte de a fi utilizată.

4. Apelați funcția `calcul(3)` din script-ul plasat în corpul documentului (vezi figura 2.75).

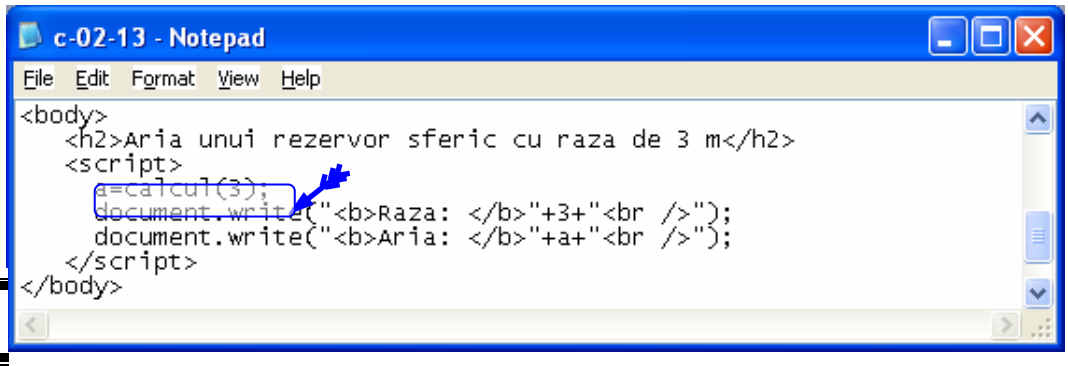


Figura 2.75

Remarci:

- ✓ Apelați funcția `calcul()` pentru mai multe valori ale razei: `calcul(4)`, `calcul(5)`.
- ✓ În figura 2.76 este prezentat documentul XHTML complet editat cu Macromedia Dreamweaver.

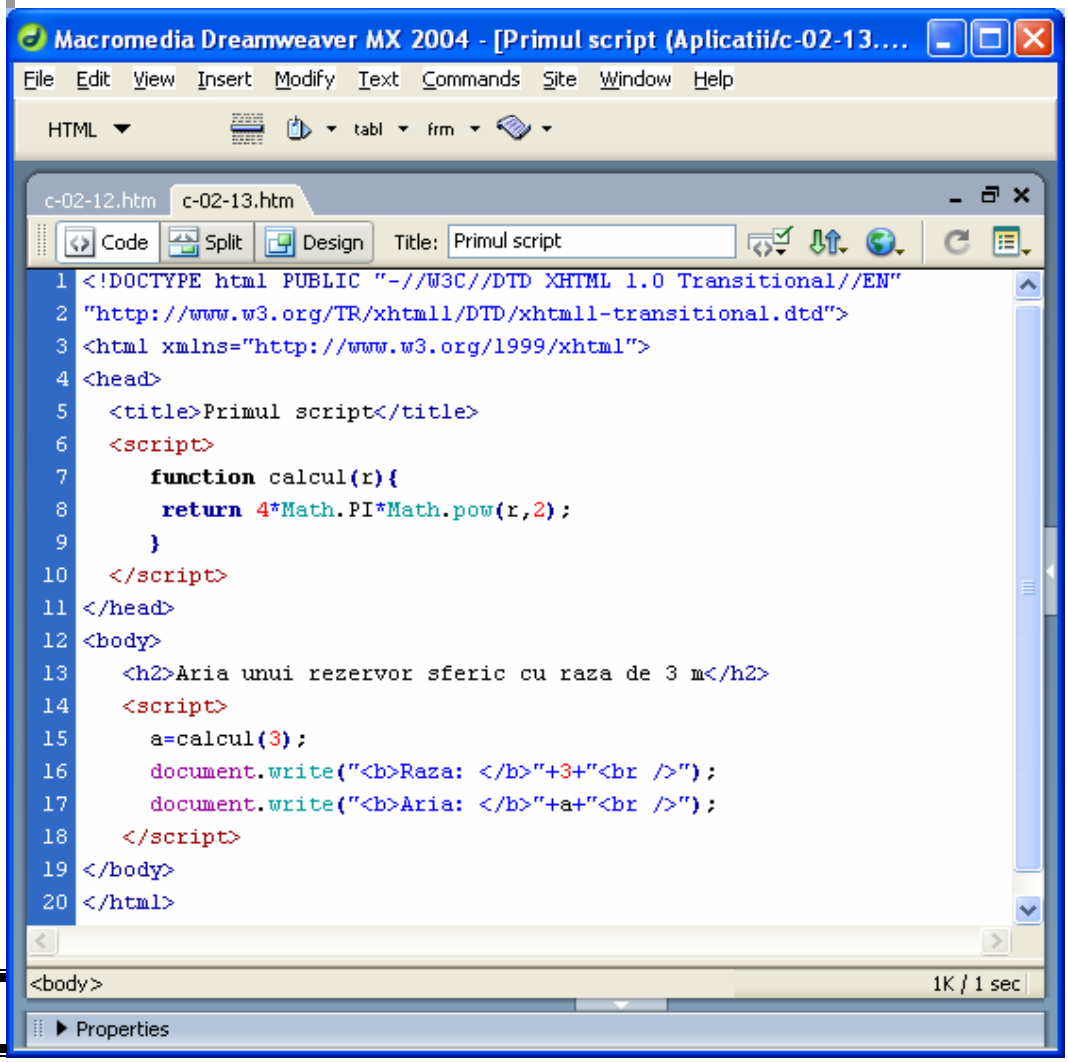


Figura 2.76

5. Afişați pagina Web într-un navigator (vezi figura 2.77).

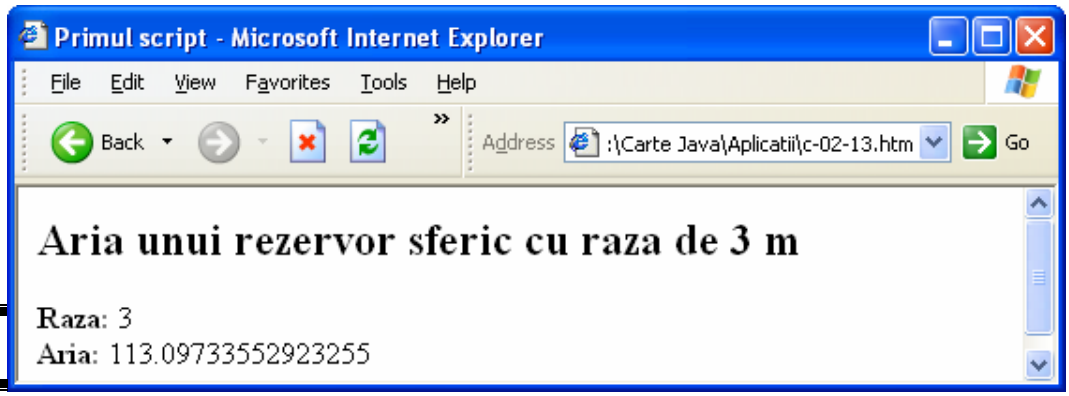


Figura 2.77

Metoda 3

1. Creați fișierul `calcul.js` care conține funcția `calcul()`, figura 2.78.

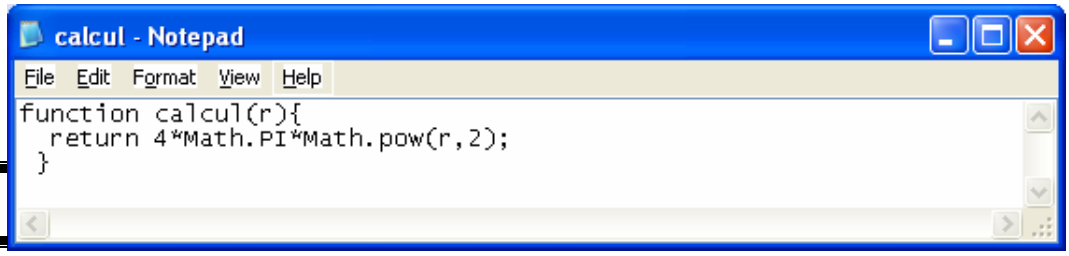


Figura 2.78

2. Introduceți atributul `src="calcul.js"` în tag-ul de deschidere al elementului `<script>` (figura 2.79).

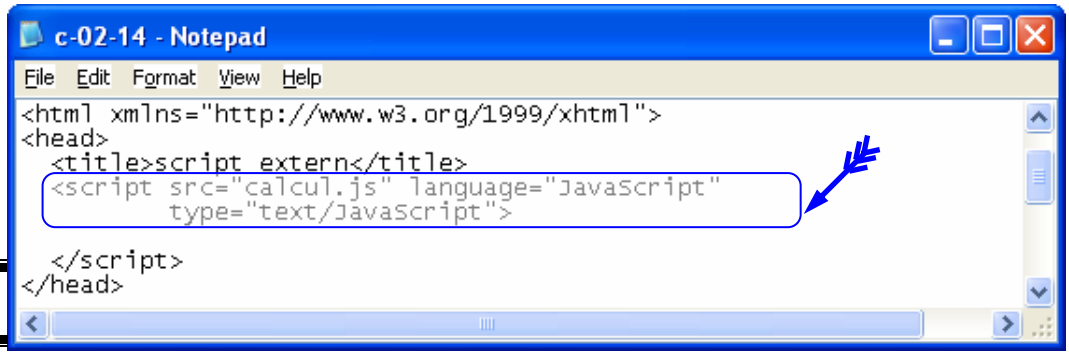


Figura 2.79

Remarci:

- ✓ Atributul `src` transmite interpretorului JavaScript că doriți să includeți codul JavaScript în fișierul `calcul.js`.
- ✓ Extensia `.js` nu este obligatorie dar prezența ei identifică fișierele care conțin codul sursă JavaScript.

Metoda 4

Indicație. Folosiți gestionarul de evenimente `onClick` în tag-ul `<input>` al unui formular (vezi figura 2.80).

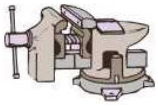
...

```
<form name="form1">
   Raza?<input type="text"name="raza" /><br />
   Aria:<input type="text"name="raza" /><br />
   <input type="button" name="buton1" value="calculeaza aria rezervorului sferic"
          onClick="document.form1.aria.value=document.form1.raza.value=
          document.form1.raza.value=4*Math.PI;" />
</form>
...
```

Figura 2.80

Remarcă. Vezi Conversația 8.

Testați-vă cunoștințele



1. Cum clasificaţi operatorii JavaScript?
2. Care este semnificaţia operatorilor: +=; –=; *=; /=; %=?
3. Care sunt obiectele matematice pe care le cunoaşteţi?
4. Cum se creează obiectul `Math`?
5. Cum creaţi un nou obiect `Number` şi un nou obiect `Boolean`?
6. Precizaţi rezultatele execuţiei următoarelor script-uri (figura 2.81):

Figura 2.81

```
<script>
   x=12; y=4; z=2;
   x*=z++*++y;
   document.write(x);
</script>
```

```
<script>
   y=75;
   x=(y=100)?29:42;
   document.write(x);
</script>
```

7. Care va fi rezultatul expresiei:

   25+”de zile”?

   - un mesaj de eroare;
   - 26;
   - ”25 de zile”.

Vizitați site-urile

- ✓ http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsorijscript.asp
- ✓ http://developer.netscape.com/tech/javascript
- ✓ http://www.zdnet.com/devhead/filters/0,,2133214,00.html
- ✓ http://webdeveloper.earthweb.com/webjs
- ✓ http://javascript.about.com/?once=true&

Conversația 3

Instrucțiunile limbajului JavaScript

În această conversație:

Instrucțiunile limbajului JavaScript

JavaScript este un limbaj de nivel înalt, bazat pe obiecte, care permite programatorilor să creeze cu uşurinţă documente (pagini) Web interactive.

Instrucţiunile limbajului JavaScript sunt uşor de înţeles şi aplicat. Ele pilotează script-ul, permiţându-i să ia o decizie, să creeze bucle (iteraţii) etc.

Cea mai mare parte a instrucţiunilor JavaScript sunt recunoscute de toate navigatoarele, dar ... mai există şi divergenţe!

Pentru a programa în limbajul JavaScript folosiţi următoarele tipuri de instrucţiuni (vezi figura 3.1).

Tipuri de instrucţiuni	***Instrucţiuni JavaScript***
iteraţii (cicluri)	`for, for...in, while, do...while`
decizii	`if, if...else, switch`
documentarea script-urilor	`/*...*/, //`
funcţii	`function(), return`
generarea unei erori, testarea unei porţiuni de cod şi interceptarea eventualelor erori	`throw şi try...catch`
declararea unei variabile	`var`
simplificarea efortului de programare (reducerea codului)	`with`
instrucţiunile `break, continue`	

Figura 3.1

Iteraţii

În limbajul JavaScript, iteraţiile (repetiţiile) se codifică cu una din următoarele instrucţiuni:

- ✓ `for` (pentru) – structură de iteraţie cu număr cunoscut de paşi;
- ✓ `for...in` – structură de tip `for` care permite baleierea tuturor elementelor unei matrici sau ale unui obiect;
- ✓ `while` (cât timp) – structură de iteraţie cu număr necunoscut de paşi, condiţionată anterior;
- ✓ `do...while` – structură de tip `while`, condiţionată posterior.

for (pentru)

Utilizaţi instrucţiunea `for` pentru a crea cicluri (bucle) în programele dumneavoastră JavaScript. Instrucţiunea `for` este folosită pentru a repeta o instrucţiune sau o secvenţă de instrucţiuni cât timp rezultatul evaluării unei condiţii este `TRUE`.

Instrucţiunea `for` este prezentată în detaliu în figura 3.2.

Instrucţiune	***Sintaxă***
`for`	`for(`*`ExpresieIniţială;CondiţieContinuare;`* *`ExpresieFinală`*`){` `cod JavaScript` `}` unde, *ExpresieIniţială* este expresia de iniţializare a variabilei de control a instrucţiunii; *CondiţieContinuare* este cel de-al doilea parametru al instrucţiunii, care reprezintă o expresie logică ce se evaluează la valoarea `true` sau `false`; *ExpresieFinală* actualizează valoarea variabilei de control a instrucţiunii.
	Bucla este executată cât timp *CondiţieContinuare* returnează `TRUE`.

Exemplu:
```
<script>
for(i=1;i<13;i++){
  document.write(i+" <br />");
}
</script>
```

Exemplu:
```
<script>
for(i=1;i<13;i+=2){
  document.write(i+" <br />");
}
</script>
```

Exemplu:
```
<script>          //două bucle imbricate (nested)
for(x=1;x<11;x++){
  for(y=1;y<11;y++){
    document.write(x*y+" ");
  }
  document.write("<br />");
}
</script>
```

Exemplu:
```
<script>
for(var i=true;i==true;){
  i=confirm(" Doriti sa continuati?");
}
</script>
```

Exemplu:
```
<script>     //ciclu infinit;
for(;;){
  document.write("Iar innebunesc salcamii!");
}
</script>
```

Exemplu:
```
<script>
for(i=0;i<13;i++){
  ;//instructiunile ciclului
}
</script>
```

Figura 3.2

Mai multe despre for

- ✓ `for` conţine trei parametrii separaţi prin punct şi virgulă.

- ✓ Cei trei parametrii ai instrucţiunii `for` sunt opţionali; dacă omiteţi unul dintre ei, separatorul punct şi virgulă (;) este obligatoriu.
- ✓ Acoladele nu sunt indispensabile în iteraţiile care conţin o singură instrucţiune, dar se recomandă să le folosiţi fără a ţine cont de numărul de instrucţiuni din corpul iteraţiei. Procedând în acest mod veţi putea adăuga cu uşurinţă noi instrucţiuni fără a provoca erori de sintaxă.
- ✓ Veţi constata că utilizăm foarte des numele de variabilă i, ca identificator al variabilei de control. Este o tradiţie (vezi limbajul Forth) în programare.
- ✓ Instrucţiunile `for` pot fi imbricate (suprapuse) dar nu trebuie să se intersecteze.
- ✓ Principiul de funcţionare al instrucţiunii `for` este următorul:
  1. Se iniţializează variabila de control a instrucţiunii (*ExpresieIniţială*).
  2. Se testează *CondiţieContinuare*.
  3. Dacă rezultatul evaluării *CondiţieContinuare* este `true` se execută instrucţiunile din corpul buclei; în caz contrar script-ul execută instrucţiunea imediat următoare după acolada de sfârşit (}).
  4. Se execută *ExpresieFinală* şi se testează *CondiţieContinuare*.
- ✓ Pentru a provoca o ieşire imediată din bucla `for` folosiţi instrucţiunea `break`.
- ✓ Pentru a relua ciclul fără a mai fi executate instrucţiunile care urmează folosiţi instrucţiunea `Continue`.

EXEMPLUL 3.1 JAVASCRIPT

❑ Formularea problemei

Vom relua problema din conversația precedentă (vezi EXEMPLUL 2 JAVASCRIPT) considerând de această dată mai multe rezervoare cilindrice echilaterale (*generatoarea este egală cu diametrul*) toate pline cu benzină. Raza acestora variază de la 2 la 10 m (vezi figura 3.3) cu pasul 1 (2, 3, ..., 10 m).

Dorim să calculăm şi să afişăm într-o pagină Web cantitatea de lichid (benzină) din cele nouă rezervoare cilindrice echilaterale, iar în final să afişăm cantitatea totală de benzină din toate rezervoarele.

Densitatea benzinei (0.7 g/cm^3) se va introduce în mod dinamic.

Cum s-ar putea rezolva această problemă?

Cei mai grăbiți dintre dumneavoastră se şi gândesc deja la instrucțiunile JavaScript necesare scrierii script-ului.

Banal, extrem de banal vor exclama poate mulți dintre dumneavoastră!

Formulele de calcul pot constitui un impediment, dar vi le vom aminti noi:

$$\rho = \frac{m}{V} \quad (1)$$

unde, ρ este densitatea, m reprezintă masa, iar V este volumul.

$$V = \pi R^2 * G \quad (2)$$

unde, G este generatoarea cilindrului.

$$V = \pi R^2 * 2R = 2\pi R^3 \quad (3)$$

unde, V este volumul cilindrului echilateral.

Cam multe formule, dintr-o dată!

Este adevărat, dar cu siguranță că ele nu vă vor strica viața personală! Promitem!

Remarcă. A gândi în limbaj informatic (JavaScript) încă de la formularea problemei ni se pare o concepție total greşită! Vă propunem metodologia de analiză, proiectare şi realizare a aplicațiilor pe care v-am prezentat-o în Conversația precedentă.

❑ Analiza problemei

Formatul datelor de ieşire (fereastra în care se afişează pagina Web), *tabela de variabile şi specificațiile de programare* sunt ilustrate în figurile: 3.3, 3.4, 3.5.

Formatul datelor de ieşire

RAZA	MASA
2	XXX.XX
3	XXX.XX
.	.
.	.
.	.
10	XXX.XX
MASA TOTALA	XXX.XX TONE

Figura 3.3

Tabela de variabile

Variabile de intrare	*Variabile de stare*	*Variabile de iesire*

Figura 3.4

d: densitatea benzinei	v: volumul cilindrului echilateral r: raza rezervorului	s: masa totală de benzină m: masa de benzină

Specificații de programare

Descriere. Script-ul calculează şi afişează cantitatea de benzină din mai multe rezervoare cilindrice echilaterale a căror rază variază de la 2 la 10 m, cu pasul de 1 m.

Intrări. Densitatea benzinei (0.7 g/cm^3).

Ieşiri. Masa de benzină din fiecare rezervor şi masa totală de benzină.

Lista de funcțiuni ale script-ului

1. Citeşte densitate benzină (d)
2. Inițializează variabila s cu zero
3. Tipăreşte cap tabel
4. Calculează volumul rezervorului (V)
5. Calculează masa (m) de benzină din fiecare rezervor
6. Însumează m în s
7. Afişează raza (r) şi masa rezervorului (m)
8. Inițializează variabila de control a buclei (r)
9. Incrementează şi testează variabila de control a buclei (r)
10. Afişează masa totală de benzină (s)
11. Stop

Figura 3.5

❑ Proiectarea script-ului

Vom folosi pentru proiectarea script-ului cele două variante de pseudocod (vezi Conversația 2):

- ✓ *Varianta 1* – limbaj natural structurat;
- ✓ *Varianta 2* – scriere formalizată (apropiată de limbajul JavaScript).

Varianta 1

În figura 3.6 se prezintă pseudocodul în limbajul natural structurat (*Varianta 1*).

Pseudocodul (*Varianta 1*)

1. Citeşte densitatea benzinei, afişează capul de tabel, calculează, însumează şi afişează cantitatea de benzină din rezervoarele cilindrice echilaterale a căror rază variază de la 2 la 10 m cu pasul de 1 m.
   - **1.1.** Citeşte densitatea benzinei.
   - **1.2.** Inițializează cu zero masa totală de benzină.
   - **1.3.** Tipăreşte un rând de 30 de liniuțe.
   - **1.4.** Tipăreşte capul de tabel (RAZA, MASA).

Figura 3.6

1.5. Tipăreşte un rând de 30 de liniuțe.

1.6. `Pentru` fiecare rezervor cilindric echilateral a cărui rază variază de la 2 la 10 m, cu pasul de 1 m, `repet•` acțiunile:

1.6.1. Calculează volumul.

1.6.2. Calculează masa de benzină.

1.6.3. Însumează masa de benzină în cantitatea totală de benzină.

1.6.4. Afişează raza şi masa de benzină.

2. Tipăreşte un rând de 30 de liniuțe.

3. Afişează masa totală de benzină.

4. Stop.

Figura 3.6 *(continuare)*

Remarci:

- ✓ Evident, acțiunea **1** nu reprezintă o acțiune primitivă, impunându-se în acest sens o rafinare (descompunere) a acestora (acțiunile **1.1**, ..., **1.6**).
- ✓ Pentru rafinarea acțiunii neprimitive **1**, vom folosi metoda analizei descendente (top-down).
- ✓ Prezentarea algoritmului în limbaj natural structurat are, după cum ați putut constata şi singuri următoarele dezavantaje: exprimări lungi, greoaie etc.

Varianta 2

În figura 3.7 se prezintă pseudocodul în scriere formalizată (*Varianta 2*).

Pseudocodul (*Varianta 2*)

```
REZERVOARE   BEGIN
INIT                 BEGIN
                           Citeşte densitatea benzinei (d)
                           s=0
INIT                 END
PRELUCR•RI           BEGIN
                           DO LINIUTZA
                           WRITE('RAZA'+'---'+'MASA');
                           DO LINIUTZA
PENTRU                     FOR(r=2;r<=10;r++)
                                 v=2∏r^3
                                 m=d*v
                                 s=s+m
                                 WRITE(r+'...'+m)
PENTRU                     ENDFOR
PRELUCR•RI           END
                     DO LINIUTZA
                     WRITE('MASA TOTALA'+' '+s)
REZERVOARE   END
LINIUTZA     BEGIN
                     Afişează un rând de 30 liniuțe
LINIUTZ•     END
```

Figura 3.7

Remarci:

- ✓ În general, script-urile conțin secvențe alcătuite din grupuri de instrucțiuni care se execută numai în anumite condiții, cât şi grupuri de instrucțiuni care se execută de atâtea ori cât timp sau până când este îndeplinită o condiție.
- ✓ Formatul general al blocului de secvență este ilustrat în figura 3.8.

Figura 3.8

```
Eticheta      BEGIN
                 Grup de instrucțiuni
Eticheta      END
```

unde,

`BEGIN` şi `END` joacă rolul de delimitatori; ei reprezintă punctul unic de intrare, respectiv punctul unic de ieşire din bloc şi poartă aceeaşi etichetă (cu majuscule sau minuscule).

✓ Formatul general al structurii de iterație FOR este ilustrat în figura 3.9.

Figura 3.9

```
Eticheta      FOR(v=vi;v<=vf;v++)
                 Corpul buclei
Eticheta      ENDFOR
```

unde,

`v` reprezintă variabila de control a buclei; v_i reprezintă valoarea inițială a variabilei `v`; v_f reprezintă valoarea finală a variabilei `v`, iar `v++` serveşte, în general la incrementarea variabilei `v`. `FOR` şi `ENDFOR` sunt delimitatorii structurii.

❑ Codificarea în limbajul JavaScript

În figura 3.10 este prezentat documentul (X)HTML complet.

Figura 3.10

c-03-01 - Notepad

File Edit Format View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Rezervoare</title>
<script language="JavaScript" type="text/JavaScript">
  d=parseFloat(prompt('Densitatea benzinei=?',0));
  s=0;
  function liniutza(){
    document.write("<hr width='50%' align='left'>");
  }
</script>
</head>
<body>
  <script language="JavaScript" type="text/JavaScript">
    liniutza();
        document.write('<pre>RAZA                    MASA</pre>');
        liniutza();
        for(r=2;r<=10;r++){
          v=2*Math.PI*(Math.pow(r,3));
          m=d*v;
          s=s+m;
          document.write(r+'  '+m+'<br />');
        }
        liniutza();
        document.write('MASA TOTALA='+s);
  </script>
</body>
</html>
```

În figura 3.11 şi figura 3.12 se prezintă rezultatele execuției script-ului.

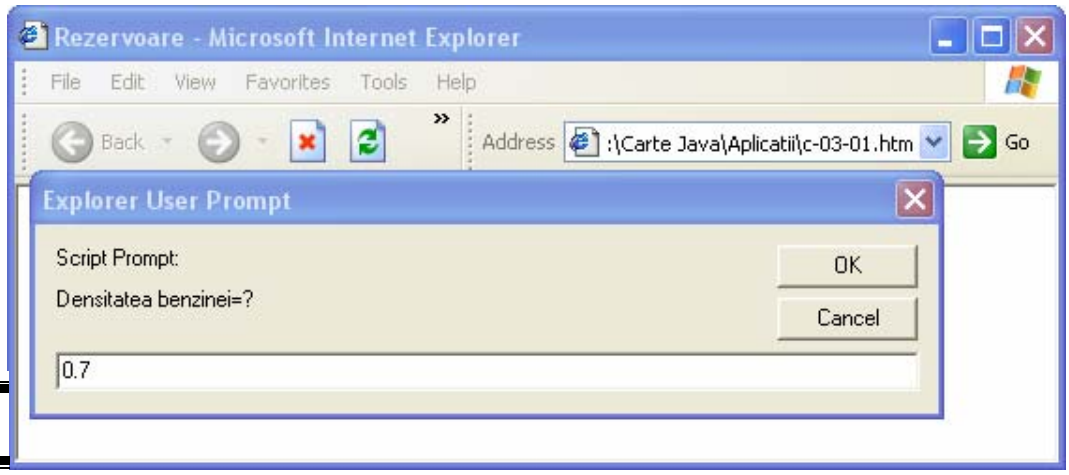


Figura 3.11

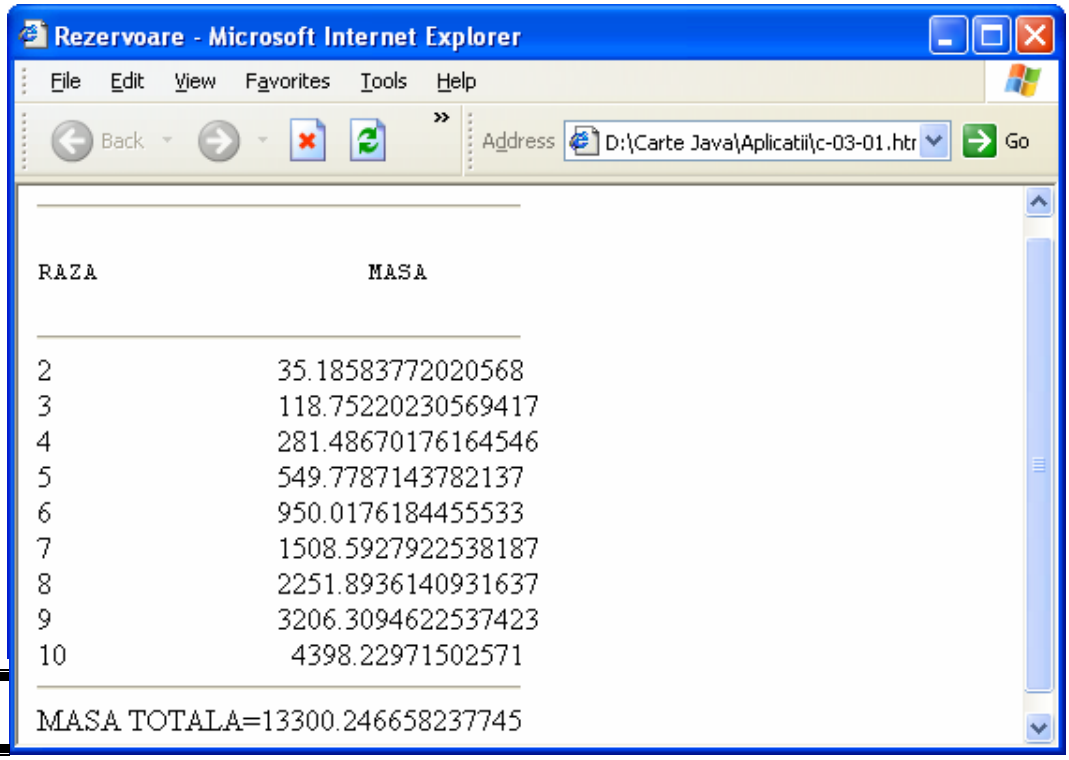


Figura 3.12

Comentarii

- ✓ Funcția `liniutza()` este apelată din trei puncte ale corpului principal (<body>...</body>) al documentului.
- ✓ Funcția `liniutza()` mai poate fi scrisă şi sub forma (vezi figura 3.13).

```
function liniutza(){
  document.write('<br />');
  for(i=1;i<=30;i++){
    document.write('-');
  }
}
```

Figura 3.13

- ✓ Pentru calculul volumului rezervorului cilindric echilateral am utilizat proprietatea `Math.PI` şi metoda `Math.pow(r,3)` care aparțin obiectului `Math` (vezi obiectul matematic `Math`).

Aplicaţii

❑ Scrieţi un script care calculează şi afişează suma primelor 100 numere întregi.

Indicaţie. În figura 3.14 este prezentat script-ul aplicaţiei. Inseraţi acest script într-un document (X)HTML.

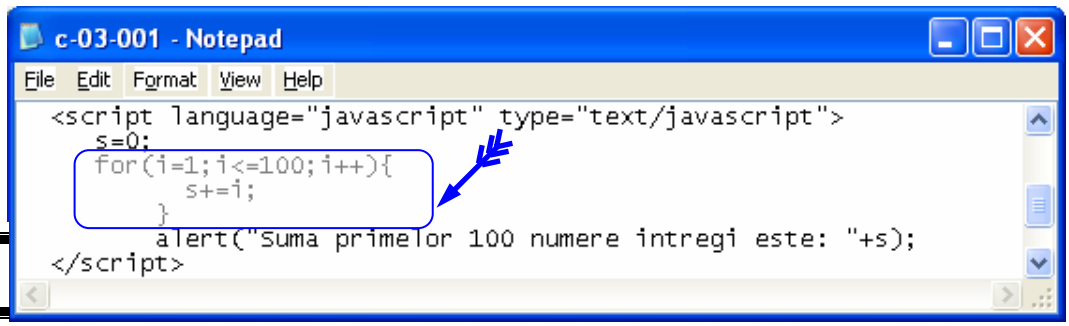


Figura 3.14

❑ Scrieţi un script care generează 25 de seturi de coordonate, generate de două bucle `for` imbricate:

(0,0), (0,1), (0,2), (0,3), (0,4), (0,5)

(1,0), (1,1), (1,2), (1,3), (1,4), (1,5)

(2,0), (2,1), (2,2), (2,3), (2,4), (2,5)

(3,0), (3,1), (3,2), (3,3), (3,4), (3,5)

(4,0), (4,1), (4,2), (4,3), (4,4), (4,5)

(5,0), (5,1), (5,2), (5,3), (5,4), (5,5)

Indicaţie. În figura 3.15 este prezentat script-ul aplicaţiei. Inseraţi acest script într-un document (X)HTML.

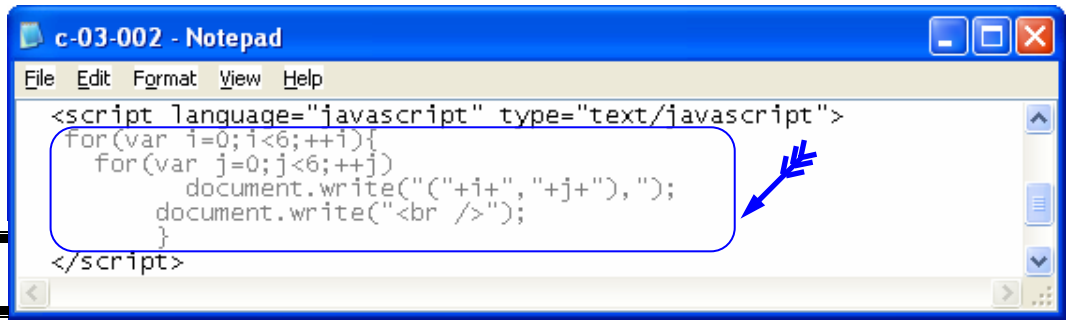


Figura 3.15

❑ Scrieţi două bucle imbricate care generează şi afişează tabla înmulţirii.

Indicaţie. În figura 3.16 este prezentat script-ul aplicaţiei. Inseraţi acest script într-un document (X)HTML.

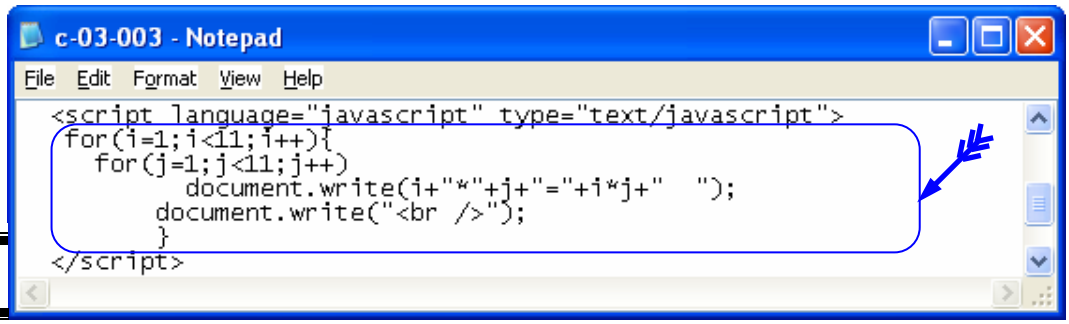


Figura 3.16

❑ Scrieţi un script care calculează (cu o structură iterativă) şi afişează: 1!+2!+ … +7!

Indicaţie. În figura 3.17 este prezentat script-ul aplicaţiei. Inseraţi acest script într-un document (X)HTML.

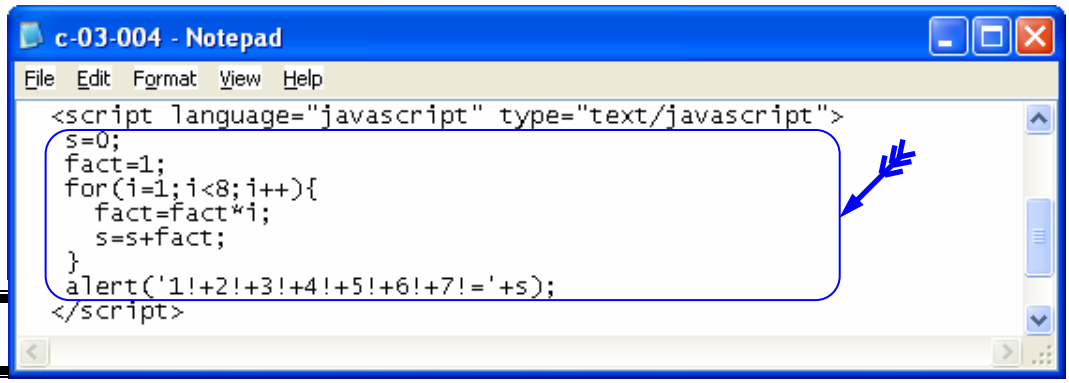


Figura 3.17

❑ Scrieţi un script care editează următorul triunghi al numerelor (figura 3.18).

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```

Figura 3.18

while (cât timp)

Una din instrucţiunile de iteraţie (repetiţie) cele mai simple este `while`, cunoscută şi sub denumirea de structură de iteraţie cu număr necunoscut de paşi, condiţionată anterior.

Deosebirea dintre `while` şi `for` constă în faptul că instrucţiunea `while` nu include în declaraţia sa funcţia de iniţializare a variabilei de control (vezi

ExpresieIniţială) şi funcţia de actualizare a variabilei de control (vezi *ExpresieFinală*).

Instrucţiunea `while` este prezentată în detaliu în figura 3.19.

	Instrucţiune	***Sintaxă***
	`while`	`while (condiţie){` `  cod JavaScript` `}`
	Instrucţiunile (din corpul buclei (cod JavaScript) sunt executate cât timp rezultatul evaluării *condiţie* este `TRUE`.	
Exemplu:	<script> i=1; while(i<=7){ document.writeln(i); ++i; } document.writeln(”Ati afisat 7 numere.”); </script>	
Exemplu:	<script> x=1; while(x<10){ document.write(x+” ”); x++; } </script>	
Exemplu:	<script> var k=20; while(k<11){ document.write(k+” ”); k++; } </script>	
Exemplu:	<script> i=10; while(i>0){ document.write(i); --i; } document.write(”La revedere!”); </script>	
Exemplu:	<script> i=1; while(i<=100){ document.writeln(i); i*=2; } </script>	

Figura 3.19

Mai multe despre while

- ✓ `while` este foarte asemănător cu `if`. `if` execută o singură dată instrucţiunile din corpul buclei (cod JavaScript), în timp ce `while` le execută într-o buclă cât timp rezultatul evaluării *condiţie* este `TRUE`.
- ✓ În realitate, bucla `for` nu este decât un caz particular al buclei `while`, care "integrează" direct funcţiile de iniţializare şi de incrementare a variabilei de control a buclei.
- ✓ Buclele `while` au o arie de utilizare mai largă decât buclele `for`, dar decizia în a le alege vă aparţine numai dumneavoastră!
- ✓ Numărul minim de execuţii a `cod JavaScript` este zero.
- ✓ Instrucţiunea `break` provoacă o ieşire imediată din buclă.
- ✓ Instrucţiunea `continue` provoacă reluarea buclei fără a mai fi executate instrucţiunile care urmează.

EXEMPLUL 3.2 JAVASCRIPT

Problema din EXEMPLUL 3.2 JAVASCRIPT este aceeaşi cu problema din EXEMPLUL 3.1 JAVASCRIPT. Singurele deosebiri apar în următoarele două faze:

- ✓ proiectarea script-ului – în pseudocod s-a înlocuit `for` cu `while` (vezi figura 3.20).

Pseudocodul

```
REZERVOARE   BEGIN
             ...
PRELUCR•RI           BEGIN
                              ...
                              r=2
CATTIMP                       WHILE(r<=10)
                                     v=2∏r^3
                                     m=d*v
                                     s=s+m
                                     WRITE(r+'...'+m)
                                     r++
CATTIMP                       ENDWHILE
PRELUCR•RI           END
REZERVOARE   END
```

Figura 3.20

- ✓ codificarea în limbajul JavaScript – în script-ul din corpul principal al documentului (X)HTML s-a înlocuit `for` cu `while` (figura 3.21).

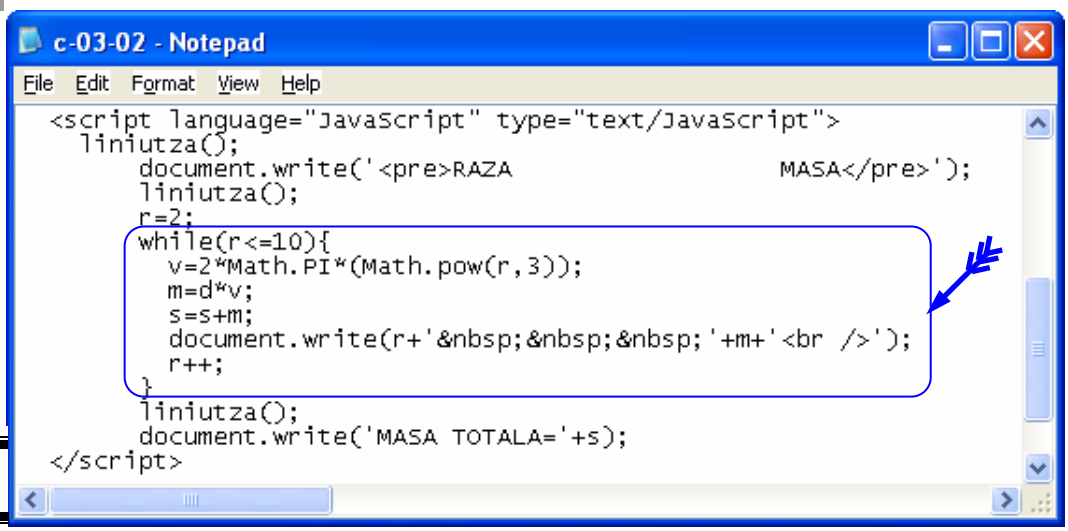


Figura 3.21

Rezultatele obținute sunt evident, identice.

do while

Dacă `while` execută una sau mai multe instrucţiuni (corpul buclei) de cel puţin zero (0) ori, `do…while` execută cel puţin o dată instrucţiunea (instrucţiunile) din corpul buclei; apoi repetă execuţia cât timp rezultatul evaluării condiţiei (vezi *condiţie* din `do…while`) este `TRUE`.

Instrucţiunea `do…while` este prezentată în detaliu în figura 3.22.

	Instrucţiune	***Sintaxă***
	`do…while`	`do{` `  cod JavaScript` `}` `while (condiţie)`
	`do...while` execută cod JavaScript cel puţin o dată, apoi este reexecutat cât timp rezultatul evaluării *condiţie* este `TRUE`.	
Exemplu:	<script> i=0; do{ document.writeln(i+" "); ++i; } while(i<11); </script>	

Figura 3.22

Exemplu:

```
<script>
  var i=20;
  do{
    document.write(i+" <br />");
    i++;
  }
  while(i<11);
</script>
```

Exemplu:

```
<script>
  s=20; i=1;
  do{
    s+=i;
    i++;
  }
  while(i<=100);
  alert('Suma primelor 100 de numere naturale este'+s);
</script>
```

Figura 3.22 *(continuare)*

Mai multe despre do...while

- ✓ `do...while` funcţionează exact ca instrucţiunea `while`, cu deosebirea că nu evaluează *condiţie* decât după prima iteraţie. În felul acesta se garantează că script-ul dintre acolade va fi executat cel puţin o dată.
- ✓ Instrucţiunea `break` provoacă o ieşire forţată din buclă.
- ✓ Instrucţiunea `continue` provoacă reluarea buclei fără a mai fi executate instrucţiunile care urmează.

EXEMPLUL 3.3 JAVASCRIPT

Problema din EXEMPLUL 3.3 JAVASCRIPT este aceeaşi cu problema din EXEMPLUL 3.1 JAVASCRIPT.

Singurele deosebiri apar în fazele de:

- ✓ Proiectarea script-ului – în pseudocod s-a înlocuit `for` cu `do...while` (vezi figura 3.23).

Pseudocodul

```
PRELUCRARI          BEGIN
                        ...
                        r=2
PRELDOWHILE             DO
                            v=2∏r^3
                            m=d*v
                            s=s+m
                            WRITE(r+'...'+m)
                            r++
PRELDOWHILE             WHILE(r<=10)
PRELUCRARI          END
```

Figura 3.23

- ✓ Codificarea în limbajul JavaScript – în script-ul din corpul principal al documentului (X)HTML s-a înlocuit `for` cu `do...while` (figura 3.24).

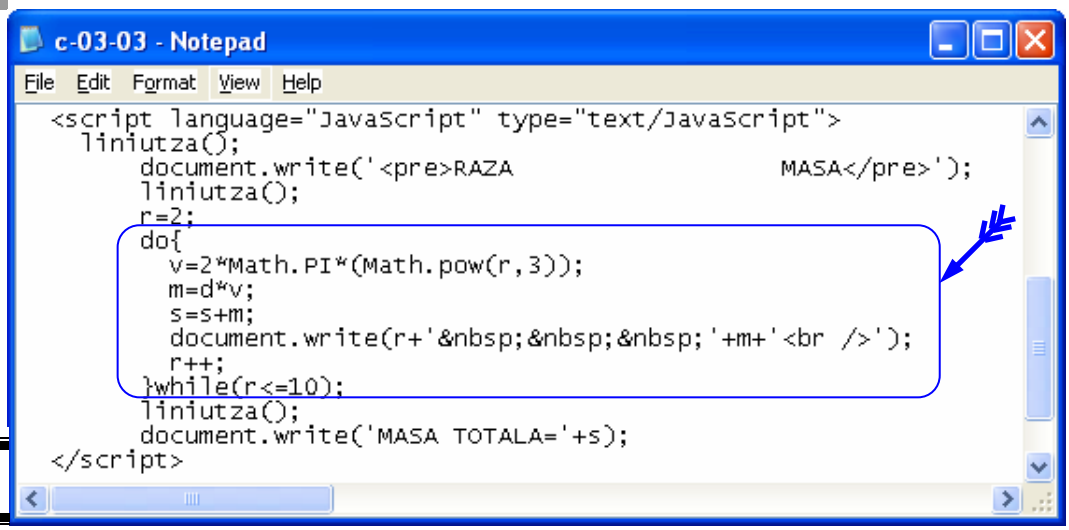


Figura 3.24

for ... in

Instrucţiunea `for…in` este o structură de iteraţie care execută aceeaşi instrucţiune sau aceeaşi secvenţă de instrucţiuni pentru toate proprietăţile unui obiect. Această instrucţiune poate fi utilă, de exemplu, atunci când doriţi să afişaţi numele tuturor proprietăţilor unui obiect.

Instrucţiunea `for…in` este prezentată în detaliu în figura 3.25.

Instrucţiune	*Sintaxă*
`for…in`	`for (Proprietate in Obiect) { cod JavaScript }`

unde,
Proprietate este un element literal generat de JavaScript.

Bucla listează proprietăţile unui obiect. Variabila de control a buclei (*Proprietate*) reprezintă o proprietate a obiectului.

Exemplu:

```
<script>
//afişează toate proprietăţile obiectului window şi valorile acestuia
 for (x in window){
   document.writeln(x+":"+window[x]+"<br />");
 }
</script>
```

Exemplu:

```
<script>
//afişează toate proprietăţile unui obiect creat
  obiect=new object;
  obiect.nume="maxim";
  obiect.zi="luni";
  obiect.culoare="cepia";
 for(x in obiect){
     document.write(x+":"+ obiect[x]+"<br />");
 }
</script>
```

Figura 3.25

Mai multe despre for...in

- ✓ Pentru a folosi cu uşurinţă instrucţiunea `for...in` este bine să aveţi cunoştinţe elementare despre obiectele JavaScript.
- ✓ Puteţi folosi `for...in` cu orice obiect JavaScript, indiferent dacă are sau nu proprietăţi.
- ✓ `for...in` funcţionează şi cu alte obiecte particularizate, ca de exemplu o variabilă.
- ✓ `for...in` se termină în mod automat, atunci când a fost identificată ultima proprietate a obiectului.

În limbajul JavaScript, deciziile (selecţiile) se codifică cu una din următoarele instrucţiuni:

- ✓ `if` (dacă) – structură de decizie cu o singură alternativă.
- ✓ `if … else` (dacă … în caz contrar) – structură de decizie cu două alternative.
- ✓ `switch` – structură de decizie cu mai multe alternative.

if (dacă)

Una din principalele facilităţi ale unui limbaj de programare este aceea de a putea testa şi compara valorile.

Putem crea astfel script-uri care vor reacţiona în mod diferit în funcţie de valorile variabilelor sau de informaţiile furnizate de către utilizator.

Principala instrucţiune de decizie (selecţie) a limbajului JavaScript este `if`.

Instrucţiunea `if` este prezentată în detaliu în figura 3.26.

	Instrucţiune	***Sintaxă***
	`if`	`if (condiţie){` `  cod JavaScript` `}` *unde,* *condiţie* poate fi orice expresie logică
	Execută secvenţa de instrucţiuni dintre acolade (cod JavaScript) dacă rezultatul evaluării *condiţie* este `TRUE`. În caz contrar, JavaScript ignoră *cod* şi continuă.	
Exemplu:	<script> a=13;b=2; if(a>b){ alert("a este mai mare ca b"); } </script>	
Exemplu:	<script> numarator=4; numitor=0; if(numitor>0){ a=numarator/numitor; document.write(a); } </script>	
Exemplu:	<script>	

Figura 3.26

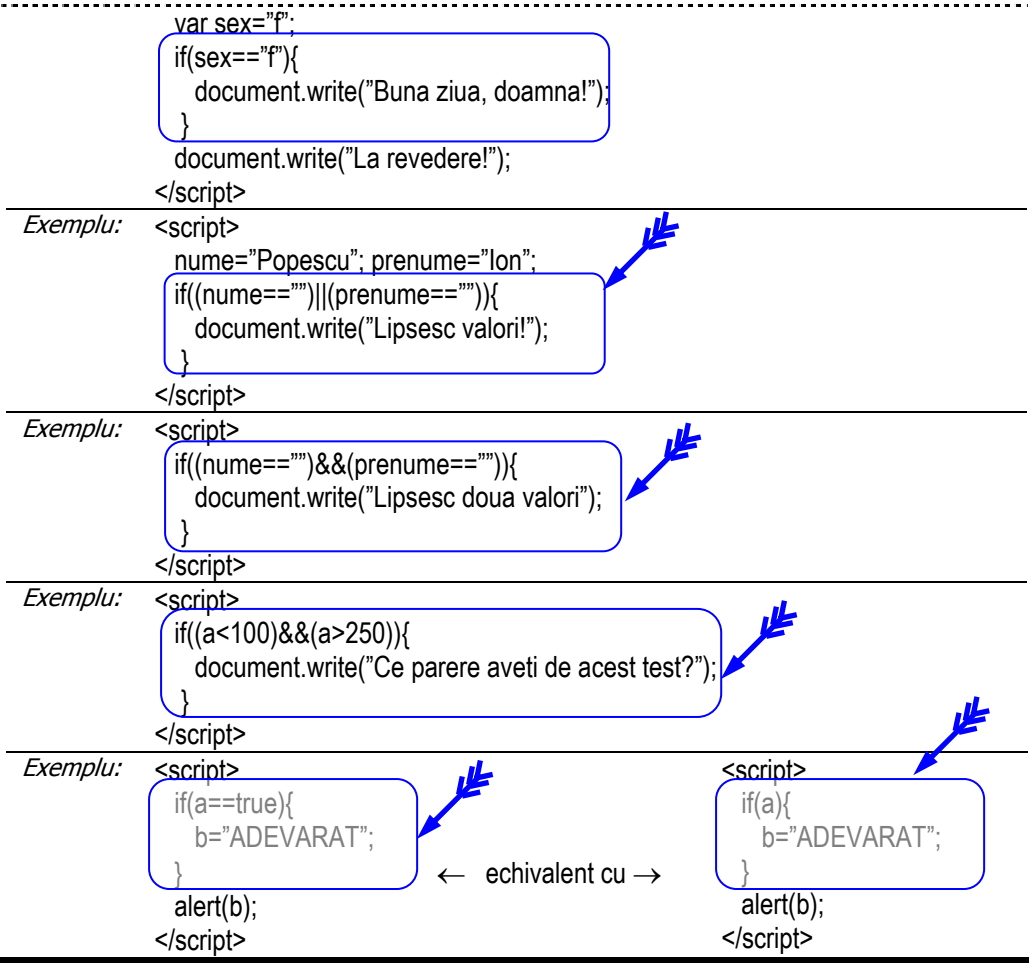


Figura 3.26 *(continuare)*

Mai multe despre if

- ✓ `if` este o structură de decizie (*decision structure*, în limba engleză) care testează şi compară diferite valori.
- ✓ Dacă partea a doua a unei instrucţiuni `if` poate conţine orice instrucţiune JavaScript, prima parte (*condiţie*) trebuie să respecte o sintaxă precisă.
- ✓ *Condiţie* (expresie condiţională) conţine în general două valori (constante, expresii, variabile etc.) care se compară una cu cealaltă. Cele două valori sunt separate printr-un operator relaţional.
- ✓ Nu confundaţi operatorul de egalitate (== sau ===) cu operatorul de afectare (=), chiar dacă ele se citesc „egal".
- ✓ Nu de puţine ori suntem puşi în situaţia să comparăm o variabilă cu mai multe valori sau cu mai multe variabile o dată. Pentru astfel de cazuri folosiţi operatorii logici.

if ... else (dacă ... în caz contrar)

Uneori, simpla folosire a instrucţiunii `if` nu este suficientă. În acest caz, completaţi instrucţiunea `if` cu cuvântul cheie `else` (*în caz contrar*, în limba română). `else` precizează interpretorului ce anume trebuie să facă atunci când rezultatul evaluării *condiţie* este `false`.

Instrucţiunea `if … else` este prezentată în detaliu în figura 3.27.

	Instrucţiune	***Sintaxă***
	`if … else`	`if (condiţie) {` `  cod1 JavaScript` `}` `else {` `  cod2 JavaScript` `}` *unde*, *cod1 JavaScript*: instrucţiuni executate când rezultatul evaluării *condiţie* este `TRUE`. *cod2 JavaScript*: instrucţiuni executate când rezultatul evaluării *condiţie* este `FALSE`.
	Execută prima secvenţă de instrucţiuni (*cod1 JavaScript*) dintre acolade ({}) dacă rezultatul evaluării *condiţie* este `TRUE`. Blocul (*cod2 JavaScript*) `else` nu este obligatoriu. El conţine instrucţiuni care se execută dacă rezultatul evaluării *condiţie* este `FALSE`. Dacă blocul `else` nu există şi dacă rezultatul evaluării *condiţie* este `FALSE`, instrucţiunile blocului `if` nu sunt executate. Script-ul transmite controlul execuţiei instrucţiunii imediat următoare după bloc. Dacă rezultatul evaluării *condiţie* este `TRUE` instrucţiunile blocului `else` nu sunt executate. Script-ul transmite controlul execuţiei instrucţiunii imediat următoare după blocul `else`.	
Exemplu:	<script> var sex="f"; if(sex=="f"){ document.write("Buna ziua, doamna!"); } else{ document.write("Buna ziua, domnule!"); } document.write(" La revedere!"); </script>	

Figura 3.27

Exemplu:

```
<script>
  var sex="f";
  if((sex=="f")||(sex=="m")){
    document.write("Sex:"+sex);
  }
  else{
    alert("Sex nedefinit!");
  }
  document.write("<br />");
  document.write("La revedere!");
</script>
```

Exemplu:

```
<script>
  var oras="Ploiesti";
  if(oras=="Paris"){
    document.write("Locuinta dumneavoastră este in capitala Frantei");
  }
  else{
    if (oras=="Sinaia"){
      document.write("Locuinta dumneavoastră este la Sinaia");
    }
    else{
      if(oras=="Ploiesti"){
        document.write("Locuinta dumneavoastră este la Ploiesti");
      }
    }
  }
</script>
```

Exemplu:

```
<script>
  var oras="Caracal";
  if(oras=="Paris"){
    document.write("Locuinta dumneavoastră este in capitala Frantei");}
  else if (oras=="Sinaia"){
      document.write("Locuinta dumneavoastră este la Sinaia");}
    else if(oras=="Ploiesti"){
        document.write("Locuinta dumneavoastră este la Ploiesti");}
</script>
```

Exemplu:

```
<script>
  var tara="Romania";
  var oras="Vaslui";
  if(tara==""Romania"){
    if(oras=="Bucuresti"){
      document.write("Locuinta ta de vara este in capitala Romaniei");}
    else{
      document.write("Locuinta ta de vara este in Romania, in provincie");}
  }
  else{
    document.write("Locuinta ta de vara nu este in Romania");
  }
</script>
```

Figura 3.27
(continuare)

Mai multe despre if ... else

- ✓ Instrucţiunile `if ... else` pot fi imbricate.
- ✓ Ca şi instrucţiunea `if`, `else` poate fi urmat de una sau mai multe instrucţiuni delimitate de acolade ({}).
- ✓ Nu uitaţi de expresiile condiţionale prescurtate (vezi figura 3.28, figura 3.29).

```
<script>
    x=(a==1)?1:0;        ← echivalent cu →        if(a==1){
</script>                                             x=1;
                                                    }
                                                   else {
                                                      x=0;
                                                    }
                                                </script>
```

Figura 3.28

```
<script>
    document.write(i+((i==1)?"cuvant gasit":"cuvinte gasite."));
</script>
```

Figura 3.29

switch

Instrucţiunea `switch` este prezentată în detaliu în figura 3.30.

Instrucţiune	*Sintaxă*
`switch`	`switch (variabila)` `  case Valoare1` `  cod1 JavaScript` `  break;` `  case Valoare2` `  cod2 JavaScript` `  break;` `  default:` `  cod3 JavaScript` `}`
	`Switch` controlează conţinutul unei variabile cu una sau mai multe instrucţiuni `case`. Instrucţiunea `break` este indispensabilă. Instrucţiunea `default` este facultativă. Ea conţine instrucţiunile care urmează a fi executate în situaţia în care nici unul din `case`-urile precedente nu conţin valoarea variabilei testate.
Exemplu:	<script> var n=100;

Figura 3.30

Figura 3.30 *(continuare)*

```
switch(n){
  case 10: document.write(”10”); break;
  case 20: document.write(”20”); break;
  case 30: document.write(”30”); break;
  default: document.write(”Alta valoare”);
}
document.write(”<br />La revedere!”);
</script>
```

Aplicaţie

❑ Precizaţi dacă rezultatele execuţiei celor două script-uri (figura 3.31 şi figura 3.32) sunt echivalente. Comentaţi rezultatele.

```
<script>
  var oras=”Ploiesti”;
  if (oras==”Ploiesti”)
    document.write(”Locuiti la Ploiesti”);
  if (oras==”Peris”)
    document.write(”Locuiti la Peris”);
  if (oras==”Paris”)
    document.write(”Locuiti la Paris”);
  if (oras==”Roman”)
    document.write(”Locuiti la Roman”);
  if (oras==”Roma”)
    document.write(”Locuiti la Roma”);
  document.write(”Sunteti un itinerant!”);
  document.write(”La revedere!”);
</script>
```

Figura 3.31

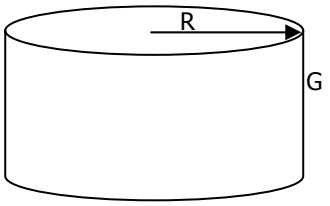
```
<script>
  var oras=”Ploiesti”;
  switch(oras){
    case ”Ploiesti”:
      document.write(”Locuiti la Ploiesti”); break;
    case ”Peris”:
      document.write(”Locuiti la Peris”); break;
    case ”Paris”:
      document.write(”Locuiti la Paris”); break;
    case ”Roman”:
      document.write(”Locuiti la Roman”); break;
    case ”Roma”:
      document.write(”Locuiti la Roma”); break;
    default:
      document.write(”Sunteti un itinerant!”);
  }
  document.write(”La revedere!”);
</script>
```

Figura 3.32

EXEMPLUL 3.4 JAVASCRIPT

Vom relua problema din EXEMPLUL 3.1 JAVASCRIPT simplificând-o după cum urmează.

Vom considera de această dată un singur rezervor cilindric echilateral (generatoarea este egală cu diametrul) plin cu benzină (vezi figura 3.33).



$G=2R$

$V=2\prod R^3$

Figura 3.33

Dorim să calculăm şi să afişăm într-o pagină Web cantitatea de lichid (benzină) din rezervor. Raza rezervorului trebuie să aibă o valoare pozitivă. Densitatea benzinei şi raza se vor introduce în mod dinamic.

❑ Analiza problemei

Formatul datelor de ieşire (fereastra în care se afişează pagina Web), tabela de variabile şi specificațiile de programare sunt ilustrate în figurile: 3.34, 3.35, 3.36.

a)

RAZA	**MASA**
-2	RAZA NEGATIVA

b)

RAZA	**MASA**
3	xxxxx.xxxxx

Figura 3.34

Tabela de variabile

Variabile de intrare	***Variabile de stare***	***Variabile de iesire***
d: densitatea benzinei r: raza rezervorului	m: cantitatea de benzină v: volumul rezervorului	m: cantitatea de benzină

Figura 3.35

Specificații de programare

Descriere. Script-ul calculează şi afişează cantitatea de benzină dintr-un rezervor cilindric echilateral de rază pozitivă.

Intrări. Densitatea benzinei, raza rezervorului.

Ieşiri. Masa de benzină din rezervor.

Lista de funcțiuni ale script-ului

1. Citeşte densitate benzină (d)
2. Citeşte raza rezervorului (r)
3. Afişează capul de tabel
4. Testează valoarea (introdusă dinamic) razei
5. Calculează volumul rezervorului (v).
6. Calculează masa (m) de benzină din rezervor
7. Afişează raza (r) şi masa rezervorului
8. În caz de eroare, afişează raza (r) şi mesajul: ”RAZA NEGATIVĂ”
9. Afişează un rând de liniuțe
10. Stop

Figura 3.36

❑ Proiectarea script-ului

În figura 3.37 se prezintă pseudocodul în scriere formalizată.

Pseudocodul

```
REZERVOR    BEGIN
INIT                BEGIN
                        //Citeşte densitatea benzinei (d)
                        READ(d)
                        //Citeşte raza rezervorului (r)
                        READ(r)
INIT                END
PRELUCR•RI          BEGIN
                        DO LINIUTZA
                        WRITE('RAZA'+'---'+'MASA');
                        DO LINIUTZA
DECIZIE                 IF(r<0)
                                WRITE(r+'  '+'Raza negativa');
DECIZIE                 ELSE
                                v=2∏r^3
                                m=d*v
                                WRITE(r+'...'+m)
DECIZIE                 ENDIF
PRELUCR•RI          END
                    DO LINIUTZA
REZERVOR    END
LINIUTZA    BEGIN
                    Afişează un rând de 30 de liniuțe
LINIUTZ•    END
```

Figura 3.37

Remarcă. Pseudocodul structurii de selecție `IF ... ELSE` este ilustrat în figura 3.38.

Figura 3.38

```
Eticheta    IF (condiţie)
                Grup de instrucţiuni
Eticheta    ELSE
                Grup de instrucţiuni
Eticheta    ENDIF
```

unde, `IF`, `ELSE` şi `ENDIF` sunt delimitatori ce poartă aceeaşi etichetă.

❑ Codificarea în limbajul JavaScript

În figura 3.39 este prezentat documentul (X)HTML complet.

Figura 3.39

c-03-04 - Notepad

File Edit Format View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Rezervoare</title>
<script language="JavaScript" type="text/JavaScript">
  d=parseFloat(prompt('Densitatea benzinei=?',0));
  r=parseFloat(prompt('Raza rezervorului=?',0));
  s=0;
  function liniutza(){
    document.write("<hr width='75%' align='left'>");
  }
</script>
</head>
<body>
  <script language="JavaScript" type="text/JavaScript">
    liniutza();
    document.write('<pre>RAZA                    MASA</pre>');
    liniutza();
    if(r<0){
        document.write(r+"  "+"Raza negativa"+"<br />");
      }
    else{
        v=2*Math.PI*(Math.pow(r,3));
        m=d*v;
        document.write(r+'  '+m+'<br />');
        }
    liniutza();
  </script>
</body>
</html>
```

În figura 3.40 şi figura 3.41 se prezintă rezultatele execuţiei script-ului.

Figura 3.40

Rezervoare - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Address D:\Carte Java\Aplicatii\c-03-04.htr Go

RAZA MASA

-5 Raza negativa

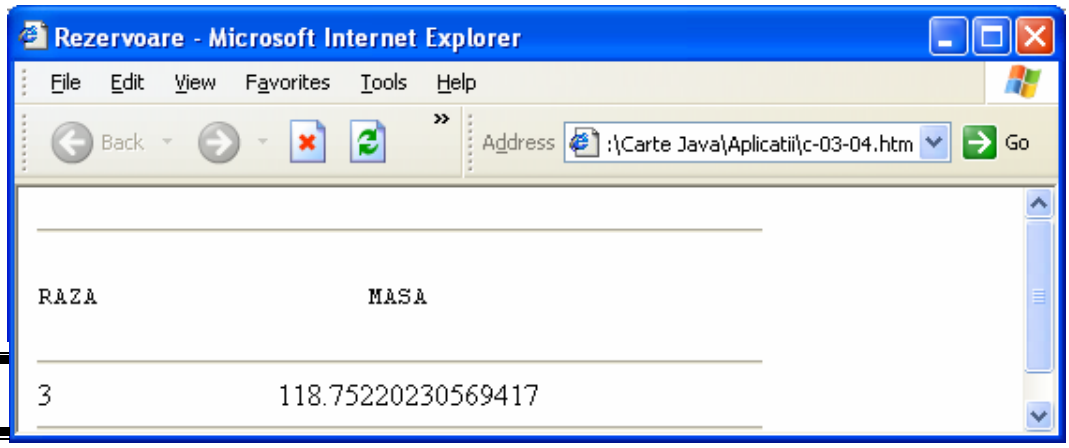


Figura 3.41

Instrucţiunile continue şi break

continue

Atunci când interpretorul JavaScript întâlneşte instrucţiunea `continue` el revine la începutul buclei şi ignoră celelalte instrucţiuni din interiorul buclei.

Instrucţiunea `continue` este prezentată în detaliu în figura 3.42.

	Instrucţiune	***Sintaxă***
	`continue`	`continue;` `continue` *`Eticheta`*
	Instrucţiunea este utilizată în buclele: `while`, `do ... while`, `for`. De cele mai multe ori, instrucţiunea este inclusă într-o instrucţiune `if`. `continue` poate fi de asemenea utilizată cu o etichetă.	
Exemplu:	<script> var i=1; while(i<21){ document.write(i); i++; if(i<10){ continue; } document.write(" "); } </script>	

Figura 3.42

Exemplu:

```
<script>
  iesire:;
  for(i=0;i<3;i++){
    document.write("i="+i+"<br />");
    for(j=0;j<5;j++){
      if(j==3){
      continue iesire;
      }
      document.write("j="+j+"<br />");
    }
  }
  document.write("SFARSIT");
</script>
```

Exemplu:

```
<script>
  iesire:;
  for(i=0;i<3;i++){
    document.write("i="+i+"<br />");
    for(j=0;j<5;j++){
      if(j==3){
      continue;
      }
      document.write("j="+j+"<br />");
    }
  }
  document.write("SFARSIT");
</script>
```

Figura 3.42 *(continuare)*

break

Instrucţiunea `break` provoacă o ieşire imediată din blocul de instrucţiuni curent.

Instrucţiunea `break` este prezentată în detaliu în figura 3.43.

Instrucţiune	***Sintaxă***
`break`	`break;` `break Eticheta`
Provoacă o ieşire imediată din cele trei bucle: `while`, `do … while`, `for` şi din blocul `switch`. De cele mai multe ori, instrucţiunea `break` este inclusă într-o instrucţiune `if`. Dacă rezultatul evaluării condiţiei este `TRUE` atunci se iese din buclă. `break` poate fi de asemenea utilizat cu o etichetă (un nume urmat de două puncte).	

Exemplu:

```
<script>
  var i=1; j=5;
  while(i<20){
    rezultat=i*j;
    document.write(rezultat+"<br />");
    if(r>50) break;
    i++;
  }
</script>
```

Figura 3.43

Exemplu:

```
<script>
  iesire:;
  for(i=0;i<3;i++){
    document.write("i="+i+" <br />");
    for(j=0;j<5;j++){
      if(j==3){
      break iesire;
      }
    document.write("j="+j+"<br />");
    }
  }
  document.write("SFARSIT");
</script>
```

Exemplu:

```
<script>
  iesire:;
  for(i=0;i<3;i++){
    document.write("i="+i+" <br />");
    for(j=0;j<5;j++){
      if(j==3){
      break;
      }
    }
  }
  document.write("SFARSIT");
</script>
```

Figura 3.43 *(continuare)*

Instrucţiunea with

`with` simplifică scrierea programelor JavaScript sau reduce pe cât posibil cantitatea de cod JavaScript. `with` permite specificarea unui obiect şi este urmat de un bloc de instrucţiuni plasat între acolade. Pentru fiecare din instrucţiunile blocului, proprietăţile menţionate fără ca obiectul corespunzător să fie indicat se referă la obiectul specificat prin `with`.

Instrucţiunea `with` este prezentată în detaliu în figura 3.44.

Instrucţiune	***Sintaxă***
`with`	`with (`*`obiect`*`) {` `  cod JavaScript` `}`
Modifică temporar contextul punând în evidenţă un obiect la care se fac referiri într-un bloc de instrucţiuni.	

Exemplu:

```
<script>
  //Următorul cod repetă metoda document.write() de patru ori
  document.write(”Pe trotuar, alături saltă”+”<br />”);
  document.write(”Două fete vesele,”+”<br />”);
  document.write(”Zău că-mi vine să las baltă,”+”<br />”);
  document.write(”Toate interesele”+”<br />”);
  //cuvântul cheie with elimină referințele multiple la obiectul document
  with(document){
    write(”Pe trotuar, alături saltă”+”<br />”);
    write(”Două fete vesele,”+”<br />”);
    write(”Zău că-mi vine să las baltă,”+”<br />”);
    write(”Toate interesele”+”<br />”);
  }
</script>
```

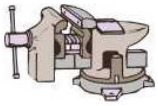
Figura 3.44

Pentru un cod scurt (vezi şi figura 3.45) interesul utilizării instrucţiunii `with` nu este evident, dar când trebuie să accesăm acelaşi obiect în cadrul unei proceduri sau când utilizăm un obiect predefinit precum `Math`, `with` vă ajută să câştigaţi foarte mult timp.

```
with(nume){
  window.alert(”Lungimea numelui este:” + length);
  toUpperCase();}
```

Figura 3.45

Testați-vă cunoștințele



1. Ce tipuri de instrucţiuni JavaScript cunoaşteţi?
2. Ce tip de buclă JavaScript testează condiţia la început:
   - `for`;
   - `while`;
   - `do ... while`.
3. Instrucţiunea `do ... while` asigură:
   - minimum o iteraţie;
   - maximum o iteraţie;
   - zero iteraţii.
4. Comentaţi următorul script (figura 3.46).

```
<script>
  i=0; rez=0; stare=true;
  document.write("0");
  while(stare){
    rez+=++I;
    if(i==10){
      rez=false;
    }
  }
  document.writeln("="+rez);
</script>
```

Figura 3.46

5. Care este rolul instrucţiunii switch?
6. Care este rolul instrucţiunilor break şi continue?
7. Scrieţi un script care calculează şi afişează valoarea polinomului:

   $5X^2+4X+10$

   pentru x=2.
8. Identificaţi erorile de sintaxă din următoarele script-uri:
   - Figura 11.6 (Conversaţia 11);
   - Figura 11.7 (Conversaţia 11);
   - Figura 11.8 (Conversaţia 11);
   - Figura 11.9 (Conversaţia 11);
   - Figura 11.10 (Conversaţia 11);

- Figura 11.12 (Conversaţia 11);
- Figura 11.13 (Conversaţia 11).

Vizitați site-urile

- ✓ www.bdml.net/listes
- ✓ www.natural.com/JavaScript
- ✓ http://www.webcoder.com
- ✓ http://JavaScript.internet.com/miscellaneous/
- ✓ http://www.geocities.com/jeffery_p_sanders/
- ✓ http://JavaScript.internet.com/forms/checkbox_changer.html

Conversația 4

Obiectele interne String, Array

În această conversație:

Obiectele limbajului JavaScript. Prezentare generală

După cum aţi putut constata, variabilele din programele pe care le-aţi realizat până acum servesc la stocarea valorilor: numerice, caracter, şir de caractere, booleene etc.

În egală măsură, JavaScript permite şi gestionarea obiectelor.

Obiectele JavaScript sunt de trei tipuri:

- ✓ Obiecte interne (integrate) – fac parte din limbajul JavaScript.
- ✓ Obiectele navigatorului – nu fac parte din limbajul JavaScript, dar sunt recunoscute de navigatoare.
- ✓ Obiecte personalizate – obiecte create de utilizator.

Obiectele interne

Cu trei din obiectele interne ale limbajului JavaScript, dumneavoastră aţi făcut deja cunoştinţă: `Math` care este un obiect predefinit JavaScript şi cuprinde numeroase constante şi funcţii; `Number` şi `Boolean` (vezi Conversaţia 2).

Lista completă a obiectelor interne (integrate) ale limbajului JavaScript este prezentată (în ordine alfabetică) în figura 4.1.

Obiect intern	***Descriere***
`Arguments`	Reprezintă (sub forma unei matrici) valoarea argumentelor transmise unei funcţii.
`Array`	matrice
`Boolean`	Conţine valorile: `TRUE` (adevărat) sau `FALSE` (fals).
`Date`	Obiect predefinit JavaScript care permite manipularea orelor şi a datelor calendaristice.
`Function`	Numele de obiect al unui obiect `Function` este considerat ca o variabilă reprezentând valoarea curentă a funcţiei create cu `new Function()`, în timp ce numele unei funcţii JavaScript standard nu este o variabilă, ci doar numele unei funcţii.
`Math`	Obiect predefinit JavaScript care conţine mai multe funcţii şi constante matematice.
`Number`	Facilitează gestiunea numerelor.
`Object`	Obiect JavaScript din care derivă toate celelalte obiecte.
`RegExp`	Obiect predefinit JavaScript dedicat expresiilor regulate.
`String`	Furnizează suportul pentru manipularea şi prelucrarea şirurilor de caractere.

Figura 4.1

Obiectul String

JavaScript stochează şirurile de caractere ca `obiecte String`.

Fişa obiectului `String` este prezentată în figura 4.2.

Fişa obiectului String	
Cum se creează obiectul?	`Constructorul String()`
Proprietăţi:	`length`
Metode:	`anchor(), big(), blink(), bold(), charAt(), charCodeAt(), concat(), fixed(), fontcolor(), fontsize(), fromCharCode(), indexOf(), italics(), lastindexOf(), link(), match(), replace(), search(), slice(), small(), split(), strike(), sub(), substr(), substring(), sup(), toLowerCase(), toString(), toUpperCase()`
Gestionarii de evenimente:	–

Figura 4.2

Constructorul String()

Constructorul `String()` creează un nou şir de caractere.

Constructorul `String()` este prezentat în detaliu în figura 4.3.

	Constructor	***Sintaxă***
	`String()`	`variabila=new String()` `variabila=new String(şir)` `variabila=şir`
	Se poate crea un şir de caractere vid sau transmiţând ca argument constructorului şirul de caractere care va conţine noul obiect `String`. Crearea şirurilor de caractere este de cele mai multe ori implicită. Constructorul `String()` poate fi de asemenea utilizat ca funcţie. În acest caz el converteşte valoarea în şir de caractere. Dacă nici o valoare nu este transmisă funcţiei, atunci acesta returnează un şir vid.	
Exemplu:	<script> //creare implicită a şirurilor de caractere anotimp="toamna"; celebrul="702"; //String() utilizat ca funcţie a=7; b=FALSE; document.write(String(a)+" "); document.write(String(b)+" "); </script>	

Figura 4.3

Proprietăţile obiectului String

Proprietăţile obiectului `String` sunt prezentate în detaliu în figura 4.4.

	Proprietate	***Sintaxă***
	`length`	`şir.length`
	Conţine lungimea şirului de caractere.	
Exemplu:	<script> demo_sir="Învăţaţi să priviţi dincolo de aparenţe!"; lungime=demo_sir.length; //afişează 40 </script>	

Figura 4.4

Remarcă. Un obiect `String` (scrieţi `S` cu majusculă) este diferit de datele (şir de caractere) pe care le conţine.

Metodele obiectului String

Metodele obiectului String pot fi clasificate după cum urmează:

- ✓ Metode pentru formatarea şirurilor de caractere – `big()`, `blink()`, `bold()`, `fontcolor()`, `fontsize()`, `italics()`, `small()`, `strike()`, `sub()`, `sup()`.
- ✓ Metode pentru manipularea şirurilor de caractere – `anchor()`, `charAt()`, `charCodeAt()`, `concat()`, `fixed()`, `fromCharCode()`, `indexOf()`, `lastIndexOf()`, `link()`, `match()`, `replace()`, `search()`, `slice()`, `split()`, `substr()`, `substring()`, `toLowerCase()`, `toString()`, `toUpperCase()`.

Metode pentru formatarea şirurilor de caractere

Metodele obiectului `String` pentru formatarea şirurilor de caractere sunt prezentate în detaliu în figura 4.5.

	Metode	***Sintaxă***
	`big()`	`şir.big()`
	Încadrează un şir de caractere cu tag-urile `<big>...</big>`.	
Exemplu:	<script> demo_sir="Învăţaţi să priviţi dincolo de aparenţe!"; nou_demo_sir=demo_sir.big(); document.write(nou_demo_sir); </script>	

Figura 4.5

`blink()`	•`ir.blink()`
	Încadrează un şir de caractere cu tag-urile `<blink>...</blink>`.
Exemplu:	<script> demo_sir="Învăţaţi să priviţi dincolo de aparenţe!"; nou_demo_sir=demo_sir.blink(); document.write(nou_demo_sir); </script>
`bold()`	`şir.bold()`
	Încadrează un şir de caractere cu tag-urile `<bold>...</bold>`.
Exemplu:	<script> demo_sir= "Învăţaţi să priviţi dincolo de aparenţe!"; nou_demo_sir=demo_sir.bold(); document.write(nou_demo_sir); </script>
`fontcolor()`	`şir.fontcolor(`*`şir`*`)`
	Atribuie şirului de caractere culoarea indicată în tag-ul `<font>` de atributul `color`.
Exemplu:	<script> demo_sir= "Învăţaţi să priviţi dincolo de aparenţe!"; nou_demo_sir=demo_sir.fontcolor("#FF0000"); document.write(nou_demo_sir); </script>
`fontsize()`	`şir.fontsize(`*`Valoare`*`)`
	Atribuie şirului de caractere dimensiunea indicată cu tag-ul `<font>` de atributul `size`.
Exemplu:	<script> demo_sir= "Învăţaţi să priviţi dincolo de aparenţe!"; nou_demo_sir=demo_sir.fontsize(4); document.write(nou_demo_sir); </script>
`italics()`	`şir.italics(`*`şir`*`)`
	Încadrează un şir de caractere cu tag-urile `<i>...</i>`.
Exemplu:	<script> demo_sir= "Învăţaţi să priviţi dincolo de aparenţe!"; nou_demo_sir=demo_sir.italics(); document.write(nou_demo_sir); </script>
`small()`	`şir.small(`*`şir`*`)`
	Încadrează un şir de caractere cu tag-urile `<small>...</small>`.
Exemplu:	<script> demo_sir= "Învăţaţi să priviţi dincolo de aparenţe!"; nou_demo_sir=demo_sir.small(); document.write(nou_demo_sir); </script>

Figura 4.5
(continuare)

	`strike()`	`şir.strike()`
	Încadrează un şir de caractere cu tag-urile `<strike>...</strike>`.	
Exemplu:	<script> demo_sir= ”Învățați să priviți dincolo de aparențe!”; nou_demo_sir=demo_sir.strike(); document.write(nou_demo_sir); </script>	
	`sub()`	`şir.sub()`
	Încadrează un şir de caractere cu tag-urile `<sub>...</sub>`.	
	`sup()`	`şir.sup()`
	Încadrează un şir de caractere cu tag-urile `<sup>...</sup>`.	

Figura 4.5 *(continuare)*

Metode pentru manipularea şirurilor de caractere

Metodele obiectului `String` pentru manipularea şirurilor de caractere sunt prezentate în detaliu în figura 4.6.

	Metode	***Sintaxă***
	`anchor()`	`şir.anchor(`*`şir`*`)`
	Converteşte un şir de caractere într-o ancoră numită cu tag-ul `<a>` şi atributul `name`.	
	`charAt()`	`şir.charAt(`*`Valoare`*`)`
	Returnează caracterul poziţionat în şirul de caractere.	
Exemplu:	<script> vocale=”aeiou”; r=vocale.charAt(3); document.write(r); //afişează o </script>	
	`charCodeAt()`	`şir.charCodeAt(`*`Valoare`*`)`
	Returnează valoarea codului Unicode al caracterului situat în poziţia indicată în şirul de caractere.	
	`concat()`	`şir.concat(`*`şir`*`)`
	Adaugă un nou şir de caractere la sfârşitul unui alt şir de caractere. Metoda este echivalentă cu operatorul `+`.	
Exemplu:	<script> sir_1=sir_1.concat(”Felicitări pentru răbdarea de a ne fi descoperit!”); document.write(sir_1); </script>	
	`fixed()`	`şir.fixed()`
	Încadrează un şir de caractere cu tag-urile `<tt>...</tt>`.	

Figura 4.6

	`fromCharCode()` `şir.fromCharCode(`*`Valoare`*`)`
	Returnează codul Unicode corespunzător caracterului indicat.
	`indexOf()` `şir.indexOf(`*`Şir,Valoare`*`)`
	Caută un caracter într-un şir de caractere şi returnează indexul primei apariţii (primul caracter are indexul 0). Metoda face diferenţă între majuscule şi minuscule. În caz contrar metoda returnează valoarea -1.
Exemplu:	<script> email="ld@canaba.com"; r=email.indexOf("@"); document.write(r); //afişează 2 </script>
Exemplu:	<script> email="pepito@brazil.com"; r=email.indexOf("i",6); document.write(r); //afişează 11 </script>
Exemplu:	<script> email="ld@canaba.com"; r=email.indexOf("canaba"); document.write(r); //afişează 3 </script>
	`lastIndexOf()` `şir.lastIndexOf(`*`şir`*`)`
	Caută ultima apariţie a şirului de caractere şi returnează indexul poziţiei sale. În caz de eşec, se returnează valoarea –1. Metoda face deosebire între majuscule şi minuscule.
Exemplu:	<script> email="ld@canaba.com"; r=email.lastindexOf("@"); document.write(r); //afişează 2 </script>
	`link()` `şir.link(`*`şir`*`)`
	Încadrează un şir de caractere cu tag-urile `<a>..</a>`.
	`match` `şir.match(`*`expresieregulată`*`)`
	Caută o expresie regulată într-un şir de caractere şi returnează rezultatul într-o matrice. În caz de eşec, metoda returnează valoarea `null`.
Exemplu:	<script> model=/\d{1,3}/; adresa="195.14.45.78"; document.write(adresa.match(model)); </script>
	`replace()` `şir.replace(`*`expresieregulată, şir`*`)`
	Caută o expresie regulată şi înlocuieşte caracterele găsite prin şirul de caractere indicat în cel de-al doilea argument. În caz de eşec, înlocuirea nu se efectuează.

Figura 4.6
(continuare)

Exemplu:
```
<script>
  model=/ION/i;
  text="ion@yahoo.com";
  r=text.replace(model."ION");
  document.write(r);  //afişează yahoo.com
</script>
```

`search()` — `şir.search(expresieregulată)`

Caută o expresie regulată într-un şir de caractere şi returnează valoarea poziţiei primului caracter găsit. În caz de eşec, metoda returnează valoarea –1.

Exemplu:
```
<script>
  model=/pion/i;
  text="Nebunul şi Pionul";
  document.write(text.search(model));
  //afişează 11
</script>
```

`slice()` — `şir.slice(Valoare1,Valoare2)`

Extrage o parte dintr-un şir de caractere şi returnează şirul extras.

Exemplu:
```
<script>
  text=liv@canaba.com;
  a=text.slice(4,100);
  document.write(a);
  //afişează canaba.com
</script>
```

Exemplu:
```
<script>
  text="ion@yahoo.com";
  rezultat=text.slice(0,3);
  document.write(rezultat);
  //afişează ion
</script>
```

`split()` — `şir.split(şir)`

Divizează un şir de caractere în subşiruri şi returnează o matrice. Metoda recunoaşte expresiile regulate.

`substr()` — `şir.substr(Valoare1,Valoare2)`

Extrage o parte din şirurile de caractere şi returnează şirul extras.

Exemplu:
```
<script>
  text="ion@yahoo.com";
  r=text.substr(0,3);
  document.write(r);
  //afişează ion
</script>
```

`substring()` — `şir.substring(Valoare1,Valoare2)`

Extrage o parte dintr-un şir de caractere şi returnează şirul extras.

Exemplu:
```
<script>
  email="ion@yahoo.com";
  r=email.substring(0,3);
  document.write(r);  //afişează ion
</script>
```

Figura 4.6
(continuare)

	`toLowerCase()` `şir.toLowerCase()`
	Converteşte un şir de caractere în minuscule.
Exemplu:	<script> text="Mitică şi Petrică"; r=text.toLowerCase(); document.write(r); //afişează mitică şi petrică </script>
	`toString()` `şir.toString()`
	Converteşte o valoare numerică sau booleană în şir de caractere.
Exemplu:	<script> a=TRUE; b=314; document.write(a.toString()+","); document.write(b.toString()); //afişează TRUE , 314 </script>
	`toUpperCase()` `şir.toUpperCase(şir)`
	Converteşte un şir de caractere în majuscule.
Exemplu:	<script> email="ion@yahoo.com"; rezultat=email.toUpperCase(); document.write(rezultat); //afişează ION@YAHOO.COM </script>

Figura 4.6
(continuare)

Obiectul Array

Aplicaţiile din conversaţiile anterioare au fost construite pe tipuri de date simple cărora le-au fost asociate variabile simple.

Sunteţi un ... gurmand de variabile? Dacă da, folosiţi *obiectul intern Array* (matrice).

O matrice (array) este o listă de valori sau de referinţe către alte obiecte. O matrice poate conţine, de exemplu o listă de date numerice sau alfanumerice.

Fişa obiectului intern `Array` este prezentată în figura 4.7.

Fişa obiectului Array	
Cum se creează obiectul?	`constructorul Array()`
Proprietăţi:	`length`
Metode:	`concat(), join(), pop(), push(), reverse(), shift(), slice(), sort(), splice(), toString(), unshift()`
Gestionarii de evenimente:	–

Figura 4.7

Constructorul Array()

Pentru a crea o nouă matrice, utilizaţi constructorul `Array()`.

Constructorul `Array()` este prezentat în detaliu în figura 4.8.

	Constructor	***Sintaxă***
	`Array()`	`variabila=new Array()` `variabila=new Array(`*`număr_elemente`*`)` `variabila=new Array(`*`element1,element2,...`*`)` `variabila=[]` `variabila=[,,,,]` `variabila=[`*`element1,element2,…`*`]`
	Creează o nouă matrice. Noua matrice poate fi vidă, dimensiunea sa (numărul de elemente) poate fi predefinită sau poate fi, de asemenea iniţializată cu o listă de elemente. Numerotarea elementelor unei matrici începe de la zero. Spre deosebire de majoritatea tipurilor de variabile JavaScript, array-urile trebuie să fie declarate înainte de a fi utilizate. Începând cu versiunea 1.2 JavaScript, instrucţiunea `new Array()` nu mai este indispensabilă. Se pot utiliza, în mod simplu parantezele drepte ([]). În limbajul JavaScript nu se pot crea matrici cu mai multe dimensiuni, dar puteţi imbrica mai multe matrici. Începeţi prin a crea o matrice clasică (cu o dimensiune) iar apoi, în fiecare element al vectorului inseraţi câte un `array`.	
Exemplu:	<script> ListaCulori=new Array(); ListaCulori=[“rosu”,”galben”,”albastru”]; </script>	
Exemplu:	<script> ListaCulori=new Array(7); </script>	
Exemplu:	<script> ListaCulori=new Array(“rosu”,”galben”,”albastru”); </script>	

Figura 4.8

Exemplu:

```
<script>
  ListaCulori=[“rosu”,”galben”];
</script>
```

Exemplu:

```
<script>
  culoare1=”rosu”;
  culoare2=”galben”;
  culoare3=”albastru”;
  ListaCulori=[culoare1,culoare2, culoare3];
</script>
```

Exemplu:

Tabelul 1

a11	a12	a13	a14
a21	a22	a23	a24
a31	a32	a33	a34

```
<script>
  //creare matrice (vezi tabelul 1) cu trei linii şi patru coloane
  linie=new Array(3);
  linie[0]=new Array(“a11”,”a12”,”a13”,”a14”);
  linie[1]=new Array(“a21”,”a22”,”a23”,”a24”);
  linie[2]=new Array(“a31”,”a32”,”a33”,”a34”);
  document.write(linie[1][2]+”<br>”);
//Se afişează a23
  document.write(linie[0]);
//Se afişează a11, a12, a13, a14
</script>
```

Exemplu:

```
<script>
  ListaCulori=new Array(”rosu”,”galben”,”albastru”,”verde”);
  document write(ListaCulori+”<br />”);
//Se afişează rosu, galben, albastru, verde
  ListaCulori[1]=”violet”;
  document.write(ListaCulori);
//Se afişează rosu, violet, albastru, verde
</script>
```

Exemplu:

```
<script>
/*Crearea unei matrici şi modificarea celui de-al doilea element*/
  mere=new Array(”ionatan”,”cretesti”,”dulci”);
  document.write(mere+”<br>”);
//Se afişează ionatan, creţeşti, dulci
  mere[1]=”parmen-auriu”;
  document.write(mere);
</script>
```

Exemplu:

```
<script>
/*Afişarea elementelor unei matrici cu o buclă for*/
  student=new Array(”Alin”,”Bogdan”,”Catalin”,”Dan”);
  for(i=0;i<4;i++){
    document.write(student[i]+”<br>”);
  }
</script>
```

Figura 4.8
(continuare)

Proprietăţile obiectului Array

Proprietăţile obiectului `Array()` sunt prezentate în detaliu în figura 4.9.

	Proprietate	***Sintaxă***
	`length`	`matrice.length`
	Conţine numărul de elemente al unei matrici. Puteţi modifica valoarea acestei proprietăţi. Reducând numărul de elemente, elementele se suprimă pornind din dreapta matricii.	
Exemplu:	<script> lista_numere=new Array(3,7,9,50,20,23,2); document.write(lista_numere.length+" numere"); //Se afişează 7 numere </script>	
Exemplu:	<script> lista_numere=new Array(6,9,7,56,48); document.write(lista_numere.length+" numere "); for(i=0;i<lista_numere.length,i++){ document.write(lista_numere[i]+'–'); } lista_numere.length–=2; document.write(" "+lista_numere.length+" numere "); for(i=0;i<lista_numere.length;i++){ document.write(lista_numere[i]+" –"); } </script>	

Figura 4.9

Metodele obiectului Array

Metodele obiectului `Array` sunt prezentate în detaliu în figura 4.10.

	Metodă	***Sintaxă***
	`concat()`	`matrice=matrice.concat(`*`matrice`*`)` `matrice=matrice.concat(`*`Element1, Element2,...`*`)`
	Metoda este utilizată pentru concatenarea a două matrici.	
Exemplu:	<script language="javascript" type="text/javascript"> //concatenarea a două matrici lista_numere1=new Array(6,9,7); lista_numere2=new Array(5,68); lista_numere3=lista_numere1.concat(lista_numere2); document.write(lista_numere3.length+" numere "); for(i=0;i<lista_numere3.length;i++){ document.write(lista_numere3[i]+"–"); } </script>	

Figura 4.10

Exemplu:
```
<script language="javascript" type="text/javascript">
  //concatenarea a două matrici
  lista_numere1=new Array(6,9,7);
  lista_numere3=lista_numere1.concat(13,22);
  document.write(lista_numere3.length+" numere <br />");
  for(i=0;i<lista_numere3.length;i++){
    document.write(lista_numere3[i]+"–");
  }
</script>
```

`join()` `matrice.join()` `matrice.join("Separator")`

Metoda este utilizată pentru convertirea unei matrici într-un şir de caractere. Argumentul "*Separator*" serveşte la separarea elementelor in şir. Fără argument, metoda `join()` utilizează virgula.

Exemplu:
```
<script>
  subsir=new Array (6,9,7);
  sir=subsir.join();
  document.write(sir+"<br />");
  sir=subsir.join("*"); document.write(sir);
</script>
```
Rezultatele execuţiei script-ului
6,9,7 6*9*7

`pop()` `matrice.pop()`

Metoda suprimă ultimul element al unei matrici. Proprietatea `length` este modificată în mod automat. Metoda returnează valoarea elementului suprimat. Metoda `pop()` nu cere nici un argument.

Exemplu:
```
<script language="javascript" type="text/javascript">
  lista=new Array(1,9,47);
  el_suprimat=lista.pop();
  document.write("Elementul suprimat: "+el_suprimat+"<br />");
  document.write(lista);
</script>
```

`push()` `matrice.push()`

Metoda adaugă noi elemente la sfârşitul matricii. Proprietatea `length` este modificată în mod automat.

Exemplu:
```
<script language="javascript" type="text/javascript">
  culori=new Array("rosu","galben");
  adaugă=culori.push("albastru");
  document.write("Numărul de elemente: "culori.length+ "<br />");
  document.write(culori);
</script>
```

`reverse()` `matrice.reverse()`

Metoda inversează ordinea elementelor unei matrici (nu cere nici un argument).

Exemplu:
```
<script language="javascript" type="text/javascript">
  culori=new Array("rosu","galben","albastru");
  document.write(culori+"<br />");
  culori_invers=culori.reverse();
  document.write(culori);
</script>
```

Figura 4.10
(continuare)

	`shift()` `matrice.shift()`
	Metoda suprimă primul element al unei matrici. Elementul de rang 1 devine de rang 0, cel de rang 2 devine de rang 1 ş.a.m.d. Proprietatea `length` se modifică în mod automat. Metoda returnează valoarea elementului suprimat şi nu cere nici un argument.
Exemplu:	<script language="javascript" type="text/javascript"> lista_numere=new Array(13,20,22); suprim=lista_numere.shift(); document.write("Elementul suprimat: "+suprim+" "); document.write(lista_numere); </script>
	`slice()` `matrice.slice(`*`ValoareNumerica1`*`,` *`ValoareNumerica2`*`)`
	Metoda extrage elementele unei matrici şi returnează o nouă matrice care conţine aceste elemente. Matricea iniţială nu este modificată. Elementele extrase depind de două argumente: *ValoareNumerica1* (indică rangul primului element care urmează a fi extras), *ValoareNumerica2* (indică rangul imediat următor ultimului element care urmează a fi extras). Dacă primul argument este negativ, atunci rangul se calculează pornind de la sfârşitul matricii. El este întotdeauna de rang 0. Dacă cel de-al doilea argument este negativ, el indică rangul (calculat, pornind de la sfârşitul matricii) ultimului element care urmează a fi extras. Dacă nu este menţionat, caracterele sunt extrase începând cu rangul furnizat de primul argument până la sfârşitul matricii.
Exemplu:	<script language="javascript" type="text/javascript"> lista_numere=new Array(1,2,3,4,5,6,7,8); lista_1=lista_numere.slice(2,5); document.write(lista_1+" "); //se afişează 3,4,5 lista_2=lista_numere.slice(3); document.write(lista_2+" "); //se afişează 4,5,6,7,8 lista_3=lista_numere.slice(2,–1); document.write(lista_3+" "); //se afişează 3,4,5,6,7 lista_4=lista_numere.slice(–2); document.write(lista_4+" "); //se afişează 7,8 </script>
	`splice()` `matrice.splice(`*`Rang,NumarElemente,`* *`Element1,Element2,...`*`)`
	Metoda suprimă elementele unei matrici şi le înlocuieşte cu altele. Rangul elementului de la care trebuie să înceapă suprimarea elementelor matricii este obligatoriu. Al doilea argument (*NumarElemente*) precizează numărul elementelor ce urmează a fi suprimate. Argumentele următoare (*Element1,Element2,...*) reprezintă elementele care urmează a fi inserate în matrice începând cu rangul indicat pentru primul element. Metoda returnează lista elementelor suprimate.

Figura 4.10
(continuare)

Exemplu:
```
<script language="javascript" type="text/javascript">
  lista_numere=new Array(1,2,3,4,5,6,7,8);
  a=lista_numere.splice(2,2,35,36);
  document.write("Numere: "lista_numere+"<br />");
  //se afişează 1,2,35,36,5,6,7,8
</script>
```

`toString()` `matrice.toString()`

Metoda returnează conţinutul unei matrici sub forma unui şir de caractere.

Exemplu:
```
<script language="javascript" type="text/javascript">
  culori=new Array("rosu","galben","albastru");
  a=culori.toString();
  document.write(a);
  //se afişează rosu,galben,albastru
</script>
```

`unshift()` `matrice.unshift(`*`Element1,Element2,..`*`)`

Metoda inserează la începutul matricii elementele pe care le menţionaţi. Proprietatea `length` este modificată în mod automat.

Exemplu:
```
<script language="javascript" type="text/javascript">
  culori=new Array("rosu","galben","albastru");
  culori.unshift("verde","violet");
  document.write(culori);
  //se afişează verde,violet,rosu,galben,albastru
</script>
```

`sort()` `matrice.sort()`
`matrice.sort(`*`Funcţie`*`)`

`matrice.sort()` sau metoda `sort` fără argument sortează elementele unei matrici care conţine valori alfanumerice; `matrice.sort(`*`Func•ie`*`)` permite sortarea valorilor numerice. *`Func•ie`* primeşte doi parametri şi returnează o valoare numerică.

Figura 4.10
(continuare)

Exemplu:

c_5-11 - Notepad

File Edit Format View Help

```
<html>
<head>
  <title>Sortare alfanumerica</title>
</head>

<body>
 <p>Introduceti numele mai multor rezervoare
    cilindrice echilaterale. Afisati aceste nume
    intr-o lista ordonata sortata</p>
<script language="javascript" type="text/javascript">
  nume_rezervor=new Array();
  i=0;
  do{
    num_rez=window.prompt("Introduceti urmatorul
                           nume al rezervorului","");
    if(num_rez>"") nume_rezervor[i]=num_rez;
    i++;
  }
  while(num_rez>"");
  document.write("<h2>"+"Au fost introduse: "+
    (nume_rezervor.length)+" nume de rezervoare."+
     "</h2>");
  nume_rezervor_sort=nume_rezervor.sort();
  document.write("<ol>");
  for(i in nume_rezervor_sort){
       document.write("<li>"+nume_rezervor_sort[i]+
       "<br>");
  }
  document.write("</ol>")
</script>
</body>
</html>
```

Rezultatele execuţiei script-ului:

Sortare alfanumerica - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Address D:\Carte Java\ Go

Introduceti numele mai multor rezervoare cilindrice echilaterale. Afisati aceste nume intr-o lista ordonata sortata

Au fost introduse: 4 nume de rezervoare.

1. Brazi
2. Onesti
3. Petromidia
4. Pitesti

Figura 4.10
(continuare)

Exemplu:

c_5-16 - Notepad

File Edit Format View Help

```
<script language="javascript" type="text/javascript">
  tabel=new Array(13);
  tabel=[33,57,31,220,79,80,49,81,220,999,0,4,3];
  function permutare(a,b){
    c=tabel[a];
    tabel[a]=tabel[b];
    tabel[b]=c;
   }
  function quicksort(stanga,dreapta){
    if(stanga>=dreapta) return;
    permutare(stanga,Math.floor(stanga+dreapta)/2);
    ultimul=stanga;
    for(i=stanga+1;i<=dreapta;i++)
     if(tabel[i]<tabel[stanga])
       permutare(++ultimul,i);
     permutare(stanga,ultimul);
     quicksort(stanga,ultimul-1);
     quicksort(ultimul+1,dreapta);
    }
  function afisare(){
     for(i=0;i<13;i++)
       document.write(tabel[i]+" ");
    }
  document.write("<h3> Matricea nesortata </h3>");
  afisare();
  quicksort(0,12);
  document.write("<h3> Matricea sortata </h3>");
//  afisare();
  document.write("0 3 4 31 33 49 57 79 80 81 220
                  220 999");
</script>
```

Rezultatele execuţiei script-ului:

Sortare prin metoda QuickSort - Microsoft Internet Exp...

File Edit View Favorites Tools Help

Back Address c_5-16.htm Go

Matricea nesortata

33 57 31 220 79 80 49 81 220 999 0 4 3

Matricea sortata

0 3 4 31 33 49 57 79 80 81 220 220 999

Figura 4.10
(continuare)

EXEMPLUL 4 JAVASCRIPT

❑ Formularea problemei

Problema care se pune acum seamănă cu celelalte, dar ... este puțin mai complicată. Se citesc de la tastatură, într-o matrice livrările de benzină efectuate zilnic (luni, marți, miercuri, joi, vineri) din cele trei rezervoare (cilindrice echilaterale) R1, R2, R3. Se doreşte ca, pentru fiecare rezervor în parte, să se afişeze, pe zile, cantitățile de benzină livrate beneficiarilor. Se va tipări, de asemenea, media livrărilor pe zile şi pe rezervoare. La finele raportului se vor afişa maximul şi minimul livrărilor, precizându-se atât rezervorul cât şi ziua respectivă.

❑ Analiza problemei

Formatul datelor de input/output (intrare/ieşire), tabela de variabile, specificațiile de programare sunt ilustrate în figurile: 4.11, 4.12, 4.13.

Figura 4.11

Tabela de variabile		
Variabile de intrare	***Variabile de stare***	***Variabile de iesire***
a: matricea livrărilor pe zile şi rezervoare	st: livrările totale pe zile şi rezervoare	b: vectorul livrărilor medii zilnice
	s: sume parțiale, livrări pe zile	d: vectorul livrărilor medii pe rezervoare
	z: vectorul numelor zilelor săptămânii	max: maximul livrărilor
	imax, imin, jmax, jmin: indicii livrărilor maxime şi minime din matricea a	min: minimul livrărilor

Figura 4.12

Specificații de programare

Descriere. Programul editează situația livrărilor de benzină efectuate zilnic din trei rezervoare cilindrice echilaterale.

Livrările, pe zile şi pentru fiecare rezervor în parte, se citesc de la tastatură. Programul mai afişează maximul şi minimul livrărilor însoțite de rezervorul şi ziua respective.

Intrări. Matricea livrărilor pe rezervoare şi zile.

Ieşiri. Lista cu situația livrărilor.

Lista de funcțiuni ale script-ului

1. Citeşte livrările zilnice de benzină (luni, marți, miercuri, joi, vineri) pentru rezervorul R1.
2. Citeşte livrările zilnice de benzină (luni, marți, miercuri, joi, vineri) pentru rezervorul R2.
3. Citeşte livrările zilnice de benzină (luni, marți, miercuri, joi, vineri) pentru rezervorul R3.
4. Calculează media livrărilor pe zile.
5. Calculează media livrărilor pe rezervoare.
6. Calculează maximul livrărilor.
7. Calculează minimul livrărilor.
8. Afişează maximul livrărilor, numărul rezervorului şi ziua.
9. Afişează minimul livrărilor, numărul rezervorului şi ziua.
10. Trunchiază media (livrărilor pe zile şi pe rezervoare).
11. Stop

Figura 4.13

❑ Proiectarea script-ului

În figura 4.14 se prezintă pseudocodul, varianta formalizată.

Pseudocodul

```
Exemplul5   BEGIN
            //initializeaza vectorul Z cu numele zilelor
            z=luni,marti,miercuri,joi vineri
            //aloca spatiu de memorie si
            //citeste livrarile pe fiecare rezervor de la tastatura
LIVI        FOR(i=0;i<3;i++)
LIVJ                FOR(j=0;j<5;j++)
                        READ a_i,j
LIVJ                ENDFOR
LIV         ENDFOR
            // aloca spatiu de memorie pentru vectorul b si d
            // calculeaza mediile pe rezervoare
FORJ        FOR(j=0;j<5;j++)
                    s=0
FORI                FOR(i=0;i<3;i++)
                        s=s+a_i,j
FORI                ENDFOR
                    b_j=s/3
FORJ        ENDFOR
        // calculul mediilor pe rezervoare
            st=0
FORMEDII    FOR(i=0;i<3;i++)
                    s=0
FORMEDIIJ           FOR(j=0;j<5;j++)
                        s=s+a_i,j
                        st=st+a_ij
FORMEDIIJ           ENDFOR
                    d_i=s/5
FORMEDII    ENDFOR
            d_3=s/15
            // determinarea maximului si minimului
            max=a_0,0
            min=a_0,0
            imax=0
            imin=0
            jmax=0
            jmin=0
FORMAXI     FOR(i=0;i<3;i++)
FORMAXJ             FOR(j=0;j<5;j++)
IFMAX                       IF(max<a_i,j)
                                max=a_ij
                                imax=i
                                jmax=j
IFMAX                       ENDIF
IFMIN                       IF(min>a_i,j)
                                min=a_ij
                                imin=i
                                jmin=j
IFMIN                       ENDIF
FORMAXJ             ENDFOR
FORMAXI     ENDFOR
            imin=imin+1
            imax=imax+1
            //afisare rezultate
            WRITE "SITUATIA REZERVOARELOR R1 R2 R3"
            WRITE "ZIUA  R1  R2  R3  MEDIA"
```

Figura 4.14

```
FORK            FOR(k=0;k<5;k++)
                        WRITE Z[k]
FORJ                    FOR(j=0;j<3;j++)
                                WRITE  a[j][k]
FORJ                    ENDFOR
                        WRITE  trunchiaza( b[k])
FORK            ENDFOR
                WRITE  "MEDIA"
FORTRUNC        FOR(j=0;j<4;j++)
                        WRITE trunchiaza(d[j])
FORTRUNC        ENDFOR
                WRITE "Livrarea maxima: " max
                WRITE "s-a facut din rezervorul: R" imax
                WRITE "in ziua de" Zjmax
                WRITE "Livrarea minima:" min
                WRITE " s-a facut din rezervorul: R"imin
                WRITE " in ziua de "+Zjmin
Exemplul5       END
// transforma o valoare reala in sir de caractere
// si trunchiaza la doua zecimale
TRUNCHIAZA BEGIN
                Parametrii:
                        x- valoare reala
                s=transforma in sir de caractere x
                i=pozitia punctului zecimal in sir
IFCOPY          IF(i≠-1)
                        s=copiazasubsir(s,0,i+2)
IFCOPY          ENDIF
                RETURN s
TRUNCHIAZA END
```

Figura 4.14 *(continuare)*

❑ Codificarea în limbajul JavaScript

În figura 4.15 este prezentat documentul (X)HTML complet.

```
<html>
<head>
<title>Exemplul 4</title>
<script language="javascript">
<!---
var Z = new Array("Luni","Marti","Miercuri","Joi","Vineri");
a=new Array(3);
a[0]=new Array(5);
a[1]=new Array(5);
a[2]=new Array(5);
for(i=0;i<3;i++)
 for(j=0;j<5;j++)
   {k=i+1;
   a[i][j]=parseFloat(prompt("Livrare din rezervorul R"+k+
         " ziua:"+Z[j],0));
   }
// -->
</script>
</head>
```

Figura 4.15

```
<body>
<center>
<h3>SITUATIA LIVRARILOR DE BENZINA PENTRU REZERVOARELE R1 R2 R3
</h3>
<script language="javascript">
   function trunchiaza(x) {
    var s=""+x;
     i=s.indexOf(".");
     if(i!=-1){
       s=s.substring(0,i+3);
       }
     return s;
     }

   // CALCULUL MEDIILOR PE ZILE
   var i,j;
   B = new Array(5);
   for(j=0;j<5;j++) {
      S=0;
      for(i=0;i<3;i++)
         S=S+a[i][j];
      B[j]=S/3;
     }
   // CALCULUL MEDIILOR PE REZERVOARE
   D = new Array(4);
   ST=0;
   for(i=0;i<3;i++) {
      S=0;
      for(j=0;j<5;j++){
           S=S+a[i][j];
          ST=ST+a[i][j];
       }
      D[i]=S/5;
     }
    D[3]=ST/15;

// DETERMINAREA MAXIMULUI SI MINIMULUI
   max=a[0][0];
   min=a[0][0];
   imax=0;imin=0;
    jmax=0;jmin=0;
    for(i=0;i<3;i++){
      for(j=0;j<5;j++){
        if(max<a[i][j]){max=a[i][j];imax=i;jmax=j;}
        if(min>a[i][j]){min=a[i][j];imin=i;jmin=j;}
   }
}
  imin++;imax++;
```

Figura 4.15 *(continuare)*

```
   //AFISARE REZULTATE
  document.writeln("<center><table border=1 bgcolor'#efefff><tr>");
  document.writeln("<td><b>ZIUA</b><td><b>R1</b><td>
<b>R2</b><td><b>R3</b><td><b>MEDIA</b></td></tr>");

    for(k=0;k<5;k++) {
     document.writeln("<tr><td>" + Z[k]+"</td>");
     for(j=0;j<3;j++) {
       document.writeln("<td>" + a[j][k]+ "</td>");
      }
     document.writeln(" <td> " +trunchiaza( B[k])+ "</td></tr>");
    }
     document.writeln("<tr><td><b>MEDIA</b>");
   for(j=0;j<4;j++)
     document.writeln("<td><b>" + trunchiaza(D[j])+" </b></td>");
   document.writeln("</tr>");
   document.writeln("</table></center><p><p>");
   document.writeln("<font size=+1 color=green>Livrarea maxima:"+max+" s-a facut din
rezervorul: R"+imax+" in ziua de "+Z[jmax]+"</font>");
document.writeln("<p><font size=+1 color=green>Livrarea minima:"+min+" s-a facut din
rezervorul: R"+imin+" in ziua de "+Z[jmin]+"</font>");
</script>
</body>
</html>
```

Figura 4.15 *(continuare)*

Rezultatele execuției script-ului sunt prezentate în figura 4.16.

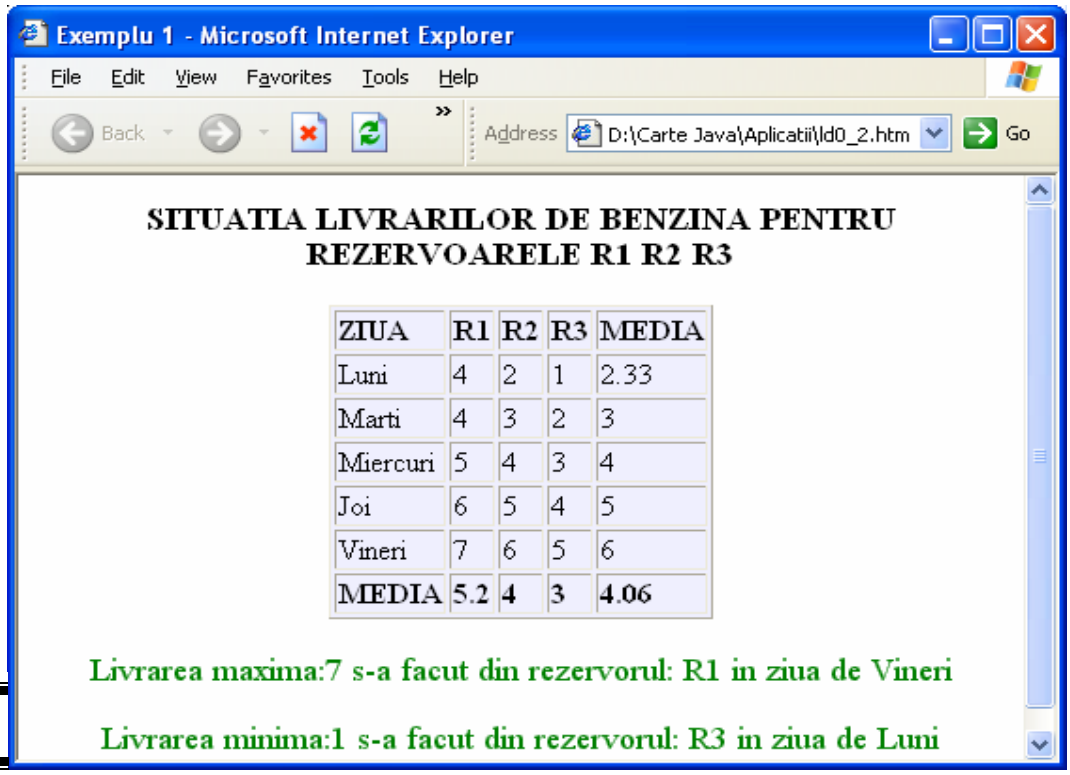


Figura 4.16

Aplicație

❑ Problema care se pune acum seamănă cu cea din EXEMPLELE precedente, dar este ... puțin mai complicată! Se citesc de la tastatură un număr oarecare (maxim 30) de valori ale razelor unor rezervoare cilindrice echilaterale. Pentru fiecare valoare a razei se calculează şi afişează cantitatea de benzină din rezervor. *Script-ul verifică totodată ca valorile razelor rezervoarelor să fie numere pozitive.* De asemenea, script-ul calculează şi afişează masa totală de benzină din rezervoare. În felul acesta, cu acelaşi script se pot rezolva probleme ce diferă prin numărul de rezervoare.

Analiza problemei

Problemele care se pun în această etapă privesc în mod special alcătuirea tabelei de variabile, mai precis identificarea variabilelor de intrare ale script-ului.

Numărul rezervoarelor, necunoscut în problemă, trebuie furnizat ca o variabilă de intrare (n) în momentul execuției script-ului, înaintea citirii datelor propriu-zise (razele rezervoarelor). Cât priveşte numărul variabil de raze, de asemenea necunoscut în momentul scrierii script-ului vom folosi ca variabilă de intrare o matrice(array) cu o singură dimensiune – un vector.

Remarcă. În informatică, se desemnează prin abuz de limbaj sub numele de vector, mulțimea {V(1), V(2), …, V(n)}, unde V(1), V(2), …, V(n) sunt elementele vectorului. Aşadar, un vector este o mulțime de elemente identificate prin poziția pe care acestea o ocupă. Prelucrările ce se efectuează asupra unui vector sunt funcție de valoarea unui indice de poziție (0<=i<=n). Acestuia i se fixează o valoare care, în general, corespunde primei poziții, iar prelucrarea se efectuează cât timp valoarea acestui indice nu depăşeşte o valoare finală.

În figura 4.17, 4.18, 4.19 sunt prezentate: formatul datelor de ieşire, tabela de variabile, specificațiile de programare.

Formatul datelor de ieşire

RAZA	MASA
xx	xxx.xx
xx	xxx.xx
.	.
xx	xxx.xx
MASA TOTALA=	xxx.xx TONE

Figura 4.17

Tabela de variabile

Variabile de intrare	*Variabile de stare*	*Variabile de iesire*
n: numărul de rezervoare r: vectorul razelor rezervoarelor	d: densitatea benzinei i: indicele de poziție (indexul) v: volumul rezervorului s: masa totală de benzină m: masa de benzină din rezervor	m: cantitatea de benzină din rezervor s: cantitatea (masa) totală de benzină

Figura 4.18

Specificații de programare

Descriere. Script-ul calculează şi afişează cantitatea de benzină (s) dintr-un număr oarecare (n) de rezervoare cilindrice echilaterale (acest număr este furnizat ca parametru). Script-ul citeşte valorile razelor rezervoarelor într-un vector de date (r) printr-o procedură de introducere dinamică a datelor (razele cu valori negative nu se iau în considerare!). Pentru raze cu valori negative, script-ul afişează mesajul: „*Raza negativă*". Valoarea densității benzinei (d) se introduce în mod static.

Intrări. Se introduc de la tastatură numărul rezervoarelor cilindrice echilaterale şi valorile razelor acestora.

Ieşiri. Masa de benzină (m) din fiecare rezervor şi masa totală (s) de benzină.

Lista de funcțiuni ale script-ului

1. Citeşte număr rezervoare (n).
2. Inițializează variabila (s).
3. Atribuie variabilei d valoarea 0.7 (densitatea benzinei).
4. Afişează un rând de 30 de "**=**".
5. Afişează "raza, masa".
6. Pentru fiecare rezervor (cilindric echilateral):
   - **6.1** Citeşte valoarea razei unui rezervor.
   - **6.2** Validează datele introduse după cum urmează: pentru fiecare articol se verifică dacă valoarea razei citite este negativă. Script-ul afişează mesajul: "*Raza negativă"*.
7. Pentru fiecare rezervor (cilindric echilateral):
   - **7.1** Calculează volumul rezervorului.
   - **7.2** Calculează masa de benzină din rezervor.
   - **7.3** Însumează m în s.
   - **7.4** Afişează r, m
8. Afişează masa totală de benzină
9. Stop

Figura 4.19

Proiectarea programului

În figura 4.20 se prezintă pseudocodul, varianta formalizată.

Pseudocodul

```
REZERVOARE   BEGIN
                    d=0.7
                    s=0
                    //Citeste numar rezervoare(n)
                    READ(n)
CAP-TABEL           BEGIN
                          Do LINIUTZA
                          WRITE('raza'+'   '+'masa');
                          DO LINIUTZA
CAP-TABEL           END
```

Figura 4.20

```
PREL-VALID          FOR(i=1;i<=n;i++)
                            READ(r[i])
                            WRITE(r[i]);
VALID                       WHILE (r[i]<0)
                                    WRITE('Raza negativa')
                                    READ(r[i])
                                    WRITE(r[i])
VALID                       ENDWHILE
PREL-VALID          ENDFOR
CALC-MASA           FOR(i=1;i<=n;i++)
                            m=2∏dr[i]3
                            s=s+m
                            WRITE(r[i]+m)
CALC-MASA           ENDFOR
                    WRITE(s)
REZERVOARE  END
LINIUTZA    BEGIN
                    Afiseaza un rand de 30 de "="
LINIUTZA    END
```

Figura 4.20 *(continuare)*

Codificarea în limbajul JavaScript

În figura 4.21 este prezentat documentul (X)HTML complet.

c_5-20 - Notepad

File Edit Format View Help

```
<html>
 <head>
   <title>n rezervoare cilindrice echilaterale</title>
 </head>
 <body>
 <script language="javascript" type="text/javascript">
  function rand(){
    document.write("<br>");
    for(i=1;i<=13;i++)
      document.write("=");
    document.write("<br>");
   }
   d=0.7;
   s=0;
   //citeste numarul de rezervoare
   n=prompt('Numarul de rezervoare=',0);
   rand();
   document.write('RAZA'+'---------'+'MASA');
   rand();
   r=new Array();
   for(i=1;i<=n;i++){
     document.write('<br>Rezervorul numarul '+i);
     r[i]=parseFloat(prompt('Raza rezervorului',0));
     document.write(' Ati specificat :'+r[i]);
     while(r[i]<0){
       alert('raza negativa');
       r[i]=parseFloat(prompt('Raza rezervorului',0));
       document.write('Ati specificat :'+r[i]);
      }
     v=2*Math.PI*(Math.pow(r[i],3));
     m=d*v;
     s=s+m;
     document.write('<br>'+r[i]+'---------'+m);
    }
   rand();
   document.write('Masa totala '+s+' tone');
   rand();
 </script>
</body>
</html>
```

Figura 4.21

Rezultatele execuţiei script-ului sunt prezentate în figurile 4.22, 4.23.

Figura 4.22

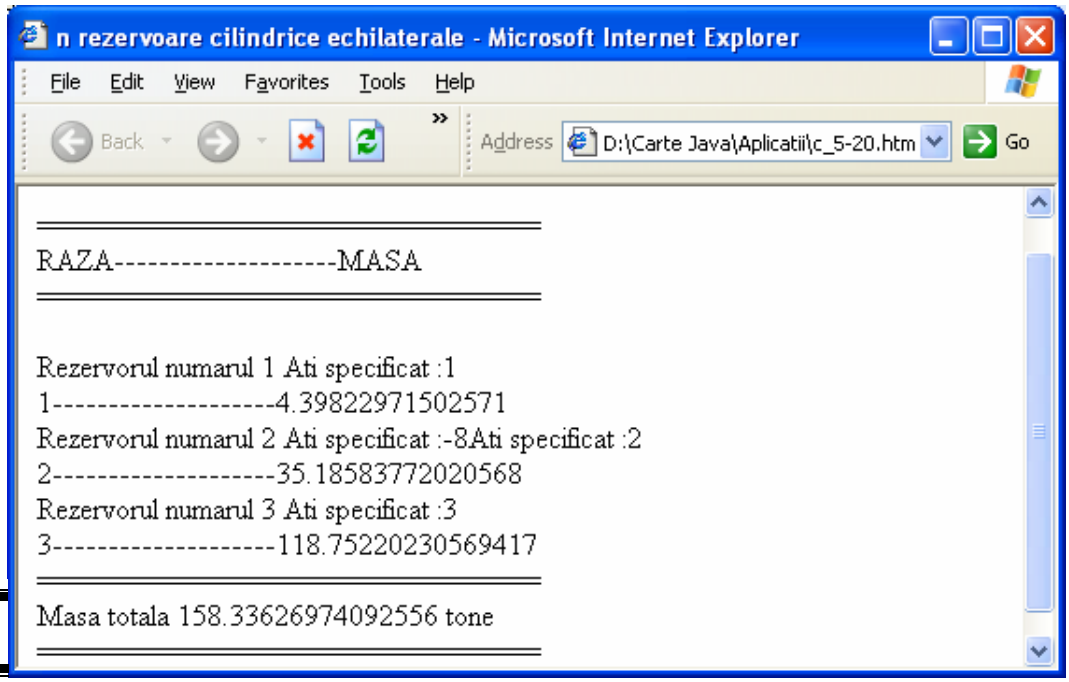
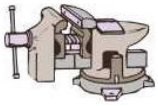


Figura 4.23

Testați-vă cunoștințele



1. Care sunt obiectele interne (integrate) ale limbajului JavaScript?
2. Cum se creează obiectul `String`?
3. Care sunt proprietăţile obiectului `String`?
4. Care sunt metodele pentru formatarea şirurilor de caractere?
5. Ce realizează următoarele metode ale obiectului `String`:
   - `indexOf();`
   - `substr();`
   - `toLowerCase().`
6. Cum se creează obiectul `Array`?
7. Ce realizează următoarele metode ale obiectului `Array`?
   - `concat();`
   - `toString;`
   - `sort().`
8. Se consideră şirul de caractere:
   sir=”Protopopescu Augustin”;
   Precizaţi ce valori returnează următoarele metode ale obiectului `String`:
   - alert(sir.charAt(3)); //afişează ...?...
   - alert(sir.charCodeAt(3)); //afişează ...?...
   - alert(sir.indexOf(”0”)); //afişează ...?...
   - alert(sir.indexOf(”0”,8)); //afişează ...?...
   - alert(sir.indexOf(”z”,8)); //afişează ...?...
9. Se consideră şirul de caractere:
   sir=”abcdef”;
   Precizaţi ce valori returnează următoarele metode ale obiectului `String`:
   - alert(sir.slice(0,3)); //afişează ...?...
   - alert(sir.slice(1,3)); //afişează ...?...
   - alert(sir.slice(3)); //afişează ...?...
   - alert(sir.slice(2,-1)); //afişează ...?...
   - alert(sir.slice(2,-2)); //afişează ...?...

- alert(sir.slice(-2,-1)); //afişează ...?...

10. Se consideră şirul de caractere:

    sir="a,b,c,d,e,f";

    Precizaţi ce valori returnează următoarele metode ale obiectului `String`:

    - alert(sir.split(",")); //afişează ...?...
    - alert(sir.substr(2,2)); //afişează ...?...
    - alert(sir.substr(3)); //afişează ...?...
    - alert(sir.substr(-3,2)); //afişează ...?...
    - alert(sir.substr(0,2)); //afişează ...?...

11. Se consideră matricea:

    g=[1,2,3];

    Precizaţi ce valori returnează următoarele metode ale obiectului `Array`:

    - alert(g.concat(4,5)); //afişează ...?...
    - alert(g.concat([4,5])); //afişează ...?...
    - alert(g.concat([4,5],[6,7])); //afişează ...?...
    - alert(g.concat(4,[5,[6,7]])); //afişează ...?...

12. Se consideră matricea:

    g=[1,2,3,4,5];

    Precizaţi ce valori returnează următoarele metode ale obiectului `Array`:

    - alert(g.slice(0,3)); //afişează ...?...
    - alert(g.slice(3)); //afişează ...?...
    - alert(g.slice(1,-1); //afişează ...?...
    - alert(g.slice(-3,-2)); //afişează ...?...

13. Se consideră funcţia:

```
function sortare(x,y){
  return x-y;
}
```

    şi

```
var g=new array(7,5,100,21);
```

    Precizaţi rezultatele obţinute în urma execuţiei următoarelor metode:

    - alert(g.sort()); //afişează ...?...
    - alert(g.sort(sortare)); //afişează ...?...

14. Care este efectul metodei `join()`? Care este relaţia sa cu metoda `split()` a obiectului `String`?

15. Identificaţi erorile de sintaxă din următoarele script-uri:

    - Figura 11.3 (Conversaţia 11, pagina 5);
    - Figura 11.5 (Conversaţia 11, pagina 5).

Vizitați site-urile

- ✓ www.bdml.net/listes
- ✓ www.natural.com/JavaScript
- ✓ http://www.webcoder.com
- ✓ http://www,dannyg.com/examples/ol2/index.htm

Conversația 5

Obiectele (interne) RegExp, Date, Arguments

În această conversație:

Expresii regulate

Expresiile regulate (*Regular Expressions*, în limba engleză) sunt utilizate ca modele (tipare) de căutare pentru regăsirea, ştergerea şi înlocuirea caracterelor.

Expresiile regulate nu sunt primordiale în programarea JavaScript, dar ele sunt universale (foarte multe limbaje de programare le utilizează) şi pot să vă simplifice ... viaţa odată ce ele au fort asimilate.

Remarci:

- ✓ Implementarea expresiilor regulate în limbajul JavaScript a fost preluată direct din limbajul Perl.
- ✓ Expresiile regulate sunt puţin mai ... greu de digerat, dar ele reprezintă o funcţie puternică a limbajului, începând cu JavaScript 1.4. Versiunile ulterioare oferă numeroase îmbunătăţiri.
- ✓ Expresiile regulate pot fi utilizate în `e-commerce` (comerţul electronic) pentru validarea numerelor cărţilor de credit, a numerelor de telefon, a adreselor etc.
- ✓ O expresie regulată este un obiect JavaScript.
- ✓ Utilizaţi (la maximum) expresiile regulate în formularele dumneavoastră deoarece ele vă garantează date fiabile (vezi Conversaţia 8).
- ✓ Crearea expresiilor regulate este foarte asemănătoare cu crearea şirurilor (*strings*, în limba engleză).

Obiectul RegExp

Obiectul `RegExp()` este un obiect predefinit (built-in), care nu depinde de navigator.

Fişa obiectului `RegExp()` este prezentată în figura 5.1.

	Fişa obiectului RegExp
Cum se creează obiectul?	`constructorul RegExp()`
Proprietăţi:	`$'`, `$&`, `$_`, `$\`, `$+`, `$n`, `global`, `ignoreCase`, `index`, `input`, `lastIndex`, `lastMatch`, `lastParen`, `leftContext`, `multiline`, `rightContext`, `source`
Metode:	`compile()`, `exec()`, `test()`
Gestionarii de evenimente:	–

Figura 5.1

Constructorul RegExp()

Constructorul `RegExp()` este prezentat în detaliu în figura 5.2.

Constructor	***Sintaxă***
`RegExp()`	`Variabila=new RegExp("Model","Atribut")` `Variabila=/Model/Atribut`

Variabila va conţine modelul expresiei regulate transmis ca parametru ("*Model*"). "*Atribut*" (cel de-al doilea parametru) stochează sub forma unui şir de caractere atributele de identificare ale modelelor (tiparelor). Cele trei atribute ale obiectului `RegExp` sunt următoarele:

- `g` – căutare globală; găseşte toate apariţiile posibile.
- `i` – în cursul căutării nu face deosebire între majuscule şi minuscule.
- `m` – căutare multilinie (vezi zona textarea din cadrul unui formular).
- `gi` – cumulează opţiunile `g` şi `i`.

Expresiile regulate pot fi create şi în mod implicit. În acest caz, se atribuie modelul (tiparul) – încadrat între mai multe bare oblice, direct unei variabile.

Cele două metode de definire a unei expresii regulate sunt riguros echivalente. Slash-urile (/) nu sunt necesare în constructorul `RegExp()`, deoarece se poate face distincţie între acesta şi constructorul `String()`.

Figura 5.2

Caracterele de repetiţie utilizate în identificarea modelelor (tiparelor) sunt prezentate în figura 5.3.

Caracter	*Descriere*
`x{m,n}`	Caracterul precedent (x) este identificat de cel puţin *m* ori, dar nu mai mult de *n* ori (m şi n reprezintă numere).
`x{m, }`	Caracterul precedent (x) este identificat de cel puţin *m* ori.
`x{m}`	Caracterul precedent este identificat de *m* ori.
`x?`	Caracterul precedent (x) este identificat o dată sau nici o dată.
`x+`	Caracterul precedent (x) este identificat cel puţin o dată.
`x*`	Căutare multilinie (vezi zona `textarea` din cadrul unui formular).

Figura 5.3

Clasele de caractere utilizate în identificarea modelelor (tiparelor) sunt prezentate în figura 5.4:

Clasa de caractere	*Descriere*
`[...]`	Orice caracter inclus între paranteze.
.	Orice caracter, cu excepţia caracterului (`\n`) – salt la linie nouă.
`[^…]`	Orice caracter care nu apare între paranteze.
`[m-n]`	Orice caracter din intervalul m la n care apare în şirul de căutare.
`[^m-n]`	Orice caracter care nu se află în intervalul de la m la n.
`\w`	w mic (`\w`) identifică orice caracter (identic cu [a-z A-Z 0-9 ...])
`\W`	W mare (`\W`) este opusul lui (`\w`) (identic cu [^a-z A-Z 0-9 ...]).
`\s`	s mic (`\s`) identifică orice caracter de spaţiu (identic cu [\t\n\r\v]).
`\S`	S mare (`\S`) este opusul lui (`\s`) (identic cu [^\t\n\r\v]).
`\d`	d mic (`\d`) identifică orice cifră între 0 şi 9 (identic cu [0-9]).
`\D`	D mare (`\D`) este opusul lui d mic (`\d`) (identic cu [^0-9])
`[\b]`	Identificare caracter backspace (`\b` este plasat între paranteze drepte).

Figura 5.4

Remarcă. Pentru a defini mai multe modele de căutare în aceeaşi expresie regulată, utilizaţi parantezele rotunde şi operatorul SAU logic ("|").

Indicatorii de poziţie pe care îi puteţi folosi în identificarea modelelor sunt prezentaţi în figura 5.5.

Indicator de poziţie	*Poziţia în şirul de caractere*
^	Începutul şirului de caractere.
$	Sfârşitul şirului de caractere.
`\b`	Identifică o poziţie dintre un caracter de text şi un caracter care nu este de text.
`\B`	Opusul lui `\b`.

Figura 5.5

Aplicaţie

❑ În Tabelul 5.1 sunt prezentate mai multe exemple de utilizare a expresiilor regulate. Comentaţi rezultatele căutării lor.

Tabelul 5.1

Atribut	*Găseşte*	*Nu găseşte*
/12+3/	123,122223	12
/ab?c/	abc,ac	abbc
/bon{1,2}us/	bonus,bonnus	bonnnus
/1*2{5}3/	1222223	123,122222

Proprietăţile obiectului RegExp

Proprietăţile obiectului `RegExp` sunt prezentate în detaliu (în ordine alfabetică) în figura 5.6.

	Proprietate	*Sintaxă*
	`$\'`	`RegExp["$\'"]`
	Returnează textul situat la dreapta ultimului text găsit. Echivalent cu `RightContext`.	
Exemplu:	<script language="javascript" type="text/javascript"> var model=/ion/i; model.test("Ion Ionescu"); document.write(RegExp["$\"]); //afiseaza Ionescu </script>	
	`$&`	`RegExp["$&"]`
	Returnează ultimul text găsit. Echivalent cu `lastMatch`.	
Exemplu:	<script language="javascript" type="text/javascript"> var model=/ion/i; model.test("Ion Ionescu"); document.write(RegExp["$&"]); //afiseaza Ion </script>	
	`$_`	`RegExp["$_"]`
	Returnează şirul de caractere în care a fost efectuată căutarea. Echivalent cu `input`.	
Exemplu:	<script language="javascript" type="text/javascript"> var model=/ion/i; model.test("Ion Ionescu"); document.write(RegExp["$_"]); //afiseaza Ion </script>	

Figura 5.6

	`$'` `RegExp["$'"]`
	Returnează textul situat la stânga ultimului text găsit. Echivalent cu `LeftContext`.
Exemplu:	<script language="javascript" type="text/javascript"> var model=/ion/i; model.test("Nume: Ion Ionescu"); document.write(RegExp["$'"]); //afiseaza Nume: </script>
	`$+` `RegExp["$+"]`
	Returnează textul găsit prin ultima sub-expresie a modelului utilizat. Echivalent cu `lastParen`.
Exemplu:	<script language="javascript" type="text/javascript"> var model=/(ion)|(ionescu)/ig; model.test("Nume: Ion Ionescu"); document.write(RegExp["$+"]); //afiseaza Ionescu </script>
	`$n` `RegExp.$n`
	$1, $2, ..., $9 returnează primele 9 sub-şiruri ale modelului.
	`global` `RegExp.global`
	Returnează *true* dacă atributul *g* (căutare globală) a fost definit; în caz contrar returnează *false*.
Exemplu:	<script language="javascript" type="text/javascript"> model=/cartof/g; model.test("Un cartof sau doi cartofi?"); document.write(model.global); //afiseaza true </script>
	`ignoreCase` `RegExp.ignoreCase`
	Returnează *true* dacă atributul *i* (în cursul căutării nu se face deosebire între majuscule şi minuscule) a fost definit; în caz contrar returnează *false.*
Exemplu:	<script language="javascript" type="text/javascript"> model=/ion/i; model.test("Ion Ionescu"); document.write(model.ignoreCase); //afiseaza true </script>
	`index` `RegExp.index`
	Returnează poziţia primului caracter al textului găsit.
Exemplu:	<script language="javascript" type="text/javascript"> model=/ion/i; model.test("Nume: Ion Ionescu"); document.write(RegExp.index); //afiseaza 6 </script>

Figura 5.6
(continuare)

	`input` `RegExp.input`
	Returnează şirul de caractere în care a fost efectuată căutarea.
Exemplu:	<script language="javascript" type="text/javascript"> model=/ion/i; model.test("Nume: Ion Ionescu"); document.write(RegExp.input); //afiseaza Nume: Ion Ionescu </script>
	`lastIndex` `RegExp.lastIndex`
	Returnează poziţia, în şirul de caractere de la care va începe viitoarea căutare. Primul caracter se află pe poziţia 0. Returnează 0 dacă precedenta căutare a eşuat. Pentru că viitoarea căutare începe cu primul caracter, atribuiţi valoarea 0 acestei proprietăţi.
Exemplu:	<script language="javascript" type="text/javascript"> model=/ion/i; model.test("Ion Ionescu"); document.write(RegExp.lastIndex); //afiseaza 3 </script>
	`lastMatch` `RegExp.lastMatch`
	Returnează ultimul text găsit.
Exemplu:	<script language="javascript" type="text/javascript"> model=/ion/i; model.test("Nume: Ion Ionescu"); document.write(RegExp.lastMatch); //afiseaza Ion </script>
	`lastParen` `RegExp.lastParen`
	Returnează textul găsit prin ultima sub-expresie a modelului utilizat.
Exemplu:	<script language="javascript" type="text/javascript"> model=/(ion)|(ionescu)/ig; model.test("Nume: Ion Ionescu"); document.write(RegExp.lastParen); //afiseaza Ionescu </script>
	`leftContext` `RegExp.leftContext`
	Returnează textul situat la stânga ultimului text găsit.
Exemplu:	<script language="javascript" type="text/javascript"> model=/ion/i; model.test("Nume: Ion Ionescu"); document.write(RegExp.leftContext); //afiseaza Nume: </script>
	`multiline` `regExp.multiline`
	Returnează *true* dacă atributul *m* (multilinie) a fost definit; în caz contrar returnează *false.*

Figura 5.6
(continuare)

Exemplu:
```
<script language="javascript" type="text/javascript">
  model=/ion/m;
  model.test("Ion Ionescu");
  document.write(model.multiline);
  //afiseaza true
 </script>
```

`rightContext`	`RegExp.rightContext`

Returnează textul situat la dreapta ultimului text găsit.

Exemplu:
```
<script language="javascript" type="text/javascript">
  model=/ion/i;
  model.test("Ion Ionescu");
  document.write(RegExp.rightContext);
  //afiseaza Ionescu
 </script>
```

`source`	`regExp.source`

Returnează expresia regulată.

Exemplu:
```
<script language="javascript" type="text/javascript">
  model=/ion/i;
  model.test("Ion Ionescu");
  document.write(model.source);
  //afiseaza ion
</script>
```

Figura 5.6 *(continuare)*

Metodele obiectului RegExp

Metodele obiectului `RegExp`, prezentate în detaliu în figura 5.7 servesc pentru identificarea modelelor (tiparelor) în obiectul `RegExp`.

Metodă	***Sintaxă***
`compile()`	`model.compile(`*`model`*`)`

Înlocuieşte un model vechi cu unul nou.

Exemplu:
```
<script language="javascript" type="text/javascript">
  var model=/ftp.*/i;
  model.compile("http.*","i");
  document.write(model.source);
 </script>
```

`exec()`	`model.exec(`*"text"*`)`

Caută un model în "text" şi returnează rezultatul (o matrice `Array`).

Exemplu:
```
<script language="javascript" type="text/javascript">
  var model=/ion/i;
  rezultat=model.exec("Nume: Ion Ionescu");
  document.write(rezultat+"<br />");
  //afiseaza Ion
  document.write(rezultat[0]+"<br />"); //afiseaza Ion
 </script>
```

Figura 5.7

	`test()` `model.test("text")`
	Caută un model în "*text*" şi returnează `true/false`.
Exemplu:	<script language="javascript" type="text/javascript"> var model=/ion/i; rezultat=model.test("Nume: Ion Ionescu"); document.write(rezultat); //se afiseaza true </script>

Figura 5.7 *(continuare)*

Remarci:

- ✓ Metodele: `compile()`, `exec()` şi `test()` de tip `RegExp` impun ca obiectul `String` în care se face căutarea să fie transmis ca argument.
- ✓ Metoda `exec()` modifică de asemenea mai multe proprietăţi ale matricei returnate, ale modelului şi ale obiectului `RegExp`:
  - Matricea (`Array`) `index`, `input`;
  - Modelul `lastIndex`, `source`, `global`, `ignoreCase`;
  - `RegExp` – `lastMatch`, `leftContext`, `rightContext`, `multiline`.

EXEMPLUL 5 JAVASCRIPT

❑ Formularea problemei

Se reia problema rezervoarelor din exemplele precedente. În acest exemplu (EXEMPLUL 5 JAVASCRIPT) se doreşte ca situaţia livrărilor de benzină să fie expediată prin e-mail, la o adresă specificată de către utilizator. EXEMPLUL 5 JAVASCRIPT verifică dacă adresa e-mail este formată din caractere valide. În acest sens se va crea un model RegExp(`re`) care să includă următoarele reguli:

- ✓ adresa trebuie să înceapă cu cel puţin un caracter (plasat la stânga simbolului @);
- ✓ adresa trebuie să conţină simbolul @;
- ✓ adresa trebuie să conţină cel puţin un caracter după simbolul @;
- ✓ adresa poate conţine caracterele: „.” sau „-” dar nu unul după celălalt;
- ✓ adresa trebuie să se termine cu punct („.”) urmat de cel puţin două caractere.

În caz de eroare, într-o casetă de dialog se va afişa mesajul de avertizare: „*Adresa E-mail: xxxxxxxx este incorectă*”. Expedierea prin e-mail a situaţiei livrărilor de benzină din cele trei rezervoare se va efectua prin acţionarea unui buton „*Trimite email*”.

❑ Analiza problemei

În figura 5.8 este prezentat ecranul cu „Situaţia livrărilor de benzină pentru rezervoarele R1, R2, R3”, care conţine de asemenea şi zona de text pentru adresa e-mail (*Adresa*) şi

butonul „*Trimite email*”. În situația în care adresa de e-mail introdusă în zona *Adresa* nu este validă se afişează un mesaj de eroare, într-o casetă de dialog.

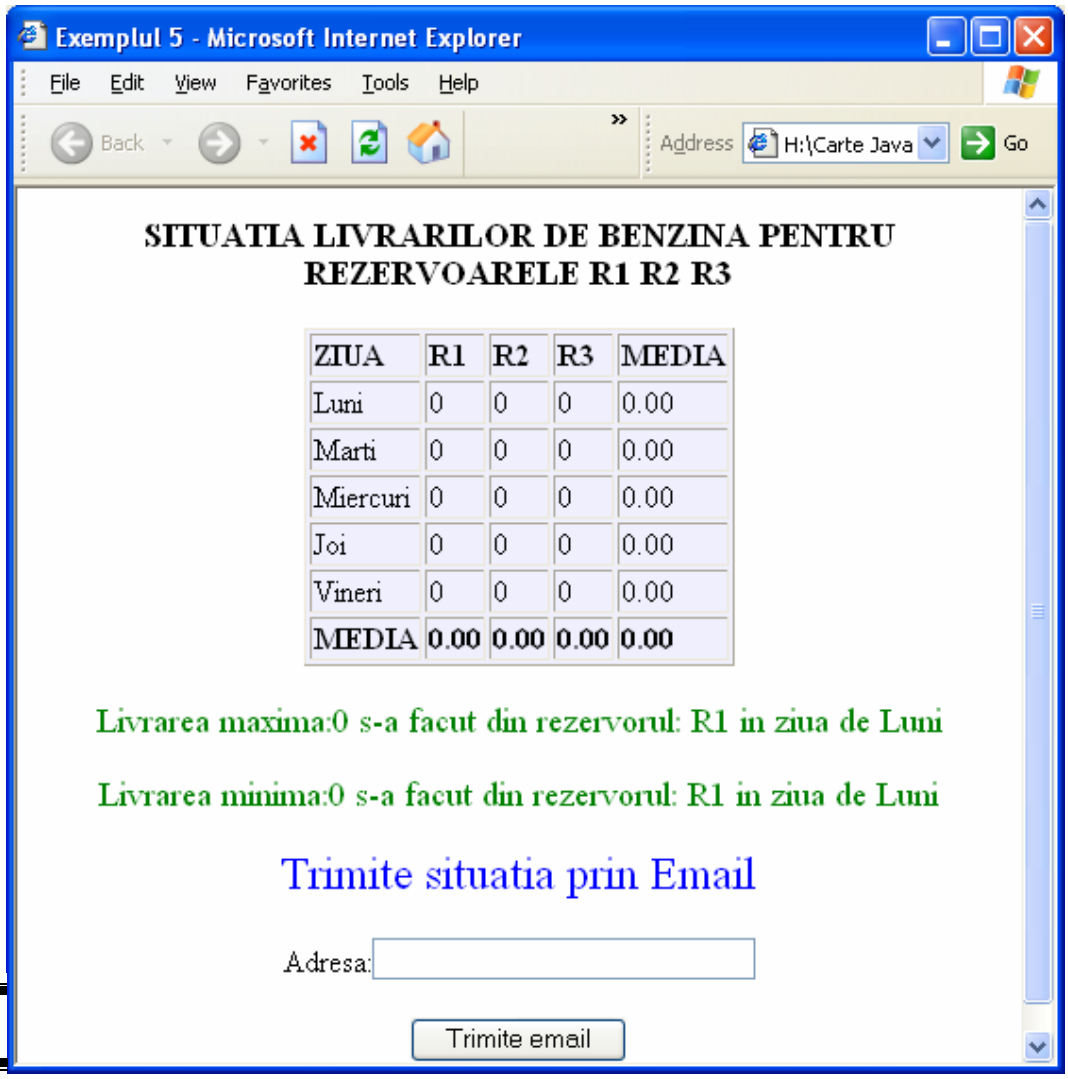


Figura 5.8

Tabela de variabile şi specificațiile de programare sunt prezentate în figura 5.9, respectiv figura 5.10.

Tabela de variabile

Variabile de intrare	***Variabile de stare***	***Variabile de iesire***
adresa: obiect, zonă de editare text pentru introducerea adresei de email	st: suma livrărilor totale pe zile şi rezervoare s: sume parțiale pe zile rval: starea zonei de editare (validă sau nu) x: valoarea reală a textului introdus în zona de editare z: numele zilelor săptămânii imax, imin, jmax, jmin: indicii livrărilor maxime şi minime din matricea a	a: matricea livrărilor pe zile şi rezervoare b: vectorul mediilor livrărilor pe zile d: vectorul mediilor livrărilor pe rezervor max, min: valoarea maximă şi minimă a livrărilor

Figura 5.9

Specificații de programare

Descriere. Programul expediază prin e-mail „Situația livrărilor de benzină pentru rezervoarele R1, R2, R3” la o adresă specificată de utilizator.

Intrări. Adresa e-mail.

Ieşiri. Raportul cu situația livrărilor expediate prin e-mail.

Lista de funcțiuni ale script-ului

1. Afişează „Trimite situația prin Email”.
2. Afişează forma de introducere a adresei de email.
3. Răspunde la evenimentele generate de butonul „Trimite email”.
4. Răspunde la evenimentele generate de zona de editare.
5. Validează valoarea introdusă în zona de text (adresa).
6. Afişează mesajul de eroare „Adresa E-mail xxx este incorectă”.
7. Formează corpul mesajului ce urmează a fi trimis prin email.
8. Stop

Figura 5.10

❑ Proiectarea script-ului

Pseudocodul pentru EXEMPLUL 5 JAVASCRIPT este prezentat în figura 5.11.

Pseudocodul

```
Exemplul5   BEGIN
            //initializeaza vectorul Z cu numele zilelor
            z=luni,marti,miercuri,joi vineri
            //aloca spatiu de memorie si
            //citeste livrarile pe fiecare rezervor de la tastatura
LIV         FOR(i=0;i<3;i++)
LIV1                FOR(j=0;j<5;j++)
                        READ a[i,j]
LIV1                ENDFOR
LIV         ENDFOR
            // aloca spatiu de memorie pentru vectorul b si d
            // calculeaza mediile pe rezervoare
FORJ        FOR(j=0;j<5;j++)
                    s=0
FORI                FOR(i=0;i<3;i++)
                        s=s+a[i,j]
FORI                ENDFOR
                    b[j]=s/3
FORJ        ENDFOR
             // calculul mediilor pe rezervoare
            st=0
FORMEDII    FOR(i=0;i<3;i++)
                    s=0
FORMEDIIJ           FOR(j=0;j<5;j++)
                        s=s+a[i,j]
                        st=st+a[ij]
FORMEDIIJ           ENDFOR
                    d[i]=s/5
```

Figura 5.11

```
FORMEDII    ENDFOR
            d3=s/15
            // determinarea maximului si minimului
            max=a0,0
            min=a0,0
            imax=0
            imin=0
            jmax=0
            jmin=0
FORMAXI     FOR(i=0;i<3;i++)
FORMAXJ             FOR(j=0;j<5;j++)
IFMAX                       IF(max<ai,j)
                                    max=aij
                                    imax=i
                                    jmax=j
IFMAX                       ENDIF
IFMIN                       IF(min>ai,j)
                                    min=aij
                                    imin=i
                                    jmin=j
IFMIN                       ENDIF
FORMAXJ             ENDFOR
FORMAXI     ENDFOR
            imin=imin+1
            imax=imax+1
            //formeaza corpul mesajului
            //ce urmeaza a fi trimis prin email
            DO formeaza_afis
            //afisare rezultate
            WRITE "SITUATIA REZERVOARELOR R1 R2 R3"
            WRITE "ZIUA  R1  R2  R3  MEDIA"
FORK        FOR(k=0;k<5;k++)
                    WRITE Z[k]
FORJA               FOR(j=0;j<3;j++)
                            WRITE  a[j][k]
FORJA               ENDFOR
                    WRITE  trunchiaza( b[k])
FORK        ENDFOR
            WRITE  "MEDIA"
FORTRUNCJ   FOR(j=0;j<4;j++)
                    WRITE trunchiaza(d[j])
FORTRUNCJ   ENDFOR
            WRITE "Livrarea maxima: " max
            WRITE "s-a facut din rezervorul: R" imax
            WRITE "in ziua de" Zjmax
            WRITE "Livrarea minima:" min
            WRITE " s-a facut din rezervorul: R"imin
            WRITE " in ziua de "+Zjmin
            WRITE " Trimite situatie prin e-mail"
            //afiseaza forma de introducere a adresei de email
            WRITE "<Form> ... "
            //raspunde la evenimentele generate de butonul Trimite Email
IFMAIL      IF(se apasa butonul Trimite Email)
                    DO trimite_email()
IFMAIL      ENDIF
           //raspunde la evenimentele generate de zona de editare
```

Figura 5.11 *(continuare)*

```
IFEDIT      IF(se paraseste zona de editare adresa)
                    DO verificaAdresa()
IFEDIT      ENDIF

Exemplul5   END
// transforma o valoare reala in sir de caractere
// si trunchiaza la doua zecimale
TRUNCHIAZA BEGIN
            Parametrii:
                    x- valoare reala
            s=transforma in sir de caractere x
            i=pozitia punctului zecimal in sir
IFCOPY      IF(i≠-1)
                    s=copiazasubsir(s,0,i+2)
IFCOPY      ENDIF
            RETURN s
TRUNCHIAZA END
VERIFICAADRESA    BEGIN
                  //valideaza valoarea introdusa in zona de text adresa
                  rval=fals
                  Initializeaza REGEXP re cu
                    modelul: /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,})+$/
                  s=adresa.value
                  rval=re.test
IFVALID           IF(!rval)
                          WRITE mesaj de eroare: "Adresa gresita"
IFVALID           ENDIF
                  RETURN rval
VERIFICAADRESA    END
//formeaza corpul mesajului de email
FORMEAZA_AFIS     BEGIN
                          sbody= "SITUATIA REZERVOARELOR R1 R2 R3"
                          sbody+= "ZIUA  R1  R2  R3  MEDIA"
FORKMESAJ                 FOR(k=0;k<5;k++)
                                  sbody+= Z[k]
FORJMESAJ                         FOR(j=0;j<3;j++)
                                          sbody+=  a[j][k]
FORJMESAJ                         ENDFOR
                                  sbody+= trunchiaza( b[k])
FORKMESAJ                 ENDFOR
                          sbody+= "MEDIA"
FORTRUNC                  FOR(j=0;j<4;j++)
                                  sbody+= trunchiaza(d[j])
FORTRUNC                  ENDFOR
                          sbody+= "Livrarea maxima: " max
                          sbody+= "s-a facut din rezervorul: R" imax
                          sbody+= "in ziua de" Zjmax
                          sbody+= "Livrarea minima:" min
                          sbody+= " s-a facut din rezervorul: R"imin
                          sbody+= " in ziua de "+Zjmin
FORMEAZA_AFIS     END
```

Figura 5.11 *(continuare)*

❑ **Codificarea în limbajul JavaScript**

În figura 5.12 este prezentat documentul (X)HTML complet.

```
<html>
<head>
<title>Exemplul 5</title>
<script language="javascript">
<!---
var Z = new Array("  Luni  "," Marti  ","Miercuri","   Joi  "," Vineri ");
var sbody;
a=new Array(3);
a[0]=new Array(5);
a[1]=new Array(5);
a[2]=new Array(5);
for(i=0;i<3;i++)
 for(j=0;j<5;j++) {
   k=i+1;
   a[i][j]=parseFloat(prompt("Livrare din rezervorul R"+k+" ziua:"+Z[j],0));
   }
  B = new Array(5);
  D = new Array(4);
function trimite_email() {
  var sadr=document.f2.adresa.value;
  document.location.href="mailto:"+sadr+"?subject= teste"+"&body="+sbody;
   }
 function verificaAdresa() {
       var rval=false;  var re;
        re = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,})+$/;
      var s=document.f2.adresa.value;
       rval=re.test(s);
        if ( !rval) {
            alert('Adresa E-mail:'+s+' este incorecta');
            document.f2.adresa.focus(); document.f2.adresa.select();
        }
    return rval;
    }
 function trunchiaza(x) {
     var s=""+x;
      i=s.indexOf(".");
      if(i!=-1){
        s=s.substring(0,i+3);
        }
   else  {
     s=s+".00";
   }
  return s;
}
function formeaza_afis(){
sbody="|  ZIUA  |  R1  |   R2  |  R3   | MEDIA |"+"&lt;br&gt;";
    for(k=0;k<5;k++) {
     sbody+="|" + Z[k];
     for(j=0;j<3;j++) {
       sbody+="|" + trunchiaza(a[j][k]);
      }
```

Figura 5.12

```
    sbody+="|" +trunchiaza( B[k])+ "|%13";
}
```

```
    sbody+="|  MEDIA  ";
  for(j=0;j<4;j++)
    sbody+="|" + trunchiaza(D[j]);
  sbody+="| \r\n \r\n";
  sbody+="Livrarea maxima:"+max+" s-a facut din rezervorul: R"+imax+" in ziua de "+Z[jmax]+"\r\n";
sbody+=" Livrarea minima:"+min+" s-a facut din rezervorul: R"+imin+" in ziua de "+Z[jmin]+"\r\n";
}
// -->
</script>
</head>
<body>
<center>
<h3>SITUATIA LIVRARILOR DE BENZINA PENTRU REZERVOARELE R1 R2 R3</h3>
<script language="javascript">
   // CALCULUL MEDIILOR PE ZILE
   var i,j;
   for(j=0;j<5;j++){
      S=0;
      for(i=0;i<3;i++)
         S=S+a[i][j];
      B[j]=S/3;
    }
   // CALCULUL MEDIILOR PE REZERVOARE
   ST=0;
   for(i=0;i<3;i++){
      S=0;
      for(j=0;j<5;j++){
          S=S+a[i][j];
         ST=ST+a[i][j];
       }
      D[i]=S/5;
    }
    D[3]=ST/15;
 // DETERMINAREA MAXIMULUI SI MINIMULUI
   max=a[0][0];
   min=a[0][0];
   imax=0;imin=0;
    jmax=0;jmin=0;
    for(i=0;i<3;i++){
      for(j=0;j<5;j++){
        if(max<a[i][j]){max=a[i][j];imax=i;jmax=j;}
        if(min>a[i][j]){min=a[i][j];imin=i;jmin=j;}
  }}
 imin++;imax++;
```

```
  //AFISARE REZULTATE
formeaza_afis();
 document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
 document.writeln("<td><b>ZIUA</b><td><b>R1</b><td>
<b>R2</b><td><b>R3</b><td><b>MEDIA</b></td></tr>");
 for(k=0;k<5;k++){
   document.writeln("<tr><td>" + Z[k]+"</td>");
```

Figura 5.12 *(continuare)*

```
    for(j=0;j<3;j++){
     document.writeln("<td>" + a[j][k]+ "</td>");
     }
    document.writeln(" <td> " +trunchiaza( B[k])+ "</TD></Tr>");
   }
   document.writeln("<tr><td><b>MEDIA</b>");
  for(j=0;j<4;j++)
  document.writeln("<td><b>" + trunchiaza(D[j])+" </b></td>");
  document.writeln("</tr>");
  document.writeln("</table></center><p><p>");
  document.writeln("<font size=+1 color=green>Livrarea maxima:"+max+" s-a facut din
rezervorul: R"+imax+" in ziua de "+Z[jmax]+"</font>");
document.writeln("<p><font size=+1 color=green>Livrarea minima:"+min+" s-a facut din
rezervorul: R"+imin+" in ziua de "+Z[jmin]+"</font>");
document.writeln("<p><p><font size=+2 color=#0000ff>Trimite situatia prin Email </font><p>");
document.writeln("<p><p><form name=\"f2\" enctype=\"text/plain\">");
document.writeln("<p>Adresa:<input type=\"text\" name=\"adresa\" size=\"30\"
onBlur=\"verificaAdresa();\">");
document.writeln("<p><input type=\"button\" value=\"Trimite email\" onClick=\"trimite_email();\">");
</script>
</body>
</html>
```

Figura 5.12 *(continuare)*

În figura 5.13 şi figura 5.14 se prezintă rezultatele execuției script-ului.

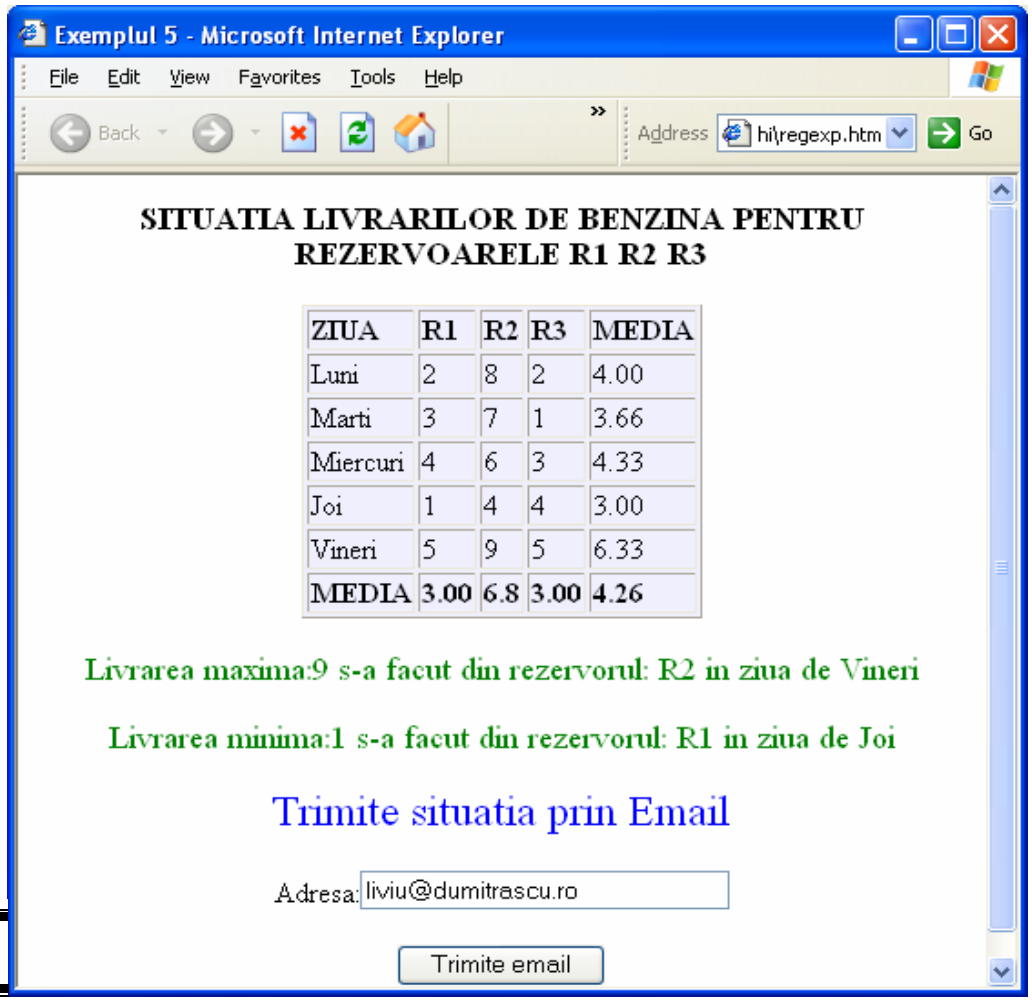

Exemplul 5 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address hi\regexp.htm
Go
SITUATIA LIVRARILOR DE BENZINA PENTRU REZERVOARELE R1 R2 R3
ZIUA R1 R2 R3 MEDIA
Luni 2 8 2 4.00
Marti 3 7 1 3.66
Miercuri 4 6 3 4.33
Joi 1 4 4 3.00
Vineri 5 9 5 6.33
MEDIA 3.00 6.8 3.00 4.26
Livrarea maxima:9 s-a facut din rezervorul: R2 in ziua de Vineri
Livrarea minima:1 s-a facut din rezervorul: R1 in ziua de Joi
Trimite situatia prin Email
Adresa: liviu@dumitrascu.ro
Trimite email

Figura 5.13

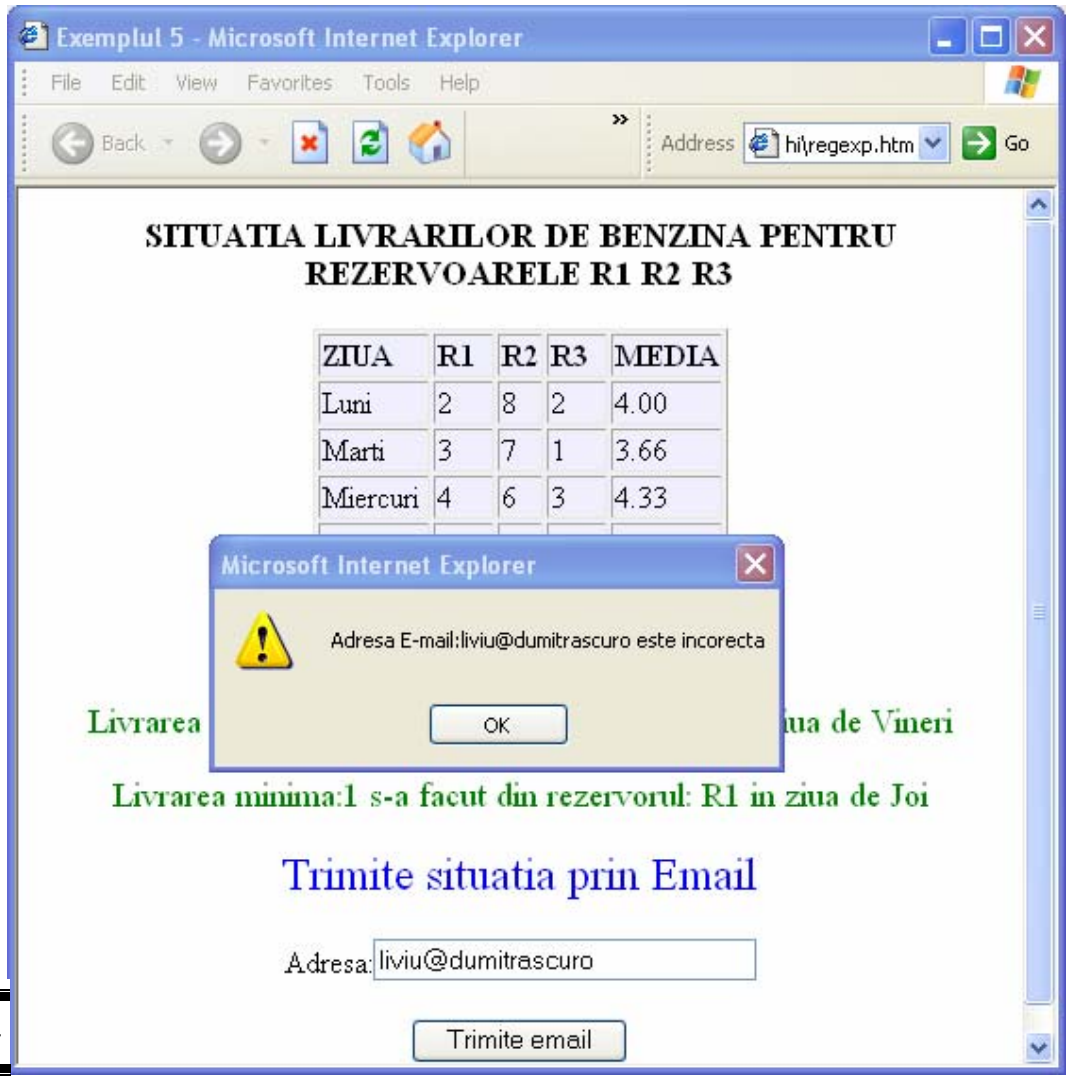


Figura 5.14

Comentarii:

- ✓ Verificarea adresei de e-mail se realizează cu gestionarul de evenimente `onBlur` care lansează în execuție funcția `VerificaAdresa()` (figura 5.12).
- ✓ Funcția `VerificaAdresa()` utilizează obiectul `re` cu modelul prezentat în figura 5.15.

Figura 5.15

```
re = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,})+$/;
```

- ✓ Expedierea prin e-mail a raportului se realizează prin acționarea butonului „`Trimite email`” care lansează în execuție funcția `trimite_email()` (figura 5.12).
- ✓ Corpul mesajului trimis prin email este generat de funcția `formeaza_afis`(figura 5.12).

Aplicație

❑ Scrieți un program JavaScript care verifică data calendaristică în formatul zz/ll/aa.

Indicație. O dată calendaristică trebuie formată dintr-un şir de caractere de forma zz/ll/aaaa. Zilele trebuie să aibă valori cuprinse în intervalul 1 şi 31. Prima cifră (dacă sunt două) trebuie să fie maxim 3, urmată eventual de 0 sau 1. Prima cifră poate fi de asemenea 0 urmată de cifre cuprinse în intervalul 1-9. Urmează caracterul „/”. Lunile trebuie să aibă valori cuprinse în intervalul 1 şi 12. Prima cifră poate fi 0 urmată de cifre cuprinse în intervalul 1-9 sau poate fi urmată de 0, 1 sau 2. Urmează caracterul „/”. Anul este un număr alcătuit din patru cifre.

În figura 5.16 este prezentat documentul XHTML complet.

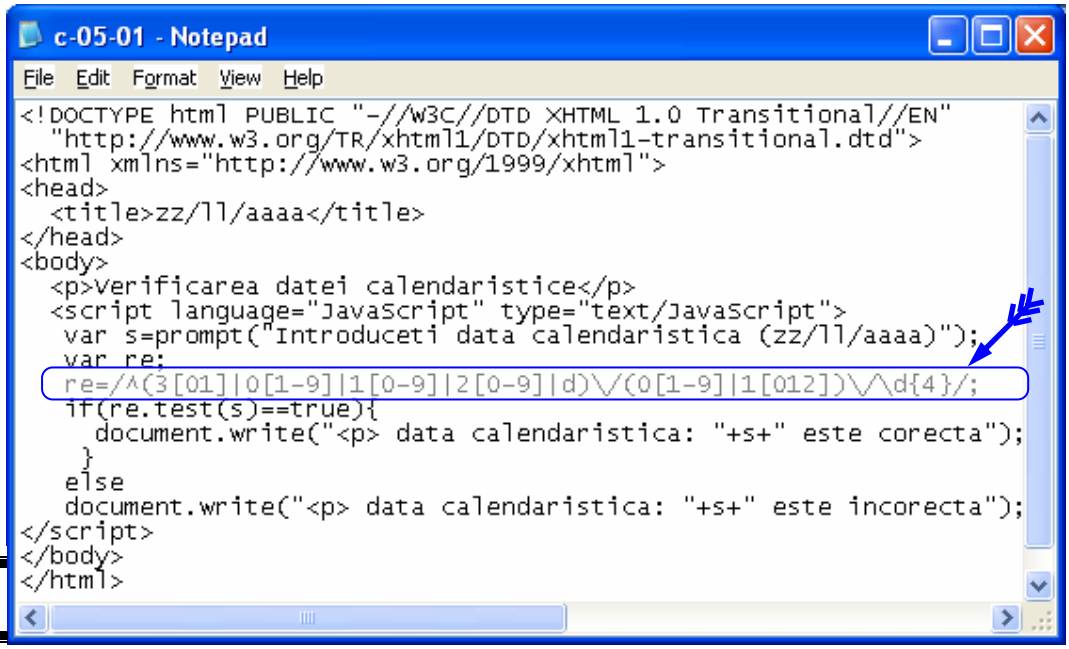


Figura 5.16

Rezultatele execuției script-ului sunt prezentate în figura 5.17.

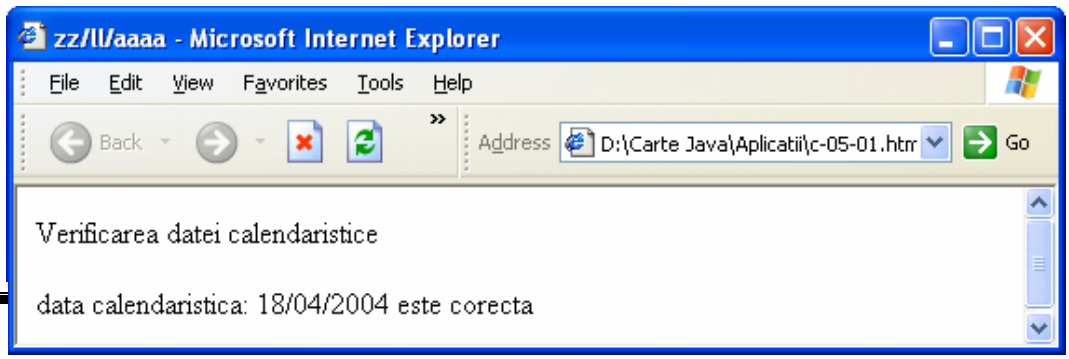


Figura 5.17

Obiectul Date

`Date` este un obiect predefinit al limbajului JavaScript care vă permite să lucraţi cu valori reprezentând orele şi datele calendaristice.

Remarci:

- ✓ JavaScript consideră că data iniţială (de referinţă) este 1 ianuarie 1970 (convenţia UNIX).
- ✓ JavaScript ţine evidenţa valorilor: oră sau data calendaristică în milisecunde, începând cu 1 ianuarie 1970.

Fişa obiectului `Date` este prezentată în figura 5.18.

Fişa obiectului DATE

Cum se creează obiectul?	`constructorul Date()`
Proprietăţi:	–
Metode:	`getDate(), getDay(), getFullYear(), getHours(), getMilliseconds(), getMinutes(), getMonth(), getSeconds(), getTime(), getTimezoneOffset(), getUTCDate(), getUTCDay(), getUTCFullYear(), getUTCHours(), getUTCMilliseconds(), getUTCMinutes(), getUTCMonth(), getUTCSeconds(), getVarDate(), getYear(), parse(), setDate(), setFullYear(), SetHours(), SetMilliseconds(), SetMinutes, SetTime(), setUTCDate(), setUTCFullYear(), setUTCHours(), setUTCMilliseconds, setUTCMinutes, setUTCMonth(), setUTCSeconds(), setYear, toDateString(), toGMTString(), toLocalDateString(), toLocaleString(), toLocalTimeString(), toString(), toTimeString(), toUTCString(), valueOf()`
Gestionarii de evenimente:	-

Figura 5.18

Constructorul Date()

Constructorul `Date()` creează o nouă dată.

Constructorul `Date()` este prezentat în detaliu în figura 5.19.

	Constructor	Sintaxă
	`Date()`	`Variabila=new Date()` `Variabila=new Date(An, Lună, Zi, Ore, Minute, Secunde, Milisecunde)`
	`new Date()` creează un obiect având data şi ora curente. Cel de-al doilea format creează un obiect cu data şi ora specificate. Puteţi crea un obiect `Date` folosind numai argumentele: *An, Lună, Zi*. Respectaţi ordinea argumentelor. Lunile se numără de la zero. Crearea unui obiect `Date` este similară cu crearea obiectelor `String, Array`.	
Exemplu:	<script> azi=new Date(); document.write(azi); //afişează data şi ora curente </script>	
Exemplu:	<script> azi=new Date(2002,5,9); alert(azi); //obiectul Date conține 9 mai 2002 </script>	

Figura 5.19

Metodele obiectului Date

Metodele obiectului `Date` permit: *definirea valorilor obiectelor Date* (`setDate(); setMonth(); setFullYear(); setTime(); setHours(); setMinutes(); setSeconds()` etc.); *citirea valorilor obiectelor Date* (`getDate(); getMonth(); getFullYear(); getTime(); getHours()` etc.); *gestionarea fusurilor orare şi a orelor locale* (`getTimeZoneOffset(); toUTCString(); toLocalString()` etc.); *conversia între formatele datelor* (`Date.parse(); Date.UTC()` etc.).

Metodele obiectului `Date` sunt prezentate în detaliu în figura 5.20.

	Metodă	***Sintaxă***
	`getDate()`	`Data.getDate()`
	Metoda returnează ziua din lună.	
Exemplu:	<script> data=new Date(2004,4,1); a=data.getDate(); document.write(a); ...//afişează 1 </script>	
	`getDay()`	`Data.getDay()`
	Metoda returnează un număr care corespunde zilei săptămânii: 0 pentru duminică; 1 pentru luni etc.	
Exemplu:	<script> data=new Date(2004,4,1); //1 aprilie 2004 ..a=data.getDay() document.write(a); ...//afişează 4 (pentru joi) </script>	
	`getFullYear()`	`Data.getFullYear()`
	Metoda returnează anul în 4 cifre.	
Exemplu:	<script> data=new Date(2004,4,1); a=data.getFullYear(); document.write(a); ...//afişează 2004 </script>	
	`getHours()`	`Data.getHours()`
	Metoda returnează numărul de ore (între 0 şi 23).	
Exemplu:	<script> data=new Date(2004,5,9,8,25,0); a=data.getHours(); document.write(a);...//afişează 8 </script>	
	`getMilliseconds()`	`Data.getMilliseconds()`
	Metoda returnează numărul de milisecunde.	
Exemplu:	<script> data=new Date(2004,5,9,8,25,8,750); a=data.getMilliseconds(); document.write(a); ...//afişează 750 </script>	
	`getMinutes()`	`Data.getMinutes()`
	Metoda returnează numărul de minute.	

Figura 5.20

Exemplu:
```
<script>
  data=new Date(2004,5,9,8,25,8,750);
  a=data.getMinutes();
  document.write(a);
...//afişează 25
</script>
```

`getMonth()` `Data.getMonth()`

Metoda returnează numărul lunii (de la 0 la 11).

Exemplu:
```
<script>
  data=new Date(2004,4,1);
  a=data.getMonth();
  document.write(a+1);
...//afişează 4
</script>
```

Exemplu:
```
<script>
  luna=new
Array("ianuarie","februarie","martie","aprilie","mai","iunie","iulie","august",
"septembrie","octombrie","noiembrie","decembrie");
  data=new Date(2004,4,1);
  a=data.getMonth();
  document.write(luna[a]); ...//afişează aprilie
</script>
```

`getSeconds()` `Data.getSeconds()`

Metoda returnează numărul de secunde (între 0 şi 59).

Exemplu:
```
<script>
  data=new Date(2004,5,9,8,25,8,750);
  a=data.getSeconds();
  document.write(a);...//afişează 8
</script>
```

`getTime()` `Data.getTime()`

Metoda returnează data în milisecunde, calculată de la 1 ianuarie 1970.

Exemplu:
```
<script>
  data=new Date(2002,5,9,8,25,8,750);
  a=data.getTime();
  document.write(a); ..//afişează 1023603908750
</script>
```

`getUTCDate()` `Data.getUTCDate()`

Metoda returnează zona din lună transpusă în timp `UTC` (Universal Coordinated Time).

Exemplu:
```
<script>
  data=new Date(2004,4,1);
  //1 aprilie 2004
  a=data.getUTCDate();
...//afişează 1
</script>
```

Figura 5.20 *(continuare)*

	`getUTCDay()` `Data.getUTCDay()`
	Metoda returnează ziua din săptămână transpusă în timp `UTC` (0 pentru duminică; 1 pentru luni etc.).
Exemplu:	<script> data=new Date(2002,5,9); //9 mai 2002 a=data.getUTCDay(); document.write(a); ...//afişează 0 (pentru duminică) </script>
Exemplu:	<script> ziua=new Array("duminică","luni","marți","miercuri","joi","vineri", „sâmbătă"); data=new Date(2002,5,9); //9 mai 2002 a=data.getUTCDay(); document.write(ziua[a]); ...//afişează duminică </script>
	`getUTCFullYear()` `Data.getUTCFullYear()`
	Metoda returnează anul transpus în timp `UTC`, în format de 4 cifre.
Exemplu:	<script> data=new Date(2004,4,1); //1 aprilie 2004 a=data.getUTCFullYear(); document.write(a);...//afişează 2004 </script>
	`getUTCHours()` `Data.getUTCHours()`
	Metoda returnează ora (între 0 şi 23) transpus în timp `UTC`.
Exemplu:	<script> data=new Date(2002,5,9,8,25,0); a=data.getUTCHours(); document.write(a);...//afişează 6 </script>
	`getUTCMilliseconds()` `Data.getUTCMilliseconds()`
	Metoda returnează numărul de milisecunde (transpus în timp `UTC`).
Exemplu:	<script> data=new Date(2002,5,9,8,25,8,750); a=data.getUTCMilliseconds(); document.write(a);...//afişează 750 </script>
	`getUTCMinutes()` `Data.getUTCMinutes()`
	Metoda returnează numărul de minute (transpus în timp `UTC`).
Exemplu:	<script> data=new Date(2002,5,9,8,25,8,750); a=data.getUTCMinutes(); document.write(a);...//afişează 25 </script>

Figura 5.20
(continuare)

`getUTCMonth()`	`Data.getUTCMonth()`
	Metoda returnează numărul lunii (transpus în timp `UTC`).
Exemplu:	<script> data=new Date(2002,5,9); //9 mai 2002 a=data.getUTCMonth(); document.write(a); ...//afişează 5 </script>
Exemplu:	<script> luna=new Array("Ianuarie","Februarie","Martie","Aprilie","Mai","Iunie","Iulie", "August", "Septembie","Octombrie","Noiembrie","Decembrie"); data=new Date(2004,4,1); //1 aprilie 2004 a=data.getUTCMonth(); document.write(luna[a–1]); ...//afişează aprilie </script>
`getUTCSeconds`	`Data.getUTCSeconds()`
	Returnează numărul de secunde (între 0 şi 59) transpus în timp `UTC`.
Exemplu:	<script> data=new Date(2002,5,9,8,25,8,750); a=data.getUTCSeconds(); document.write(a);...//afişează 8 </script>
`getVarDate()`	`Data.getVarDate()`
	Returnează data şi ora în litere (în limba engleză).
Exemplu:	<script> data=new Date(2004,4,1); a=data.getVarDate(); document.write(a);...//afişează Sun Jun 9 00:00:00 UTC+0200 2002 </script>
`setDate()`	`Data.setDate(`*`Număr`*`)`
	Metoda modifică ziua din lună într-un obiect `Date`. Metoda acceptă un argument (un număr cuprins între 1 şi 31) şi returnează noua dată în milisecunde.
Exemplu:	<script> data=new Date(2002,7,15); //15 august 2004 a=data.setDate(3); document.write(a+" "); afişează 10283256000000 document.write(data.toString()); ...//afişează Sat Aug 3 00:00:00 UTC+0200 2002 </script>

Figura 5.20
(continuare)

	`setFullYear()` `Data.setFullYear(Număr)`
	Metoda modifică anul într-un obiect `Date`. Această metodă acceptă un argument (anul în patru cifre) şi returnează noua dată în milisecunde.
Exemplu:	<script> data=new Date(2002,7,15); //15 august 2002 a=data.setFullYear(2004); document.write(a+" "); ...//afişează 109252 08 00000 document.write(data.toString()); //afişează Sun Aug 15 00:00:00 UTC+0200 2004 </script>
	`setHours()` `Data.setHours(Număr)`
	Metoda modifică ora într-un obiect `Date`. Această metodă acceptă un argument între (0 şi 23) şi returnează noua dată în milisecunde.
Exemplu:	<script> data=new Date(2002,7,15,14,25,32); //15 august 2002 a=data.setHours(3); document.write(a+" "); ...//afişează 1029374732000 document.write(data.toString()); //afişează Thu Aug 15 03 25:32 UTC+ 0200 2002 </script>
	`setMilliseconds()` `Data.setMilliseconds(Număr)`
	Metoda modifică numărul de secunde într-un obiect `Date`. Această metodă acceptă un argument (numărul de milisecunde) şi returnează noua dată în milisecunde.
Exemplu:	<script> data=new Date(2002,7,15,14,25,32,750); a=data.setMilliseconds(3); document.write(a+" "); document.write(data.toString()); ...//afişează The Aug 15 14 25:32 UTC+0200 2002 </script>
	`setMinutes()` `Data.setMinutes(Număr)`
	Metoda modifică minutele într-un obiect `Date`. Această metodă acceptă un argument (minutele între 0 şi 59) şi returnează noua dată în milisecunde.
	`setMonth()` `Data.setMonth(Număr)`
	Metoda modifică luna într-un obiect `Date`. Această metodă acceptă un argument (luna între 0 şi 12) şi returnează noua dată în milisecunde.

Figura 5.20
(continuare)

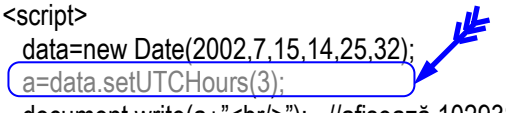
	`setSeconds()` `Data.setSeconds(`*`Număr`*`)`
	Metoda modifică secundele într-un obiect `Date`. Această metodă acceptă un argument (numărul de secunde între 0 şi 59) şi returnează noua dată în milisecunde.
	`setTime()` `Data.setTime()`
	Metoda modifică data într-un obiect `Date`. Această metodă acceptă un argument (data în milisecunde) şi returnează noua dată în milisecunde.
	`setUTCDate()` `Data.setUTCDate()`
	Modifică ziua din lună într-un obiect `Date` (transpusă în `UTC`). Această metodă acceptă un argument (un număr cuprins între 1 şi 31) şi returnează noua dată în milisecunde.
	`setUTCFullYear()` `Data.setUTCFullYear()`
	Modifică anul (în timp `UTC`) într-un obiect `Date`. Această metodă acceptă un argument (anul în patru cifre) şi returnează noua dată în milisecunde.
	`setUTCHours()` `Data.setUTCHours()`
	Modifică ora `UTC` într-un obiect `Date`. Această metodă acceptă un argument (ora cuprinsă între 0 şi 23) şi returnează noua dată în milisecunde.
Exemplu:	
	`setUTCMilliseconds()` `Data.setUTCMilliseconds()`
	Metoda modifică numărul `UTC` de milisecunde într-un obiect `Date`. Această metodă acceptă un argument (numărul de milisecunde) şi returnează noua dată în milisecunde.
	`setUTCMinutes()` `Data.setUTCMinutes()`
	Metoda modifică minutele `UTC` într-un obiect `Date`. Această metodă acceptă un argument (minutele între 0 şi 59) şi returnează noua dată în milisecunde.
	`setUTCMonth()` `Data.setUTCMonth()`
	Metoda modifică luna `UTC` într-un obiect `Date`. Această metodă acceptă un argument (luna între 0 şi 11) şi returnează noua dată în milisecunde.

Figura 5.20
(continuare)

`setUTCSeconds()`	`Data.setUTCSeconds()`
	Metoda modifică secundele în timp universal (`UTC`) într-un obiect `Date`. Această metodă acceptă un argument (numărul de secunde între 0 şi 59) şi returnează noua dată în milisecunde.
`setYear()`	`Data.setYear(`*`Număr`*`)`
	Metoda modifică anul într-un obiect `Date`. Această metodă acceptă un argument (anul cu două cifre) şi returnează noua dată în milisecunde. Dată fiind interpretarea foarte diferită a navigatoarelor, este mai bine să utilizaţi în locul acestei metode, metoda `setFullYear`().
`toDateString()`	`Data.toDateString()`
	Metoda returnează data în litere, în limba engleză.
Exemplu:	<script> data=new Date(2002,7,15,14,25,32,750); a=data.toDateString(); document.write(a+" "); …//afişează The Aug 15 2002 </script>
`toGMTString()`	`Data.toGMTString()`
	Metoda returnează data şi ora `GMT` în litere, în limba engleză. Este bine să utilizaţi în locul acestei metode, din motive de interpretare diferită de către browser, metoda `toString()`.
`toLocalDateString()`	`Data.toLocalDateString()`
	Metoda returnează data în litere, în formatul şi în limba maşinii pe care este executat script-ul.
Exemplu:	<script> data=new Date(2002,7,15,14,25,32,750); a=data.toLocalDateString(); document.write(a+" "); …//afişează Jeudi 15 août 2002 </script>
`toLocaleString()`	`Data.toLocaleString()`
	Metoda returnează data şi ora în litere, în formatul şi în limba maşinii pe care este executat script-ul.
`toLocalTimeString()`	`Data.toLocalTimeString()`
	Metoda returnează ora în formatul maşinii pe care este executat script-ul.
`toString()`	`Data.toString()`
	Metoda returnează data şi ora în litere, în limba engleză.
Exemplu:	<script>

Figura 5.20 *(continuare)*

	data=new Date(2002,7,15,14,25,32,750); a=data.toString(); document.write(a+" "); ...//afişează Thu Aug 15 14:32 UTC+0200 2002 </script>
	`toTimeString()` `Data.toTimeString()`
	Metoda returnează ora în format englezesc.
Exemplu:	<script> data=new Date(2002,7,15,14,25,32,750); a=data.toTimeString(); document.write(a+" "); ...//afişează 14:25:32 UTC+0200 </script>
	`toUTCString()` `Data.toUTCString()`
	Metoda returnează data şi ora UTC în litere, în limba engleză.
	`valueOf()` `Data.valueOf()`
	Metoda returnează data şi ora în milisecunde.
Exemplu:	<script> data=new Date(2002,7,15,14,25,32,750); a=data.valueOf(); document.write(a+" ");...//afişează 1029414332750 </script>

Figura 5.20 *(continuare)*

Aplicaţii

❑ Inseraţi data şi ora într-o pagină Web.

Indicaţie. Utilizaţi formatul: zz/ll/aaaa. (*Exemplu*, 30/1/2004 – 10:25 PM).

În figura 5.21 este prezentat documentul HTML ([4]) complet al aplicaţiei.

Figura 5.21

În figura 5.22 sunt vizualizate rezultatele execuţiei script-ului.

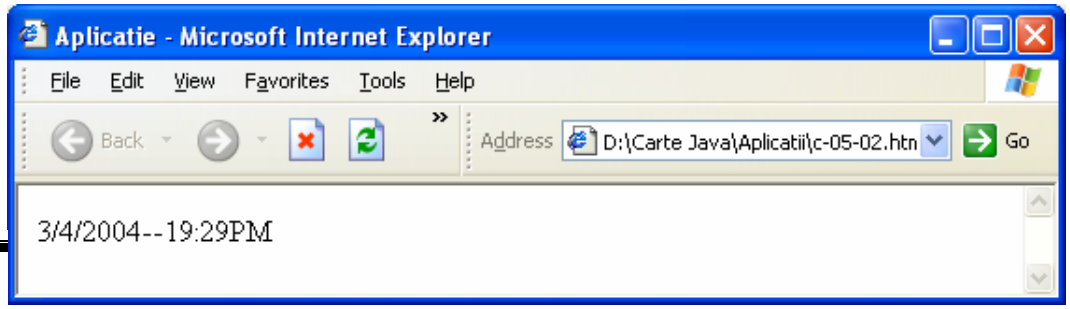


Figura 5.22

❑ Personalizaţi script-ul din aplicaţia precedentă.

❑ Afişaţi data calendaristică în litere, în limba română (Exemplu, joi 1 aprilie 2004).

În figura 5.23 este prezentat documentul HTML ([4]) complet.

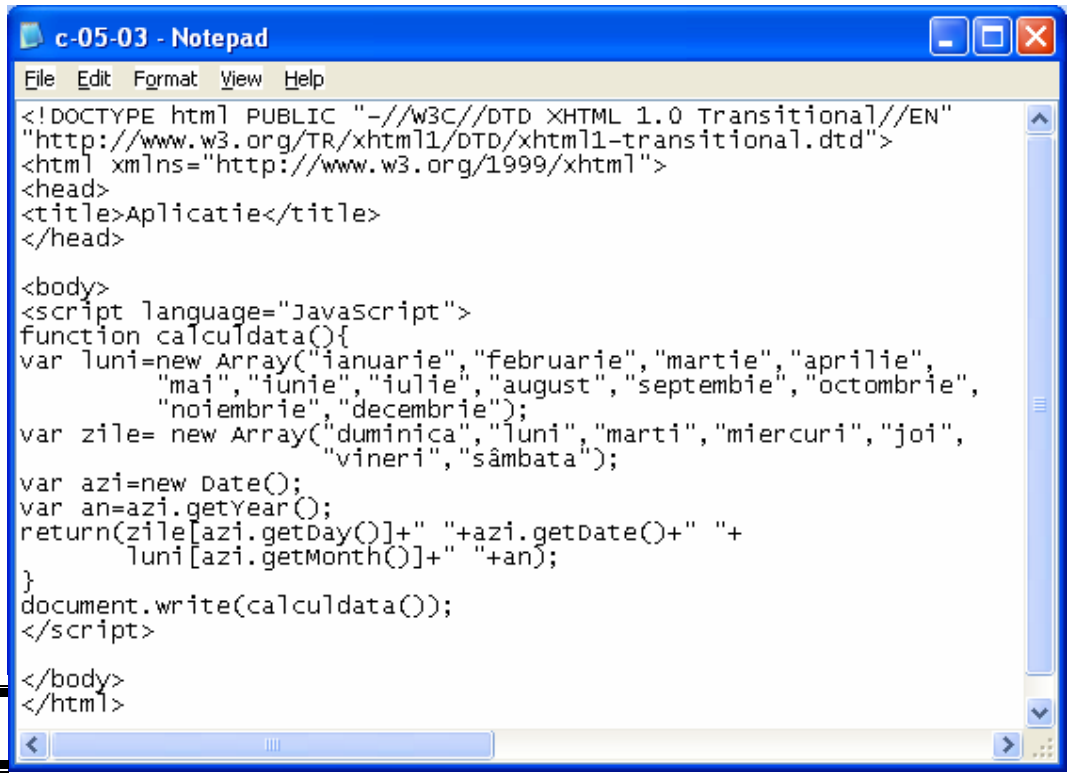


Figura 5.23

Rezultatele execuţiei script-ului sunt vizualizate în figura 5.24.

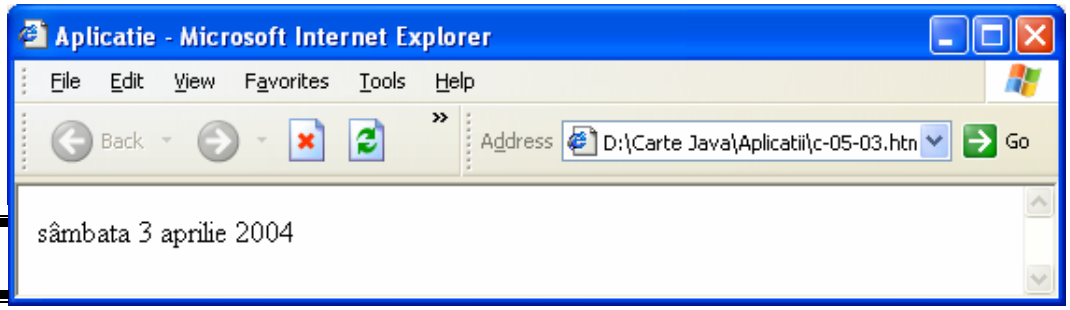


Figura 5.24

Obiectul Arguments

Obiectul `Arguments` reprezintă sub forma unei matrici (`Array`) valoarea argumentelor transmise unei funcţii.

În figura 5.25 este prezentată fişa obiectului `Arguments`.

Fişa obiectului Arguments	
Cum se creează obiectul?	Constructorul `Function()`
Proprietăţi:	`calee, length`
Metode:	–
Gestionarii de evenimente:	–

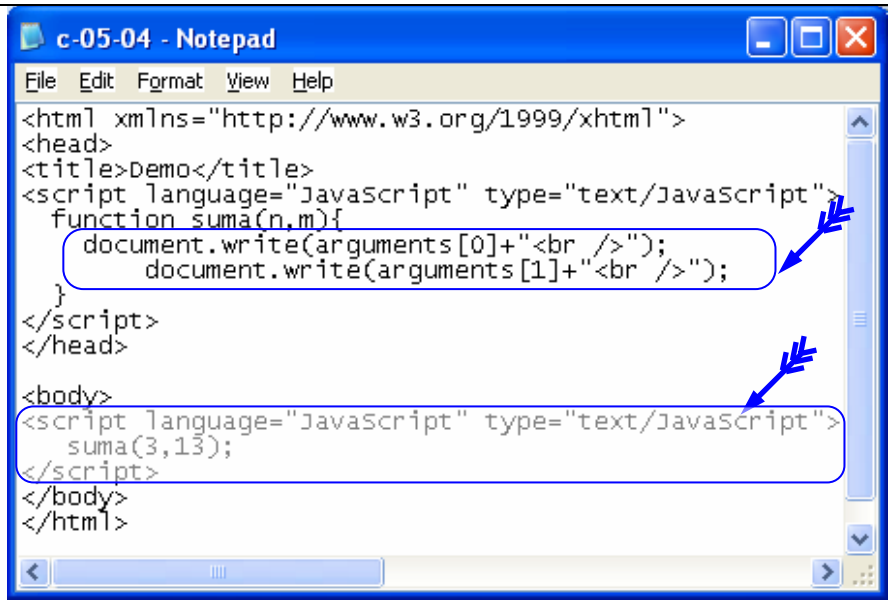
Figura 5.25

Obiectul `Arguments` este prezentat în detaliu în figura 5.26.

Obiect	***Sintaxă***
`Arguments`	`funcţie.arguments`

`Arguments` este un subobiect al obiectului (intern) `Function`. Atunci când apelaţi o funcţie, puteţi să-i transmiteţi argumente. Aceste argumente sunt disponibile sub numele lor dar şi ca elemente ale unei matrici `Array: arguments[]`.

Exemplu:



Rezultatele execuţiei script-ului:

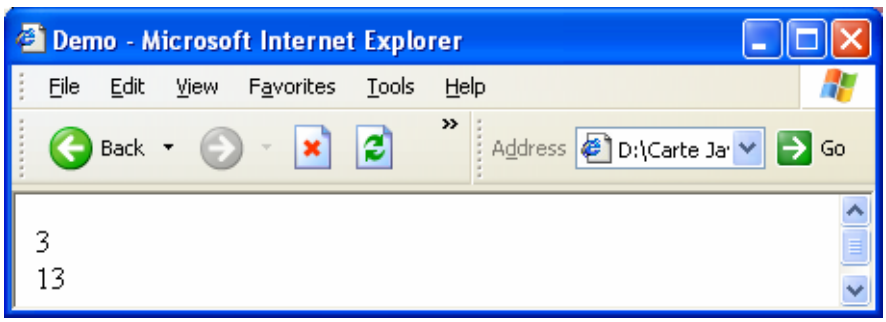


Figura 5.26

Proprietăţile obiectului Arguments

Proprietăţile obiectului `Arguments` sunt prezentate în detaliu în figura 5.27.

	Proprietate	***Sintaxă***
	`callee`	`funcţie.arguments.calle`
	Returnează conţinutul complet al funcţiei.	

Exemplu:

c-05-05 - Notepad

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Proprietatea calle</title>
<script language="JavaScript" type="text/JavaScript">
  function suma(n,m){
    document.write(suma.arguments.callee);
  }
</script>
</head>

<body>
<script language="JavaScript" type="text/JavaScript">
   suma(3,13);
</script>
</body>
</html>
```

Rezultatele execuţiei script-ului:

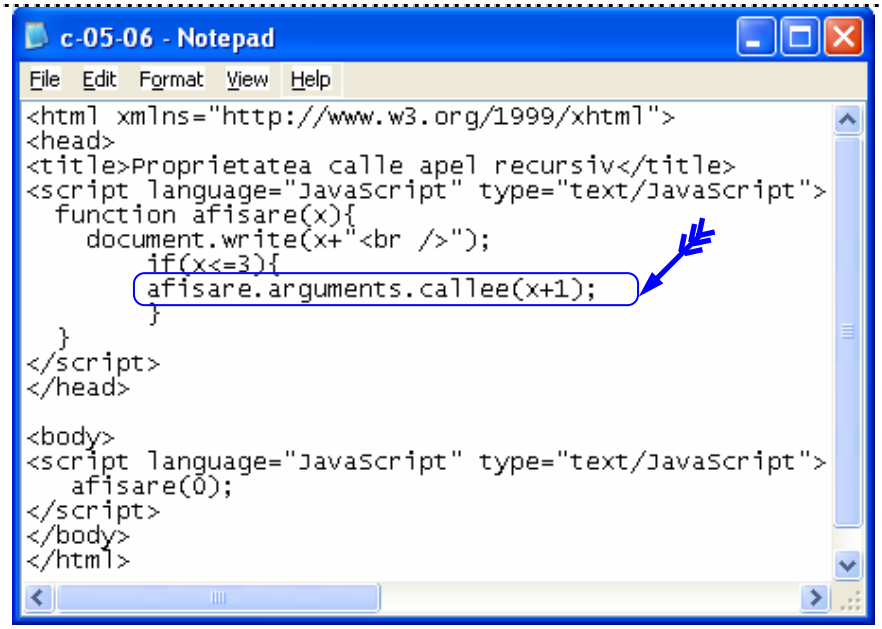
Proprietatea calle - Microsoft Internet Explorer

File Edit View Favorites Tools Help

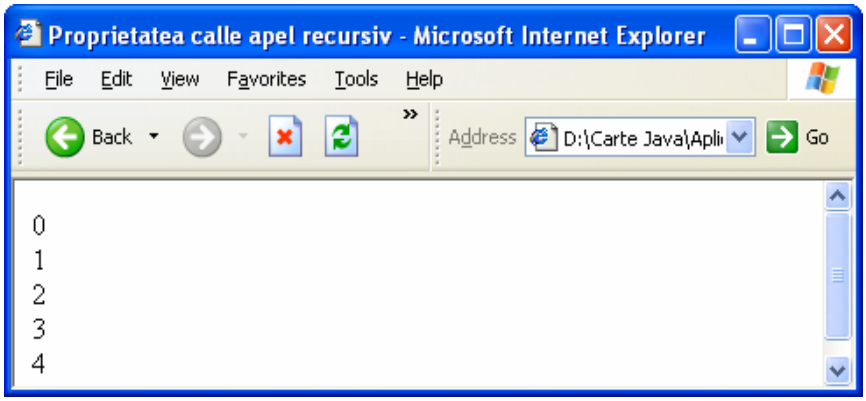
Back Address licatii\c-05-05.htm Go

```
function suma(n,m){ document.write(suma.arguments.callee); }
```

Figura 5.27



Rezultatele execuţiei script-ului:



	`length`	`funcţie.arguments.length`
	Returnează numărul de argumente transmise funcţiei atunci când ea este apelată.	

Exemplu:

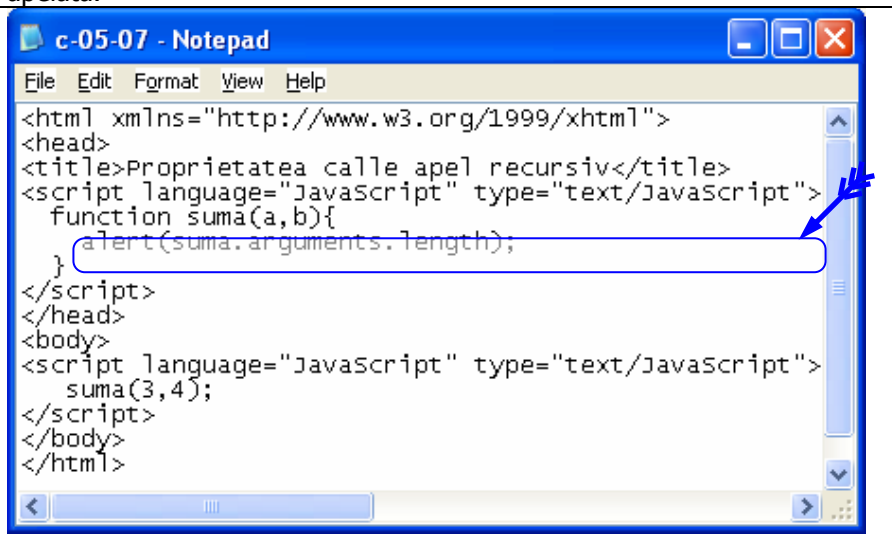


Figura 5.27
(continuare)

Exemplu:

c-05-08 - Notepad

File Edit Format View Help

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Proprietatea calle apel recursiv</title>
<script language="JavaScript" type="text/JavaScript">
  function suma(a,b){
    for(i=0;i<suma.arguments.length;i++){
        document.write(arguments[i]+"<br />");
      }
  }
</script>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
   suma(3,13);
</script>
</body>
</html>
```

Rezultatele execuţiei script-ului:

Proprietatea calle apel recursiv - Microsoft Internet Explorer

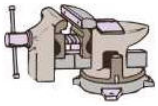
File Edit View Favorites Tools Help

Back Address D:\Carte Java\Apli Go

3
13

Figura 5.27
(continuare)

Testați-vă cunoștințele



1. Ce reprezintă expresiile regulate?
2. Cum se creează obiectul `RegExp`?
3. Care sunt proprietăţile şi metodele obiectului `RegExp`?
4. Care sunt caracterele de repetiţie utilizate în identificarea modelelor (tiparelor)?
5. Care sunt clasele de caractere utilizate în identificarea modelelor?
6. Care sunt identificatorii de poziţie utilizaţi în identificarea modelelor?
7. Care este efectul utilizării următoarelor metode `RegExp`:
   - compile();
   - test().
8. Următoarele două script-uri verifică un cod poştal (figura 5.28) şi o adresă de e-mail (figura 5.29). Comentaţi structura expresiilor din cadrul celor două script-uri.

```
<script>
cod=”85015”;
model=/^[0–9]{5}$/;
if(model.test(cod)){
  document.write(“OK”);
}
else{
  document.write(“cod eronat”);
}
</script>
```

Figura 5.28

```
<script>
adresa=”vasile@vasile.com”;
model=/^[a–zA–Z0–9._–]+@[a–zA–Z0–9._–]+\.[a–zA–Z0–9._–]{2,4}/$;
if(model.test(adresa)){
  document.write(“OK”);
}
else{
  document.write(“adresa e-mail eronată”);
}
</script>
```

Figura 5.29

9. Analizaţi funcţia `compdate()` din figura 5.30.

```
function compdate(data1,data2){
   difMilisecunde=data1–data2;
   difSecunde=difMilisecunde/1000;
   difMinute=difSecunde/60;
   difOre=difMinute/60;
   difZi=difOre/24;
   return difZi;
}
```

Figura 5.30

10.Cum creaţi un obiect `Date` care conţine data şi ora curente?

11.Ce reprezintă GMT şi UTC?

12.Care sunt proprietăţile obiectului `Date`?

13.Precizaţi rezultatul execuţiei următoarelor instrucţiuni JavaScript (figura 5.31).

```
<script>
  astăzi=new Date();
  a=atăzi.toLocalesString();
  alert(a);  //afişează ...?...
  b=astăzi.toString();
  alert(b);  //afişează ...?...
  c=astăzi.getTime();
  alert(c);  //afişează ...?...
  d=astăzi.getTimezoneOffset(); //afişează ...?...
</script>
```

Figura 5.31

14.Precizaţi care este efectul următoarelor metode JavaScript (figura 5.32).

```
<script>
  var data=new Date("May 13, 2004 21:40:00");
  alert(data.toGMTString());  //afişează ...?...
  alert(data.setMonth(9));  //afişează ...?...
</script>
```

Figura 5.32

15.Care este diferenţa între metodele `getDay()` şi `getDate()`?

16.Cum se creează obiectul `Arguments`?

17.Care sunt proprietăţile şi metodele obiectului `Arguments`?

Vizitaţi site-urile

- ✓ http://www.webreference.com/programming/Javascript.html
- ✓ http://scriptsearch.internet.com
- ✓ microsoft.public.scripting.jscript
- ✓ netscape.public.beta.feedback.javascript
- ✓ netscape-devs.javascript
- ✓ comp.infosystems.www.authoring.html

Conversația 6

Gestionarii de evenimente JavaScript

În această conversație:

- *Evenimente şi gestionari de evenimente*
- *Gestionarii de evenimente JavaScript*
- *Aplicaţii*
- *Temă*

Evenimente şi gestionari de evenimente

După cum am afirmat, JavaScript este un limbaj condus de evenimente. Foarte multe din programele pe care le scrieţi în limbajul JavaScript vor răspunde la un `eveniment` iniţiat fie de către utilizator, fie de către browser.

Reacţia la un eveniment este cunoscută sub numele de `prelucrarea` evenimentului, iar codul JavaScript corespunzător (pe care îl scrieţi) este cunoscut sub numele de `gestionar de evenimente`.

Un `gestionar de evenimente` este o metodă puţin specială care va fi apelată în mod automat de către navigator ori de câte ori va surveni un eveniment particular.

`Gestionarii de evenimente` sunt funcţii puternice JavaScript. Din fericire ei sunt foarte uşor de programat. De foarte multe ori este suficientă o singură instrucţiune pentru a putea fi creaţi.

Remarci:

- ✓ Un eveniment este o `acţiune` care se produce în raport cu un element (fereastră, document, buton etc.). El poate fi detectat şi prelucrat de către un script care va declanşa o acţiune. Script-ul este executat dacă evenimentul se produce pe obiectul căruia îi este asociat. Acţiunile utilizatorilor sunt elementele cele mai frecvente.
- ✓ Evenimentele generate de utilizator nu sunt singurele evenimente generate de JavaScript. De exemplu, evenimentul `load`, declanşat automat de către navigator, nu se produce decât atunci când este încheiată încărcarea unui document (X)HTML într-un navigator.
- ✓ Acelaşi eveniment se poate aplica mai multor obiecte.
- ✓ Evenimentele stau la baza interactivităţii documentelor Web.
- ✓ Câteva din evenimentele care survin în mod frecvent cât şi momentul în care acestea se declanşează sunt prezentate în figura 6.1.

Eveniment	***Se declanşează atunci când ...***
`abort`	încărcarea unui obiect este întreruptă.
`blur`	un element nu este activat.
`click`	utilizatorul execută clic pe un obiect.
`change`	valoarea unui element este modificată.
`error`	se produce o eroare la încărcarea unui document sau a unei imagini.
`focus`	un element devine inactiv.
`load`	un document sau o imagine se încarcă.
`mouseOut`	mouse-ul se deplasează în afara elementului.
`mouseOver`	mouse-ul se deplasează pe deasupra elementului.
`reset`	datele introduse într-un formular sunt şterse.
`select`	utilizatorul selectează un câmp într-un formular.
`submit`	utilizatorul expediază un formular.
`unload`	un document este descărcat.

Figura 6.1

După cum aţi putut observa, fiecare element are propriul său nume. De exemplu, evenimentul `click` se produce atunci când utilizatorul execută clic pe un obiect. În acest caz, evenimentul `click` este trimis gestionarului de evenimente, dacă acesta există. Pentru a defini un gestionar de evenimente adăugaţi prefixul `on` la numele evenimentului.

Codul gestionarului de evenimente trebuie să fie inclus ca un atribut al unui tag (X)HTML care declanşează evenimentul:

```
<tag (X)HTML gestionarEvenimente="cod JavaScript">
```

Remarci:

- ✓ Codul JavaScript al unui gestionar de evenimente este inclus între ghilimele.
- ✓ Observaţi combinaţia … ciudată de majuscule şi minuscule la notaţia standard a gestionarilor de evenimente: `on` este scris mereu cu minuscule iar iniţiala fiecărui cuvânt al evenimentului, cu majuscule. *Exemple:* `onAbort`, `onBlur`, `onClick`,

onChange, onError, onFocus, onLoad, onMouseOut, onMouseOver, onReset, onSelect, onSubmit, onUnLoad.

În figura 6.2 sunt listaţi, în ordine alfabetică gestionarii de evenimente ai limbajului JavaScript.

onAbort	onActivate	onAfterPrint
onAfterUpdate	onBeforeActivate	onBeforeCopy
onBeforeCut	onBeforeDeActivate	onBeforeEditFocus
onBeforePaste	onBeforePrint	onBeforeUnload
onBeforeUpdate	onBlur	onCellChange
onChange	onClick	onContextMenu
onControlSelect	onCopy	onCut
onDblClick	onDeActivate	onDrag
onDragDrop	onDragEnd	onDragStart
onDrop	onError	onFocus
onFocusIn	onFocusOut	onHelp
onKeyDown	onKeyPress	onKeyUp
onLoad	onMouseDown	onMouseMove
onMouseOut	onMouseOver	onMouseUp
onMove	onMoveEnd	onPropertyChange
onReset	onResize	onResizeError
onResizeEnd	onResizeStart	onScroll
onSelect	onSubmit	onUnLoad

Figura 6.2

Gestionarii de evenimente JavaScript

Gestionarii de evenimente JavaScript sunt prezentaţi în detaliu în figura 6.3.

	Gestionari de evenimente
	onAbort
	Se declanşează atunci când încărcarea unui element este anulată.
Se aplică la:	document
	onActivate
	Se declanşează atunci când elementul devine element activ.
Se aplică la:	window, document, frame, form

Figura 6.3

	onAfterPrint
	Se declanşează după imprimarea unui element.
Se aplică la:	window, frame
	onAfterUpDate
	Se declanşează după actualizarea datelor sursă ale elementului.
Se aplică la:	document, frame, form.button, form.checkbox, form.fileupdate, form.hidden, form.password, form.select, form.radio, form.reset, form.text, form.textarea, form.submit
	onBeforeActivate
	Se declanşează înainte ca elementul să devină element activ.
Se aplică la:	window, form
	onBeforeCopy
	Se declanşează înainte ca datele să fie copiate (Copy).
Se aplică la:	form
	onBeforeCut
	Se declanşează înainte ca datele să fie mutate (Cut).
Se aplică la:	document, form
	onBeforeDeActivate
	Se declanşează înainte ca un alt element să devină activ în documentul părinte.
Se aplică la:	document, form
	onBeforeEditFocus
	Se declanşează înainte ca elementul să primească focus-ul pentru a fi editat.
Se aplică la:	document, form
	onBeforePaste
	Se declanşează înainte ca elementul ţintă să primească datele (Paste).
Se aplică la:	document, form
	onBeforePrint
	Se declanşează înainte ca datele elementului să fie imprimate sau personalizate.
Se aplică la:	window, frame
	onBeforeUnload

Figura 6.3
(continuare)

	Se declanşează înainte ca elementul să fie descărcat.
Se aplică la:	`window`
	`onBeforeUpdate`
	Se declanşează înainte ca datele sursă ale elementului să fie actualizate.
Se aplică la:	`document, form.button, form.checkbox, form.fileupdate, form.hidden, form.password, form.select, form.radio, form.reset, form.text, form.textarea, form.submit`
	`onBlur`
	Se declanşează atunci când elementul pierde focus-ul.
Se aplică la:	`window, frame, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.radio, form.reset, form.text, form.textarea, form.submit`
	`onCellChange`
	Se declanşează atunci când datele se schimbă în obiectul sursă.
Se aplică la:	`document`
	`onChange`
	Se declanşează atunci când elementul pierde focus-ul iar conţinutul său a fost schimbat.
Se aplică la:	`form.fileupdate, form.password, form.select, form.textarea`
	`onClick`
	Se declanşează atunci când se execută clic pe un element.
Se aplică la:	`document, form, form.button, form.checkbox, form.option, form.radio, form.reset, form.text, form.submit`
	`onContextMenu`
	Se declanşează atunci când se execută clic cu butonul drept al mouse-ului pentru a deschide meniul contextual.
Se aplică la:	`document, form`
	`onControlSelect`
	Se declanşează înainte ca elementul să fie selectat.
Se aplică la:	`window, document, frame, form`
	`onCopy`
	Se declanşează atunci când se copiază un element.
Se aplică la:	`form`
	`onCut`
	Se declanşează atunci când se mută (`Cut`) un element.

Figura 6.3 *(continuare)*

Se aplică la:	`document, form`
	`onDblClick`
	Se declanşează atunci când se execută dublu click pe element.
Se aplică la:	`document, form, form.button, form.checkbox, form.option, form.reset`
	`onDeActivate`
	Se declanşează atunci când un alt element devine activ în documentul părinte.
Se aplică la:	`window, frame, form`
	`onDrag`
	Se declanşează atunci când un element este deplasat.
Se aplică la:	`document, form`
	`onDragDrop`
	Se declanşează atunci când ceva este deplasat şi apoi eliberat.
Se aplică la:	`window, frame`
	`onDragEnd`
	Se declanşează atunci când un element deplasat este la finele deplasării.
Se aplică la:	`document, form`
	`onDragStart`
	Se declanşează la începutul deplasării elementului.
Se aplică la:	`document, form, form.fileupdate, form.select, form.textarea`
	`onDrop`
	Se declanşează atunci când un obiect este depus peste un element după deplasare.
Se aplică la:	`document, form`
	`onError`
	Se declanşează atunci când se generează o eroare în raport cu un element.
Se aplică la:	`window, frame`

Figura 6.3 *(continuare)*

	`onFocus`
	Se declanşează atunci când elementul primeşte focus-ul.
Se aplică la:	`window, frame, form, form.button, form.checkbox,`

	form.fileupdate, form.password, form.select, form.radio, form.reset, form.text, form.textarea, form.submit
	onFocusIn
	Se declanşează înainte ca elementul să primească focus-ul (înaintea evenimentului focus).
Se aplică la:	form
	onFocusOut
	Se declanşează după ce elementul pierde focus-ul, imediat după ce un alt element a primit focus-ul.
Se aplică la:	form
	onHelp
	Se declanşează atunci când se acţionează tasta **F1** în fereastra activă.
Se aplică la:	window, document, frame, form, form.button, form.checkbox, form.fileupdate, form.hidden, form.password, form.select, form.option, form.radio, form.reset, form.text, form.textarea, form.submit
	onKeyDown
	Se declanşează atunci când se acţionează o tastă în timp ce elementul a primit focus-ul.
Se aplică la:	document, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.radio, form.reset, form.text, form.textarea, form.submit
	onKeyPress
	Se declanşează atunci când se menţine apăsată o tastă, în timp ce elementul a primit focus-ul.
Se aplică la:	document, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.radio, form.reset, form.text, form.textarea, form.submit
	onKeyUp
	Se declanşează atunci când este eliberată o tastă în timp ce elementul a primit focus-ul.
Se aplică la:	document, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.radio, form.reset, form.text, form.textarea, form.submit

Figura 6.3
(continuare)

	`onLoad`
	Se declanşează atunci când elementul este complet încărcat.
Se aplică la:	`window, frame`
	`onMouseDown`
	Se declanşează atunci când se acţionează un buton al mouse-ului în timp ce elementul a primit focus-ul.
Se aplică la:	`document, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.reset, form.text, form.textarea, form.submit`
	`onMouseMove`
	Se declanşează atunci când se deplasează mouse-ul iar pointer-ul este deasupra elementului.
Se aplică la:	`window, frame, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.reset, form.text, form.textarea, form.submit`
	`onMouseOut`
	Se declanşează atunci când pointer-ul mouse-ului părăseşte elementul.
Se aplică la:	`document, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.reset, form.text, form.textarea, form.submit`
	`onMouseOver`
	Se declanşează atunci când pointer-ul mouse-ului trece pe deasupra elementului.
Se aplică la:	`document, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.reset, form.text, form.textarea, form.submit`
	`onMouseUp`
	Se declanşează atunci când un buton al mouse-ului se relaxează în timp ce elementul a primit focus-ul.
Se aplică la:	`document, form, form.button, form.checkbox, form.fileupdate, form.password, form.select, form.option, form.reset, form.text, form.textarea, form.submit`
	`onMove`
	Se declanşează atunci când elementul este deplasat de către utilizator sau de script.
Se aplică la:	`window, frame, form`
	`onMoveEnd`
	Se declanşează la finele deplasării elementului.
Se aplică la:	`window, frame, form`

Figura 6.3
(continuare)

	`onPropertyChange`
	Se declanşează atunci când se modifică o proprietate a elementului.
Se aplică la:	`document, form`
	`onReset`
	Se declanşează atunci când formularul este resetat (se execută clic pe butonul `Reset`).
Se aplică la:	`form`
	`onResize`
	Se declanşează înainte ca elementul să fie redimensionat.
Se aplică la:	`window, frame, form, form.fileupdate, form.password, form.select`
	`onResizeEnd`
	Se declanşează la finele redimensionării elementului.
Se aplică la:	`window, frame, form`
	`onResizeStart`
	Se declanşează atunci când utilizatorul începe redimensionarea elementului.
Se aplică la:	`window, frame, form`
	`onScroll`
	Se declanşează atunci când fereastra începe să defileze.
Se aplică la:	`window, frame`
	`onSelect`
	Se declanşează atunci când este selectat conţinutul elementului.
Se aplică la:	`form.fileupdate, form.password, form.text, form.textarea, form.submit`
	`onSubmit`
	Se declanşează înaintea expedierii unui formular.
Se aplică la:	`form`

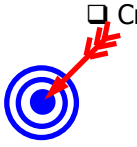
Figura 6.3 *(continuare)*

	`onUnLoad`
	Se declanşează înainte ca elementul să fie descărcat.
Se aplică la:	window

Figura 6.3 *(continuare)*

Aplicaţii

❑ Creaţi gestionarul de evenimente `onLoad`.



Iată cum procedăm pentru a crea gestionarul de evenimente `onLoad`, care conţine o instrucţiune (`alert`) care afişează o casetă de avertisment ce conţine mesajul: *"Învăţaţi să priviţi dincolo de aparenţe!"*.

1. Definiţi evenimentul care urmează a fi declanşat.

   Evenimentul care se declanşează (atunci când ultimul caracter al unui document HTML a fost încărcat în navigator) este `load` căruia îi corespunde gestionarul de evenimente `onLoad`.

2. Introduceţi numele gestionarului (`onLoad`) în tag-ul `body` (figura 6.4).

Figura 6.4

3. Introduceţi, în continuare după semnul egal, instrucţiunea JavaScript `alert`, plasată între ghilimele (figura 6.5).

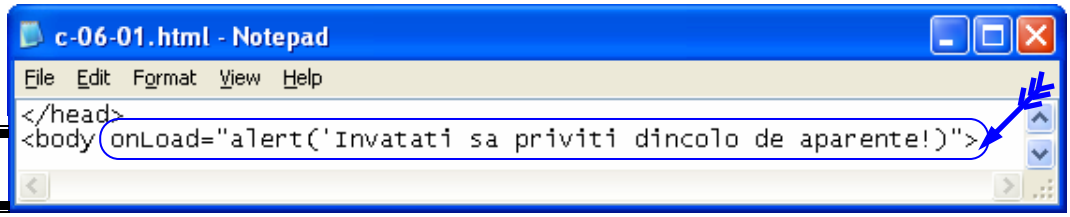


Figura 6.5

Remarci:

- ✓ În această etapă este important să vă întrebaţi: ce va produce gestionarul de evenimente `onLoad`?
- ✓ Atunci când definiţi un gestionar de evenimente `onLoad` pentru tag-ul `<body>` evenimentul (`Load`) este declanşat la finele încărcării documentului.
- ✓ Gestionarul de evenimente `onLoad` (definit în tag-ul `<body>`) afişează o casetă de avertisment odată cu terminarea încărcării paginii Web. Navigatorul verifică dacă un gestionar de evenimente `onLoad` este definit în tag-ul `<body>`. În situaţia în care acesta este identificat, se execută instrucţiunea JavaScript plasată între ghilimele, după semnul egal.
  Astfel, după încărcarea paginii se afişează o casetă de avertisment pentru a vă avertiza că nu este rău să învăţaţi să priviţi dincolo de aparenţe!
- ✓ În măsura în care gestionarul de evenimente este declanşat odată ce documentul (X)HTML a fost încărcat şi afişat nu puteţi utiliza în gestionarul de evenimente `onLoad` instrucţiunea `document.write` sau `document.open`.
- ✓ Începând cu versiunea 1.1 a limbajului JavaScript, imaginile pot avea şi ele un gestionar de evenimente `onLoad`. Atunci când definiţi un gestionar de evenimente `onLoad` pentru un tag `<img>`, evenimentul (`Load`) este declanşat la finele încărcării imaginii.
- ✓ În tag-ul `<body>` puteţi de asemenea defini gestionarul de evenimente `onUnLoad`. Evenimentul `UnLoad` este declanşat atunci când se încarcă un nou document sau când fereastra navigatorului este închisă.

4. Completaţi structura HTML a documentului (figura 6.6).

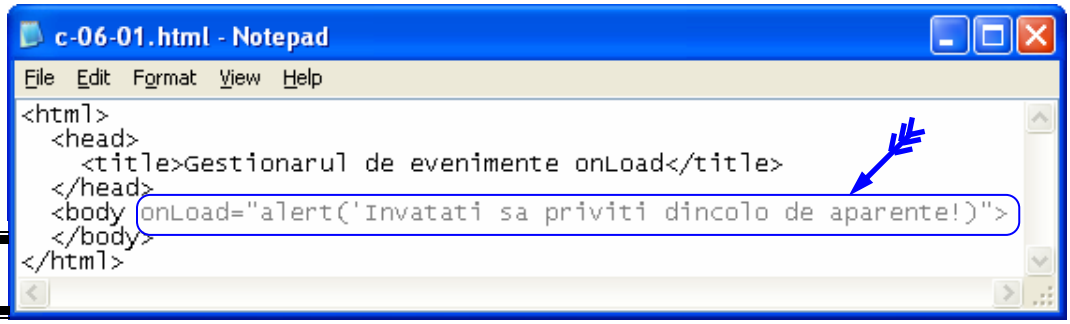


Figura 6.6

5. Afişaţi pagina Web într-un browser (figura 6.7).

Figura 6.7

Remarci:

- ✓ Atunci când utilizatorul încarcă această pagină Web într-un navigator, gestionarul de evenimente `onLoad` asociat paginii Web (sau documentului) unde se execută această acţiune, va fi activat.
- ✓ Încărcarea acestei pagini Web are ca efect apariţia mesajului: *"Învăţaţi să priviţi dincolo de aparenţe!"*.

❑ Modificaţi structura gestionarului de evenimente `onLoad` pe care l-aţi creat în aplicaţia precedentă, utilizând mai multe instrucţiuni JavaScript.

Iată cum modificăm (*Varianta 1, Varianta 2*) structura gestionarului de evenimente `onLoad` pe care l-am creat în aplicaţia precedentă, utilizând de această dată mai multe instrucţiuni JavaScript.

Varianta 1 (figura 6.8)

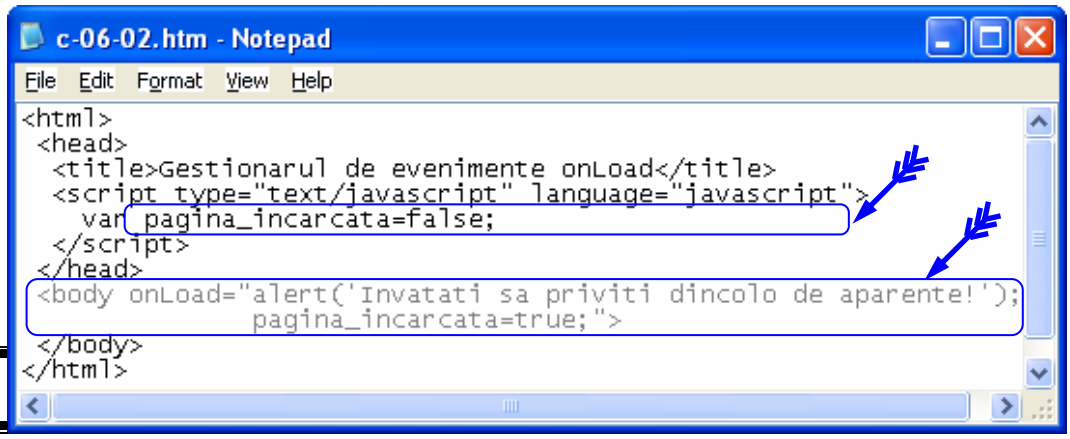


Figura 6.8

Remarci:

- ✓ În această variantă, gestionarul nu conţine decât două instrucţiuni (separate prin punct şi virgulă) dar el poate să conţină foarte bine oricâte instrucţiuni dorim.
- ✓ Codul JavaScript al gestionarului de evenimente are acces complet la variabila globală: `pagina_incarcata`. Nu uitaţi că variabilele globale JavaScript sunt accesibile peste tot, inclusiv în gestionarul de evenimente.
- ✓ În această variantă instrucţiunea de atribuire.

  `pagina_incarcata=true;`

  este necesară pentru a indica navigatorului că poate interacţiona cu utilizatorul.

Varianta 2 (figura 6.9)

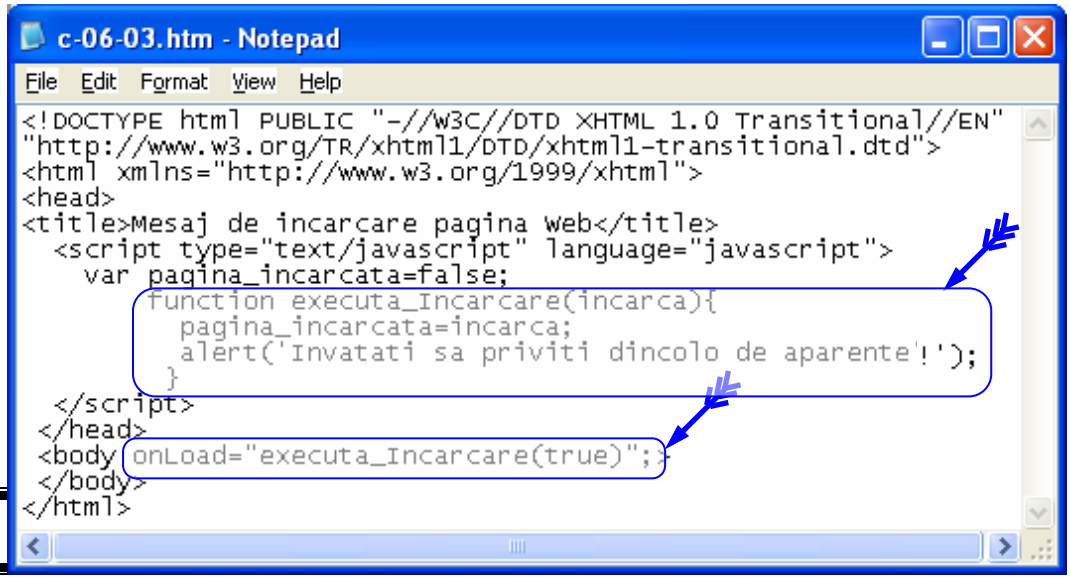


Figura 6.9

Remarci:

- ✓ Gestionarul de evenimente `onLoad` apelează o funcţie (`execut•_Inc•rcare`) cu un parametru.
- ✓ Parametrul *incarca* al funcţiei definite primeşte valoarea `TRUE` ce se atribuie variabilei (locale) `pagina_incarcata`.

❑ Utilizaţi un formular (X)HTML pentru afişarea următoarelor mesaje de încărcare şi descărcare a unei pagini Web.

- ✓ "*Pagina este în curs de încărcare!*";

- ✓ *"Pagina a fost încărcată"*
- ✓ *"La revedere!"* (mesajul se va afişa într-o casetă de avertisment!)

Remarcă. Obiectul `Form` este prezentat în detaliu în Conversaţia 8.

În figura 6.10 se prezintă documentul (X)HTML complet.

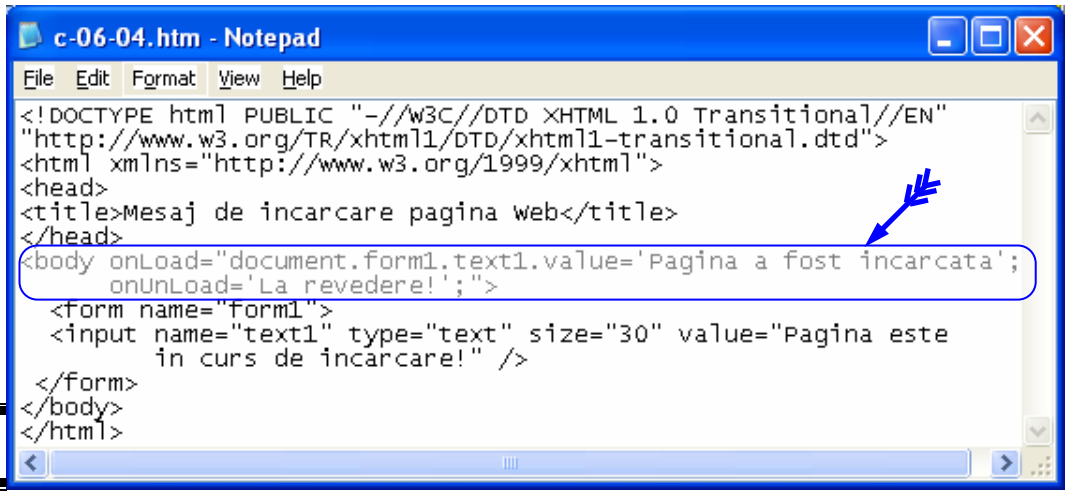


Figura 6.10

Remarci:

- ✓ Tag-ul `<body>` conţine doi gestionari de evenimente:
  - `onLoad`;
  - `onUnLoad`.
- ✓ Gestionarul `onLoad` este activat în momentul în care pagina a fost încărcată integral.
- ✓ Gestionarul `onUnLoad` este activat în momentul în care încărcaţi o altă pagină Web în navigator.
- ✓ Nu este întotdeauna recomandabil să afişaţi o casetă de avertisment atunci când vizitatorul părăseşte o pagină Web. Numeroşi vizitatori nu agreează ideea de a executa clic într-o casetă de avertisment pentru a putea părăsi pagina dumneavoastră Web. În consecinţă, această tehnică trebuie exploatată cu prudenţă!

❑ Creaţi script-uri care afişează mesaje în bara de stare a navigatorului.

Indicaţie. Una din aplicaţiile curente ale gestionarilor de evenimente este afişarea unui mesaj în bara de stare a navigatorului atunci când vizitatorul execută clic pe un link.

Pentru link-uri două sunt evenimentele care se execută în paralel: `MouseOver` şi `MouseOut`.

Evenimentul `MouseOver` se produce atunci când treceţi cu mouse-ul pe deasupra link-ului, iar evenimentul `MouseOut` se produce atunci când vă deplasaţi cu mouse-ul în zona exterioară legăturii. Cele două evenimente sunt utilizate pentru a modifica textul care se afişează în bara de stare a navigatorului.

Remarcă. În mod implicit, `URL`-ul unei legături se afişează în bara de stare a navigatorului în momentul în care treceţi cu mouse-ul pe deasupra legăturii.

Gestionarul de evenimente `onMouseOver` vă permite să afişaţi un mesaj personalizat în bara de stare a navigatorului, care corespunde unui link anume. Pentru aceasta, utilizaţi proprietatea JavaScript `status`.

Gestionarul de evenimente `onMouseOut` este utilizat pentru a şterge textul afişat în bara de stare în momentul în care mouse-ul este deplasat în exteriorul legăturii.

Remarci:

- ✓ Atunci când utilizaţi gestionarul de evenimente `onMouseOver` pentru a afişa mesaje în bara de stare, în momentul în care executaţi clic pe un link, mesajul personalizat înlocuieşte URL-ul afişat în mod implicit. `Asigura•i-v• c• mesajul dumneavoastr• este cel pu•in la fel de util ca URL-ul!`
- ✓ Gestionarul de evenimente `onMouseOver` se aplică la: `document; form; form.button; form.checkbox; form.fileupdate; form.password; form.select; form.option; form.reset; form.text; form.textarea; form.submit.`
- ✓ Gestionarul de evenimente `onMouseOut` se aplică la aceleaşi elemente ca şi `onMouseOver`.

Iată cum creăm un script care afişează următoarele mesaje în bara de stare:

- *comandaţi cartea (X)HTML;*
- *comandaţi cartea DREAMWEAVER MX;*
- *comandaţi cartea XML.*

Aceste mesaje vor dispare atunci când mouse-ul nu se va afla deasupra acestor mesaje, care reprezintă link-uri ce se afişează într-o listă simplă.

1. Definiţi o funcţie (`descriere`) care acceptă un parametru (`text`) pentru a afişa un mesaj în bara de stare (figura 6.11).

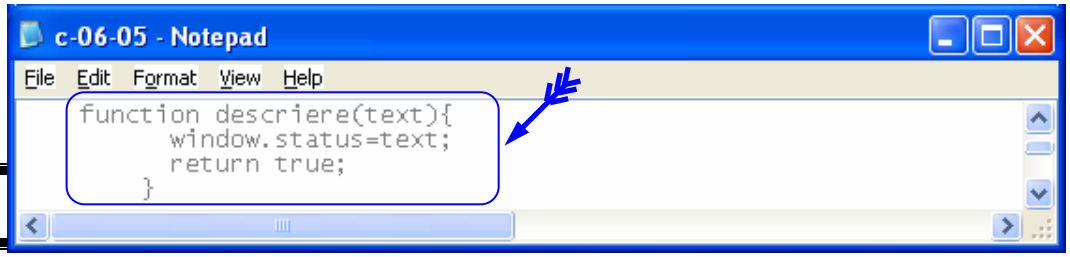


Figura 6.11

Remarci:

- ✓ Parametrul `text` conţine mesajul care se va afişa în bara de stare a navigatorului.
- ✓ Întrucât funcţia returnează `TRUE`, în bara de stare se va continua să se afişeze mesajul până când acesta va fi şters.

2. Definiţi o funcţie (`•tergere`) pentru a şterge mesajul, apelabilă din gestionarul de evenimente `onMouseOut` (figura 6.12).

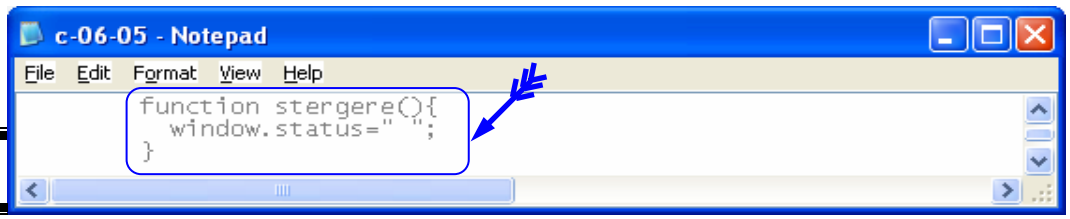


Figura 6.12

3. Completaţi script-ul, adăugând cunoscutele tag-uri `<script>` (figura 6.13).

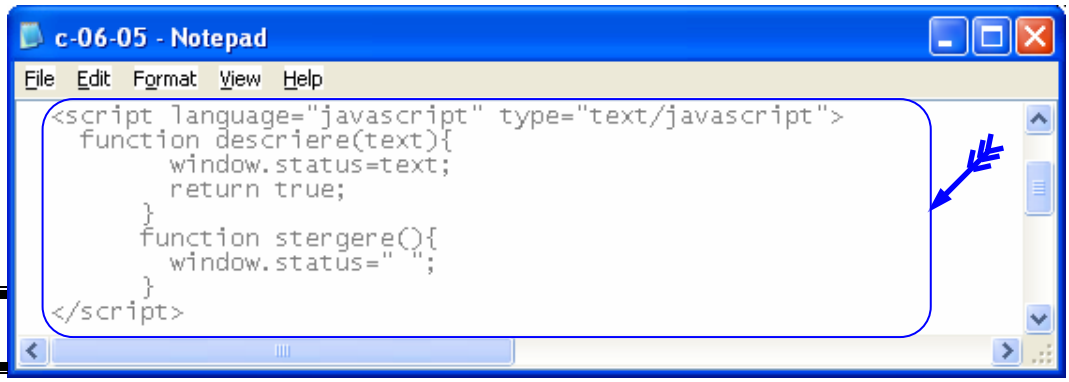


Figura 6.13

4. Completaţi structura (X)HTML a documentului adăugând: link-urile propriu-zise şi gestionarii de evenimente (`onMouseOver, onMouseOut`) care apelează cele două funcţii: `descriere, stergere` (figura 6.14).

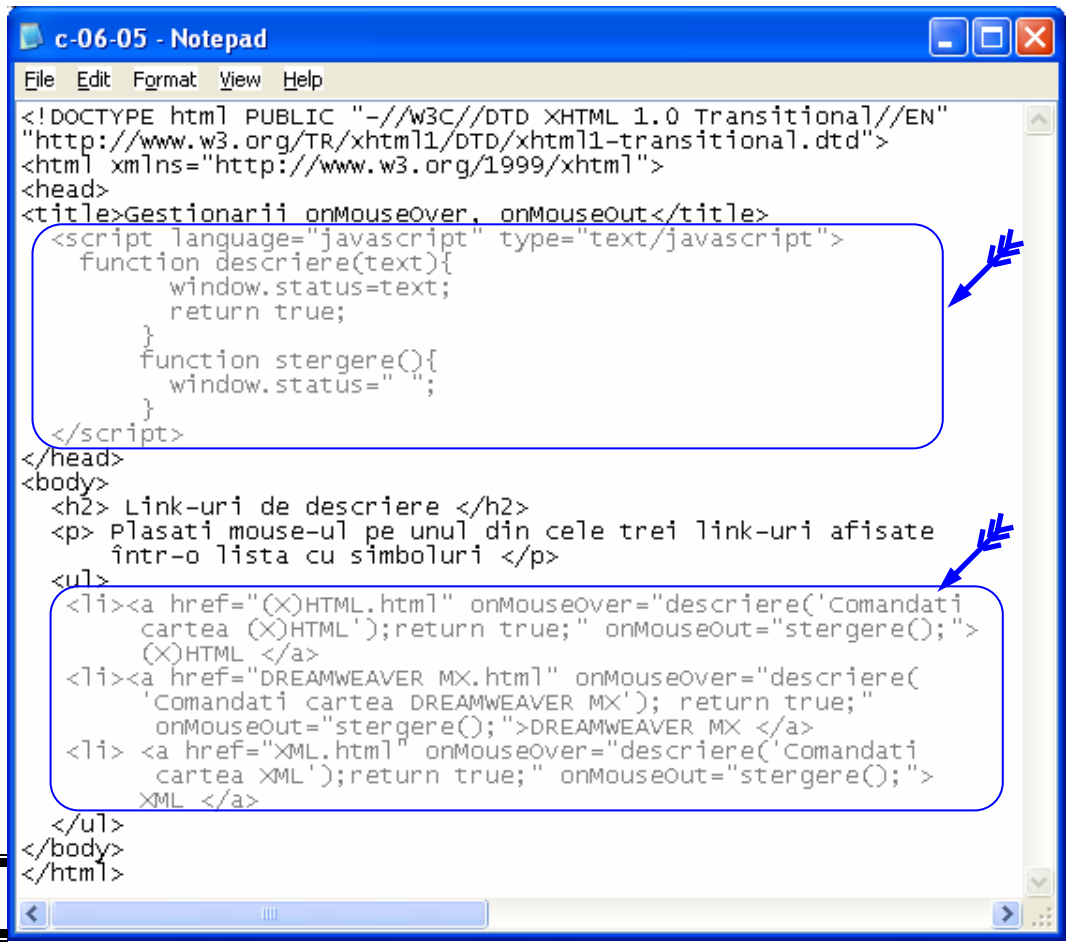


Figura 6.14

Remarci:

- ✓ Funcţiile (`descriere, stergere`) sunt definite în header-ul documentului.
- ✓ Fiecare link conţine un gestionar `onMouseOver` şi `onMouseOut` pentru a apela funcţiile de afişare şi ştergere a mesajelor în bara de stare.

5. Afişaţi pagina Web într-un navigator (figura 6.15).

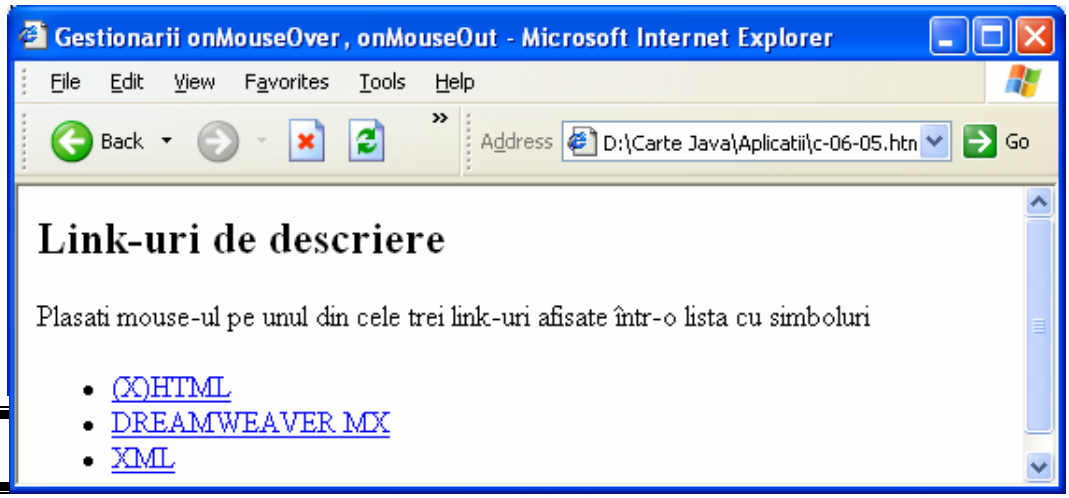


Figura 6.15

6. Testaţi script-ul (figura 6.16).

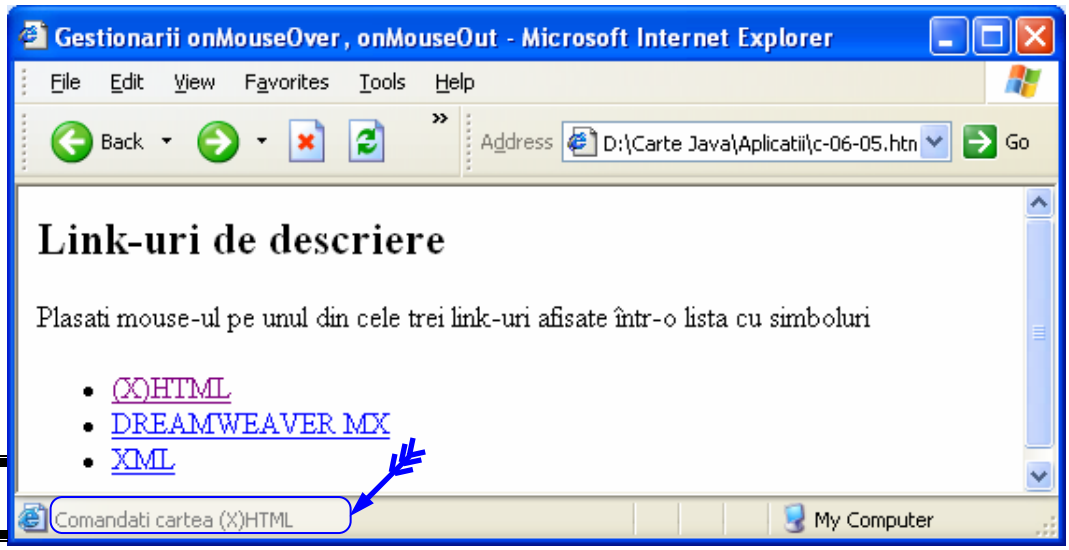


Figura 6.16

❑ Utilizaţi gestionarul de evenimente `onMouseOver` combinat cu metoda `setTimeOut`.

Indicaţie. JavaScript include o metodă `setTimeOut` care permite instalarea evenimentelor temporizator (cronometrate). De exemplu, puteţi utiliza metoda `setTimeOut` împreună cu `onMouseOver` pentru a afişa un mesaj în bara de stare a navigatorului într-o perioadă de timp predefinită şi a-l şterge după aceea.

În principiu, funcţia necesară pentru ştergerea textului din bara de stare trebuie să cuprindă două etape:

- ✓ **Etapa 1** – Definirea mesajului (textului) care urmează să se afişeze în bara de stare, ca un şir de caractere specific;
- ✓ **Etapa 2** – Aşteptarea unei perioade de timp predefinite şi apoi ştergerea mesajului afişat în bara de stare.

Iată cum creăm un script care afişează timp de 4 secunde în bara de stare a navigatorului mesajul: *"Singurătatea este mai grea decât orice boală!"* pe care apoi îl şterge!

1. Definiţi o funcţie (`descriere`) în care includeţi instrucţiunea `window.status` (figura 6.17).

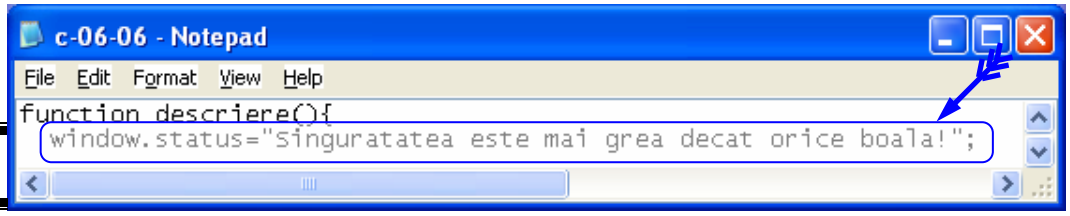


Figura 6.17

2. Adăugaţi metoda `setTimeOut` (figura 6.18).

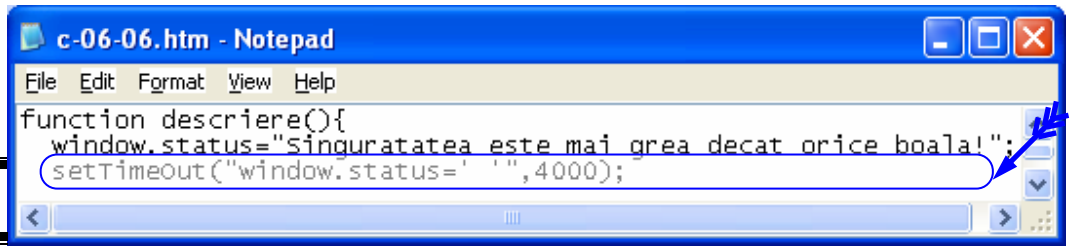


Figura 6.18

Remarci:

- ✓ Primul parametru defineşte starea ferestrei la o valoare `null` (");
- ✓ Al doilea parametru precizează perioada de timp predefinită – trebuie aşteptat 4 secunde.

3. Adăugaţi instrucţiunea `return true` (figura 6.19).

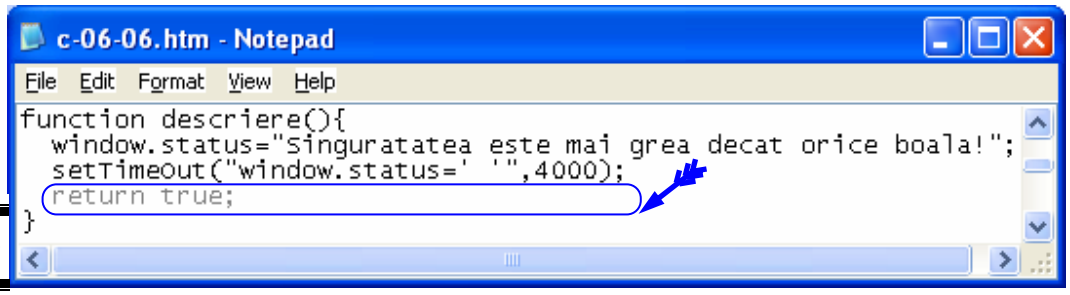


Figura 6.19

Remarcă. Instrucţiunea `return true` precizează navigatorului că funcţia `descriere` este pregătită pentru a interacţiona cu utilizatorul.

4. Completaţi script-ul şi plasaţi-l în antet-ul documentului (figura 6.20).

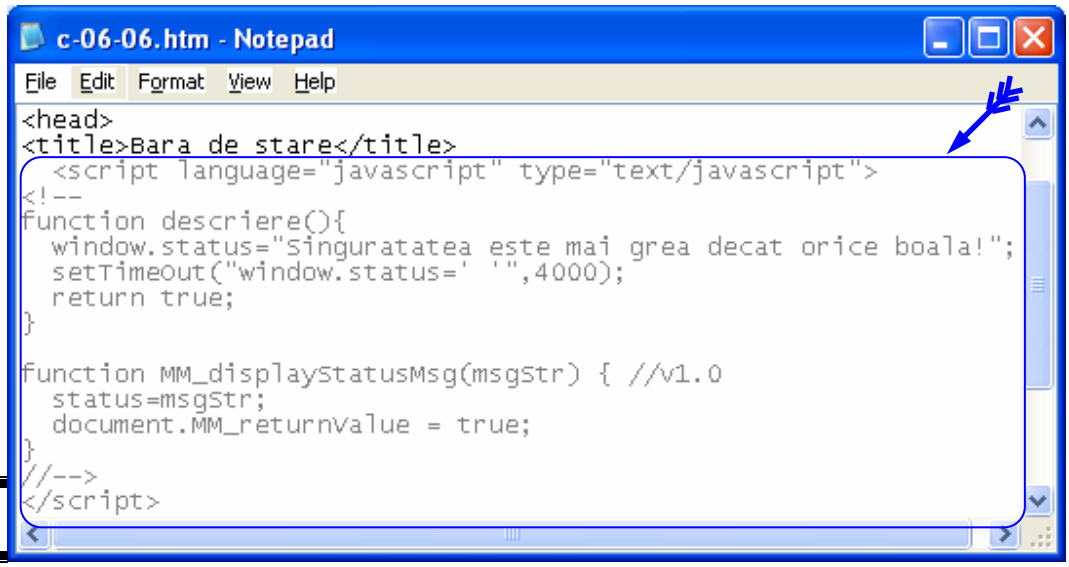


Figura 6.20

5. Adăugaţi în corpul documentului în tag-ul `<a>`, instrucţiunea `onMouseOver="descriere(); return true;"` (figura 6.21).

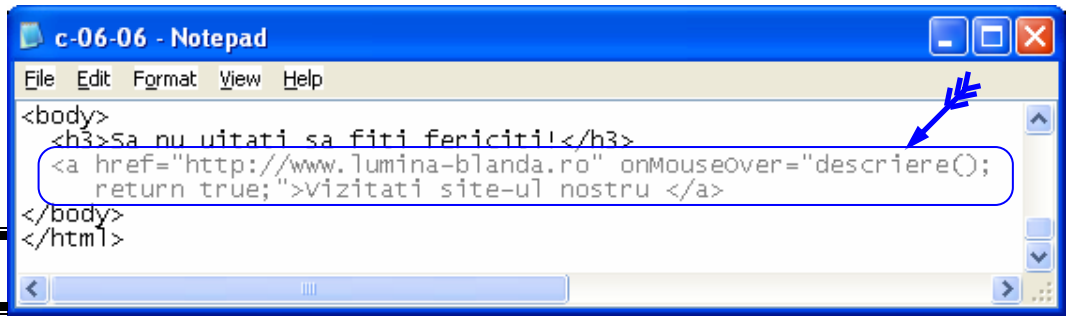


Figura 6.21

6. Afişaţi pagina Web într-un navigator (figura 6.22).

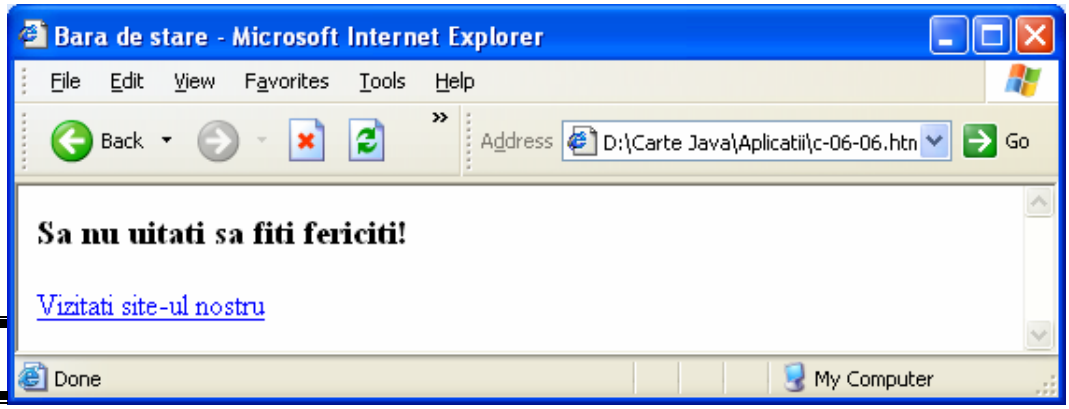


Figura 6.22

7. Testaţi script-ul (figura 6.23).

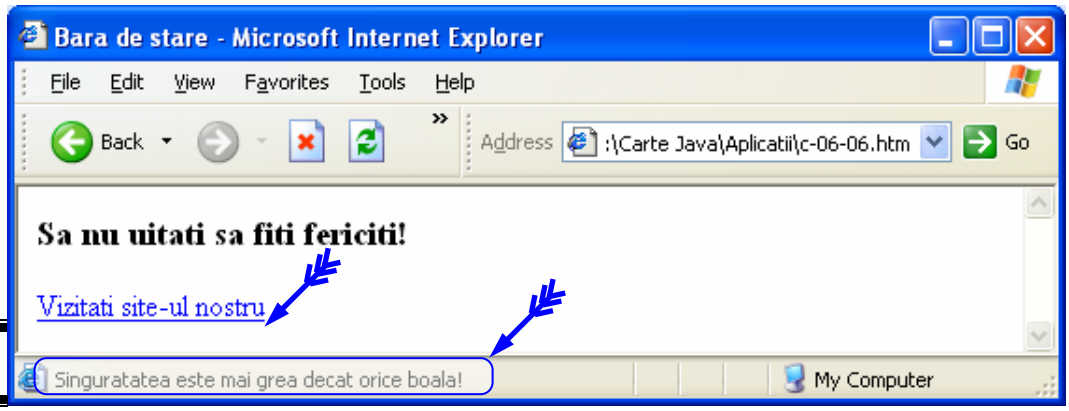


Figura 6.23

❑ Creaţi gestionari de evenimente dinamici.

Indicaţie. În loc să creaţi un gestionar de evenimente în interiorul unui document (X)HTML, puteţi crea o funcţie JavaScript care va face oficiul de gestionar de evenimente. Aveţi de asemenea posibilitatea să creaţi gestionari de evenimente condiţionali, de a-i activa sau a-i dezactiva, sau încă de a modifica funcţia (ca gestionar de evenimente) de o manieră dinamică.

Pentru aceasta este suficient de a crea funcţia şi de a o defini ca gestionar de evenimente.

Iată cum creaţi pentru un document gestionarul de evenimente dinamic `onMouseDown`.

1. Creaţi o funcţie (`clicmouse`) pe care o definiţi ca gestionar de evenimente (figura 6.24).

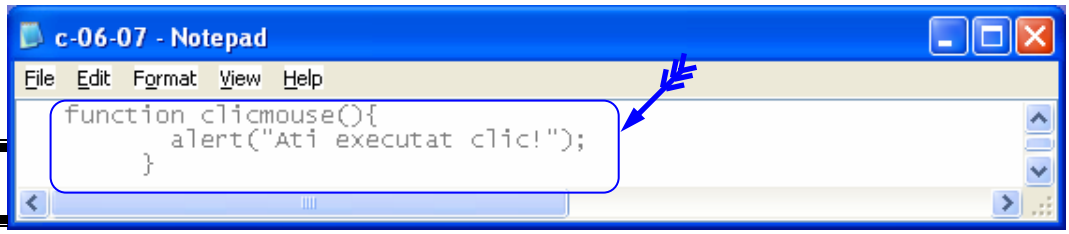


Figura 6.24

2. Atribuiţi funcţia (`clicmouse`) gestionarului de evenimente `onMouseDown` (figura 6.25).

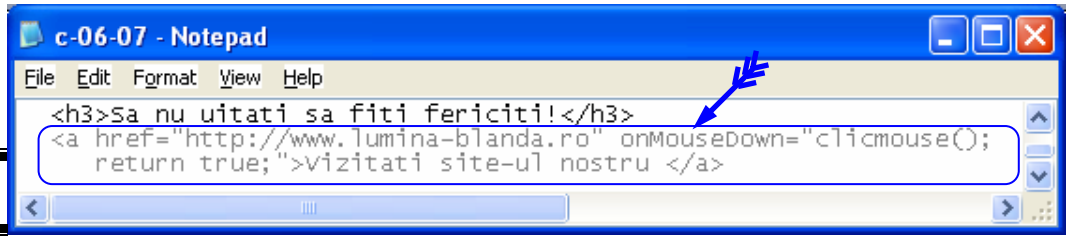


Figura 6.25

Remarcă. Gestionarii de evenimente dinamici sunt stocaţi sub formă de proprietăţi ale obiectului `Document` sau ale unui alt obiect căruia i se poate atribui un eveniment.

3. Completaţi script-ul şi structura (X)HTML a documentului (figura 6.26).

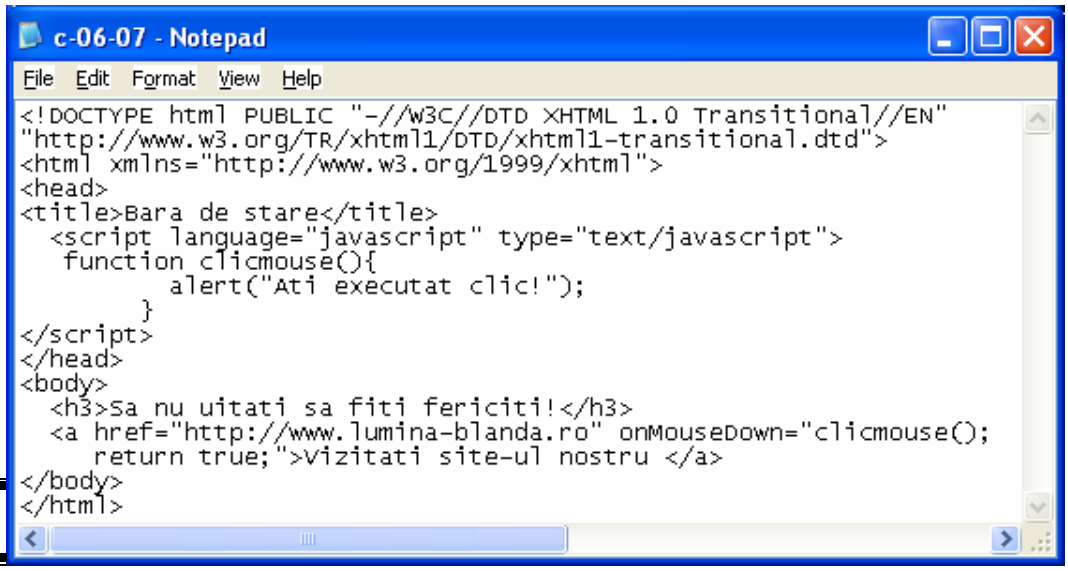


Figura 6.26

4. Afişaţi pagina Web într-un navigator (figura 6.27).

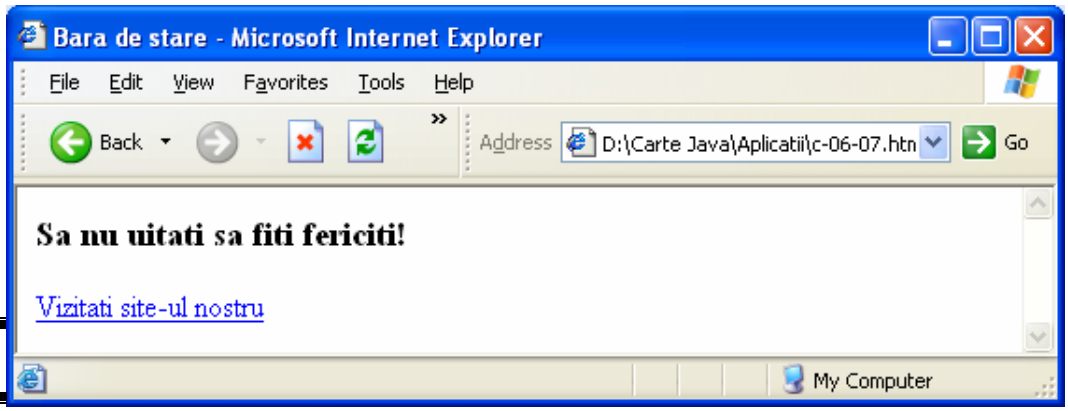


Figura 6.27

5. Testaţi script-ul (figura 6.28).

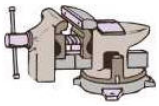
Figura 6.28

❑ Creaţi un script care afişează într-o casetă de dialog de avertizare care buton al mouse-ului a fost utilizat.

❑ Creaţi un script care afişează în bara de stare a navigatorului Internet Explorer, Netscape caracterele pe care le-aţi tastat.

Temă

Testați-vă cunoștințele



1. Ce este un gestionar de evenimente. Exemple.
2. Care sunt gestionarii de evenimente care permit detectarea unui clic pe un link?
3. Când se execută gestionarul de evenimente `onLoad` din tag-ul `<body>`?
4. Care este rolul proprietăţii `event.KeyCode` în Internet Explorer?
5. Când se apelează următorii gestionari de evenimente:
   - `onUnLoad;`
   - `onMouseOver;`
   - `onMouseOut;`
   - `onMouseDown;`
   - `onClick;`
   - `onDblClick;`
   - `onChange;`
   - `onMouseUp;`
   - `onKeyPress;`
   - `onSubmit;`
   - `onReset;`

- `onSelect`.

6. Comentaţi următorul script (figura 6.29).

```
<script language="javascript" type="text/javascript">
        function test (text){
                window.status=text;
                return true;
        }
</script>
```

Figura 6.29

7. Comentaţi următoarea instrucţiune JavaScript:

   <b onDblClick=”alert (’Nu este nimic întâmplător!’)”> Test

Vizitați site-urile

- ✓ javascript.internet.com
- ✓ www.javascripts.com
- ✓ www.javascript.com
- ✓ comp.lang.javascript
- ✓ livesoftware.javascript.developer
- ✓ livesoftware.javascript.examples
- ✓ de.comp.lang.javascript

Conversația 7

Obiectele navigatorului

În această conversație:

- *Document Object Model (DOM)*
- *Obiectul Window. Aplicații*
- *Obiecte de nivelul 1*
- *Obiecte de nivelul al doilea*
- *Obiecte de nivelul al treilea*
- *Obiecte de nivelul al patrulea*
- *Obiectul Navigator. Aplicații*
- *Temă*

Document Object Model (DOM)

În conversaţiile anterioare am acordat atenţie limbajului de programare JavaScript şi obiectelor interne ale acestuia (`Arguments`, `Array`, `Boolean`, `Date`, `Function`, `Math`, `Number`, `Object`, `RegExp`, `String`, `This`). Obiectele pe care le veţi utiliza cel mai des, pe parte de client sunt cele care aparţin **DOM**-ului (**Document Object Model**), cu ajutorul cărora script-urile dumneavoastră vor putea manipula paginile Web, ferestrele şi documentele. Obiectivul acestui model (DOM) este acela de a vă oferi o interfaţă (între două feţe există ... o interfaţă!) simplă şi coerentă între programele JavaScript şi navigatorul Web.

DOM-ul defineşte obiectele disponibile, proprietăţile, metodele şi evenimentele acestora. Cu o parte din obiectele DOM-ului aţi făcut deja cunoştinţă.

În afară de aceste obiecte, DOM-ul mai conţine şi alte obiecte ierarhizate (structură arborescentă), obiectul `Window` aflându-se în vârful arborescenţei.

Ierarhia obiectelor

Când examinaţi îndeaproape ierarhia generală a obiectelor în JavaScript, puteţi vedea că cele mai multe obiecte sunt fie pe parte de client sau pe parte de server.

În această lucrare vom examina obiectele numai pe parte de client şi vom prezenta proprietăţile şi metodele lor.

Remarcă. Majoritatea obiectelor JavaScript sunt reprezentări ca obiecte ale tag-urilor (X)HTML.

În figura 7.1 sunt prezentate obiectele pe parte de client şi tag-urile (X)HTML corespunzătoare.

Obiect JavaScript	***Tag (X)HTML corespondent***
`Button`	<input type="button" />
`Checkbox`	<input type="checkbox" />
`Hidden`	<input type="hidden" />
`Fileupload`	<input type="file" />
`Password`	<input type="password" />
`Radio`	<input type="radio" />
`Reset`	<input type="reset" />
`Select`	<select>
`Frame`	<frame>
`Document`	<body>
`Layer`	<layer> sau <ilayer>
`Link`	<a href=" " />
`Image`	<img />
`Area`	<map>
`Anchor`	<a name=" " />
`Applet`	<applet>
`Plugin`	<embed>
`Form`	<form>
`Submit`	<input type="submit" />
`Text`	<input type="text" />
`Textarea`	<textarea>
`Option`	<option>

Figura 7.1

Pe măsură ce veţi examina aceste obiecte veţi vedea modalităţile diverse în care ele sunt prezentate utilizatorilor şi programatorilor în limbajul JavaScript.

Lista obiectelor JavaScript este prezentată în ordine alfabetică în figura 7.2, iar ierarhia obiectelor este ilustrată în figura 7.3.

button	checkbox	document
event	fileUpload	form
hidden	history	location
MimeType	navigator	obiecte în general
option	Option()	password
plugins	radio	reset
screen	select	submit
text	textarea	window

Figura 7.2

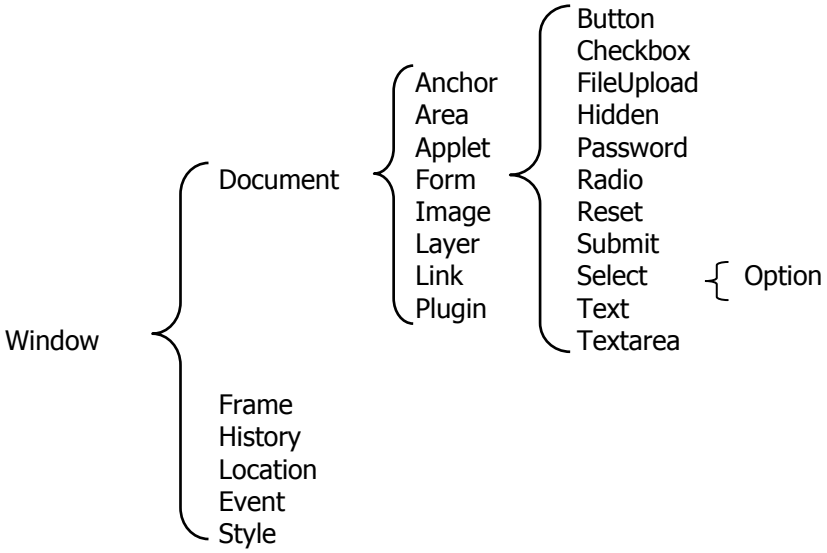


Figura 7.3

Obiectele sunt ierarhizate după cum urmează:

- ✓ Obiectul `Window`, de nivelul cel mai înalt (nivelul 0), pe parte de client;
- ✓ Obiecte de nivelul 1: `Document`, `Frame`, `History`, `Location`, `Event`, `Style`;
- ✓ Obiecte de nivelul al doilea: `Anchor`, `Area`, `Applet`, `Form`, `Image`, `Layer`, `Link`, `Plugin`;
- ✓ Obiecte de nivelul al treilea: `Button`, `Checkbox`, `FileUpload`, `Hidden`, `Password`, `Radio`, `Reset`, `Submit`, `Select`, `Text`, `Textarea`;
- ✓ Obiecte de nivelul al patrulea: `Option`.

Remarci:

- ✓ Obiectele limbajului JavaScript pe parte de client reprezintă instrumente importante cu ajutorul cărora puteţi genera script-uri.
- ✓ În cea mai mare parte modelul obiectului este constituit din elemente (X)HTML care sunt „transpuse în obiecte”.
- ✓ Dacă până acum aţi programat în limbajului (X)HTML, din acest moment trebuie să abordaţi elementele (X)HTML ca fiind nu tag-uri, ci obiecte.

- ✓ JavaScript utilizează o ierarhie de obiecte: părinte-fiu, cunoscute sub numele de `Document Object Model` (DOM). Aceste obiecte, organizate într-o structură arborescentă reprezintă conţinutul şi componentele unui document Web.

Iată un exemplu simplu (figura 7.4) care ilustrează modul în care un fişier (X)HTML stabileşte corespondenţa cu DOM-ul navigatorului.

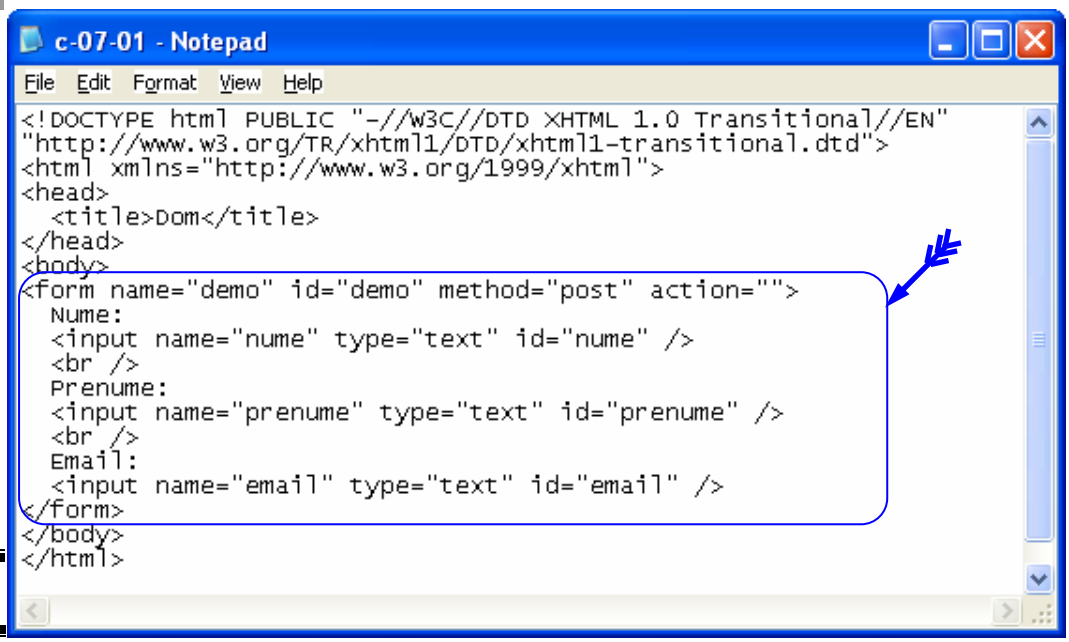


Figura 7.4

În cursul încărcării fişierului XHTML, navigatorul Web foloseşte un ansamblu complet de obiecte ale DOM-ului pentru a le reprezenta (vezi figura 7.5).

Obiect	***Descriere***
Obiectul Window	Reprezintă navigatorul Web
Obiectul Document	Reprezintă fişierul XHTML
Un obiect Form cu numele „demo”	Reprezintă tag-ul `<form>` definit în fişierul XHTML
Trei obiecte input cu numele: „nume”, „prenume”, „emai”l	Reprezintă zonele de text definite în formular

Figura 7.5

Remarcă. Evident, mai sunt create şi alte obiecte, ca de exemplu matricea formularelor `forms[]`.

În figura 7.6 se prezintă modul de imbricare al obiectelor create.

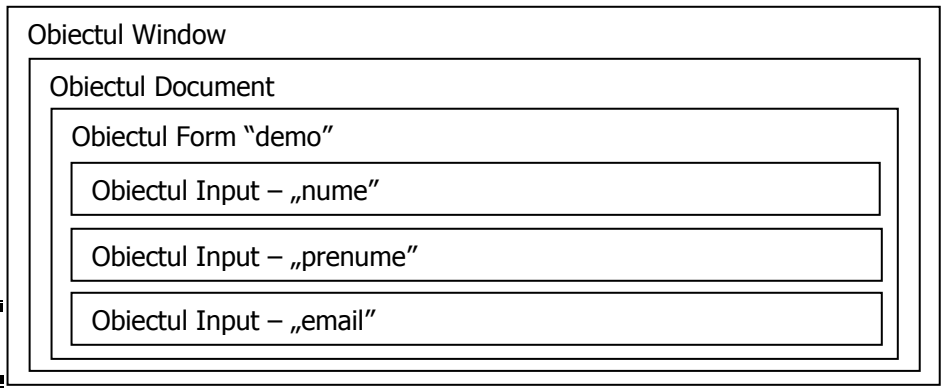


Figura 7.6

Istoricul DOM-ului

Obiectele navigatorului sunt exterioare limbajului JavaScript dar sunt recunoscute de către navigatoare. Din păcate există încă diferenţe între navigatoare, care nu de puţine ori se manifestă în mod supărător.

După Netscape 3.0 şi Internet Explorer 4.0 toate obiectele de bază sunt luate în considerare de către cele două browser-e, iar noile norme DOM sunt recunoscute de ultimele versiuni Netscape şi Internet Explorer.

W3C (World Wide Web Consortium) a pus recent la punct norma DOM de nivel 1. Această normă nu conţine decât obiectele de bază, toate obiectele acoperind integral componentele unui document (X)HTML. O normă de nivel 2 este în curs de apariţie. Ierarhia obiectelor de care noi vorbim în această conversaţie este cunoscută sub numele de DOM 0 iar obiectele sunt incluse în norma DOM 1.

Remarcă. Obiectele DOM de nivel 1 şi 2 permit modificarea unei pagini Web în timp real, după încărcarea sa (Dinamic HTML).

Obiectul Window

Obiectul `Window` de nivelul cel mai înalt reprezintă fereastra navigatorului în care este afişat obiectul `Document`. Fiecare fereastră a navigatorului şi fiecare cadru au propriul lor obiect `Window`.

Fişa obiectului `Window` este prezentată în figura 7.7.

Remarcă. Nu uitaţi că un obiect *`Window`* şi proprietăţile sale pot fi atribuite unei variabile JavaScript ca orice alt obiect.

Spre deosebire de alte obiecte care pot să fie prezente sau nu, obiectul `Window` există permanent.

Remarci:

- ✓ Mai multe obiecte `window` pot exista în acelaşi timp, fiecare reprezentând o fereastră a navigatorului deschisă.
- ✓ Cadrele (*`frames`*, în limba engleză) sunt de asemenea reprezentate prin obiecte Window (vezi Conversaţia 10).
- ✓ Straturile (*`layers`*, în limba engleză) care permit modificarea în mod dinamic a conţinutului unui document Web sunt analoage obiectelor `Window` (vezi Conversaţia 10).

Fişa obiectului Window

Subobiecte:	`document, event, history, location, navigator, style`
Proprietăţi:	`content, clientinformation, clipboard, closed, defaultstatus, dialogArguments, dialogHeight, dialogLeft, dialogTop, dialogWidth, document, event, frames[], history, innerHeight, innerWidth, length, name, navigator, offscreenBuffering, opener, outerHeight, outerWidth, pageXOFFset, pageYOffset, parent, returnValue, screen, screenLeft, screenTop, screenX, screenY, self, status, style, top, window`
Metode:	`alert(), back(), blur(), clearInterval(), clearTimeout(), close(), confirm(), createPopup(), execScript(), focus(), forward(), home(), moveBy(), moveTo(), navigate(), open(), print(), prompt(), resizeBy(), resizeTo(), scroll(), scrollBy(), scrollTo(), setActive(), setInterval(), setTimeout(), showHelp(), showModaldialog(), showModelessDialog(), stop()`
Gestionarii de evenimente:	`onActivate, onAfterPrint, onBeforeActivate, onBeforePrint, onBeforeUnload, onBlur, onControlSelect, onDeActivate, onDragDrop, onError, onFocus, onHelp, onLoad, onMouseMove, onMove, onMoveEnd, onMoveStart, onResize, onResizeEnd, onResizeStart, onScroll, onUnload`

Figura 7.7

Proprietăţile fundamentale ale obiectului Window

Cele mai importante proprietăţi ale obiectului `Window` sunt prezentate în detaliu în figura 7.8.

	Proprietate	***Sintaxă***
	`closed`	`window.closed`
	Conţine valoarea `true` (fereastra este închisă) sau `false` (fereastra este deschisă).	
	`defaultStatus`	`window.defaultStatus`
	Conţine un şir de caractere reprezentând textul afişat în mod implicit în bara de stare. El rămâne valabil atâta timp cât conţinutul ferestrei nu se schimbă.	
Exemplu:	<script>window.defaultStatus="La mulţi ani împreună!" </script>	

Figura 7.8

`document`	`window.document`
Proprietatea `document` corespunde obiectului `Document`.	
`frames[]`	`window.frames[]`
Conţine un obiect `window` pentru fiecare cadru al site-ului Web.	
`history`	`window.history`
Un obiect care conţine lista de site-uri (istoricul) Web vizitate (vezi obiectul `History`).	
`length`	`window.length`
Numărul de cadre conţinute în fereastră.	
`name`	`window.name`
Numele obiectului `Window`. Acest nume este diferit de numele variabilei JavaScript care conţine obiectul `Window`.	
Exemplu:	… var catwindow=window.open(); catwindow.name="cat"; …
`navigator`	`window.navigator`
Obiectul `Navigator` conţine informaţii cu privire la navigator.	
Exemplu:	… for (x in navigator) document.write(x+"="+navigator[x]+" "; …
`opener`	`window.opener`
Corespunde ferestrei în care a fost creată o altă fereastră (prin metoda `window.open()`. Marea majoritate a proprietăţilor şi metodelor obiectului `window` pot fi aplicate proprietăţii `opener`).	
`outerHeight`	`window.outerHeight`
Înălţimea (în pixeli) documentului afişat în fereastră (incluzând bara de meniuri, bara de instrumente, bara de stare).	
`outerWidth`	`window.outerWidth`
Lăţimea (în pixeli) documentului afişat în fereastră (incluzând bara de meniuri, bara de instrumente, bara de stare).	
`parent`	`window.parent`
Referinţă la fereastra/cadrul părinte al ferestrei/cadrului curent.	
`screen`	`window.screen`
Conţine informaţii privind configuraţia ecranului de pe monitorul clientului.	
`self`	`window.self`

Figura 7.8
(continuare)

	Referinţa la fereastra curentă. Identică cu `window` (`window.window`).
Exemplu:	window.close() *identic cu:* window.self.close
`status`	`window.status`
	Conţine un şir de caractere care va fi temporar afişat în bara de stare a navigatorului.
`top`	`window.top`
	Referinţa către fereastra navigatorului situată în prim plan.
`window`	`window.window`
	Identic cu `window.self`.

Figura 7.8 *(continuare)*

Metodele fundamentale ale obiectului Window

Metodele fundamentale ale obiectului `Window` sunt prezentate în detaliu în figura 7.9.

Metode	***Sintaxă***
`alert()`	`window.alert ("Mesaj")` `alert ("Mesaj")`
Afişează o casetă de dialog care conţine un buton OK şi mesajul specificat. Apelarea metodei opreşte execuţia programului JavaScript până când utilizatorul execută clic pe butonul OK pentru a continua. Metoda `alert()` este folosită şi ca instrument de depanare a script-urilor (vezi Conversaţia 11).	
`back()`	`window.back()` `back()`
Retur la pagina precedentă afişată în fereastră (vezi obiectul `History`).	
`blur()`	`window.blur()` `blur()`
Retrage focus-ul ferestrei curente.	
Exemplu: <script>window.blur()</script>	
`close()`	`window.close()` `close()`
Închide fereastra curentă (după confirmarea de către utilizator).	
`confirm()`	`window.confirm("Mesaj")` `confirm("Mesaj")`
Afişează o casetă de dialog care conţine mesajul specificat, un buton OK şi un buton de anulare. Returnează `TRUE` dacă utilizatorul execută clic pe butonul OK şi `FALSE` dacă utilizatorul execută clic pe butonul de anulare.	
Exemplu: <script> a=window.confirm("Deschide o nouă fereastră");	

Figura 7.9

```
if (a==true) {
  window.open(”nou.htm”);
  document.write(”Noua fereastră este deschisă”);
}
if (a==false) {
  window.open(”nou.htm”);
  document.write(”Noua fereastră nu va fi deschisă”);
}
</script>
```

`execScript()` `window.execScript(`*`Expresie,Limbaj`*`)`
`execScript(`*`Expresie,Limbaj`*`)`

Execută expresia menţionată ca şi când ar fi un script, în limbajul indicat.

Exemplu:
```
<script>
window.execScript(alert(”Fatalitate”),”JavaScript”);
</script>
```

`focus()` `window.focus()`
`focus()`

Transmite focus-ul ferestrei curente.

`forward()` `window.forward()`
`forward()`

Obligă navigatorul să încarce documentul următor în istoric.

`home()` `window.home()`
`home()`

Navigatorul încarcă pagina definită ca pagină de index de către utilizator.

`moveBy()` `window.moveBy(`*`x,y`*`)`
`moveBy(`*`x,y`*`)`

Deplasează fereastra cu *x* pixeli orizontal şi cu *y* pixeli vertical. Cele două valori pot fi pozitive sau negative.

Exemplu:
```
<script>
window.moveBy(100,200);
</script>
```

`moveTo()` `window.moveTo(`*`x,y`*`)`
`moveTo(`*`x,y`*`)`

Deplasează fereastra la poziţia *x* (orizontal) şi *y* (vertical). Cele două valori au ca origine colţul superior stânga al ecranului.

Exemplu:
```
<script>
window.moveTo(0,0);
</script>
```

Figura 7.9 *(continuare)*

`open()` `window.open(`*`URL,Nume,Atribute,Înlocuieşte`*`)`

open(*URL*,*Nume*,*Atribute*, *Înlocuieşte*)

Deschide o nouă fereastră a navigatorului şi returnează un obiect `Window` pentru a o reprezenta. Dacă apelaţi metoda `open()` fără parametri, veţi obţine o fereastră vidă. Pentru a vedea ceva mai interesant, transmiteţi următorii patru parametri respectând secvenţa de mai jos:

- *URL* – adresa Web a paginii Web care doriţi să se încarce automat în noua fereastră.
- *Nume* – şir de caractere care va fi plasat în proprietatea `window.name` a noii ferestre.
- *Atribute* – `width, height, menubar, resizable, scrollbars, status, directories`.
- *Înlocuieşte* este o valoare logică (`true`, dacă istoricul ferestrei este şters).

Metoda returnează o referinţă la noua fereastră. Atributele pot lua valoarea `adev•rat` sub forma `true, yes` sau `1` sau valoarea `false` sub forma `false`, `no` sau `0`.

Semnificaţia atributelor este următoarea:

- ✓ `toolbar` – afişează/ascunde bara de instrumente a browser-ului
- ✓ `location` – prezenţa barei de adrese
- ✓ `directories` – afişează/ascunde o bară de instrumente (Netscape) secundară cu butoane familiare
- ✓ `status` – afişează/ascunde bara de stare a navigatorului
- ✓ `menubar` – afişează/ascunde bara de meniuri a navigatorului
- ✓ `scrollbars` – afişează/ascunde barele de derulare pe verticală şi orizontală ale browser-ului
- ✓ `resizable` – permite/interzice redimensionarea ferestrei de browser
- ✓ `width` – lăţimea în pixeli a ferestrei
- ✓ `height` – înălţimea în pixeli a ferestrei

Exemplu:

```
<script>
newfereastra=window.open("test.htm","titlu","toolbar'no,
location=no, directories=no, status=no, menubar=no,
scrollbars=no, resizable=no, width=100, height=100");
</script>
```

`print()` `window.print()` `print()`

Imprimă conţinutul ferestrei.

`prompt()` `window.prompt("Mesaj",ConţinutImplicit)` `prompt("Mesaj",ConţinutImplicit)`

Afişează o casetă de dialog care conţine mesajul specificat, o zonă de text pe care o completează utilizatorul, butonul OK şi butonul de anulare. Al doilea argument este facultativ. Răspunsul implicit specificat în `prompt()` este afişat în zona de text. Utilizatorul poate introduce o altă valoare sau poate executa clic pe butonul OK pentru a accepta răspunsul implicit. Dacă el execută clic pe butonul OK, valoarea conţinută în zona de text este returnată script-ului; dacă el execută clic pe butonul de anulare, atunci valoarea `null` este returnată script-ului. Valoarea `null` reprezintă "nimic" sau un obiect vid.

Exemplu:

```
<script>
var raspuns=prompt("Care este culoarea preferata?", "cepia");
if (raspuns==null){
```

Figura 7.9 *(continuare)*

```
    alert("Ati apasat butonul Cancel");
  }
  else{
    alert("Raspunsul dumneavoastră este "+raspuns);
  }
</script>
```

`resizeTo()` — `window.resizeTo(Lungime,Lăţime)` / `resizeTo(Lungime,Lăţime)`

Redimesionează fereastra navigatorului la *Lungime* şi *Lăţime* specificate (în pixeli).

Exemplu:

```
<script>
self.resizeTo(50,325);
</script>
```

`scroll()` — `window.scroll(x,y)` / `scroll(x,y)`

Face să defileze conţinutul unei ferestre până la poziţia specificată (*x* şi *y*) în raport cu colţul din stânga sus al ferestrei.

Remarcă. Metoda este depăşită.

`scrollTo()` — `window.scrollTo(x,y)` / `scrollTo(x,y)`

Face să defileze conţinutul unei ferestre până la poziţia *x* (lungime), *y* (lăţime) în raport cu colţul din stânga sus al ferestrei.

Exemplu:

```
<script>
window.scrollTo(0,100);
</script>
```

Figura 7.9 *(continuare)*

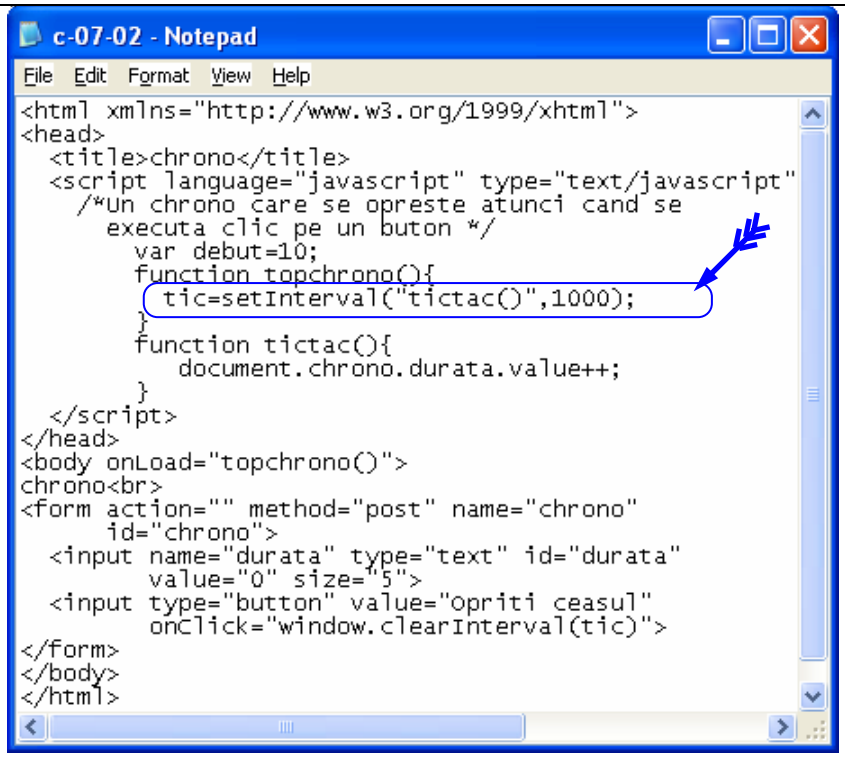
`setInterval()` — `window.setInterval(Expresie, Interval,Argumente)` / `setInterval(Expresie, Interval,Argumente)`

Execută *Expresie* în intervalele fixate. *Expresie* poate fi o instrucţiune JavaScript sau apelul unei funcţii. *Interval* este prezentat în milisecunde. *Argumente* sunt transmise funcţiei apelate.

Remarci:

- ✓Metoda `clearInterval()` anulează efectul metodei `setInterval()`.
- ✓Metoda `setInterval()` este aproape identică cu metoda `setTimeout()`. Principala diferenţă între cele două metode (vezi `setTimeout()`) este aceea că parametrul *Expresie,* care poate fi o instrucţiune sau apelul unei funcţii executată fără sfârşit.

Exemplu:



`setTimeout()` `window.setTimeOut(`*`Expresie, Durata,Argumente`*`)`
`setTimeOut(`*`Expresie, Durata,Argumente`*`)`

Execută *Expresie* după *Durata* (în milisecunde). *Expresie* poate fi apelul unei funcţii. Argumentele (facultative) sunt transmise funcţiei apelate.

Remarcă. Metoda `clearTimeOut()` anulează efectul metodei `setTimeOut()`.

Exemplu:

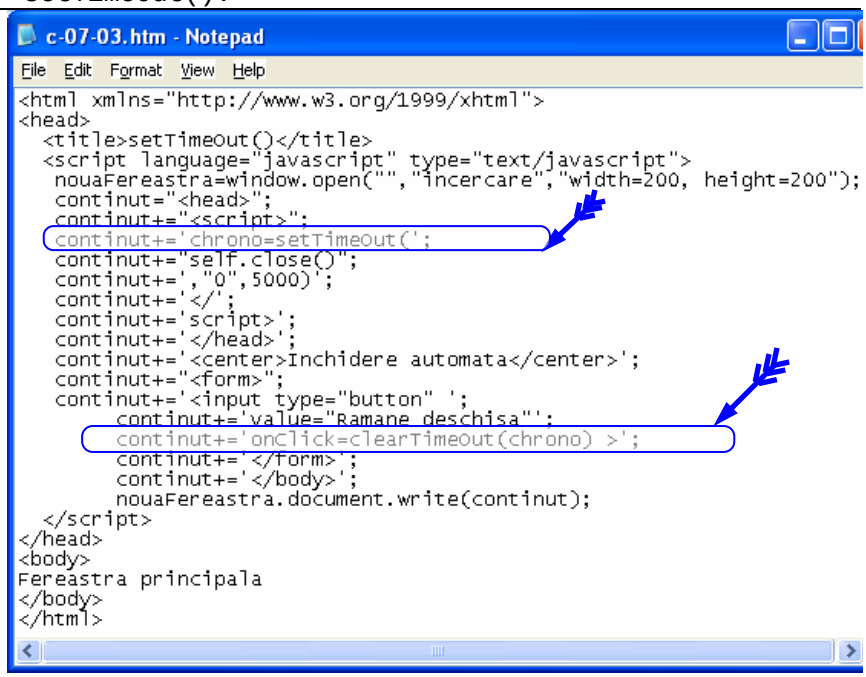


Figura 7.9
(continuare)

Rezultatele execuţiei script-ului.

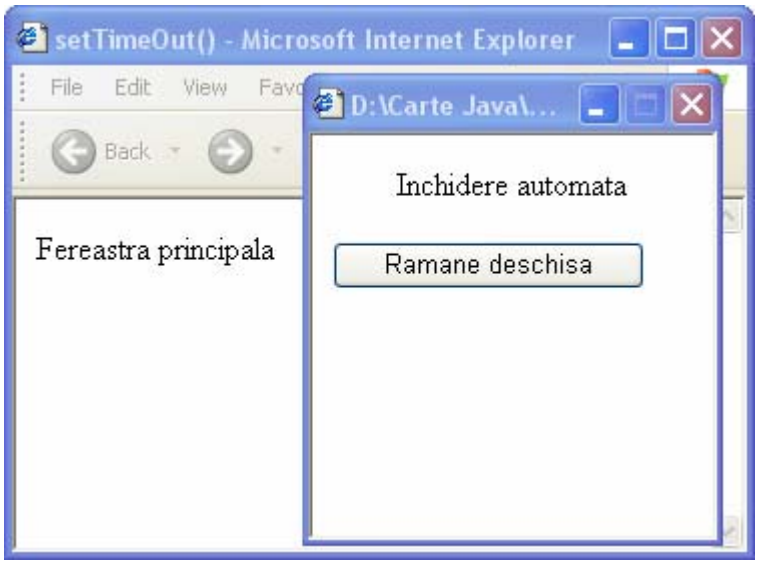


Figura 7.9
(continuare)

Aplicaţii ale obiectului Window

Una din aplicaţiile cele mai interesante ale obiectului `Window` este crearea şi închiderea unei noi ferestre. Vă prezentăm în cele ce urmează câteva aplicaţii simple, reprezentative pentru metodele şi proprietăţile obiectului `Window`.

❑ Creaţi un document (X)HTML care conţine un script (simplu) ce utilizează butoane pentru crearea şi închiderea unei ferestre.

Indicaţie. Folosiţi metodele: `open()` şi `close()` prezentate în figura 7.9.

În figura 7.10 este prezentat documentul (X)HTML complet.

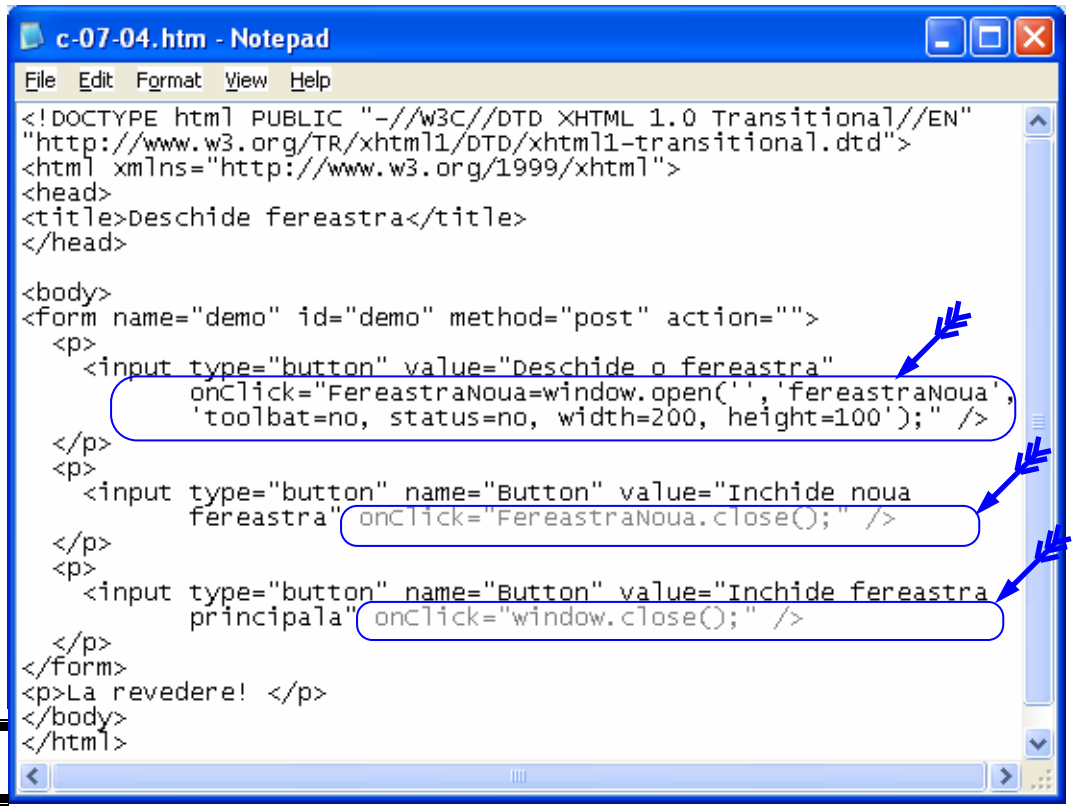


Figura 7.10

Remarci:

- ✓ Documentul XHTML permite deschiderea unei noi ferestre executând clic pe un buton (primul buton).
- ✓ A doua fereastră este foarte mică pentru a se putea distinge de prima fereastră.
- ✓ Al doilea buton permite închiderea acestei ferestre.
- ✓ Al treilea buton închide fereastra principală a navigatorului, după acordul utilizatorului.
- ✓ Acest document face apel la gestionarii de evenimente `onClick` (câte unul pentru fiecare buton).

În figura 7.11 este prezentată pagina Web afişată în Internet Explorer, cu noua fereastră deschisă.

Figura 7.11

❑ Creaţi un document (X)HTML care conţine un script (simplu) pentru deplasarea şi modificarea ferestrelor.

În figura 7.12 este prezentat documentul XHTML complet.

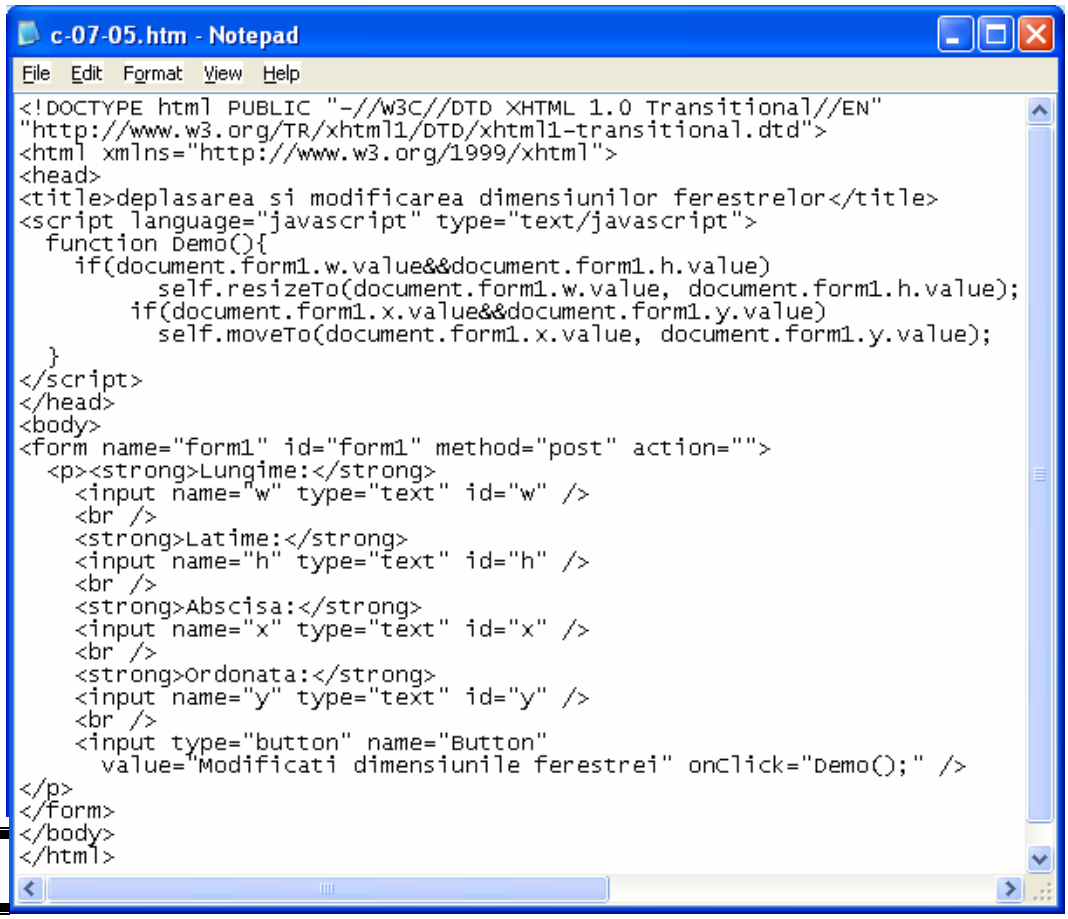


Figura 7.12

Remarci:

- ✓ Funcţia `Demo()` este apelată ca un gestionar de evenimente atunci când executaţi clic pe butonul „modificaţi dimensiunile ferestrei”.
- ✓ Funcţia `Demo()` verifică dacă aţi introdus valori pentru lungime şi lăţime. Dacă aceste valori au fost introduse se apelează metoda `self.resizeTo()` pentru a modifica dimensiunile ferestrei curente.
- ✓ Funcţia `Demo()` verifică dacă aţi introdus valori pentru abscisă(x) şi ordonată(y). Dacă aceste valori au fost introduse se apelează metoda `self.moveTo()` pentru a deplasa fereastra.

❑ Creaţi un mic dicţionar pentru vizitatorii site-ului dumneavoastră. Afişaţi în bleu cuvintele pe care le definiţi. Ori de câte ori utilizatorul va executa un clic pe unul din aceste cuvinte, se va deschide o fereastră în care se afişează o scurtă definiţie a cuvântului selectat (figura 7.13). Personalizaţi script-ul creat.

Figura 7.13

❑ Scrieţi un program JavaScript care ajustează în mod automat dimensiunea caracterelor unui text la dimensiunile ferestrei. Cu cât fereastra este mai mare, cu atât caracterele sunt mai mari. Punerea în pagină va fi asigurată în toate circumstanţele. Definiţi dimensiunea de bază a caracterelor pentru dimensiuni precise ale ferestrei. Dimensiunea caracterelor se va recalcula la fiecare redimensionare a ferestrei.

Script-ul redefineşte stilul caracterelor pentru întreg documentul de fiecare dată când fereastra îşi schimbă dimensiunile (figura 7.14, figura 7.15).

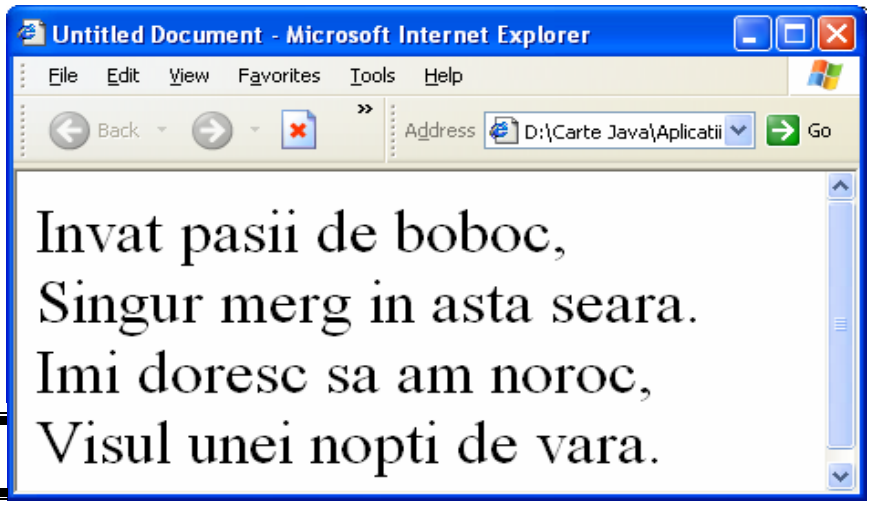


Figura 7.14

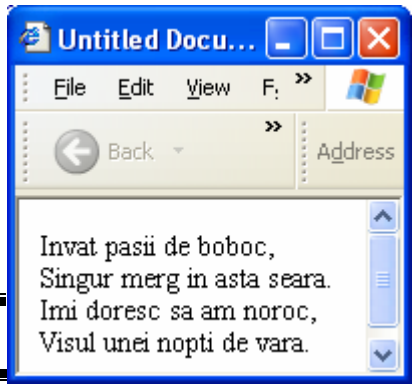


Figura 7.15

❑ Comentaţi următorul document XHTML care conţine un script pentru actualizarea periodică a conţinutului unei pagini Web (figura 7.16).

Indicaţie. Revedeţi metoda `setTimeOut()` prezentată în figura 7.9.

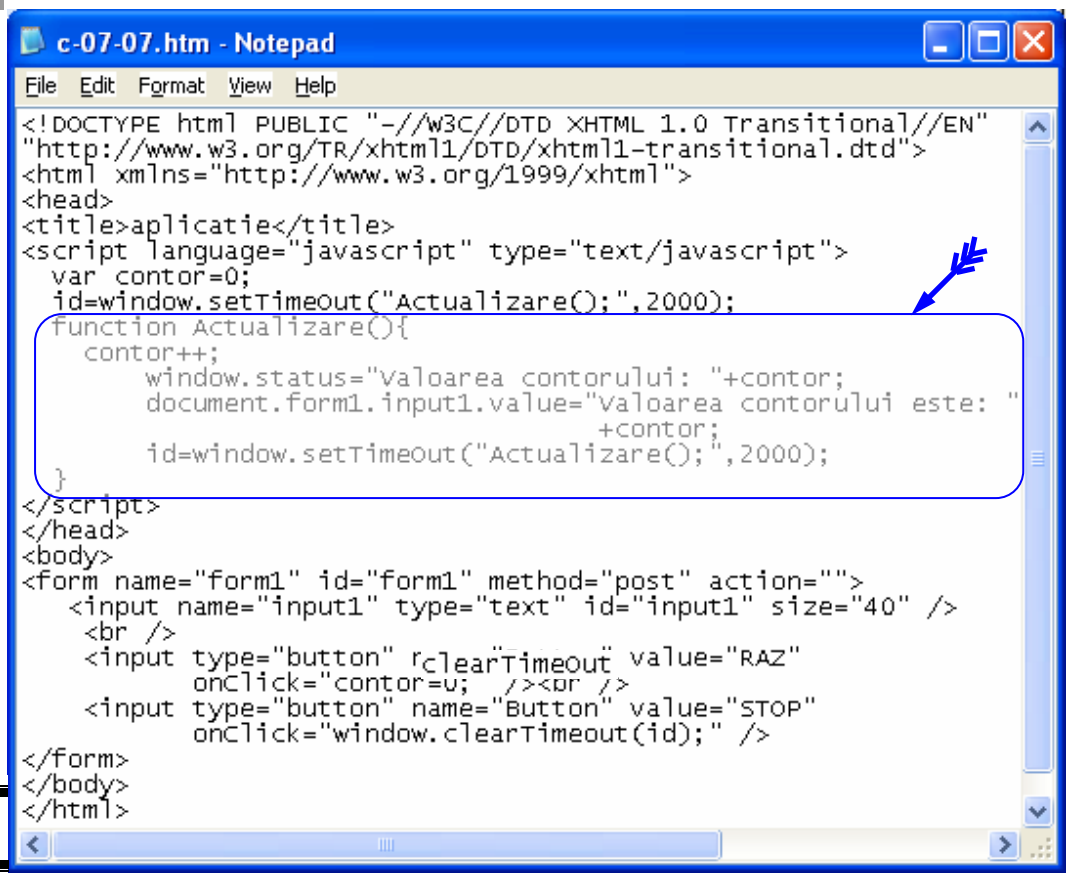


Figura 7.16

❑ Creaţi un script pentru a afişa trei casete de dialog cu ajutorul metodelor `alert()`, `confirm()`, `prompt()`.

În figura 7.17 este afişat documentul complet XHTML.

c-07-08.htm - Notepad

File Edit Format View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Casete de dialog JavaScript</title>
</head>

<body>
<form name="ferestre" id="ferestre" method="post" action="">
  <p>
    <input type="submit" name="Submit" value="alert"
     onClick="window.alert('Acesta este un test pentru
                            metoda alert');" />
  </p>
  <p>
    <input type="button" name="Submit2" value="confirm"
     onClick="temp=window.confirm('Va rog sa confirmati!');
     window.status=(temp)?'confirm:true':'confirm:false';" />
  </p>
  <p>
    <input type="button" name="Submit3" value="prompt"
     onClick="var temp=window.prompt('Introduceti text: ',
      'Aceasta este valoarea implicita'); window.status=temp;" />
  </p>
</form>
</body>
</html>
```

Figura 7.17

Remarcă. Documentul afişează trei butoane, fiecare dintre ele apelând câte un gestionar de evenimente pentru a afişa o casetă de dialog.

În figurile 7.18, 7.19, 7.20 sunt prezentate rezultatele execuţiei programului JavaScript.

Figura 7.18

Figura 7.19

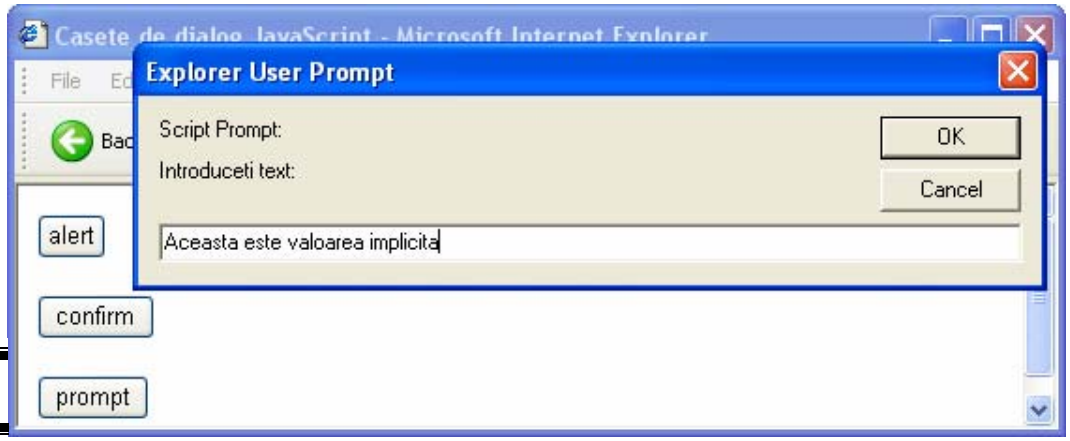


Figura 7.20

Obiecte de nivelul 1

Obiectul `Window`, considerat ca fiind obiectul de cel mai înalt nivel în ierarhia obiectelor JavaScript pe parte de client conţine toate celelalte obiecte pe parte de client, cu excepţia obiectului `Navigator`.

Este important să faceţi cunoştinţă cu aceste obiecte, şi ... să rămâneţi împreună! Ele sunt:

- ✓ `Document;`
- ✓ `Frame;`
- ✓ `History;`
- ✓ `Location;`
- ✓ `Event;`
- ✓ `Style.`

Remarcă. Aceste obiecte sunt considerate obiecte de nivelul 1.

Obiectul Document

Manipularea documentelor Web

Obiectul `Document` reprezintă pagina Web afişată într-o fereastră a navigatorului. Nu pare deci surprinzător faptul că obiectele `Document` sunt fii ai obiectelor `Window`. În măsura în care obiectul `Window` reprezintă întotdeauna fereastra activă (cea care conţine script-ul), puteţi utiliza `window.document` pentru a vă referi la documentul curent sau, mai simplu, utilizaţi numai `document`.

Remarcă. În programele JavaScript din conversaţiile precedente aţi făcut apel la metoda `document.write` pentru a afişa un text într-un document Web. Programele la care ne referim nu utilizau decât o fereastră; în consecinţă, era inutil să folosiţi descrierea completă `window.document.write`.

Dacă mai multe ferestre sau mai multe cadre sunt deschise, atunci vor exista mai multe obiecte `window` şi un singur obiect `Document` pentru fiecare dintre ele. Pentru a utiliza un astfel de obiect `Document` ne folosim de numele ferestrei urmat de numele obiectului.

Fişa obiectului `Document` este prezentată în figura 7.21.

Fişa obiectului Document

Obiectul părinte:	`Window`
Subobiecte:	`Anchor, Area, Applet, Form, Image, Layer, Link, Plugin`
Proprietăţi:	`activeElement, alinkColor, all[], anchors[], applets[], attributes[], background, bgColor, body, characterSet, charSet, childNodes[], cookie, defaultcharset, domain, embeds[], fgcolor, fileCreatedDate, fileModifiedDate, fileSize, forms[], frames[], height, images[], innerHTML, innerText, lastModified, layers[], links[], location, outerHTML, outerText, parentWindow, plugins[], protocol, readyState, referrer, scripts[], selection, style, stylesheets[], title, URL, vlinkColor, width`
Metode:	`clear(), close(), createElement(), createStyleSheet(), elementFormPoint(), getElementByld(), getElementsByName(), getElementsByTagName(), getSelection(), open(), setActive(), write(), writeln`
Gestionarii de evenimente:	`onActivate, onBeforeCut, onBeforeDeActivate, onBeforeEditFocus, onBeforePaste, onBeforeUpdate, onCellChange, onClick, onContextMenu, onControlSelect, onCut, onDblClick, onDrag, onDragEnd, onDragEnter, onDragLeave, onDragOver, onDragStart, onDrop, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseOut, onMouseOver, onMouseUp, onPaste, onPropertyChange, onSelectionChange, onSelectStart, onStop`

Figura 7.21

Proprietăţile fundamentale ale obiectului Document

Proprietăţile fundamentale ale obiectului `Document` sunt prezentate în detaliu în figura 7.22.

	Proprietate	***Sintaxă***
	`alinkColor`	`document.alinkColor`
	Conţine culoarea atribuită link-urilor active, definită prin atributul `alink` al tag-ului `<body>`. Pentru a modifica culoarea link-ului, atribuiţi o nouă valoare a culorii.	
Exemplu:	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html xmlns="http://www.w3.org/1999/xhtml"> <head> <title>Aplicatie</title> <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> <script language="JavaScript" type="text/JavaScript"> document.write(" Legaturi active: "+document.alinkColor); document.write(" Legaturi: "+document.linkColor); document.write(" Legaturi vizitate: "+document.vlinkColor); </script> </head> <body link="blue" alink="yellow" vlink="purple"> </body> </html>	
	`all[]`	`document.all[]`
	Toate elementele documentului. Fiecare element este o dată al matricii (array) `all[]`. Elementele sunt referite prin numărul lor de ordine sau prin numele lor. Utilizaţi metoda `tags()` pentru a obţine setul de elemente ale unui tag precis.	
Exemplu:	<html xmlns="http://www.w3.org/1999/xhtml"> <head> <title>all[]</title> </head> <body> O imagine A doua imagine <script language="JavaScript" type="text/javascript"> //Afiseaza id-ul tag-urilor para=document.all.tags("p"); document.write(para[0].id+" "); document.write(para[1].id+" "); </script> </body> </html>	

Figura 7.22

`anchors[]`	`document.anchors[]`
Matricea `anchors[]` conţine un obiect `Anchor` (ancoră) pentru fiecare set de tag-uri:	
`applets[]`	`documet.applets[]`
Matricea `applets[]` conţine toate applet-urile Java inserate în documentul (X)HTML. Fiecare element al matricii corespunde unui set de tag-uri `<applet> ... </applet>`. `document.applets.length` returnează numărul de applet-uri ale documentului.	
`bgColor`	`document.bgColor`
Culoarea de fond a documentului – numai atributul `bgColor` al tag-ului `<body>` (vezi proprietatea `alinkColor`).	
`body`	`document.body`
Referinţă la secţiunea `body` a documentului. `body` este un obiect care recunoaşte proprietăţile obiectului `document`.	

Exemplu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>document.body</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
 nouFer=window.open("","nou","height=200, width=200");
 nouFer.document.write("<head><title>demo</title></head><body><b>La
multi ani impreuna!</b></body>");
 document.write(nouFer.document.body.innerHTML);
 nouFer.close();
</script>
</body>
</html>
```

`cookie`	`document.cookie`
Permite citirea şi configurarea valorii cookie-ului clientului pentru un document (X)HTML. Citirea şi înregistrarea cookie-ului sunt foarte simple. **Remarcă.** Utilizarea unui `cookie` este puţin mai complicată.	

Exemplu:

```
...
 //citeste cookie
 var numCookie=document.cookie;
 //scrie cookie
 document.cookie=numCookie;
...
```

Figura 7.22 *(continuare)*

`embeds[]`	`document.embeds[]`

Matricea `embeds[]` conţine toate elementele incorporate în documentul (X)HTML. Într-o pagină Web, un element incorporat este plasat între tag-urile <embed> şi </embed>.
`document.embeds.length` returnează numărul de elemente incorporate în document.

`fgColor`	`document.fgColor`

Culoarea textului. Corespunde atributului `text` al tag-ului <body> (vezi proprietatea `alinkColor`).

`forms[]`	`document.forms[]`

Setul de formulare conţinute în documentul (X)HTML. Formularele sunt reprezentate în documentul (X)HTML prin tag-ul <form>. Fiecare formular este un element al matricii (array) `forms[]`. Primul formular are rangul 0.
`document.forms.length` returnează numărul de formulare ale documentului.

Exemplu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>form</title>
</head>
<body>
<form name="form1">
  <input name="zona1" type="text" id="zona1" />
  <br />
  <input name="zona2" type="text" id="zona2" />
</form>
<form name="form2" id="form2" method="post" action="">
  <input name="zona3" type="text" id="zona3" />
  <br />
  <input name="zona4" type="text" id="zona4" />
</form>
<form name="form3" id="form3" method="post" action="">
  <p>
    <input name="zona5" type="text" id="zona5" />
    <br />
    <input name="zona6" type="text" id="zona6" />
  </p>
</form>
<script language="JavaScript" type="text/JavaScript">
  for(i=0;i<document.forms.length;i++){
        document.write(document.forms[i].name+"<br />");
  }
</script>
</body>
</html>
```

Figura 7.22
(continuare)

`frames[]`	`document.frames[]`
	Setul de cadre (*frames*, în limba engleză) afişate. Fiecare cadru este un element al matricii `frames[]`. Ele sunt referite prin numărul lor de ordine sau prin numele lor. Primul este de rang 0. `Document.frames.length` returnează numărul de cadre ale documentului.
`images[]`	`document.images[]`
	Setul de imagini ale documentului. Ele sunt inserate cu tag-ul `<img/>`. Fiecare imagine este un element al matricii (array) `images[]`. Ele sunt referite prin numărul lor de ordine sau prin numele lor. `document.images.length` returnează numărul de imagini ale documentului. Imaginile recunosc proprietăţile care corespund atributelor tag-ului `<img/>`: `border, height, hspace, lowsrc, name, src, vspace, width`. În plus, proprietatea `complete` conţine valoarea `true` dacă imaginea s-a încărcat integral.

Exemplu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>demo</title>
</head>
<body>
<img src="sigla.jpg" name="sigla" width="150" height="300" hspace="5"
vspace="10" border="0" lowsrc="siglamin.JPG" id="sigla" />
<script language="JavaScript" type="text/JavaScript">
 text="Proprietatile imaginii: <br />";
 text+="border= "+document.sigla.border+"<br />";
 text+="complet= "+document.sigla.complete+"<br />";
 text+="height= "+document.sigla.height+"<br />";
 text+="hspace= "+document.sigla.hspace+"<br />";
 text+="lowsrc= "+document.sigla.lowsrc+"<br />";
 text+="name= "+document.sigla.name+"<br />";
 text+="src= "+document.sigla.src+"<br />";
 text+="vspace= "+document.sigla.vspace+"<br />";
 text+="width= "+document.sigla.width+"<br />";
  document.write(text);
</script>
</body>
</html>
```

`innerHTML`	`document.innerHTML`
	Conţine codul HTML al documentului. Acest atribut poate fi aplicat întregului obiect din document pentru a recupera sau pentru a modifica codul (X)HTML care îl închide.

Figura 7.22
(continuare)

Exemplu:

```
...
<body>
  <p id=paragraf><b>text</b>demo</p>
  <script>
    element=document.getElementByld("paragraf");
    alert(element.innerHTML);
  </script>
...
```

`innerText`	`document.innerText`

Recuperează sau defineşte codul (X)HTML ca pe un text, fără interpretarea tag-urilor.

`lastModified`	`document.lastModified`

Data şi ora ultimei modificări a documentului. Este un obiect `String`.

Exemplu:

```
<script>
  Document.write(document.lastModified);
</script>
```

`layers[]`	`document.layers[]`

Setul de straturi reprezentate în pagina (X)HTML prin tag-urile `<div>` sau `<layer>`. Pot fi accesate prin atributele `id` sau `name`.

`document.layers.length` returnează numărul de straturi ale documentului.

Remarcă. Setul de layer-e nu este recunoscut de Netscape.

`linkColor`	`document.linkColor`

Codul culorii atribuite link-urilor din documentul (X)HTML, definite prin atributul `link` al tag-ului `<body>`.

`links[]`	`document.links[]`

Matricea `links[]` conţine un obiect `Link` pentru fiecare tag `<a href="..."> </a>` prezent în documentul (X)HTML. Obiectele `links` au aceleaşi proprietăţi ca obiectul `location`.

Exemplu:

```
...
  <a href="http://www.dumitrascu.ro">Link</a>
  <form>
    <input type="button" value="Schimba link-ul"
    onClick="document.links[0].href='http://www.upg-ploiesti.ro'">
  </form>
```

`location`	`document.location`

Conţine `URL`-ul documentului.

Figura 7.22 *(continuare)*

Exemplu:

```
...
<body>
  <script>
    document.write(document.location);
      //se afişează, de exemplu http://www.dumitrascu.ro
  </script>
</body>
```

Exemplu:

```
…
<script>
document.write("adresa actuala: "+document.location);
 document.location="http://www.multimedia.fr";
</script>
```

`plugins[]`	`document.plugins[]`
Setul de `plug-ins` conţinute în documentul (X)HTML. Fiecare `plug-in` este un element al matricii `plugins[]`. Ele sunt referite prin numărul lor de ordine. `document.plugins.length` returnează numărul de plug-ins ale documentului.	
`referrer`	`document.referrer`
URL-ul documentului de la care documentul curent a fost încărcat.	
`scripts[]`	`document.scripts[]`
Setul de script-uri conţinute în documentul (X)HTML. Ele sunt reprezentate în document prin tag-ul `<script>`. Fiecare script este un element al matricii (`array`) `scripts[]`. Ele sunt referite prin numărul lor de ordine. `document.scripts.length` returnează numărul de script-uri ale documentului.	
`title`	`document.title`
Titlul documentului, cel care este specificat în tag-ul `<title>`.	
URL	`document.URL`
URL-ul documentului. Această proprietate este puţin utilizată căci ea este echivalentă cu `document.location.href`.	
`vlinkColor`	`document.vlinkColor`
Codul culorii atribuit link-urilor vizitate în document, acelaşi care este definit prin atributul `vlink` al tag-ului `<body>` (vezi proprietatea `alinkColor`).	

Figura 7.22 *(continuare)*

Metodele fundamentale ale obiectului Document

Cele mai utilizate metode ale obiectului `Document` sunt prezentate în detaliu în figura 7.23.

Metodă	*Sintaxă*
`close()`	`document.close()`
Afişează datele încărcate dar care nu sunt încă afişate şi opreşte fluxul de intrare al datelor în document. Dacă toate datele nu au fost încărcate, documentul nu este afişat integral. Această metodă nu cere nici un argument. Metoda nu este indispensabilă întrucât metoda `write()` care scrie în document realizează în mod automat deschiderea (`open`) şi închiderea (`close`) fluxului de date (vedeţi paragraful următor: „Ştergerea şi rescrierea conţinutului unui document").	
`getElementById()`	`document.getElementById(id)`
Returnează sub formă de matrice (array) lista elementelor identificate prin nume sau `id`-ul indicat. Dacă nu există nici un element de tipul indicat, metoda returnează `null`. Elementele dispun de proprietăţi şi de metode specifice obiectelor. Această metodă este folosită de asemenea pentru detectarea navigatorului.	

Exemplu:

```
<body>
<p id="unu">Primul paragraf </p>
<p id="doi">Al doilea paragraf  </p>
<script language="JavaScript" type="text/JavaScript">
  //Modifica aliniamentul unui paragraf
  para=document.getElementById("unu");
  para.align="center";
</script>
</body>
<script language="JavaScript" type="text/JavaScript">
   //Detecteaza navigatorul
   if(document.layers) Nav="NN4";
   else if((document.all)&&(document.getElementById))
        Nav="IE5-IE6";
              else if(document.getElementByID)Nav="NN6";
   alert(Nav);
</script>
```

`getElementsByName()`	`document.getElementsByName(Nume)`
Returnează sub formă de matrice (array) lista elementelor identificate prin nume sau `id`-ul indicat. Dacă nu există nici un element de tipul indicat, metoda returnează `null`.	

Exemplu:

```
<body>
<p id="unu">Primul paragraf </p>
<p id="doi">Al doilea paragraf  </p>
<script language="JavaScript" type="text/JavaScript">
  //Modifica aliniamentul unui paragraf
  para=document.getElementsByName("unu");
  para[0].align="center";
</script>
</body>
```

Figura 7.23

`getElementsByTagName()` `document.getElementsByTagName(tag)`

Returnează sub formă de matrice (array) lista elementelor identificate prin tag-ul indicat.

Dacă nu există nici un element de tipul indicat, metoda returnează **`null`**.

Exemplu:
```
<body>
<p id="unu">Primul paragraf </p>
<p id="doi">Al doilea paragraf  </p>
<script language="JavaScript" type="text/JavaScript">
 //Modifica aliniamentul unui paragraf
el=document.getElementsByTagName("p");
 el[0].align="center";
</script>
</body>
```

`open()` `document.open(TipMime,replace)`

Deschide documentul pentru a trimite datele care vor fi afişate cu metoda `write()`. Primul argument indică tipul `MIME` al datelor (`text/html`; `text/plain`; `image/gif`; `image/jpeg`; `image/sc-bitmap`; `plugIn` (orice model plug-in Netscape de tip `MIME`)).

Al doilea argument precizează dacă datele vor înlocui documentul deja prezent în fereastra navigatorului (`true`) sau se vor adăuga celor existente (`false`). Cele două argumente sunt facultative.

Această metodă nu este indispensabilă întrucât metoda `write()` care scrie în document realizează în mod automat deschiderea (`open`) şi închiderea (`close`) fluxului de date.

Remarcă. Cele două script-uri sunt echivalente.

Exemplu:
```
<script language="JavaScript" type="text/JavaScript">
 fernoua=window.open("","nou","height=200, width=400");
 fernoua.document.open();
 fernoua.document.write("<head><title>demo</title></head> <body><b>La multi
   ani impreuna!</b></body>");
 fernoua.document.close();
</script>
```

Exemplu:
```
<script language="JavaScript" type="text/JavaScript">
 fernoua=window.open("","nou","height=200, width=400");
 fernoua.document.write("<head><title>demo</title></head><body><b>La multi
   ani impreuna!</b></body>");
</script>
```

`write()` `document.write(şir1,şir2)`

Introduce în document: date de tip caracter, variabile şi tag-uri. Pentru caractere speciale (apostrof, de exemplu) introduceţi caracterele corespunzătoare.

Figura 7.23 *(continuare)*

Exemplu:
```
<script>
document.write("D\' ale Carnavalului");
 //Afişează D'ale Carnavalului
</script>
```

`writeln()`	`document.writeln(şir1, şir2)`
	Introduce date de tip caracter, variabile şi tag-uri şi inserează un salt de linie la sfârşit. Cum salturile de linie sunt ignorate de către navigatoare, această metodă este identică cu `write()`.

Figura 7.23 *(continuare)*

Revenind la metodele open() şi close() ale obiectului Document

Obiectul `Document` conţine metodele `open()` şi `close()` (vezi metodele obiectului `Document`, figura 7.23).

Atunci când utilizaţi metoda `write()` sau `writeln()` (vezi metodele obiectului `Document`, figura 7.23) trimiteţi, de fapt, datele către ceva care se cheamă stream (flux), care corespunde unei zone rezervate de date. Datele rămân în stream până în momentul în care navigatorul este pregătit să le afişeze în fereastra sa. Va trebui să deschideţi (`open`) iar stream-ul înainte de a-l utiliza apoi să-l închideţi (`close`).

Metodele `open()` şi `close()` efectuează cele două acţiuni.

Metoda `open()` indică navigatorului că doriţi să scrieţi un nou flux de date în obiectul `Document` curent.

Metoda `close()` semnalează sfârşitul acţiunii de scriere; ea obligă navigatorul să afişeze toate datele care au mai rămas în flux (stream).

Utilizarea celor două metode nu este însă obligatorie întrucât `write()` şi `writeln()` realizează în mod automat deschiderea (`open`) şi închiderea (`close`) stream-ului.

În concluzie, pentru a trimite un flux de date către fereastra navigatorului, parcurgeţi următorii paşi:

- ✓ Utilizaţi `document.open()` pentru a deschide obiectul `Document` şi pentru a putea din nou scrie.
- ✓ Utilizaţi `document.write()/document.writeln()`.

- ✓ Utilizaţi `document.close()` pentru a indica navigatorului că aţi terminat de scris în fereastra sa.

Remarci:

- ✓ Metodele `open()` şi `close()` nu deschid şi nici nu închid noile ferestre.
- ✓ Dacă utilizaţi metoda `document.open()`, în fereastra curentă, script-ul dumneavoastră, care face parte din documentul curent, va fi şters şi în consecinţă va înceta să se execute. Mult mai bine este să utilizaţi metodele `open()` şi `close()` cu ferestre sau cadre separate.

Cu metoda `document.open` puteţi specifica un tip `MIME` care vă oferă posibilitatea de a crea un document de un anume tip ce conţine imagini şi documente utilizate în cadrul plug-ins.

Remarcă. `MIME` (Multipurpose Internet Mail Extension) este un standard pentru documentele Internet. Atunci când server-ul trimite un document unui browser, i se indică şi tipul `MIME` al documentului pentru ca navigatorul să ştie cum să-l afişeze. Printre tipurile `MIME` cele mai răspândite amintim formatul HTML (tipul `MIME text/html`) şi formatul text (tipul `text/plain`).

Aplicaţii ale metodelor open() şi close()

❑ Analizaţi următorul document HTML (vezi figura 7.24). Precizaţi rolul instrucţiunilor: `document.open()` şi `document.close()`.

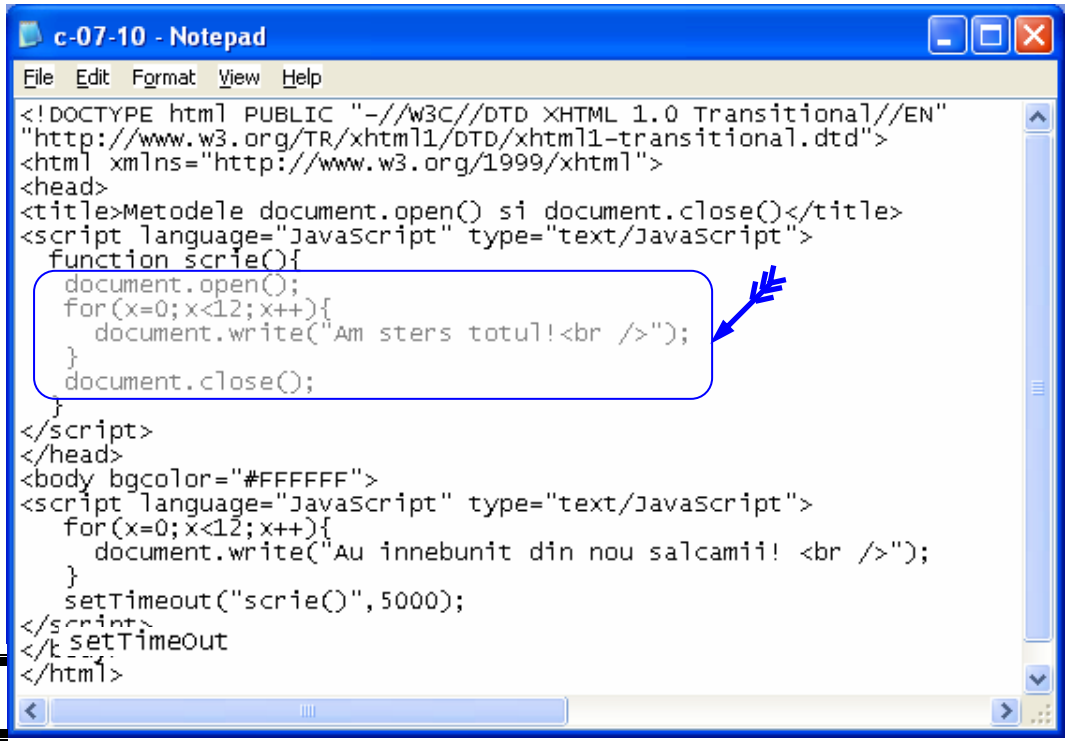


Figura 7.24

❑ Script-ul prezentat ([1]) în figura 7.25 evaluează browser-ul (Netscape, Internet Explorer) şi modifică textul în funcţie de tipul său. Comentaţi script-ul.

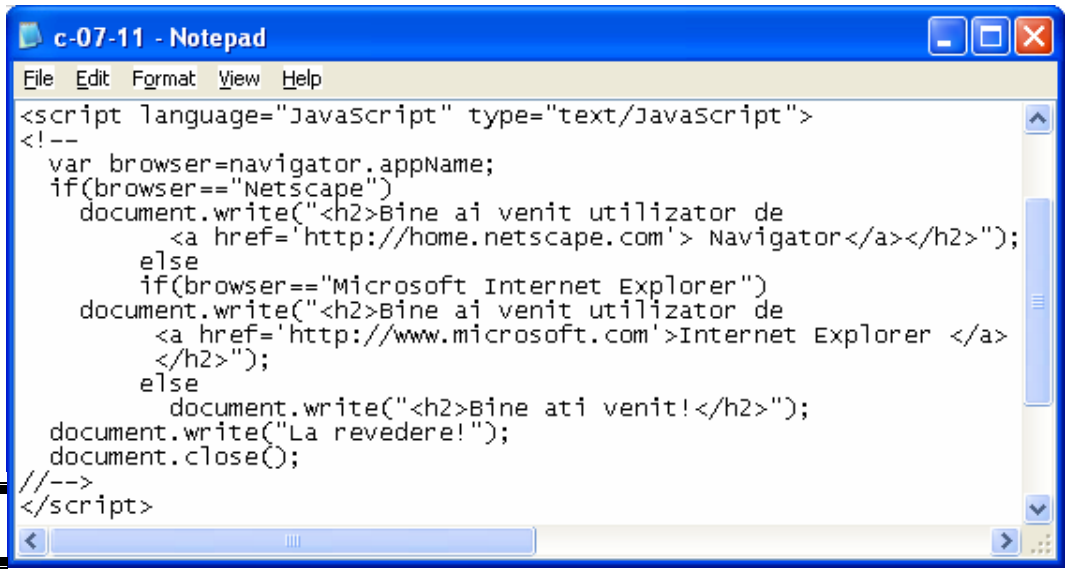


Figura 7.25

Remarcă. Nu uitaţi că atunci când folosiţi metodele `write ()` sau `writeln()` nu puteţi schimba conţinutul documentului curent fără să reîncărcaţi complet pagina.

Obiectul Frame

Atunci când o fereastră conţine mai multe cadre, fiecare dintre ele este reprezentat în JavaScript printr-un obiect `Frame`.

Remarcă. Obiectul `Frame` este prezentat în detaliu în Conversaţia 10.

Obiectul Location

Obiectul `Location` este un fiu al obiectului `Window`. El stochează informaţii despre adresa `URL` a unei ferestre specificate. Fişa obiectului `Location` este prezentată în figura 7.26.

Remarcă. Deşi utilizatorii văd `URL`-ul afişat în caseta de locaţie (`Location`) a navigatorului, programatorii JavaScript lucrează cu obiectul `Location`.

Fişa obiectului Location	
Obiectul părinte:	`Window`
Proprietăţi:	`hash, host, hostname, href, pathname, port, protocol, search`
Metode:	`assign(), reload(), replace()`
Gestionarii de evenimente:	-

Figura 7.26

Proprietăţile obiectului Location

Proprietăţile obiectului `Location` sunt prezentate în detaliu, în figura 7.27. Ele reprezintă părţi ale `URL`-ului.

	Proprietate	***Sintaxă***
	`hash`	`document.location.hash`
	Conţine partea # (diez) a adresei Web specificate în `URL`.	
Exemplu:	<script> document.write(document.location.hash); //afişează top, dacă adresa este http://www.ionescu.com/index.htm#top </script>	
	`host`	`document.location.host`
	Numele server-ului şi postul `URL`.	
Exemplu:	<script> document.write(document.location.host); //afişează www.ionescu.com (vezi hash) </script>	
	`hostname`	`document.location.hostname`
	Numele server-ului.	
Exemplu:	<script> document.write(document.location.hostname); //afişează www.ionescu.com (vezi hash) </script>	
	`href`	`document.location.href`
	Adresa Web (`URL`-ul) completă.	
Exemplu:	<script> document.write(document.location.href); /*afişează www.ionescu.com/linkuri/pref.htm dacă pagina afişată este la adresa http://www.ionescu.com/linkuri/pref.html */ </script>	
	`pathname`	`document.location.pathname`
	Calea de acces la document.	
Exemplu:	<script> document.write(document.location.pathname); //afişează /linkuri/pref.html (vezi href) </script>	
	`port`	`document.location.port`
	Numărul de port al `URL`-ului (în general 80).	
Exemplu:	<script> document.write(document.location.port); /*afişează 80, dacă adresa este http://www.ionescu.com:80/linkuri/pref.html*/ </script>	

Figura 7.27

	`protocol`	`document.location.protocol`
	Partea „protocol” a `URL`-ului (în general `http`).	
	`search`	`document.location.search`
	Partea de căutare a adresei `URL` de după semnul ?	
Exemplu:	<script> document.write(document.location.search); </script>	

Figura 7.27 *(continuare)*

Metodele obiectului Location

Metodele obiectului `Location` sunt prezentate în detaliu în figura 7.28.

	Metodă	***Sintaxă***
	`assign()`	`document.location.assign(`*`URL`*`)`
	Modifică URL-ul documentului.	
Exemplu:	<form> <input type=”button” value=”Yahoo!” onClick=’window.location.assign(”http://www.yahoo.com”)’> </form>	
Exemplu:	<form> <input type=”button” value=”Yahoo!” onClick=’window.location=”http://www.yahoo.com”’> </form>	
	`reload()`	`document.location.reload()`
	Reîncarcă documentul curent în fereastra browser-ului. Dacă argumentul (reîncărcare) are valoarea logică `true` documentul este reîncărcat de pe server; în caz contrar el este recuperat în cache-ul navigatorului.	
	`replace()`	`document.location.replace(`*`URL`*`)`
	Înlocuieşte pagina curentă cu o nouă pagină. Metoda nu afectează istoricul navigatorului. Utilizarea metodei este justificată pentru a evita revenirea la ecranele de prezentare sau la paginile temporare de interes minim.	

Figura 7.28

Aplicaţii

❑ Modificaţi `URL`-ul documentului curent.

În figura 7.29 este prezentată secvenţa HTML corespunzătoare.

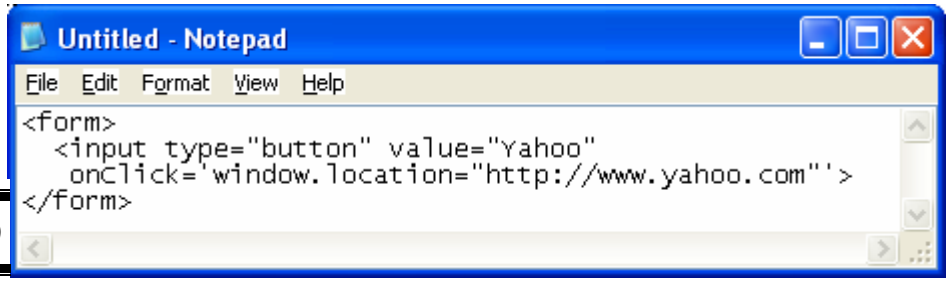


Figura 7.29

❑ Modificaţi URL-ul documentului într-o fereastră secundară pornind de la fereastra principală.

În figura 7.30 este prezentat documentul XHTML complet al aplicaţiei.

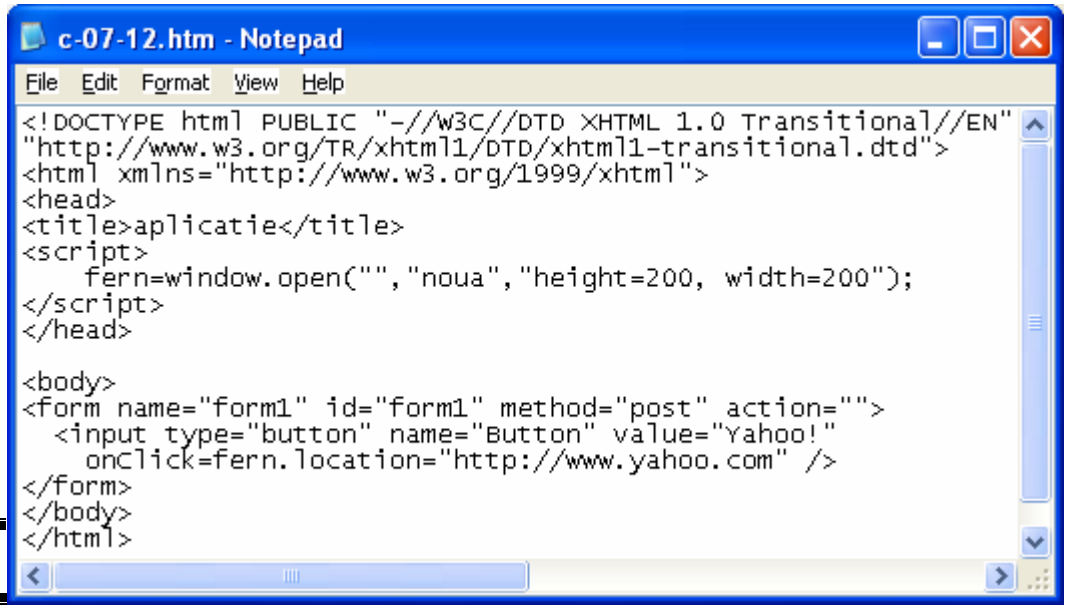


Figura 7.30

Obiectul History

Obiectul `History` este un alt obiect fiu al obiectului `Window`. Acest obiect conţine URL-ul paginilor vizitate înainte şi după pagina curentă.

Fişa obiectului `History` este prezentată în figura 7.31.

Fişa obiectului History	
Obiectul părinte:	`window`
Proprietăţi:	`length`
Metode:	`back(), forward(), go()`
Gestionarii de evenimente:	-

Figura 7.31

Proprietăţile obiectului History

Proprietăţile obiectului `History` sunt prezentate în detaliu în figura 7.32.

	Proprietate	***Sintaxă***
	`length()`	`window.history.length`
	Numărul de pagini din istoric.	
Exemplu:	<body> Ultimele pagini </body>	

Figura 7.32

Remarci:

- ✓ Obiectul `History` mai conţine şi proprietăţile: `current`, `previous` şi `next` care stochează `URL`-ul curent, anterior şi următor al documentului în lista de istoric.
- ✓ Din motive de securitate, aceste obiecte sunt normal inaccesibile în browser-ele actuale.

Metodele obiectului History

Trei metode ale obiectului `History` permit deplasarea în lista de istoric (vezi figura 7.33).

	Metodă	***Sintaxă***
	`back()`	`window.history.back()`
	Retur la `URL`-ul precedent în istoric. Această metodă corespunde unui clic pe butonul `back` al navigatorului.	
Exemplu:	<body> <form> <input type="button" value="Pagina precedenta" onClick="window.history.back()"> </form> </body>	
	`forward()`	`window.history.forward()`
	Trece la `URL`-ul următor în istoric. Această metodă corespunde unui clic pe butonul `forward` al navigatorului.	
	`go()`	`window.history.go(`*`Număr`*`)`
	Avansează sau se întoarce în istoric cu numărul de pagini indicate.	
Exemplu:	<body> Mergi inapoi cu 2 pagini Mergi inainte cu 2 pagini </body>	

Figura 7.33

Obiectul Event

Obiectul `Event` este un alt fiu al obiectului `Window`.

Fişa obiectului `Event` este prezentată în figura 7.34.

Fişa obiectului Event	
Obiectul părinte:	`window`

Proprietăţi:	`altkey, button, cancelable, clientX, ctrlKey, currentTarget, data, fromElement, height, keyCode, layerX, layerY, modifiers, pageX, pageY, reason, returnValue, screenX, screenY, shiftkey, srcElement, srcFilter, target, toElement, type, which, width, x, y`
Metode:	-
Gestionarii de evenimente:	-

Figura 7.34

Proprietăţile obiectului Event

Câteva din proprietăţile obiectului `Event` sunt prezentate în detaliu în figura 7.35.

Proprietate	***Sintaxă***
`altkey`	`event.altkey`
Returnează valoarea logică `true` (adevărat) dacă tasta `Alt` a fost acţionată.	
`button`	`event.button`
Determină care din butoanele mouse-ului a fost acţionat: `left, right` sau `middle`.	
`clientX`	`event.clientX`
Returnează coordonata orizontală de amplasare a evenimentului.	
`clientY`	`event.clientY`
Returnează coordonata verticală de amplasare a evenimentului.	
`data`	`event.data`
Atunci când se generează evenimentul `dragdrop`, proprietatea conţine o matrice (array) cu URL-ul obiectului stocat.	
`height`	`event.height`
Atunci când se generează evenimentul `resize`, proprietatea conţine noua valoare a înălţimii elementului.	
`layerX`	`event.layerX`
Coordonata orizontală a stratului în raport cu stratul care o conţine.	
`layerY`	`event.layerY`
Coordonata verticală a stratului în raport cu stratul care o conţine.	

Figura 7.35

pageX	event.pageX
Coordonata orizontală de amplasare a evenimentului care se produce.	
pageY	event.pageY
Coordonata verticală de amplasare a evenimentului care se produce.	
returnValue	event.returnValue
Această proprietate are prioritate în raport cu valoarea returnată de către un gestionar de evenimente. Pentru a anula efectul evenimentului, modificaţi conţinutul său atribuindu-i valoarea logică `false`.	
srcElement	event.srcElement
Referinţă la elementul care a generat evenimentul.	
type	event.type
Specifică numele evenimentului care se produce.	

Figura 7.35 *(continuare)*

Obiecte de nivelul al doilea

Aşa cum există subobiecte ale obiectului `Window`, tot aşa există şi subobiecte – obiecte de nivelul al doilea ale obiectului `Document`. Subobiectele obiectului `Document` sunt următoarele:

- ✓ `Anchor;`
- ✓ `Area;`
- ✓ `Applet;`
- ✓ `Form;`
- ✓ `Image;`
- ✓ `Layer;`
- ✓ `Link;`
- ✓ `Plugin.`

Obiectul Anchor

Obiectele Anchor sunt fii ai obiectului Document. Fiecare obiect Anchor reprezintă o ancoră a documentului curent, adică un loc specific în document care poate fi atins direct printr-un link.

Matricea anchors[]

Singurul mod în care puteţi folosi realmente un obiect Anchor în limbajul JavaScript este prin intermediul matricii anchors[] al obiectului Document (document.anchors[]).

Folosiţi matricea document.anchors[] pentru a determina numărul de ancore dintr-un document (figura 7.36).

Folosiţi proprietatea name pentru a recupera numele unei ancore şi proprietatea innerText pentru a recupera textul său (figura 7.36).

```
...
  <a name="ancora1">test1</a><br>
  <a name="ancora2">test2</a><br>
  <script>
    document.write(document.anchors.length+"ancore numite");
   //afişează două ancore numite
  <//script>
…
  <a name="ancora1">test1</a><br>
  <a name="ancora2">test2</a><br>
  <script>
    document.write(document.anchors[0].name);
    document.write("→");
    document.write(document.anchors[1].innerText);
    //afişează ancora 1→ test2
  </script>
…
```

Figura 7.36

Obiectul Area

Obiectul Area vă permite să definiţi o suprafaţă a unei imagini reactive.

Proprietăţile obiectului Area *sunt*: hash (porţiunea de adresă URL care este ancora, inclusiv simbolul #); host (numele calculatorului gazdă (adresa IP) şi portul specificate în adresa URL); hostname (numele calculatorului gazdă specificat în URL); href (adresa URL completă); pathname (calea fişierului

specificat în adresa URL, începând cu simbolul (”/”); port (portul specificat în adresa URL); protocol (protocolul specificat în adresa URL, inclusiv simbolul final (”:”); search (partea de căutare a adresei URL, inclusiv simbolul (”?”); target (numele ferestrei ţintă în care se afişează adresa URL); text (textul care apare între tag-urile <area> ...</area>).

Obiectul Applet

Obiectul Applet reprezintă echivalentul JavaScript al tag-ului (X)HTML <applet>.

Obiectul Form

Obiectul Form este un obiect JavaScript reprezentat prin perechea de tag-uri (X)HTML: <form> şi </form>.

Remarcă. Obiectul Form este prezentat în detaliu în Conversaţia 8.

Obiectul Image

Obiectul Image este echivalentul JavaScript al tag-ului (X)HTML <img />.

Remarcă. Obiectul Image este prezentat în detaliu în Conversaţia 9.

Obiectul Layer

Obiectul Layer permite accesarea straturilor în interiorul documentelor.

Remarcă. Obiectul Layer este prezentat în detaliu în Conversaţia 10.

Obiectul Link

Obiectul Link este echivalentul JavaScript al unei legături hipertext.

Fişa obiectului Link este prezentată în figura 7.37.

Fişa obiectului Link	
Obiectul părinte:	Document
Proprietăţi:	hash, host, hostname, href, pathname, port, protocol, search, target (vezi obiectul Location)
Gestionarii de evenimente:	-

Figura 7.37

Un document poate avea mai multe obiecte Link, fiecare dintre ele conţinând informaţii cu privire la URL-ul sau ancora corespunzătoare.

Remarcă. Ancorele sunt elemente numite ale unui document (X)HTML la care puteţi avea acces direct. Pentru a defini o ancoră se utilizează o sintaxă de tipul <a name="ancora2">. Pentru a crea după aceea un link către această ancoră, se utilizează un tag de tipul <a href=#ancora2">.

Matricea links[]

Obiectele Link nu au o proprietate name, deci nu puteţi referi un obiect Link prin el însuşi.

Puteţi accesa obiecte Link cu ajutorul matricii links[] (document.links[]) care este o colecţie a tuturor legăturilor din documentul curent. Ordinea din matrice se bazează pe ordinea în care sunt localizate legăturile în fişierul sursă. O proprietate a matricii, documents.links.length precizează numărul de link-uri ale paginii.

Remarci:

- ✓ Obiectul Link vă permite să lucraţi cu legături în limbajul JavaScript.
- ✓ Obiectul Link este similar cu obiectul Location, care conţine aceleaşi informaţii pentru pagina (X)HTML curentă.

Proprietăţile obiectului Link

Obiectul Link conţine numeroase proprietăţi (acelaşi ca şi obiectul Location) care permit cunoaşterea cu precizie a legăturii (X)HTML care îl reprezintă. Aceste proprietăţi reprezintă părţi ale adresei URL.

Proprietăţile obiectului Link sunt prezentate în detaliu în figura 7.38.

Proprietate	*Sintaxă*
hash	Link.hash
Reprezintă o denumire de ancoră în adresa URL pentru legătură, care începe cu caracterul #.	
host	Link.host
Reprezintă porţiunea de calculator gazdă din adresa URL asociată cu o legătură.	
hostname	Link.hostname
Reprezintă porţiunea de nume al calculatorului gazdă din adresa URL asociată cu o legătură.	
href	Link.href
Reprezintă adresa URL completă asociată cu o legătură.	
pathname	Link.pathname
Reprezintă porţiunea numelui de cale a legăturii URL.	
port	Link.port
Reprezintă porţiunea de port a legăturii URL.	
protocol	Link.protocol
Reprezintă porţiunea de protocol a legăturii URL.	
search	Link.search
Reprezintă porţiunea de interogare a legăturii.	
target	Link.target
Reprezintă numele obiectului Window în care este afişată legătura.	

Figura 7.38

Aplicaţie

❑ Comentaţi secvenţa de cod HTML prezentată în figura 7.39.

```
<a href="http://www.altavista.fr">Legătură</a>
<form>
  <input type="button" value="Schimbați legătura"
   onCLick="document.links[o].href='http://www.yahoo.com'">
</form>
```

Figura 7.39

Obiectul Plugin

Obiectul Plugin, foarte asemănător obiectului Applet reprezintă o modalitate de accesare a modulelor plug-in instalate în browser.

Proprietăţile obiectului `Plugin` *sunt*: `description` (conţine descrierea modulului plug-in); `filename` (conţine numele fişierului unui program plug-in); `length` (conţine numărul de tipuri `MIME` acceptate de modulul plug-in); `name` (conţine numele modulului plug-in).

Obiecte de nivelul al treilea

Următorul nivel de obiecte pe parte de client ale limbajului JavaScript este nivelul al treilea. Toate obiectele de nivelul al treilea sunt subobiecte ale obiectului `Form`: `Button`; `Checkbox`; `FileUpload`; `Hidden`; `Password`; `Radio`; `Reset`; `Submit`; `Select`; `Text`; `Textarea`.

Remarcă. Obiectele de nivelul al treilea pe parte de client sunt prezentate în Conversaţia 8.

Obiecte de nivelul al patrulea

Ultimul nivel de obiecte pe parte de client ale limbajului JavaScript este nivelul al patrulea. Există un singur obiect de nivelul al patrulea: obiectul `Option`, care este un subobiect al obiectului `Select`.

Obiectul `Option` este folosit pentru referirea la elementele `<option>` care apar între tag-urile `<select> ... </select>`.

Obiectul Option

Fişa obiectului `Option` este prezentată în figura 7.40.

Fişa obiectului Option	
Obiectul părinte:	`Select`
Cum se creează obiectul?	`Constructorul option()`
Proprietăţi:	`defaultSelected, disabled, form, index, label, selected, text, value`

Metode:	
Gestionarii de evenimente:	`onClick, onDblClick, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp`

Figura 7.40

Obiectul `Option` este prezentat în detaliu în figura 7.41.

	Obiect	***Sintaxă***
	`Option()`	`document.forms[].option`
	Reprezintă, în cadrul unui formular o listă derulantă de opţiuni incluse între tag-urile `<select>` şi `</select>`. Ea este definită prin tag-urile (X)HTML `<option> ... </option>`. Obiectul este accesat prin proprietatea `form.select.elements[]` sau prin numele său. Pot fi create noi opţiuni cu ajutorul constructorului `Option()`.	
Exemplu	<form name="form1"> <select name="optiune"> <option value="optiune1">prima optiune</option> <option value="optiune2">a doua optiune</option> <option value="optiune3">a treia optiune</option> </select> </form>	

Figura 7.41

Proprietăţile obiectului Option

Proprietăţile obiectului `Option` sunt prezentate în detaliu în figura 7.42.

	Proprietate	***Sintaxă***
	`defaultSelected`	`document.forms[].option.defaultSelected`
	Valoare logică (`true/false`) – indică dacă o opţiune este implicită.	
	`disabled`	`document.forms[].option.disabled`
	Valoare logică (`true/false`) – opţiunea este dezactivată (`true`) sau activată (`false`).	
	`form`	`document.forms[].option.form`
	Renunţă la formularul care conţine zona.	
	`index`	`document.forms[].option.index`

	Rangul opţiunii în lista `select`.
`label`	`document.forms[].option.label`
	Textul alternativ al opţiunii. Proprietatea corespunde atributului `label` al tag-ului.
`selected`	`document.forms[].option.selected`
	Valoare logică (`true/false`) – opţiune selectată (`true`) sau nu (`false`).
`text`	`document.forms[].option.text`
	Textul opţiunii.
`value`	`document.forms[].option.value`
	Valoarea opţiunii. Această proprietate corespunde atributului `value` al tag-ului `<option>`.

Figura 7.42

Obiectul Navigator

Noul standard `DOM` (Document Object Model) a eliminat multe dintre diferenţele care există (încă!) între navigatoare, dar veţi întâlni numeroase situaţii care impun scrierea unui script pentru fiecare navigator, separat. Puteţi utiliza JavaScript pentru a identifica navigatorul pe care îl utilizaţi, folosind obiectul `window.navigator`.

Obiectul `Navigator` nu face parte din `DOM` deci, îl puteţi referi direct.

El este unic şi deci, imposibil de instanţiat (creat).

Puteţi adăuga proprietăţi personale obiectului `Navigator`. Ele sunt disponibile pentru toate obiectele `Window` ale navigatorului Netscape. Spre deosebire de Netscape Navigator, Internet Explorer creează un obiect `Window` pentru fiecare o nouă fereastră. Noile proprietăţi rămân deci limitate la obiectul `Window` în care ele au fost create.

Fişa obiectului `Navigator` este prezentată în figura 7.43.

Fişa obiectului Navigator

Obiectul părinte:	`window`
Subobiecte:	`mimeType, plugin`
Proprietăţi:	`appCodename, appMinorVersion, appName, appVersion, browserLanguage, cooKieEnabled, cpuclass, language, mimeTypes[], onLine, platform, plugins[], product, productSub, systemLanguage, userAgent, userLanguage, vendor`
Metode:	`javaEnabled()`
Gestionarii de evenimente:	-

Figura 7.43

Proprietăţile obiectului Navigator

Proprietăţile obiectului `Navigator` sunt prezentate în detaliu în figura 7.44.

	Proprietate	***Sintaxă***
	`appCodeName`	`navigator.appCodeName`
	Returnează numele codului intern al navigatorului în general "Mozilla".	
Exemplu:	<script> x=navigator.appCodeName; document.write(x); // afişează, de exemplu Mozilla </script>	
	`appName`	`navigator.appName`
	Returnează numele oficial al navigatorului: Microsoft Internet Explorer sau Netscape.	
Exemplu:	<script> x=navigator.appName; document.write(x); // afişează, de exemplu Microsoft Internet Explorer </script>	
	`appVersion`	`navigator.appVersion`
	Returnează numărul versiunii navigatorului actualizate şi versiunea sistemului de operare. Opera este polimorf.	
Exemplu:	<script> x=navigator.appVersion; document.write(x); /* afişează, de exemplu 4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)*/ </script>	
	`browserLanguage`	`navigator.browserLanguage`
	Returnează limba navigatorului. Această proprietate este recunoscută numai de Internet Explorer.	

Figura 7.44

Exemplu:
```
<script>
 x=navigator.browserLanguage;
  document.write(x);
// afişează, de exemplu en
</script>
```

`cookieEnabled` `navigator.cookieEnabled`

Returnează `true` sau `false` numai dacă cookies-urile sunt activate sau nu în navigator.

Exemplu:
```
<script>
 x=navigator.cookieEnabled;
document.write(x);
// afişează, de exemplu true
</script>
```

`language` `navigator.language`

Returnează limba navigatorului . Această proprietate este recunoscută numai de Netscape.

Exemplu:
```
<script>
 x=navigator.language;
  document.write(x);  // afişează, de exemplu ro
</script>
```

`mimeTypes[]` `navigator.mimeTypes[]`

Vezi obiectul `mimeType`.

`platform` `navigator.platform`

Returnează sistemul de operare.

Exemplu:
```
<script>
 x=navigator.platform;
  document.write(x);
// afişează, de exemplu Win32
</script>
```

Figura 7.44 *(continuare)*

`plugins[]` `navigator.plugins[]`

Matricea `plugins[]` conţine lista tuturor extensiilor instalate în navigator. Vezi obiectul `Plugin`.

`userAgent` `navigator.userAgent`

Returnează numele navigatorului urmat de numărul său de versiune. De fapt, este o combinaţie de proprietăţi `appCodename` şi `appVersion`. Opera este polimorf.

Exemplu:
```
<script>
 x=navigator.userAgent;
document.write(x);
/* afişează, de exemplu Mozilla /4.0 (compatible; MSIE 6.0; Windows NT 5.1;
.NET CLR 1.0.3705)*/
</script>
```

`vendor` `navigator.vendor`

	Returnează numele navigatorului dacă este vorba de Netscape 6. Pentru orice alt navigator, returnează `undefined`.
Exemplu:	<script> x=navigator.vendor; document.write(x); // afişează, de exemplu Netscape 6 </script>

Figura 7.44 *(continuare)*

Metodele obiectului Navigator

Metoda `JavaEnabled()` a obiectului `Navigator` este prezentată în figura 7.45.

	Metodă	***Sintaxă***
	`JavaEnabled()`	`navigator.JavaEnabled()`
	Returnează valoarea logică `true` sau `false` numai dacă Java este activat sau nu în navigator.	
Exemplu:	<script> x=navigator.JavaEnabled(); document.write(x); // afişează, de exemplu true </script>	

Figura 7.45

Aplicaţii ale obiectului Navigator

❑ Creaţi un document (X)HTML simplu care conţine un script ce afişează proprietăţile obiectului `Navigator` cu ajutorul instrucţiunii `document.write`. Afişaţi proprietăţile: `appCodeName`, `appName`, `appVersion`, `userAgent`, `language`, `platform` în cele două navigatoare – Netscape şi Internet Explorer.

Indicaţie. Folosiţi o listă neordonată.

În figura 7.46 este prezentat documentul XHTML complet în care s-a inserat script-ul aplicaţiei.

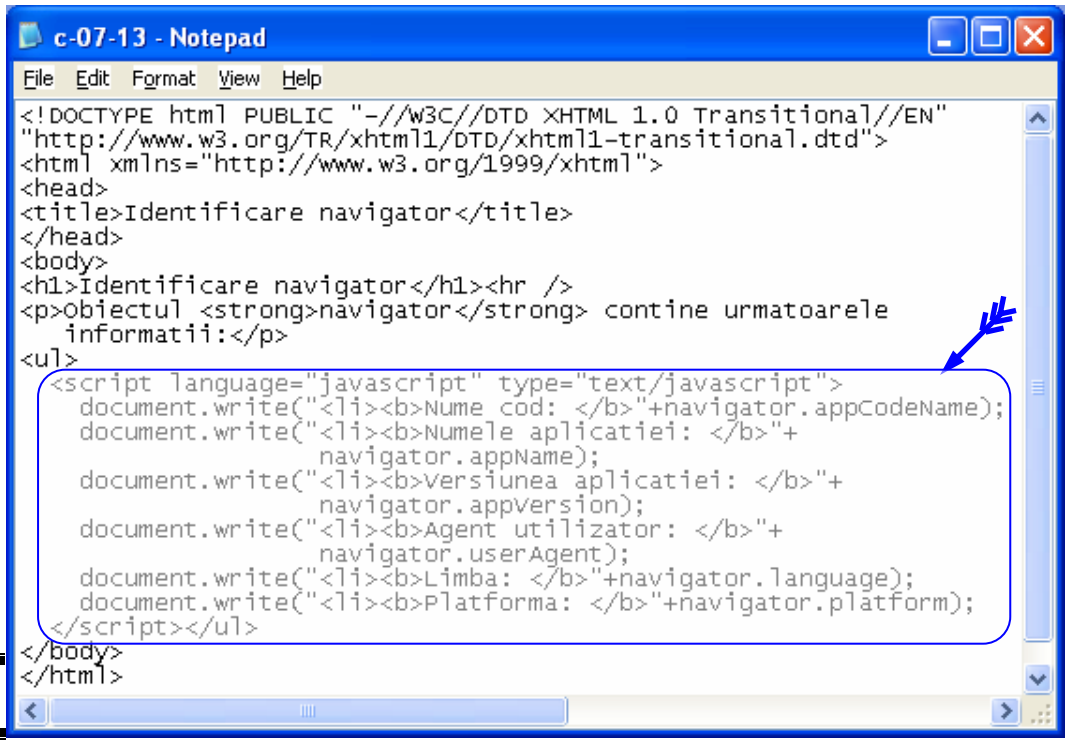


Figura 7.46

În figura 7.47 şi figura 7.48 se prezintă cele două pagini Web afişate în Netscape 6, respectiv Internet Explorer.

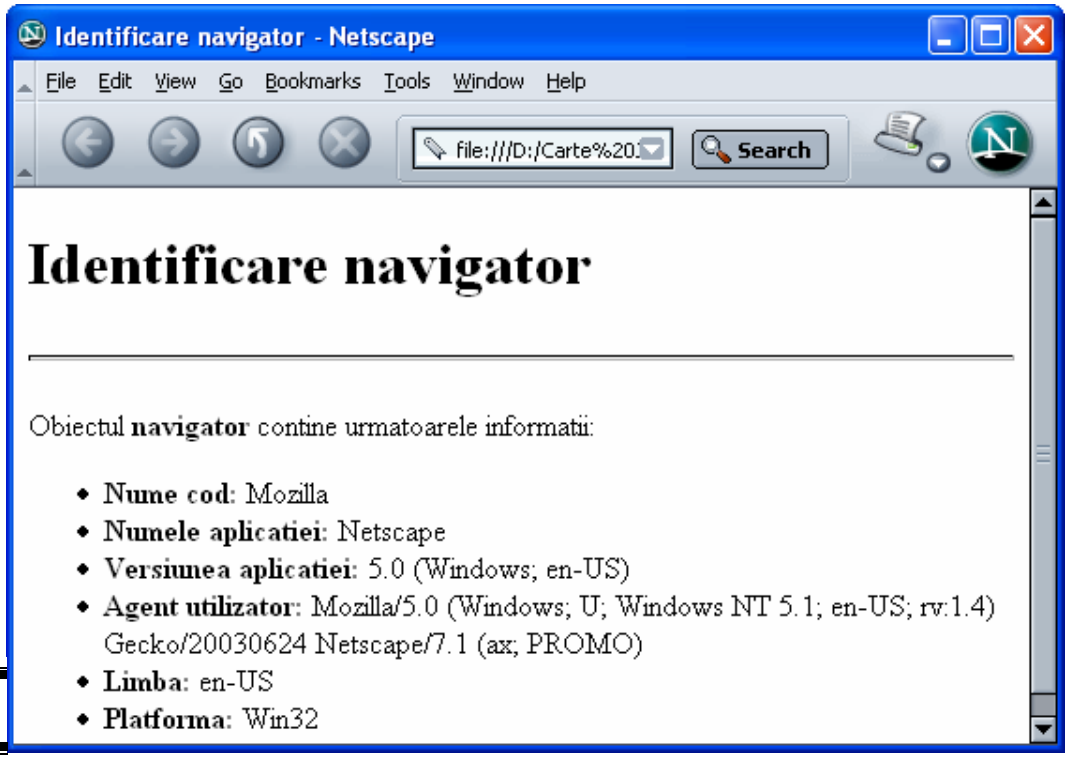


Figura 7.47

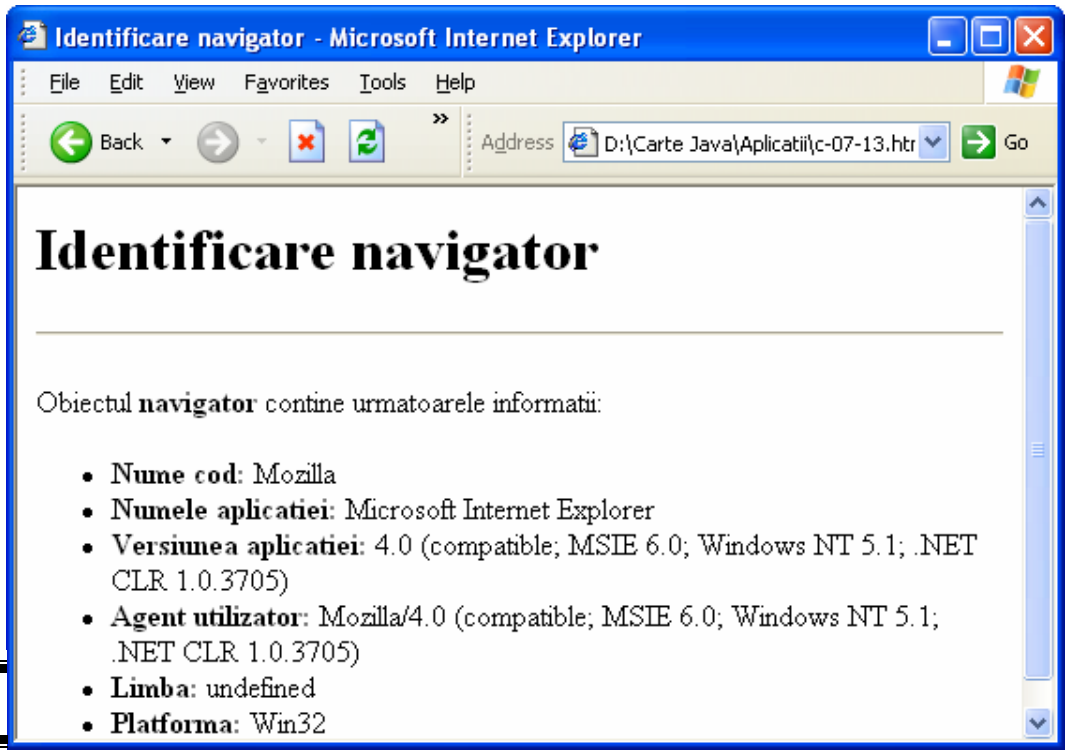


Figura 7.48

Remarcă. Două elemente reţin atenţia:

- proprietatea `navigator.language` este afişată ca nedefinită (`undefined`) în Internet Explorer;
- `Mozilla` apare în câmpurile: `Nume cod` şi `Agent utilizator`.

❑ Scrieţi un script care afişează următoarea pagină Web (figura 7.49).

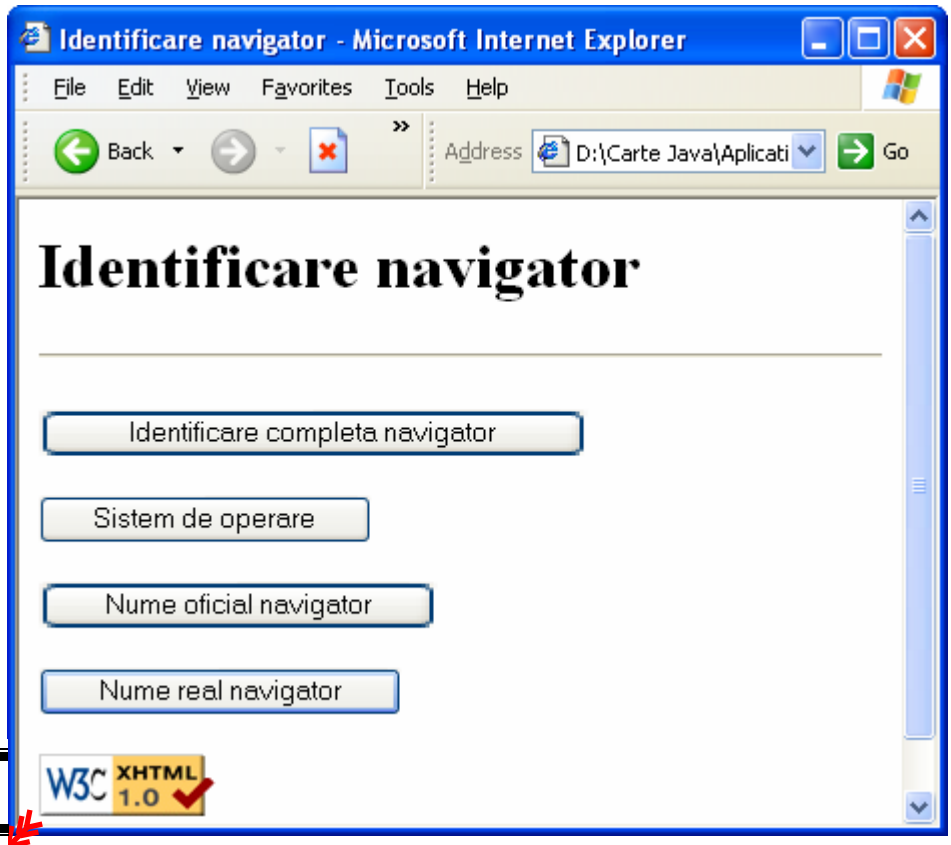


Figura 7.49

Indicaţie. Script-ul afişează patru butoane în pagina Web. Executând clic pe fiecare din cele patru butoane se vor afişa proprietăţile obiectului `Navigator`: `userAgent`; `platform`; `appName`. Folosiţi gestionarul de evenimente `onClick` şi metoda `alert()`.

În figura 7.50 este prezentat documentul XHTML complet în care s-a inserat script-ul aplicaţiei.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Identificare navigator</title>
<script language="javascript" type="text/javascript">
 function idnavig(){
   nav=navigator.userAgent;
            alert(nav);
            }
 function platforma(){
   platf=navigator.platform;
            alert(platf);
            }
 function numenavig(){
   nume=navigator.appName;
            alert(nume);
            }
```

Figura 7.50

```
  function numereal(){
    nav=navigator.userAgent;
    if(nav.indexOf("Opera")>-1)
              {navig="Opera";}
    if(nav.indexOf("Opera")==-1&&nav.indexOf("MSIE 4")>-1)
              {navig="Internet Explorer 4";}
    if(nav.indexOf("Opera")==-1&&nav.indexOf("MSIE 5")>-1)
              {navig="Internet Explorer 5";}
    if(nav.indexOf("Opera")==-1&&nav.indexOf("MSIE 6")>-1)
              {navig="Internet Explorer 6";}
    if(nav.indexOf("Netscape 6")>-1)
              {navig="Netscape 6";}
    if(nav.indexOf("Opera")==-1&&nav.indexOf("MSIE")==-1&&nav.indexOf("Netscape 6")==-1)
              {navig="Netscape 4";}
             alert(navig);
             }
  </script>
</head>

<body>
<h1>Identificare navigator
</h1>
<hr />
<form name="form1" id="form1" method="post" action="">
  <p>
    <input type="button" name="Button" value="Identificare completa navigator"
onClick="idnavig();" />
  </p>
  <p>    <input type="button" name="Submit2" value="Sistem de operare"
              onClick="platforma();" />
  </p>
  <p>    <input type="button" name="Submit3" value="Nume oficial navigator"
onClick="numenavig();" />
  </p>
  <p>  <input type="button" name="Submit4" value="Nume real navigator" onClick="numereal();"
/>  </p>
</form>
<p><img src="vxhtml10.gif" width="88" height="31" /></p>
<ul>
</ul>
</body>
</html>
```

Figura 7.50 *(continuare)*

În figura 7.51, figura 7.52, figura 7.53, figura 7.54 sunt prezentate rezultatele afişării paginii Web în navigatorul Internet Explorer.

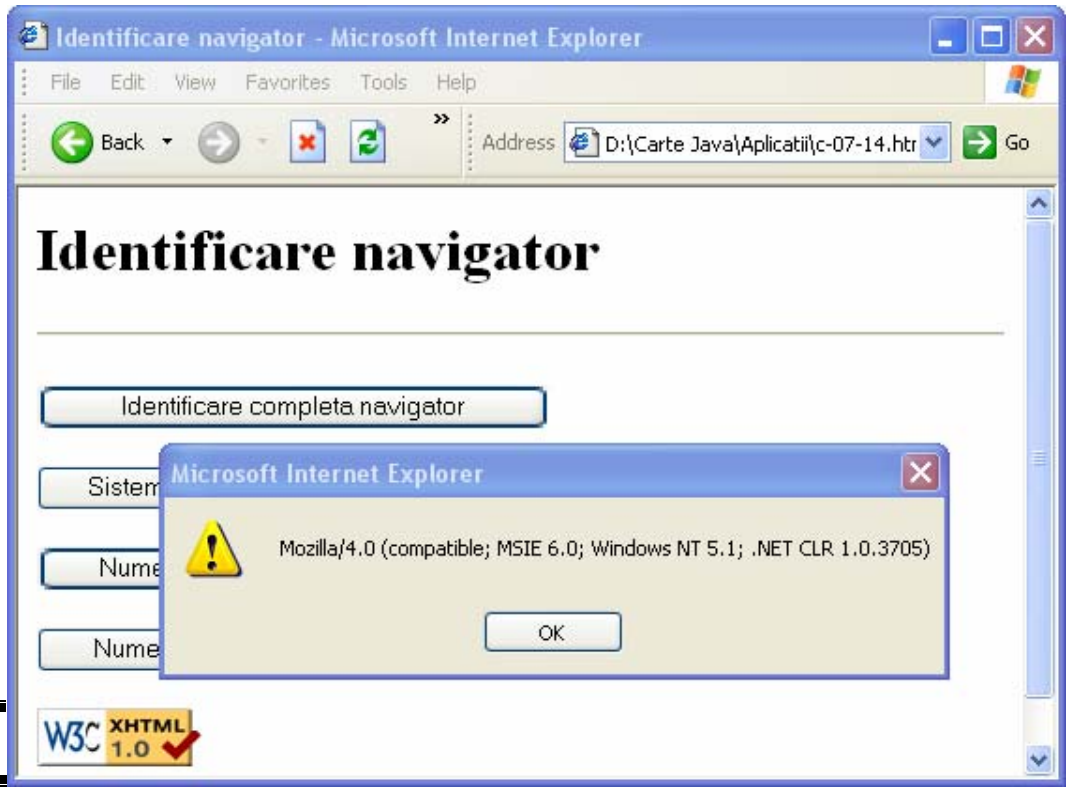


Figura 7.51

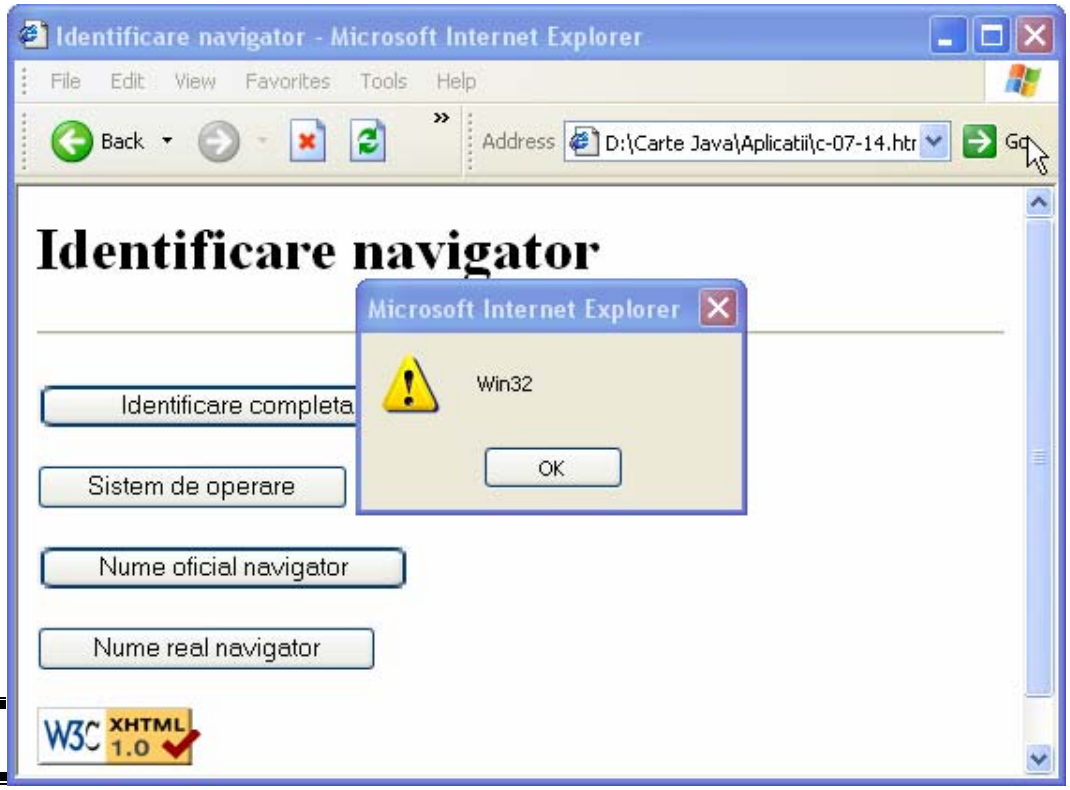


Figura 7.52

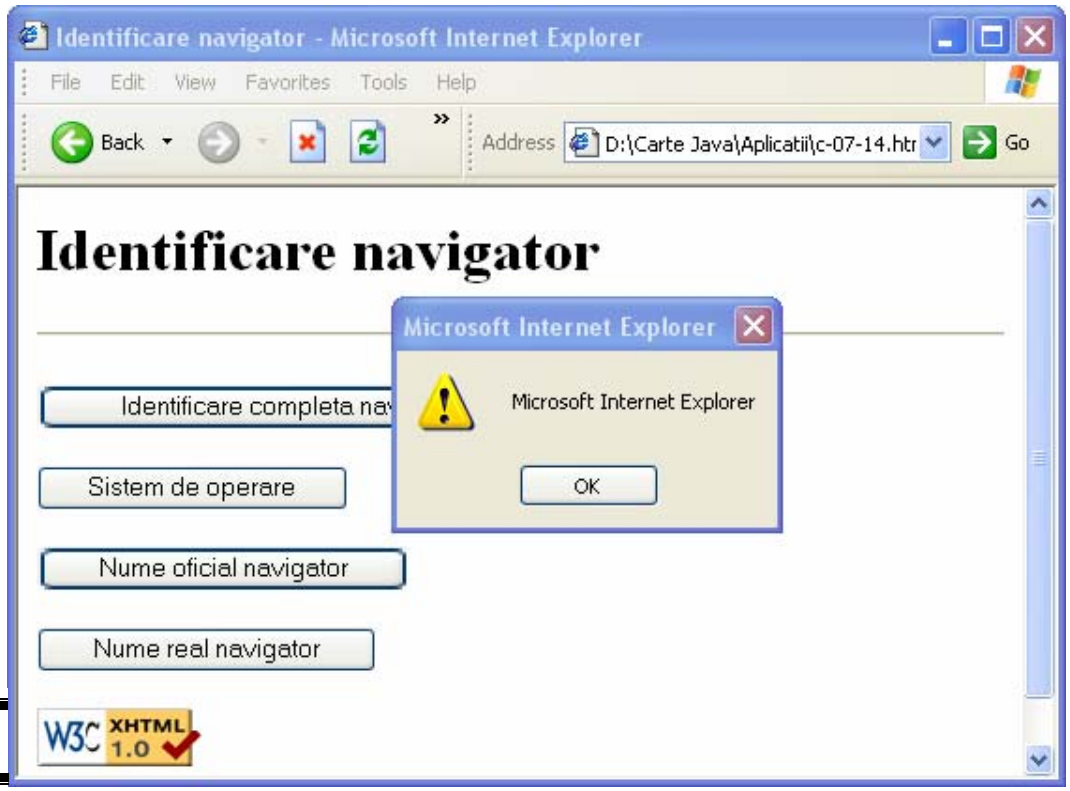


Figura 7.53

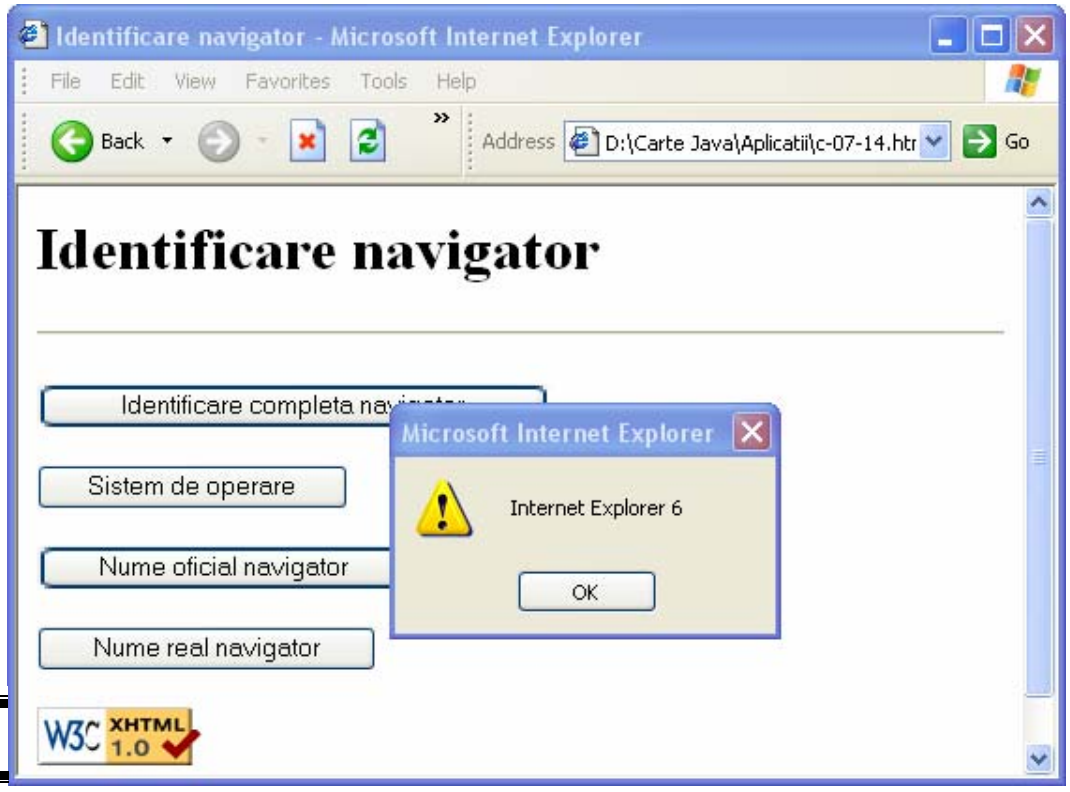


Figura 7.54

❑ Personalizaţi script-ul pe care l-aţi creat în aplicaţia precedentă. Utilizaţi cele patru funcţii numai în script-urile pe care le exploataţi.

Indicaţie. Definiţi gestionarul de evenimente `onLoad` în tag-ul `<body>` după cum urmează: `<body onLoad="idnavig()">`.

❑ Scrieţi o funcţie simplă care să vă permită să recunoaşteţi browser-ul Netscape sau Internet Explorer sau pe nici unul dintre ele.

În figura 7.55 se prezintă o soluţie a aplicaţiei pe care vă rugăm să o comentaţi.

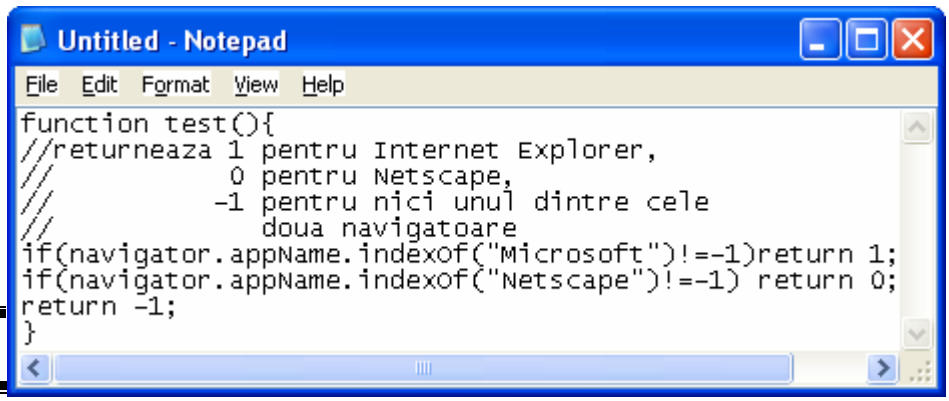


Figura 7.55

❑ Cum procedaţi pentru a detecta navigatorul şi versiunea sa în JavaScript?

Subobiectele obiectului Navigator

Obiectul MimeType

Obiectul `MimeType` este un subobiect al obiectului `Navigator`. El vă permite să accesaţi informaţii despre tipurile `MIME` pe care le recunosc modulele plug-in. Ca şi obiectul `Plugin`, el nu este recunoscut de Internet Explorer.

Instanţele obiectului `MimeType` sunt accesibile prin intermediul matricii `navigator.mimeTypes[]` ale cărei elemente reprezintă tipurile `MIME`.

Fişa obiectului `MimeType` este prezentată în figura 7.56.

Fişa obiectului MimeType

Obiectul părinte:	`Navigator`
Proprietăţi:	`description, enabledPlugin, length, name, suffixes, type`
Metode:	-
Gestionarii de evenimente:	-

Figura 7.56

Obiectul `MimeType` este prezentat în detaliu în figura 7.57.

	Obiect	***Sintaxă***
	`MimeType`	`navigator.mimeTypes[]`
	Face referire la aplicaţiile şi tipurile de fişiere recunoscute de navigator.	
Exemplu:	<script> for(tip in navigator.mimeTypes){ document.write(tip) } </script>	

Figura 7.57

Proprietăţile obiectului MimeType

Proprietăţile obiectului `MimeType` sunt prezentate în detaliu în figura 7.58.

	Proprietate	***Sintaxă***
	`description`	`navigator.MimeTypes[].description`
	Descrierea obiectului `MimeType`.	
Exemplu:	<script> for(i=0; i<navigator.mimeType.length;i++){ tip=navigator.mimeTypes[i].description; //afiseaza descrierea plug-ins } </script>	
	`enabledPlugin`	`navigator.MimeTypes[].enabledPlugin`
	Determină dacă un plug-in este disponibil. Nu este suficient ca un plug-in să fie numai instalat; el trebuie să fie şi disponibil.	
Exemplu:	<script> tip=navigator.mimeTypes["application/pdf"].enablePlugin; // afişează [object plugin] dacă plugin este disponibil </script>	
	`length`	`navigator.MimeTypes.length`
	Numărul de tipuri `MIME` recunoscute.	
Exemplu:	<script> a=navigator.mimeTypes.length document.write(a); /*afisează, de exemplu 7*/ </script>	

Figura 7.58

	`name`	`navigator.MimeTypes[].name`
	Numele plug-ins.	
Exemplu:	<script> for(i=0;i<navigator.mimeTypes.length;i++){ tip=navigator.mimeTypes[i].name; document.write(tip+" "); } </script>	
	`suffixes`	`navigator.MimeTypes[].suffixes`
	Extensia de fişier pentru Mimetype.	
Exemplu:	<script> for(i=0;i<navigator.mimeTypes.length;i++){ tip=navigator.mimeTypes[i].suffixes; document.write(tip+" "); /*afişează wjp, wpg, …*/ } </script>	
	`type`	`navigator.MimeTypes[].type`
	Tipul de fişiere recunoscute de plug-ins.	
Exemplu:	<script> for(i=0;i<navigator.mimeTypes.length;i++){ tip=navigator.mimeTypes[i].type; document.write(tip+" "); //afişează, de exemplu image/x-quicktime }</script>	

Figura 7.58 *(continuare)*

Aplicaţie

❑ Scrieţi un script care afişează tipurile MIME recunoscute de navigator.

Indicaţie. Folosiţi o buclă `for-in`.

În figura 7.59 este prezentat script-ul aplicaţiei.

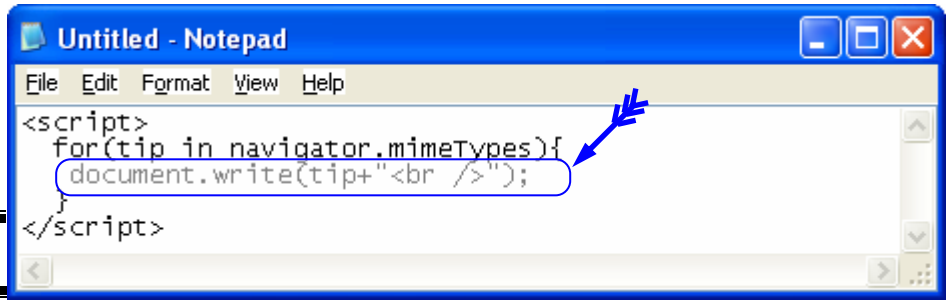


Figura 7.59

Obiectul Plugin

Obiectul `Plugin` este un subobiect al obiectului `Navigator`. El este creat prin instalarea de module plug-in pentru browser. Obiectul `Plugin` conţine o matrice de elemente şi tipuri `MIME` tratate de fiecare modul plug-in.

În figura 7.60 este prezentată fişa obiectului `Plugin`.

Figura 7.60

Fişa obiectului Plugin	
Obiectul părinte:	`Navigator`
Proprietăţi:	`description, filename, length, name`
Metode:	-
Gestionarii de evenimente:	-

Remarci:

- ✓ `description` – face referire la o descriere a modulului plug-in;
- ✓ `filename` – face referire la numele fişierului unui program plug-in;
- ✓ `length` – face referire la numărul de tipuri MIME conţinute în matrice;
- ✓ `name` – face referire la numele modulului plug-in.

Aplicaţie

❑ Folosiţi obiectul `Plugin` cu metoda `document.write` pentru a afişa informaţii despre modulele plug-in instalate.

Manipularea plug-ins cu ajutorul obiectelor

Plug-ins au apărut cu versiunea 3.0 a navigatorului Netscape.

Există sute de plug-in pentru Internet Explorer şi Netscape. Prezentăm în continuare pe cele mai cunoscute:

- ✓ Macromedia Shockwave şi Flash;
- ✓ Adobe Acrobat;
- ✓ Real Player;
- ✓ Beatnik.

Obiectul JavaScript `Navigator` posedă un obiect fiu numit `Plugins`. Acest obiect este o matrice, fiecare plug-in instalat în navigator reprezentând un element al matricii.

Obiectul Plugins

Obiectul `Plugins` face referire la plug-ins instalate în navigator.

Instanţele obiectului `Plugins` sunt accesibile prin matricea `navigator.plugins[]` ale cărei elemente reprezintă plug-ins.

Fiecare plug-in este de asemenea inclus în matricea `MimeTypes[]`.

Remarci:

- ✓ Nu confundaţi obiectele `Plugin`, proprietăţi ale obiectului `Navigator`, cu obiectele `Embed`, proprietăţi ale obiectului `Document`. Primele fac parte din `Navigator` în timp ce următoarele sunt incorporate în document cu ajutorul tag-ului `<object>` (pentru Internet Explorer) sau `<embed>` pentru Netscape.
- ✓ Obiectul `Plugins` nu este recunoscut în mod constant de către toate versiunile navigatoarelor. Pe de altă parte, Internet Explorer recunoaşte plug-ins şi ignoră obiectele `Plugin`.

Fişa obiectului `Plugins` este prezentată în figura 7.61.

Fişa obiectului Plugins

Obiectul părinte:	`Navigator`
Proprietăţi:	`description, filename, length, name`
Metode:	-
Gestionarii de evenimente:	-

Figura 7.61

Aplicaţie

❑ Generaţi lista tipurilor `MIME` recunoscute de către navigatorul Internet Explorer.

În figura 7.62 este prezentat script-ul aplicaţiei.

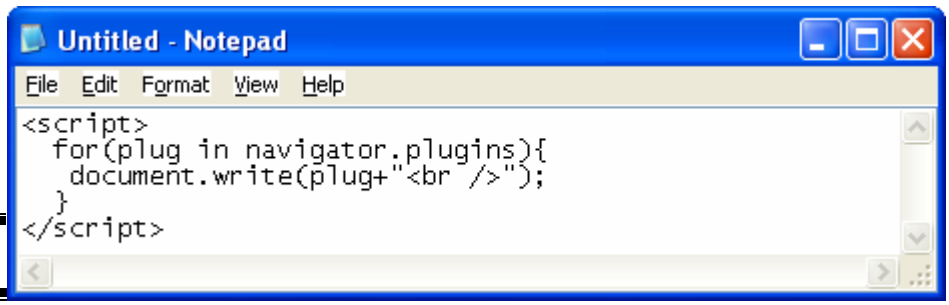


Figura 7.62

Proprietăţile obiectului Plugins

Proprietăţile obiectului `Plugins` sunt prezentate în detaliu în figura 7.63.

Proprietate	Sintaxă
`description`	`navigator.plugins[].description`

Descrierea plug-ins.

Exemplu:

```
<script>
  for(i=0;i<navigator.plugins.length;i++){
    plug=navigator.plugins[i].description;
     document.write(plug+"<br />");
   //afişează descrierea plug-ins
  }
</script>
```

`filename`	`navigator.plugins[].filename`

Numele şi amplasarea fişierelor plug-ins.

```
<script>
  for(i=0;i<navigator.plugins.length;i++){
    plug=navigator.plugins[i].filename;
     document.write(plug+"<br />");
   //afişează numele fişierelor de plug-ins
  }
</script>
```

`length`	`navigator.plugins.length`

Numărul de plug-ins.

Exemplu:

```
<script>
  x=navigator.plugins.length;
  document.write(x);
   //afişează, de exemplu 13
</script>
```

`name`	`navigator.plugins[].name`

Numele plug-ins.

Exemplu:

```
<script>
  for(i=0;i<navigator.plugins.length;i++){
    plug=navigator.plugins[i].name;
     document.write(plug+"<br />");
   //afişează numele plug-ins
  }
</script>
```

Figura 7.63

Remarcă. Obiectului `Navigator` posedă de asemenea un alt obiect fiu numit `MimeTypes`, care conţine câte un element al matricii pentru fiecare tip `MIME`.

Aplicaţie

❑ Analizaţi documentul (X)HTML din figura 7.64. Analizaţi script-ul inserat şi instrucţiunile `document.write()` utilizate. Afişaţi documentul în navigatorul Netscape 6.

Figura 7.64

Rezultatele execuţiei script-ului (figura 7.65).

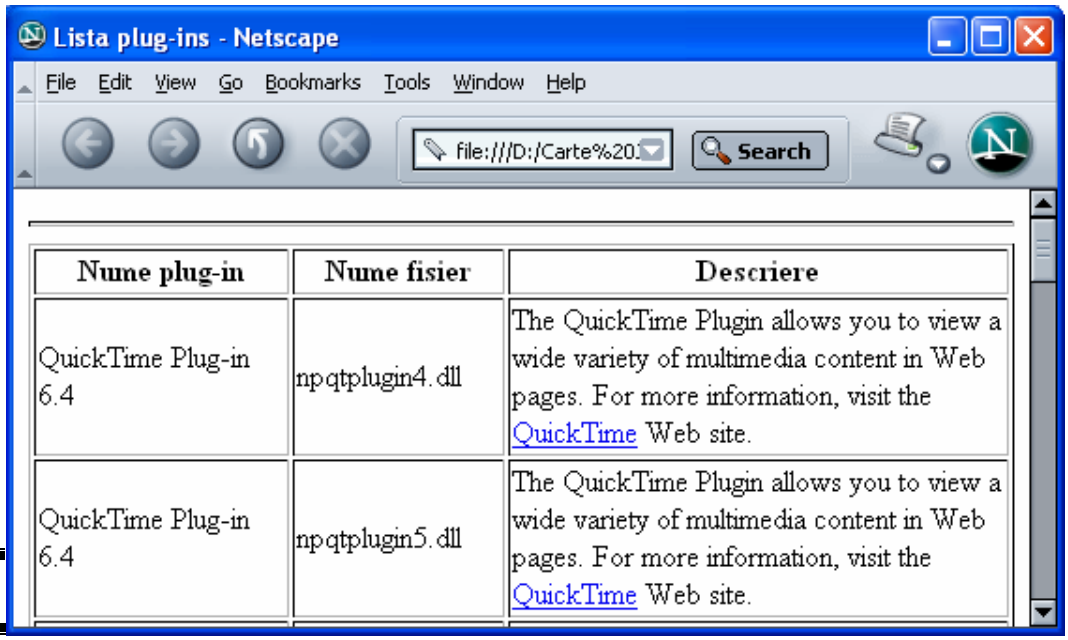
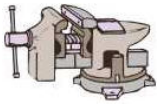


Figura 7.65

Testați-vă cunoștințele:



1. Ce este obiectul `Document`?
2. Care este conţinutul matricii `all[]`?
3. Ce conţine matricea `anchors[]`?
4. Care este conţinutul matricii `applets[]`?
5. Care este conţinutul matricii `forms[]`?
6. Care este conţinutul matricii `frames[]`?
7. Care este conţinutul matricii `layers[]`?
8. Ce semnificaţie are proprietatea `lastModified`?
9. Care este conţinutul matricii `links[]`?
10. Ce este obiectul `Location`?
11. Ce este `MIME`?
12. Care este rolul metodelor `document.open()` şi `document.close()`?
13. Care este deosebirea dintre metodele `document.write()` şi `document.writeln()`?
14. Ce este obiectul `Link`?
15. Ce este obiectul `Layer`?

Vizitați site-urile

- ✓ http://www.hkedcity.net/
- ✓ http://webdesign.about.com
- ✓ http://www.useit.com
- ✓ http://www.bigonion.com

Conversația 8

Obiectul Form. Validarea formularelor

În această conversație:

Obiectul Form

Momentul apariţiei formularelor (X)HTML a fost considerat ca decisiv în evoluţia Web-ului.

Formularele, structurate în câmpuri şi zone de date sunt ideale pentru a realiza o interactivitate între dumneavoastră şi vizitatorii site-ului pe care l-aţi creat.

O dată cu apariţia limbajului JavaScript puterea formularelor (X)HTML a crescut şi mai mult.

Cu JavaScript puteţi modifica în mod dinamic conţinutul elementelor unui formular (zonă simplă de text, zonă de text multilinie, casetă de validare etc.).

JavaScript vă permite să adăugaţi noi funcţii de prelucrare a formularelor dumneavoastră (X)HTML, atât pe partea de client cât şi pe partea de server.

În concluzie, crearea unui site Web cu adevărat performant presupune cunoaşterea atât a facilităţilor formularelor (X)HTML cât şi a obiectului `Form` al limbajului JavaScript.

Fişa obiectului `Form` este prezentată în figura 8.1.

Fişa obiectului Form	
Obiectul părinte:	`Document`
Subobiecte:	`button, checkbox, fileupload, hidden, input, password, select, option, radio, reset, text, textarea, submit`
Proprietăţi:	`acceptCharset, action, elements[], encoding, enctype, length, method, name, target`
Metode:	`reset(), submit(), tags()`
Gestionarii de evenimente:	`onContextMenu, onControlSelect, onCopy, onCut, onDblClick, onDeActivate, onDrag, onDragend, onDragenter, onDragLeave, onDragOver, onDragStart, onFocusOut, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseEnter, onMouseLeave, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onMousewheel, onMove, onMoveend, onMovestart, onPaste, onPropertyChange, onReset, onResize, onResizeEnd, onResizeStart, onSelectStart, onSubmit` (vezi Conversaţia 6).

Figura 8.1

Relaţia dintre obiectul Form şi tag-ul <form>

Fiecare formular care aparţine unei pagini (X)HTML este reprezentat în JavaScript printr-un obiect `Form`, al cărui nume este identic cu atributul `name` al tag-ului `<form>`.

Aşadar, obiectul `Form` este un obiect JavaScript reprezentat prin perechea de tag-uri (X)HTML: `<form>` şi `</form>`.

Remarci:

- ✓ Obiectul `Document` conţine o matrice de obiecte `Form` care poartă numele de `forms`.
- ✓ După încărcarea şi analiza documentului (X)HTML, această matrice (forms) va conţine câte un obiect `Form` pentru fiecare pereche de tag-uri `<form>` şi `</form>` prezentată în documentul (X)HTML.

Documentul (X)HTML din figura 8.2 conţine trei obiecte `Form`.

Figura 8.2

Deoarece matricea `forms` este o proprietate a obiectului `Document` puteţi referi cele trei obiecte `Form` după cum urmează (figura 8.3).

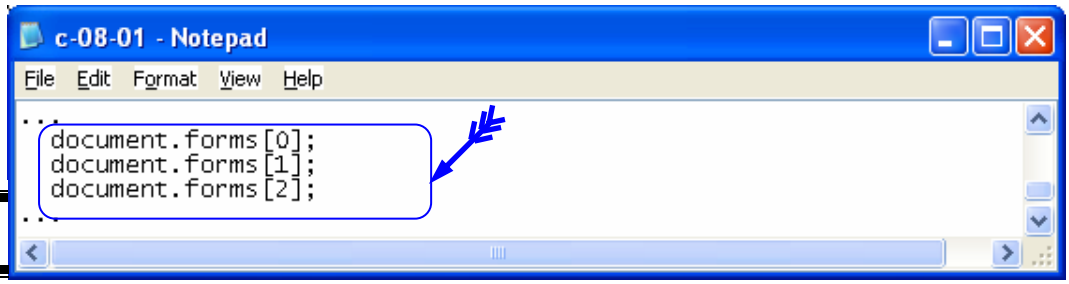


Figura 8.3

Dacă nu agreaţi referinţele cu numere (forms[0], forms[1], ...) puteţi atribui un nume fiecărui obiect `Form` (vezi figura 8.4).

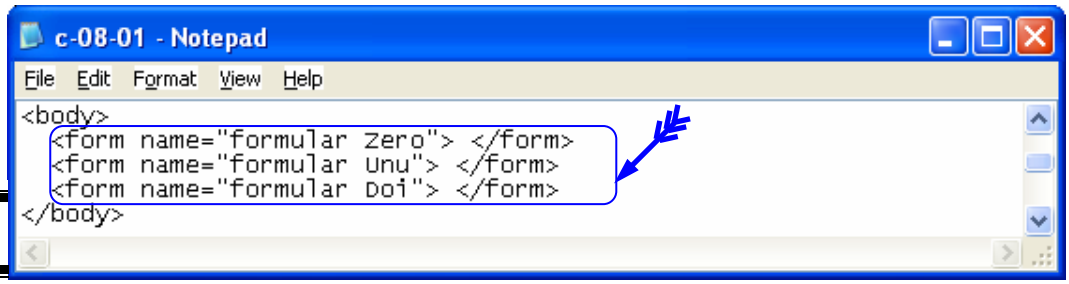


Figura 8.4

Atunci când documentul (X)HTML va fi analizat, obiectele `Form` ale limbajului JavaScript vor avea proprietatea `name` atribuită în mod automat cu numele: formularZero; FormularUnu; formularDoi. Puteţi accesa cele trei obiecte `Form`, după cum urmează (figura 8.5).

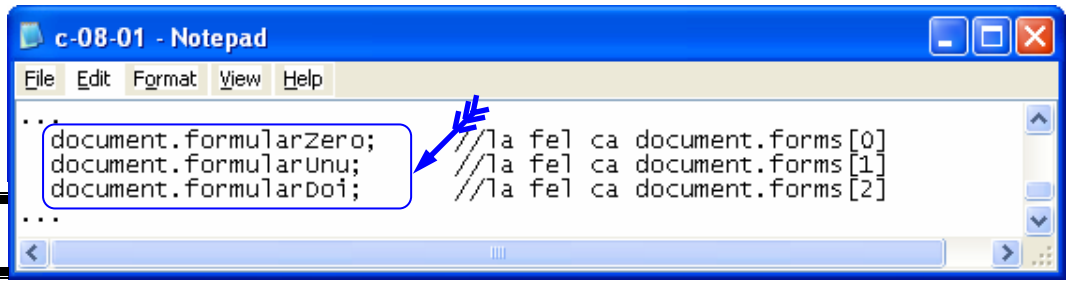


Figura 8.5

Remarci:

- ✓ Este important să observaţi că numele atribuite obiectelor `Form` sunt în totalitate nume de variabile JavaScript valide (formularZero; formularUnu; formularDoi).
- ✓ (X)HTML nu impune nici o limită valorilor posibile pentru un atribut; în consecinţă, numele celor trei atribute `name` (conţin spaţii) din figura 8.6 sunt perfect valabile.

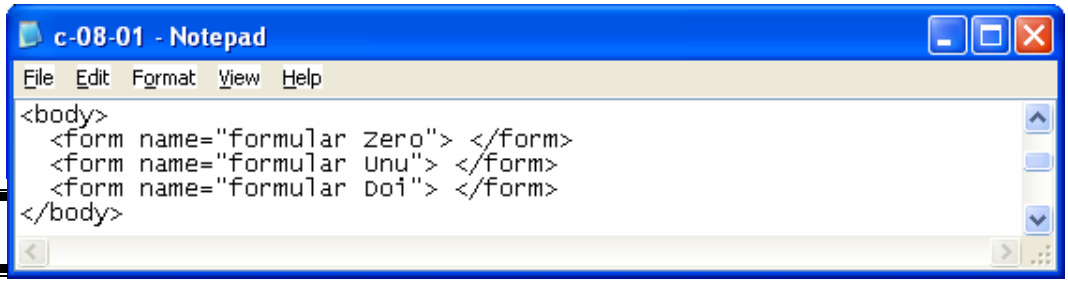


Figura 8.6

- ✓ Dacă dorim să referim aceste nume în JavaScript, vom scrie următorul cod JavaScript (figura 8.7).

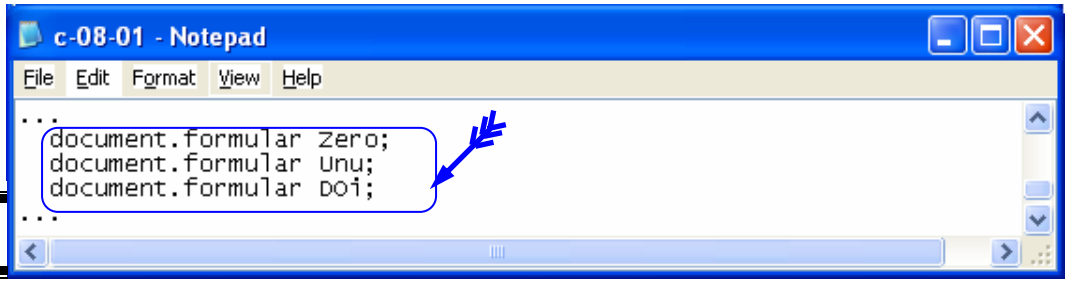


Figura 8.7

- ✓ Din nefericire, nici una din cele trei instrucţiuni nu este validă (!) deoarece JavaScript nu va şti să trateze spaţiul (blank-ul) din mijlocul fiecărui nume. Aşadar, singurul mod de a accesa obiectele `Form` este acela de a utiliza matricea `forms`.
- ✓ Pentru a atribui un nume elementelor ce aparţin documentelor (X)HTML va trebui să folosiţi nume valide pentru variabilele JavaScript.

Relaţia dintre elementele (obiectele) Form şi obiectul Form

Proprietatea cea mai importantă a obiectului `Form` este `elements` care conţine un obiect pentru fiecare din elementele formularului. Proprietatea `elements` poate fi folosită fie pentru referirea unui element (obiect) dintr-un formular, fie pentru validarea tuturor elementelor dintr-un formular.

Aplicaţie

❑ Analizaţi documentul (X)HTML din figura 8.8. Scrieţi instrucţiunile JavaScript pentru accesul la cele trei obiecte (`Text`) ale formularului.

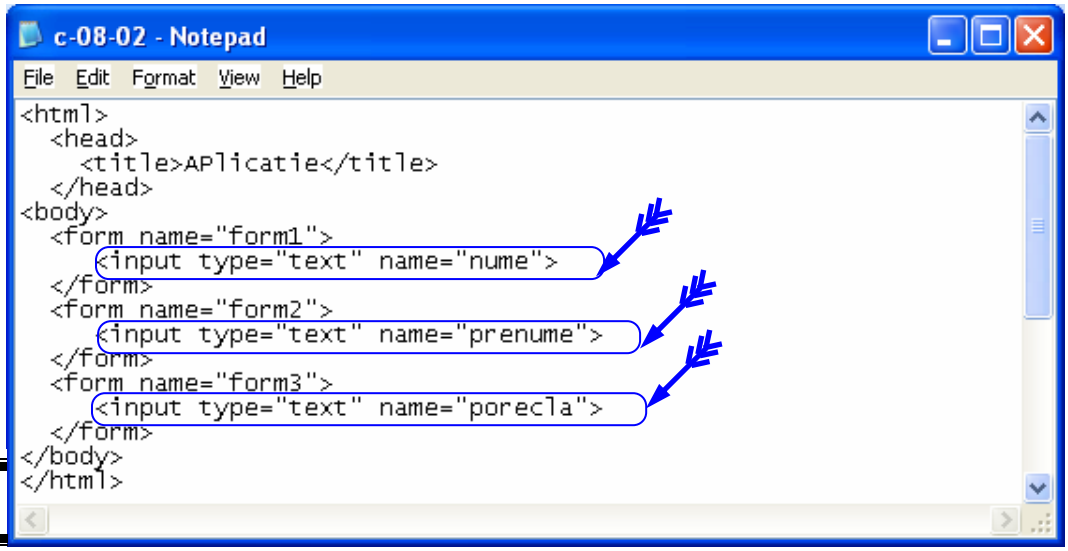


Figura 8.8

Pentru a avea acces la toate cele trei obiecte `text` ale formularului, putem scrie (fără efort!) următorul cod JavaScript (figura 8.9).

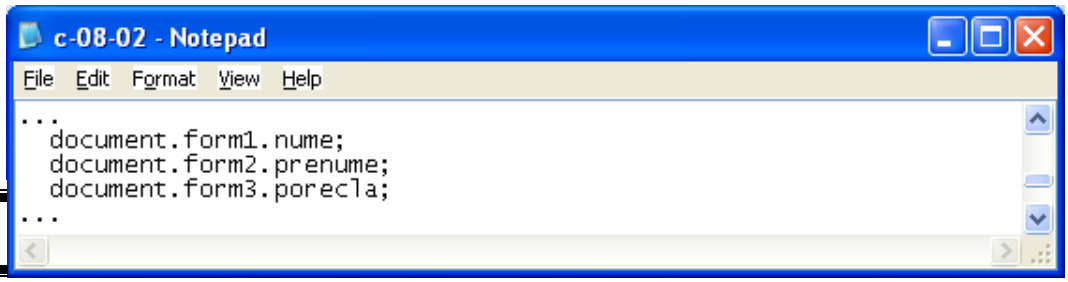


Figura 8.9

Remarcă. Am specificat un atribut `name` pentru fiecare obiect `Text` (ca şi în cazul formularelor).

Să presupunem că doriţi să încărcaţi documentul (X)HTML din figura 8.8 într-un browser, iar apoi introduceţi numele dumneavoastră (*exemplu,* Popescu) în zona simplă de text *nume*. Cum procedaţi pentru a extrage (afişa) această valoare? Răspunsul nu este complicat! Cu mici diferenţe, toate obiectele dintr-un formular respectă acelaşi format de bază; în particular ele posedă proprietatea `value` care conţine valoarea de control curentă. Conceptul de valoare de control depinde de tipul de control; în cazul unei zone simple de text, proprietatea `value` va conţine şirul de caractere curent în obiectul afişat în fereastra navigatorului. Revenind la aplicaţia noastră, pentru a afişa numele pe care l-aţi introdus, scrieţi codul (figura 8.10):

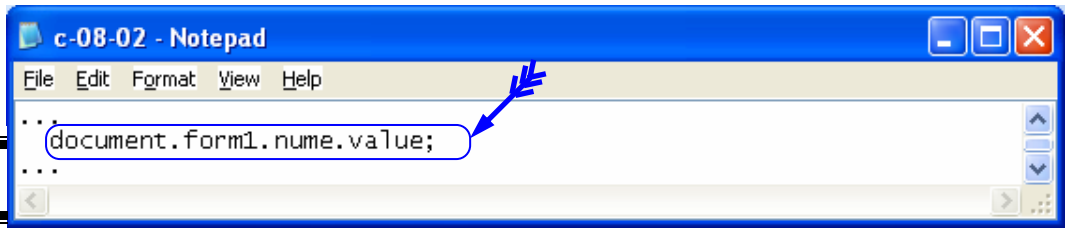


Figura 8.10

În mod analog, pentru a defini numele dumneavoastră, scrieţi codul (figura 8.11):

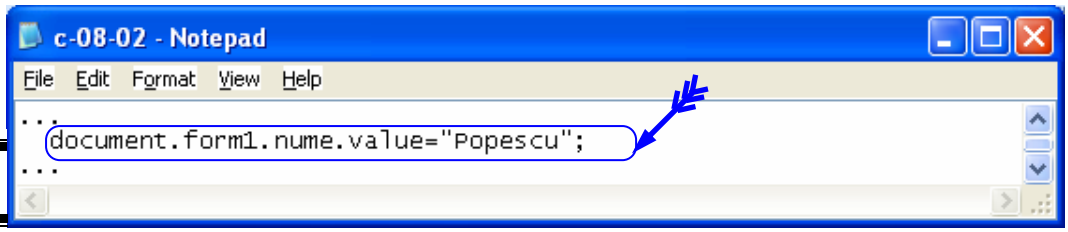


Figura 8.11

Remarcă. Fiecare obiect JavaScript al unui formular conţine propriile proprietăţi şi metode care permit şi alte acţiuni în afară de citirea şi definirea valorilor acestora.

Matricea elements

Este agreabil să poţi avea acces la un element (obiect) al formularului direct prin numele său, dar uneori dorim să prelucrăm în mod automat obiectele formularului. În plus, obiectele nu au tot timpul un nume, sau numele lor nu este valid pentru o variabilă JavaScript. Pentru astfel de situaţii, veţi putea folosi matricea `elements` a obiectului `Form`.

Există două variante de acces la elementele matricii:

- ✓ prin numărul de index;
- ✓ prin nume.

Iată cum procedăm pentru a accesa prin numărul de index, elementele formularului din figura 8.12.

Varianta 1

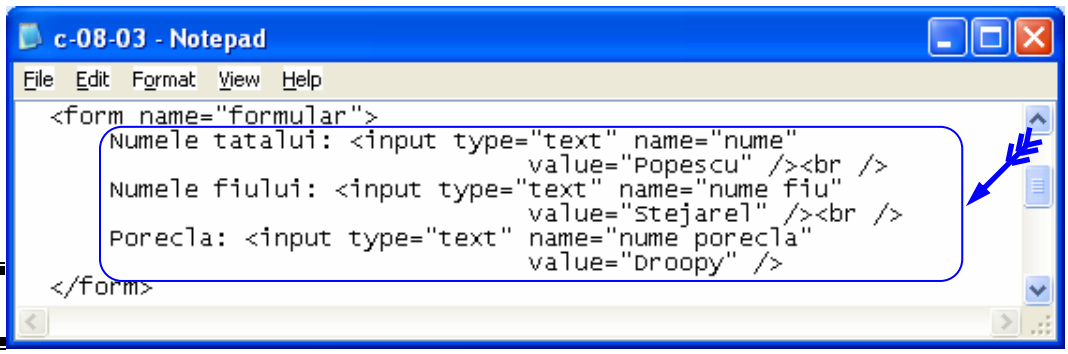


Figura 8.12

Remarcă. În acest formular am definit trei obiecte `Text`. Ultimele două obiecte au un nume care nu este un nume de variabilă JavaScript valid.

În figura 8.13 este prezentat codul sursă JavaScript pentru *Varianta 1*.

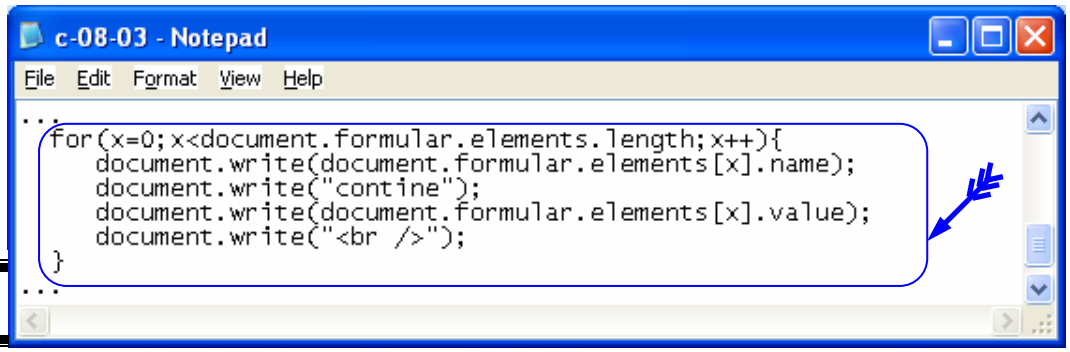


Figura 8.13

Varianta 2

În figura 8.14 este prezentat codul sursă JavaScript pentru *Varianta 2*.

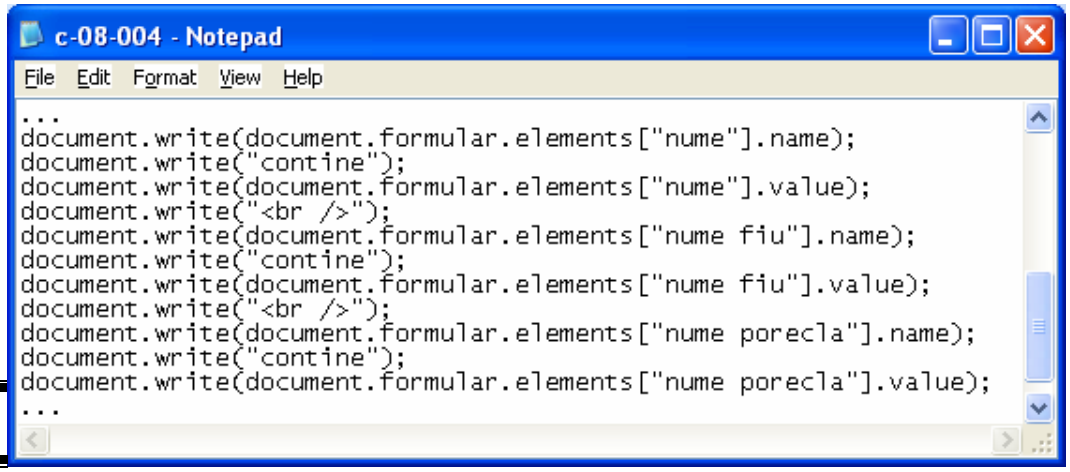


Figura 8.14

Proprietăţile obiectului Form

Proprietăţile obiectului `Form` sunt prezentate în detaliu în figura 8.15.

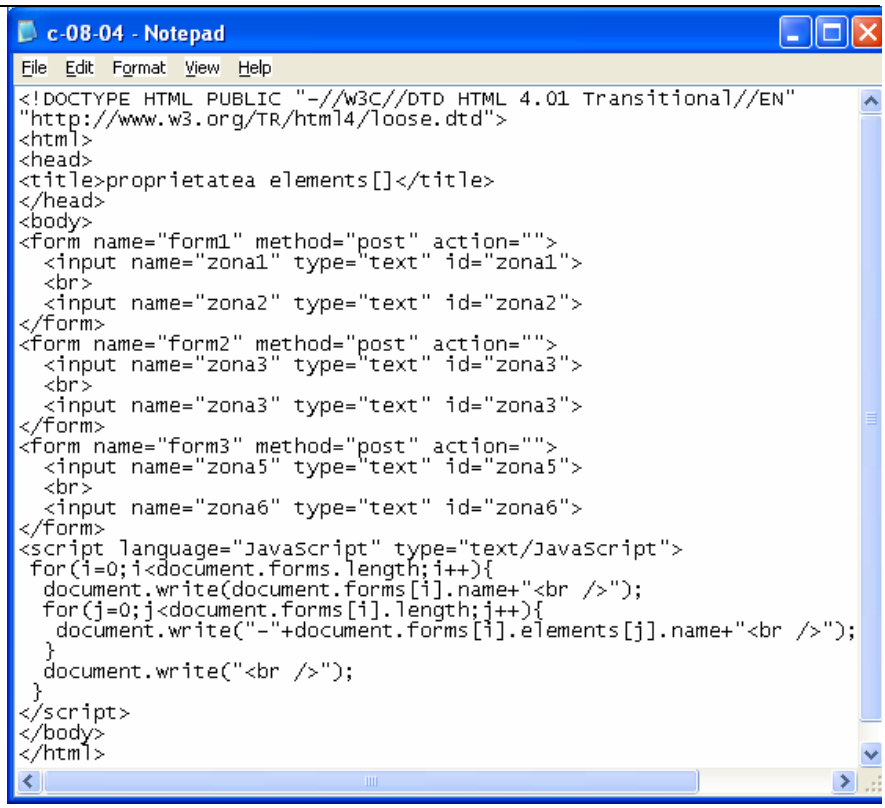
	Proprietate	*Sintaxă*
	`acceptCharset`	`document.form.acceptCharset`
📖	Setul de caractere acceptat de către server-ul care prelucrează datele.	
	`action`	`document.form.action`
📖	Corespunde atributului `action` al tag-ului `<form>`.	
	`elements[]`	`document.form.elements[]`

Figura 8.15

Elementele unui formular (zone de text şi butoane). Fiecare element este o dată a matricii (array) `elements[]`. Ele sunt referite prin numărul de ordine (*exemplu:* elements[2]) sau prin numele lor (*exemplu:* elements["zona1"]).

Proprietatea `length` returnează numărul de elemente: `form.elements.length`. Un formular poate conţine următoarele tipuri de elemente: `button`, `checkbox`, `fileUpload`, `hidden`, `input`, `password`, `select`, `option`, `radio`, `reset`, `text`, `textarea`. Fiecărui element îi corespunde un obiect.

Exemplu:



`encoding`	`document.form.encoding`

Corespunde atributului `encoding` al tag-ului `<form>`.

`enctype`	`document.form.enctype`

Corespunde atributului `enctype` al tag-ului `<form>`.

`length`	`document.form.length`

Numărul de elemente ale formularului.

`method`	`document.form.method`

Corespunde atributului `method` al tag-ului `<form>`.

`name`	`document.form.name`

Figura 8.15 *(continuare)*

	Corespunde atributului `name` al tag-ului `<form>`.
`target`	`document.form.target`
	Corespunde atributului `target` al tag-ului `<form>`.

Figura 8.15 *(continuare)*

Metodele obiectului Form

Metodele obiectului `Form` sunt prezentate în detaliu în figura 8.16.

Metodă	***Sintaxă***
`reset()`	`document.form.reset()`
Corespunde acţiunii butonului `reset`.	
`submit()`	`document.form.submit()`
Corespunde acţiunii butonului `submit`.	
`tags()`	`document.form.tags(tag)`
Recuperează toate elementele introduse cu tag-ul menţionat ca argument.	

Figura 8.16

Subobiectele Form

După cum am precizat în Conversaţia anterioară, toate obiectele de nivelul al treilea, pe parte de client sunt subobiecte ale obiectului `Form`: `Button`, `Checkbox`, `FileUpload`, `Hidden`, `Password`, `Radio`, `Reset`, `Submit`, `Select`, `Text`, `Textarea`.

Vom prezenta în continuare, în ordine alfabetică toate subobiectele obiectului `Form`.

(sub)Obiectul Button

Fişa obiectului (subobiectului) `Button` este prezentată în figura 8.17.

Fişa (sub)obiectului Button	
Obiectul părinte:	`Form`
Proprietăţi:	`align, defaultValue, disabled, form, name, size, type, value`
Metode:	`blur(), click(), focus()`
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUpdate, onBlur, onClick, onDblClick, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp` (vezi Conversaţia 6).

Figura 8.17

(sub)Obiectul `Button` este prezentat în detaliu în figura 8.18.

	Obiect	*Sintaxă*
	`Button`	`document.forms[].button`
	Reprezintă un obiect generic într-un formular. Subobiectul `Button` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<input type="button">`. A nu se confunda cu butoanele `reset` şi `submit`. Subobiectul `Button` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	
Exemplu:	<form name="form1"> <input type="button" name="Button1" value="calcul"/> </value>	

Figura 8.18

Proprietăţile (sub)obiectului Button

Proprietăţile (sub)obiectului `Button` sunt prezentate în detaliu în figura 8.19.

Metodă	Sintaxă
align	document.forms[].button.align
Alinierea butonului. Proprietatea poate conţine valorile: left, center, right.	
defaultValue	document.forms[].button.defaultValue
Valoarea implicită a butonului.	
disabled	document.forms[].button.disabled
Valoarea logică (true/false) care indică starea butonului: dezactivat (true), activat (false).	
form	document.forms[].button.form
Referinţă la formularul care conţine butonul.	
name	document.forms[].button.name
Numele butonului. Corespunde atributului name al tag-ului <input>.	
size	document.forms[].button.size
Dimensiunea (în pixeli) butonului. Corespunde atributului size al tag-ului <input>.	
type	document.forms[].button.type
Tipul de element în cadrul formularului. În acest caz, proprietatea returnează button. Corespunde atributului type al tag-ului <input>.	
value	document.forms[].button.value
Valoarea butonului. Corespunde atributului value al tag-ului <input>.	

Figura 8.19

Metodele obiectului Button

Metodele obiectului Button sunt prezentate în detaliu în figura 8.20.

Metodă	Sintaxă
blur()	document.form.button.blur()
Butonul pierde focus-ul.	
click()	document.form.button.click()
Simularea clic-ului mouse-ului pe buton. Această metodă nu este detectată de onClick.	
focus()	document.form.button.focus()
Butonul preia focus-ul.	

Figura 8.20

(sub)Obiectul Checkbox

Fişa (sub)obiectului `Checkbox` este prezentată în figura 8.21.

Fişa (sub)obiectului Checkbox	
Obiectul părinte:	`Form`
Proprietăţi:	`align, checked, defaultChecked, defaultValue, disabled, form, name, size, status, type, value, width`
Metode:	`blur(), click(), focus()`
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUpdate, onBlur, onClick, onDblClick, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp` (vezi Conversaţia 6).

Figura 8.21

Obiectul `Checkbox` este prezentat în detaliu în figura 8.22.

Obiect	***Sintaxă***
`checkbox`	`document.forms[].checkbox`

Reprezintă o casetă de validare în formular. Subobiectul `Checkbox` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<input type="checkbox">`. Subobiectul `Checkbox` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.

Exemplu:

```
<head>
<script>
  function calcule() {
    document.calcul.elements[0].value="0";
    for(i=0;i<document.calcul.elements.length;i++) {
      if(document.calcul.elements[0].value=
                parseInt(document.calcul.elements[0].value)
                 +parseInt(document.calcul.elements[i].value)
        }
    }
</script>
</head>
<form name="calcul">
TOTAL:<input type="text" size="9"/> euro <br />;
<input type="checkbox" value="2" onClick="calcule()"/>Articolul2<br />
<input type="checkbox" value="3" onClick="calcule()"/>Articolul3<br />
<input type="checkbox" value="4" onClick="calcule()"/>Articolul4<br />
<input type="checkbox" value="5" onClick="calcule()"/>Articolul5<br />
</form>
</body>
</html>
```

Figura 8.22

Proprietăţile obiectului Checkbox

Proprietăţile obiectului `Checkbox` sunt prezentate în detaliu în figura 8.23.

Proprietate	*Sintaxă*
`align`	`document.forms[].checkbox.align`
Alinierea casetei de validare. Proprietatea poate conţine valorile: `left`, `center` sau `right`.	
`checked`	`document.forms[].checkbox.checked`
Valoare logică (true/false) care indică starea casetei de validare: activată (`true`) sau dezactivată (`false`).	
`defaultchecked`	`document.forms[].checkbox.defaultchecked`
Valoarea implicită a casetei de validare (`true` sau `false`).	
`defaultValue`	`document.forms[].checkbox.defaultValue`
Valoarea implicită a casetei de validare.	
`disabled`	`document.forms[].checkbox.disabled`
Valoare logică (`true`/`false`) care indică starea casetei de validare: dezactivată (`true`) sau activată (`false`).	
`form`	`document.forms[].checkbox.form`
Referinţă la formularul care conţine caseta de validare.	
`name`	`document.forms[].checkbox.name`
Numele casetei de validare; corespunde atributului `name` al tag-ului `<input>`.	
`size`	`document.forms[].checkbox.size`
Dimensiunea (în pixeli) a casetei de validare. Corespunde atributului `size` al tag-ului `<input>`.	
`status`	`document.forms[].checkbox.status`
Valoare logică (`true`/`false`) care indică starea casetei de validare: activată (`true`) sau dezactivată (`false`).	
`type`	`document.forms[].checkbox.type`
Tipul de element în cadrul formularului. În acest caz, proprietatea returnează `checkbox`. Corespunde atributului `type` al tag-ului `<input>`.	
`value`	`document.forms[].checkbox.value`
Valoarea casetei de validare. Corespunde atributului `value` al tag-ului `<input>`.	
`width`	`document.forms[].checkbox.width`
Dimensiunea (în pixeli) casetei de validare.	

Figura 8.23

Metodele (sub)obiectului Checkbox

Metodele (sub)obiectului `Checkbox` sunt prezentate în detaliu în figura 8.24.

	Metodă	***Sintaxă***
	`blur()`	`document.form.checkbox.blur()`
	Caseta de validare pierde focus-ul.	
	`click()`	`document.form.checkbox.click()`
	Simularea clic-ului mouse-ului pe caseta de validare. Această metodă nu este detectată de `onClick`.	
	`focus()`	`document.form.checkbox.focus()`
	Caseta de validare preia focus-ul.	

Figura 8.24

(sub)Obiectul FileUpload

Fişa (sub)obiectului `FileUpload` este prezentată în figura 8.25.

Fişa (sub)obiectului FileUpload	
Obiectul părinte:	`Form`
Proprietăţi:	`defaultValue, disabled, form, name, size, type, value, width`
Metode:	`blur(), focus(), select()`
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUpdate, onBlur, onChange, onDragStart, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onResize, onSelect, onSelectStart` (vezi Conversaţia 6).

Figura 8.25

(sub)Obiectul `FileUpload` este prezentat în detaliu în figura 8.26.

	Obiect	***Sintaxă***
	`fileUpload`	`document.forms[].fileUpload`
	Reprezintă o zonă de text în care utilizatorul poate introduce numele unui fişier care urmează a fi transmis server-ului. Subobiectul `fileUpload` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<input type="file">`. Subobiectul `fileUpload` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	

Figura 8.26

Figura 8.26 *(continuare)*

Exemplu:
```
<form name="form" >
    <input type="file" name="fişier1"/>
</form>
```

Proprietăţile (sub)obiectului FileUpload

Proprietăţile (sub)obiectului `FileUpload` sunt prezentate în detaliu în figura 8.27.

Proprietate	*Sintaxă*
`defaultValue`	`document.forms[].fileUpload.defaultValue`
Conţinutul implicit al zonei.	
`disabled`	`document.forms[].fileUpload.disabled`
Valoare logică (`true/false`) care indică starea zonei: dezactivată (`true`) sau activată (`false`).	
`form`	`document.forms[].fileUpload.form`
Referinţă la formularul care conţine zona.	
`name`	`document.forms[].fileUpload.name`
Numele zonei. Corespunde atributului `name` al tag-ului `<input>`.	
`size`	`document.forms[].fileUpload.size`
Dimensiunea (în pixeli) zonei. Corespunde atributului `size` al tag-ului `<input>`.	
`type`	`document.forms[].fileUpload.type`
Tipul de element în cadrul formularului. În acest caz proprietatea returnează `file`. Corespunde atributului `type` al tag-ului `<input>`.	
`value`	`document.forms[].fileUpload.value`
Conţinutul zonei. Corespunde atributului `value` al tag-ului `<input>`.	
`width`	`document.forms[].fileUpload.width`
Dimensiunea (în pixeli) casetei de validare.	

Figura 8.27

Metodele (sub)obiectului FileUpload

Metodele (sub)obiectului `FileUpload` sunt prezentate în detaliu în figura 8.28.

	Metodă	***Sintaxă***
	`blur()`	`document.form.fileUpload.blur()`
	Zona pierde focus-ul.	
	`focus()`	`document.form.fileUpload.focus()`
	Zona preia focus-ul.	
	`select()`	`document.form.fileUpload.select()`
	Selectează conţinutul zonei.	

Figura 8.28

(sub)Obiectul Hidden

Fişa (sub)obiectului `Hidden` este prezentată în figura 8.29.

Fişa (sub)obiectului Hidden	
Obiectul părinte:	`Form`
Proprietăţi:	`defaultValue, disabled, form, name, readOnly, size, type, value`
Metode:	-
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUpdate, onHelp` (vezi Conversaţia 6)

Figura 8.29

(sub)Obiectul `Hidden` este prezentat în detaliu în figura 8.30.

	Obiect	***Sintaxă***
	`hidden`	`document.forms[].hidden`
	Reprezintă o zonă de text ascunsă pentru utilizator. Subobiectul `Hidden` este versiunea „transpusă în obiect” a tag-ului (X)HTML `<input type="hidden">`. Subobiectul `Hidden` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	
Exemplu:	<form name="form1"> <input type="hidden" name="Ascuns"/> </form>	

Figura 8.30

Proprietăţile (sub)obiectului Hidden

Proprietăţile (sub)obiectului `Hidden` sunt prezentate în detaliu în figura 8.31.

	Proprietate	***Sintaxă***

defaultValue	document.forms[].hidden.defaultValue
Conţinutul implicit al zonei.	
disabled	document.forms[].hidden.disabled
Valoare logică (true/false) care indică starea zonei: dezactivată (true) sau activată (false).	
form	document.forms[].hidden.form
Referinţă la formularul care conţine zona.	
name	document.forms[].hidden.name
Numele zonei. Corespunde atributului name al tag-ului <input>.	
readOnly	document.forms[].hidden.readOnly
Valoare logică (true/false) care indică starea zonei.	
size	document.forms[].hidden.size
Dimensiunea (în pixeli) zonei. Corespunde atributului size al tag-ului <input>.	
type	document.forms[].hidden.type
Tipul de element în cadrul formularului. În acest caz proprietatea returnează text.	
value	document.forms[].hidden.value
Conţinutul zonei. Corespunde atributului value al tag-ului <input>.	

Figura 8.31

(sub)Obiectul Password

Fişa (sub)obiectului Password este prezentată în figura 8.32.

Fişa (sub)obiectului Password

Obiectul părinte:	Form
Proprietăţi:	defaultValue, disabled, form, maxLength, name, readOnly, size, type, value
Metode:	select(), blur(), click(), focus()
Gestionarii de evenimente:	onAfterUpdate, onBeforeUpdate, onBlur, onChange, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onResize, onSelect, onSelectStart (vezi Conversaţia 6).

Figura 8.32

(sub)Obiectul Password este prezentat în detaliu în figura 8.33.

	Obiect	*Sintaxă*
	`Password`	`document.forms[].password`
	Reprezintă un câmp în cadrul unui formular care afişează asteriscuri atunci când utilizatorul introduce parola. Subobiectul `Password` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<input type="password">`. Subobiectul `Password` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	
Exemplu:	<form name="f1"> <input type="password" name="Pass" /> </form>	

Figura 8.33

Proprietăţile (sub)obiectului Password

Proprietăţile (sub)obiectului `Password` sunt prezentate în detaliu în figura 8.34.

	Proprietate	*Sintaxă*
	`defaultValue`	`document.forms[].password.defaultValue`
	Conţinutul implicit al zonei.	
	`disabled`	`document.forms[].password.disabled`
	Valoare logică (`true/false`) care indică starea zonei: dezactivată (`true`), activată (`false`).	
	`form`	`document.forms[].password.form`
	Referinţă la formularul care conţine zona.	
	`maxLength`	`document.forms[].password.maxLength`
	Numărul maxim de caractere care pot fi introduse în zonă.	
	`name`	`document.forms[].password.name`
	Numele zonei. Proprietatea corespunde atributului `name` al tag-ului `<input>`.	
	`readOnly`	`document.forms[].password.readOnly`
	Valoare logică (`true/false`) – dacă zona este numai citită (`true`) sau nu (`false`).	
	`size`	`document.forms[].password.size`
	Dimensiunea (în pixeli) zonei. Corespunde atributului `size` al tag-ului `<input>`.	
	`type`	`document.forms[].password.type`
	Tipul elementului în cadrul formularului. În acest caz, proprietatea returnează `text`. Această proprietate corespunde atributului `type` al tag-ului `<input>`.	
	`value`	`document.forms[].password.value`

Figura 8.34

Figura 8.34 *(continuare)*

Conţinutul zonei. Proprietatea corespunde atributului `value` al tag-ului `<input>`. `value.length` returnează numărul de caractere al zonei.

Metodele (sub)obiectului Password

Metodele (sub)obiectului `Password` sunt prezentate în detaliu în figura 8.35.

Metodă	*Sintaxă*
`blur()`	`document.form.password.blur()`
Zona pierde focus-ul.	
`click()`	`document.form.password.click()`
Simularea clic-ului mouse-ului pe zona de text. Această metodă nu este detectată prin `onClick`.	
`focus()`	`document.form.password.focus()`
Zona de text preia focus-ul.	
`select()`	`document.form.password.select()`
Selectează conţinutul zonei.	

Figura 8.35

(sub)Obiectul Radio

Fişa (sub)obiectului `Radio` este prezentată în figura 8.36.

Fişa (sub)obiectului Radio	
Obiectul părinte:	`Form`
Proprietăţi:	`align, checked, defaultChecked, defaultValue, disabled, form, name, size, status, type, value, width`
Metode:	`blur(), click(), focus()`
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUpdate, onBlur, onClick, onErrorUpdate, onFilterChange, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp` (vezi Conversaţia 6).

Figura 8.36

(sub)Obiectul `Radio` este prezentat în detaliu în figura 8.37.

	Obiect	*Sintaxă*
	`radio`	`document.forms[].radio`
	Reprezintă un buton radio într-un formular. Subobiectul `Radio` permite utilizatorului să selecteze o singură opţiune dintr-un grup de acţiuni. Subobiectul `Radio` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<input type="radio">`. Subobiectul `Radio` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	
Exemplu:	<form name="form1"> <input type="radio" name="optiune" /> </form>	

Figura 8.37

Proprietăţile (sub)obiectului Radio

Proprietăţile (sub)obiectului `Radio` sunt prezentat în detaliu în figura 8.38.

	Metodă	*Sintaxă*
	`align`	`document.forms[].radio.align`
	Alinierea butonului radio. Proprietatea poate conţine valorile: `left, center, right`.	
	`checked`	`document.forms[].radio.checked`
	Valoare logică (`true/false`) – butonul este activat (`true`) sau dezactivat (`false`).	
	`defaultChecked`	`document.forms[].radio.defaultChecked`
	Valoarea implicită a butonului radio (`true/false`).	
	`defaultValue`	`document.forms[].radio.defaultValue`
	Valoarea implicită a butonului radio.	
	`disabled`	`document.forms[].radio.disabled`
	Valoarea logică (`true/false`) care indică starea butonului: dezactivat (`true`), activat (`false`).	
	`form`	`document.forms[].radio.form`
	Referinţă la formularul care conţine butonul radio.	
	`name`	`document.forms[].radio.name`
	Numele butonului radio. Această proprietate corespunde atributului name al tag-ului `<input />`.	

Figura 8.38

`size`	`document.forms[].radio.size`
Dimensiunea (în pixeli) butonului radio. Această proprietate corespunde atributului `size` al tag-ului `<input />`.	
`status`	`document.forms[].radio.status`
Valoare logică (`true/false`) – butonul este activat (`true`) sau nu (`false`).	
`type`	`document.forms[].radio.type`
Tipul de element în cadrul formularului. În acest caz, proprietatea returnează `radio`. Această proprietate corespunde atributului `type` al tag-ului `<input />`.	
`value`	`document.forms[].radio.value`
Valoarea butonului radio. Corespunde atributului `value` al tag-ului `<input>`.	
`width`	`document.forms[].radio.width`
Mărimea (în pixeli) butonului radio.	

Figura 8.38 *(continuare)*

Metodele (sub)obiectului Radio

Metodele (sub)obiectului `Radio` sunt prezentate în detaliu în figura 8.39.

Metodă	***Sintaxă***
`blur()`	`document.form.radio.blur()`
Butonul radio pierde focus-ul.	
`click()`	`document.form.radio.click()`
Simularea clic-ului mouse-ului pe zona de text. Această metodă nu este detectată prin `onClick`.	
`focus()`	`document.form.radio.focus()`
Butonul radio preia focus-ul.	

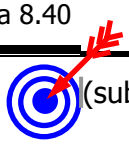
Figura 8.39

(sub)Obiectul Reset

Fişa obiectului `Reset` este prezentată în figura 8.40.

Fişa (sub)obiectului Reset	
Obiectul părinte:	`Form`
Proprietăţi:	`align, defaultValue, disabled, form, name, type, value`
Metode:	`blur(), click(), focus()`
Gestionarii de evenimente:	`onAfterDate, onBeforeUpdate, onBlur, onClick, onDblClick, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp` (vezi conversaţia 6)

Figura 8.40



(sub)Obiectul `Reset` este prezentat în detaliu în figura 8.41.

	Obiect	***Sintaxă***
	`reset`	`document.forms[].reset`
	Reprezintă un buton care şterge valorile din toate zonele formularului curent. Subobiectul `Reset` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<input type="reset">`. Ca şi în cazul butonului `submit`, pentru executarea acţiunii nu este necesar cod sursă JavaScript.	
Exemplu:	`<form name="form1">` `  <input type="reset" />` `</form>`	

Figura 8.41

Proprietăţile (sub)obiectului Reset

Proprietăţile (sub)obiectului `Reset` sunt prezentate în detaliu în figura 8.42.

	Proprietate	***Sintaxă***
	`align`	`document.forms[].reset.align`
	Alinierea butonului. Proprietatea poate conţine valoarea: `left, center` sau `right.`	
	`defaultValue`	`document.forms[].reset.defaultValue`
	Valoarea implicită a butonului.	
	`disabled`	`document.forms[].reset.disabled`
	Valoare logică (`true/false`) – butonul este dezactivat (`true`) sau activat (`false`).	
	`form`	`document.forms[].reset.form`

Figura 8.42

	Referinţă la formularul care conţine butonul.
`name`	`document.forms[].reset.name`
	Numele butonului. Această proprietate corespunde atributului `name` al tag-ului `<input />`.
`type`	`document.forms[].reset.type`
	Tipul de element în cadrul formularului. În acest caz, proprietatea returnează `reset`. Această proprietate corespunde atributului `type` al tag-ului `<input />`.
`value`	`document.forms[].reset.value`
	Valoarea butonului. Această proprietate corespunde atributului `value` al tag-ului `<input />`.

Figura 8.42 *(continuare)*

Metodele (sub)obiectului Reset

Metodele (sub)obiectului `Reset` sunt prezentate în detaliu în figura 8.43.

Metodă	***Sintaxă***
`blur()`	`document.form.reset.blur()`
Butonul pierde focus-ul.	
`click()`	`document.form.reset.click()`
Simularea clic-ului mouse-ului pe zona de text. Această metodă nu este detectată prin `onClick`.	
`focus()`	`document.form.reset.focus()`
Butonul preia focus-ul.	

Figura 8.43

(sub)Obiectul Submit

Fişa (sub)obiectului `Submit` este prezentată în figura 8.44.

Fişa (sub)obiectului Submit	
Obiectul părinte:	`Form`
Proprietăţi:	`align, disabled, form, name, type, value`
Metode:	`blur(), click(), focus()`
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUpdate, onBlur, onClick, onDblClick, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp` (vezi conversaţia 6).

Figura 8.44

(sub)Obiectul `Submit` este prezentat în detaliu în figura 8.45.

	Obiect	***Sintaxă***
	`Submit`	`document.forms[].submit`
	Reprezintă un buton care înaintează server-ului datele conţinute în formular. Subobiectul `submit` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<input type="submit">`. Subobiectul `submit` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său. Pentru executarea acţiunii nu este necesar cod sursă JavaScript.	
Exemplu:	<form name="form1"> <input type="submit" name="init" /> </form>	

Figura 8.45

Proprietăţile (sub)obiectului Submit

Proprietăţile (sub)obiectului `Submit` sunt prezentate în detaliu în figura 8.46.

	Proprietate	***Sintaxă***
	`align`	`document.forms[].submit.align`
	Alinierea butonului. Proprietatea poate conţine valoarea: `left, center` sau `right`.	
	`disabled`	`document.forms[].submit.disabled`
	Valoare logică (`true/false`) – butonul este dezactivat (`true`) sau activat (`false`).	
	`form`	`document.forms[].submit.form`
	Referinţă la formularul care conţine butonul.	
	`name`	`document.forms[].submit.name`

Figura 8.46

	Numele butonului. Această proprietate corespunde atributului `name` al tag-ului `<input />`.
`type`	`document.forms[].submit.type`
	Tipul de element în cadrul formularului. În acest caz, proprietatea returnează `submit`. Această proprietate corespunde atributului `type` al tag-ului `<input />`.
`value`	`document.forms[].submit.value`
	Valoarea butonului. Această proprietate corespunde atributului `value` al tag-ului `<input />`.

Figura 8.46 *(continuare)*

Metodele (sub)obiectului Submit

Metodele (sub)obiectului `Submit` sunt prezentate în detaliu în figura 8.47.

Metodă	***Sintaxă***
`blur()`	`document.form.submit.blur()`
Butonul pierde focus-ul.	
`click()`	`document.form.submit.click()`
Simularea clic-ului mouse-ului pe zona de text. Această metodă nu este detectată prin `onClick`.	
`focus()`	`document.form.submit.focus()`
Butonul preia focus-ul.	

Figura 8.47

(sub)Obiectul Select

Fişa (sub)obiectului `Select` este prezentată în figura 8.48.

Fişa (sub)obiectului Select	
Obiectul părinte:	`Form`
Subobiecte:	`Option`
Proprietăţi:	`disabled, form, length, multiple, name, options[], selectedIndex, size, type, value`
Metode:	`add(), blur(), focus(), option.add(), option.remove(), remove()`
Gestionarii de evenimente:	`onAfterDate, onBeforeUpdate, onBlur, onChange, onDregStart, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onResize, onRowExit, onSelectStart` (vezi conversaţia 6)

Figura 8.48

(sub)Obiectul `Select` este prezentat în detaliu în figura 8.49.

	Obiect	*Sintaxă*
	`Select`	`document.forms[].select`
	Reprezintă o listă derulantă care afişează o serie de opţiuni într-un formular. Subobiectul `Select` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<select> ... </select>`. Subobiectul `Select` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	
Exemplu:	<form name="form1"> <select name="optiuni"> <option value="opțiune1">Prima opțiune</option> <option value="opțiune2">A doua opțiune</option> <option value="opțiune3">A treia opțiune</option> </select> </form>	

Figura 8.49

Proprietăţile (sub)obiectului Select

Proprietăţile (sub)obiectului `Select` sunt prezentate în detaliu în figura 8.50.

	Proprietate	*Sintaxă*
	`disabled`	`document.forms[].select.disabled`
	Valoare logică (`true/false`) – zona este dezactivată (`true`) sau activată (`false`).	
	`form`	`document.forms[].select.form`
	Referinţă la formularul care conţine zona.	
	`length`	`document.forms[].select.length`
	Numărul de opţiuni din listă.	
	`multiple`	`document.forms[].select.multiple`
	Valoare logică (`true/false`) – selecţia multiplă este autorizată (`true`) sau nu (`false`).	
	`name`	`document.forms[].select.name`
	Numele zonei. Această proprietate corespunde atributului `name` al tag-ului `<select/>`.	
	`options[]`	`document.forms[].select.options[]`
	Setul de opţiuni propuse în listă.	
	`selectedIndex`	`document.forms[].select.selectedIndex`
	Valoare care indică rangul opţiunii selectate în listă. Această proprietate conţine valoarea –1 dacă nici o opţiune nu este selectată.	
	`Size`	`document.forms[].select.size`

Figura 8.50

	Valoare care indică numărul de opţiuni vizibile simultan în listă.
`type`	`document.forms[].select.type`
	Această proprietate conţine fie *select-one* fie *select-multiple*.
`value`	`document.forms[].select.value`
	Valoarea opţiunii selectate.

Figura 8.50 *(continuare)*

Metodele (sub)obiectului Select

Metodele (sub)obiectului `Select` sunt prezentate în detaliu în figura 8.51.

Metodă	***Sintaxă***
`add()`	`document.form.select.add(`*`Element, Amonte`*`)`
Adaugă în listă opţiunea referită prin argumentul *`Element`* şi o inserează înaintea opţiunii referite prin argumentul *`Amonte`*. Dacă argumentul *`Amonte`* este omis, elementul este adăugat la finele listei.	
`blur()`	`document.form.select.blur()`
Zona pierde focus-ul.	
`focus()`	`document.form.select.focus()`
Zona preia focus-ul.	
`option.add()`	`document.form.select.option.add(`*`Element,Amonte`*`)`
Adaugă în listă opţiunea referită prin argumentul *`Element`* şi o inserează înaintea opţiunii referite prin argumentul *`Amonte`*. Dacă argumentul *`Amonte`* este omis, elementul este adăugat la finele listei.	
`option.remove()`	`document.form.select.option.remove(`*`Rang`*`)`
Elimină din listă opţiunea al cărei rang este indicat în argument.	
`remove()`	`document.form.select.remove(`*`Rang`*`)`
Elimină din listă opţiunea al cărei rang este indicat în argument.	

Figura 8.51

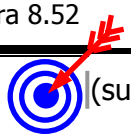
(sub)Obiectul Text

Fişa (sub)obiectului `Text` este prezentată în figura 8.52.

Fişa (sub)obiectului Text

Obiectul părinte:	`Form`
Proprietăţi:	`defaultValue, disabled, form, maxLength, name, readOnly, size, type, value`
Metode:	`blur(), click(), focus(), select()`
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUpdate, onBlur, onClick, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onSelect` (vezi conversaţia 6).

Figura 8.52



(sub)Obiectul `Text` este prezentat în detaliu în figura 8.53.

	Obiect	***Sintaxă***
	`Text`	`document.forms[].text`
	Reprezintă o zonă de text formată dintr-o singură linie. Subobiectul `Text` este versiunea „transpusă în obiect” a tag-ului (X)HTML `<input type="text">`. Subobiectul `Text` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	
Exemplu:	<form name="form1"> <input type="text" name="text1" /> </form>	

Figura 8.53

Proprietăţile (sub)obiectului Text

Proprietăţile (sub)obiectului `Text` sunt prezentate în detaliu în figura 8.54.

	Proprietate	***Sintaxă***
	`defaultValue`	`document.form[].text.defaultValue`
	Conţinutul implicit al zonei.	
	`disabled`	`document.form[].text.disabled`
	Valoare logică (`true/false`) – zona este dezactivată (`true`) sau activată (`false`).	
	`form`	`document.form[].text.form`
	Referinţă la formularul care conţine zona.	
	`maxLength`	`document.form[].text.maxLength`

Figura 8.54

	Numărul maxim de caractere care pot fi introduse în zonă.
`name`	`document.form[].text.name`
	Numele zonei. Această proprietate corespunde atributului `name` al tag-ului `<input />`.
`readOnly`	`document.form[].text.readOnly`
	Valoare logică (`true/false`) – zona este numai citită (`true`) sau nu (`false`).
`size`	`document.form[].text.size`
	Mărimea (în pixeli) zonei. Această proprietate corespunde atributului `size` al tag-ului `<input />`.
`type`	`document.form[].text.type`
	Tipul elementului în cadrul formularului. În acest caz, proprietatea returnează `text`. Această proprietate corespunde atributului `type` al tag-ului `<input />`.
`value`	`document.form[].text.value`
	Conţinutul zonei. Această proprietate corespunde atributului `value` al tag-ului `<input />`. `value.length` returnează numărul de caractere ale zonei.

Figura 8.54 *(continuare)*

Metodele (sub)obiectului Text

Metodele (sub)obiectului `Text` sunt prezentate în detaliu în figura 8.55.

Metodă	***Sintaxă***
`blur()`	`document.form.text.blur()`
Zona pierde focus-ul.	
`click()`	`document.form.text.click()`
Simularea clic-ului mouse-ului pe zona de text. Această metodă nu este detectată prin `onClick`.	
`focus()`	`document.form.text.focus()`
Zona de text preia focus-ul.	
`select()`	`document.form.text.select()`
Selectează conţinutul zonei.	

Figura 8.55

(sub)Obiectul Textarea

Fişa (sub)obiectului `Textarea` este prezentată în figura 8.56.

Fişa (sub)obiectului Textarea	
Obiectul părinte:	`Form`
Proprietăţi:	`cols, defaultValue, disabled, form, maxLength, name, readOnly, rows, size, type, value, wrap`
Metode:	`blur(), click(), focus(), select()`
Gestionarii de evenimente:	`onAfterUpdate, onBeforeUnload, onBeforeUpdate, onBlur, onChange, onDragStart, onErrorUpdate, onFilterChange, onFocus, onHelp, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onRowEnter, onRowExit, onSelect, onSelectStart` (vezi conversaţia 6).

Figura 8.56

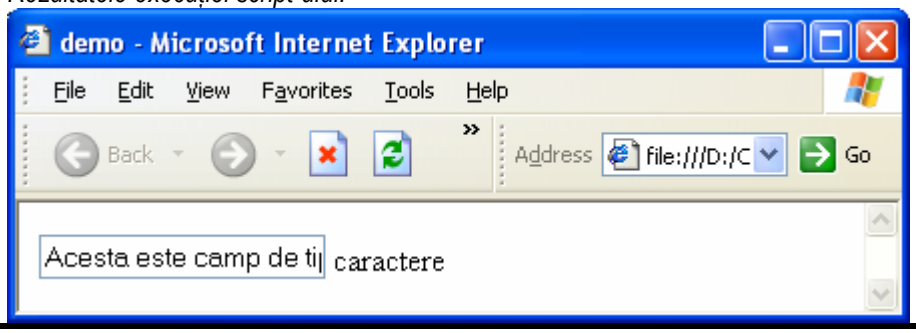
(sub)obiectul `Textarea` este prezentat în detaliu în figura 8.57.

	Obiect	***Sintaxă***
	`Textarea`	`document.form[].textarea`
	Reprezintă o zonă de text multilinie. Subobiectul `Textarea` este versiunea „transpusă în obiect” a tag-ului (X)HTML `<input type="textarea">`. Subobiectul `Textarea` poate fi accesat prin proprietatea `form.elements[]` sau prin numele său.	
Exemplu:	<html> <head> <title>demo</title> <script> function control(f){ lungmax=30; document.form1.numar.value=f.value.length; if(f.value.length>lungmax){ f.value=f.value.substring(0,lungmax); document.form1.numar.value=lungmax; } } </script> </head> <body> <form name="form1">	

Figura 8.57

```
        <input type="text" name="numar" size="1"> caractere<br>
        <textarea name="text" cols="40" row="5" onKeyUp="control(this)">
        </textarea>
      </form>
    </body>
</html>
```

Rezultatele execuției script-ului.



Proprietăţile (sub)obiectului Textarea

Proprietăţile (sub)obiectului `Textarea` sunt prezentate în detaliu în figura 8.58.

	Proprietate	***Sintaxă***
	`cols`	`document.forms[].textarea.cols`
	Numărul de coloane ale zonei.	
	`defaultValue`	`document.forms[].textarea.defaultValue`
	Conţinutul implicit al zonei.	
	`disabled`	`document.forms[].textarea.disabled`
	Valoare logică (`true/false`) – zona este dezactivată (`true`) sau activată (`false`).	
	`form`	`document.forms[].textarea.form`
	Referinţă la formularul care conţine zona.	
	`maxLength`	`document.forms[].textarea.maxLength`
	Numărul maxim de caractere care pot fi introduse în zonă.	
	`name`	`document.forms[].textarea.name`
	Numele zonei. Această proprietate corespunde atributului `name` al tag-ului `<input />`.	
	`readOnly`	`document.forms[].textarea.readOnly`
	Valoare logică (`true/false`) – zona este numai citită (`true`) sau nu (`false`).	
	`rows`	`document.forms[].textarea.rows`
	Numărul de linii ale zonei.	

Figura 8.58

`size`	`document.forms[].textarea.size`
Dimensiunea (în pixeli) zonei. Această proprietate corespunde atributului `size` al tag-ului `<input />`.	
`type`	`document.forms[].textarea.type`
Tipul elementului în formular. În acest caz, proprietatea returnează `text`. Această proprietate corespunde atributului `type` al tag-ului `<input />`.	
`value`	`document.forms[].textarea.value`
Conţinutul zonei. Această proprietate corespunde atributului `value` al tag-ului `<input />`. `value.length` returnează numărul de caractere al zonei.	
`wrap`	`document.forms[].textarea.wrap`
Indică modul de gestionare al sfârşitului de linie. Această proprietate poate conţine trei valori: `hard` (un <CR> este inserat la sfârşitul fiecărei linii), `soft` (textul trece pe linia următoare fără <CR>), `none` (fără trecere la linia următoare în mod automat).	

Figura 8.58 *(continuare)*

Metodele (sub)obiectului Textarea

Metodele (sub)obiectului `Textarea` sunt prezentate în detaliu în figura 8.59.

Metodă	***Sintaxă***
`blur()`	`document.form.textarea.blur()`
Zona pierde focus-ul.	
`click()`	`document.form.textarea.click()`
Simularea clic-ului mouse-ului pe zona de text. Această metodă nu este detectată prin `onClick()`.	
`focus()`	`document.form.textarea.focus()`
Zona de text preia focus-ul.	
`select()`	`document.form.textarea.select()`
Selectează conţinutul zonei.	

Figura 8.59

Aplicaţii

❑ Creaţi un formular care afişează o listă de cărţi (pe care le-am folosit în cadrul acestei lucrări) JavaScript cu preţul de vânzare, precedate de o casetă de validare. În momentul selectării/deselectării cărţilor, totalul este recalculat şi afişat într-o zonă de text (figura 8.60).

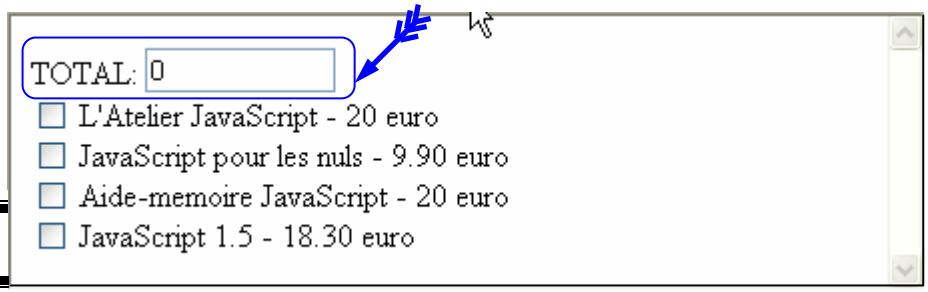


Figura 8.60

În figura 8.61 este prezentat documentul XHTML în care s-a inserat script-ul de calcul. Rezultatele execuţiei sunt prezentate în figura 8.62.

c-08-005.htm - Notepad

File Edit Format View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Aplicatie</title>
<script>
 function calcule(){
  document.calcul.elements[0].value="0";
  for(i=1;i<document.calcul.elements.length;i++){
   if(document.calcul.elements[i].checked)
          document.calcul.elements[0].value=
             parseFloat(document.calcul.elements[0].value)+
             parseFloat(document.calcul.elements[i].value);
   }
  }
</script>
</head>
<body>
<form name="calcul" id="calcul">
  <p>TOTAL:
    <input name="textfield" type="text" size="10" />
    <br />
    <input type="checkbox" value="20" onClick="calcule()" />
    L'Atelier JavaScript - 20 euro<br />
    <input type="checkbox" value="9.90" onClick="calcule()" />
    JavaScript pour les nuls - 9.90 euro<br />
    <input type="checkbox" value="20" onClick="calcule()" />
    Aide-memoire JavaScript - 20 euro<br />
    <input type="checkbox" value="18.30" onClick="calcule()" />
    JavaScript 1.5 - 18.30 euro<br />
  </p>
</form>
</body>
</html>
```

Figura 8.61

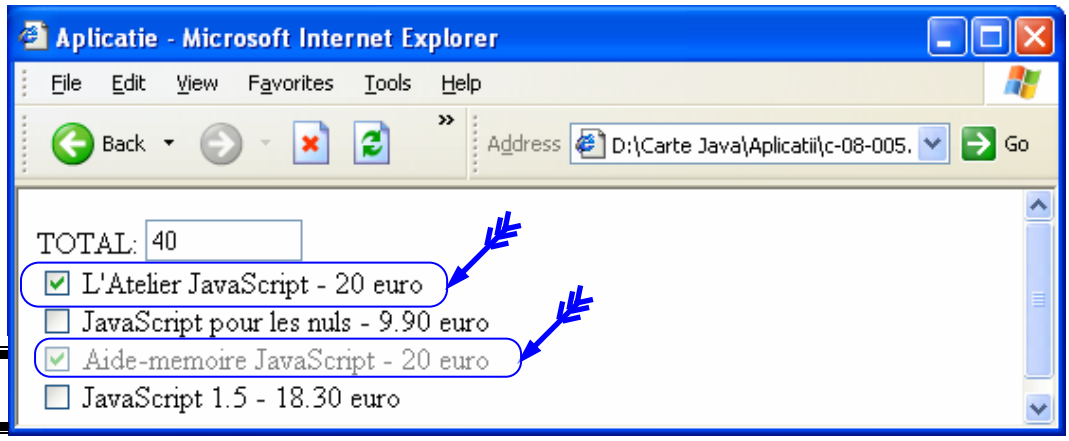


Figura 8.62

Comentarii:

- ✓ Formularul (`calcul`) este clasic; el conţine o zonă de text (pentru `TOTAL`) şi câte o casetă de validare pentru fiecare articol. Bifaţi un articol! Preţul său este adunat la totalul deja prezent în câmpul `TOTAL`. Dezactivaţi un articol! Preţul său este scăzut din total.
- ✓ Zona care afişează totalul general este un câmp text a cărui dimensiune a fost limitată la 10.
- ✓ Urmează definiţiile casetelor de validare (checkbox). Valoarea (value) fiecărei casete de validare reprezintă preţul articolului: 20, 9.90, 20, 18.30.
- ✓ Pentru fiecare casetă de validare în tag-ul `<input>` s-a definit gestionarul de evenimente `onClick`, care execută funcţia JavaScript `calcule()` al cărui cod este definit în secţiunea `<head>` a paginii Web (figura 8.61).
- ✓ Fiecare element (subobiect) al formularului este identificat prin numărul de ordine: `elements[0]` pentru primul subobiect (total), `elements[1]` pentru al doilea subobiect etc. Sintaxa pentru identificarea unui subobiect (element) precis al formularului este puţin lungă, dar logică. De exemplu, elementul (subobiectul) 0 este situat în formularul cu numele `calcul`, el însuşi situat în documentul curent. Valoarea 0 este atribuită proprietăţii `value`. În consecinţă, atunci când activez sau dezactivez un articol, totalul existent este anulat.
- ✓ Prin intermediul unei bucle `for`, poate fi recuperată valoarea fiecărui subarticol (element) al formularului. Bucla `for` execută instrucţiunile cuprinse între cele două acolade ({ şi }), figura 8.63.

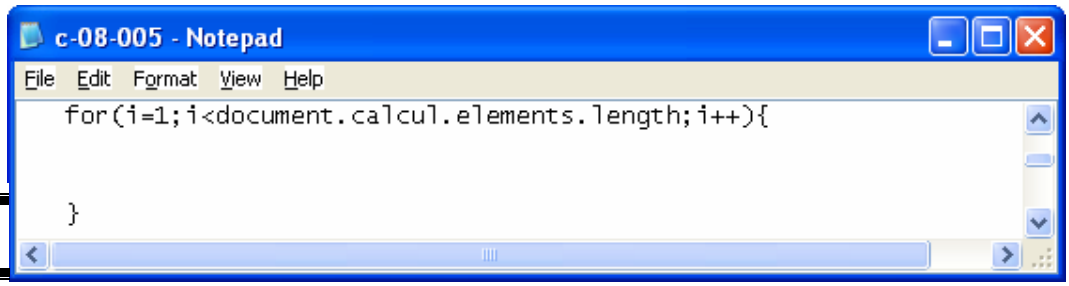


Figura 8.63

Observaţi cei trei parametrii ai buclei, plasaţi între paranteze:

1. *Variabila de control a buclei i* este iniţializată cu 1. Primul element are rangul 0, dar, în acest caz el nu trebuie prelucrat de către bucla `for` întrucât este vorba de câmpul `Total`.
2. *Testul de ieşire din buclă:* `i<document.calcul.elements.length`. Dacă în urma evaluării condiţiei rezultatul este FALSE, atunci instrucţiunile din corpul buclei nu mai sunt efectuate. Aici testul compară valoarea variabilei i cu numărul de elemente ale formularului. Putem simplifica testul scriind `i<4`. Dar, dacă se adaugă ori se suprimă o casetă de validare, trebuie modificată de asemenea şi condiţia de test. Este mai uşor să utilizăm proprietatea **`length`** a setului de elemente; ea returnează în mod automat numărul de elemente ale formularului.
3. *Pasul.* El indică modul în care evoluează variabila de control a buclei la fiecare trecere. Aici se adaugă 1 la valoarea variabilei `i (i++)`.

✓ Proprietatea `checked` (casetă bifată) a fiecărei casete de validare a formularului este testată în continuare. Această proprietate returnează valoarea TRUE sau FALSE. Dacă testul este TRUE, atunci TOTAL (`elements[0]`) este recalculat; valoarea sa este adăugată la valoarea elementului ce urmează a fi testat (figura 8.64).

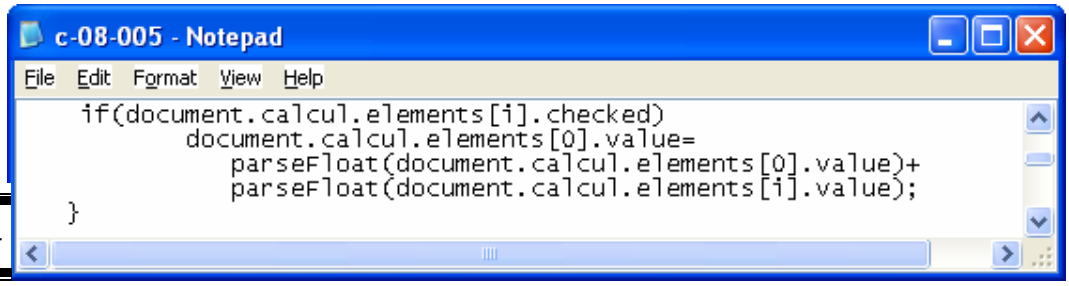


Figura 8.64

✓ Funcţia `parseFloat` este obligatorie întrucât valorile checkbox-urilor sunt date de tip text. `ParseFloat` converteşte un şir de caractere într-o valoare numerică reală. `ParseFloat` este indispensabilă dacă valoarea unui câmp urmează să fie utilizată într-o expresie aritmetică.

❑ Simulaţi funcţionarea buclei `for` din aplicaţia precedentă pentru cazul în care articolul 1 a fost bifat (activat), figura 8.65.

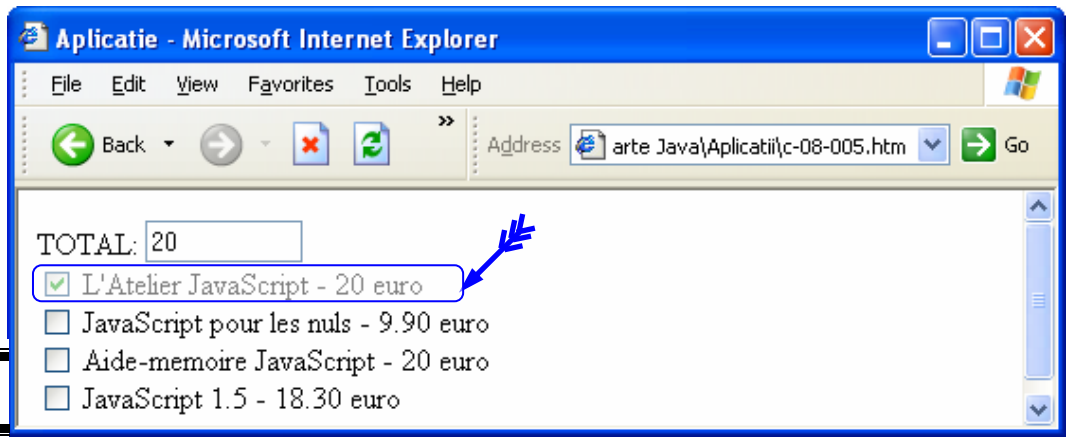


Figura 8.65

1. *Variabila i ia valoarea 1.*
2. *Proprietatea* `checked` *a elementului (subobiectului) 1 al formularului este testată.*

   `if(document.calcul.elements[1].checked);` *ea returnează valoarea* `TRUE`*.*
3. *Valoarea elementului 1 este adăugată la conţinutul câmpului TOTAL.*
4. *Sfârşitul buclei simbolizat prin acolada }.*
5. *Cel de-al treilea parametru al buclei este executat: i++. Variabila i ia valoarea 2.*
6. *Testul (al doilea argument al buclei) buclei este efectuat: i<4. El returnează valoarea FALSE. Bucla este executată încă o dată.*

❑ Personalizaţi script-ul din aplicaţia precedentă. Puteţi adăuga oricâte articole doriţi, având grijă de a preciza valoarea fiecăruia dintre ele. Expresia:

`document.calcul.elemnts.length` din iniţializarea buclei va furniza tot timpul numărul de elemente fără a mai fi nevoie ca dumneavoastră să mai interveniţi. Dacă lucraţi numai cu valori întregi, utilizaţi funcţia `parseInt`. Înlocuiţi în acest caz funcţia `parseFloat` cu funcţia `parseInt` (figura 8.66).

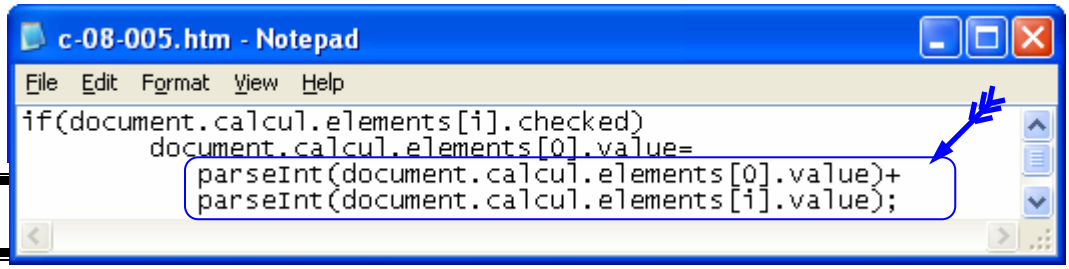


Figura 8.66

❑ Scrieţi un program JavaScript care calculează suma primelor n numere naturale.

Indicaţie. Introduceţi valoarea lui n într-o zonă simplă de text, subobiect al obiectului `Form`.

În figura 8.67 este prezentat codul sursă (X)HTML complet (vezi script-ul aplicaţiei).

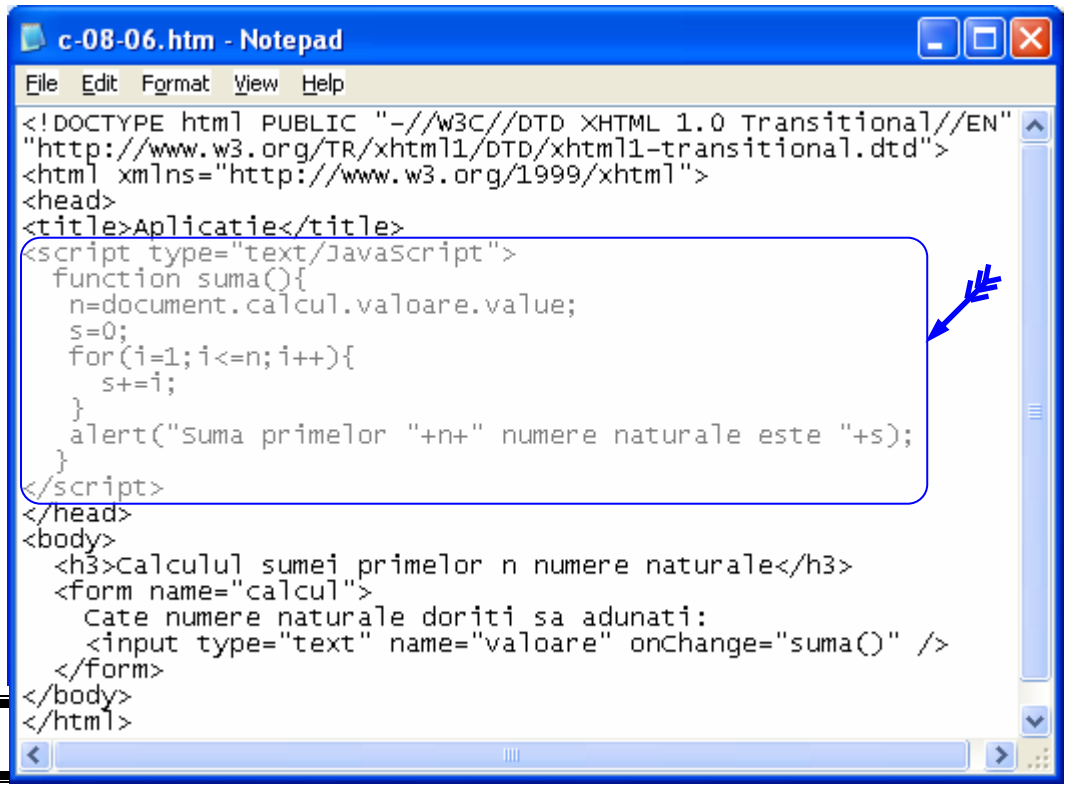


Figura 8.67

În figura 8.68 este prezentat rezultatul execuţiei programului JavaScript pentru 13 numere naturale.

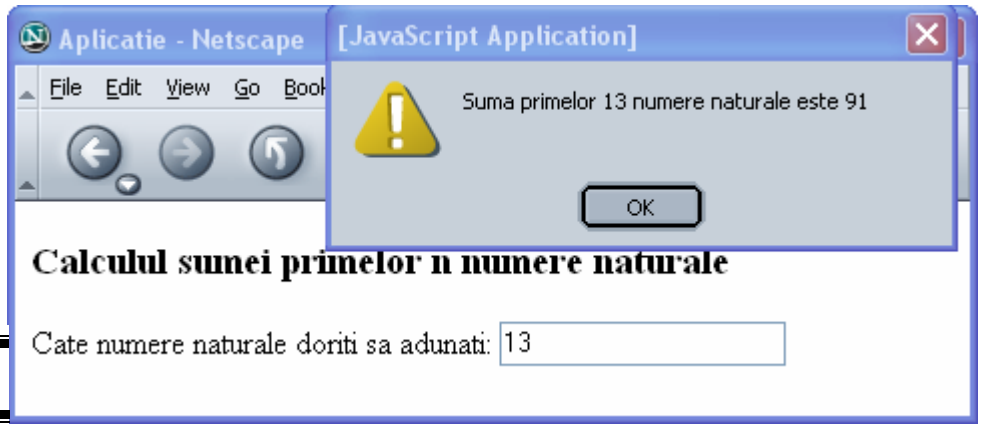


Figura 8.68

❑ Scrieţi un program JavaScript pentru expedierea (conţinutului) formularului din figura 8.69, prin poştă electronică la adresa (fictivă) webmaster@abc.ro. Personalizaţi script-ul.

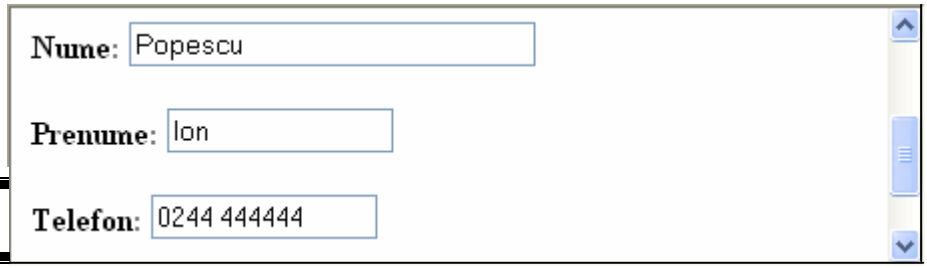


Figura 8.69

În figura 8.70 se prezintă documentul XHTML.

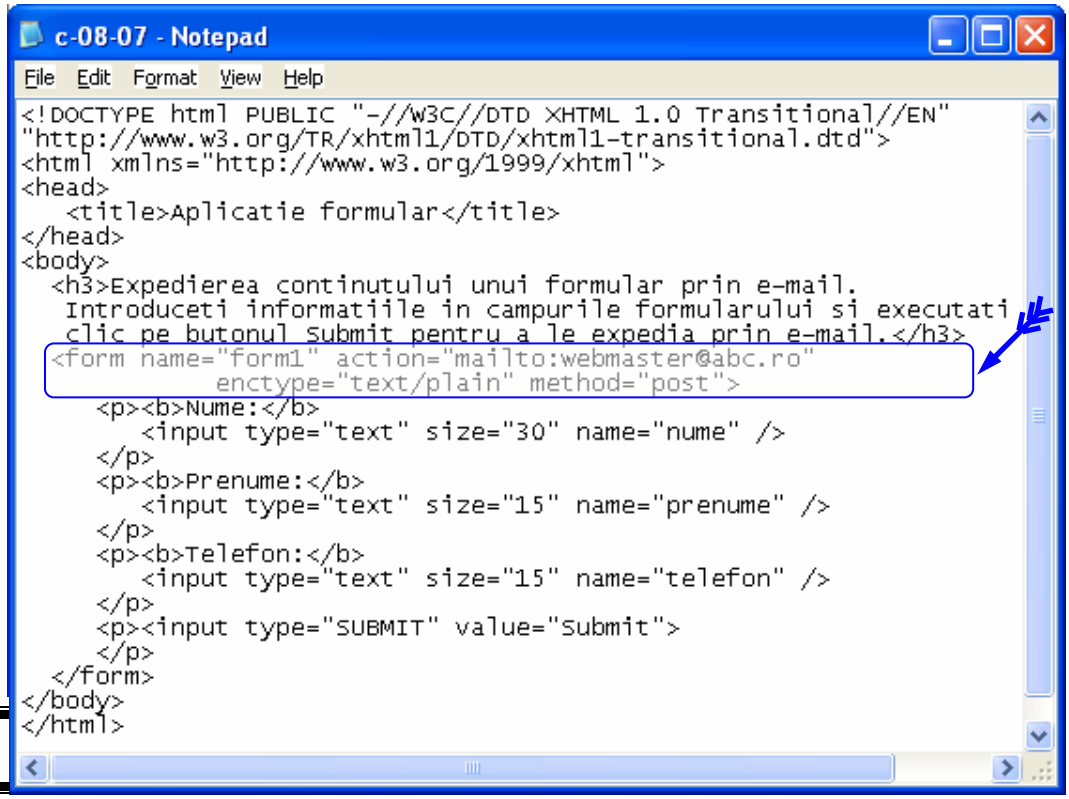


Figura 8.70

Remarci:

- ✓ Pentru a personaliza acest formular, înlocuiţi `webmaster@abc.ro` cu propria dumneavoastră adresă de e-mail.
- ✓ Atributul `enctype="text/plain"` al tag-ului `form` asigură utilizatorul că informaţiile din cadrul mesajului vor fi expediate într-o formă lizibilă. Această tehnică prezintă avantajul că permite citirea rapidă a datelor care aparţin unui formular.
- ✓ Există numeroase script-uri şi servicii `CGI` gratuite (vezi http://www.jsworkshop.com/).

Validați un formular cu JavaScript

Validarea formularelor reprezintă, fără îndoială aplicaţia cea mai preţioasă a limbajului JavaScript. Validarea unui formular constă în verificarea, cu ajutorul unui script a corectitudinii informaţiilor introduse de către utilizator.

Puteţi utiliza limbajul JavaScript pentru:

- ✓ validarea datelor introduse de utilizator. Puteţi valida câmpuri de intrare, grupuri de câmpuri sau întregul formular, utilizând gestionari de evenimente şi funcţii JavaScript.
- ✓ construirea formularelor interactive, în care o parte sau întreaga prelucrare are loc pe parte de client.
- ✓ a testa conformitatea datelor introduse de utilizator cu politicile de procedură impuse (*exemplu de politică de procedură:* data de livrare a unei comenzi nu poate fi în ziua de sâmbătă/duminică a săptămânii).
- ✓ a testa prezenţa datelor în câmpurile obligatorii ale unui formular (un câmp este prezent dacă nu este vid).

Aplicaţii

❑ Calculaţi rădăcină pătrată dintr-un număr ($\sqrt{x}$). Validaţi datele introduse de utilizator ($x \geq 0$). În caz de eroare (x<0) se va afişa mesajul: „*Rădăcină imaginară*".

În figura 8.71 este prezentat codul (X)HTML complet, în care s-a inserat script-ul de calcul.

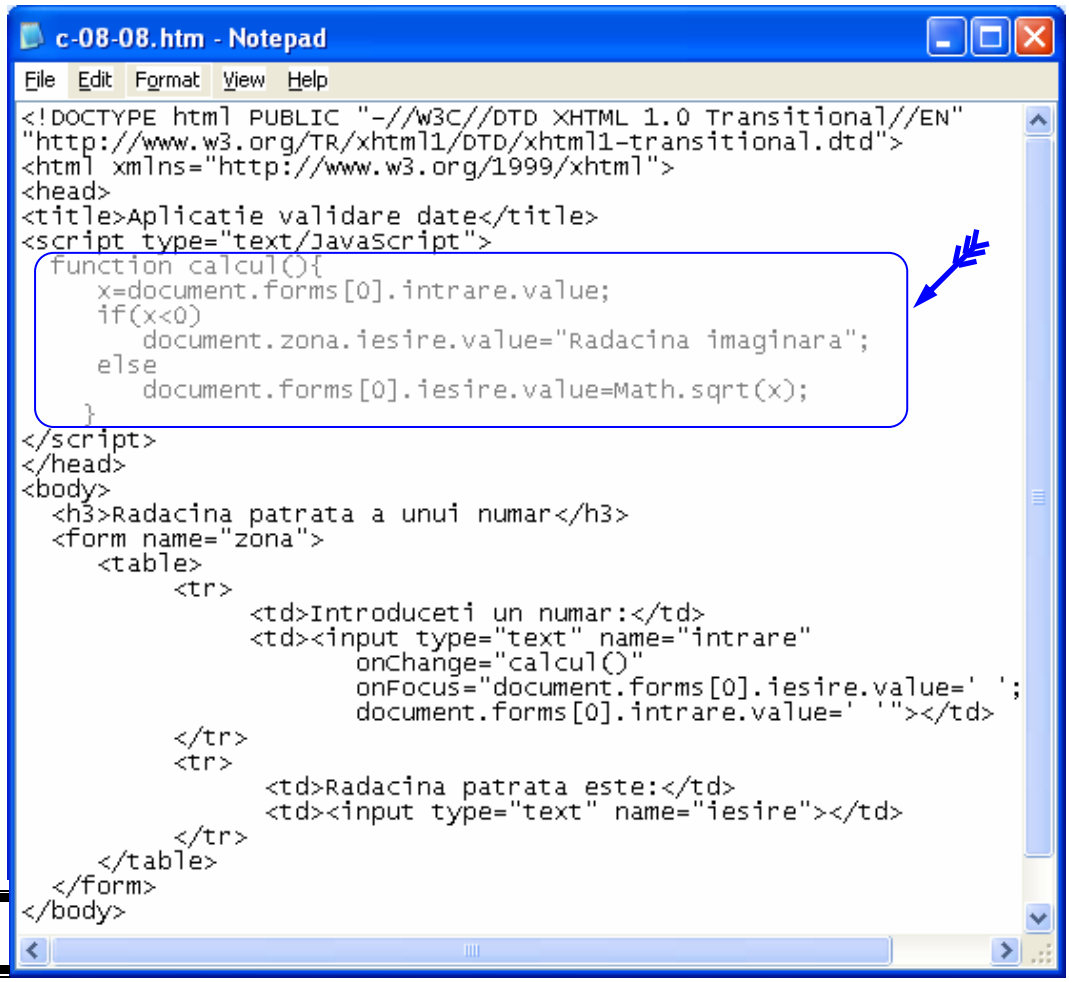


Figura 8.71

În figurile 8.72 şi 8.73 se prezintă rezultatele execuţiei programului pentru x=-16 şi pentru x=16.

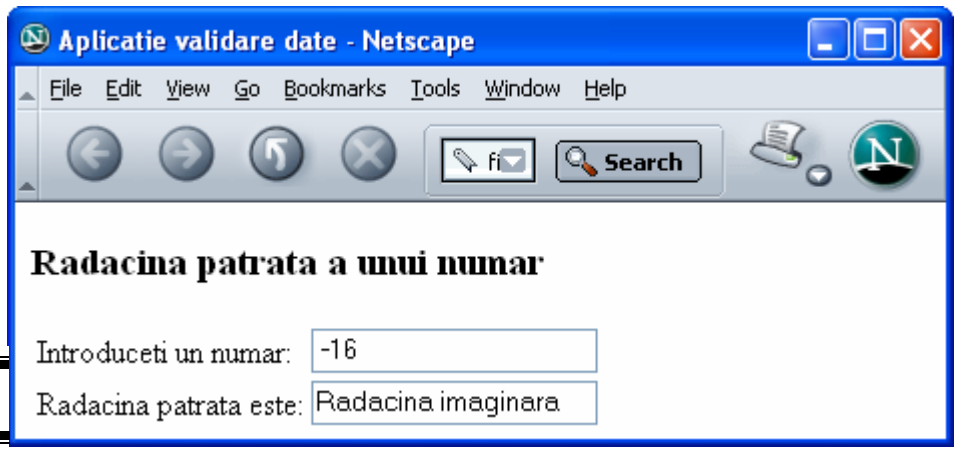


Figura 8.72

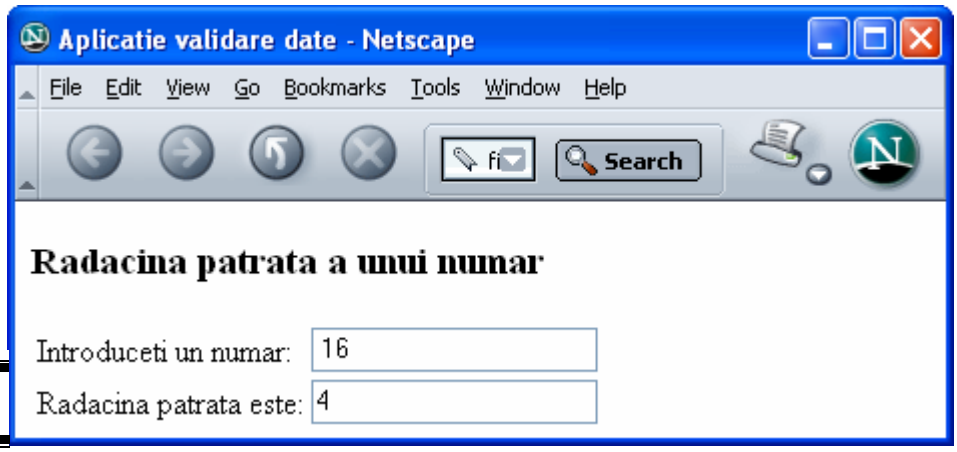


Figura 8.73

Comentaţi codul XHTML al aplicaţiei.

❑ Scrieţi un program JavaScript care verifică dacă conţinutul celor trei câmpuri ale unui formular: nume, prenume, codcard nu este vid. În caz de eroare afişaţi unul din mesajele de mai jos:

- ✓ *„Aţi uitat să introduceţi numele dumneavoastră!"*
- ✓ *„Aţi uitat să introduceţi prenumele dumneavoastră!"*
- ✓ *„Aţi uitat să introduceţi numărul dumneavoastră de card!"*

În figura 8.74 se prezintă codul complet al documentului HTML.

c-08-09 - Notepad

File Edit Format View Help

```
<html>
<head>
<title>Validare formular</title>
<script type="text/JavaScript">
  function verif(){
   if(document.comanda.nume.value=="")
     alert("Ati uitat sa introduceti numele dumneavoastra!");
   else
     if(document.comanda.prenume.value=="")
        alert("Ati uitat sa introduceti prenumele dumneavoastra!");
      else
        if(document.comanda.card.value=="")
           alert("Ati uitat sa introduceti numarul dumneavoastra
                        de card!");
        else  return true;
     return false;
    }
</script>
</head>
<body>
  <h3>Validare formular</h3>
  <form name="comanda" action=" " method="post"
             onSubmit="return verif()">
     <table border="1">
       <tr><td>Nume:</td>
           <td><input type="text" name="nume"></td>
           <td>Prenume:</td>
           <td><input type="text" name="prenume"></td></tr>
         <tr><td>Numar card:</td>
           <td><input type="text" name="card"></td>
           <td><input type="submit" value="OK"
                       onSubmit="return verif()"></td>
            <td><input type="reset" value="Resetare"></td></tr>
     </table>
  </form>
</body>
</html>
```

Figura 8.74

În figura 8.75, figura 8.76 se prezintă rezultatele execuţiei programului JavaScript.

Figura 8.75

Figura 8.76

Remarcă. Comentaţi linia de cod onsubmit="return verif()" (vezi tag-ul `form`).

❑ Scrieţi un script care validează adresa de e-mail introdusă într-o zonă de tip text din cadrul unui formular (figura 8.77).

Dacă zona de e-mail este vidă se va genera mesajul de eroare: *„Vă rog, introduceţi adresa de e-mail!”.* Dacă din adresa de e-mail lipsesc @ sau punctul (.), atunci adresa de e-mail va fi considerată incorectă. În acest caz, se va genera mesajul de eroare: *„Adresa de e-mail este incorectă! Vă rog, s-o corectaţi!*”. Formularul poartă numele de „aplicaţie”. Acest nume va fi utilizat în cadrul script-ului. De notat că formularul va fi expediat la adresa e-mail (imaginară) `webmaster@abc.ro`.

Figura 8.77

c-08-10 - Notepad

File Edit Format View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Aplicatie validare e-mail</title>
<script type="text/javascript">
function validare(){
 if(document.aplicatie.email.value==""){
    alert("Va rog introduceti adresa de e-mail!");
    aplicatie.email.focus();
    return false;
   }
 if((document.aplicatie.email.value.indexOf("@",0)<0)&&
     (document.aplicatie.email.value.indexOf(".",0)<0)){
   alert("Adresa e-mail este incorecta! \n Va rog s-o corectati!");
   document.aplicatie.email.focus();
   return false;
  }
   alert("Formularul a fost expediat");
   document.aplicatie.submit();
}
</script>
</head>
<body>
  <form name="aplicatie" action="mailto:webmaster@abc.ro"
        method="post">
      <p>
          Adresa e-mail: <br />
          <input type="text" name="email" size="20">
      </p>
      <p>
          <input type="button" value="Submit" onClick="validare()">
          <input type="reset" value="Reset">
      </p>
  </form>
</body>
</html>
```

Figura 8.78

În figura 8.79, figura 8.80 se prezintă câteva exemple de test ale script-ului pe care l-aţi realizat.

Figura 8.79

Figura 8.80

Remarci

- ✓ Dacă doriţi, puteţi personaliza script-ul.
- ✓ Pentru informaţii suplimentare privind o validare mai complexă a unei adrese e-mail consultaţi site-ul: http://developer.netscape.com/library/exemples/javascript/formval/overview.html.

❑ Simulaţi funcţionarea următorului program JavaScript (figura 8.81).

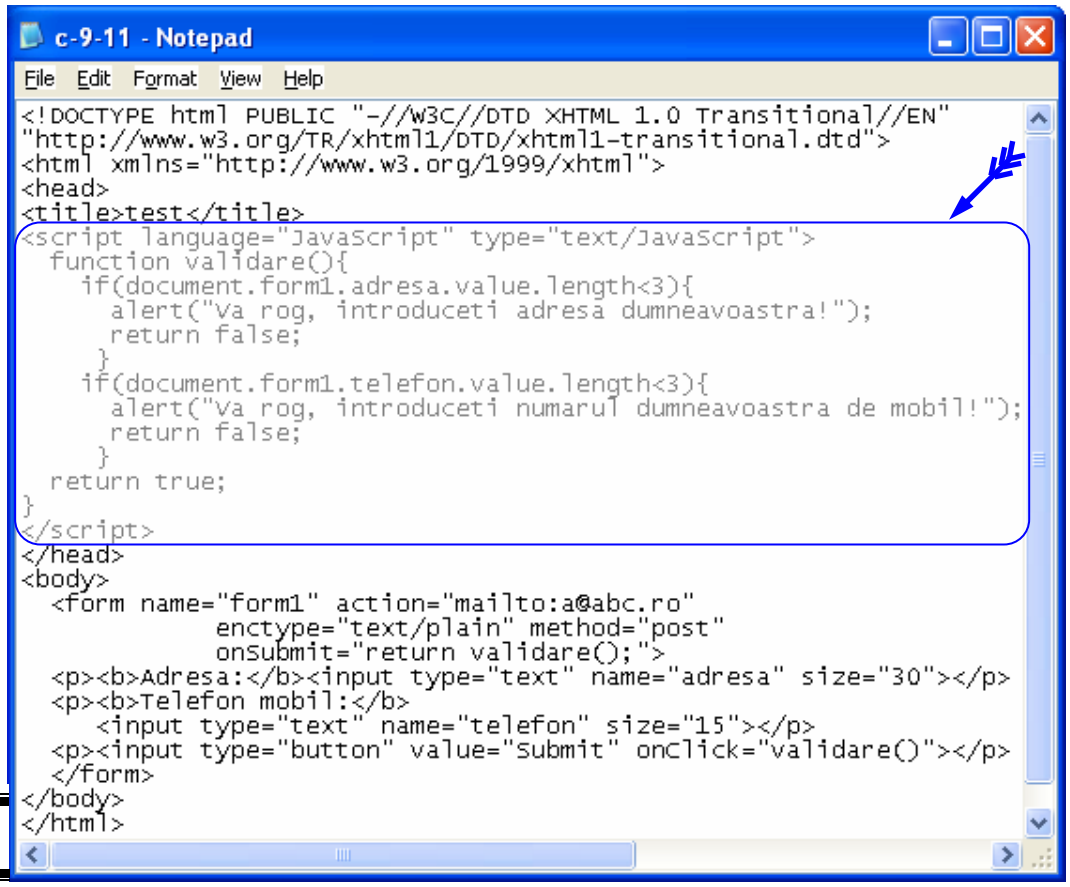


Figura 8.81

EXEMPLUL 8 JAVASCRIPT (varianta 1)

Vom aborda şi în cadrul acestei conversații aceeaşi problemă pe care am abordat-o în conversația precedentă (EXEMPLUL 7 JAVASCRIPT), cu singura deosebire (importantă!) că datele sunt introduse cu ajutorul unui formular (X)HTML, care dă un plus de interactivitate paginilor Web.

Datele care se vor introduce vor fi validate, iar în caz de eroare se vor afişa mesaje corespunzătoare. Se vor realiza două variante:

- ✓ *Varianta 1* – Introducerea livrărilor se face într-o zonă text asociată fiecărei zile şi fiecărui rezervor. Afişarea rezultatelor se face în aceeaşi fereastră în zone de text distincte.
- ✓ *Varianta 2* – Introducerea livrărilor se face printr-o singură zonă de text. Selecția rezervorului şi a zilei se face printr-o listă de selectare. Afişarea rezultatelor se face într-o fereastră distinctă.

❑ Specificații de programare

În figurile 8.82 şi 8.83 sunt prezentate: ecranul (intrare/ieşire) cu „Situația livrărilor de benzină pentru rezervoarele R1 R2 R3” şi specificațiile de programare. Mesajele de validare a datelor şi tabela de variabile sunt prezentate în figura 8.84, respectiv figura 8.85.

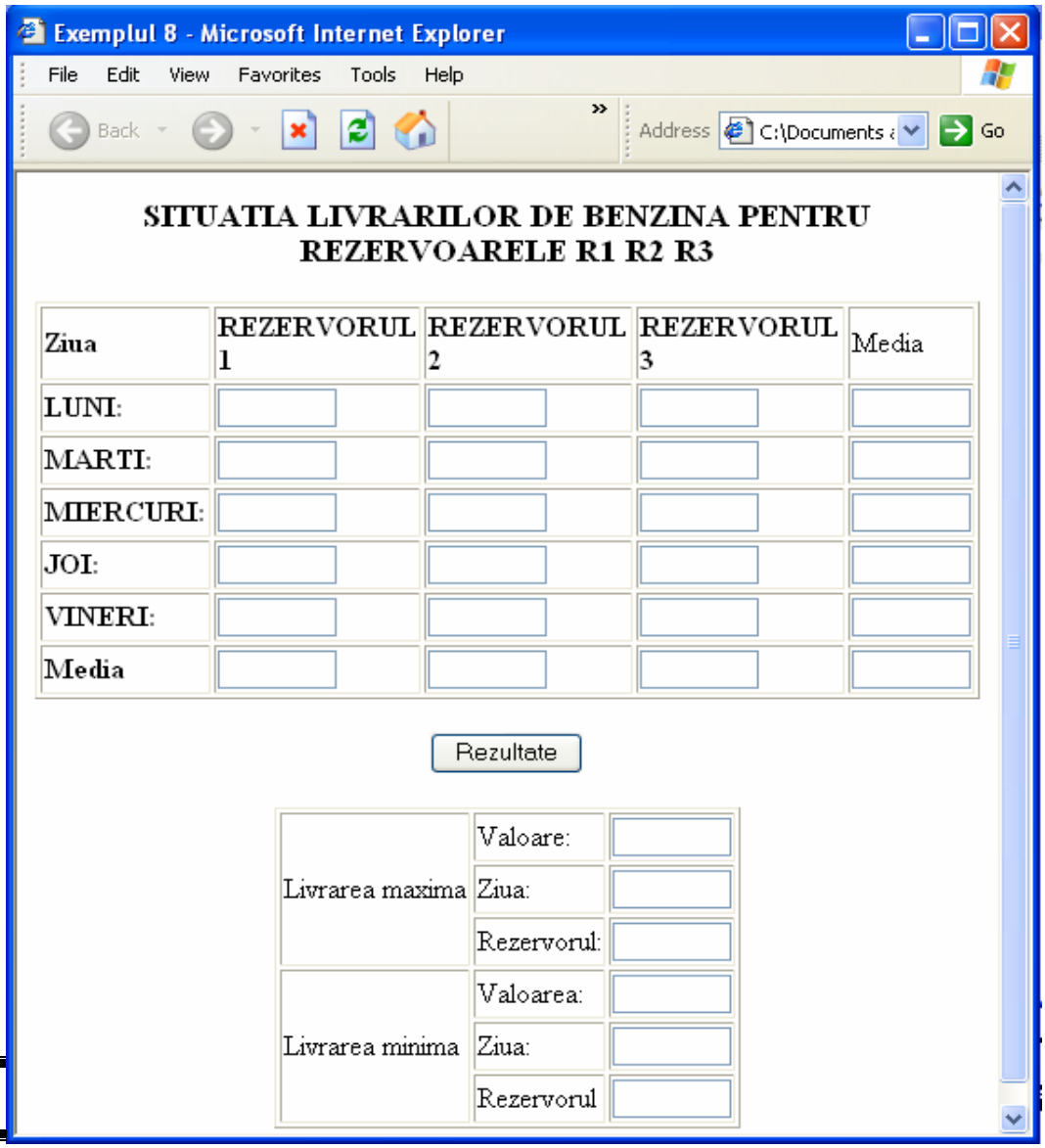


Figura 8.82

Specificații de programare

Descriere. Programul editează într-o pagină Web situația livrărilor de benzină efectuate zilnic din trei rezervoare cilindrice echilaterale (R1, R2, R3). De asemenea programul afişează livrările maxime şi minime (valoare, ziua, rezervor). Afişarea tuturor rezultatelor se realizează atunci când se execută clic pe butonul `Rezultate` (vezi figura 8.82).

Livrările, pe zile (luni, marți, miercuri, joi, vineri) şi pe rezervoare (R1, R2, R3) se introduc într-un formular (vezi figura 8.82) de la tastatură.

Intrări. Valorile livrărilor, pe zile, pentru fiecare rezervor (R1, R2, R3). În situația în care datele de intrare nu sunt numerice şi nu respectă intervalul (0, 20) se generează un mesaj de eroare.

Figura 8.83

Ieşiri. Media livrărilor pe zile şi pe rezervoare (în cadrul aceluiaşi formular); livrarea maximă (valoare, ziua, rezervor); livrarea minimă (valoare, ziua, rezervor).

Lista de funcțiuni ale programului

1. Creare şi inițializare matrice livrări.
2. Validare date.
3. Trunchiere valori.
4. Plasare valori valide (întregi) în formular.
5. Plasare valoare în matrice livrări.
6. Calculul mediilor livrărilor pe zile şi pe rezervoare.
7. Afişarea rezultatelor.
8. Stop.

Figura 8.83 *(continuare)*

Mesaje

Mesaj	***Descriere***
Valoare greşită pentru ...! Caractere invalide.	Valorile introduse nu sunt numerice.
Valoare greşită pentru ...! Valoarea trebuie să fie >0.	Valoarea introdusă este negativă.
Valoare greşită pentru ...! Valoarea trebuie să fie <20.	Valoarea introdusă este mai mare ca 20.

Figura 8.84

Tabela de variabile

Variabile de intrare	***Variabile de stare***	***Variabile de ieşire***
T11,T12, T13,T21, T22, T23, T31, T32, T33, T41, T42, T43, T51, T52, T53: Obiecte, zone de editare text pentru introducerea valorilor livrărilor pe fiecare rezervor, în fiecare zi	st: (real) folosit pentru calculul sumei livrărilor totale pe zile şi rezervoare s: (real) folosit pentru calculul sumelor parțiale pe zile rval: (logic) indică faptul că valoarea introdusă în zona de editare este validă sau nu x: (real) valoarea reală a textului introdus în zona de editare Z: (vector) numele zilelor săptămânii imax,imin,jmax,jmin: numere întregi, păstrează indicii livrărilor maxime şi minime din matricea a	a: matrice de numere reale, păstrează valorile livrărilor pe zile şi rezervoare M1, M2,M3, M4, M5, M6, T61, T62, T63, M6, vmax, zmax, vmin, zmin, rmax, rmin: Obiecte, zone de text pentru afişarea rezultatelor mediilor b: vector de numere reale, păstrează mediile livrărilor pe zile d: vector de numere reale, păstrează mediile livrărilor pe rezervor max,min: numere reale, păstrează valoarea maximă şi minimă pentru livrări

Figura 8.85

❑ Documentația de proiectare

Pseudocodul pentru EXEMPLUL 8 JAVASCRIPT (*Varianta 1*) este prezentat în figura 8.85.

Pseudocodul

```
EXEMPLUL8   BEGIN
            //initializeaza vectorul Z cu numele zilelor
            z=luni,marti,miercuri,joi vineri
            //aloca spatiu de memorie si
            //initializeaza matricea livrarilor
FORI        FOR(i=0;i<3;i++)
FORJ          FOR(j=0;j<5;j++)
                a[i,j]=0.0
FORJ          ENDFOR
FORI        ENDFOR
            //raspunde la evenimentele generate de  butonul Rezultate
IF1         IF(se apasa butonul Rezultate)
                DO calculeaza
IF1         ENDIF
            //raspunde la evenimentele generate de zonele de editare
IF2         IF(se paraseste zona de editare)
                DO valideaza(zona_de_editare,0,20)
IF2         ENDIF
EXEMPLUL8   END
// transforma o valoare reala in sir de caractere
// si trunchiaza la doua zecimale
TRUNCHIAZA  BEGIN
            Parametrii:
                x- valoare reala
            s=transforma in sir de caractere x
            i=pozitia punctului zecimal in sir
IF3         IF(i≠-1)
                s=copiazasubsir(s,0,i+2)
IF3         ENDIF
            RETURN s
TRUNCHIAZA  END

VALIDEAZA   BEGIN
            //valideaza valoarea introdusa in zona de text
            Date intrare:
                item-zona de text care a generat evenimentul onBlur
                min- valoarea minima permisa
                max- valoarea maxima permisa
            rval=fals
            x=transforma in real item.value
IF4         IF(x nu este numar)
                Afiseaza mesaj de eroare: Valoare gresita pentru cantitate
IF4           ELSE
IF5             IF(x<min)
                    Afiseaza mesaj de eroare:  Valoarea este prea mica
IF5               ELSE
IF6                 IF(x>max)
                    Afiseaza mesaj eroare: Valoarea este prea mare
IF6                 ELSE
                      Rval=adevarat
                        DO puneVal(item,x)
```

Figura 8.86

```
IF6                         ENDIF
IF5                 ENDIF
IF4           ENDIF
              RETURN rval
VALIDEAZA     END

//depune valoarea din zona de editare in matricea livrarilor a
PUNEVAL       BEGIN
              Date intrare:
                    item-zona de text care a generat evenimentul onBlur
                    x- valoarea ce va fi depusa
                 s=item.nume
               s1=extrage_subsirul(s,1,2)
              sir=transforma_in_integer(s1)-1
              s1= extrage_subsirul(s,2,3)
              ic= transforma_in_integer(s1)-1
              a_ir,ic=transforma_in_real(T1.value)
PUNEVAL       END

// calculeaza mediile determina valorile minime si maxime si afiseaza rezultatele
CALCULEAZA    BEGIN
              // aloca spatiu de memorie pentru vectorul b si d
              // calculeaza mediile pe rezervoare
FORJ          FOR(j=0;j<5;j++)
                   s=0
FORI           FOR(i=0;i<3;i++)
                     s=s+a_i,j
FORI           ENDFOR
               b_j=s/3
FORJ          ENDFOR
                // calculul mediilor pe rezervoare
              st=0
FORI1         FOR(i=0;i<3;i++)
                    s=0
FORJ1               FOR(j=0;j<5;j++)
                          s=s+a_i,j
                         st=st+a_ij
FORJ1               ENDFOR
                    d_i=s/5
FORI1         ENDFOR
              d_3=s/15
              // determinarea maximului si minimului
              max=a_0,0
              min=a_0,0
              imax=0
              imin=0
              jmax=0
              jmin=0
FORI2         FOR(i=0;i<3;i++)
FORJ2               FOR(j=0;j<5;j++)
IF7                       IF(max<a_i,j)
                                  max=a_ij
                                  imax=i
                                  jmax=j
```

Figura 8.86
(continuare)

```
IF7                         ENDIF
IF8                         IF(min>a[i,j])
                                    min=a[ij]
                                    imin=i
                                    jmin=j
IF8                         ENDIF
FORJ2               ENDFOR
FORI2         ENDFOR
              imin=imin+1
              imax=imax+1
              //afisare rezultate
              M1.value=trunchiaza(B[0])
              M2.value=trunchiaza(B[1])
              M3.value=trunchiaza(B[2])
              M4.value=trunchiaza(B[3])
              M5.value=trunchiaza(B[4])
              T61.value=trunchiaza(D[0])
              T62.value=trunchiaza(D[1])
              T63.value=trunchiaza(D[2])
              M6.value=trunchiaza(D[3])
              vmax.value=trunchiaza(max)
              vmin.value=trunchiaza(min)
              rmax.value="R"+imax
              rmin.value="R"+imin
              zmax.value=Z[jmax]
              zmin.value=Z[jmin]
CALCULEAZA    END
```

Figura 8.86 *(continuare)*

❑ Codificarea în limbajul JavaScript

Documentul complet (X)HTML, în care s-a inserat script-ul (programul JavaScript) este prezentat în figura 8.87.

```
<html>
<head>
<title>Exemplul 8</title>
<script language="JavaScript">
<!---
a=new Array(3);
a[0]=new Array(5);
a[1]=new Array(5);
a[2]=new Array(5);
for(i=0;i<3;i++)
 for(j=0;j<5;j++)
     a[i][j]=0.0;
var Z = new Array("Luni","Marti","Miercuri","Joi","Vineri");
  function trunchiaza(x) {
     var s=""+x;
      i=s.indexOf(".");
      if(i!=-1){
        s=s.substring(0,i+3);
        }
      return s;
      }
```

Figura 8.87

```
function puneVal(item,x) {
 s=item.name;
 var ic=parseInt(s.substring(1,2))-1;
 var ir=parseInt(s.substring(2,3))-1;
 a[ir][ic]=x;
}
function valideaza(item, min, max) {
   var rVal = false;
   var x=parseFloat(item.value);
     if(isNaN(x)){
       alert("Valoare gresita pentru " + item.name + "!Caractere invalide");
       }
      else
     if (x < min)
    alert("Valoare gresita pentru " + item.name + "!Valoarea trebuie >" + min);
      else if (x> max)
       alert("Valoare gresita pentru " + item.name + "!  Valoarea trebuie sa fie < " + max);
          else {
      rVal = true;
      puneVal(item,x);
     }
   return rVal;
}

function calculeaza()  {
    var i,j;
    B = new Array(5);
    for(j=0;j<5;j++) {
       S=0;
        for(i=0;i<3;i++)
           S=S+a[i][j];
       B[j]=S/3;
     }
    // CALCULUL MEDIILOR PE REZERVOARE
    D = new Array(4);
    ST=0;
    for(i=0;i<3;i++) {
       S=0;
        for(j=0;j<5;j++){
             S=S+a[i][j];
            ST=ST+a[i][j];
         }
       D[i]=S/5;
     }
     D[3]=ST/15;

// DETERMINAREA MAXIMULUI SI MINIMULUI
    max=a[0][0];
    min=a[0][0];
    imax=0;imin=0;
     jmax=0;jmin=0;
     for(i=0;i<3;i++){
       for(j=0;j<5;j++){
```

Figura 8.87 *(continuare)*

```
        if(max<a[i][j]){max=a[i][j];imax=i;jmax=j;}
        if(min>a[i][j]){min=a[i][j];imin=i;jmin=j;}
   }}
  imin++;imax++;
document.input_form.M1.value=trunchiaza(B[0]);
document.input_form.M2.value=trunchiaza(B[1]);
document.input_form.M3.value=trunchiaza(B[2]);
document.input_form.M4.value=trunchiaza(B[3]);
document.input_form.M5.value=trunchiaza(B[4]);
document.input_form.T61.value=trunchiaza(D[0]);
document.input_form.T62.value=trunchiaza(D[1]);
document.input_form.T63.value=trunchiaza(D[2]);
document.input_form.M6.value=trunchiaza(D[3]);
document.f1.vmax.value=trunchiaza(max);
document.f1.vmin.value=trunchiaza(min);
document.f1.rmax.value="R"+imax;
document.f1.rmin.value="R"+imin;
document.f1.zmax.value=Z[jmax];
document.f1.zmin.value=Z[jmin];
    }

// -->
</SCRIPT>
</head>

<body>

<center><h3>SITUATIA LIVRARILOR DE BENZINA PENTRU REZERVOARELE R1 R2
R3</h3>
<form name = "input_form">
<table Border=1>
<tr><td><b>Ziua</b><td><b>REZERVORUL 1</b><td><b>REZERVORUL 2</b>
<td><b>REZERVORUL 3</b><td> Media</td></tr>
<tr>
<td><b>LUNI:</b></td>
<td><input type="text" name="T11" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T12" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T13" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="M1" size="7" readonly>

<tr>
<td><b>MARTI:</b>
<td><input type="text" name="T21" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T22" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T23" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="M2" size="7" readonly>
<tr>
<td><b>MIERCURI:</b>
<td><input type="text" name="T31" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T32" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T33" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="M3" size="7" readonly>
```

Figura 8.87 *(continuare)*

```
<tr>
<td><b>JOI:</b>
<td><input type="text" name="T41" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T42" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T43" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="M4" size="7" readonly>

<tr>
<td><b>VINERI:</b>
<td><input type="text" name="T51" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T52" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="T53" size="7" onBlur="valideaza(this,0,20);">
<td><input type="text" name="M5" size="7" readonly>
<tr>
<td><b>Media</b>
<td><input type="text" name="T61" size="7" readonly>
<td><input type="text" name="T62" size="7" readonly>
<td><input type="text" name="T63" size="7" readonly>
<td><input type="text" name="M6" size="7" readonly>
</tr>
</table>
<p><input type="button" value="Rezultate" name="B1" onClick="calculeaza();"></p>
</form>
<P>
<P>
<P>
<form name="f1">
<table border=1><tr><td rowspan=3>
  Livrarea maxima <td>Valoare:<td><input type="text" name="vmax" size="7" readonly>
<tr>
<td>Ziua:<td><input type="text" name="zmax" size="7" readonly>
<tr>
<td>Rezervorul:<td><input type="text" name="rmax" size="7" readonly>
<tr><td ROWSPAN=3>
Livrarea minima<td>Valoarea:<td><input type="text" name="vmin" size="7" readonly>
<tr>
<td>Ziua:<td><input type="text" name="zmin" size="7" readonly>
<tr>
<td>Rezervorul<td><input type="text" name="rmin" size="7" readonly>
</table>
</body>
</html>
```

Figura 8.87 *(continuare)*

Comentarii:

- ✓ Script-ul (inserat în secțiunea `<head>` a documentului HTML) conține patru funcții: `trunchiaz•(x); puneVal(item, x); validate(item, min, max); compute()`.
- ✓ Pentru calculul mediei mediilor livrărilor s-a utilizat instrucțiunea de atribuire
  D[3]=ST/15
  unde:
  D este vectorul mediilor livrărilor pe rezervoare iar ST reprezintă total livrări.
- ✓ S-au folosit gestionarii de evenimente `onBlur`, cu funcția de validare date `validate()` pentru obiectele zone de text şi `onClick` cu funcția `compute()` pentru butonul de calcul rezultate.

Vizualizați documentul într-un browser (figura 8.88) şi testați script-ul.

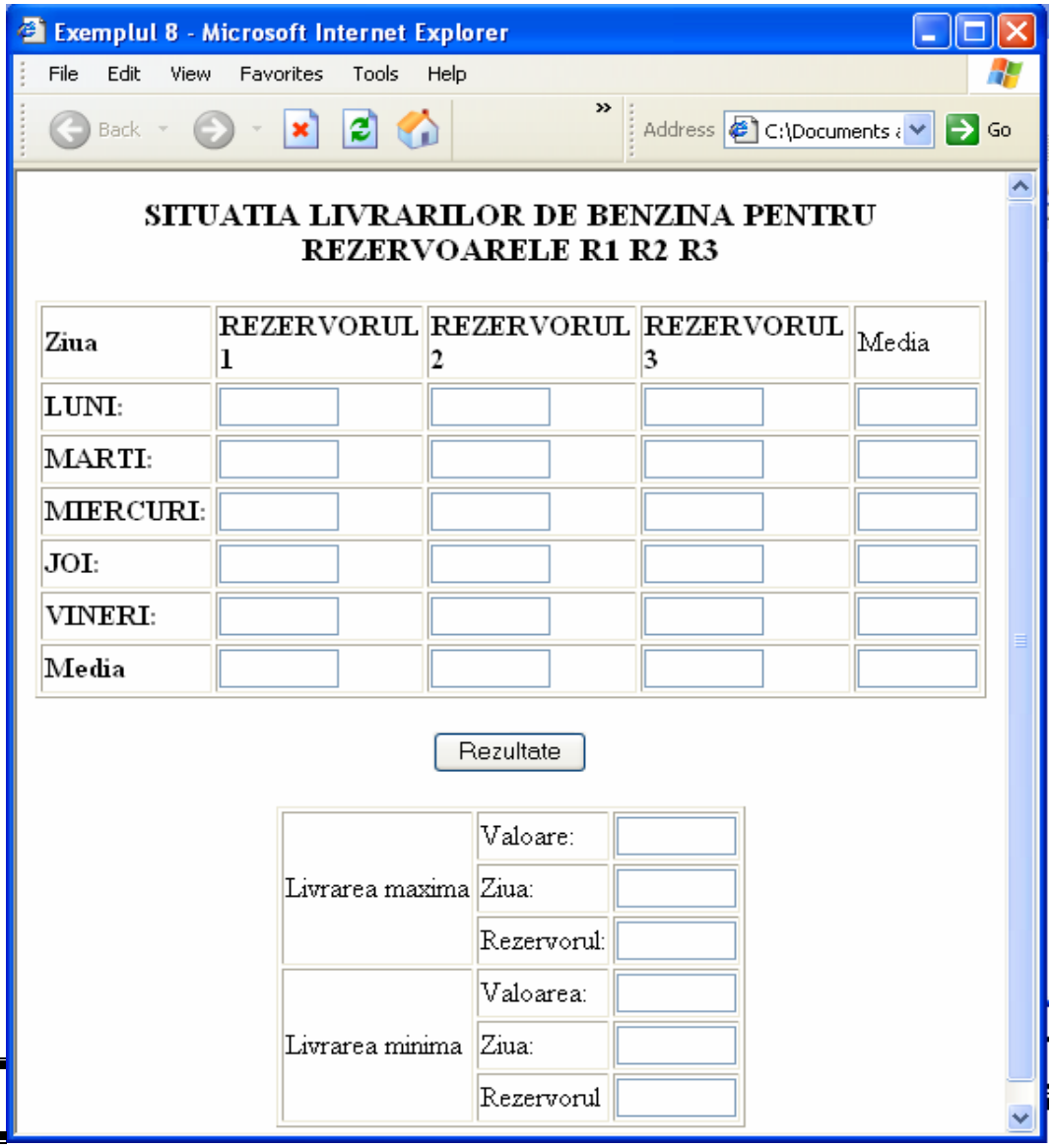


Figura 8.88

În figura 8.89 se prezintă rezultatele execuției programului JavaScript pentru un set de date ilustrat în cadrul formularului afişat.

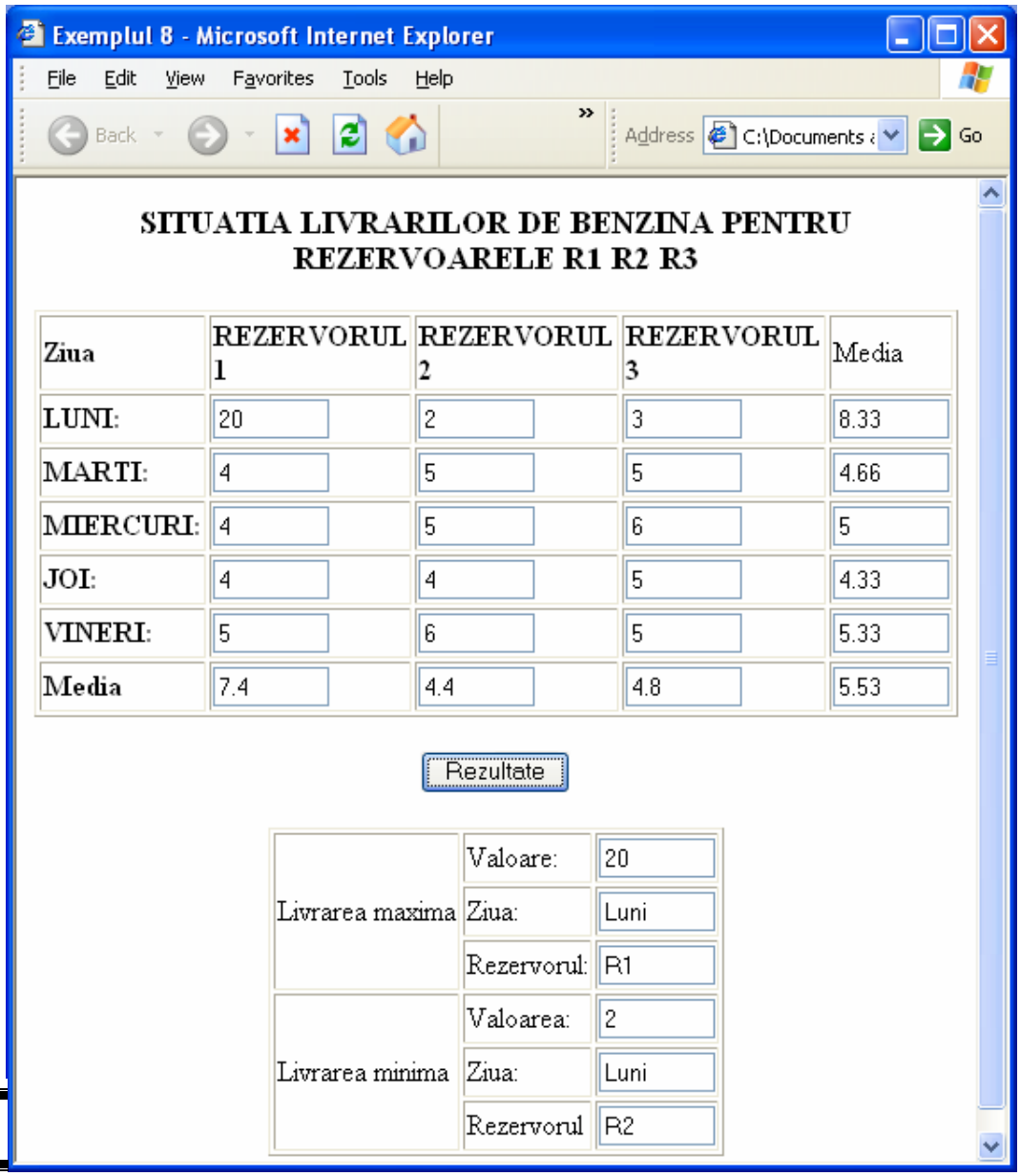


Figura 8.89

EXEMPLUL 8 JAVASCRIPT (varianta 2)

❑ Specificații de programare

În figurile 8.90, 8.91, 8.92 şi 8.93 sunt prezentate: ecranul pentru introducerea livrărilor (pe zile şi pe rezervoare); ecranul cu „Situația livrărilor de benzină pentru rezervoarele R1, R2, R3; specificațiile de programare respectiv tabela de variabile.

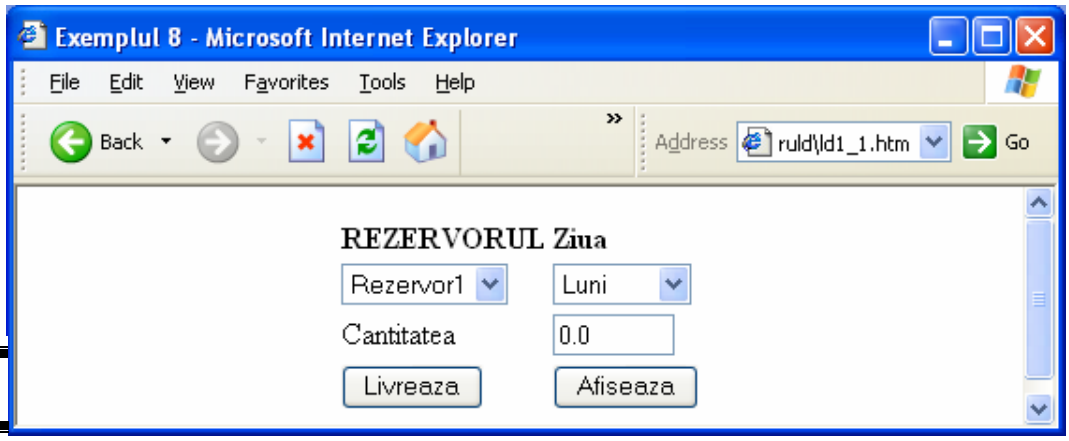


Figura 8.90

Figura 8.91

Specificații de programare

Descriere. Programul editează într-o pagină Web situația livrărilor de benzină efectuate zilnic din trei rezervoare cilindrice echilaterale (R1, R2, R3).

Programul afişează de asemenea livrările maxime şi minime cu precizarea numărului rezervorului şi a zilei în care s-a efectuat livrarea respectivă. Afişarea tuturor rezultatelor se face într-o fereastră distinctă, în momentul în care se execută clic pe butonul `Afi•eaz•` (vezi figura 8.90).

Figura 8.92

Livrările, pe zile (luni, marți, miercuri, joi, vineri) şi pe rezervoare (Rezervor1, Rezervor2, Rezervor3) se introduc printr-o singură zonă de text a unui formular (vezi figura 8.90). Selecția rezervorului şi a zilei se face printr-o listă de selectare.

Intrări. Valorile livrărilor, pe zile, pentru fiecare rezervor se introduc printr-o singură zonă de text a unui formular. În caz de eroare – datele de intrare nu sunt numerice şi nu respectă intervalul (0, 20) se generează un mesaj de eroare.

La acționarea butonului `Livreaz•` se vor depune livrările din zona `Cantitatea` în matricea `a`, pe linia şi coloana corespunzătoare rezervorului şi zilei respective (vezi funcția `Livreaz•()`).

Ieşiri. Situația livrărilor, mediile (pe zile şi pe rezervoare) sub formă de tabel; livrarea maximă (valoare, ziua, rezervor); livrarea minimă (valoare, ziua, rezervor).

Lista de funcțiuni ale programului

1. Inițializează vectorul z cu numele zilelor săptămânii.
2. Alocă spațiu de memorie şi inițializează matricea livrărilor.
3. Răspunde la evenimentele generate de butonul `Livreaz•`.
4. Răspunde la evenimentele generate de butonul `Afi•eaz•`.
5. Răspunde la evenimentele generate de zona de editare `T1`.
6. Transformă în şir de caractere şi trunchiază la două zecimale.
7. Validează valoarea introdusă în zona de text.
8. Depune valoarea din zona de editare în matricea livrărilor `a`.
9. Calculează mediile pe zile şi pe rezervoare.
10. Determină valorile maxime şi minime.
11. Afişează rezultatele în fereastra `RezWindow`.
12. Stop.

Figura 8.92 *(continuare)*

Tabela de variabile

Variabile de intrare	*Variabile de stare*	*Variabile de ieşire*
Rezervor: obiect, listă simplă de selecție pentru cele trei rezervoare Zile: obiect, listă simplă de opțiuni pentru selecția zilei din săptămână T1: obiect, zona de editare în care se va introduce valoarea livrărilor	st: (real) folosit pentru calculul sumei livrărilor totale pe zile şi rezervoare s: (real) folosit pentru calculul sumelor parțiale pe zile rval: (logic) indică faptul că valoarea introdusă în zona de editare este validă sau nu x: (real) valoarea reală a textului introdus în zona de editare Z: (vector) numele zilelor săptămânii imax,imin,jmax,jmin: (numere întregi) păstrează indicii livrărilor maxime şi minime din matricea a	RezWindow: obiect, fereastra în care se vor afişa rezultatele b: vector de numere reale, păstrează mediile livrărilor d: vector de numere reale, păstrează mediile livrărilor a: matrice de numere reale, păstrează valorile livrărilor pe zile şi rezervoare max,min: numere reale, păstrează valoarea maximă şi minimă pentru livrări

Figura 8.93

❑ Documentația de proiectare

Pseudocodul pentru EXEMPLUL 8 JAVASCRIPT (*Varianta 2*) este prezentat în figura 8.94.

Pseudocodul

```
EXEMPLUL8   BEGIN
//initializeaza vectorul Z cu numele zilelor
            z=luni,marti,miercuri,joi vineri
//aloca spatiu de memorie si initializeaza matricea livrarilor
FOR1        FOR(i=0;i<3;i++)
FOR2          FOR(j=0;j<5;j++)
                a[i,j]=0.0
FOR2          ENDFOR
FOR1        ENDFOR
            Initializeaza RezWindow
            //raspunde la evenimentele generate de butonul Livreaza
IF1         IF(se apasa butonul Livreaza)
                    DO livreaza
IF1         ENDIF
            //raspunde la evenimentele generate de butonul Afiseaza
IF2         IF(se apasa butonul Afiseaza)
                    DO afis
IF2         ENDIF
            //raspunde la evenimentele generate de zona de editare T1
IF3         IF(se paraseste zona T1)
                    DO valideaza(T1,0,20)
IF3         ENDIF
EXEMPLUL8   END

// transforma in sir de caractere si trunchiaza la doua zecimale
TRUNCHIAZA  BEGIN
            Parametrii:
                    x- valoare reala
            s=transforma in sir de caractere x
            i=pozitia punctului zecimal in sir
IF4         IF(i≠-1)
                    s=copiazasubsir(s,0,i+2)
IF4         ENDIF
            RETURN s
TRUNCHIAZA  END

VALIDEAZA   BEGIN
            //valideaza valoarea introdusa in zona de text
            Date intrare:
                    item-zona de text care a generat evenimentul onBlur
                    min- valoarea minima permisa
                    max- valoarea maxima permisa
            rval=fals
            x=transforma in real item.value
IF5         IF(x nu este numar)
                    Afiseaza mesaj de eroare:
                    Valoare gresita pentru cantitate
IF5           ELSE
```

Figura 8.94

```
IF6                 IF(x<min)
                            Afiseaza mesaj de eroare:
                              Valoarea este prea mica
IF6                  ELSE
IF7                         IF(x>max)
                            Afiseaza mesaj eroare:
                                    Valoarea este prea mare
IF7                         ELSE
                             Rval=adevarat
IF7                         ENDIF
IF6                 ENDIF
IF5           ENDIF
              RETURN rval
VALIDEAZA     END

//depune valoarea din zona de editare in matricea livrarilor a
LIVREAZA      BEGIN
              ir=indicele elementului selectat in lista Rezervor
              ic=indicele elementului selectat in lista Zile
              air,ic=transforma_in_real(T1.value)
LIVREAZA      END

// calculeaza mediile determina valorile minime si maxime si afiseaza rezultatele
AFIS          BEGIN
              // aloca spatiu de memorie pentru vectorul b si d
              // calculeaza mediile pe rezervoare
FOR3          FOR(j=0;j<5;j++)
                      s=0
FOR4                  FOR(i=0;i<3;i++)
                             s=s+a_{i,j}
FOR4                  ENDFOR
                      b_j=s/3
FOR3          ENDFOR
              // calculul mediilor pe rezervoare
              st=0
FOR5          FOR(i=0;i<3;i++)
                       s=0
FOR6                   FOR(j=0;j<5;j++)
                              s=s+a_{i,j}
                               st=st+a_{ij}
FOR6                   ENDFOR
                       d_i=s/5
FOR5          ENDFOR
              d_3=s/15
              // determinarea maximului si minimului
              max=a_{0,0}
              min=a_{0,0}
              imax=0
              imin=0
              jmax=0
              jmin=0
FOR7          FOR(i=0;i<3;i++)
FOR8                  FOR(j=0;j<5;j++)
```

Figura 8.94 *(continuare)*

```
IF8                 IF(max<a[i,j])
                            max=a[ij]
                            imax=i
                            jmax=j
IF8                 ENDIF
IF9                 IF(min>a[i,j])
                            min=a[ij]
                            imin=i
                            jmin=j
IF9                 ENDIF
FOR8                ENDFOR
FOR7          ENDFOR
              imin=imin+1
              imax=imax+1
              //afisare rezultate
IF10          IF(RezWindow exista)
                    Închide RezWindow
IF10          ENDIF
              Creaza RezWindow
               //scrierea rezultatelor in fereastra RezWindow
              RezWindow.WRITE "SITUATIA REZERVOARELOR R1 R2 R3"
              RezWindow.WRITE "ZIUA  R1  R2  R3  MEDIA"
FOR9          FOR(k=0;k<5;k++)
              RezWindow.WRITE Z[k]
FOR10               FOR(j=0;j<3;j++)
                            RezWindow.WRITE  a[j][k]
FOR10                       ENDFOR
                    RezWindow.WRITE  trunchiaza( b[k])
FOR9          ENDFOR
              RezWindow.WRITE  "MEDIA"
FOR11         FOR(j=0;j<4;j++)
                    RezWindow.WRITE trunchiaza(d[j])
FOR11         ENDFOR
              RezWindow.WRITE "Livrarea maxima: " max
              RezWindow.WRITE "s-a facut din rezervorul: R" imax
              RezWindow.WRITE "in ziua de" Z[jmax]
              RezWindow.WRITE "Livrarea minima:" min
              RezWindow.WRITE " s-a facut din rezervorul: R"imin
              RezWindow.WRITE " in ziua de "+Z[jmin]
AFIS          END
```

Figura 8.94 *(continuare)*

❑ Codificarea în limbajul JavaScript

Documentul complet (X)HTML este prezentat în figura 8.95.

```
<html>
<head>
<title>Exemplul 8</title>
<script language="JavaScript">
<!---
var Z = new Array("Luni","Marti","Miercuri","Joi","Vineri");
  function trunchiaza(x) {
```

```
    var s=""+x;
     i=s.indexOf(".");
     if(i!=-1){
       s=s.substring(0,i+3);
       }
     return s;
     }
function validate(item, min, max) {
  var rVal = false;
  var x=parseFloat(item.value);
    if(isNaN(x))
      alert("Valoare gresita pentru cantitate!");
     else
    if (x < min)
   alert("Valoare gresita pentru cantitate!Valoarea trebuie >" + min);
     else if (x> max)
      alert("Valoare gresita pentru cantitate!  Valoarea trebuie sa fie < " + max);
        else
     rVal = true;
  return rVal;
}
a=new Array(3);
a[0]=new Array(5);
a[1]=new Array(5);
a[2]=new Array(5);
for(i=0;i<3;i++)
 for(j=0;j<5;j++)
     a[i][j]=0.0;
var RezWindow=null;

function livreaza() {
 var ir=f1.Rezervor.selectedIndex;
 var z=f1.Zile.selectedIndex;
  a[ir][z]+=parseFloat(f1.T1.value);
}
function afis(){
  // CALCULUL MEDIILOR PE ZILE
    var i,j;
    B = new Array(5);
    for(j=0;j<5;j++)
     {
      S=0;
       for(i=0;i<3;i++)
          S=S+a[i][j];
      B[j]=S/3;
     }
```

Figura 8.95

```
    // CALCULUL MEDIILOR PE REZERVOARE
    D = new Array(4);
    ST=0;
    for(i=0;i<3;i++) {
       S=0;
        for(j=0;j<5;j++){
            S=S+a[i][j];
           ST=ST+a[i][j];
         }
       D[i]=S/5;
      }
     D[3]=ST/15;
 // DETERMINAREA MAXIMULUI SI MINIMULUI
    max=a[0][0];
    min=a[0][0];
    imax=0;imin=0;
     jmax=0;jmin=0;
     for(i=0;i<3;i++){
       for(j=0;j<5;j++){
         if(max<a[i][j]){max=a[i][j];imax=i;jmax=j;}
         if(min>a[i][j]){min=a[i][j];imin=i;jmin=j;}
   }}
  imin++;imax++;
    //AFISARE REZULTATE
 if(RezWindow!=null)RezWindow.close();
 RezWindow=window.open("","   'toolbar=yes,scrollbars=yes,menubar=no,
width=500,height=300');
 RezWindow.document.writeln("<center><p>SITUATIA REZERVOARELOR R1 R2
R3</p></center>");
 RezWindow.document.writeln("<center><table border=1 bgcolor=#EFEFFF><tr>");
 RezWindow.document.writeln("<td><b>ZIUA</b><td><b>R1</b><td>
<b>R2</b><td><b>R3</b><td><b>MEDIA</b></td></tr>");
    for(k=0;k<5;k++) {
     RezWindow.document.writeln("<tr><td>" + Z[k]+"</td>");
     for(j=0;j<3;j++) {
       RezWindow.document.writeln("<td>" + a[j][k]+ "</td>");
      }
     RezWindow.document.writeln(" <td> " +trunchiaza( B[k])+ "</td></tr>");
    }
    RezWindow.document.writeln("<tr><td><b>MEDIA</b>");
   for(j=0;j<4;j++)
     RezWindow.document.writeln("<td><b>" + trunchiaza(D[j])+" </b></td>");
   RezWindow.document.writeln("</tr>");
   RezWindow.document.writeln("</table></center><p><p>");
   RezWindow.document.writeln("<font color=green>Livrarea maxima:"+max+" s-a facut din
rezervorul: R"+imax+" in ziua de "+Z[jmax]+"</font>");
RezWindow.document.writeln("<p><font color=green>Livrarea minima:"+min+" s-a facut din
rezervorul: R"+imin+" in ziua de "+Z[jmin]+"</font>");
RezWindow.document.writeln("</html>");
}
 // -->
</script>
</head>
```

Figura 8.95 *(continuare)*

```
<body>
<center>
<form name="f1">
<table border=0>
<tr><td><b>REZERVORUL</b><td><b>Ziua</b>
<tr><td><select size="1" name="Rezervor">
  <option selected>Rezervor1</option>
  <option>Rezervor2</option>
  <option>Rezervor3</option>
  </select>
<td>
<select size="1" name="Zile">
  <option>Luni</option>
  <option>Marti</option>
  <option>Miercuri</option>
  <option>Joi</option>
  <option>Vineri</option>
  </select>
<tr>
<td>Cantitatea
<td><input type="text" name="T1" size="7" value="0.0" onChange="validate(this,0,20);">
<tr>
<td><input type="button" value="Livreaza" onClick="trimite();">
<td><input type="button" value="Afiseaza" onClick="afis();">
</table>
</center>
</form>
</body>
</html>
```

Figura 8.95 *(continuare)*

Comentarii:

- ✓ Script-ul (inserat în secțiunea `<head>` a documentului XHTML) conține patru funcții: `trunchiaz•(x);` `ivalidate(item, min, max);` `livreaz•();` `afi•()`.
- ✓ Formularul conține subobiectele: `Select` şi `Option`; `Text`; butoanele `Livreaz•` şi `Afi•eaz•`.
- ✓ Validarea datelor de intrare introduse în zona de text se realizează cu gestionarul de evenimente `onBlur` (vezi funcția `valideaz•()`).
- ✓ Depunerea livrărilor din zona de editare (subobiectul `Text`) în matricea livrărilor se realizează cu gestionarul de evenimente `onClick` (vezi funcția `Livreaz•()`).
- ✓ Afişarea rezultatelor se realizează cu gestionarul de evenimente `onClick` (vezi funcția `afis()`). Prin acționarea butonului `Afi•eaz•` se deschide o nouă fereastră care va afişa: situația livrărilor; mediile livrărilor pe zile şi rezervoare; livrările maxime şi minime.
- ✓ Pentru crearea unei ferestre, am utilizat metoda `window.open()` (figura 8.96).

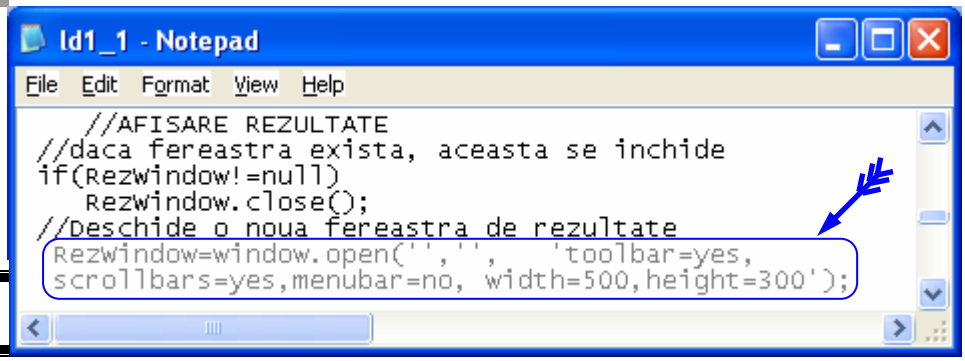


Figura 8.96

✓ Pentru crearea listei de selectare a rezervoarelor şi a zilelor săptămânii am utilizat subobiectele `Form`: `Select` şi `Option` (vezi figura 8.97).

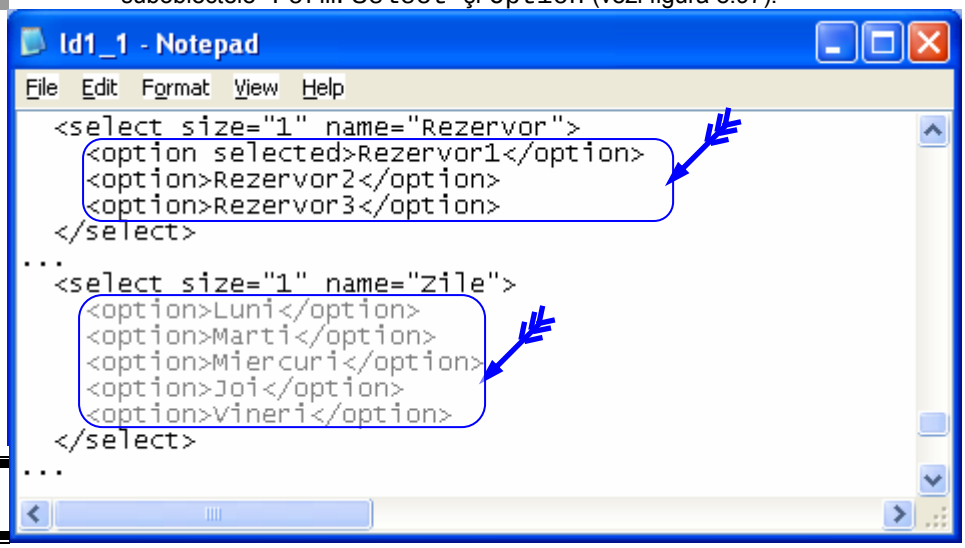


Figura 8.97

✓ Subobiectul `Select` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<select> ... </select>`. Subobiectul `Select` este unul dintre cele mai utile şi mai flexibile subobiecte ale obiectului Form. Subobiectul `Select` permite crearea următoarelor tipuri de liste:
- listă de selectare (o listă de opțiuni în care utilizatorul poate selecta un singur element. Pentru a vă asigura că lista afişează o singură linie la un moment dat atribuiți proprietății `size` valoarea 1 sau nu-l folosiți deloc).
- listă derulantă (o listă care afişează la un moment dat un anumit număr de elemente. Ea include bare de derulare. Pentru a defini o listă derulantă atribuiți lui `size` o valoare mai mare ca 1).
- listă derulantă cu selectări multiple (din acest tip de obiect `Select`, puteți selecta unul sau mai multe elemente. Pentru a defini o listă derulantă cu selectări multiple, adăugați proprietatea `multiple` la definirea obiectului `Select`).

✓ Subobiectul `Option` este versiunea „transpusă în obiect" a tag-ului (X)HTML `<option> ... </option>`.

Vizualizați documentul într-un browser (figura 8.98) şi testați script-ul.

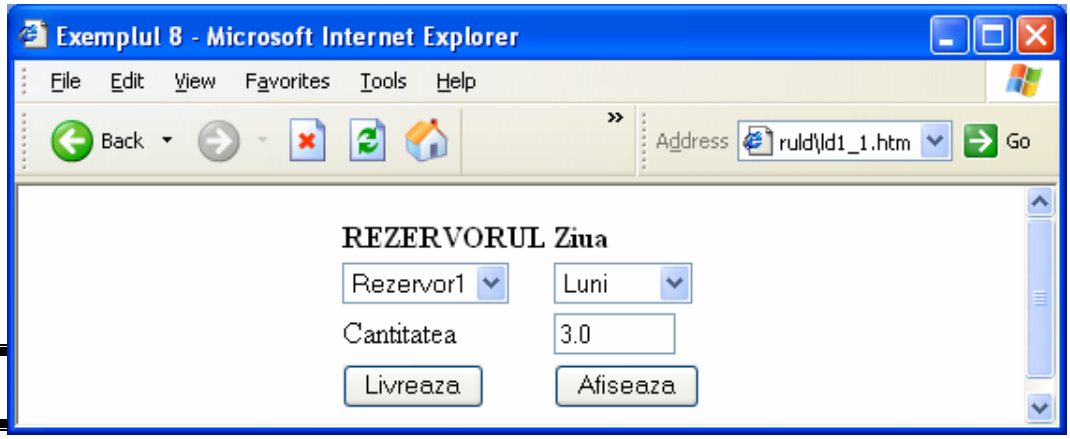


Figura 8.98

În figura 8.99 se prezintă rezultatele execuției programului JavaScript pentru un set de date afişat.

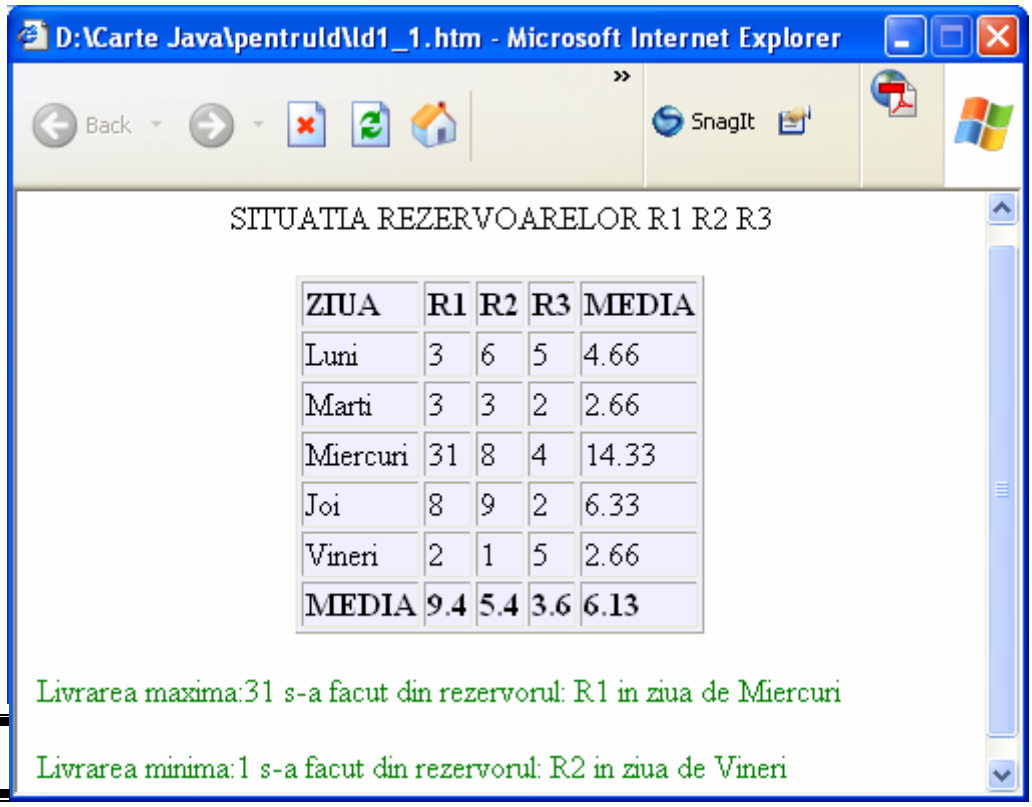


Figura 8.99

Aplicație

❑ Scrieți un program JavaScript pentru afişarea datelor conținute în formularul din figura 8.100, într-o nouă fereastră. Personalizați script-ul.

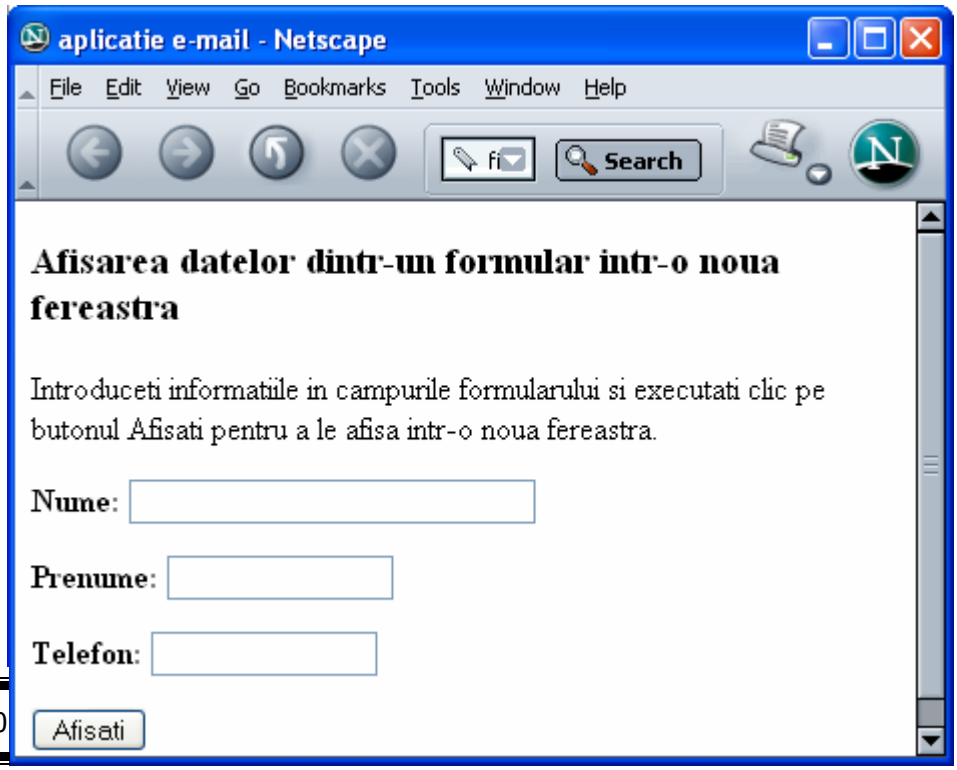


Figura 8.100

În figura 8.101 se prezintă documentul complet XHTML, în care s-a inserat script-ul aplicației.

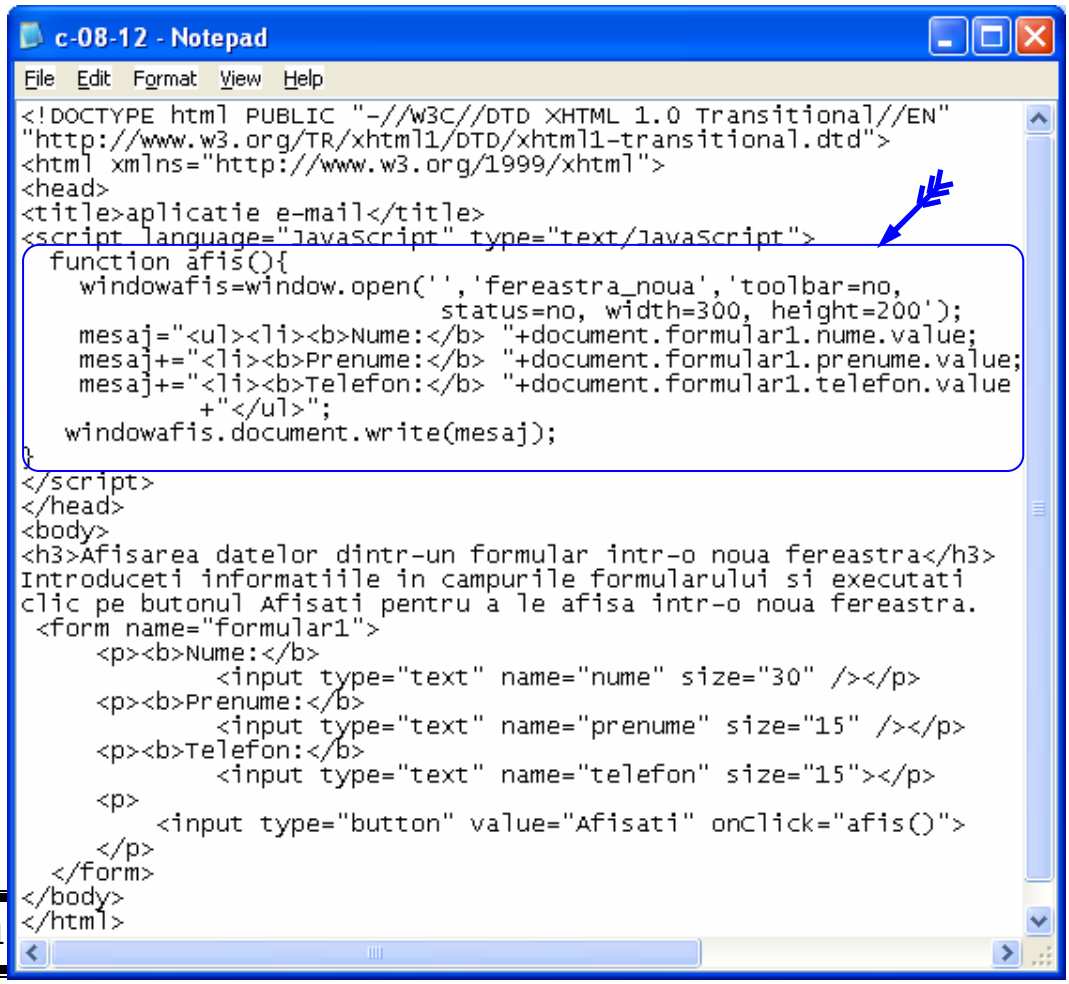


Figura 8.101

În figura 8.102 este prezentat rezultatul execuției script-ului.

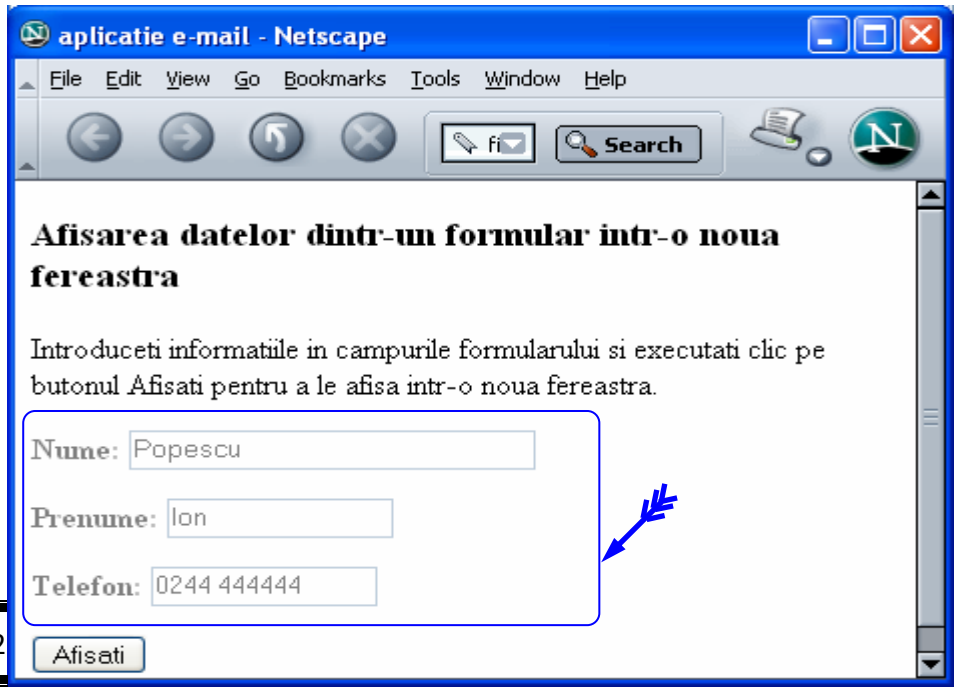


Figura 8.102

Testați script-ul (figura 8.103) executând clic pe butonul `Afi•a•i`.

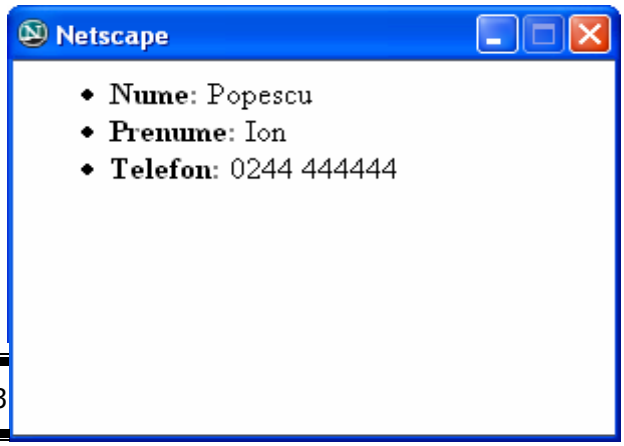
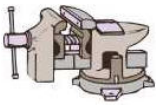


Figura 8.103

Remarcă. Informațiile din cadrul formularului sunt afişate într-o nouă fereastră (figura 8.103).

JavaScript — Temă

Testați-vă cunoştințele



1. Care din atributele tag-ului `<form>` determină locul unde vor fi expediate datele?
   - `action;`
   - `method;`
   - `name.`
2. Care sunt subobiectele obiectului `Form`?
3. Care sunt proprietăţile obiectului `Form`?
4. Care sunt metodele obiectului `Form`?
5. Care sunt gestionarii de evenimente ai obiectului `Form`?
6. Care este relaţia dintre obiectul Form şi tag-ul `<form>`?
7. Care este relaţia dintre subobiectele unui formular şi obiectul `Form`?
8. Care este sintaxa subobiectelor: `Text; Textarea; Submit; Checkbox; Radio; Select`?
9. Care sunt proprietăţile subobiectului `Select`?
10. Care sunt metodele subobiectelor: `Text; Textarea; Submit; Reset; Button; Checkbox; Select; Fileupload; Option`?

11. Care sunt gestionarii de evenimente au subobiectelor: `Reset`; `Option`; `Checkbox`.
12. Cum puteţi utiliza limbajul JavaScript pentru validarea unui formular?
13. Scrieţi un program JavaScript care calculează şi afişează produsul primelor n numere naturale. Utilizaţi un formular (X)HTML.
14. Scrieţi un program JavaScript care afişează un număr, intervalul [0, 9] în cuvinte. Utilizaţi un formular (X)HTML.
15. Scrieţi un program JavaScript care generează seria Ulam. Utilizaţi un formular XHTML.

Indicaţie. Seria Ulam se generează după cum urmează: se porneşte de la un număr întreg pozitiv(n) şi se ajunge la 1 pe două căi:

- ✓ dacă numărul este par, noul număr se obţine ca rezultat al împărţirii cu 2. (*exemplu:* 8, 4, 2, 1).
- ✓ dacă numărul este impar, noul număr se obţine cu formula: 3*n+1 (*exemplu:* 5, 16, 8, 4, 2, 1).

16. Simulaţi funcţionarea următoarelor script-uri (vezi figura 8.104 şi figura 8.105).

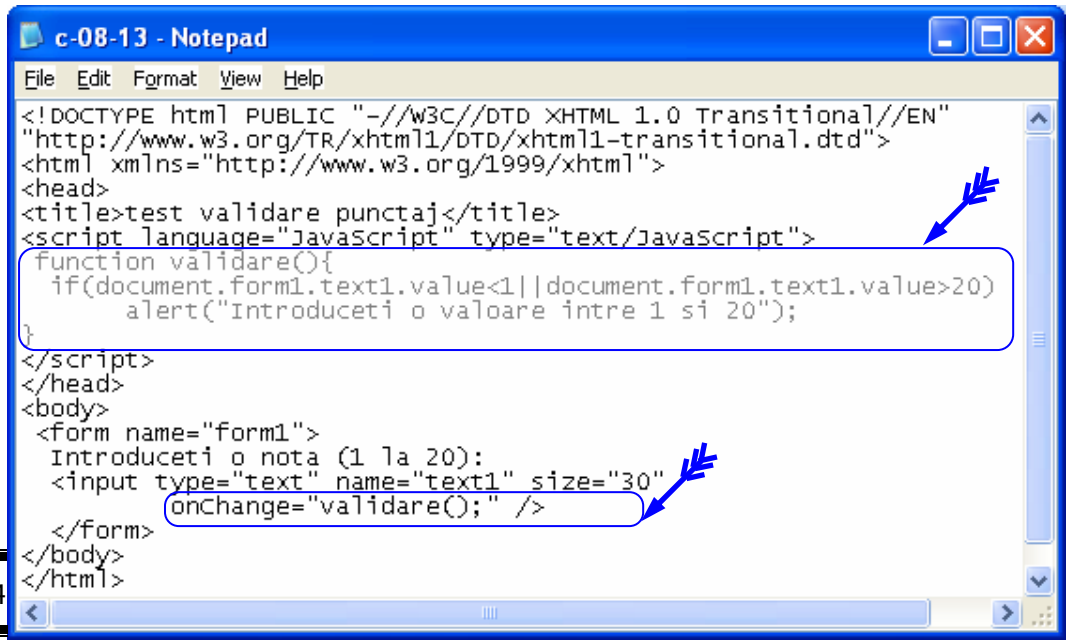


Figura 8.104

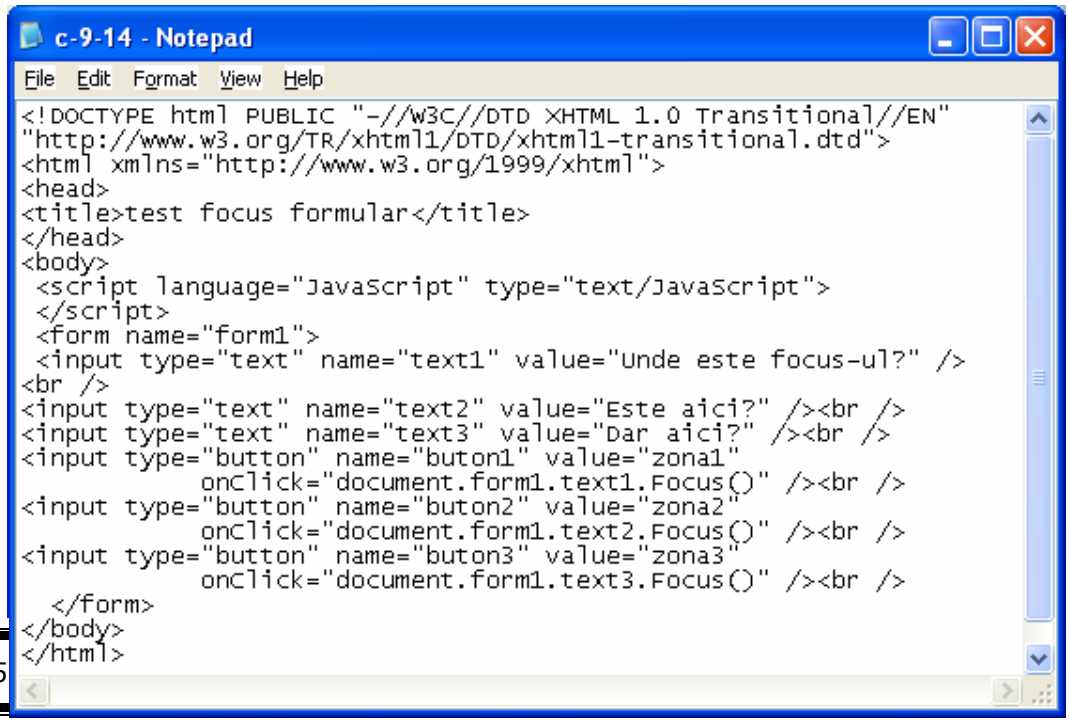


Figura 8.105

17. Înlocuiţi liniile de cod (vezi figura 8.104).

onClick="document.form1.text1.Focus()"

onClick="document.form1.text2.Focus()"

onClick="document.form1.text3.Focus()"

prin:

onClick="document.form1.text1.blur()"

onClick="document.form1.text2.blur()"

onClick="document.form1.text2.blur()"

Precizaţi rezultatele execuţiei noului script.

Vizitaţi site-urile

- ✓ http://www.javascriptgate.com/
- ✓ http://www.gatescript.com/
- ✓ http://www.biblioscript.com/
- ✓ http://www.w3schools.com/
- ✓ http://www.siteexperts.com/
- ✓ http://developer.irt.org/script/script.htm
- ✓ http://www.wsabsract.com/cutpastejava.shtml

Conversația 9

Obiectul Image

În această conversație:

Obiectul Image

Ca urmare a popularităţii lor, imaginile au pătruns în aproape toate domeniile navigării pe Web. De exemplu, în numeroase site-uri Web, butoanele (X)HTML au fost înlocuite prin imagini clicabile (rollovere) care se transformă atunci când treceţi mouse-ul pe deasupra acestora. Cu siguranţă că apreciaţi aceste interesante efecte grafice dar ignoraţi poate faptul că puterea acestora se află în limbajul JavaScript.

În consecinţă, crearea unui site Web cu adevărat puternic presupune cunoaşterea facilităţilor imaginilor (X)HTML şi a obiectului `Image` al limbajului JavaScript.

Fişa obiectului `Image` este prezentată în figura 9.1.

Fişa obiectului Image	
Obiectul părinte	`Document`
Cum se creează obiectul?	`Constructorul Image()`
Proprietăţi:	`border, complete, height, width, hspace, vspace, name, src, lowsrc`
Metode:	Obiectul `Image` nu posedă nici o metodă dar puteţi modifica proprietăţile imaginilor în mod dinamic.
Gestionarii de evenimente:	`onAbort, onError, onKeyDown, onKeyPress, onKeyUp, onLoad, onClick, onDblClick, onMouseDown, onMouseOut, onMouseOver` (vezi Conversaţia 6).

Figura 9.1

Relaţia dintre obiectul Image şi tag-ul <img>

Manipularea imaginilor din cadrul unui document (X)HTML depinde în mare măsură de obiectul `Image` al limbajului JavaScript. În cursul încărcării documentului (X)HTML în navigatorul Web, interpretorul JavaScript creează un obiect `Image` pentru fiecare tag `<img>` prezent în document. El plasează toate obiectele **Image** într-o matrice `images` (subobiect al obiectului `Document`).

În loc să utilizaţi matricea `images`, puteţi specifica pentru fiecare tag `<img>` un atribut `"name="`, care va da un acces direct obiectelor `Image`.

Iată cum procedăm pentru a accesa obiectul `Image` asociat tag-ului `<img>` din figura 9.2, care reprezintă prima imagine a unui document (X)HTML.

```
...
<img src="sigla.jpg" name="LUMINABLANDA" />
...
```

Figura 9.2

Obiectul `Image` poate fi accesat cu una din instrucţiunile prezentate mai jos:

- ✓ `document.images[0];`
- ✓ document.LUMINABLANDA.

Remarcă. Dacă dintr-un motiv sau altul nu aţi putut da imaginilor dumneavoastră un *nume valid* de variabilă JavaScript, atunci accesul prin nume rămâne posibil numai prin intermediul matricei `images`, ca şi în cazul matricei `elements` din obiectul `Form`. De exemplu, definiţia imaginii: `<img name="PLOIESTI VEST" src="gara.jpg" />` este accesibilă prin utilizarea matricei `images`, după cum urmează: `var gara=document.images['PLOIESTI VEST'].`

Constructorul Image()

Interpretorul JavaScript creează un obiect `Image` pentru fiecare tag `<img>` pe care îl întâlneşte într-un document (X)HTML, dar … atenţie, chiar şi dumneavoastră îl puteţi crea fără asocierea unui tag `<img>`, utilizând constructorul de obiecte `Image()` (vezi figura 9.3).

Figura 9.3

```
…
var imagine1=new Image();
```

Remarci

- ✓ Instrucţiunea JavaScript din figura 9.3 generează un obiect `Image` fără a mai fi nevoie de un tag `<img>`.
- ✓ Puteţi încărca un fişier imagine în obiectul `Image` atribuind în mod simplu o adresă Web proprietăţii `src` (figura 9.4).

Figura 9.4

```
…
imagine1.src="sigla.jpg"
…
```

- ✓ După încărcarea acestei imagini, s-a creat un obiect `Image` care conţine o imagine (încărcată), dar *non* vizibilă în navigatorul Web.
- ✓ Pentru ca o imagine să se afişeze în navigatorul Web, ea trebuie să fie asociată în mod obligatoriu unui tag `<img>` aflat în documentul (X)HTML; în caz contrar, navigatorul Web ignoră … totul!

Iată cum folosim imagini fără a utiliza tag-ul `<img>` (vezi figura 9.5).

Figura 9.5

```
<img src="moara.jpg" name="MOULINROUGE" />
document.MOULINROUGE.src=MOARACUNOROC.src;
…
```

Imaginea conţinută în `document.MOULINROUGE` va fi înlocuită prin imaginea conţinută în MOARACUNOROC. În plus, întrucât `document.MOULINROUGE` este asociat unui tag `<img>` noua imagine va fi afişată în locul precedentei imagini.

Proprietăţile obiectului Image

Proprietăţile obiectului `Image` sunt prezentate în detaliu în figura 9.6.

Figura 9.6

	Proprietate	***Sintaxa***
	`border`	`document.images[].border`
	Corespunde atributului `border` al tag-ului `<img>`.	
	`complete`	`document.images[].complete`
	Valoare logică (`true/false`); `true` - imaginea a fost în întregime încărcată; `false` – imaginea nu a fost încărcată în întregime.	
	`height`	`document.images[].height`
	Corespunde atributului `height` al tag-ului `<img>`.	

`width`	`document.images[].width`
Corespunde atributului `width` al tag-ului `<img>`.	
`hspace`	`document.images[].hspace`
Corespunde atributului `hspace` al tag-ului `<img>`.	
`vspace`	`document.images[].vspace`
Corespunde atributului `vspace` al tag-ului `<img>`.	
`name`	`document.images[].name`
Corespunde atributului `name` al tag-ului `<img>`.	
`src`	`document.images[].src`
Corespunde atributului `src` al tag-ului `<img>`.	
`lowsrc`	`document.images[].lowsrc`
Corespunde atributului `lowsrc` al tag-ului `<img>`.	

Figura 9.6 *(continuare)*

Aplicaţii

❑ Precizaţi rezultatul execuţiei următorului program JavaScript (figura 9.7).

c-09-01 - Notepad

File Edit Format View Help

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Obiectul IMAGE</title>
</head>

<body>
 <h3>Obiectul IMAGE</h3>
 <p>
  <a href="http://validator.w3.org/check/referer">
   <img border="0" src="valid-html401.png"
    alt="Valid HTML 4.01!" height="31" width="88">
  </a>
 </p>
 <script language="javascript" type="text/javascript">
   document.write("Textul alternativa pentru aceasta imagine este:"
                  +document.images[0].alt+"<br>");
   document.write("Lungimea pentru aceasta imagine este: "
                  +document.images[0].width+"<br>");
   document.write("Latimea pentru aceasta imagine este: "
                  +document.images[0].height+"<br>");
 </script>
</body>
</html>
```

Figura 9.7

În figura 9.8 este prezentat rezultatul execuţiei programului JavaScript, pe care vă rugăm să-l comentaţi.

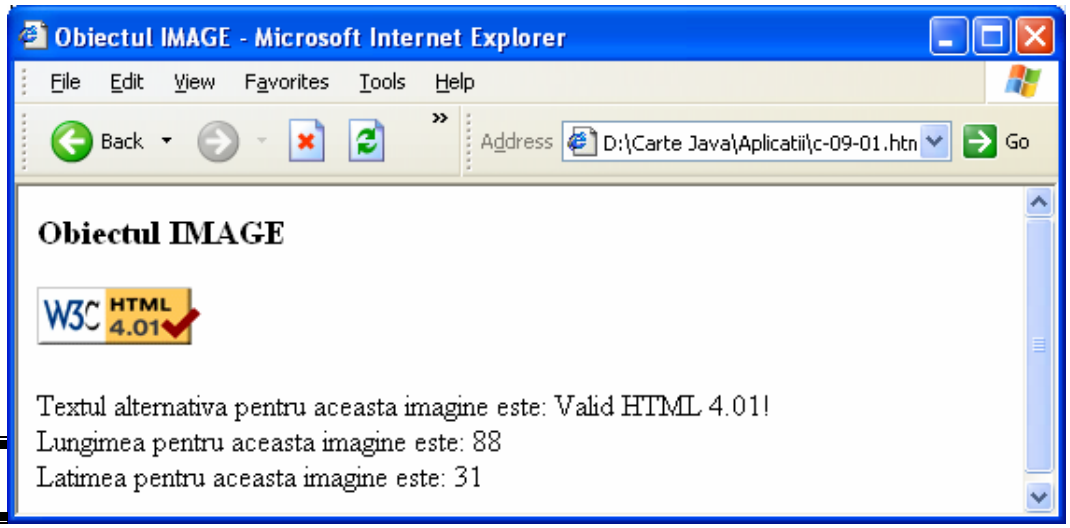


Figura 9.8

❑ Comentaţi următorul program JavaScript (figura 9.9).

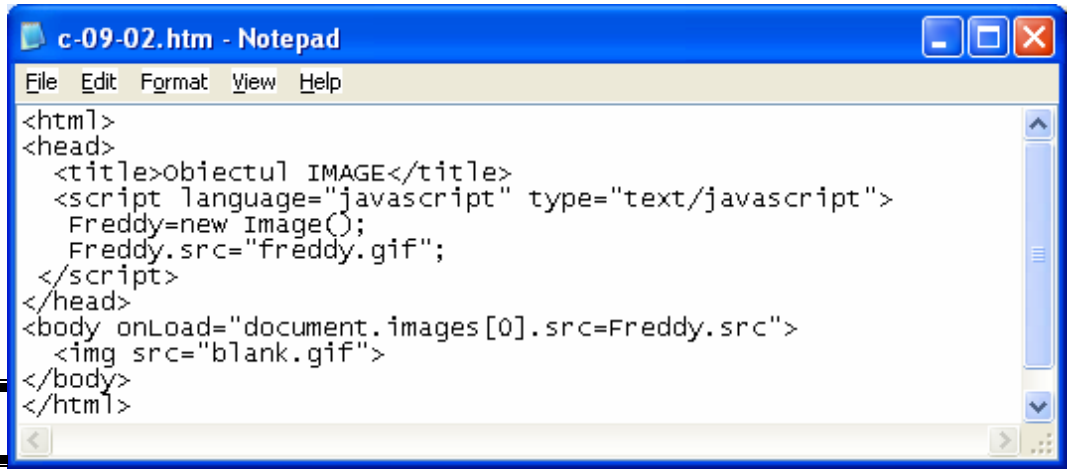


Figura 9.9

❑ Scrieţi un program JavaScript care afişează proprietăţile unui obiect `Image`.

În figura 9.10 este prezentat documentul (X)HTML complet.

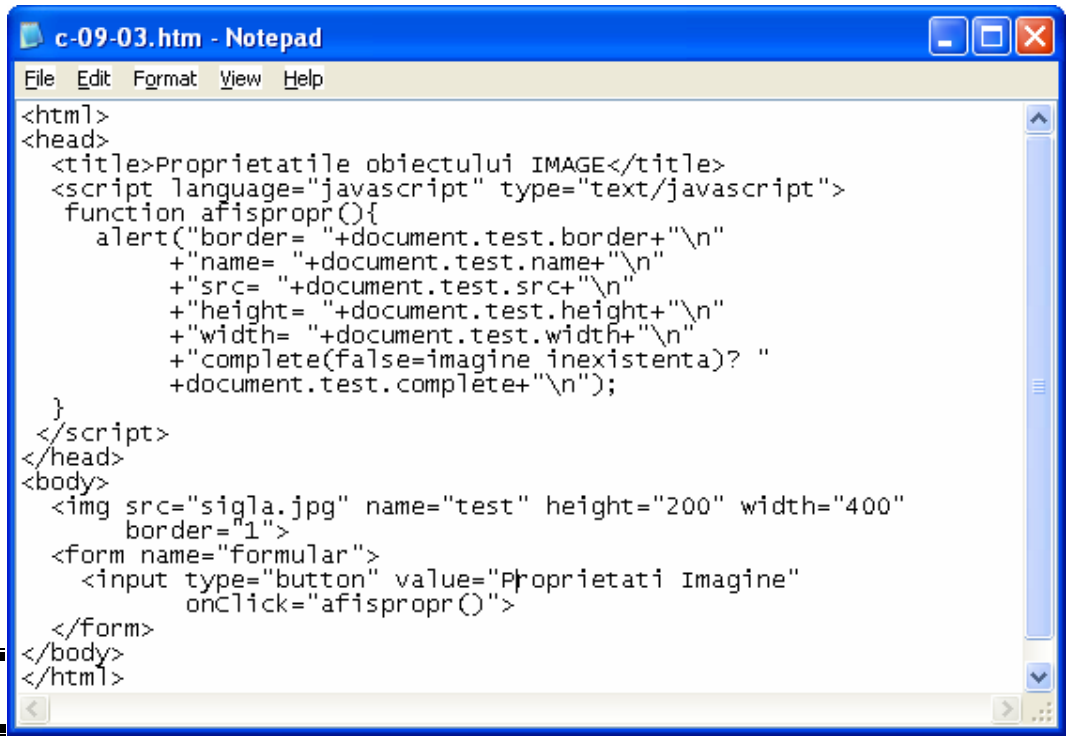


Figura 9.10

Comentarii:

- ✓ Obiectul `Document` (tot ceea ce se găseşte între tag-urile <body> şi </body>) conţine imaginea cu numele `sigla.jpg`. La rândul său, această imagine conţine proprietăţile: `src`, `name`, `height`, `width`, `border` şi `complete`. Această ierarhie explică modul în care sunt accesate proprietăţile imaginii: `document.test.src`, `document.test.width` etc.
- ✓ Valorile proprietăţilor (`src`, `name`, `height`, `width`, `border`) sunt extrase din definiţia acestui obiect.
- ✓ Valoarea `true` (vezi proprietatea `complete`) rezultă din modul în care imaginea a putut fi corect încărcată în navigator. În caz contrar valoarea va fi `false`.

Creați un joc cu JavaScript

❑ De la joc la program

Vom părăsi pentru scurt timp problema rezervoarelor cu care v-am pisat în conversațiile anterioare, invitându-vă în cele ce urmează să … vă jucați puțin! Da, ați înțeles bine! Ne vom juca împreună! Despre ce este vorba? Vom proiecta şi realiza un program JavaScript pentru jocul: `ZECE BE•E DE CHIBRIT`!

Pentru aceia dintre dumneavoastră care nu cunosc acest joc deoarece până în prezent au avut doar preocupări … serioase, precizăm în cele ce urmează care sunt regulile jocului.

Există la început 10 bețe de chibrit şi doi jucători. Când îi vine rândul, un jucător poate ridica un bâț, cel mult două! Va pierde (!) acel jucător care rămâne cu ultimul bâț de chibrit.

Ceea ce se cere este să realizați un program JavaScript pentru jocul „ZECE BE•E DE CHIBRIT" avându-i ca jucători pe dumneavoastră şi Măria sa … calculatorul!

❑ Specificații de programare

În figurile 9.11 şi 9.12 sunt prezentate: ecranul cu 10 bețe de chibrit, specificațiile de programare. Tabela de variabile, mesajele generate în timpul jocului sunt prezentate în figurile 9.13, respectiv 9.14.

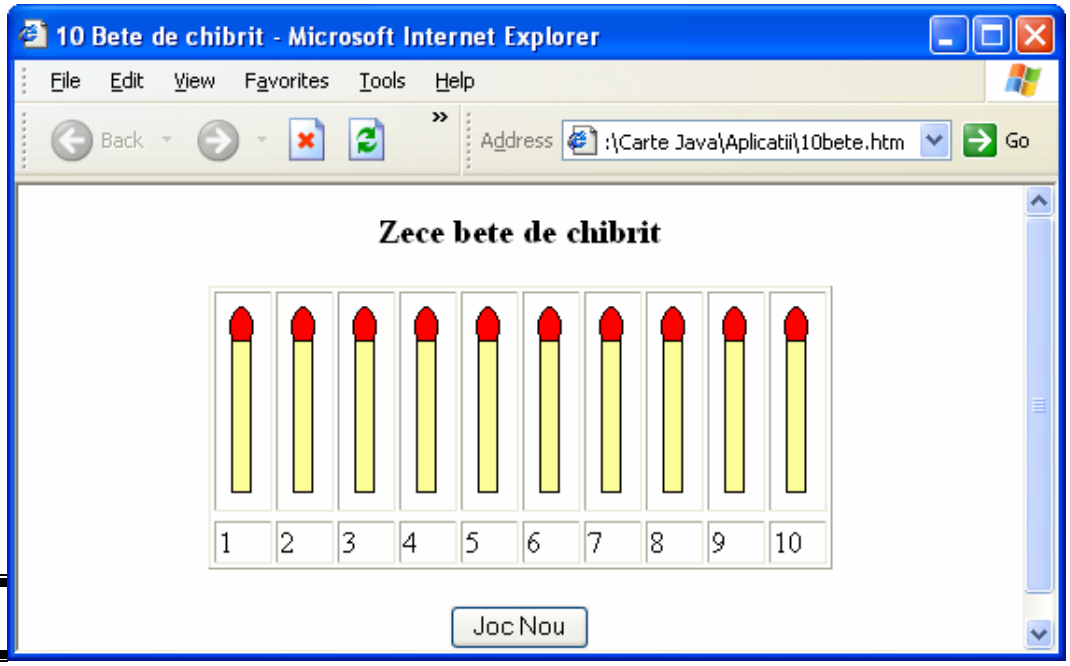


Figura 9.11

Specificații de programare
Descriere. Programul afişează zece bețe de chibrit (1-10) şi un buton *Joc Nou* care permite reluarea jocului. **Lista de funcții ale programului** **1.** Permite jucătorului să extragă unu/două bețe. **2.** Stabileşte strategia de joc a calculatorului. **3.** Actualizează numărul de bețe rămase. **4.** Stabileşte câştigătorul (în final). **5.** Permite reluarea jocului. **6.** Stop.

Figura 9.12

Tabela de variabile

Variabile de intrare	*Variabile de stare*	*Variabile de ieşire*
	i: variabila de control a buclei `for` x: numărul bățului extras (1-10) bete: numărul de bețe (rămase) nsrc: rangul obiectului `Image` în matricea images chibrit1-chibrit10: numele imaginilor cu cele 10 chibrite f1: numele formularului Buton: numele butonului Joc Nou	

Figura 9.13

Mesaje

Mesaj	*Descriere*
Jocul s-a terminat! Ai câştigat!	Desemnează câştigătorul
Jocul s-a terminat! Ai pierdut!	Desemnează învinsul.
Calculatorul intervine: Acum e rândul meu!	Este rândul calculatorului să joace.
Calculatorul intervine: Acum e rândul tău!	Este rândul jucătorului.

Figura 9.14

❑ Documentația de proiectare

Pseudocodul este prezentat în figura 9.15.

Pseudocodul

```
10Betechibrit      BEGIN
                        //formeaza pagina html cu imaginile chibritului
                        bete=10
                        //Trateaza evenimentele generate
IF1                     IF(se apasa butonul Joc Nou)
                              DO actualizare
IF1                     ENDIF
IF2                     IF(se executa click pe imagine chibrit)
                              DO alegeChibrit(numar_bat_chibrit)
IF2                     ENDIF
10betechibrit      END

ACTUALIZARE        BEGIN
FOR1                    bete=10
                        FOR(i=1;i<=bete;i++)
                              nsrc='chibrit'+i
                              imagine_nsrc.src="chibrit.gif"
FOR1                    ENDFOR
ACTUALIZARE        END

AlegeChibrit       BEGIN
                        Date intrare: x-numarul batului de chibrit extras
                        //se verifica daca extragerea batului a fost
                        //din pozitia corecta
```

Figura 9.15

```
IF3             IF(x<=bete) AND (bete-x)<2
                        //se sterge imaginea betelor de chibrit extrase
FOR2                    FOR(i=x;i<=bete;i++)
                                inlocuieste imaginea chibrit.jpg prin gol.jpg
FOR2                    ENDFOR
                        //se actualizeaza numarul de bete ramase
                        bete=x-1;
                        //verifica numarul de bete ramase si
                        //stabileste castigatorul daca este cazul
IF4                     IF(bete<=0)
                                WRITE "Jocul s-a terminat! Ai pierdut!"
IF4                     ELSE
IF5                             IF(bete==1)
                                        //inlocuie imaginea primului bat de chibrit
                                        chibrit1.src=gol.gif
                                        bete=0
                                        WRITE "Jocul s-a terminat!Ai Castigat!"
IF5                             ELSE
                                        WRITE "Calculatorul zice:\n Acum e randul meu");
                                        //inteligenta calculatorului este aleatoare
                                        rand=genereaza_numar_aleator
IF6                                     IF(rand<0.5)
                                                x=bete
IF6                                     ELSE
                                                x=bete-1
IF6                                     ENDIF
FOR3                                    FOR(i=x;i<=bete;i++)
                                                nsrc='chibrit'+i
                                                imaginea_nsrc.src="gol.gif"
FOR3                                    ENDFOR
                                        bete=x-1
 IF7                                    IF(bete==0)
                                                WRITE "Jocul s-a terminat! Ai pierdut!"
 IF7                                    ELSE
                                                WRITE " Acum e randul tau!"
IF7                                     ENDIF
IF5                             ENDIF
IF4                     ENDIF
IF3             ENDIF
Alegechibrit END
```

Figura 9.15 *(continuare)*

❑ Codificarea în limbajul JavaScript

Documentul complet (X)HTML este prezentat în figura 9.16.

```
<html>
<head>
<title>10 Bete de chibrit</title>
<script language="JavaScript">
<!---
function AlegeChibrit(x) {
 //se verifica daca extragerea batului a fost din pozitia corecta
 if((x<=bete)&&((bete-x)<2)) {
 //se sterg betele extrase
for(i=x;i<=bete;i++){
      nsrc='chibrit'+i;
      document.images[nsrc].src="gol.gif";
    }
```

Figura 9.16

```
 //se actualizeaza numarul de bete ramase
  bete=x-1;
//verifica numarul de bete ramase si
//stabileste castigatorul daca este cazul
  if(bete<=0)  {
   alert("Jocul s-a terminat! Ai pierdut!");
  }else
    if(bete==1) {
     document.images['chibrit1'].src="gol.gif";
      bete=0;
      alert("Jocul s-a terminat!Ai Castigat!");
    }
   else{
  alert("Calculatorul zice:\n Acum e randul meu");
//inteligenta calculatorului este aleatoare
   rand=Math.random();
   if(rand<0.5)x=bete;
    else
      x=bete-1;
   for(i=x;i<=bete;i++){
      nsrc='chibrit'+i;
     document.images[nsrc].src="gol.gif";
     }
     bete=x-1;
    if(bete==0)
   alert("Jocul s-a terminat! Ai pierdut!");
   else
   alert("Calculatorul zice:\n Acum e randul tau!");
  }
 }
}
bete=10;
function actualizare()
{bete=10;
  for(i=1;i<=bete;i++){
      nsrc='chibrit'+i;
     document.images[nsrc].src="chibrit.gif";
     }
}
// -->
</script>
</head>
<body>
<center>
<h3>Zece bete de chibrit</h3>
<table border=1>
<tr><td>
<img src="chibrit.gif" name="chibrit1" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(1)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit2" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(2)" loop="0">
```

Figura 9.16 *(continuare)*

```
<td>
<img src="chibrit.gif" name="chibrit3" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(3)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit4" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(4)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit5" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(5)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit6" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(6)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit7" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(7)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit8" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(8)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit9" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(9)" loop="0">
<td>
<img src="chibrit.gif" name="chibrit10" border="0" width="27" height="110"
onMouseDown="AlegeChibrit(10)" loop="0">
</td><tr>
<tr><td>1<td>2<td>3<td>4<td>5<td>6<td>7<td>8<td>9<td>10</td></tr>
</table>
<form name="f1">
<p><p><input type="button" value="Joc Nou" name="Buton" onClick="actualizare();"></p>
</form>
</body>
</html>
```

Figura 9.16 *(continuare)*

Remarci

- ✓ Script-ul (inserat în secțiunea `<head>` a documentului HTML) conține două funcții: `AlegeChibrit(x)` şi `actualizare()`.
- ✓ Funcția `AlegeChibrit(x)` permite jucătorului să extragă unul sau două bețe; stabileşte strategia de joc a calculatorului; actualizează numărul de bețe rămase, stabileşte câştigătorul. Argumentul x ia valori de la 1 la 10.
- ✓ Funcția `actualizare()` permite reluarea jocului la acționarea butonului *Joc Nou*.
- ✓ Gestionarii de evenimente utilizați sunt: `onMouseDown` şi `onClick` (tag-ul `<input />`).

În figura 9.17, 9.18, 9.19 sunt prezentate secvențe din timpul jocului.

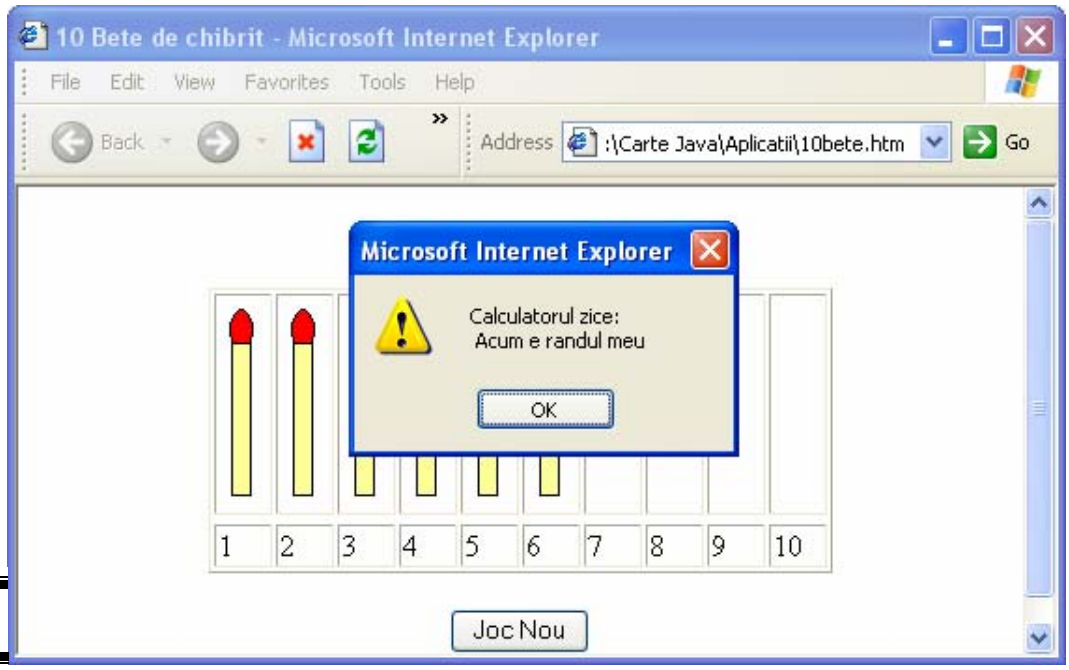


Figura 9.17

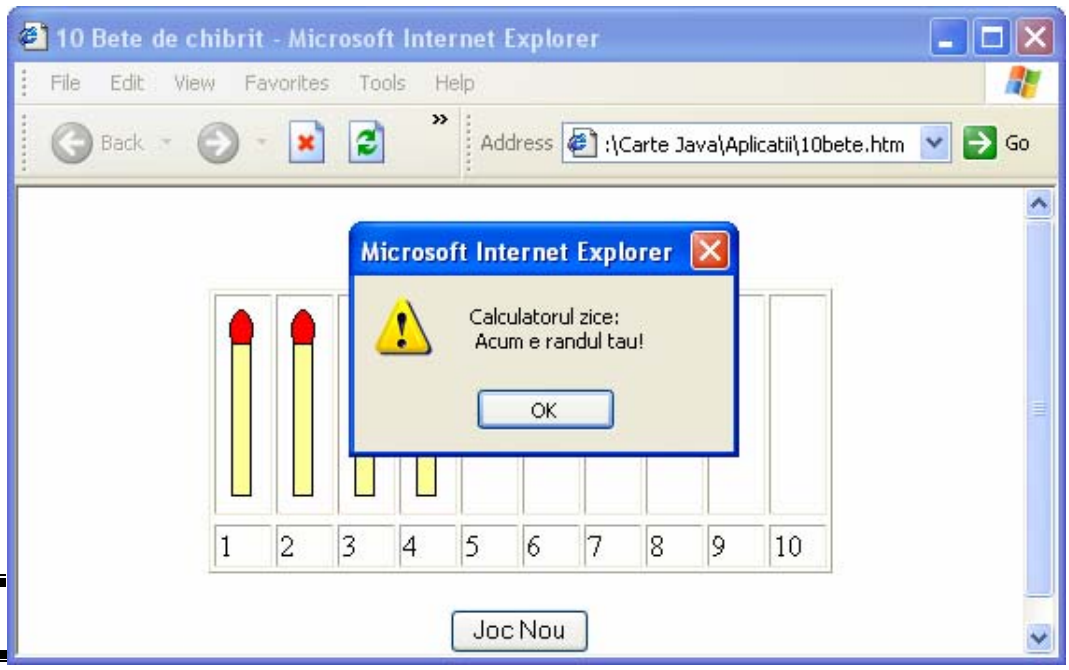


Figura 9.18

Figura 9.19

Creați un rollover cu JavaScript

Una din aplicaţiile JavaScript de mare succes în domeniul imaginilor o constituie crearea de `rollovers` (*`survols`*, în limba franceză; *`rollovers`*, în limba engleză).

Un `rollover` este o imagine – folosită în general pentru navigare – al cărui aspect se modifică atunci când se trece cu mouse-ul pe deasupra acesteia.

Remarci

- ✓ Un `rollover` este o metodă asociată unui eveniment de mouse: `mouseOver`, `mouseOut`, `mouseDown`, `mouseUp`.
- ✓ Există mai multe tipuri de metode `rollover`:
  - metoda `rollover` pentru imagine;
  - metoda `rollover strat` (layer).
- ✓ Va trebui să decideţi singuri: ce tip de metodă `rollover` necesită aplicaţia dumneavoastră şi cum doriţi s-o implementaţi.

Iată cum creăm un buton `IMAG1` care se afişează normal la încărcarea paginii şi se schimbă în `IMAG2` la trecerea mouse-ului pe deasupra acestuia.

1. Introduceţi între tag-urile `<head>` şi `</head>` ale documentului HTML, script-ul de preîncărcare al imaginii (`imag2.gif`) utilizată în `rollover` (figura 9.20).

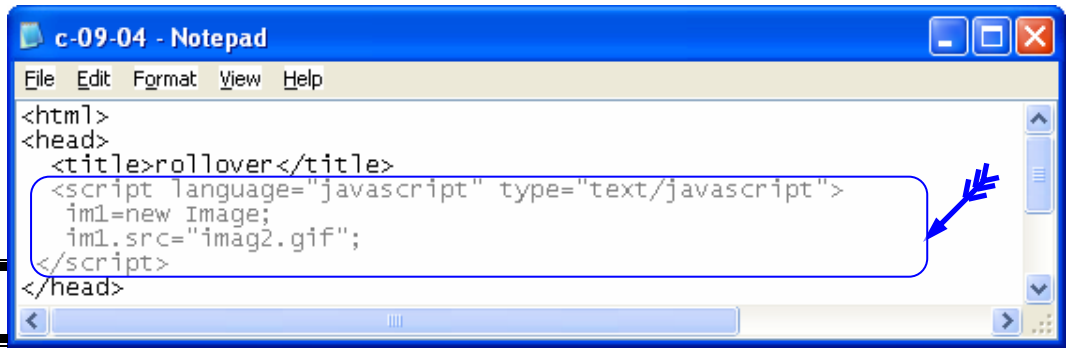


Figura 9.20

2. Stabiliţi o legătură (link) către o imagine (figura 9.21).

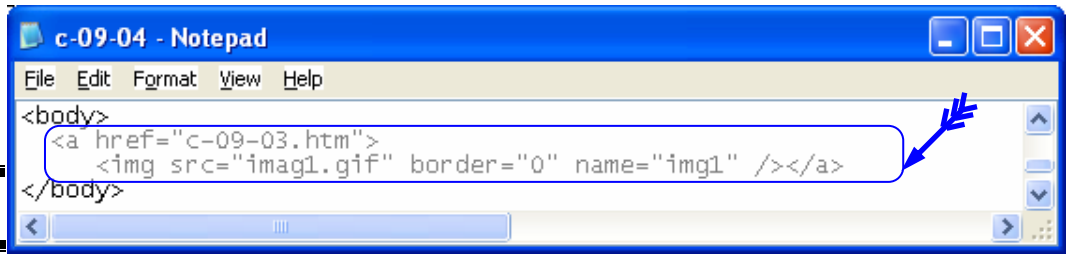


Figura 9.21

3. Introduceţi (în tag-ul `<a>`) gestionarii de evenimente `onMouseOver` şi `onMouseOut` (figura 9.22).

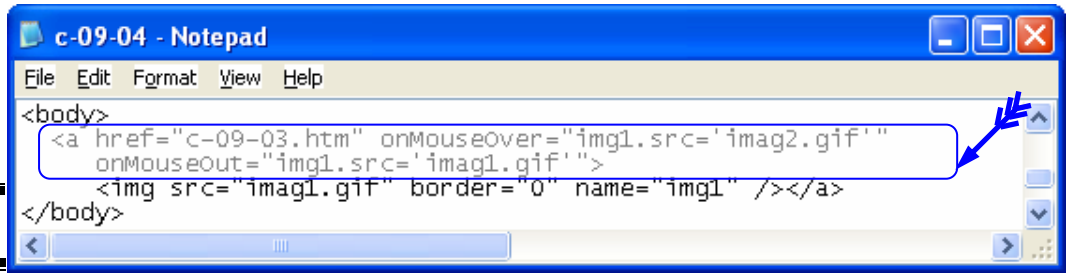


Figura 9.22

4. Testaţi script-ul (figura 9.23, figura 9.24).

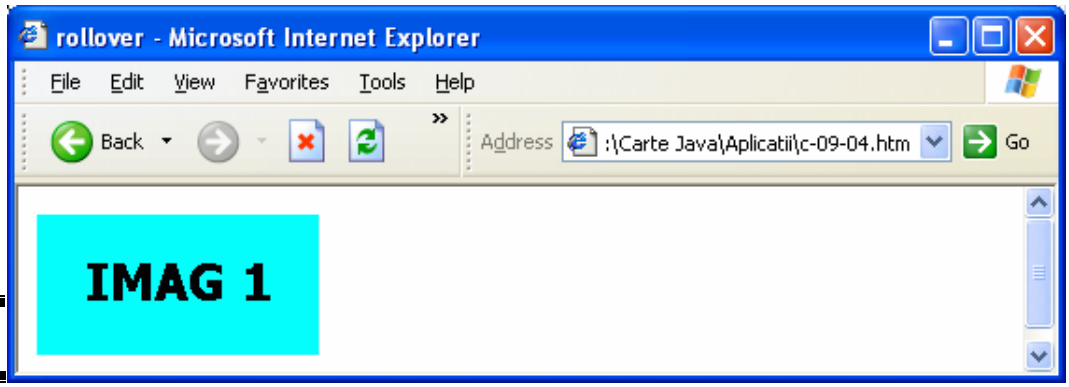


Figura 9.23

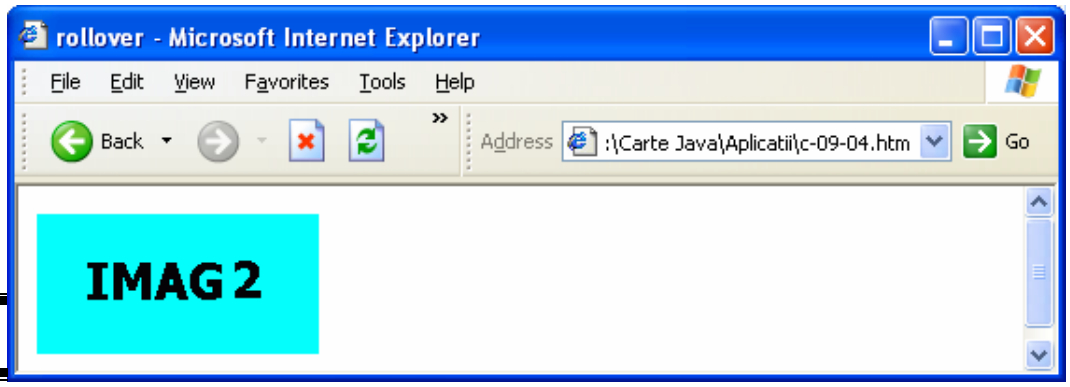


Figura 9.24

Remarci

- ✓ Nu vă lăsaţi impresionaţi de dimensiunea codului JavaScript, deoarece este foarte simplu!
- ✓ Am definit *im1* ca fiind un nou obiect `Image` căruia îi atribuim calea `imag2.gif`.
- ✓ La trecerea mouse-ului peste butonul `IMAG1`, proprietatea *src* a imaginii ce poartă numele *`img1`* devine `imag2.gif`, care provoacă efectul de `rollover`.
- ✓ Dacă doriţi să aplicaţi un `rollover` peste mai multe imagini prezentate într-o pagină, va trebui:
  - Să adăugaţi noi linii de preîncărcare
    ```
    im2=new Image;
    im2.src="imag3.gif";
    ```
  - Să schimbaţi numele imaginii respective:
    ```
    <img src="imag1.gif" border="0" name="img2">.
    ```

Aplicaţii

❑ Creaţi acelaşi `rollover` utilizând matricea `images`.

În figura 9.25 este prezentat documentul HTML complet.

Figura 9.25

Rezultatele execuţiei sunt prezentate în figura 9.26, respectiv figura 9.27.

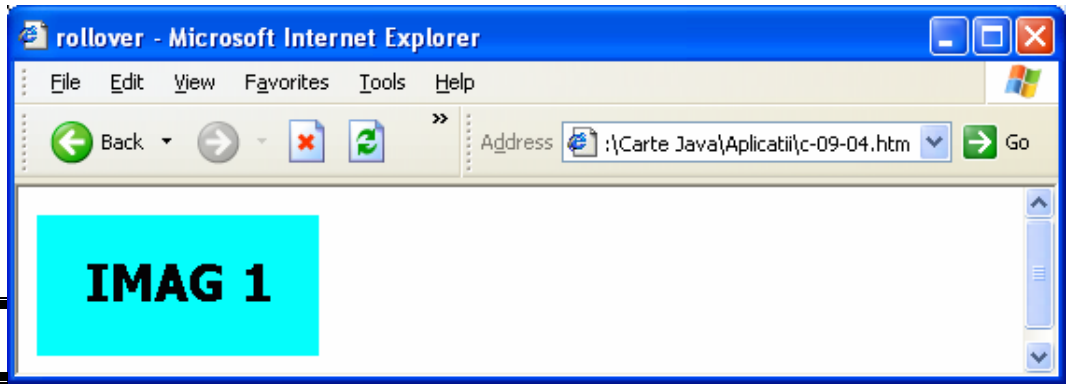


Figura 9.26

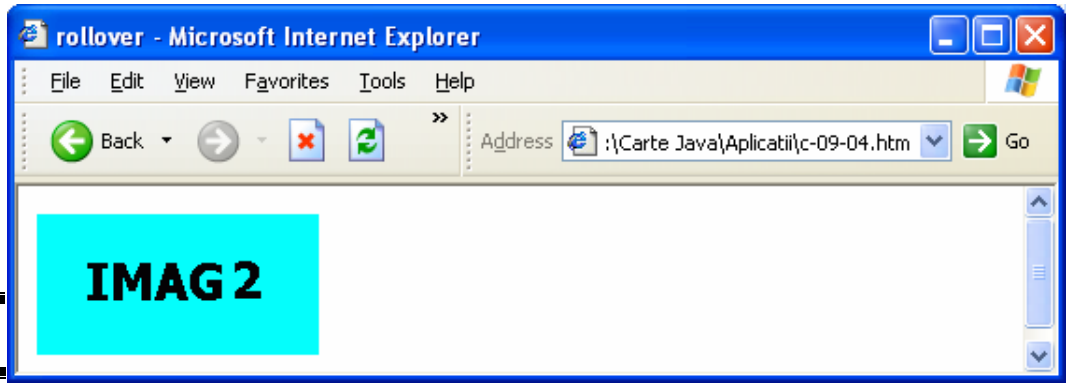


Figura 9.27

Remarci

- ✓ Există şi alte tehnici de a modifica imaginile, după cum urmează:
  - afişarea unei borduri în jurul fiecărei imagini;
  - afişarea unei imagini în miniatură (de exemplu, o săgeată mică) alături de imaginea propriu-zisă, unde se găseşte pointer-ul mouse-ului.
- ✓ Există mai multe soluţii de implementare a metodelor `rollover` pentru imagini, dar cea mai simplă este următoarea:
  - creaţi matricea care să conţină imagini;
  - preîncărcaţi imaginile în matrice;
  - creaţi o funcţie care să realizeze schimbarea imaginilor;
  - introduceţi gestionarii de evenimente `onMouseOver` şi `onMouseOut`.

❑ Scrieţi o funcţie pentru a verifica dacă o imagine este complet încărcată.

În figura 9.28 este prezentat script-ul aplicaţiei.

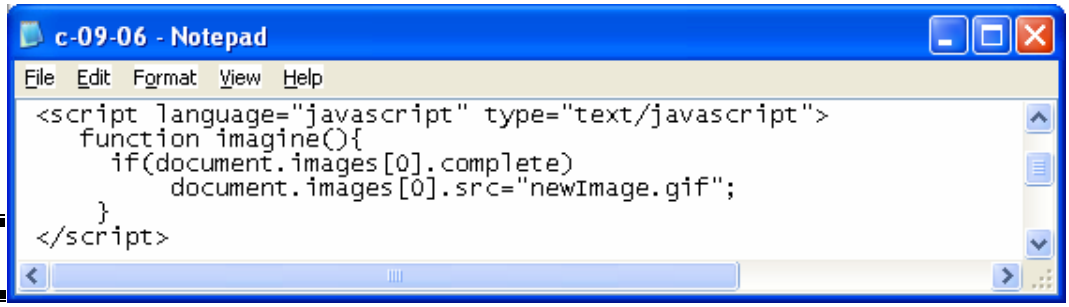


Figura 9.28

❑ Comentaţi următorul program JavaScript (figura 9.29).

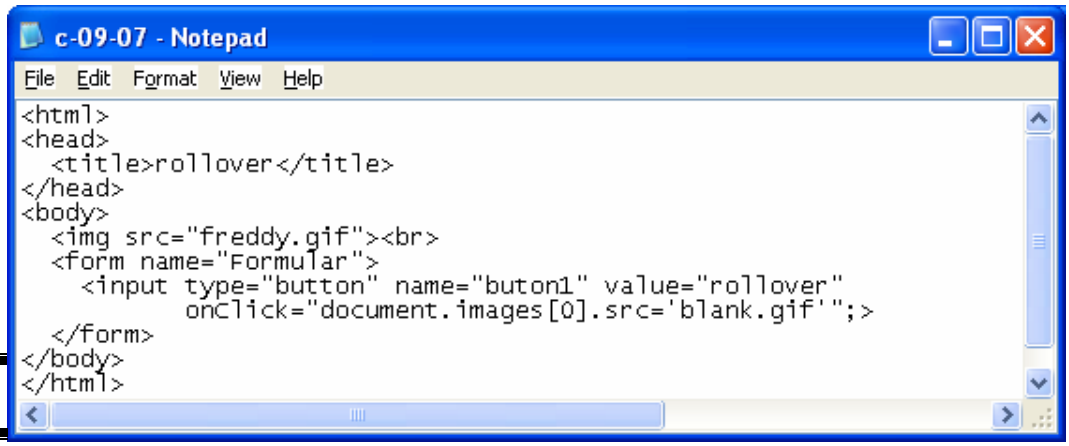


Figura 9.29

În figura 9.30, respectiv figura 9.31 sunt prezentate rezultatele execuţiei programului JavaScript.

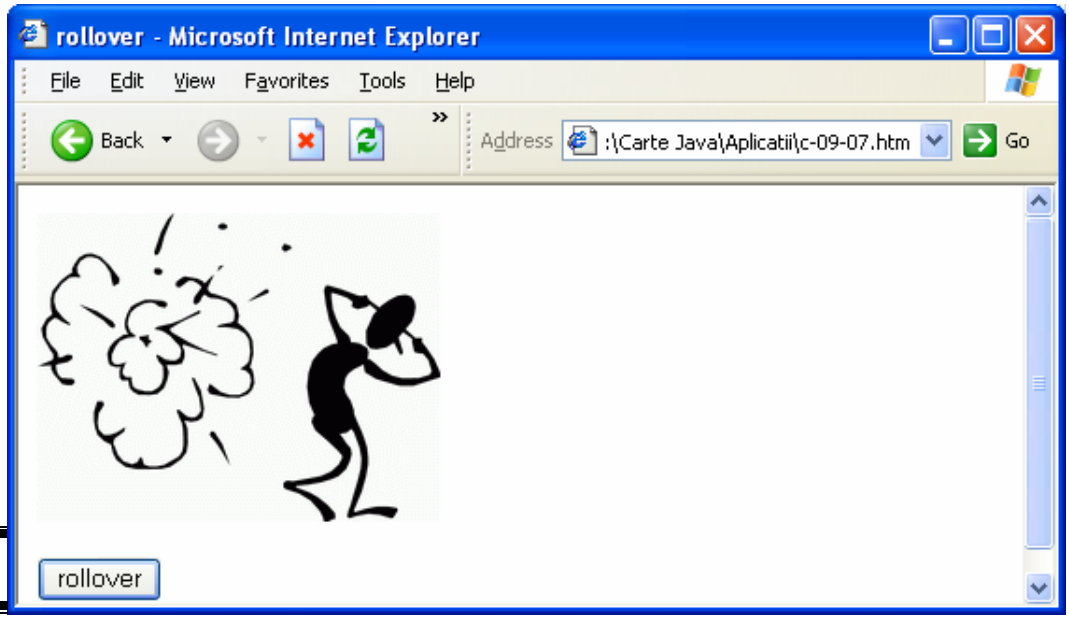


Figura 9.30

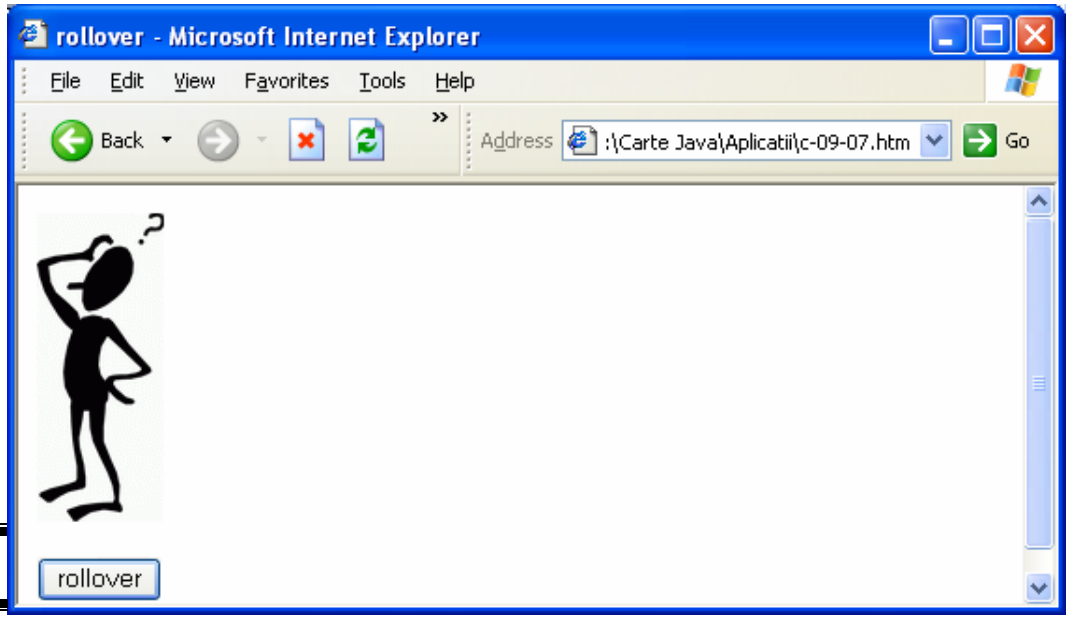


Figura 9.31

Iată cum creăm un mic catalog ilustrat care conţine ultimele trei cărţi ale autorului acestei lucrări: (X)HTML, Dreamweaver MX, XML, care au apărut în Editura Universităţii din Ploieşti. Pornind de la o listă ordonată ale cărei elemente conţin titlurile celor trei cărţi, se afişează coperta întâi a cărţii selectate trecând cursorul pe deasupra unuia din cele trei titluri de carte afişate (figura 9.32).

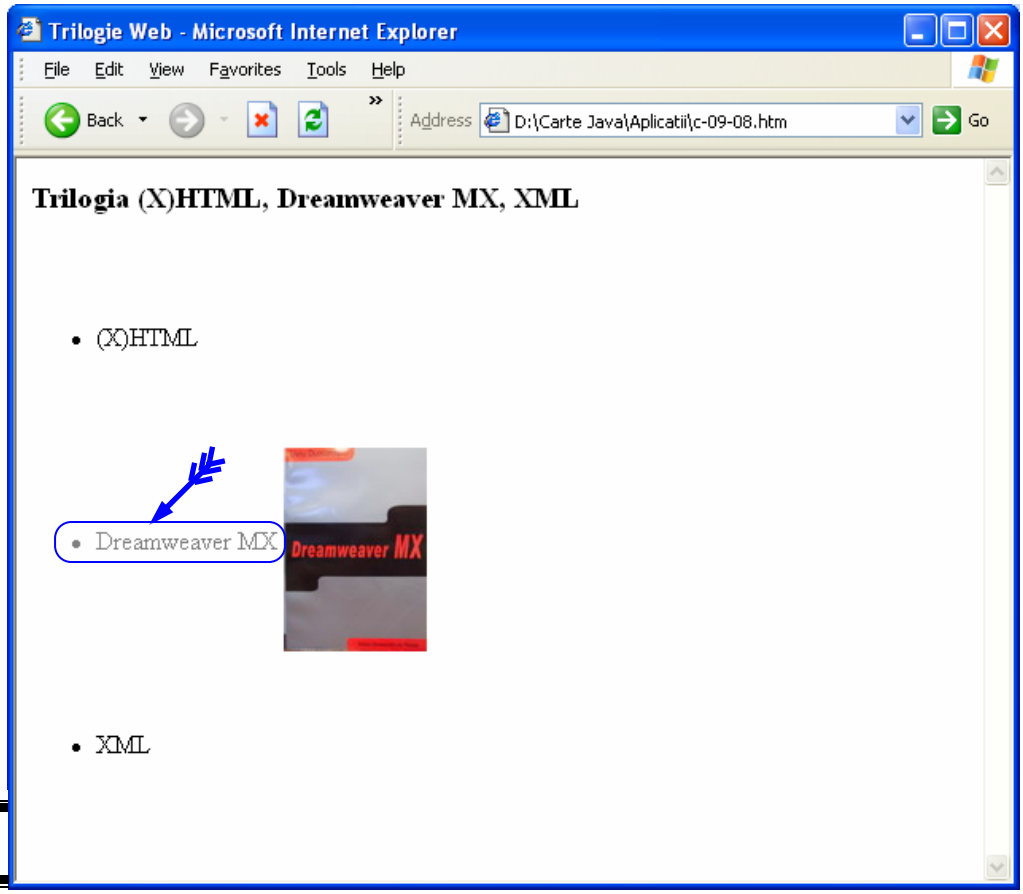


Figura 9.32

Documentul complet XHTML este prezentat în figura 9.33.

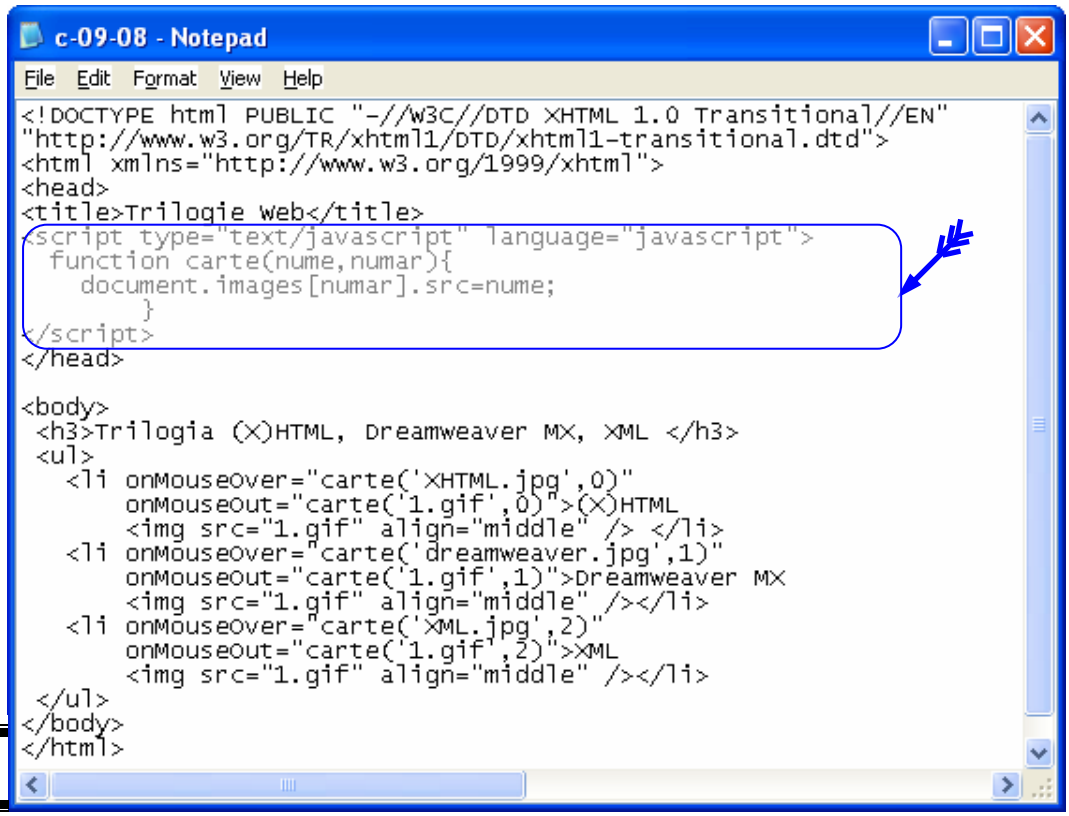


Figura 9.33

Remarci

- ✓ Funcţia `carte(nume,num•r)` conţine două argumente: numele fişierului utilizat (`nume`) şi numărul imaginii (`num•r`).
- ✓ Pentru ca proporţia imaginilor să fie respectată este necesar ca ele să aibă aceleaşi dimensiuni.
- ✓ Afişarea imaginii (prima copertă) se produce atunci când deplasaţi mouse-ul pe linia unde se găseşte numele cărţii şi nu doar pe numele afişat al cărţii. Puteţi evita acest lucru, modificând puţin codul JavaScript. Succes!

❑ În figura 9.34 se prezintă un document (X)HTML care ilustrează tehnica `rollovers`-urilor.

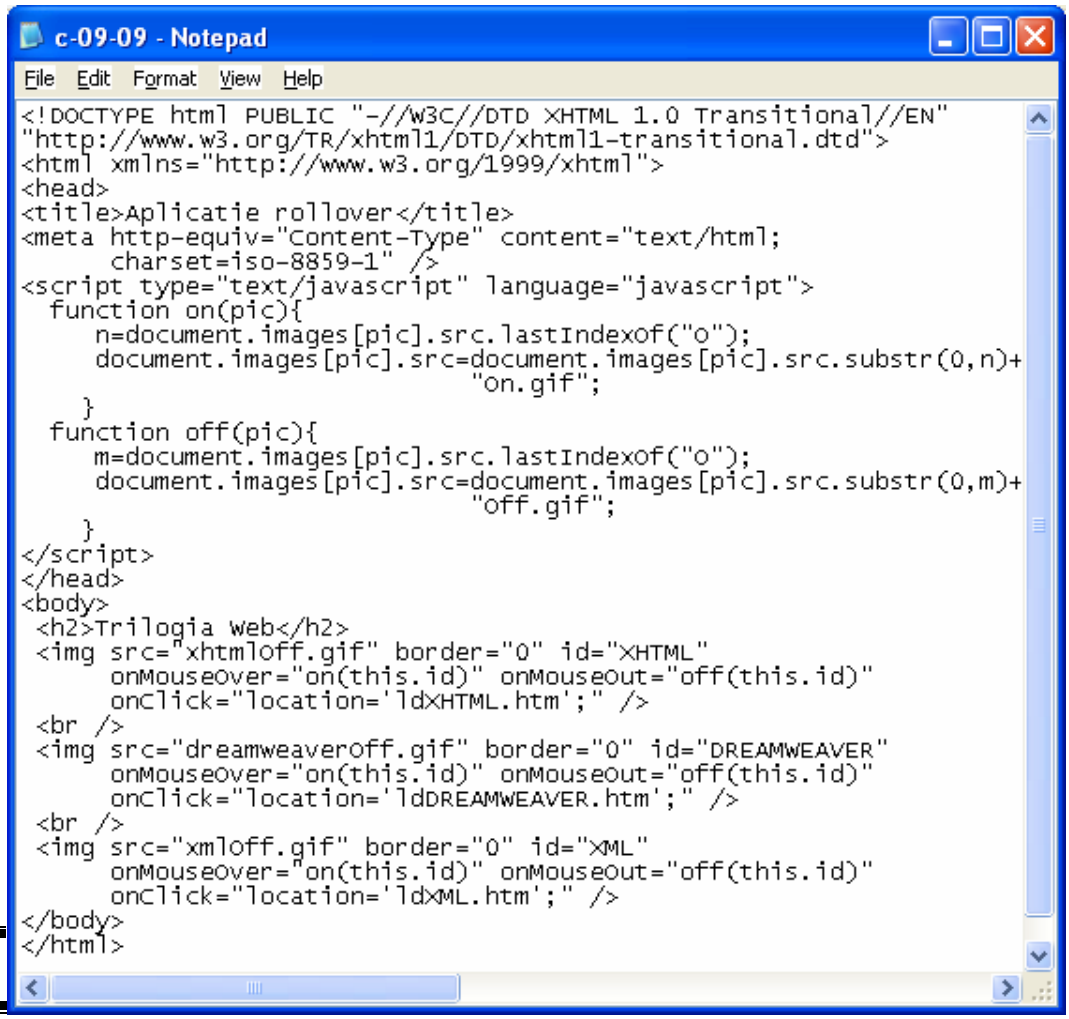


Figura 9.34

Atunci când se încarcă acest document se obţine imaginea din figura 9.35.

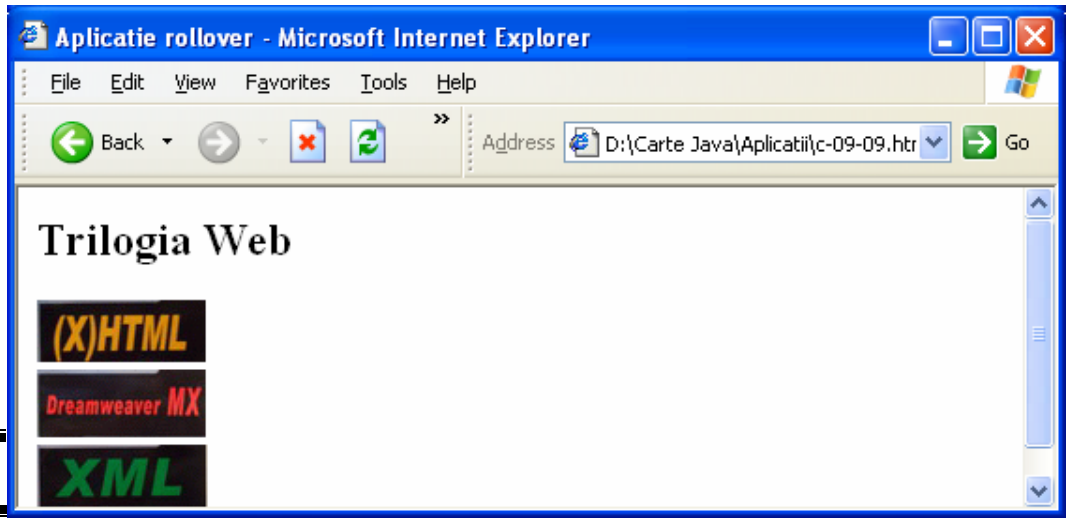


Figura 9.35

În momentul în care plimbaţi mouse-ul (fără a executa clic) pe una din cele trei imagini (figura 9.35) obţineţi ceea ce este ilustrat în figura 9.36.

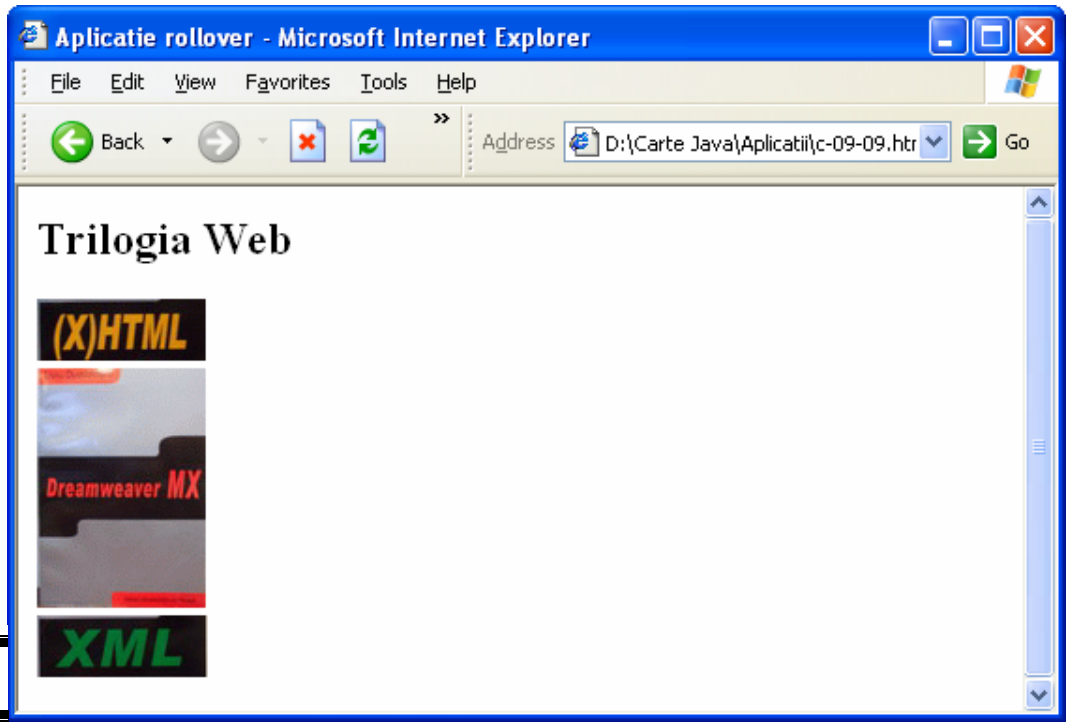


Figura 9.36

Analizaţi documentul XHTML şi răspundeţi la următoarele întrebări:

- ✓ Ce semnifică notaţia `this.id`?
- ✓ Care este rolul gestionarilor de evenimente:
  - `onClick="location='ldXHTML.htm'"`
  - `onClick="location='ldDREAMWEAVER.htm'"`
  - `onClick="location='ldXML.htm'"`
- ✓ Care sunt deficienţele acestui document. Precizaţi cel puţin două!

Creaţi animaţii simple cu JavaScript

Chiar dacă `rollover-ul` este aplicaţia cea mai folosită a imaginilor dinamice, JavaScript poate fi de asemenea utilizat şi pentru crearea animaţiilor simple. Pentru animaţii mai complexe va trebui însă să folosiţi Java sau un soft dedicat, ca de exemplu, Macromedia Flash.

Pentru a crea o animaţie simplă cu JavaScript [3] se recomandă parcurgerea următoarelor etape:

- ✓ salvaţi cadrele de animaţie într-o matrice;
- ✓ preîncărcaţi imaginile (cadrele de animaţie);

- ✓ stabiliţi viteza de derulare a animaţiei.

Pentru mai multe detalii privind crearea animaţiilor simple JavaScript, consultaţi:

- ✓ Michael Moncur, JavaScript 1.5 Nouvelle edition, Campus Press, 2003, pag. 227-234
- ✓ Michael Dreyfus, Codes en Stock JavaScript, Ed. Campus Press, 2001
- ✓ Micro Application, JavaScript Professionel, Paris, 2001
- ✓ David Flanagan, JavaScript, La référence 4′ édition, Ed. O'Reilly, 2002
- ✓ Jean-Christophe Gigniac, Cédric Nilly, JavaScript, Micro Application 2002, Paris, pag. 250 – 260
- ✓ http://JavaScript.Internet.com/scrolls/animated-message.html
- ✓ http://www.wsabstract.com/cutplastejava.shtml
- ✓ http://hotvired.lycos.com/webmonkey/reference/ JavaScript_code_library
- ✓ http://webopedia.Internet.com/TERM/J/JavaScript.html
- ✓ http://Javascript.internet.com/message/animated-tooltip.html
- ✓ http://Javascript.internet.com/page-details/delayed-gif.html

Creaţi imagini reactive (client) cu JavaScript

Imaginile reactive (*imagemaps*, în limba engleză) reprezintă un mijloc practic de simplificare a navigării într-un site Web.

Există două tipuri de imagini reactive: *client* şi *server.*

Zonele (cu un anume contur geometric) unei imagini reactive *client* se definesc direct în codul (X)HTML, fiecare dintre acestea jucând rolul unui `link`. În consecinţă, acest tip de imagine `map` poate fi combinat cu JavaScript. În cazul unei imagini reactive *server*, zonele sunt definite într-un fişier dedicat care se află pe server.

Crearea unei imagini reactive (*client*) necesită următoarele resurse:

- ✓ o imagine în format *GIF* sau *JPEG*;
- ✓ un fişier (*MAP*) care conţine zonele imaginii;
- ✓ atributul *USEMAP*.

Remarci

- ✓ Navigatoarele recente nu afişează descrierea unei imagini reactive în bara de stare, atunci când atributul `href` figurează în link-uri. Dacă utilizaţi imagini reactive ca link-uri, puteţi folosi gestionarul de evenimente `onClick` cu `location.href`.

- ✓ Pentru a descrie fiecare zonă a imaginii reactive va trebui să cunoaşteţi coordonatele punctelor care le delimitează. Există numeroase programe, printre care: `MapEdit` sau `LiveImage`.
- ✓ Zonele imaginii reactive sunt definite cu tag-urile (X)HTML: `<map>` şi `<area>`.

Iată cum procedăm pentru a crea cu limbajul JavaScript o imagine reactivă client, pornind de la imaginea `follow.jpg`. În momentul în care plasaţi mouse-ul deasupra zonei reactive de formă dreptunghiulară decupate în imaginea `follow.jpg`, în bara de stare se va afişa mesajul: *„ZIUA ÎN CARE VIN PEŞTII"*.

1. Inseraţi imaginea `follow.jpg`.

2. Introduceţi în tag-ul `<img>` atributul `usemap="#?"`, înlocuind semnul de întrebare (?) cu numele (JOHNSON) pe care doriţi să-l atribuiţi imaginii `map` (figura 9.37).

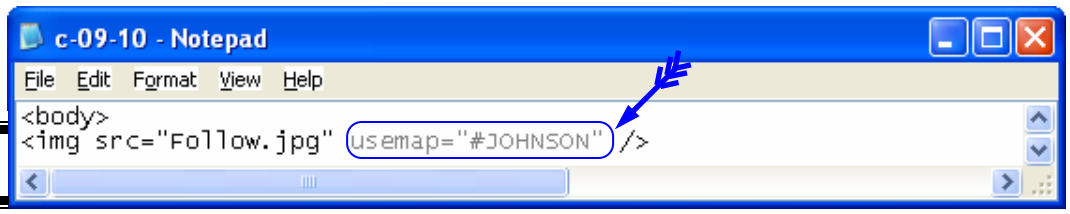


Figura 9.37

3. Introduceţi tag-ul `<map name="?">`, înlocuind semnul de întrebare (?) cu numele pe care l-aţi atribuit imaginii `map` (figura 9.38).

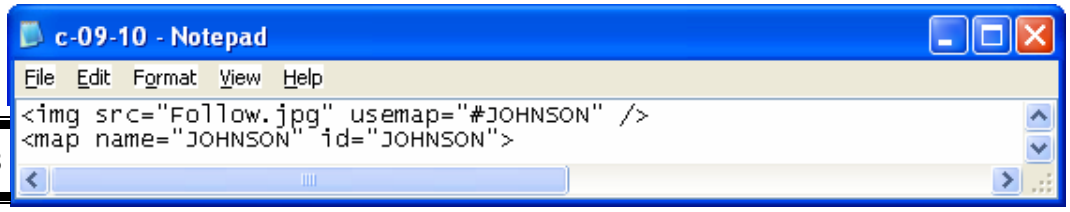


Figura 9.38

4. Introduceţi, în tag-ul `<area>` atributele `shape`, `target` şi `coords` (figura 9.39).

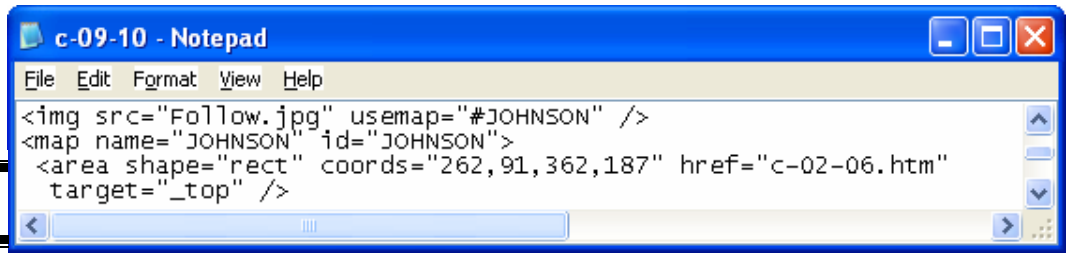
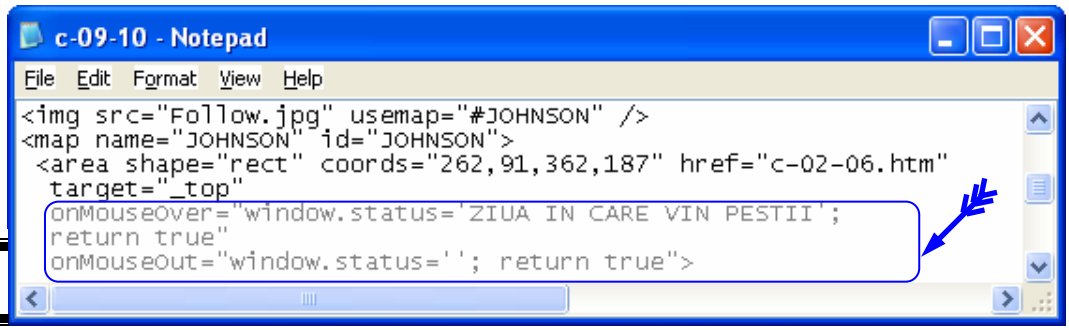
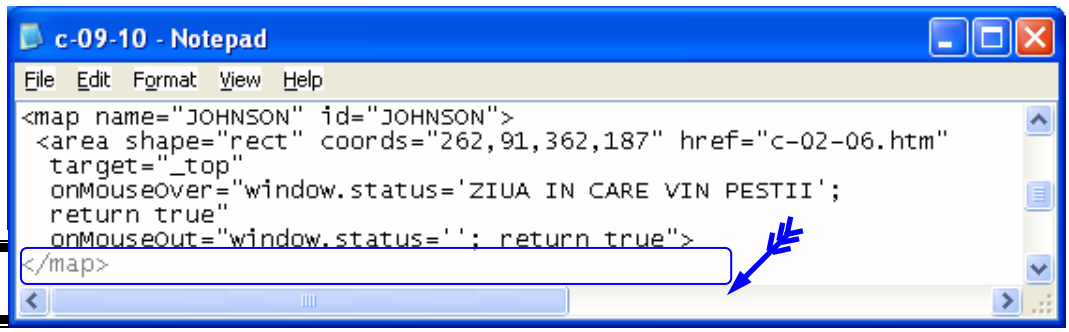


Figura 9.39

5. Introduceţi în tag-ul `<area>` gestionarul de evenimente asociat `onMouseOver` care afişează în bara de stare mesajul: *„ZIUA ÎN CARE VIN PEŞTII"*.



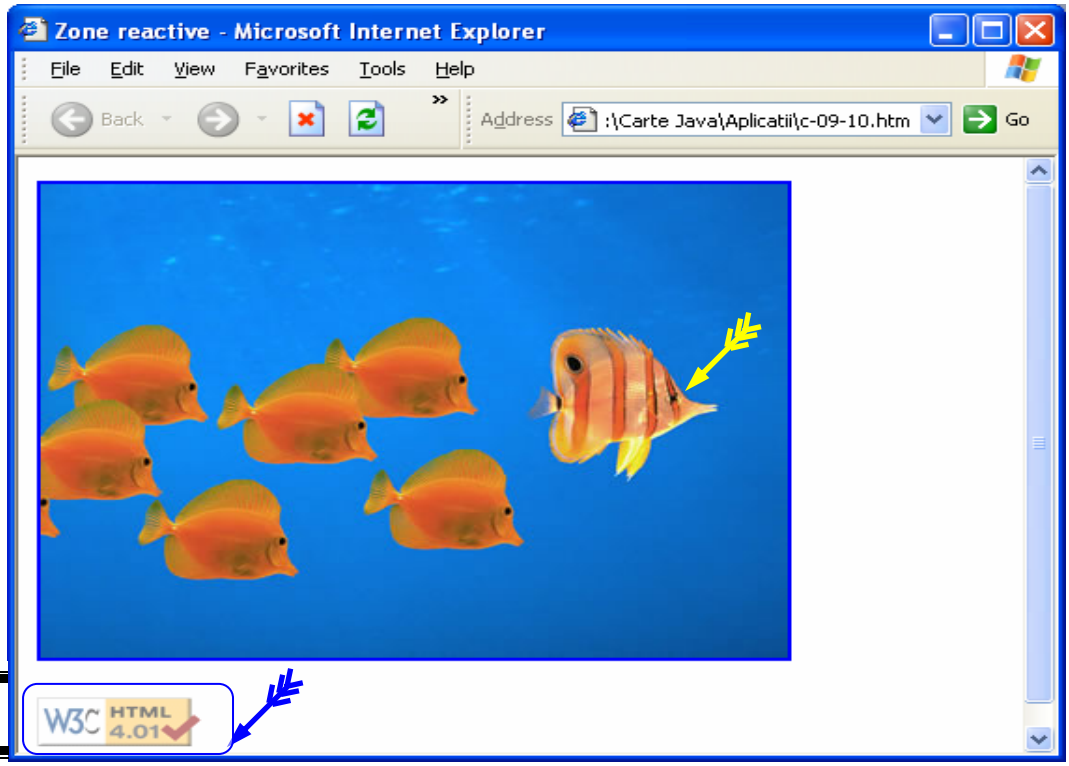
6. Introduceţi tag-ul final `</map>` (figura 9.41).



7. Validaţi documentul HTML 4 strict cu aplicaţia `validator`.

8. Inseraţi codul HTML care afişează icon-ul de conformitate W3C.

9. Vizualizaţi pagina Web într-un browser (figura 9.42).



EXEMPLUL 9 JAVASCRIPT

Vom relua în cadrul acestui exemplu problema rezervoarelor (cilindrice echilaterale), invitându-vă în cele ce urmează să utilizați imagini reactive `client` pentru generarea raportului „`Situa•ia livr•rilor`”. Zonele reactive (`client`) se referă la cele trei rezervoare (figura 9.43).

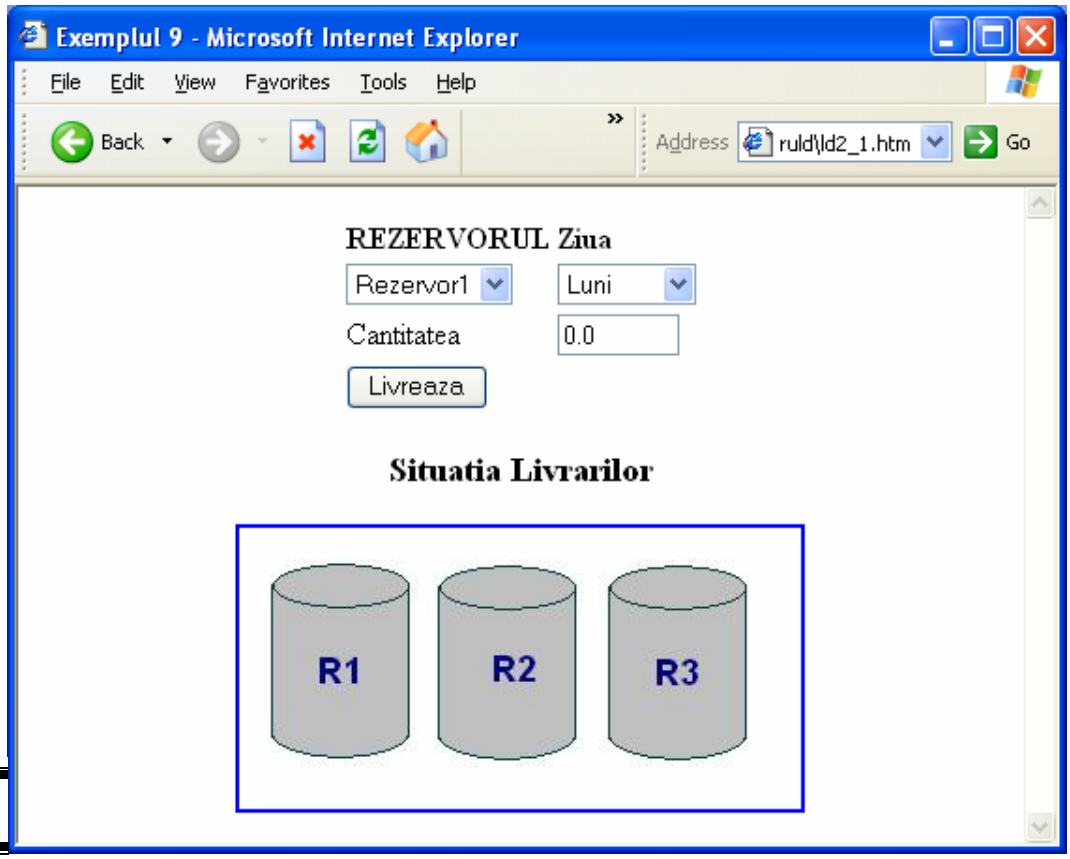


Figura 9.43

V-am pregătit şi o surpriză!

De această dată vă vom lăsa dumneavoastră plăcerea de a realiza singuri documentația de analiză şi proiectare a programului JavaScript corespunzătoare acestui exemplu. Succes!

❑ **Codificarea în limbajul JavaScript** (figura 9.44)

```
<html>
<head>
<title>Exemplul 9</title>
<script language="JavaScript">
<!---
var Z = new Array("Luni","Marti","Miercuri","Joi","Vineri");
 function trunchiaza(x)
     {
    var s=""+x;
     i=s.indexOf(".");
     if(i!=-1){
       s=s.substring(0,i+3);
       }
     return s;
     }
```

Figura 9.44

```
function validate(item, min, max) {
   var rVal = false;
   var x=parseFloat(item.value);
     if(isNaN(x))
       alert("Valoare gresita pentru cantitate!");
      else
     if (x < min)
    alert("Valoare gresita pentru cantitate!Valoarea trebuie >" + min);
      else if (x> max)
       alert("Valoare gresita pentru cantitate!  Valoarea trebuie sa fie < " + max);
         else
      rVal = true;
   return rVal;
}
a=new Array(3);
a[0]=new Array(5);
a[1]=new Array(5);
a[2]=new Array(5);
for(i=0;i<3;i++)
 for(j=0;j<5;j++)
     a[i][j]=0.0;
var RezWindow=null;
function trimite(){
 var ir=f1.Rezervor.selectedIndex;
 var z=f1.Zile.selectedIndex;
   a[ir][z]+=parseFloat(f1.T1.value);
}

function afispartial(rez){
var i,j,k;
//CALCUL TOTAL LIVRARI
 s=0.0;
 for(j=0;j<5;j++)
   s+=a[rez][j]
//CALCUL MEDIE PE ZI
med=s/5;
//CALCUL MAX SI MIN
    max=a[rez][0];
    min=a[rez][0];
      jmax=0;jmin=0;
       for(j=0;j<5;j++){
         if(max<a[rez][j]){max=a[rez][j];jmax=j;}
         if(min>a[rez][j]){min=a[rez][j];jmin=j;}
  }
 //AFISARE REZULTATE
 if(RezWindow!=null)RezWindow.close();
 RezWindow=window.open("","   'toolbar=yes,scrollbars=yes,menubar=no,
width=500,height=350');
k=rez+1;
RezWindow.document.writeln("<center><p><b>SITUATIA LIVRARILOR REZERVOR
R"+k+"</B></p></center>");
RezWindow.document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
RezWindow.document.writeln("<td><b>ZIUA</b><td><b>REZERVOR R"+k+"</b></td></tr>");
```

Figura 9.44 *(continuare)*

```
    for(i=0;i<5;i++) {
     RezWindow.document.writeln("<tr><td>" + Z[i]+"</td>");
      RezWindow.document.writeln("<td>" + a[rez][i]+ "</td></tr>");
    }
   RezWindow.document.writeln("</table></center><p><p>");
   RezWindow.document.writeln("<p><font color=green size=+1>Total Livrari:"+s+"</font>");
   RezWindow.document.writeln("<br><font color=green size=+1>Media Zilnica:" +
trunchiaza(med) + "</font>");
   RezWindow.document.writeln("<br><font color=green>Livrarea maxima:"+max+" s-a facut in
ziua de "+Z[jmax]+"</font>");
RezWindow.document.writeln("<br><font color=green>Livrarea minima:"+min+" s-a facut in ziua
de "+Z[jmin]+"</font>");
RezWindow.document.writeln("</html>");
}
// -->
</script>
</head>
<body>
<center>
<form name="f1"><table border=0>
<tr><td><b>REZERVORUL</b><td><b>Ziua</b>
<tr><td><select size="1" name="Rezervor">
  <option selected>Rezervor1</option>
  <option>Rezervor2</option>
  <option>Rezervor3</option>
  </select>
<td>
<select size="1" name="Zile">
  <option>Luni</option>
  <option>Marti</option>
  <option>Miercuri</option>
  <option>Joi</option>
  <option>Vineri</option>
  </select>
<tr>
<td>Cantitatea
<td><input type="text" name="T1" size="7" value="0.0" onChange="validate(this,0,20);">
<tr>
<td><input type="button" value="Livreaza" onClick="trimite();">
</table>
</center>
</form>
<center>
<h3>
Situatia Livrarilor
</h3>
<map name="rezervorMap">
   <area name="R1" coords="16,18,91,125" href="javascript:afispartial(0)"
    onMouseOver="window.status='Rezervorul R1';return true"
     onMouseOut="window.status=' ';return true">
   <area name="R2" coords="105,18,180,125" href="javascript:afispartial(1)"
onMouseOver="self.status='Rezervorul R2';return true"
onMouseOut="self.status=';return true">
```

Figura 9.44 *(continuare)*

```
   <area name="R3" coords="195,18,272,125" href="javascript:afispartial(2)"
onMouseOver="window.status='Rezervorul R3';return true"
onMouseOut="window.status='';return true">
</map>
<img src="rezervoare.jpg" align="top" usemap="#rezervorMap">
</body>
</html>
```

Figura 9.44 *(continuare)*

Remarci

- ✓ Script-ul (inserat în secţiunea `<head>` a documentului (X)HTML) conţine funcţiile cunoscute din exemplele anterioare.
- ✓ Pentru crearea celor trei imagini reactive, corespunzătoare rezervoarelor R1, R2, R3 s-a folosit imaginea `rezervoare.jpg`; zonele imaginilor reactive (R1, R2, R3) au fost definite cu tag-urile (X)HTML `<map>` şi `<area>`. În toate cele trei tag-uri `<area>` au fost definite atributele: `name`, `coords`, `href` şi gestionarii de evenimente `onMouseOver` şi `onMouseOut`.
- ✓ Tag-ul `<img>` conţine în mod obligatoriu `usemap="#rezervorMap"`.

În continuare, vă invităm să testaţi script-ul, plasând mouse-ul şi executând clic deasupra fiecărei zone reactive R1, R2, R3. Analizaţi rezultatele execuţiei programului (figura 9.45, figura 9.46, figura 9.47).

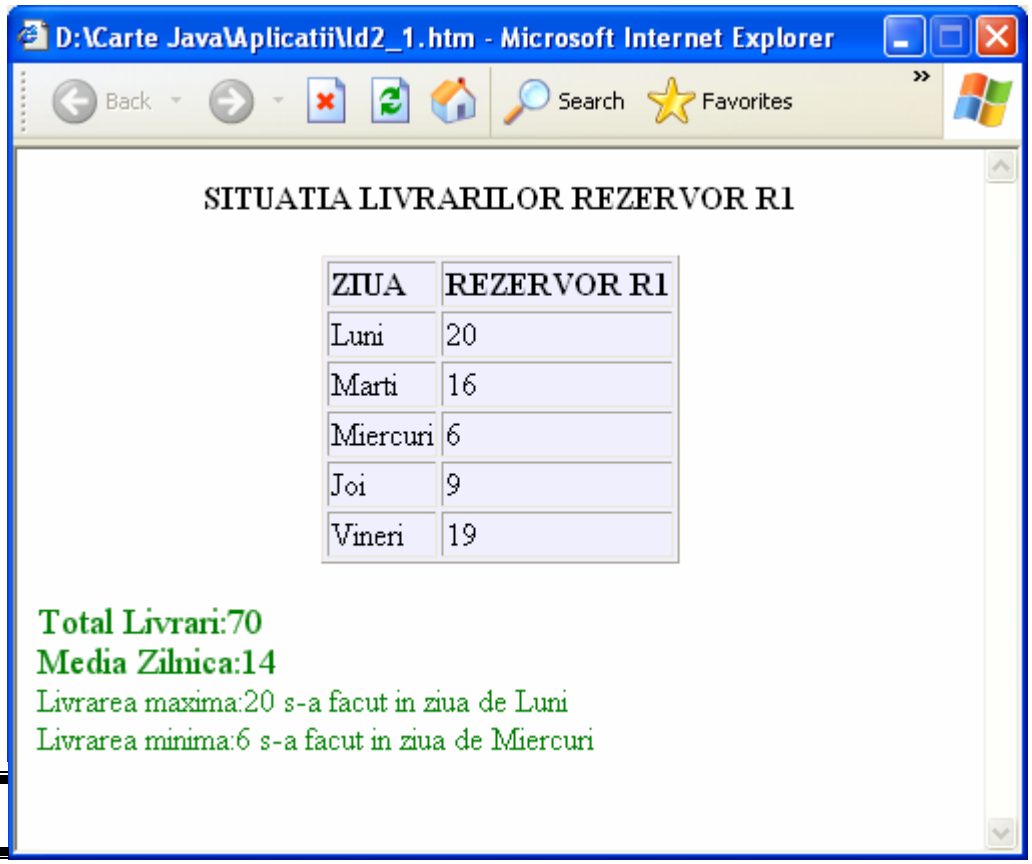


Figura 9.45

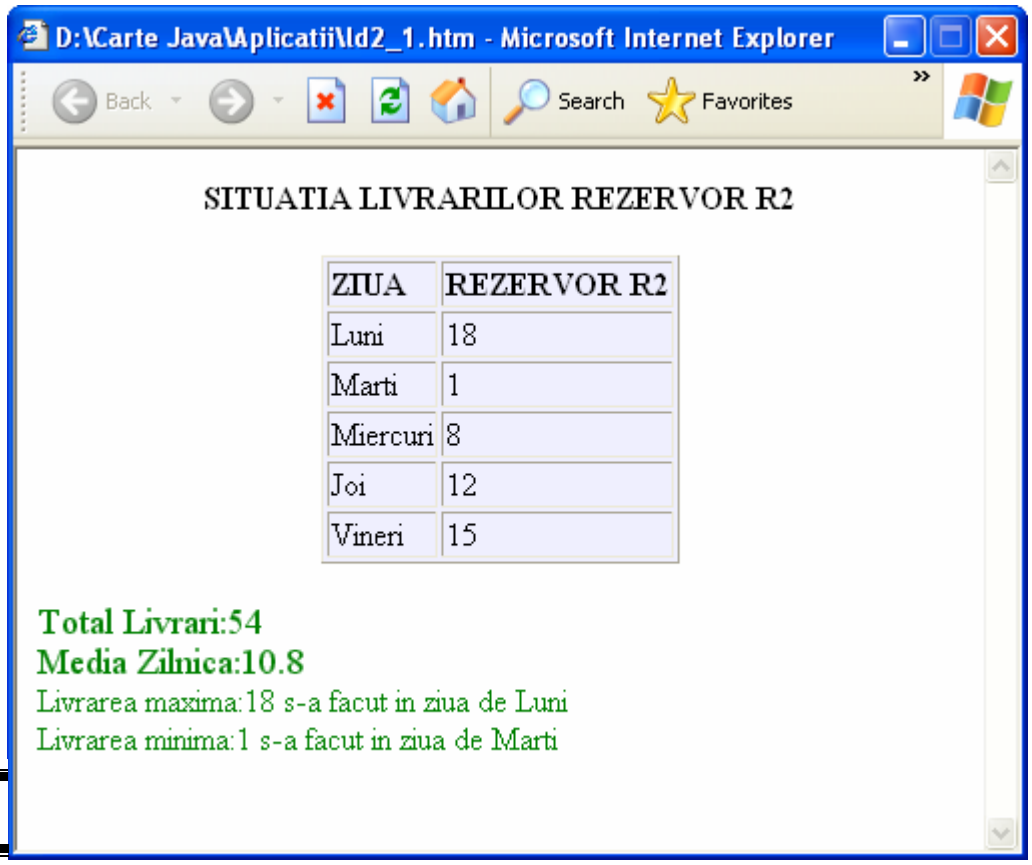


Figura 9.46

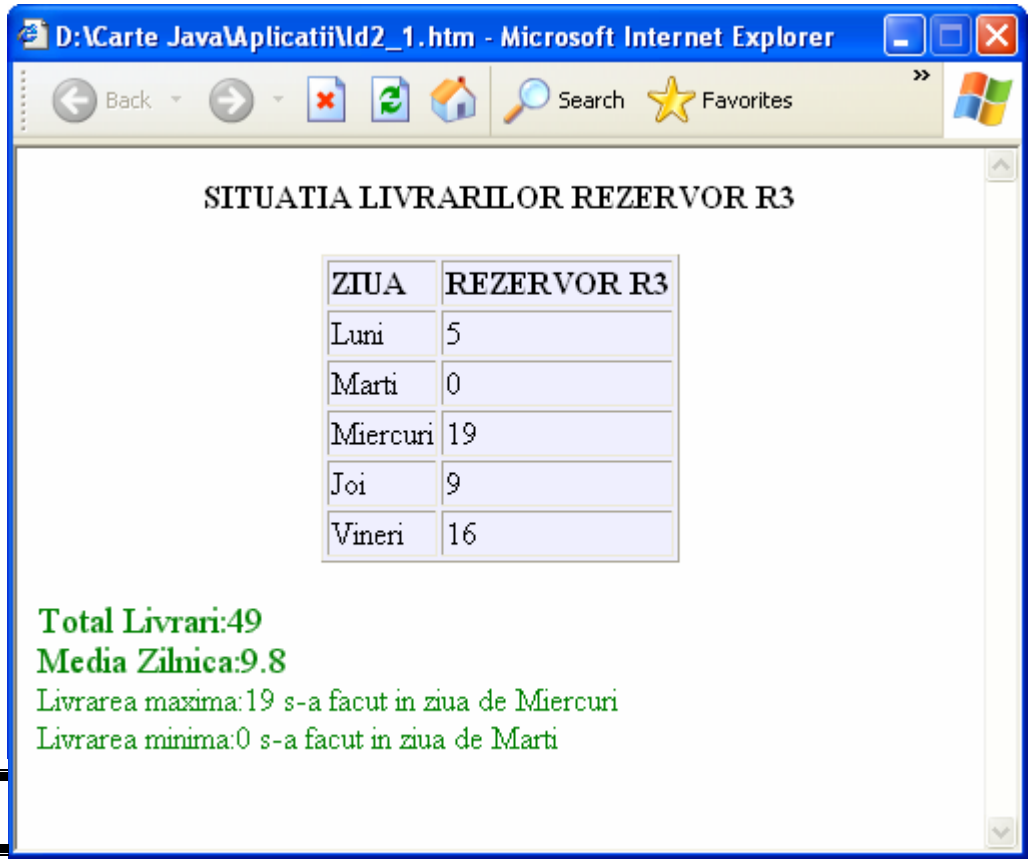


Figura 9.47

Aplicaţie

❑ Realizaţi o pagină Web care conţine o hartă interactivă a Europei. Atunci când utilizatorul plimbă mouse-ul (fără a executa clic) pe una din ţările Europei se vor afişa pentru ţara respectivă, într-un formular, următoarele informaţii: ţara, capitala, populaţia, suprafaţa, moneda (figura. 9.48).

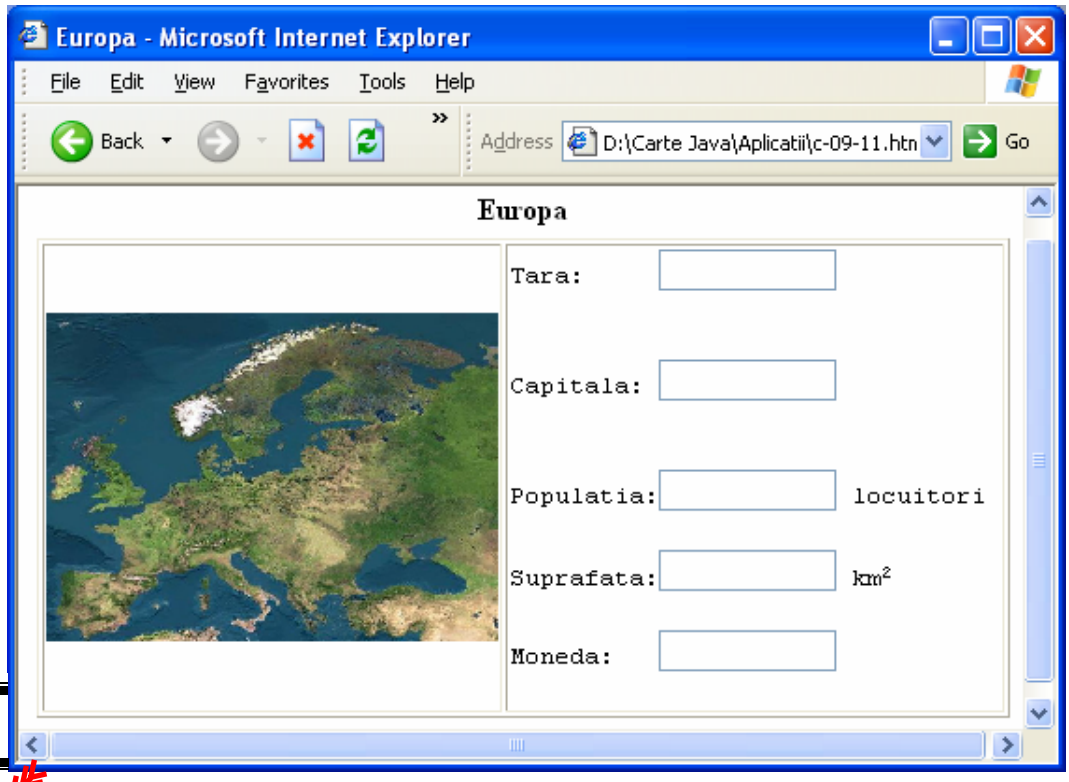


Figura 9.48

Indicaţii:

- ✓ Limitaţi (nu din motive politice!) numărul ţărilor la cel mult 12.
- ✓ Utilizaţi tag-ul `<area>` cu atributele: `coords`, `shape`, `href` (vezi figura 8.49) şi cu gestionarii de documente: `onMouseOver` şi `onMouseOut`.

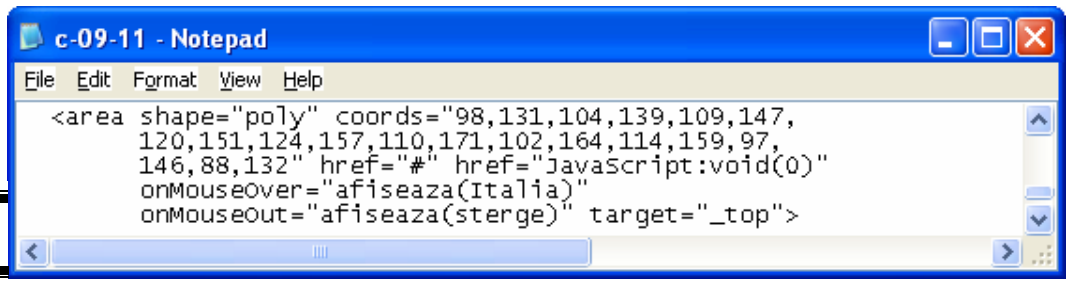


Figura 9.49

- ✓ Formularul nu conţine nici un buton ci numai zone (5) de tip text pentru afişarea informaţiilor privind ţările Europei. Aceste informaţii sunt conţinute în script, mai precis în 12 matrice cu următoarele elemente: numele ţării, capitala, populaţia, suprafaţa, moneda (vezi figura 9.50). Script-ul conţine de asemenea şi funcţia `afi•eaz•()`.

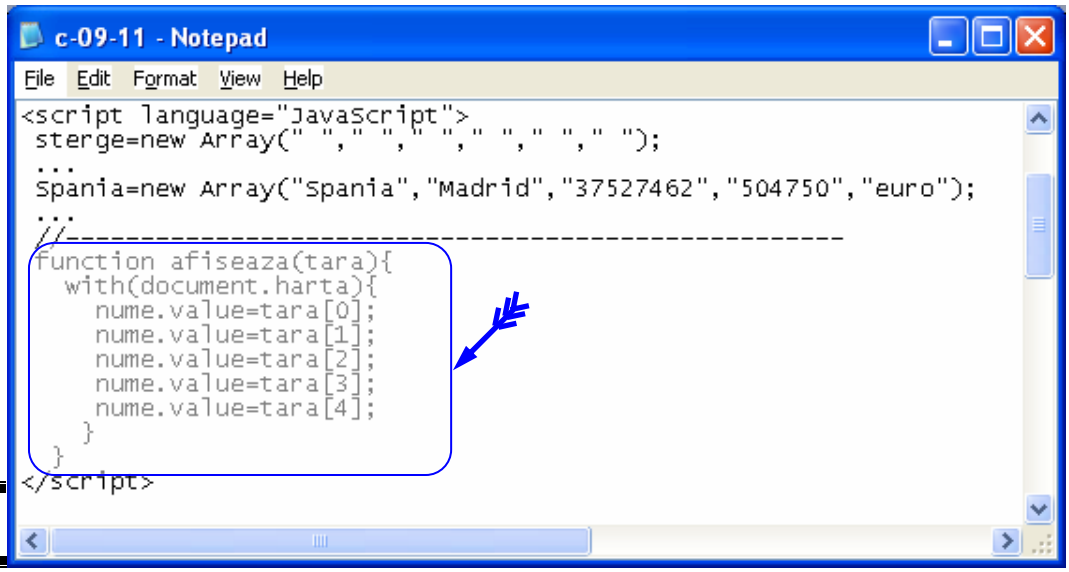
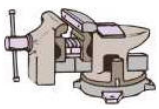


Figura 9.50

Testați-vă cunoștințele



1. Ce reprezintă obiectul `Image`?
2. Care sunt proprietăţile şi metodele obiectului `Image`?
3. Care sunt gestionarii de evenimente ai obiectului `Image`?
4. Cum creaţi o instanţă a unui obiect `Image`?
5. Ce este un `rollover`?
6. Care sunt resursele necesare pentru crearea unei imagini reactive `client`?
7. Poate fi plasat un `rollover` într-o imagine reactivă?
8. În figura 9.51 se prezintă un document HTML care conţine titlul: „Ziua în care vin peştii”.

Figura 9.51

Atunci când se încarcă acest document HTML se obţine ceea ce este ilustrat în figura 9.52.

Figura 9.52

În momentul în care plimbaţi mouse-ul (fără a executa clic) pe titlul afişat în pagina Web se obţine ceea ce este ilustrat în figura 9.53.

Figura 9.53

Ce semnifică:

- ✓ `onMouseOver="peste('follow.jpg')"`
- ✓ `document.images[0].src=nume.`

Perfecţionaţi acest mic program după cum urmează (figura 9.54).

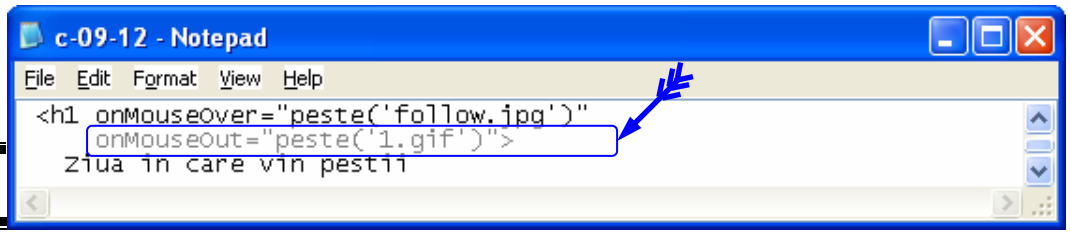


Figura 9.54

Precizaţi care este rolul gestionarului de evenimente `onMouseOut` pe care l-am introdus în tag-ul `<h1>`.

9. Care este obiectul JavaScript care reprezintă a doua imagine din cadrul unei pagini Web:
   - image[2];
   - images[2];
   - images[1].
10. Animaţiile JavaScript par a fi puţin … simpliste! Pot fi create în JavaScript animaţii mai complexe?

Vizitați site-urile

- ✓ http://Javascript.internet.com/games/terablox.htm;
- ✓ http://JavaScript.Internet.com
- ✓ http://www.geoticies.com/Silicon Valley/7116/
- ✓ http://www.serve.com/hotsyte/
- ✓ http://Javascript.internet.com/page-details/delayed-gif.html
- ✓ http://Javascript.internet.com/bgeffects/mouse-fireworks.html
- ✓ http://Javascript.internet.com/messages/elastic-bullets.html
- ✓ http://Javascript.internet.com/miscellaneous/image-slideshow.html
- ✓ http://Javascript.internet.com/navigation/thumbmail-navigator.html
- ✓ http://Javascript.internet.com/miscellaneous/kitykity's--photo-album.html
- ✓ http://www.kitykity.com/photoalbum
- ✓ http://Javascript.internet.com/scrolls/animated-message.html
- ✓ http://Javascript.internet.com/bgeffects/persistent-layer.html
- ✓ http://Javascript.internet.com/navigation/over-line-text-link.html
- ✓ http://www.webwizguide.info/directory/directory/asp?cat=java&PagePosition=1

Conversația 10

Obiectele Frame şi Layer

În această conversație:

Obiectele Frame

Cea mai mare parte a navigatoarelor actuale (în particular, navigatoarele recente Netscape şi Microsoft) recunosc cadrele (*frames,* în limba engleză) care permit divizarea paginii Web în mai multe zone. În fiecare zonă a paginii se poate afişa un document (X)HTML sau rezultatul unui script.

Decizia de a utiliza cadre vă aparţine!

Indiferent dacă apreciaţi sau nu cadrele, site-urile Web bazate pe cadre există şi vor mai exista. Din punct de vedere al programatorului JavaScript, este la fel de uşor să lucrezi pe un site care conţine cadre sau să lucrezi pe un site fără cadre.

Atunci când o fereastră conţine mai multe cadre, fiecare dintre ele este reprezentat în JavaScript printr-un obiect `Frame`.

Acest obiect este echivalent cu un obiect `Window` atâta timp cât el serveşte la manipularea cadrelor şi nu a ferestrelor. Numele obiectului `Frame` este acelaşi cu cel pe care îl afectaţi atributului `name` al tag-ului `<frame>`.

Remarci:

- ✓ Cuvintele cheie `window` şi `self` permit referirea la cadrul curent.
- ✓ Cuvântul cheie `parent` permite referirea la fereastra de nivel superior (cel mai des, fereastra principală).
- ✓ Fiecare din obiectele `Frame` ale unei ferestre este un fiu al obiectului părinte `Window`.

Documentul (X)HTML din figura 10.1 împarte fereastra în patru zone. Dacă aţi inserat un script JavaScript în documentul `stângasus.htm`, el va face referire la documentele care aparţin altor cadre: `parent.dreaptasus`, `parent.stângajos` etc. Cuvintele cheie `window` şi `self` vor face referire la însuşi cadrul `stângasus`.

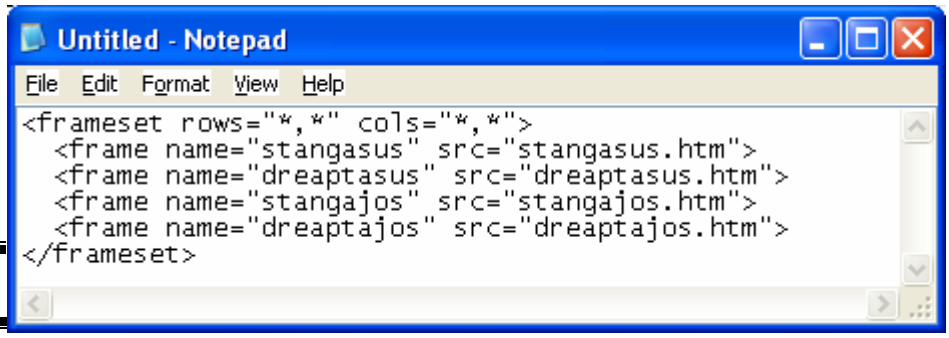


Figura 10.1

Remarcă. Dacă utilizaţi cadre situate în interiorul altor cadre (imbricate) lucrurile se complică puţin: `window` reprezintă întotdeauna cadrul curent, `parent` reprezintă setul de cadre (`frameset`) care conţine cadrul curent, iar `top` reprezintă setul de cadre principal care le conţine pe celelalte.

Matricea frames

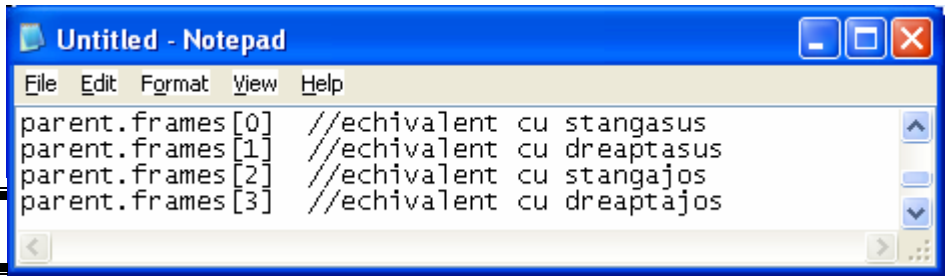
Matricea `frames` conţine un obiect `Window` pentru fiecare cadru al paginii `Web` (vezi figura 10.2).

În loc să vă referiţi la cadrele unui document prin numele lor, utilizaţi matricea `frames`.

Această matrice stochează informaţiile pentru fiecare din cadrele unui document. Numărul de indice (index) al cadrelor începe întotdeauna cu zero şi cu primul tag `<frame>` al setului de cadre al documentului.

Iată cum puteţi referi cadrele din figura 10.1 cu ajutorul matricii `frames`, (vezi figura 10.2).

Figura 10.2



Remarci:

- ✓ Puteţi folosi oricare din cele două metode de acces la elementele matricii `frames` (prin numărul de index/prin nume).
- ✓ Dacă documentul dumneavoastră conţine de exemplu 13 cadre, fără îndoială este mult mai simplu să utilizaţi matricea `frames`. Dacă însă, documentul conţine decât două cadre, evident este mult mai bine să utilizaţi numele cadrelor.

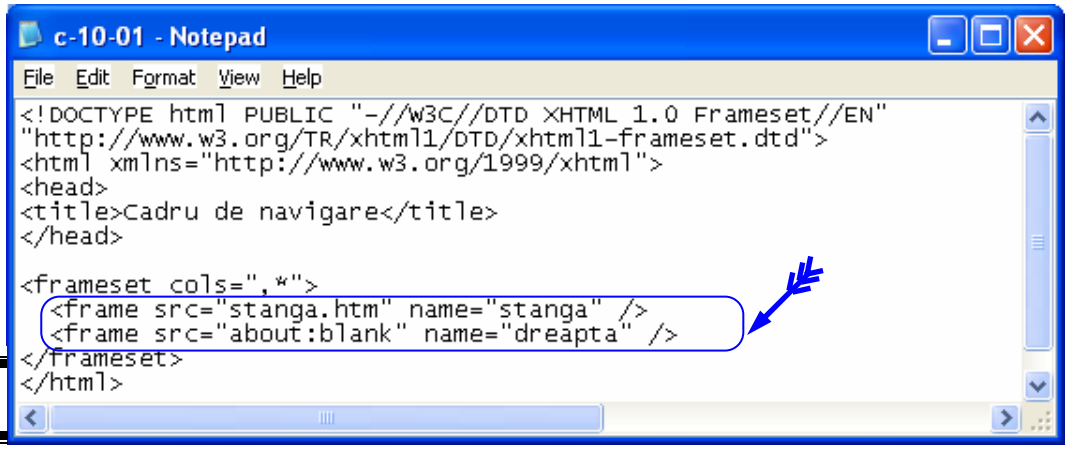
Iată cum creaţi un cadru de navigare (vezi figura 10.3) care să permită modificarea conţinutului unui alt cadru.

Figura 10.3

• (X)HTML • Dreamweaver MX • XML • JavaScript	

1. Creaţi un document (X)HTML care divizează fereastra în două cadre (figura 10.4).

Figura 10.4



Remarci:

- ✓ Programul din figura 10.4 creează două cadre (la stânga şi la dreapta ferestrei).
- ✓ Tag-urile `<body>` sunt înlocuite cu tag-urile `<frameset>`.

2. Creaţi documentul (X)HTML pentru cadrul din stânga, care va servi pentru navigarea în site (figura 10.5).

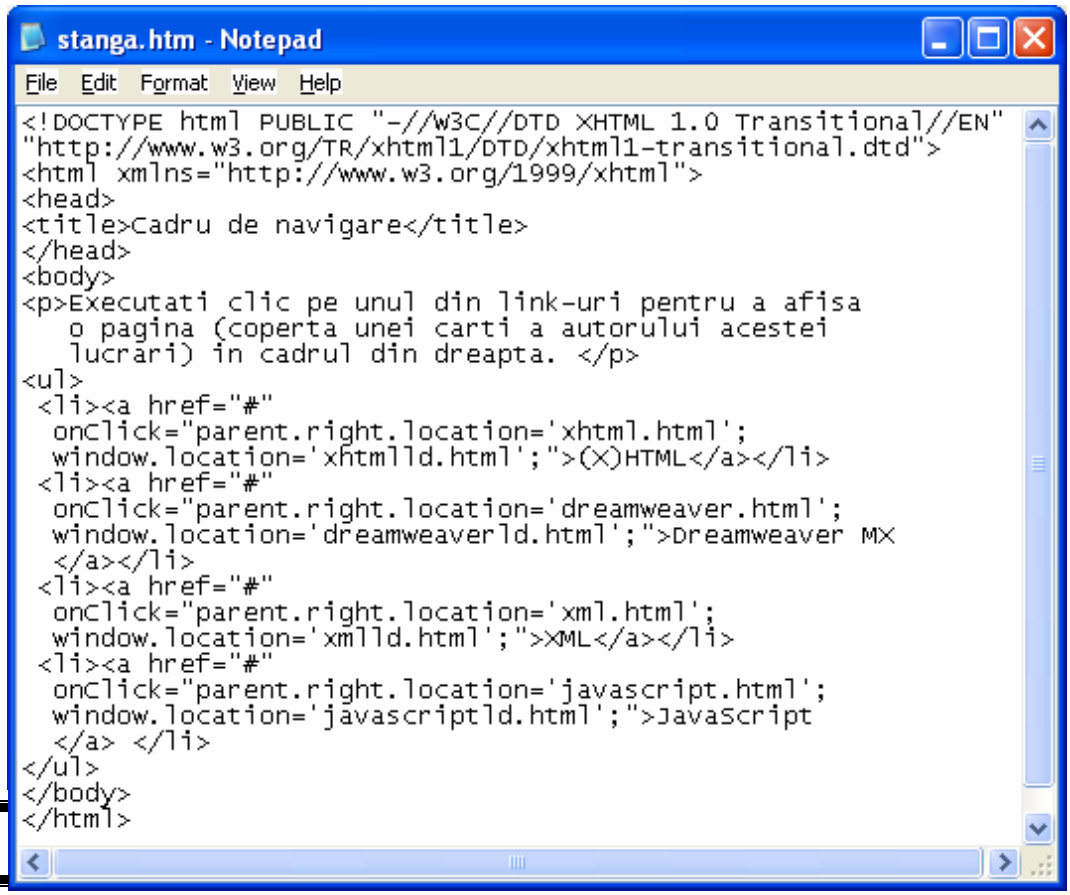


Figura 10.5

Remarci:

- ✓ Gestionarii de evenimente `onClick` se repetă (cu mici modificări) pentru fiecare din cele patru link-uri (vezi tag-urile <a>).
- ✓ Gestionarii de evenimente `onClick` sunt declanşaţi prin evenimentul `click`, care afişează un document în cadrul din dreapta.
- ✓ Deoarece script-ul se găseşte (el însuşi) într-un cadru, numele celuilalt cadru trebuie să fie precedat de cuvântul cheie `parent`.
- ✓ Fiecare din obiectele `frame` ale unei ferestre este un fiu al obiectului părinte `Window`.
- ✓ Cuvintele cheie `window` şi `self` permit referirea cadrului curent.
- ✓ Cuvântul cheie `parent` permite referirea la fereastra de nivel superior (de regulă fereastra principală).
- ✓ Utilizarea limbajului JavaScript permite modificarea simultană a conţinutului mai multor cadre.

3. Testaţi script-ul.

3.1 Salvaţi documentul XHTML al cadrului de navigare sub numele `stânga.html`.

3.2 Deschideţi în browser documentul HTML care divizează fereastra în două cadre (vezi figura 10.6).

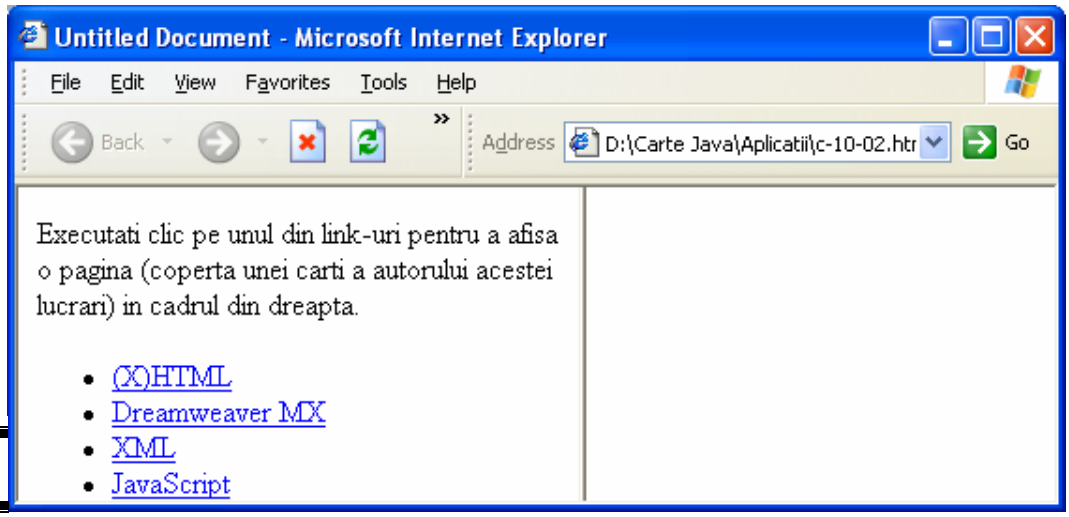


Figura 10.6

3.3 Testaţi link-urile din cadrul din stânga.

În figura 10.7 se prezintă rezultatul execuţiei script-ului în navigatorul Internet Explorer.

Figura 10.7

EXEMPLUL 10.1 JAVASCRIPT

❑ Formularea problemei

Vom aborda şi în cadrul acestei conversații aceeaşi problemă pe care am abordat-o în conversația precedentă (EXEMPLUL 9 JAVASCRIPT), cu singura deosebire (importantă!) că pentru afişarea rezultatelor vom folosi obiectele `Frame` (Rezultatele vor fi afişate într-un cadru inserat în documentul curent şi nu într-o fereastră distinctă).

Introducerea valorilor pentru livrări se face printr-o singură zonă de text (vezi EXEMPLUL 9 JavaScript, *Varianta 2*). Selecția rezervorului şi a zilei se fac printr-o listă de selectare.

❑ Specificații de programare

În figura 10.8 este prezentat ecranul (intrare/ieşire) cu „RAPORT LIVRĂRI REZERVOR R1". Tabela de variabile este prezentată în figura 10.9.

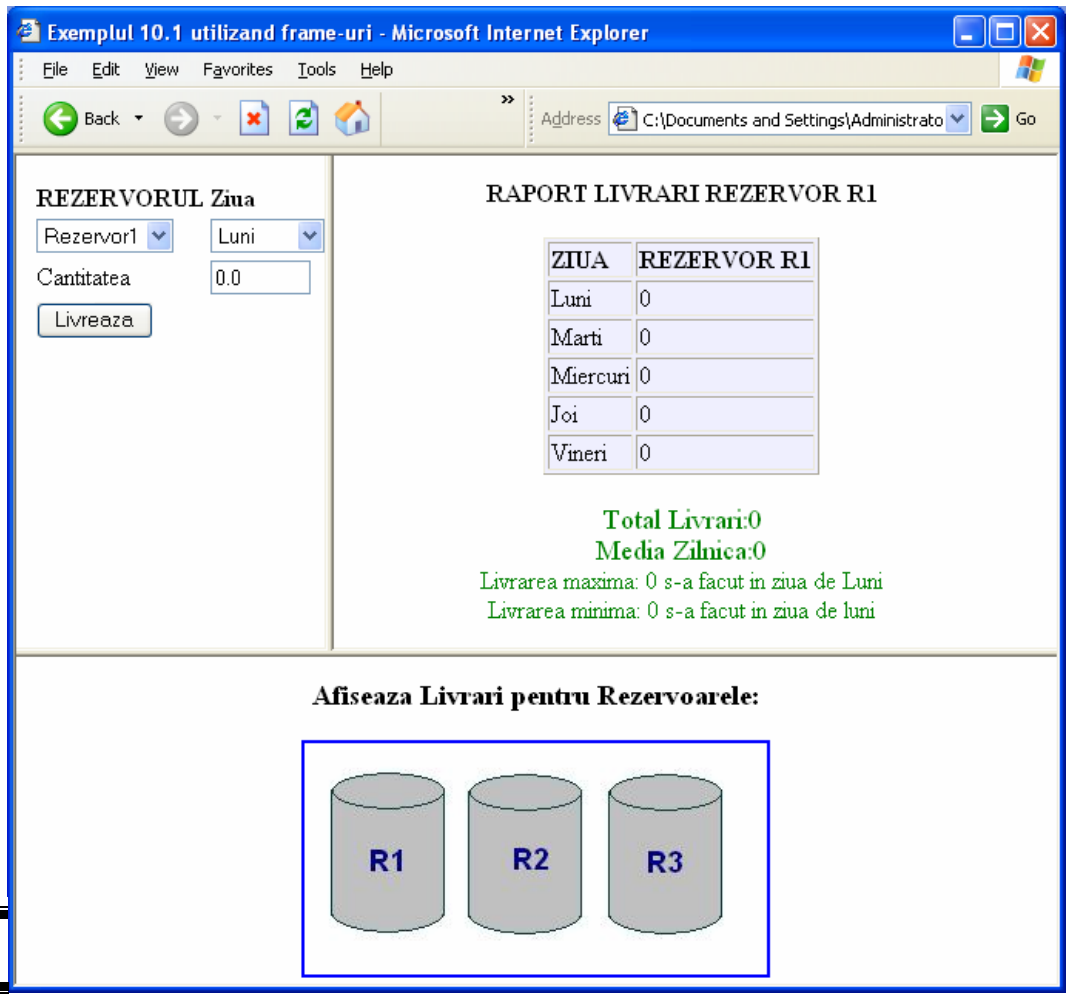


Figura 10.8

Tabela de variabile

Variabile de intrare	*Variabile de stare*	*Variabile de ieşire*
Rezervor: (obiect) listă simplă de selecție pentru cele trei rezervoare Zile: (obiect) listă simplă de opțiuni pentru selecția zilei din săptămână T1: (obiect) zona de editare în care se va introduce valoarea livrărilor Inputd: (obiect) frame-ul ce conține obiectele Rezervor, Zile, T1 sit:– (obiect) frame ce conține zonele reactive pentru afişare	s: (real) folosit pentru calculul sumelor parțiale pe zile rval: (logic) indică faptul ca valoarea introdusă în zona de editare este validă sau nu x: (real) valoarea reală a textului introdus în zona de editare Z: (vector) numele zilelor săptămânii jmax,jmin: (numere întregi) păstrează indicii livrărilor maxime şi minime din matricea a	raport: (obiect) frame-ul în care se vor afişa rezultatele pentru un rezervor med: (real) folosit pentru calculul mediei zilnice pe rezervor max,min: (numere reale) păstrează valoarea maximă şi minimă pentru livrări a: (matrice de numere reale) păstrează valorile livrărilor pe zile şi rezervoare

Figura 10.9

❑ Documentația de proiectare

Pseudocodul pentru EXEMPLUL 10.1 JAVASCRIPT este prezentat în figura 10.10.

Pseudocodul

```
EXEMPLUL10.1   BEGIN
                    //initializeaza vectorul Z cu numele zilelor
                    z=luni,marti,miercuri,joi, vineri
                    //aloca spatiu de memorie si initializeaza matricea livrarilor
FORI                FOR(i=0;i<3;i++)
FORJ                        FOR(j=0;j<5;j++)
                                a[i,j]=0.0
FORJ                        ENDFOR
FORI                ENDFOR
                    Afiseaza pagina ce contine frame-uri
                    //raspunde la evenimentele generate de butonul Livreaza
IFL                 IF(se apasa butonul Livreaza)
                            DO livreaza
IFL                 ENDIF
                    //raspunde la evenimentele generate de zonele reactive
IFA                 IF(se executa click pe zona unui rezervor)
                            DO afispartial(rezervor)
IFA                 ENDIF
                    //raspunde la evenimentele generate de zona de editare T1
IFV                 IF(se paraseste zona T1)
                            DO valideaza(T1,0,20)
IFV                 ENDIF
EXEMPLUL10.1   END

AFISPARTIAL    BEGIN
                    Date intrare:
                    rez-indicele rezervorului
                    // calculeaza total livrari
                    s=0
FORS                FOR(j=0;j<5;j++)
                            s=s+a[rez,j]
FORS                ENDFOR
                    med=s/5
                    // determinarea maximului si minimului
                    max=a[rez,0]
                    min=a[rez,0]
                    jmax=0
                    jmin=0
FORMAXMIN           FOR(j=0;j<5;j++)
IFMAX                       IF(max<a[rez,j])
                                    max=a[rez,j]
                                    jmax=j
IFMAX                       ENDIF
IFMIN                       IF(min>a[i,j])
                                    min=a[rez,j]
                                    jmin=j
IFMIN                       ENDIF
FORMAXMIN           ENDFOR
                    //afisare rezultate
                    //scrierea rezultatelor in frame-ul report
                    k=rez+1
                    raport.WRITE "SITUATIA REZERVORULUI R"+k
                    raport.WRITE "ZIUA  R"+k
FORRAP              FOR(j=0;j<5;j++)
                            raport.WRITE z[i], a[rez,j]
FORRAP              ENDFOR
```

Figura 10.10

```
                raport.WRITE "Total livrari: ",s
                raport.WRITE "Media: ",s
                raport.WRITE "Livrarea maxima: ",max,Zjmax
                raport.WRITE "Livrarea minima: ",min,Zjmin
AFISPARTIAL     END
```

Figura 10.10 *(continuare)*

❑ Codificarea în limbajul JavaScript

Documentul complet (X)HTML este prezentat în figura 10.11.

```
<html>
<head>
<script language="JavaScript">
<!---
// DEFINESTE VECTORUL CU ZILELE DIN SAPTAMANA
var Z = new Array("Luni","Marti","Miercuri","Joi","Vineri");
//functia de transformare intr-un sir de caractere a unui numar
  function trunchiaza(x){
     var s=""+x;
      i=s.indexOf(".");
      if(i!=-1){
        s=s.substring(0,i+3);
        }
      return s;
}
//VALIDAREA UNUI CAMP NUMERIC
function valideaza(item, min, max) {
   var rVal = false;
   var x=parseFloat(item.value);
     if(isNaN(x))    alert("Valoare gresita pentru cantitate!");
     else
     if (x < min)
        alert("Valoare gresita pentru cantitate!Valoarea trebuie >" + min);
    else if (x> max)
       alert("Valoare gresita pentru cantitate!  Valoarea trebuie sa fie < " + max);
          else
      rVal = true;
   return rVal;
}
// DECLARAREA MATRICEI IN CARE SE VOR PASTRA CANTITATILE LIVRATE
a=new Array(3);
a[0]=new Array(5);
a[1]=new Array(5);
a[2]=new Array(5);
for(i=0;i<3;i++)
 for(j=0;j<5;j++)
      a[i][j]=0.0;
//SALVAREA CAMPULUI NUMERIC DIN INPUT TEXT IN MATRICEA LIVRARILOR
function livreaza(){
//DETERMINA REZERVORUL
var ir=inputd.f1.Rezervor.selectedIndex;
//DETERMINA ZIUA
```

Figura 10.11

```
var z=inputd.f1.Zile.selectedIndex;
//ADAUGA CANTITATEA IN MATRICEA LIVRARILOR
   a[ir][z]+=parseFloat(inputd.f1.T1.value);
}
//CALCULEAZA SI AFISEAZA LIVRARILE DINTR-UN REZERVOR
function afispartial(rez){
var i,j,k;
//calcul total livrari
 s=0.0;
 for(j=0;j<5;j++)   s+=a[rez][j]
//CALCUL MEDIE PE ZI
med=s/5;
//CALCUL MAX SI MIN
    max=a[rez][0]; min=a[rez][0];
      jmax=0;jmin=0;
       for(j=0;j<5;j++){
           if(max<a[rez][j]){max=a[rez][j];jmax=j;}
           if(min>a[rez][j]){min=a[rez][j];jmin=j;}
        }
 //AFISARE REZULTATE
k=rez+1;
raport.document.open();
 raport.document.writeln("<center><p><b>RAPORT LIVRARI REZERVOR
R"+k+"</b></p></center>");
 raport.document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
 raport.document.writeln("<td><b>ZIUA</b><td><b>REZERVOR R"+k+"</b></td></tr>");
    for(i=0;i<5;i++) {
     raport.document.writeln("<tr><td>" + Z[i]+"</td>");
      raport.document.writeln("<td>" + a[rez][i]+ "</td></tr>"); }
    raport.document.writeln("</table></center><p><p>");
   raport.document.writeln("<center><p><font color=green size=+1>Total Livrari:"+s+"</font>");
   raport.document.writeln("<br><font color=green size=+1>Media
Zilnica:"+trunchiaza(med)+"</font>");
   raport.document.writeln("<br><font color=green>Livrarea maxima:"+max+" s-a facut in ziua
de" +Z[jmax]+"</font>");
raport.document.writeln("<br><font color=green>Livrarea minima:"+min+" s-a facut in ziua de "
+Z[jmin]+"</font>");
raport.document.writeln("</center></html>");
raport.document.close();
}
// -->
</script>
</head>
<title>Exemplul 4 utilizand frame-uri</title>
</head>
 <frameset rows="60%,40%">
    <frameset cols="30%,70%">
      <frame scrolling="no" noresize src="inputdata.html" name="inputd">
        <frame src="raport.html" name="raport">
     </frameset>
      <frame scrolling="no" noresize src="situatie.html" name="sit">
</frameset>
</html>
```

Figura 10.11 *(continuare)*

```
// INPUTDATA.HTML
<html>
<head>
</head>
<body>
<center>
<form name="f1">
<table border=0>
<tr><td><b>REZERVORUL</b><td><b>Ziua</b>
<tr><td><select size="1" name="Rezervor">
  <option selected>Rezervor1</option>
  <option>Rezervor2</option>
  <option>Rezervor3</option>
  </select>
<td>
<select size="1" name="Zile">
  <option>Luni</option>
  <option>Marti</option>
  <option>Miercuri</option>
  <option>Joi</option>
  <option>Vineri</option>
  </select>
<tr><td>Cantitatea
<td><input type="text" name="T1" size="7" value="0.0" onChange="parent.validate(this,0,20);">
<tr><td><input type="button" value="Livreaza" onClick="parent.trimite();">
</table>
</center>
</form>
</body>
</html>

//RAPORT.HTML
<html>
<head>
</head>
<body>
<center><p><b>RAPORT LIVRARI REZERVOR R1 </b></p></center>
<center><table border=1 bgcolor=#efefff><tr>
<td><b>ZIUA</b><td><b>REZERVOR R1</b></td></tr>
<tr><td>Luni</td><td>0</td></tr>
 <tr><td>Marti</td><td>0</td></tr>
<tr><td>Miercuri</td><td>0</td></tr>
<tr><td>Joi</td><td>0</td></tr>
<tr><td>Vineri</td><td>0</td></tr>
</table></center><p><p>
<center>
  <p><font color=green size=+1>Total Livrari:0</font> <br>
  <font color=green size=+1>Media Zilnica:0 </font>
  <br><font color=green>Livrarea maxima: 0 s-a facut in ziua de Luni </font>
<br><font color=green>Livrarea minima: 0 s-a facut in ziua de luni </font>
</center>
</html>
```

Figura 10.11 *(continuare)*

```
//SITUATIE.HTML

<html>
<body>
<center>
<h3>
Afiseaza Livrari pentru Rezervoarele:
</h3>
<map name="rezervorMap">
  <area name="R1" coords="16,18,91,125" href="javascript:parent.afispartial(0)"
   onMouseOver="window.status='Rezervorul R1';return true"
    onMouseOut="window.status=' ';return true">
  <area name="R2" coords="105,18,180,125" href="javascript:parent.afispartial(1)"
onMouseOver="self.status='Rezervorul R2';return true"
onMouseOut="self.status=' ';return true">
  <area name="R3" coords="195,18,272,125" href="javascript:parent.afispartial(2)"
onMouseOver="window.status='Rezervorul R3';return true"
onMouseOut="window.status=' ';return true">
</map>
<img src="rezervoare.jpg" align="top" usemap="#rezervorMap">
</body>
</html>
```

Figura 10.11 *(continuare)*

Vizualizați documentul într-un browser şi testați script-ul (vezi EXEMPLUL 9 JAVASCRIPT). În figura 10.12 se prezintă rezultatele execuției programului JavaScript pentru un set de date (Rezervorul 1).

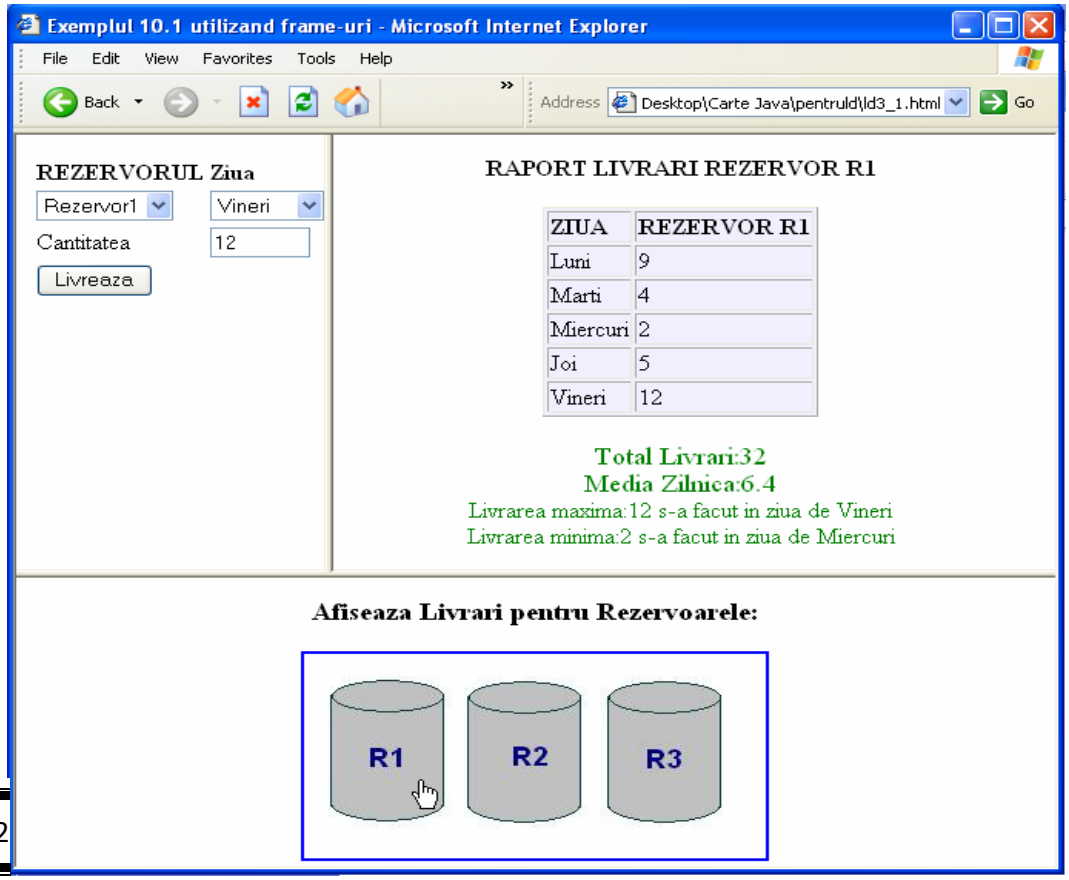


Figura 10.12

Obiectul Layer

Obiectul `Layer` este un fiu al obiectului `Document`.

Obiectul `Layer` permite limbajului JavaScript să acceseze *straturile* în interiorul documentelor.

Remarcă. Pentru mai multe informaţii privind straturile şi modul în care acestea funcţionează în browser, consultaţi lucrările:

- ✓ *Liviu Dumitraşcu, Dreamweaver MX, Editura Universităţii din Ploieşti, 2003.*
- ✓ *Richard Wagner, R.Allen Wyke, JavaScript, Editura Teora, Traducere Cora Rădulescu şi Dan Pavelescu, 2002.*

Fişa obiectului `Layer` este prezentată în figura 10.13.

Fişa obiectului Layer

Obiectul părinte	`Document`
Proprietăţi	`above, background, bellow, bgColor, clip.bottom, clip.height, clip.left, clip.right, clip.top, clip.width, document, left, name, pageX, pageY, parentLayer, siblingAbove, siblingBellow, src, top, visibility, window, zIndex`
Metode	`load(), moveAbove(), moveBellow(), moveBy(), moveTo(), moveToAbsolute(), releaseEvents(), resizeBy(), resizeTo(), routeEvent()`
Gestionarii de evenimente:	-

Figura 10.13

Matricea Layers[]

Matricea `layers[]` (`document.layers[]`) conţine setul de straturi (*layers*, în limba engleză) reprezentate în pagina (X)HTML prin tag-urile `<div>` sau `<layer>`.

Matricea `layers[]` este recunoscută numai de browser-ul Netscape.

Remarci:

- ✓ În Netscape puteţi accesa proprietăţile unui strat cu ajutorul matricii `layers[]`. Numele stratului poate fi specificat prin atributul `id` sau `name` ale tag-ului `div`.
- ✓ În Netscape puteţi accesa proprietăţile unui strat imediat după instrucţiunea `document.layers['numestrat']`.

Matricea all[]

Matricea `all[]` conţine setul de straturi ale unui document (X)HTML recunoscute de Internet Explorer. Valoarea pe care o transferaţi pentru a

specifica stratul pe care doriţi să îl accesaţi nu este păstrată în atributul `name` al tag-ului `<div>` ci în atributul `<id>` al aceluiaşi tag.

Remarcă. În Internet Explorer puteţi accesa proprietăţile unui strat prin intermediul matricii `style`, ceea ce înseamnă că sintaxa pe care o veţi folosi va fi: `document.all['numestrat'].style`.

Corespondenţa între atributele HTML ale straturilor şi proprietăţile JavaScript

Un „layer" este un obiect plan rectangular care conţine mai multe proprietăţi: dimensiuni, culoare, conţinut şi vizibilitate. O pagină Web poate conţine mai multe straturi care se pot suprapune precum straturile, parţial sau total.

Straturile sunt de regulă obiecte statice, dar ele pot fi modificate şi deplasate cu ajutorul unui script. Natural, limbajul de script ales de Netscape este JavaScript întrucât el este creatorul limbajului.

Remarci:

- ✓ Straturile au fost implementate de Netscape începând cu versiunea 4 a browser-ului Netscape Navigator.
- ✓ Pentru gestiunea straturilor, Netscape a creat trei noi tag-uri: `<LAYER> ... </LAYER>`; `<ILAYER> ... </ILAYER>` şi `<NOLAYER> ... </NOLAYER>`. Cel mai important este de departe primul tag. Al doilea este o formă „inline" iar cel de-al treilea joacă acelaşi rol în raport cu elementul LAYER ca NOFRAMES în raport cu FRAMESET.

Elementul LAYER conţine 13 proprietăţi, dintre care cea mai mare parte au valori implicite. Înainte de a trece la corespondenţa: `atribute HTML – propriet••i JavaScript` vom prezenta un exemplu de declarare a unui strat (figura 10.14).

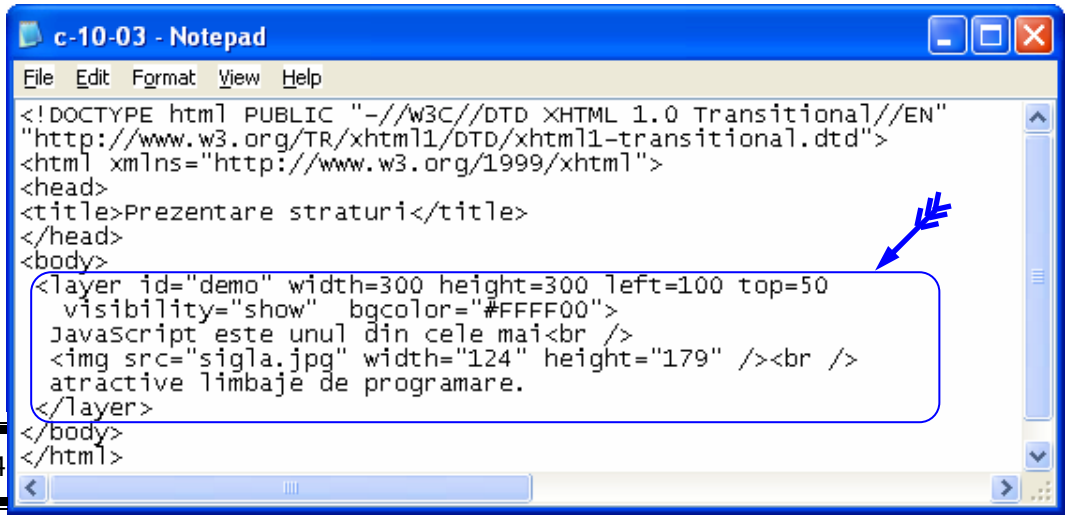


Figura 10.14

În figura 10.15 este prezentat stratul afişat de Netscape.

Figura 10.15

Pentru a putea modifica atributele elementului `LAYER` în vederea dinamizării unei pagini Web va trebui să cunoaştem corespondenţa dintre atributele HTML ale straturilor şi proprietăţile JavaScript (vezi figura 10.16).

	Atribut HTML	***Proprietăţi JavaScript***
	NAME	`name`
	Numele stratului (identic cu `id`). Nemodificabil.	
	ID	`id`
	Numele stratului (identic cu `name`). Nemodificabil.	
	LEFT	`left`
	Coordonata `X` a stratului.	
	TOP	`top`
	Coordonata `Y` a stratului.	
	PAGEX	`pagex`
	Coordonata `X` a stratului faţă de document.	
	PAGEY	`pagey`
	Coordonata `Y` a stratului faţă de document.	
	Z-INDEX	`zindex`
	Ordinea – z a stratului în raport cu fiii săi.	
	VISIBILITY	`visibility`
	Starea de vizibilitate a stratului.	

Figura 10.16

BACKGROUND	background
Imaginea de fundal a stratului.	
BGCOLOR	bgColor
Culoarea de fundal a stratului.	
PARENTLAYER	parentLayer
Numele obiectului care conţine stratul curent.	
SRC	src
URL-ul fişierului care reprezintă conţinutul stratului.	
CLIP	clip.top, clip.left, clip.right, clip.bottom, clip.width, clip.height
Proprietăţile suprafeţei decupate a stratului.	
ABOVE	above
Numele elementului HTML de deasupra stratului curent.	
BELOW	below
Numele elementului HTML de dedesubtul stratului curent.	

Figura 10.16 *(continuare)*

Remarcă. Pentru o mai bună lizibilitate am procedat la scrierea numelor atributelor HTML cu majuscule, dar ... nu este obligatoriu. În schimb, numele proprietăţilor JavaScript se scriu întotdeauna cu minuscule!

Metodele obiectului Layer

Metodele obiectului `Layer` sunt prezentate în detaliu în figura 10.17.

Metodă	***Sintaxă***
load()	document.layers[].load()
Încarcă o nouă adresă URL.	
moveAbove()	document.layers[].moveAbove()
Deplasează stratul deasupra altui strat.	
moveBelow()	document.layers[].moveBelow()
Deplasează stratul dedesubtul altui strat.	
moveBy()	document.layers[].moveBy()
Deplasează stratul într-o poziţie specifică.	
moveTo()	document.layers[].moveTo()
Deplasează colţul din stânga-sus al ferestrei la coordonatele specificate ale ecranului.	

Figura 10.17

`moveToAbsolute()`	`document.layers[].moveToAbsolute()`
Modifică poziţia stratului în pagină, conform coordonatelor specificate în pixeli.	
`releaseEvents()`	`document.layers[].releaseEvents()`
Stabileşte ca stratul să elibereze evenimentele capturate de tipul specificat.	
`resizeBy()`	`document.layers[].resizeBy()`
Redimensionează stratul cu valorile de înălţime şi lăţime specificate.	
`routeEvent()`	`document.layers[].routeEvent()`
Transferă un eveniment capturat prin ierarhia normală a evenimentelor.	

Figura 10.17 *(continuare)*

Aplicaţie

❑ Analizaţi documentul XHTML din figura 10.18 în care s-a inserat un script care conţine metoda `moveBy()`.

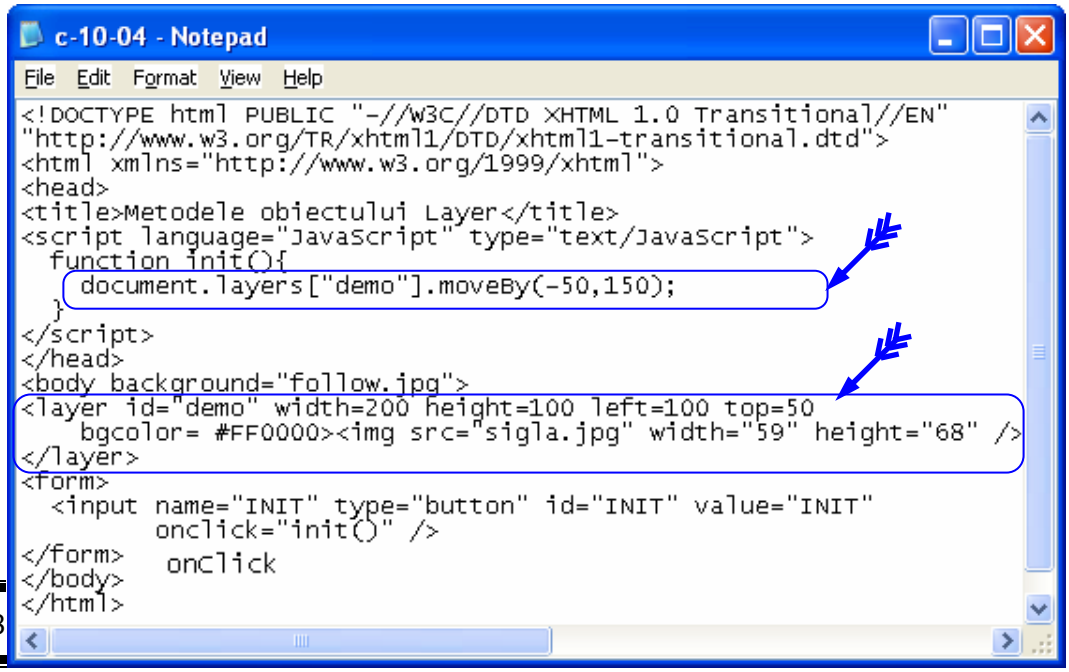


Figura 10.18

În figura 10.19 este prezentat rezultatul execuţiei programului JavaScript în navigatorul Netscape.

Figura 10.19

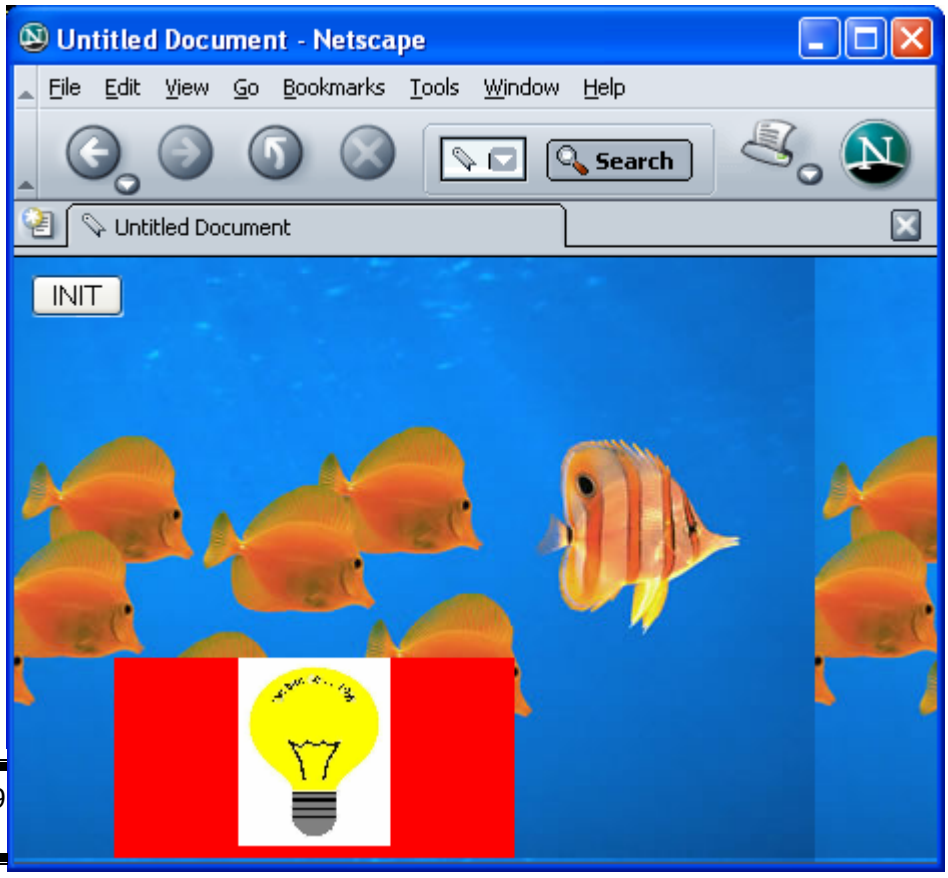


Figura 10.19 *(continuare)*

EXEMPLUL 10.2 JAVASCRIPT

❑ Formularea problemei

Vom aborda şi în cadrul acestei conversații aceeaşi problemă pe care am abordat-o în conversația precedentă, cu singura deosebire că de data acesta vom folosi pentru afişarea rezultatelor obiectul `Layer`.

Introducerea valorilor pentru livrări se realizează printr-o singură zonă de text, iar selecția rezervorului şi a zilei din lista de opțiuni.

❑ Specificații de programare

În figura 10.20 este prezentat ecranul (intrare/ieşire) în care se afişează pagina Web. Tabela de variabile este prezentată în figura 10.21.

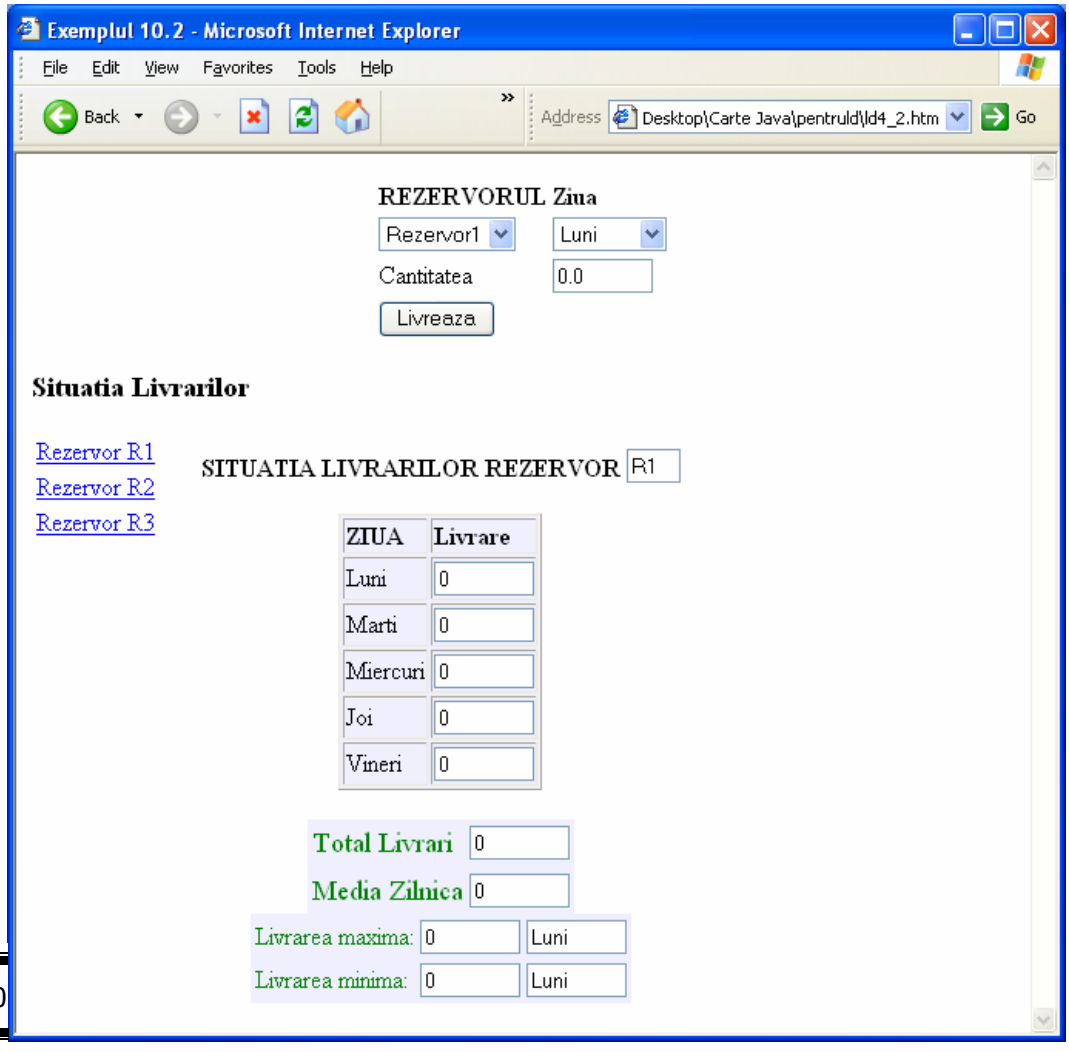


Figura 10.20

Tabela de variabile

Variabile de intrare	*Variabile de stare*	*Variabile de ieşire*
Rezervor: (obiect) lista simpla de selecție pentru cele trei rezervoare Zile: (obiect) lista simplă de opțiuni pentru selecția zilei din săptămână T1: (obiect) zona de editare în care se va introduce valoarea livrărilor a: (matrice de numere reale) păstrează valorile livrărilor pe zile şi rezervoare	s: (real) folosit pentru calculul sumelor parțiale pe zile rval: (logic) precizează dacă valoarea introdusă în zona de editare este validă sau nu x: (real) valoarea reală a textului introdus în zona de editare Z: (vector) numele zilelor săptămânii jmax,jmin: (numere întregi) păstrează indicii livrărilor maxime şi minime din matricea a	f2: (obiect) formular conținut în layer-ul R1 în care se vor afişa rezultatele pentru un rezervor med: (real) folosit pentru calculul mediei zilnice pe rezervor max,min: (numere reale) păstrează valoarea maximă şi minimă pentru livrări rnume,Z1, Z2, Z3, Z4, Z5, Tot, Med, emax, emin, zmax, zmin: (obiect) zone text pentru afişarea rezultatelor

Figura 10.21

❑ Documentația de proiectare

Pseudocodul pentru EXEMPLUL 10.2 JAVASCRIPT este prezentat în figura 10.22.

Pseudocodul

```
EXEMPLUL10.2  BEGIN
                    //initializeaza vectorul Z cu numele zilelor
                    z=luni,marti,miercuri,joi vineri
                    //aloca spatiu de memorie si
                    //initializeaza matricea livrarilor
FORI                FOR(i=0;i<3;i++)
FORJ                        FOR(j=0;j<5;j++)
                                a[i,j]=0.0
FORJ                        ENDFOR
FORI                ENDFOR
                    Afiseaza pagina ce contine layer R2
                    //raspunde la evenimentele generate de butonul Livreaza
IFL                 IF(se apasa butonul Livreaza)
                            DO livreaza
IFL                 ENDIF
                    //raspunde la evenimentele generate zonele reactive
IFAFIS              IF(cursor mouse pe zona unui rezervor)
                            DO afispartial(rezervor)
IFAFIS              ENDIF
                    //raspunde la evenimentele generate de zona de editare T1
IFVALID             IF(se paraseste zona T1)
                            DO valideaza(T1,0,20)
IFVALID             ENDIF
EXEMPLUL10.2  END
AFISPARTIAL   BEGIN
                    Date intrare:
                    rez-indicele rezervorului
                    // calculeaza total livrari
                    s=0
FORS                FOR(j=0;j<5;j++)
                            s=s+a[rez,j]
FORS                ENDFOR
```

Figura 10.22

```
                        med=s/5
                        // determinarea maximului si minimului
                        max=a[rez,0]
                        min=a[rez,0]
                        jmax=0
                        jmin=0
FORMAXMIN               FOR(j=0;j<5;j++)
IFMAX                           IF(max<a[rez,j])
                                        max=a[rez,j]
                                        jmax=j
IFMAX                           ENDIF
IFMIN                           IF(min>a[i,j])
                                        min=a[rez,j]
                                        jmin=j
IFMIN                           ENDIF
FORMAXMIN               ENDFOR
                        //afisare rezultate
                        //depune rezultate in zonele din layerul R1
                        k=rez+1
                        f2.rnume.value="R"+k;
                        f2.Z1.value=a[rez][0];
                        f2.Z2.value=a[rez][1];
                        f2.Z3.value=a[rez][2];
                        f2.Z4.value=a[rez][3];
                        f2.Z5.value=a[rez][4];
                        f2.Tot.value=s;
                        f2.Med.value=trunchiaza(med);
                        f2.emax.value=max;
                        f2.zmax.value=Z[jmax];
                        f2.emin.value=min;
                        f2.zmin.value=Z[jmin];
AFISPARTIAL     END

                //Afiseaza layerul cu rezultate
ARATA           BEGIN
                        R1.style.visibility = "visible"
ARATA           END

                // Ascunde layerul cu rezultate
ASCUNDE         BEGIN
                        R1.style.visibility = "hidden"
ASCUNDE         END
```

Figura 10.22 *(continuare)*

❑ Codificarea în limbajul JavaScript

Documentul complet (X)HTML este prezentat în figura 10.23.

```
<html>
<head>
<title>Exemplul 10.2</title>
<script language=”JavaScript”>
<!---
var Z = new Array(”Luni”,”Marti”,”Miercuri”,”Joi”,”Vineri”);
function trunchiaza(x) {
    var s=””+x;
     i=s.indexOf(„.”);
     if(i!=-1){
       s=s.substring(0,i+3);
       }
```

Figura 10.23

```
      return s;
      }

function validate(item, min, max) {
   var rVal = false;
  var x=parseFloat(item.value);
     if(isNaN(x))
       alert(„Valoare gresita pentru cantitate!”);
      else
     if (x < min)
    alert(„Valoare gresita pentru cantitate!Valoarea trebuie >” + min);
      else if (x> max)
       alert(„Valoare gresita pentru cantitate!  Valoarea trebuie sa fie < „ + max);
          else
      rVal = true;
   return rVal;
}
a=new Array(3);
a[0]=new Array(5);
a[1]=new Array(5);
a[2]=new Array(5);
for(i=0;i<3;i++)
 for(j=0;j<5;j++)
      a[i][j]=0.0;
function trimite() {
 var ir=f1.Rezervor.selectedIndex;
 var z=f1.Zile.selectedIndex;
   a[ir][z]+=parseFloat(f1.T1.value);
}
function afispartial(rez) {
var i,j,k;
//CALCUL TOTAL LIVRARI
 s=0.0;
 for(j=0;j<5;j++)
   s+=a[rez][j]
//calcul medie pe zi
med=s/5;
//calcul max si min
    max=a[rez][0];
    min=a[rez][0];
      jmax=0;jmin=0;
       for(j=0;j<5;j++){
         if(max<a[rez][j]){max=a[rez][j];jmax=j;}
         if(min>a[rez][j]){min=a[rez][j];jmin=j;}
   }
//AFISARE REZULTATE

k=rez+1;
f2.rnume.value="R"+k;
document.f2.Z1.value=a[rez][0];
 document.f2.Z2.value=a[rez][1];
document.f2.Z3.value=a[rez][2];
 document.f2.Z4.value=a[rez][3];
```

Figura 10.23 *(continuare)*

```
document.f2.Z5.value=a[rez][4];
 document.f2.Tot.value=s;
 document.f2.Med.value=trunchiaza(med);
 document.f2.emax.value=max;
 document.f2.zmax.value=Z[jmax];
 document.f2.emin.value=min;
 document.f2.zmin.value=Z[jmin];
}
function arata() {
document.all['R1'].style.visibility = "visible";
}
function ascunde() {
  document.all['R1'].style.visibility = "hidden";
}
// -->
</script>
</head>

<body>
<center>
<form name="f1">
<table border=0>
<tr><td><b>REZERVORUL</b><td><b>Ziua</b>
<tr><td><select size="1" name="Rezervor">
  <option selected>Rezervor1</option>
  <option>Rezervor2</option>
  <option>Rezervor3</option>
  </select>
<td>
<select size="1" name="Zile">
  <option>Luni</option>
  <option>Marti</option>
  <option>Miercuri</option>
  <option>Joi</option>
  <option>Vineri</option>
  </select>
<tr>
<td>Cantitatea
<td><input type="text" name="T1" size="7" value="0.0" onChange="validate(this,0,20);">
<tr>
<td><input type="button" value="Livreaza" onClick="trimite();">
</table>
</center>
</form>
<h3>Situatia Livrarilor</h3>
<table>
<tr><td><a href="javascript:void(0)"  onMouseOver="afispartial(0);arata();"
  onMouseOut="ascunde('R1');"> Rezervor R1</a>
<tr><td><a href="javascript:void(0)"
  onMouseOver="afispartial(1);arata();"
  onMouseOut="ascunde();">Rezervor R2
<tr><td><a href="javascript:void(0)" onMouseOver="afispartial(2);arata();"
  onMouseOut="ascunde();">Rezervor R3
```

Figura 10.23 *(continuare)*

```
</tr>
</table>
<div id="r1" style="position: absolute;top: 120px;z-index: 2; visibility: hidden; top:190; left:
120px;">
<form name = "f2">
<center><p><b>SITUATIA LIVRARILOR REZERVOR
<input type="text" name="rnume" size="2"></b></p></center>
<center><table border=1 bgcolor=#efefff><tr>
<td><b>ZIUA</b><td><b>Livrare</b></td></tr>
<tr><td>Luni</td><td><input type="text" name="Z1" size="7" value="0.0"></td></tr>
 <tr><td>Marti</td><td><input type="text" name="Z2" size="7" value="0.0"></td></tr>
<tr><td>Miercuri</td><td><input type="text" name="Z3" size="7" value="0.0"></td></tr>
<tr><td>Joi</td><td><input type="text" name="Z4" size="7" value="0.0"></td></tr>
<tr><td>Vineri</td><td><input type="text" name="Z5" size="7" value="0.0"></td></tr>
</table></center><p><p>
<center>
<table bgcolor=#efefff>
  <tr><td><font color=green size=+1>Total Livrari</font><td><input type="text" name="Tot"
size="7" value="0.0"><tr>
  <tr><td><font color=green size=+1>Media Zilnica</font><td><input type="text" name="Med"
size="7" value="0.0"> </table>
<table bgcolor=#efefff>
  <tr><td><font color=green>Livrarea maxima:</font><td> <input type="text" name="emax"
size="7" value="0.0"> <td> <input type="text" name="zmax" size="7" value="Luni">
<tr><td><font color=green>Livrarea minima:</font> <td><input type="text" name="emin"
size="7" value="0.0"> <td> <input type="text" name="zmin" size="7" value="Luni">
 </table>
</form>
</div>
</body>
</html>
```

Figura 10.23 *(continuare)*

Vizualizaţi documentul într-un browser şi testaţi script-ul numai în Internet Explorer. Se introduc livrările; se poziţionează mouse-ul deasupra legăturii: Rezervor R1/ Rezervor R2/ Rezervor R3.

În figura 10.24 se prezintă rezultatele execuţiei programului JavaScript pentru un set de date.

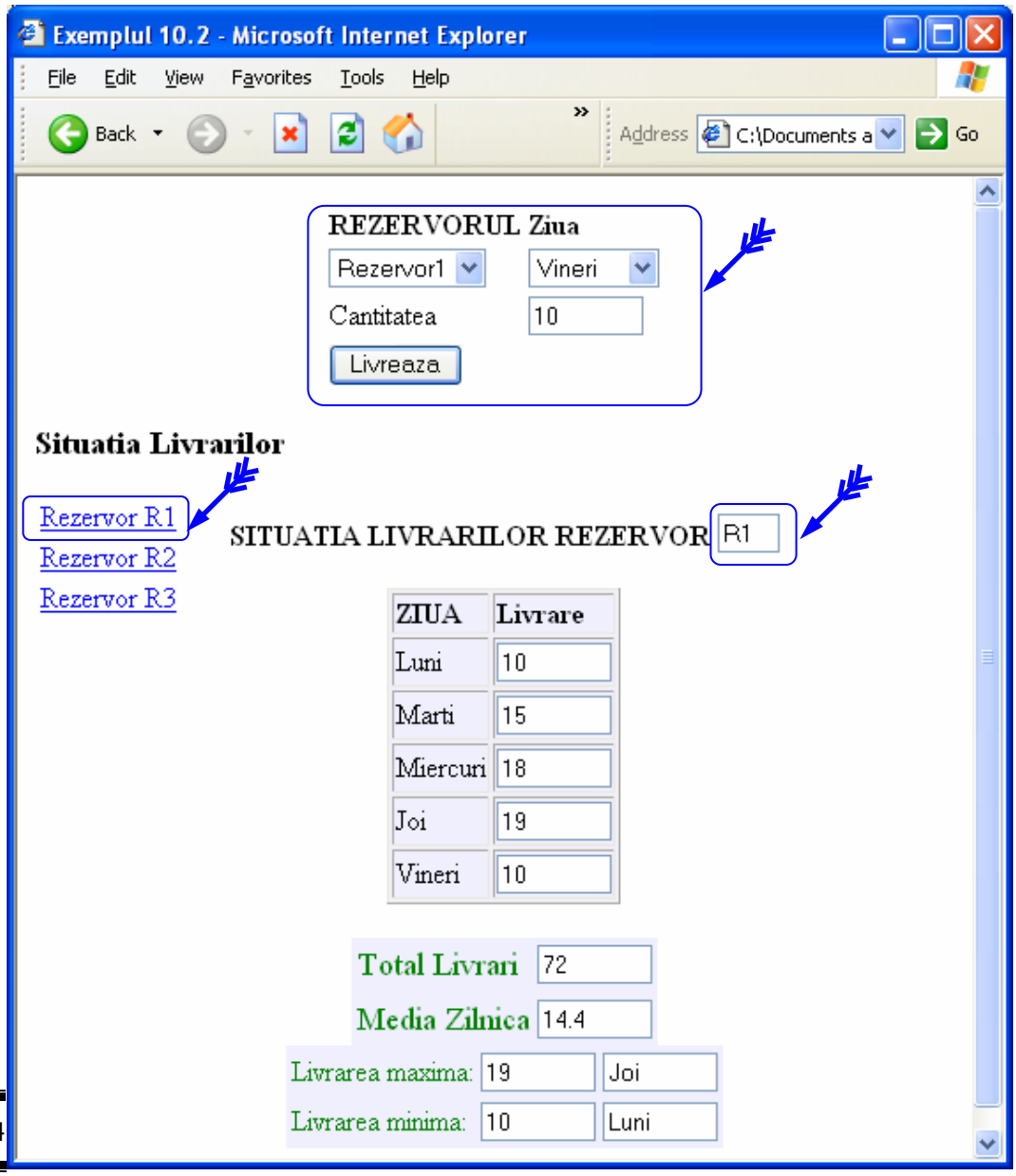
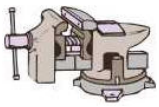

Exemplul 10.2 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address C:\Documents a
Go
REZERVORUL Ziua
Rezervor1
Vineri
Cantitatea
10
Livreaza
Situatia Livrarilor
Rezervor R1
Rezervor R2
Rezervor R3
SITUATIA LIVRARILOR REZERVOR
R1
ZIUA
Livrare
Luni
10
Marti
15
Miercuri
18
Joi
19
Vineri
10
Total Livrari
72
Media Zilnica
14.4
Livrarea maxima:
19
Joi
Livrarea minima:
10
Luni

Figura 10.24

Testați-vă cunoștințele



1. Când utilizaţi obiectele `Frame`?
2. Care este rolul obiectului `Layer`?
3. Care sunt proprietăţile obiectului `Layer`?
4. Care sunt metodele obiectului `Layer`?

Vizitați site-urile

- ✓ http://Javascript.internet.com/messages/description-layer.htm;
- ✓ http://Javascript.internet.com/bgeffects/floating-transparent-slideshow.html
- ✓ http://Javascript.internet.com/bgeffects/write-layer.html
- ✓ http://www.dannyg.com/examples/ol2/index.htm
- ✓ http://www.dannyg.com/examples/dh2/index.htm

Conversația 11

Depanarea aplicațiilor JavaScript

În această conversație:

- *Tipuri de erori JavaScript. Aplicaţii*
- *Tehnici de depanare a script-urilor*
- *Instrumente pentru depanarea script-urilor. Aplicaţii*
- *Instrucţiunile throw şi try … catch. Aplicaţii*
- *Temă*

Evitați erorile frecvente JavaScript

Trebuie să recunoaşteţi, chiar dacă nu vă face plăcere că, nu de puţine ori, dar mai ales atunci când aţi mărit gradul de complexitate al aplicaţiilor JavaScript aţi făcut şi … greşeli! Este normal să fie aşa!

În această conversaţie, ne vom apropia cu respect faţă de erorile curente JavaScript, vom încerca o clasificare a acestora urmând ca apoi să precizăm regulile pentru evitarea acestor erori frecvente JavaScript.

De asemenea, nu vor lipsi din prezentarea noastră nici tehnicile şi instrumentele de depanare folosite curent în activitatea de testare şi depanare a programelor JavaScript.

Tipuri de erori JavaScript

Fiecare dintre noi poate comite erori particulare la scrierea unui program JavaScript, dar foarte multe dintre acestea sunt curente.

Erorile JavaScript sunt identificate de browser la încărcarea paginii Web. Ele pot fi clasificate în următoarele categorii:

- ✓ erori de sintaxă;
- ✓ erori de logică;
- ✓ erori generate de incompatibilitatea între navigatoare.

Erori de sintaxă

Erorile de sintaxă JavaScript sunt erorile cele mai frecvente şi cele mai uşor de corectat. De cele mai multe ori, interpretorul JavaScript identifică corect sursa erorilor în programele pe care le-aţi realizat, dar … sunt şi cazuri când lucrurile nu stau chiar aşa!

În general, erorile de sintaxă provin din:

- ✓ greşeli de tastare
  - confuzia între majuscule şi minuscule;
  - inversarea literelor;
- ✓ greşeli de punctuaţie
  - tag-uri şi paranteze orfeline;
  - ghilimele şi apostrofuri plasate greşit;
- ✓ greşeli de plasare a instrucţiunilor JavaScript
- ✓ confuzia între şirurile de caractere şi numere
  - numerele sunt tratate ca şiruri de caractere;
  - şirurile de caractere sunt tratate ca numere.

Aplicaţie

❑ Următoarele script-uri conţin erori de sintaxă, după cum urmează:

- ✓ greşeli de tastare, greşeli de punctuaţie
  - figura 11.1, figura 11.2, figura 11.3, figura 11.4, figura 11.5, figura 11.6, figura 11.7, figura 11.8, figura 11.9
- ✓ greşeli de plasare a instrucţiunilor JavaScript
  - figura 11.10, figura 11.11, figura 11.12, figura 11.13, figura 11.14
- ✓ greşeli de tratare a şirurilor de caractere şi a numerelor
  - figura 11.15, figura 11.16, figura 11.17

Identificaţi şi corectaţi erorile de sintaxă din toate aceste script-uri.

Figura 11.1

```
<script>
        document.write (”<font size=’+5’>”);
        document.write (”La revedere!”
        x=8 y=3 z=120;
        document.write (”La revedere!”+<br>);
</script>
```

Figura 11.2

```
<script>
        DOCUMENT.write (”Să nu uitați să fiți fericiți!”);
        Document.Write (”Să nu uitați să fiți fericiți!”);
        Document.WRITE (”Să nu uitați să fiți fericiți!”);
</script>
```

Figura 11.3

```
<script>
        ”Salut”=şir;
var ”Salut”=şir_1;
</script>
```

Figura 11.4

```
<script>
        onClick=”alert(calculează);”
        onClick=alert(’Afişează’);
</script>
```

Figura 11.5

```
<script>
        a=new Array;
        b=Array(5);
        c=[2,5,7;
</script>
```

Figura 11.6

```
<script>
        if (a==10);
                alert (”a=10”);
        if (b==5
                alert (”b=5”);
        if (c==7),
                alert (”c=7”);
        if (d=8)
                alert (”d=8”);
        if (m=7) this
                alert (”m=7”);
```

```
<script>
        if (c=3)
                alert (”c=3”)
        else; document.write (”c!=3”);
        if (d=7) {
                alert (”d=7”)
        }
        else (document.write (”EROARE”);
        if (r=33)
                alert (”FELICITĂRI!”)
        else ”document.write (’EROARE’)”;
</script>
```

Figura 11.7

```
<script>
        while (i<=7);{}
                document.writeln(i);
                ++i;
        while (j<=9; document.writeln(j) {
                ++j;
                }
</script>
```

Figura 11.8

```
<script>
        do while (i<7) {
          }     alert (”La revedere!”);
        }
        do { while (i<10)
                alert (”Bucla do … while!”);
        }
do {
        alert (”FATALITATE!”);
        while (i<7)
}
</script>
```

Figura 11.9

```
<script>
        for (var i=0, i<13, ++i)
                alert (”La mulți ani frumoşi!”);
        for {
                alert (”La revedere!”);
        } while (var i=0; i<13; ++i)
        for (i=0; i<7);
                alert (”La revedere!”);
                ++i;
</script>
```

Figura 11.10

```
<script>
s=O; //inițializare cu litera O!
for (i=0; i<=10; i++) {
        s=s+i;
}
</script>
```

Figura 11.11

```
<script>
        s=0;
        for (i=0; I<=10; i++) {
                s=s+i;
        }
</script>
}
```

Figura 11.12

```
<script>
        for (i=0; i<=10; i++) {
                p_2=i*i;
                p_3=i*i*i;
        document.write (p_2);
        document.write (p_3);
```

Figura 11.13

```
<script>
        s=0;
        for (i=0; i<=10; ++i++) {
        s=s+i;
        }
        write(s);
</script>
```

Figura 11.14

```
<script>
function demo (Valoare) {
        document.write (valoare.bold()+"<br/>")
}
</script>
</head>
<body>
<form>
<input type="button" value="BOLD"
onClick="demo(3.14)'>
</form>
</body>
</html>
```

Figura 11.15

```
<html>
<head>
<title> aplicație </title>
<script>
function test (valoare) {
        alert (Valoare)
        }
</script>
function adunare (unu, doi) {
        return unu+doi
        }
</head>
</body>
</html>
```

Figura 11.16

```
<script>
function TVA (valoare) {
        return valoare*0.19
        }
</script>
</head>
<body>
<form>
<input type=”button” value=”calcul TVA”
onClick=’alert (”TVA:”+TVA(”treisprezece”))>”
</form>
</body>
</html>
```

Figura 11.17

Erori de logică

Erorile de logică se generează atunci când nu se obţin rezultatele scontate! Deşi codul sursă JavaScript este corect din punct de vedere sintactic, deşi nu sunt generate erori în timpul execuţiei programului (apeluri de funcţii incorecte; atribuiri de valori pentru variabile nedeclarate; imposibilităţi aritmetice – împărţire la zero etc.), totuşi programul conţine erori de logică (semantice)!

Identificarea erorilor de logică constituie o piatră grea de încercare pentru începători, dar nu trebuie să disperaţi!

Foarte multe erori de logică sunt generate de o analiză superficială a aplicaţiei, o proiectare defectuoasă a programului JavaScript, şi ca urmare de un cod sursă JavaScript incorect!

Aplicaţie

❑ Identificaţi şi corectaţi erorile de logică din următoarele script-uri (vezi figurile 11.18, 11.19, 11.20, 11.21).

```
<script>
stud=parseInt (prompt(”Introduceți numărul de studenți”, 0));
teste=parseInt (prompt(”Introduceți numărul de teste”, 0));
for (i=1; i<=stud; i++) {
nt=0;
media=0;
for (j=1; j<=teste; j++) {
        nota=parseInt (rezultate[i][j]);
                if(nota) {
                        media+=nota;
                }
}
media=media/nt;
media=Math.flor(media*100)/100;
}
```

Figura 11.18

Remarci:

- ✓ Aplicaţia ilustrează o eroare generată în timpul execuţiei script-ului, datorată de o împărţire la zero:
  media=media/nt
  unde, nt=0.
- ✓ Corectaţi eroarea, introducând în instrucţiunea `if`, instrucţiunea nt++ (figura 11.19).

```
nt=0;
media=0;
for (j=1; j<=teste; j++){
        nota=parseInt (rezultate [i][j]) ;
        if nota {
                media+=nota;
                nt++;
        }
}
media=media/nt;
}
```

Figura 11.19

```
<script>
//Conversatia 3
function rotunjeste (x) {
}
//Calculul mediilor pe zile
//Calculul mediilor pe rezervoare
d=new Array (4);
st=0;
for (i=0; i<3; i++) {
        s=0;
        for (j=0; j<5; j++) {
                s=s+a[i][j];
                st=st+a[i][j];
        }
        d[i]=s/5;
}
        d[3]=st/14;
```

Figura 11.20

Remarci:

- ✓ Eroarea de logică provine de la numitorul fracţiei, din enunţul de atribuire
  d[3]=st/14;
- ✓ Enunţul trebuie corectat, după cum urmează:
  d[3]=st/15;
- ✓ De ce 15? Răspunsul este simplu: 5 zile (luni, marţi, miercuri, joi, vineri) * 3 rezervoare (R1, R2, R3), (vezi Conversaţia 3).

```
<script>
        var Constant="13";
        i=7;
        for (i<=7; i>0; i--) {
                document.write (i+"+"+Constant+"="+
                (i+Constant)+"<br>");
        }
</script>
```

Figura 11.21

Remarci:

- ✓ Rezultatele afişate (vezi figura 11.22) sunt incorecte, chiar dacă execuţia s-a încheiat fără incidente.

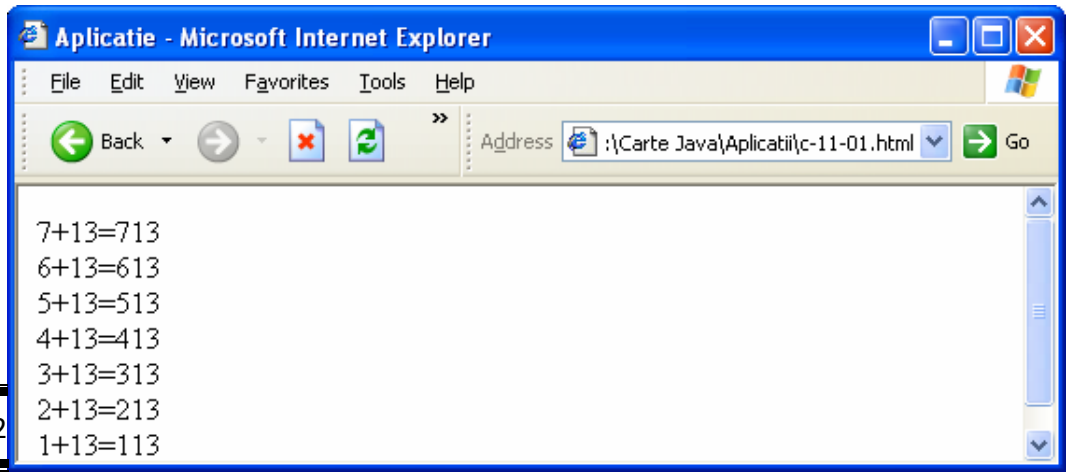


Figura 11.22

- ✓ JavaScript evaluează rezultatul expresiei (i+Constant) ca şir şi concatenează cei doi operanzi.

Erori generate de incompatibilitatea între navigatoare

Nimic nu este mai neplăcut decât să constaţi că după ce ai reuşit să creezi (cu efort!) script-uri pe care le consideri nemaipomenite, să primeşti într-o bună zi un mesaj din partea unui vizitator care să-ţi semnaleze faptul că script-urile tale nu „merg" cu navigatorul acelui vizitator.

Nu uitaţi că există un turn Babel al navigatoarelor!

Morala:

- ✓ testaţi script-urile dumneavoastră cu mai multe browser-e, înainte de a le plasa pe server-ul Web;
- ✓ consultaţi manualele de utilizare a celor mai importante browser-e:
  - http://developer.netscape.com/docs/manuals/jsref/
  - http://msdn.microsoft.com/scripting/jsscripting/jsscript/ default.htm
- ✓ detectaţi tipul de navigator utilizat de către vizitatorii dumneavoastră.

Tehnici de depanare a script-urilor

În vremurile … mai de demult programatorii începeau depanarea prin a desena scheme logice complicate pentru a urmări secvenţial, logica programului.

În prezent, schemele logice au fost înlocuite cu pseudocodul, care este o combinaţie de JavaScript şi limbaj natural structurat. Acest mod de descriere a unui algoritm s-a dovedit a fi indispensabil atunci când doriţi să localizaţi zona în care ar putea fi eroarea generată.

Utilizaţi comentariile

Pentru a vă crea primele reflexe JavaScript inseraţi cât mai multe comentarii în script-urile dumneavoastră.

Observaţie. Comentariile JavaScript (// sau /* şi */) au fost prezentate în Conversaţia 2).

Pentru a elimina anumite porţiuni din codul JavaScript (până la eliminarea erorilor) utilizaţi comentariile conform procedurii de mai jos [1]:

- ✓ transformaţi în comentariu una sau mai multe linii ale programului;
- ✓ salvaţi programul;
- ✓ reîncărcaţi pagina Web în browser;
- ✓ analizaţi rezultatul (efectul);
- ✓ modificaţi codul sursă sau transformaţi în comentariu mai multe linii de cod;
- ✓ repetaţi această procedură până când aţi eliminat eroarea.

Afişaţi valorile variabilelor

O altă tehnică de depanare frecvent folosită este aceea de a adăuga instrucţiuni JavaScript pentru a putea cunoaşte stările script-ului.

În acest sens, utilizaţi metoda `alert()` pentru a:

- ✓ afişa valorile variabilelor, matricelor şi valorile returnate de funcţii;
- ✓ afişa rezultatele expresiilor.

De asemenea, puteţi afişa mesaje în bara de stare cu ajutorul proprietăţii `status`.

Puteţi de asemenea afişa informaţii din procesul de depanare, într-o altă fereastră a navigatorului sau într-un cadru anume.

În anumite cazuri, folosiţi `document.write`, dar … atenţie (!), această metodă nu funcţionează decât atunci când documentul a fost complet încărcat!

Descompuneţi script-urile complexe în mai multe funcţii

Pentru a mări gradul de reutilizare a codului JavaScript este bine să limitaţi dimensiunea funcţiilor pe care urmează să le creaţi. Cu cât dimensiunile unei funcţii sunt mai mici cu atât mai mult cresc şansele de reutilizare a acesteia în diferite zone ale programului.

Verificaţi documentul (X)HTML

Nu trebuie să uitaţi că JavaScript nu este singurul limbaj pe care trebuie să-l utilizaţi; el interacţionează de obicei cu codul sursă (X)HTML.

Morala este următoarea: *asiguraţi-vă că documentul (X)HTML nu conţine erori! Este foarte simplu de a comite erori în documentul (X)HTML care conţine script-ul. A uita tag-ul de închidere* `</table>` *sau* `</script>` *este o eroare frecventă (X)HTML.*

Deşi (X)HTML nu reprezintă obiectul de studiu al nostru (vezi *L. Dumitraşcu, (X)HTML, Editura Universităţii din Ploieşti, 2003*) este bine să ştiţi că o eroare (X)HTML poate genera erori în programul JavaScript.

Verificaţi codul JavaScript

Atribuire şi egalitate. În JavaScript una din erorile cele mai frecvente comise de către începători este confuzia între operatorul de afectare (=) şi operatorul de egalitate (==). Aceste erori sunt câteodată dificil de identificat în măsura în care ele nu generează întotdeauna un mesaj de eroare.

Dacă nu ştiţi ce operator să folosiţi, amintiţi-vă simplu că "=" serveşte la schimbarea valorii unei variabile, iar "==" serveşte la compararea a două valori.

Iată o instrucţiune eronată (figura 11.23).

Figura 11.23

```
<script>
        if (x=7) alert ("La revedere!");
</script>
```

Această instrucţiune pare logică la prima vedere, dar x=7 va avea ca efect atribuirea *valorii* 7 variabilei x, şi nu compararea celor două. Netscape detectează de cele mai multe ori acest tip de eroare şi afişează un mesaj la consolă. Eroarea inversă (== în loc de =) nu va fi niciodată detectată!

Variabile globale şi locale. O altă eroare frecventă este confuzia între variabilele globale şi locale, atunci când se doreşte de exemplu a utiliza în exteriorul unei funcţii o variabilă care a fost declarată în interiorul unei funcţii.

Remarcă. Diferenţa dintre variabilele locale şi globale este explicată în Conversaţia 2.

Faceţi referiri corecte la obiecte. Nu de puţine ori se fac referiri la obiecte în mod incorect. Este important de a utiliza numele exacte ale obiectelor şi de a numi explicit părinţii unui obiect.

Astfel, este posibil de a ne referi la metoda `window.alert` scriind simplu `alert`. Dar sunt şi cazuri când utilizarea lui `window.alert` este obligatorie.

O altă eroare frecventă constă în neglijarea utilizării numelui obiectului `Document`, scriind de exemplu `write` în loc de `document.write`.

Instrumente pentru depanarea script-urilor

Depanarea codului sursă ocupă un loc deosebit de important în dezvoltarea aplicaţiilor JavaScript.

Dacă aţi verificat absenţa erorilor „clasice" în script-urile dumneavoastră, care totuşi nu funcţionează este momentul să treceţi la depanare, adică la căutarea şi eliminarea erorilor din program, utilizând instrumentele simple de depanare pe care le prezentăm în continuare.

Consola JavaScript

Primul lucru pe care trebuie să-l faceţi atunci când script-ul dumneavoastră nu funcţionează este consultarea mesajelor de erori afişate. În Netscape, începând cu versiunea 4.5, mesajele de eroare nu se afişează direct, ele fiind înregistrate în consola JavaScript.

Iată cum procedăm pentru a accesa consola JavaScript.

1. Introduceţi JavaScript în câmpul `Address` (figura 11.24).

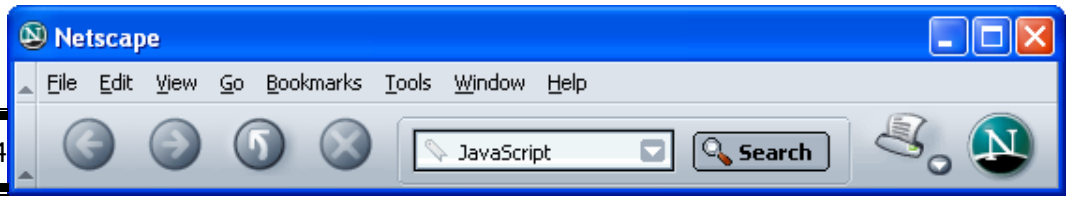


Figura 11.24

sau,

2. Executaţi clic pe `Tools` → `Web Development` → `JavaScript Console` (figura 11.25).

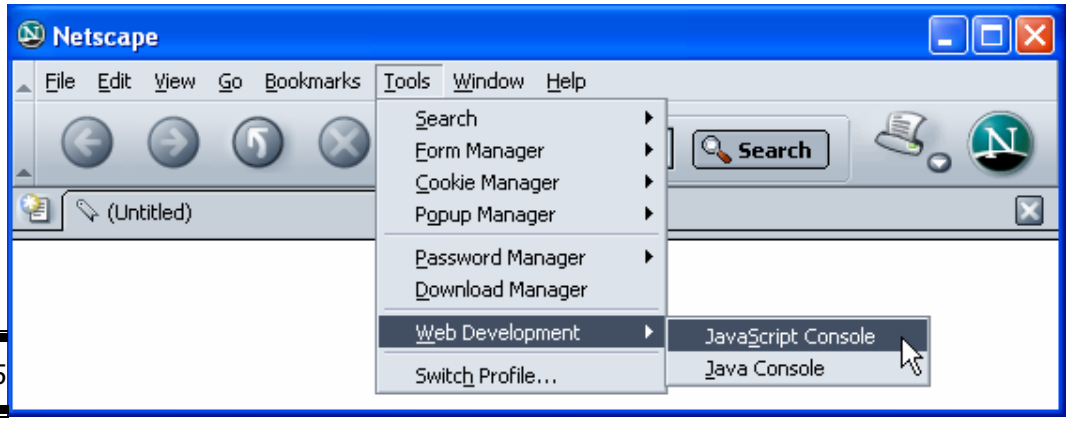


Figura 11.25

Remarcă. Consola JavaScript afişează ultimele mesaje de eroare (figura 11.26).

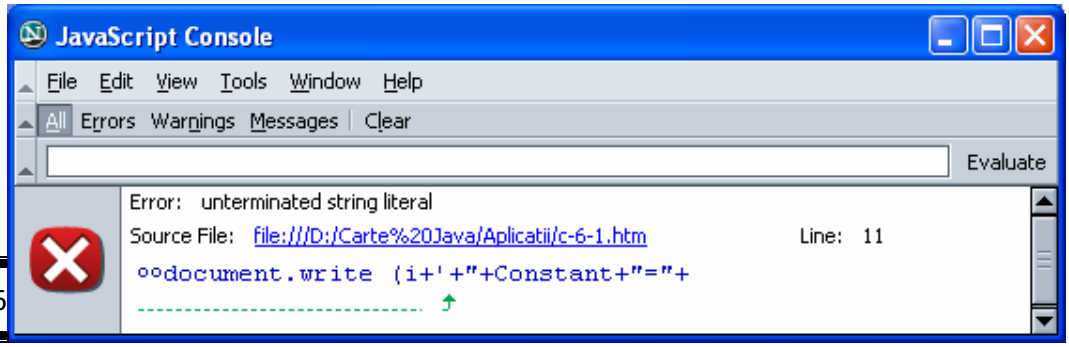


Figura 11.26

Consola JavaScript nu permite numai vizualizarea erorilor, ea permite de asemenea introducerea unei instrucţiuni/expresii, pentru a vedea apoi rezultatul.

Această funcţie este utilă pentru a verifica existenţa erorilor de sintaxă în liniile de cod ale script-ului dumneavoastră.

Afişarea mesajelor de eroare cu Internet Explorer

Internet Explorer şi versiunile mai recente nu afişează în mod implicit mesajele de eroare.

Iată cum procedăm pentru a afişa mesajele de eroare în Internet Explorer.

1. Executaţi clic pe `Tools` → `Internet Options` → `Advanced` (figura 11.27).

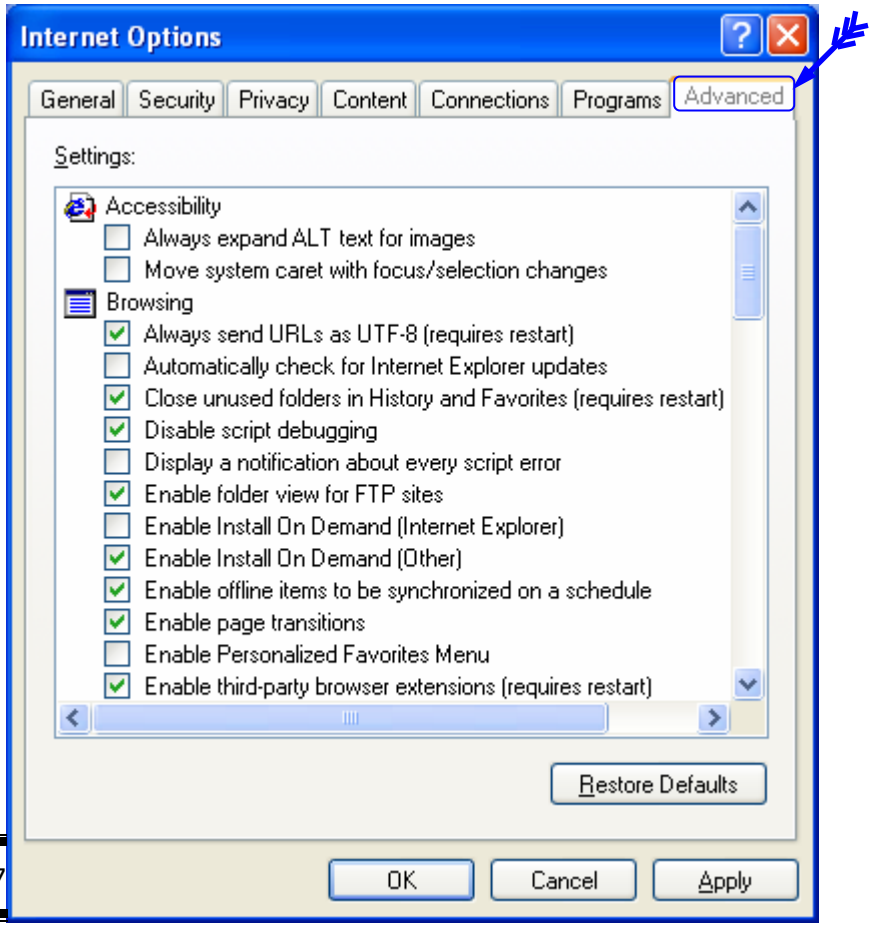


Figura 11.27

2. Dezactivaţi Disable script debugging şi activaţi opţiunea Display a notification about every script error (figura 11.28).

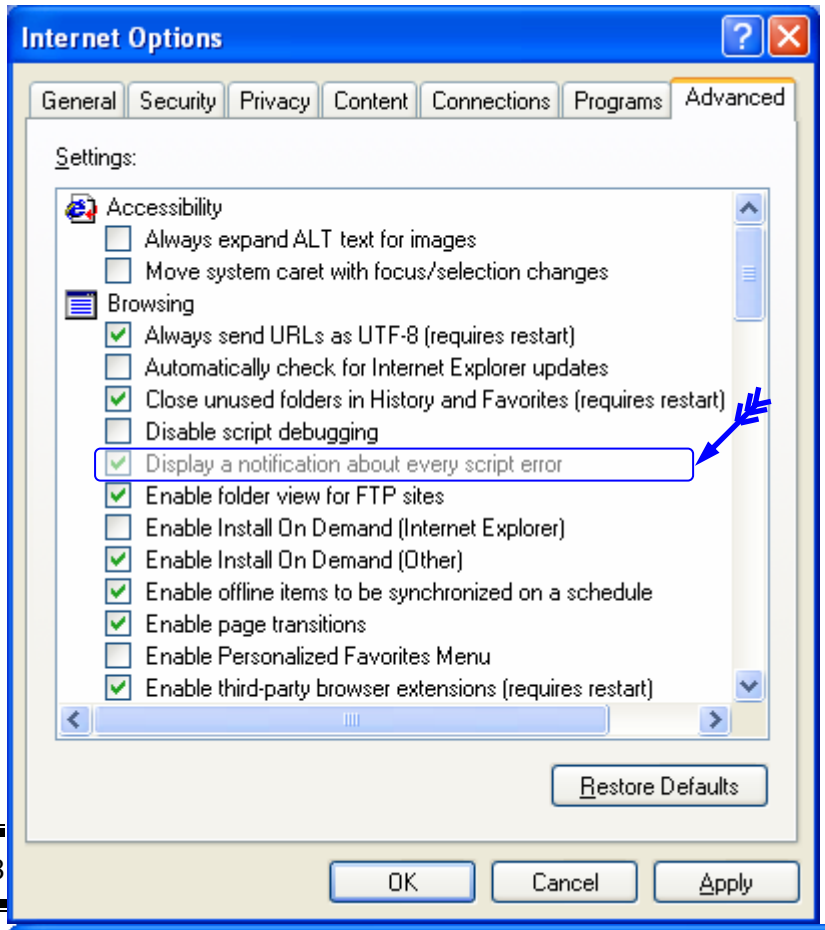


Figura 11.28

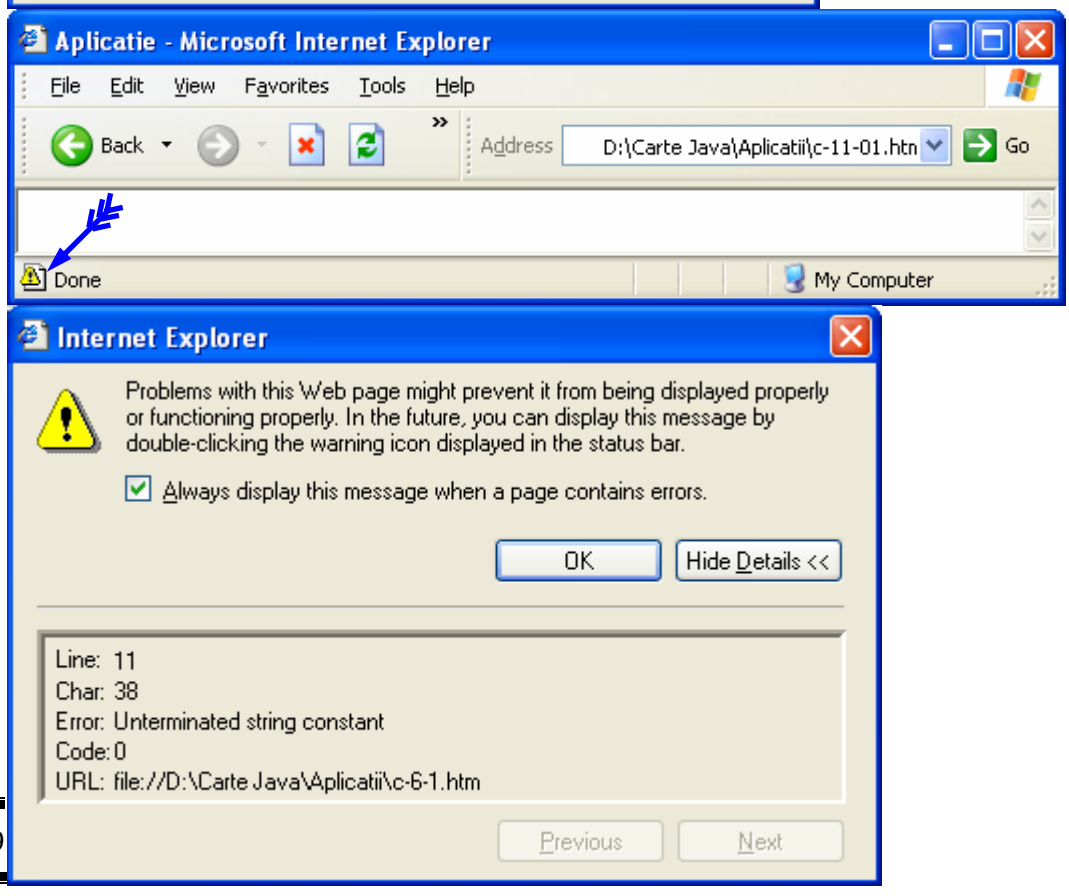


Figura 11.29

Remarci:

- ✓ Dacă nu aţi activat afişarea mesajelor de erori, Internet Explorer afişează în bara de stare un icon atunci când se generează o eroare. Executaţi dublu clic pe acest icon pentru a afişa mesajul de eroare (figura 11.29).
- ✓ După cum aţi putut constata şi Microsoft dispune de un instrument de depanare la fel de eficient ca şi consola Netscape. Indiferent de preferinţele dumneavoastră, testaţi script-urile cu cele două navigatoare.

Alte instrumente de depanare

Mesajele de avertizare (vezi metoda `alert()`) şi puţin … fler vă pot ajuta să găsiţi cu uşurinţă cauza unei erori într-un script simplu dar … aceste metode sunt insuficiente pentru depanarea script-urilor de dimensiuni mari.

În consecinţă, vă recomandăm să utilizaţi unul din instrumentele specializate, prezentate mai jos:

- ✓ **W3C validator**, instrument de depanare a documentelor (X)HTML, care poate fi descărcat gratuit, de la adresa:

  http://validator.W3.org/

  Remarcă. Consultaţi lucrarea *L. Dumitraşcu (X)HTML, Editura Universităţii din Ploieşti 2003*.

- ✓ **Netscape JavaScript Debugger**, instrument de depanare a aplicaţiilor JavaScript, care poate fi descărcat gratuit, de la adresa:

  http://developer.netscape.com/software/jsdebug.html

  sau,

  http://www.mozilla.org/projector/venkman/

  Remarcă. Principalele caracteristici [1] ale instrumentului `Netscape JavaScript Debugger` sunt: fereastra Source View; întrerupere; puncte de întrerupere; parcurgerea pas cu pas a codului sursă; omiterea liniilor din procedurile apelate; parcurgerea liniilor din procedurile apelate; fereastra Console; Object Inspector; fereastra de dialog Error Reporter.

- ✓ **Microsoft Script Debugger (MSSD)**, instrument de depanare a script-urilor, parte integrantă din Internet Explorer, care poate fi descărcat gratuit, de la adresa:

  http://msdn.microsoft.com/library/en-us/sdbug/html/sdbug_1.asp

  Remarcă. Principalele caracteristici [1] ale instrumentului `Microsoft Script Debugger` sunt: imagine dinamică a structurii (X)HTML; integrarea mai multor limbaje (JavaScript, VBScript şi Java din acelaşi document); puncte de întrerupere; parcurgerea pas cu pas a codului sursă; omiterea liniilor din procedurile apelate; parcurgerea liniilor din procedurile apelate; stivă de apeluri integrată; fereastra de expresii evaluate imediat.

- ✓ Editoarele de text şi de (X)HTML sunt foarte bune instrumente pentru procesarea de bază; ele vă pot de asemenea asista în timpul depanării,

afişând numerele de linii şi punând în relief tag-urile corecte cu ajutorul codurilor de culori.

Aplicaţie

❑ Scrieţi un program JavaScript care calculează şi afişează determinantul unei matrice de dimensiune 3*3.

Indicaţie.

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a*(e*i–f*h)–b*(d*i–f*g)+c*(d*h–e*g)$$

Iată cum procedăm pentru a scrie programul JavaScript care calculează şi afişează determinantul matricei $\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$.

1. Creaţi documentul (X)HTML (figura 11.30).

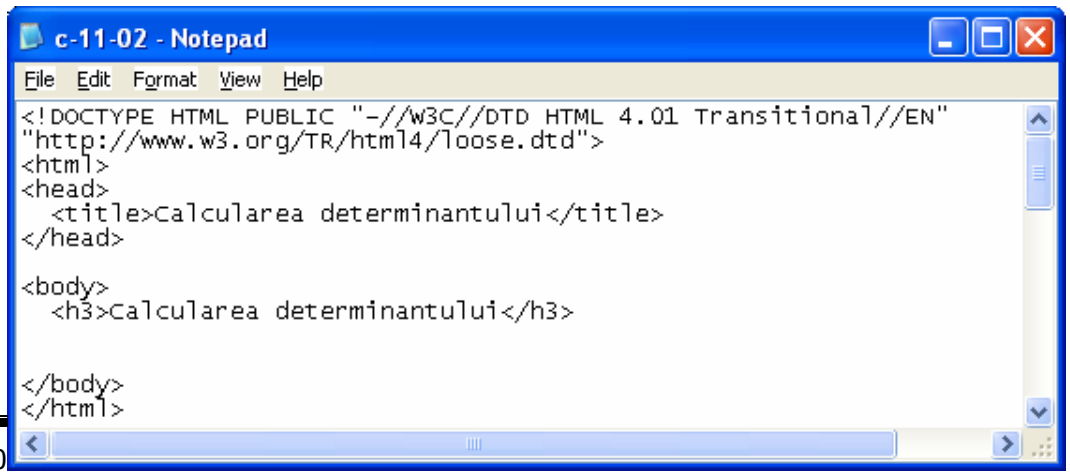


Figura 11.30

2. Plasaţi script-ul de calcul al determinantului în documentul (X)HTML (figura 11.31).

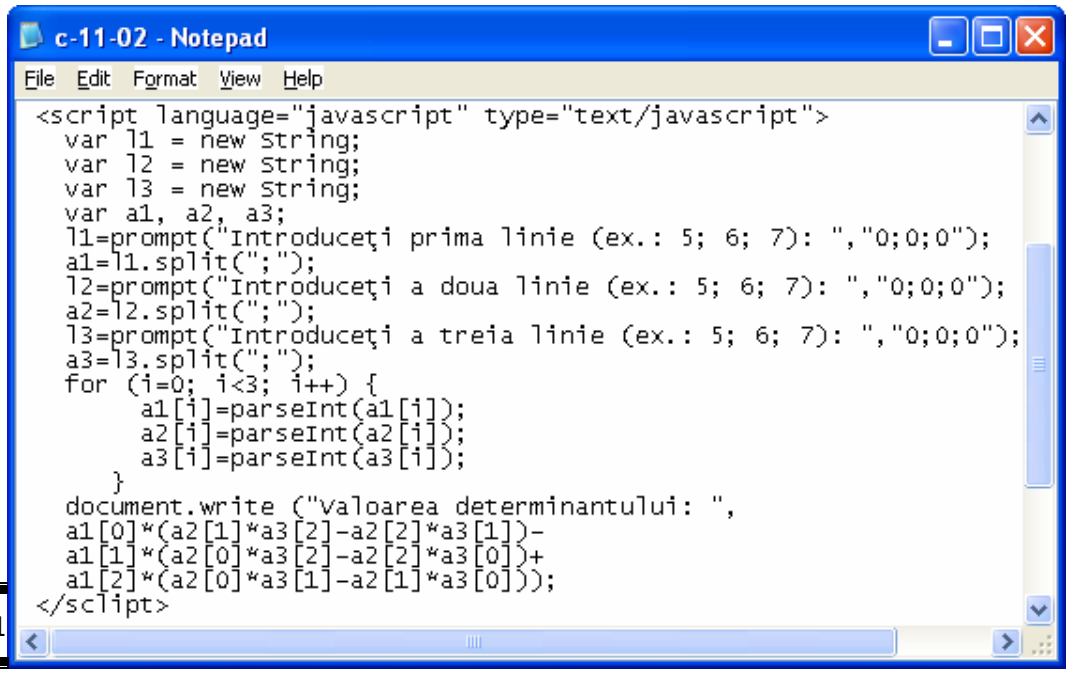


Figura 11.31

3. Validaţi documentul (X)HTML cu aplicaţia `validator` (http://validator.w3.org).

În figura 11.32 este prezentat rezultatul validării obţinut cu aplicaţia `validator`.

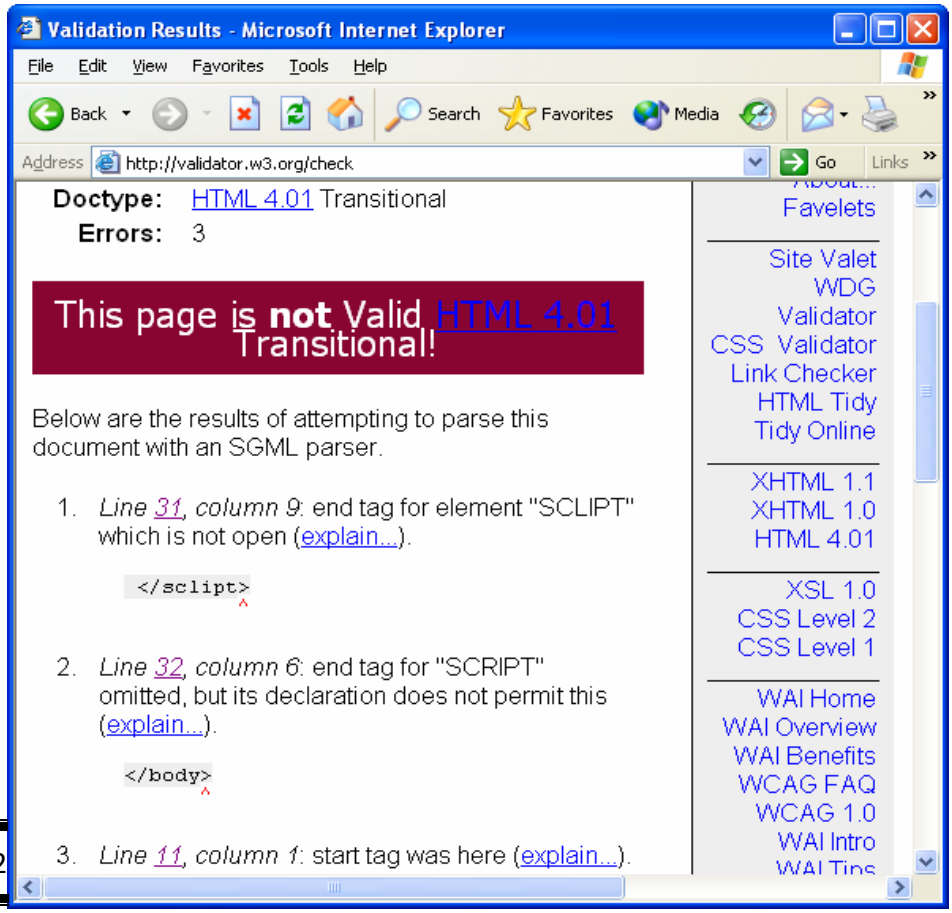


Figura 11.32

Remarcă. Nimic nu este perfect! Aţi identificat eroarea? Deşi îl pronunţ bine pe r, în loc de r am tastat l (</sclipt>). Se întâmplă, nu-i aşa! Corectaţi `</script>`!

4. Corectaţi eroarea şi validaţi documentul (X)HTML cu aplicaţia `validator` (http://validator.w3.org), figura 11.33.

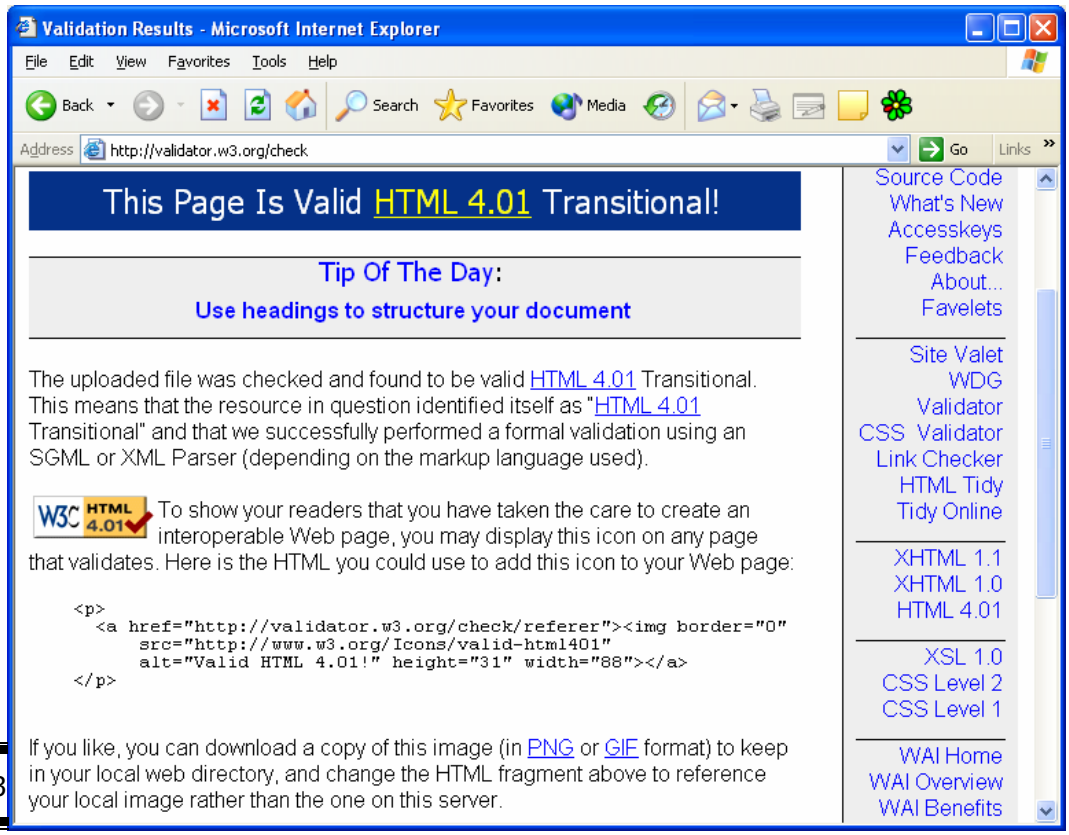


Figura 11.33

5. Inseraţi codul HTML care afişează icon-ul de conformitate (figura 11.34).

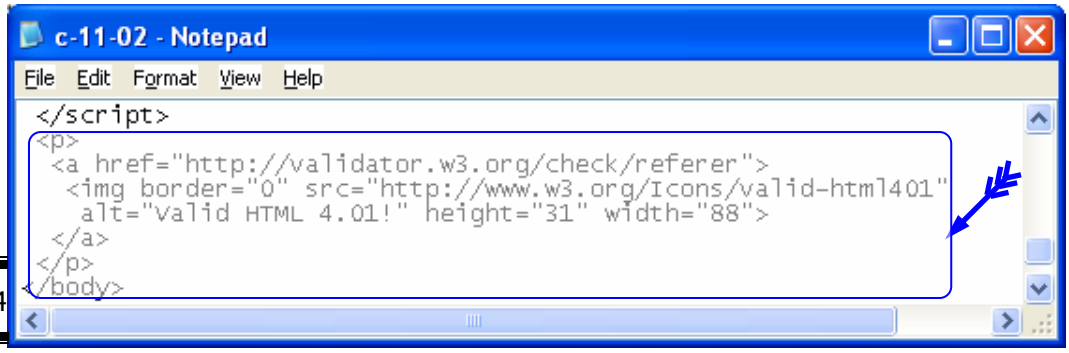


Figura 11.34

6. Vizualizaţi pagina Web într-un browser (Internet Explorer), figura 11.35.

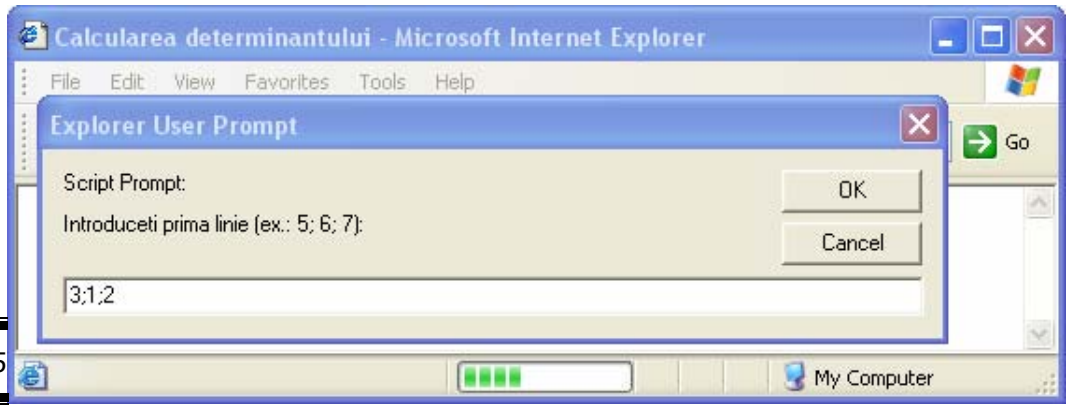


Figura 11.35

Remarcă. Internet Explorer afişează icon-ul de conformitate la baza paginii.

7. Testaţi script-ul (figura 11.36).

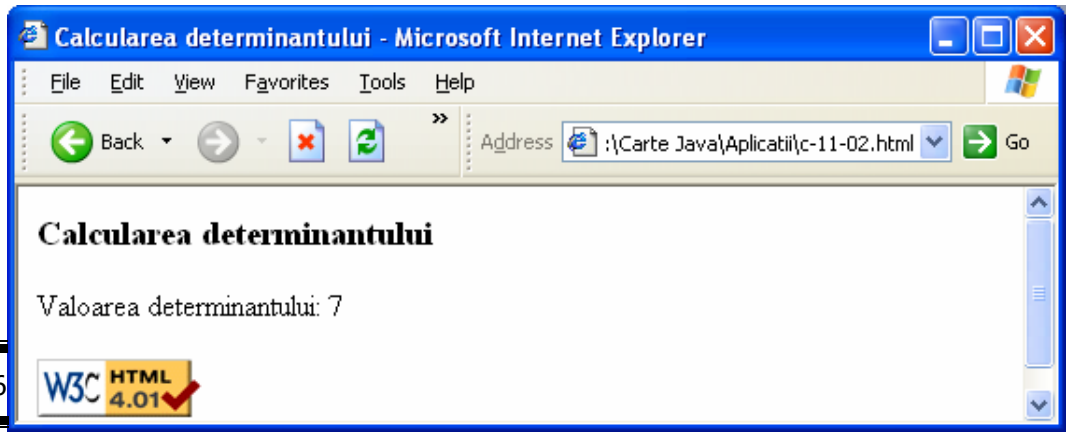


Figura 11.36

Instrucţiunile throw şi try … catch

Instrucţiunea `throw` generează o eroare. Un bloc `try` conţine instrucţiuni JavaScript; la apariţia unei erori, programul execută blocul `catch`.

Instrucţiunea throw

Sintaxa instrucţiunii throw este prezentată în figura 11.37.

Instrucţiune	***Sintaxă***
`throw`	`throw "mesaj eroare"`

Figura 11.37

Remarcă. Instrucţiunea `throw` este utilizată frecvent într-o instrucţiune `try … catch`; ea transferă eroarea blocului `catch`.

Iată cum procedăm pentru a genera două mesaje de eroare: "Eroare 1", "Eroare 2" în situaţia în care numărul de rezervoare cilindrice echilaterale care a fost introdus de la tastatură este zero, respectiv negativ (vezi EXEMPLELE JAVASCRIPT).

În figura 11.38 este prezentat script-ul aplicaţiei.

```
<script>
   try {
         n=prompt ('Introduceți numărul de rezervoare (n)',0);
         if (n==0)
               throw "Eroare 1"
         else
               if (n<0> throw "Eroare 2"
         }
   catch (er) {
         if (er=="Eroare 1")   alert ("Eroare 1 – număr rezervoare=0")
         if (er=="Eroare 2")   alert("Eroare 2 – număr rezervoare negativ")
         }
</script>
```

Figura 11.38

Comentarii:

- ✓ Programul execută blocul `try`.
- ✓ În situaţia în care **n** nu este egal cu zero şi nici negativ, nu se generează nici o eroare iar blocul `catch` nu este executat. Pentru utilizator nu se întâmplă absolut nimic.
- ✓ În cazul în care valoarea variabilei `n` a fost fixată la 0 în cursul instrucţiunilor executate anterior, eroarea "Eroare 1" este generată (această eroare este de acelaşi tip cu erorile de sintaxă). Programul părăseşte blocul `try` şi execută blocul `catch`, furnizându-i ca parametru tipul de eroare returnat prin throw. În definitiv, el afişează un mesaj `alert()`.

Instrucţiunea try ... catch

`Try ... catch` serveşte la testarea unei porţiuni de cod JavaScript şi la depistarea eventualelor erori. Ea este alcătuită din trei blocuri: `try`, `catch` şi `finally`.

Sintaxa instrucţiunii este prezentată în figura 11.39.

Instrucţiune	***Sintaxă***
`try ... catch`	`try {` `  cod1` `}` `catch (Eroare) {` `  cod2` `}` `finally {` `  cod3` `}`

Figura 11.39

Remarci:

- ✓ `Try` conţine instrucţiuni (cod1) JavaScript care ar putea fi sursa problemei. În cazul apariţiei unei erori, script-ul sare imediat în blocul `catch` transmiţându-i ca parametru obiectul `Error` generat.
- ✓ Dacă instrucţiunile din blocul `try` nu depistează nici o eroare, script-ul trece imediat la blocul `finally`, evitând blocul `catch`.

Iată cum calculăm aria unui rezervor sferic ($a = 4\pi R^2$) cu raza de 3m utilizând funcţia internă `eval()`. Dacă `eval()` recunoaşte instrucţiunea JavaScript `a=4*Math.PI*Math.pow(3,2)` atunci ea evaluează textul primit ca argument şi returnează rezultatul. În caz contrar, navigatorul afişează un mesaj de eroare.

În figura 11.40 este prezentat script-ul aplicaţiei.

```
<script>
try {
        a=prompt ("Introduceţi a=4*Math.PI*Math.pow(3,2)","");
        aeval=eval(a);
        alert ("Aria rezervorului sferic cu raza de 3m este:"+aeval);
}
catch(err) {
if (err.name=="SyntaxError")  alert ("Eroare de sintaxă!")
else  alert ("Expresia nu poate fi evaluată")
}
</script>
```

Figura 11.40

Comentarii:

- ✓ `try` aşteaptă … o eroare!
- ✓ `eval()` acceptă un şir de instrucţiuni JavaScript şi îl evaluează ca fiind cod sursă.
- ✓ Dacă utilizatorul introduce `"a=4*Math.PI*Math.pow(3,2)"`, JavaScript poate evalua expresia şi va afişa 113.097.
- ✓ Dacă utilizatorul introduce `"a=4*Math.PI*pow(3,2)+"`, spunem că avem de-a face cu o eroare de sintaxă.
- ✓ În sfârşit, dacă utilizatorul introduce `"Droopy"`, JavaScript încearcă să evalueze valoarea variabilei `Droopy`, care nu există. Nu este vorba de o eroare de sintaxă ci de o eroare de referinţă. `"Expresia nu poate fi evaluat•"` se va afişa în acest caz pe ecran.

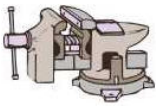
JavaScript recunoaşte şase tipuri de erori (vezi figura 11.41).

Numele erorii	***Descriere***
`EvalError`	Utilizare greşită a metodei `eval()`
`RangeError`	O valoare numerică este foarte importantă
`ReferenceError`	Referinţă la o variabilă, sau referinţă necunoscută sau invalidă
`SyntaxError`	Eroare de sintaxă
`TypeError`	Tipul de variabilă este greşit
`URLError`	Utilizare greşită a URL-ului

Figura 11.41

Remarcă. Nu uitaţi că şi dumneavoastră puteţi genera cu `throw` propriile erori. Succes!

Testați-vă cunoștințele



1. Dacă la crearea unui script, comiteţi greşeli de tastare, ce tip de eroare este generat?
   - eroare de sintaxă
   - eroare de logică
   - eroare de intrare/ieşire
2. De regulă, greşelile de punctuaţie care se comit la crearea unui script sunt:
   - confuzia între majuscule şi minuscule
   - paranteze orfeline
   - inversarea literelor
3. Foarte multe erori de logică sunt generate de:
   - o proiectare defectuoasă a programului JavaScript
   - confuzia între şirurile de caractere şi numere
   - greşeli de plasare a instrucţiunilor JavaScript
4. Inserarea comentariilor în programele JavaScript contribuie la:
   - crearea unor reflexe JavaScript
   - eliminarea erorilor de sintaxă
   - eliminarea erorilor de logică
5. În JavaScript una din erorile cele mai frecvente comise de începători este confuzia dintre:
   - operatorul de atribuire şi operatorul de egalitate
   - operatorul ”==” şi operatorul ”===”
   - operatorul ”||” şi operatorul ”&&”
6. Consola JavaScript este:
   - un instrument simplu pentru afişarea mesajelor de eroare cu Netscape
   - un instrument simplu pentru afişarea mesajelor de eroare cu Internet Explorer
   - un buton încastrat
7. W3C validator este un instrument de depanare a documentelor:
   - (X)HTML
   - XML
   - JavaScript

8. Instrucţiunea `throw` generează:
    - o eroare
    - un tabel
    - un formular
9. JavaScript recunoaşte următoarele tipuri de erori:
    - `EvalError`
    - `SintaxError`
    - `TypeError`
    - `FormatError`
    - `LogicalError`

Vizitați site-urile

- ✓ http://developer.netscape.com/docs/manuals/jsref/objintro.htm
- ✓ http://msdn.microsoft.com/scripting/jsscript/default.htm

Conversația 12

Crearea obiectelor personalizate

În această conversație:

- *Utilizarea obiectelor personalizate pentru simplificarea script-urilor*
- *Definiţi un obiect*
- *Definiţi o metodă pentru un obiect*
- *Creaţi o instanţă a unui obiect*
- *Aplicaţii*
- *EXEMPLUL 12 JAVASCRIPT*
- *Temă*

Utilizarea obiectelor personalizate pentru simplificarea script-urilor

Fără să greşim, putem afirma că aveţi deja o experienţă (un background) în utilizarea obiectelor predefinite ale limbajului JavaScript.

Sunteţi de asemenea familiarizaţi cu obiectele DOM-ului (Document Object Model) care permit manipularea documentelor Web. Aceste obiecte sunt cele mai frecvente în programarea JavaScript.

Dar ... puteţi de asemenea să creaţi propriile dumneavoastră obiecte, ceea ce trebuie să recunoaştem este ceva nemaipomenit!

Dacă variabilele şi matricile permit stocarea datelor sub diverse forme, uneori este necesar să facem apel la structuri mai sofisticate! Să presupunem că doriţi să creaţi un script care să gestioneze coordonatele persoanelor de contact (nume, adresa site, adresa email) din activitatea dumneavoastră profesională.

În situaţia în care utilizaţi variabile „normale” veţi fi obligat să definiţi o variabilă pentru fiecare coordonată a fiecărei persoane (pentru fiecare nume, fiecare adresă site şi fiecare adresă email) ceea ce este foarte complicat!

Utilizarea matricilor simplifică puţin lucrurile, dar nu este ideal!

Obiectele (aici am vrut să ajungem!) permit stocarea informaţiilor bazei de date într-o manieră mult mai „logică”.

Fiecare persoană este reprezentată printr-un obiect `Contact` care dispune de proprietăţile: nume, adresă site şi adresă email.

Puteţi adăuga după aceea obiectului `Contact`, metode pentru afişarea/manipularea informaţiilor conţinute.

Definiți un obiect

Prima etapă în crearea unui obiect constă în a-i da un nume. Am decis să numim obiectul `Contact`. Fiecare obiect `Contact` conţine următoarele proprietăţi:

- ✓ nume;
- ✓ site (adresa de site (personală));
- ✓ email (adresa de email (personală)).

Pentru a putea utiliza obiectul, va trebui mai întâi să creăm o funcţie `Constructor` care are rolul de a crea noi obiecte `Contact`.

Iată cum creăm în JavaScript o funcţie `Constructor (Contact)` cu trei parametri pentru iniţializarea noului obiect şi atribuirea proprietăţilor corespunzătoare.

În figura 12.1 este prezentat codul JavaScript al funcţiei `Constructor`.

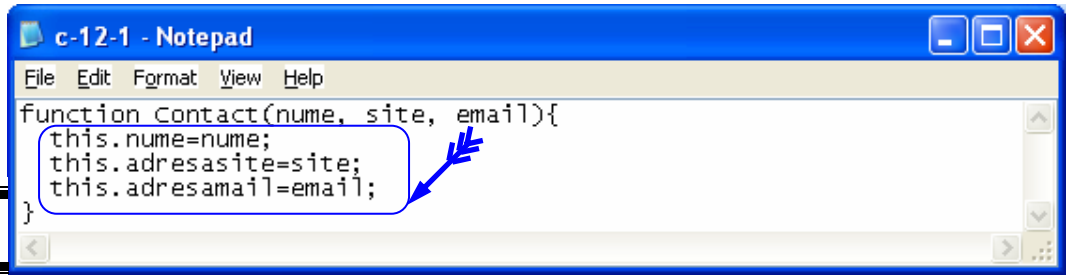


Figura 12.1

Remarci:

- ✓ Constructorul este o funcţie simplă care acceptă mai mulţi parametri ale căror valori sunt atribuite proprietăţilor obiectului.
- ✓ Obiectul are acelaşi nume ca şi funcţia `Contact`.
- ✓ Cuvântul cheie `this` se referă la obiectul curent, acela creat cu ajutorul funcţiei.

Definiţi o metodă pentru un obiect

Obiectele devin cu adevărat interesante şi comode atunci când ele sunt dotate cu metode.

În JavaScript, definirea unei metode pentru un obiect se realizează în două etape:

- ✓ Definiţi metoda în funcţia `Constructor` într-o linie de cod în care veţi atribui metodei un nume, apoi asociaţi-i funcţia de prelucrare.
- ✓ Scrieţi funcţia de prelucrare.

Iată cum definim metoda `afi•areContact` în funcţia Constructor `Contact`.

În figura 12.2 este prezentat codul complet JavaScript al funcţiei `Constructor`.

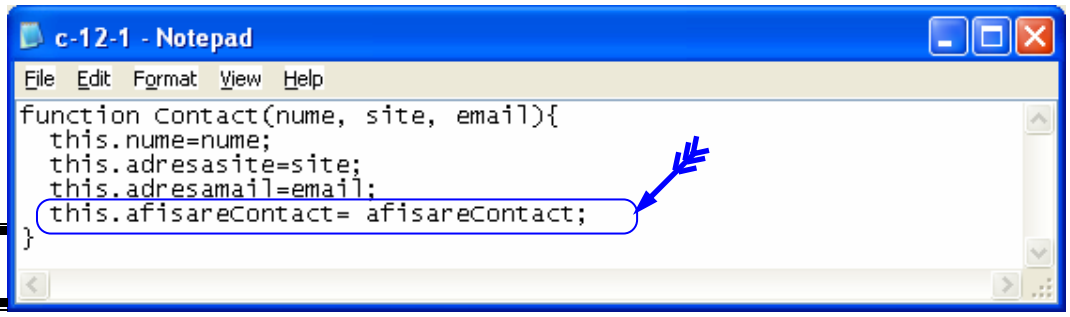


Figura 12.2

Remarci:

- ✓ Funcţia `afi•areContact` afişează coordonatele de contact ale unei persoane.
- ✓ Instrucţiunea care a fost adăugată seamănă cu o definiţie a proprietăţii, dar se referă la funcţia `afi•areContact`. Puteţi utiliza această sintaxă din momentul în care funcţia `afi•areContact` a fost definită.
- ✓ Funcţia `afi•areContact` este utilizată ca metodă, fără parametri!
- ✓ Cuvântul cheie `this` se referă la obiectul curent.

Iată cum scriem în JavaScript funcţia de prelucrare `afi•areContact` asociată metodei cu acelaşi nume.

În figura 12.3 este prezentat codul JavaScript al funcţiei `afi•areContact`.

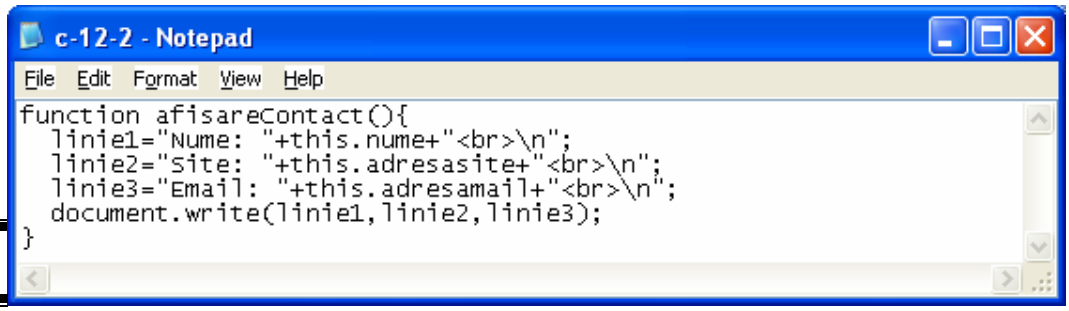


Figura 12.3

Remarcă. Cuvântul cheie `this` se referă la proprietăţile obiectului.

Creaţi o instanţă a unui obiect

În continuare, vom utiliza definiţia obiectului pe care urmează să-l creăm. Pentru a utiliza o definiţie a obiectului va trebui să creăm un nou obiect cu ajutorul cuvântului cheie new (vezi obiectele `Array, Date, String`).

În figura 12.4 se prezintă codul JavaScript care creează un nou obiect `Contact` numit `Droopy`.

Figura 12.4

Remarci:

- ✓ Crearea unui obiect nu este o operaţie complicată. Tot ceea ce trebuie făcut este apelarea funcţiei `Contact()`, care este funcţia de definire a obiectului, şi indicarea coordonatelor în aceeaşi ordine ca în definiţie.
- ✓ Odată executată instrucţiunea din figura 12.4, un nou obiect a fost creat, care conţine informaţii despre `Droopy`. Acest obiect (`Droopy`) este o instanţă a obiectului `Contact`.
- ✓ În figura 12.5 se prezintă o secvenţă de cod JavaScript care creează un obiect `Contact` vid (Romică) şi defineşte apoi proprietăţile sale.

Figura 12.5

Odată creată instanţa obiectului `Contact` cu ajutorul uneia din metodele prezentate, utilizaţi metoda `afi•areContact` pentru a afişa informaţiile corespunzătoare. Proprietăţile persoanei de contact `Droopy` sunt afişate ca în figura 12.6.

Figura 12.6

Aplicaţii

❑ Creaţi un document (X)HTML care afişează în Internet Explorer lista de contact a trei persoane (vezi figura 12.7).

Indicaţie. Utilizaţi obiectul `Contact`.

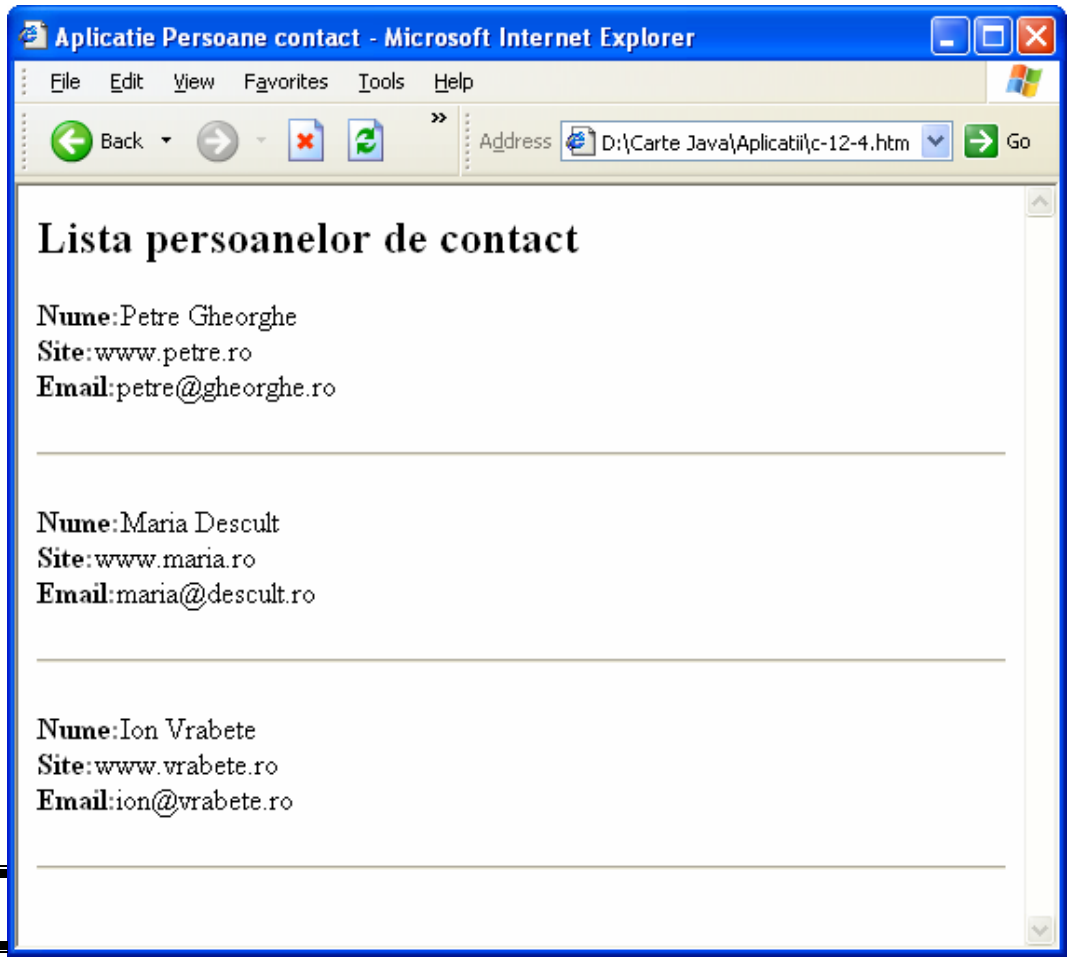


Figura 12.7

În figura 12.8 este prezentat documentul XHTML complet în care s-a inserat script-ul aplicaţiei.

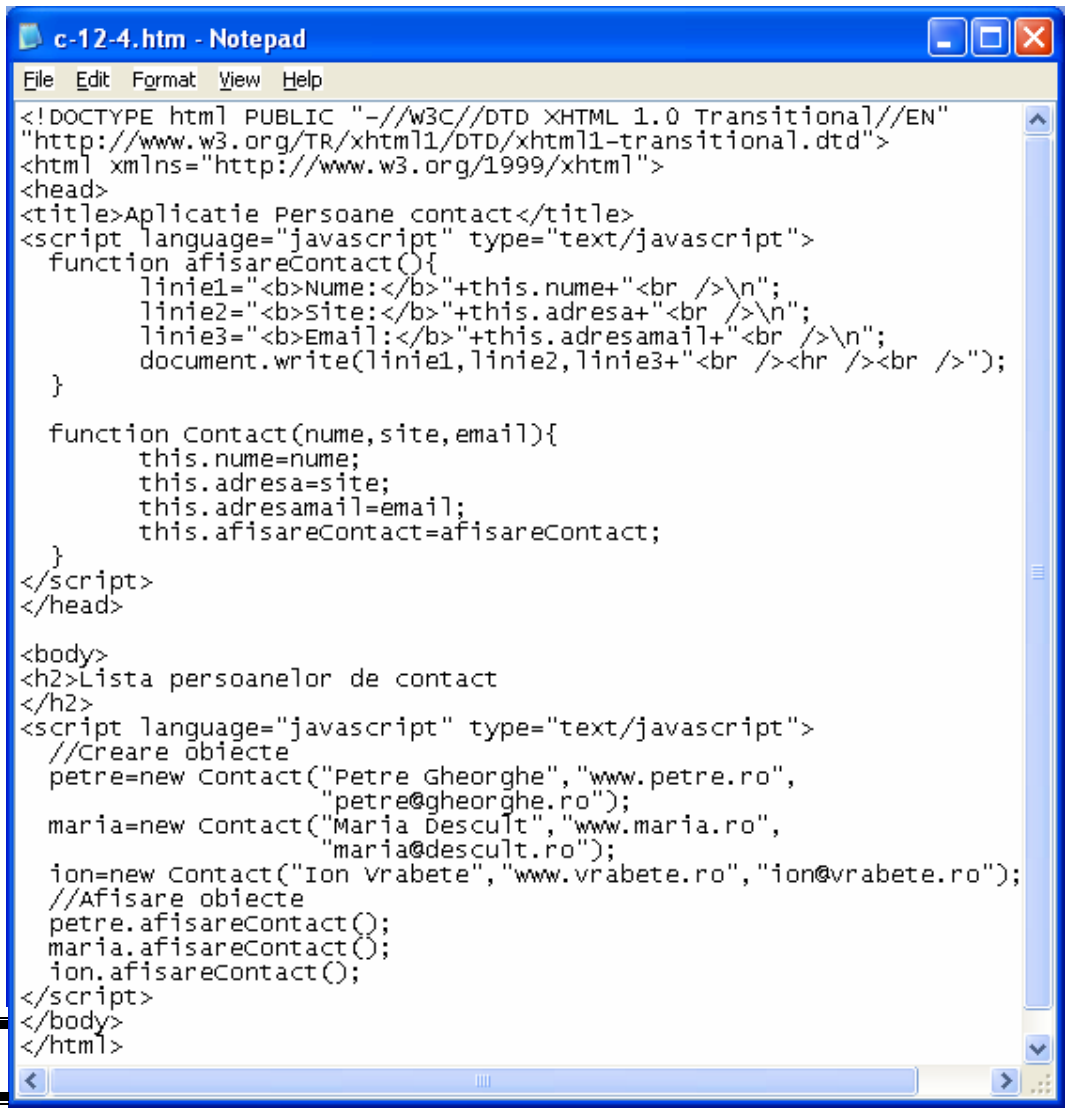


Figura 12.8

Remarcă. În cadrul acestei aplicaţii baza de date utilizată are dimensiuni extrem de reduse. Un obiect asemănător poate stoca date care provin dintr-o bază de date externă care conţine mii de înregistrări.

❑ Modificaţi definiţia obiectului `Contact` pentru a include o proprietate `adresa` care permite stocarea adresei persoanei de contact. Modificaţi funcţia `afi•areContact`.

EXEMPLUL 12 JAVASCRIPT

❑ Definirea problemei

Aplicația pe care urmează să o realizați în cadrul acestei ultime conversații (!) este o mică bază de date care conține notele obținute la mai multe materii, de către studenții unei grupe.

Pentru gestionarea automată a acestor date, vom folosi facilitățile JavaScript (*tabelul pe parte de client*), pentru a răspunde cerințelor conducerii unei instituții de învățământ.

❑ Cerințe de prelucrare

Se vor avea în vedere cerințele de prelucrare prezentate mai jos:

- ✓ studenții sunt identificați prin: cod, nume, prenume;
- ✓ materiile de studiu sunt identificate prin: cod, denumire;
- ✓ nota 0 (zero) semnifică absența studentului de la examen sau verificare.
- ✓ rapoartele generate sunt de unul din următoarele tipuri:
  - *raport general* – studenții sunt afişați în ordine alfabetică (nume, prenume şi notele obținute la toate materiile);
  - *raport pe materii* – pentru o materie selectată din lista de materii, studenții sunt listați în două moduri:
    - ordine alfabetică;
    - ordine descrescătoare a notelor;
  - *raport restanțieri* – pentru fiecare materie se afişează studenții care nu au obținut nota 5 de promovare;
- ✓ cererile de căutare în baza de date sunt:
  - identificarea unui student şi afişarea notelor obținute la toate materiile;
  - identificarea studenților restanțieri pentru o materie selectată.

❑ Proiectarea programului

Crearea tabelelor de căutare

Se vor utiliza *obiecte personalizate* care să simuleze o bază de date. Documentul (X)HTML va conține informații despre două entități:

- ✓ materie – conține atributele: cod (numărul de identificare al materiei), denumirea materiei (şir de caractere);
- ✓ student – conține atributele: numele şi prenumele studentului; notele obținute la materiile de studiu.

Remarcă. A fost aleasă o soluție simplificată care prezintă dezavantajul unei flexibilități reduse la modificarea numărului materiilor de studiu şi la adăugarea mai multor ani de studiu, dar ... din punct de vedere didactic este mult mai uşor de înțeles.

Tabelul *entități materii* pe parte de client şi tabelul *entități student* pe parte de client se vor implementa după cum urmează:

- ✓ *tabelul materii* – se va utiliza un vector şir de caractere; codul de identificare al materiei va fi indicele materiei iar elementele vectorului, un şir de caractere ce reprezintă denumirea materiei.
- ✓ *tabelul student* – se va utiliza un vector de obiecte personalizate în care indicele vectorului va reprezenta numărul de identificare al studentului iar obiectul personalizat va fi format din două şiruri de caractere:
  - Nume, Prenume – reprezintă numele şi prenumele studentului;
  - Note – vector de numere întregi reprezentând notele obținute de student (dimensiunea vectorului de note trebuie să fie egală cu dimensiunea vectorului materii care defineşte numele materiilor de studiu).

Definiți obiectul `student` (vezi metoda constructor, `def_student (Nume, Prenume, Note)` din figura 12.9) şi creați instanțe de acest tip.

Pentru aceasta va trebui să creați mai întâi lista cu note şi apoi să apelați metoda constructor (`def_student`) a obiectului `student` (vezi figura 12.10).

Figura 12.9

```
function def_student(Nume,Prenume,note){
    this.Nume=Nume;
    this.Prenume=Prenume;
    this.Note=Note;
}
```

Figura 12.10

```
...
    var lista_stud=new Array();
    var note1=new Array(5,6,7,10,10);
    lista_stud[0]=new def_student(”Avram”,”Nicolae”,note1);
}
```

Crearea interfeței cu utilizatorul

Interfața cu utilizatorul trebuie să asigure afişarea facilă a tuturor cerințelor de prelucrare. Pentru a afişa simultan opțiunile de prelucrare, criteriile şi rezultatele prelucrării cererilor de căutare se va folosi o fereastră cu mai multe (4) cadre (vezi figura 12.11).

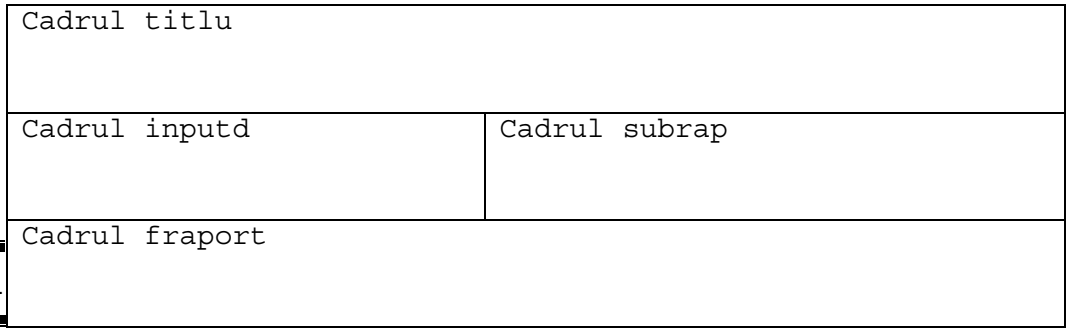


Figura 12.11

Vom prezenta în cele ce urmează funcțiile celor patru cadre în care a fost împărțită fereastra browser-ului.

Cadrul titlu

Cadrul *titlu* afişează textul `Intranet - Baza de Date cu Studen•i` (vezi figura 12.12).

Figura 12.12

Cadrul inputd

Cadrul *inputd* afişează tipul prelucrării. Se afişează două butoane radio (`Rapoarte, C•utare`) cu ajutorul cărora utilizatorul poate selecta tipul de raport (`General, Pe Materii, Restan•ieri`) şi tipul căutării (`Dup• studen•i, Restan•ieri`).

În figura 12.13 se prezintă cadrul inputd al ferestrei interfeței cu utilizatorul cu tipul prelucrării `RAPOARTE` iar în figura 12.14 se prezintă acelaşi cadru cu tipul prelucrării: `C•UTARE`.

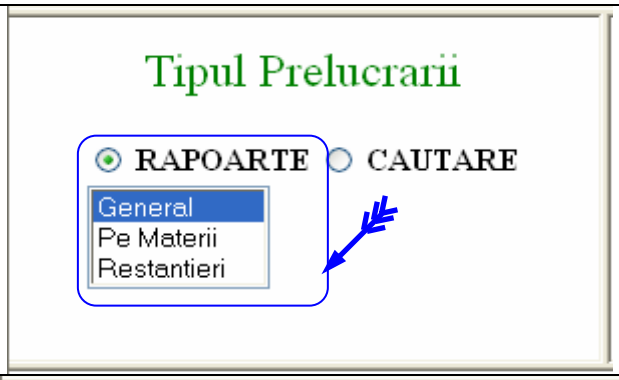


Figura 12.13

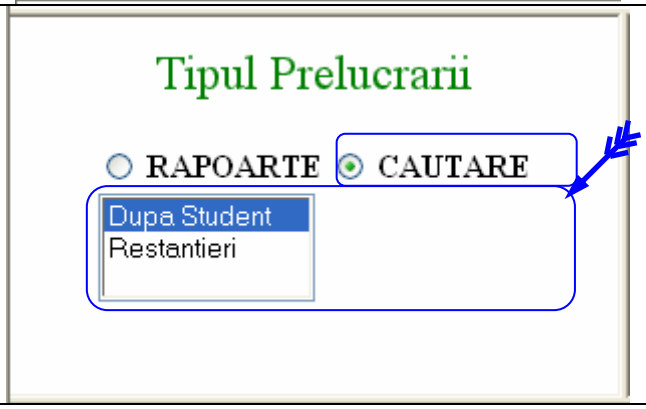


Figura 12.14

Cadrul subrap

Cadrul *subrap* este cadrul prin intermediul căruia se introduc datele necesare efectuării prelucrărilor cerute de utilizator prin selecția unei opțiuni din cadrul *inputd*. El este generat dinamic.

În figurile 12.15, 12.16, 12.17, 12.18, 12.19 este prezentat conținutul cadrului *subrap* pentru opțiunile `RAPOARTE` şi `C•UTARE`.

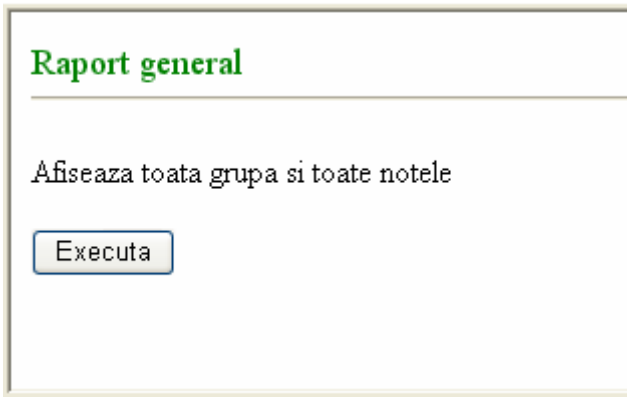


Figura 12.15

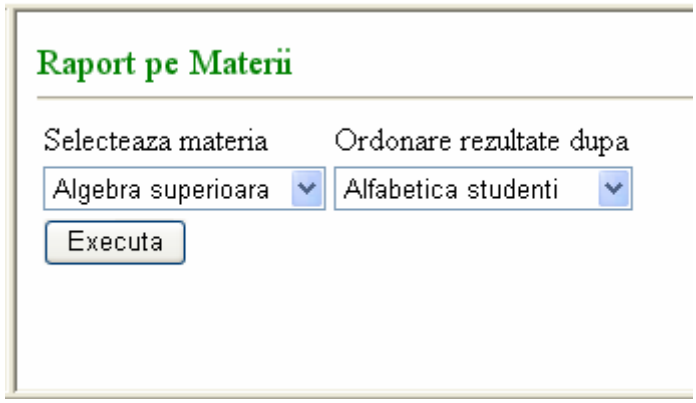


Figura 12.16

Figura 12.17

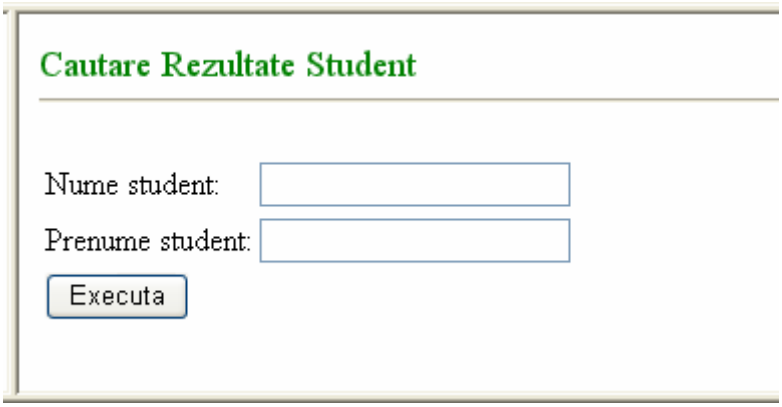


Figura 12.18

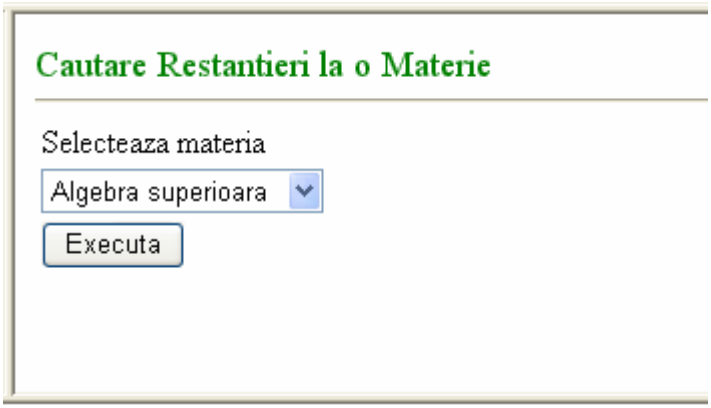


Figura 12.19

Remarcă. Observați în figurile 12.15, 12.16, 12.17, 12.18, 12.19 prezența butonului *EXECUTĂ*.

Cadrul fraport

Cadrul *fraport* este cadrul în care se afişează rezultatele prelucrărilor. Conținutul acestuia este dinamic, fiind generat în funcție de cererea de căutare selectată.

În figura 12.20 este prezentat cadrul *fraport* al ferestrei interfeței cu utilizatorul pentru `RAPORT GENERAL` (Rapoarte → General, în cadrul *inputd*).

RAPORT GENERAL

NR.	Nume Prenume	Algebra superioara	Fizica	Chimie	Informatica	Analiza Functionala
1	Avram Nicolae	5	6	7	10	10
2	Berbecaru Catalin	4	9	5	7	8
3	Botea Ion	10	9	8	10	10
4	Cazacu Marin	6	8	7	7	5
5	Ciufu Gheorge	9	8	0	7	8
6	Codescu Petre	8	6	8	6	5
7	Dinu Mihaela	4	4	0	5	5
8	Dobre Magdalena	5	6	5	4	4
9	Georgescu Dragos	9	4	7	0	5
10	Ionescu Ludovic	6	8	9	6	6
11	Mihail Gheorghe	4	0	4	4	5
12	Nastase Stefan	8	7	7	8	6

Figura 12.20

❑ Crearea funcțiilor de prelucrare a datelor

Aplicația poate genera trei rapoarte şi execută două cereri de căutare. Tipul raportului sau al căutării este selectat de utilizator prin intermediul cadrului `inputd`. Corespunzător selecției utilizatorului, în cadrul `subrap` se vor modifica zonele de interacțiune cu utilizatorul. Astfel, dacă utilizatorul selectează generarea `raportului general` atunci cadrul `subrap` va afişa un buton pentru a lansa în execuție prelucrarea (vezi figura 12.18). Cadrul `inputd` permite şi schimbarea tipului de prelucrare (`raport` sau `c•utare`) prin afişarea a două butoane radio prin care utilizatorul poate realiza selecția. Acționarea unuia din aceste butoane radio lansează în execuție funcția `afisrap` care modifică opțiunile listei `tiprap`.

În figura 12.21 este prezentat pseudocodul funcției `afisrap()`.

Pseudocodul AFISRAP

```
AFISRAP   BEGIN
                Parametrii: t- valoare intreaga
IFAFIS            IF (t==0)
                        //afisare tip raport
                        //fixeaza trei optiuni in lista tiprap
                        tiprap.optiune[0]='General'
                        tiprap.optiune[1]='Materii'
                        tiprap.optiune[2]='Restantieri'
IFAFIS            ELSE
                        tiprap.optiune[0]='Student'
                        tiprap.optiune[1]='Restantieri'
IFAFIS            ENDIF
AFISRAP   END
```

Figura 12.21

Selectarea unei opțiuni din lista `tiprap` va duce la afişarea în cadrul `subrap` a elementelor de interfață ce asigură introducerea informațiilor necesare pentru fiecare tip de prelucrare şi lansarea în execuție a prelucrării respective. Astfel, se gestionează evenimentul `onChange` care va executa funcția `afis_subopt`. Pseudocodul funcției `afis_subopt()` este prezentat în figura 12.22.

Pseudocodul AFIS_SUBOPT

```
AFIS_SUBOPT  BEGIN
                  //determinarea indexului optiunii selectate
                  ir=tiprap.selectedIndex
IF1               IF (raport[0].checked)
CASE1                   CASE (ir)
                              0:DO afis_subrap0
                              1:DO afis_subrap1
                              2:DO afis_subrap2
CASE1                   ENDCASE
IF1               ELSE
CASE2                   CASE (ir)
                              0:DO afis_subrap3
                              1:DO afis_subrap4
CASE2                   ENDCASE
IF1               ENDIF
AFIS_SUBOPT  END
```

Figura 12.22

Funcția `afis_subrap0` construieşte interfața cu utilizatorul pentru prelucrarea datelor în cazul generării `raportului general` (figura 12.20). Interfața va fi afişată în frame-ul `subrap` şi conține ca element de acțiune doar butonul `Executa` care prin apăsare va apela funcția `executaRG()` care va afişa `raportul general` (vezi pseudocodul din figura 12.23).

Pseudocodul AFIS_SUBRAPO, EXECUTARG

```
AFIS_SUBRAP0   BEGIN
                        //afişarea informațiilor despre raport
                        subrap.WRITE 'Raport General'
                        subrap.WRITE 'Afiseaza toata grupa si toate notele'
                        genereaza html pentru form f2 ce contine butonul executa
IF1                     IF (clic pe butonul Executa)
                                DO executaRG
IF1                      ENDIF
AFIS_SUBRAP0   END

EXECUTARG      BEGIN
                        //afişarea studentilor din grupa
                        subrap.WRITE 'Raport General'
                        subrap.WRITE 'Nume prenume'
FOR1                    FOR(i=0;i<materii.length;i++)
                                subrap.WRITE materii[i]
FOR1                    ENDFOR
FOR2                    FOR(i=0;i<lista_stud.length;i++)
                                subrap.WRITE lista_stud[i].nume lista_stud[i].prenume
FOR3                            FOR(j=0;j<materii.length;j++)
                                        subrap.WRITE lista_stud[i].Note[j]
FOR3                            ENDFOR
FOR2                    ENDFOR
EXECUTARG      END
```

Figura 12.23

Funcția `afis_subrap1` construieşte interfața cu utilizatorul pentru generarea `raportului pe materii` (figura 12.24).

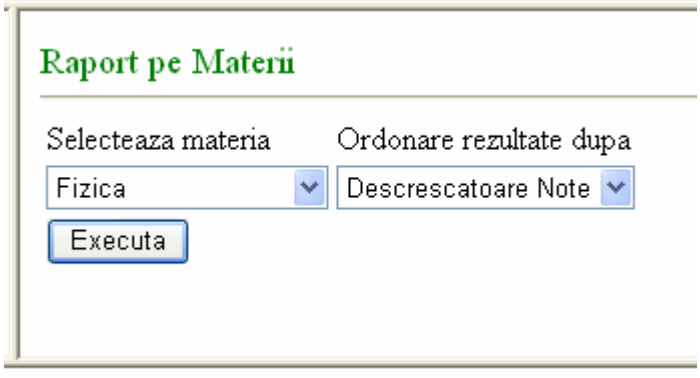


Figura 12.24

Interfața va fi afişată în frame-ul `subrap` şi conține două liste de opțiuni şi butonul `Executa`. Lista de opțiuni `smaterii` conține lista materiilor, iar lista `sordine` permite selecția modului de ordonare al studenților: ordinea alfabetică sau ordinea dată de notele la materia respectivă.

RAPORT NOTE MATERIA: Fizica

NR.	Nume Prenume	Nota
1	Berbecaru Catalin	9
2	Botea Ion	9
3	Oprea Tudor	9
4	Trifu Mihaela Elena	9
5	Cazacu Marin	8
6	Ciufu Gheorge	8
7	Ionescu Ludovic	8
8	Panait Gabriel Andrei	8
9	Patrichi Mihai Ioan	8

Figura 12.25

Butonul `Execut`• prin apăsare va apela funcția `executaRM()` care va afişa `raportul pe materii` (vezi figura 12.25). Afişarea studenților se va face în funcție de ordinea impusă de vectorul asociat `vord`. Pentru afişarea studenților în funcție de nume, ordinea este cea în care sunt păstrate în vectorul `lista_stud` informațiile despre studenți. În cazul ordonării descrescătoare după note, `vord` va conține indicii studenților din vectorul list_stud în ordinea descrescătoare a notelor. Astfel, `vord[0]` va conține indicele elementului din `lista_stud` al studentului care are nota cea mai mare la materia respectivă. Algoritmul de sortare implementat este un algoritm clasic `bubble-sort` (vezi pseudocodul din figura 12.26).

Pseudocodul AFIS_SUBRAP1, EXECUTARM

```
AFIS_SUBRAP1  BEGIN
                    //afişarea informațiilor despre raport
                    subrap.WRITE 'Raport pe Materii'
                    generaza html pentru forma f2
IF1                  IF (clic pe butonul Executa)
                            DO executaRG
IF1                  ENDIF
AFIS_SUBRAP1 END

EXECUTARM     BEGIN
                    //afişarea raportului pe materii al studentilor din grupa
                    subrap.WRITE 'Raport pe Materii'
                    //determina selectiile utilizatorului
                    imat=smaterii.selectedIndex;
                    iord=sordine.selectedIndex;
                    //aloca spatiu pentru vectorul vord
                    new vord
                    //se plaseaza ordinea initiala in vector
FOR1                FOR(i=0;i<vord.length;i++)
                            vord[i]=i;
FOR1                ENDFOR
```

Figura 12.26

```
IF2                    IF(iord==1)
                              //ordinea in functie de note
                              // ordoneaza lista_stud descrescator dupa note
                              // se aplica algoritmul de sortate bubble_sort
                              neordonat=true;
WHILE1                              WHILE (neordonat)
                                    neordonat=false;
FOR2                                FOR(i=0;i<lista_stud.length-1;i++)
IF3                                       IF(lista_stud[vord[i]].Note[imat]<
                                             lista_stud[vord[i+1]].Note[imat])
                                                  j=vord[i];
                                                  vord[i]=vord[i+1];
                                                  vord[i+1]=j;
                                                  neordonat=true;
IF3                                       ENDIF
FOR2                                ENDFOR
WHILE1                              ENDWHILE
                              // afiseaza studenti
FOR3                          FOR(i=0;i<lista_stud.length;i++)
                                    subrap.WRITE lista_stud[vord[i]].nume
                                    subrap.WRITE lista_stud[vord[i]].prenume
                                    subrap.WRITE lista_stud[i].Note[imat]
FOR3                          ENDFOR
IF2                    ENDIF
EXECUTARM     END
```

Figura 12.26 *(continuare)*

Funcția `afis_subrap2` construieşte interfața cu utilizatorul pentru generarea raportului ce afişează studenții restanțieri (figura12.27).

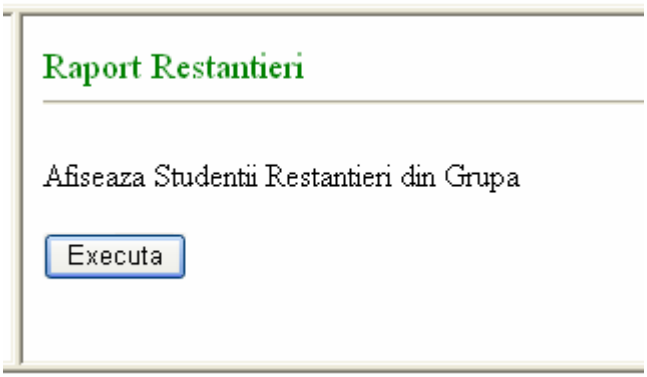


Figura 12.27

Interfaţa conţine doar butonul `Executa` care prin apăsare va apela funcţia `executaRR()` care va afişa raportul studenţilor restanţieri (vezi figura 12.28).

RAPORT RESTANTIERI

Materia:Algebra superioara

Numar restantieri:3

NR.	Nume Prenume	Nota
1	Berbecaru Catalin	4
2	Dinu Mihaela	4
3	Mihail Gheorghe	4

Materia:Fizica

Numar restantieri:3

NR.	Nume Prenume	Nota
1	Dinu Mihaela	4

Figura 12.28

Pentru afişarea studenților pe materii se apelează funcția `afis_restantieri` care are ca parametru indicele materiei ce urmează a fi prelucrată (vezi pseudocodul din figura 12.29).

Pseudocodul AFIS_SUBRAP2

```
AFIS_SUBRAP2  BEGIN
                        //afişarea informațiilor despre raport
                        subrap.WRITE 'Raport Restantieri'
                        generaza html pentru forma f2 ce conține butonul Executa
IF1                     IF (clic pe butonul Executa)
                                DO EXECUTARR
IF1                      ENDIF
AFIS_SUBRAP2  END
EXECUTARR     BEGIN
                        //afişarea raportului restantieri pe materii
                        subrap.WRITE 'Raport Restantieri'
FOR1                    FOR(j=0;j<materii.length;j++)
                                fraport.WRITE "Materia:",materii[j],
                                DO afis_restantieri(j);
FOR1                    ENDFOR
EXECUTARR      END

AFIS_RESTANTIERI BEGIN
                        Parametrii
                        imat- indicele materiei
                        nrest=0;
                         //determina numarul de restantieri
FOR2                    FOR(i=0;i<lista_stud.length;i++)
IFF                             IF (lista_stud[i].Note[imat]<5)
                                        nrest++;
IFF                             ENDIF
FOR2                     ENDFOR
IF2                     IF(nrest>0)
                                //afiseaza in tabel studentii restantieri
                                fraport.WRITE "Numar restantieri:",nrest
                        fraport.WRITE "NR. Nume Prenume  Nota
```

Figura 12.29

```
                                    k=0;
FOR3                                FOR(i=0;i<lista_stud.length;i++)
IF3                                         IF(lista_stud[i].Note[imat]<5)
                                                    fraport.WRITE i+1, lista_stud[i].Nume
                                                    fraport.WRITE lista_stud[i].Prenume
IF4                                                         IF(lista_stud[i].Note[imat]==0)
                                                                fraport.WRITE"absent"
IF4                                                         ELSE
                                                                fraport.WRITE
 lista_stud[i].Note[imat]
IF4                                                         ENDIF
IF3                                         ENDIF
FOR3                                FOR(i=0;i<lista_stud.length;i++)
IF2                         ELSE
                                    fraport.WRITE "Nu esista restantieri"
IF2                         ENDIF
AFIS_RESTANTIERI END
```

Figura 12.29 *(continuare)*

Funcţia `afis_subrap3` construieşte interfaţa cu utilizatorul pentru căutarea unui student după nume, prenume (figura 12.30).

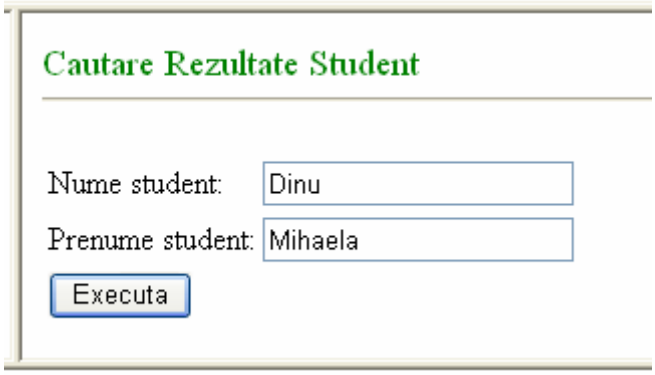


Figura 12.30

Interfața conține două zone de editare (`Nume student`, `Prenume student`) pentru specificarea numelui, respectiv prenumelui studentului căutat şi butonul `Executa`. Prin acționarea butonului `Executa` se va apela funcția `executaCS()` (vezi pseudocodul prezentat în figura 12.31) care va afişa rezultatele studentului dorit sau un mesaj de eroare dacă acesta nu există în baza de date (vezi figura 12.32).

Pseudocodul AFIS_SUBRAP3, EXECUTACS

```
AFIS_SUBRAP3 BEGIN
                        //afişarea informaţiilor despre raport
                        subrap.WRITE 'Cautare rezultate student'
                        generaza html pentru forma f2
IF1                     IF (clic pe butonul Executa)
                                DO EXECUTACS
IF1                      ENDIF
AFIS_SUBRAP3 END
EXECUTACS    BEGIN
                        //afişarea rezultatelor unui student
                        //transferul datelor introduse in zonele de editare
                        sn=snume.value;
                        pn=spnume.value;
                        fraport.WRITE"Cautare rezultate student"
IF2                     IF(sn!="")
```

Figura 12.31

```
IF3                     IF(pn!="")
                            // determina numarul de studenti ce au numele
                            // si prenumele specificat
                            nr=0;
FOR1                        FOR(i=0;i<lista_stud.length;i++)
IF4                             IF((lista_stud[i].Nume==sn)&&
                                   (lista_stud[i].Prenume==pn))
                                        Nr=nr+1
IF4                             ENDIF
FOR1                         ENDFOR
IF5                          IF(nr>0)
                                fraport.WRITE "NR. Nume Prenume
FOR2                            FOR(i=0;i<materii.length;i++)
                                        fraport.WRITE materii[i]
FOR3                                    FOR(i=0;i<lista_stud.length;i++)
IF6                                     IF((lista_stud[i].Nume==sn)&&
                                          (lista_stud[i].Prenume==pn))
                                               fraport.WRITE i+1, lista_stud[i].Nume
                                               fraport.WRITE lista_stud[i].Prenume
FOR4                                           FOR(j=0;j<materii.length;j++)
                                                    fraport.WRITE lista_stud[i].Note[j]
FOR4                                           ENDFOR
IF6                                     ENDIF
FOR3                                    ENDFOR
FOR2                            ENDFOR
IF5                          ELSE
                                fraport.WRITE "Nu exista nici un student cu numele:",
                                              sn," Prenumele:",pn
IF5                          ENDIF
IF3                     ELSE
                                //determina numarul de studenti cu numele sn
                                nr=0;
FOR5                            FOR(i=0;i<lista_stud.length;i++)
IF7                                     IF(lista_stud[i].Nume==sn)
                                                nr=nr+1
IF7                                     ENDIF
F0R5                            ENDFOR
IF8                             IF(nr>0)
                                        fraport.WRITE "Numar Studenti:",nr
                                        fraport.WRITE "NR. Nume Prenume "
FOR6                                    FOR(i=0;i<materii.length;i++)
                                                fraport.WRITE materii[i]
FOR6                                    ENDFOR
FOR7                                    FOR(i=0;i<lista_stud.length;i++)
IF9                                        IF(lista_stud[i].Nume==sn)
                                              fraport.WRITE i+1, lista_stud[i].Nume,
                                                        fraport.WRITE lista_stud[i].Prenume
FOR8                                          FOR(j=0;j<materii.length;j++)
                                                    fraport.WRITE lista_stud[i].Note[j]
FOR8                                          ENDFOR
IF9                                        ENDIF
FOR7                                    ENDFOR
IF8                                 ELSE
                                        fraport.WRITE "Nu exista nici un student
                                                       cu numele:", sn
IF8                              ENDIF
IF3                     ENDIF
IF2             ENDIF
EXECUTACS END
```

Figura 12.31 *(continuare)*

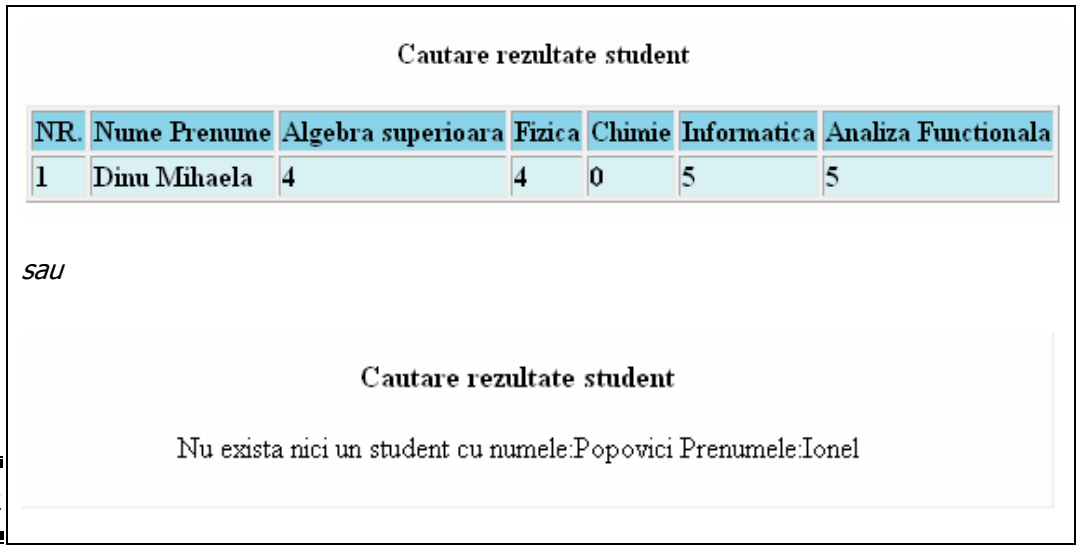


Figura 12.32

Funcţia `afis_subrap4` construieşte interfaţa cu utilizatorul pentru generarea afişării studenţilor restanţieri la o singură materie(figura 12.33).

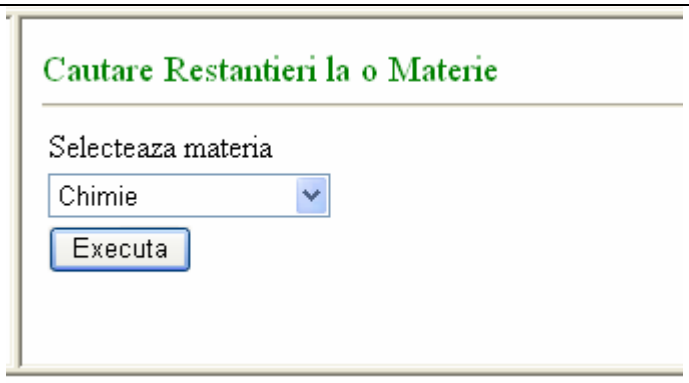


Figura 12.33

Interfața conține o listă de opțiuni, `(smaterii)`, pentru selecția materiei dorite şi butonul `Executa` care prin apăsare va apela funcția `executaCR()` ce va afişa raportul studenților restanțieri la materia respectivă(vezi figura 12.34).

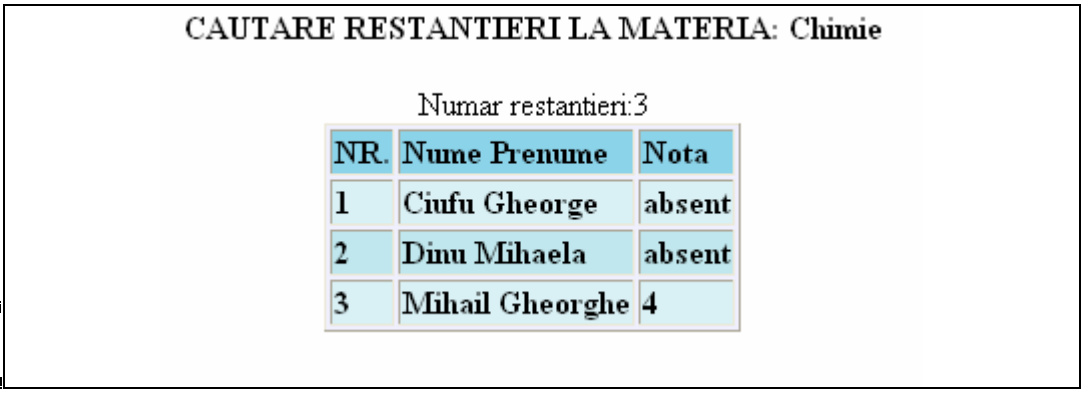


Figura 12.34

Pentru afişarea studenților restanțieri la o materie se apelează funcția `afis_restantieri` pe care am prezentat-o anterior (vezi pseudocodul prezentat în figura 12.35).

Pseudocodul AFIS_SUBRAP4, EXECUTACR

```
AFIS_SUBRAP4  BEGIN
                        //afişarea informaţiilor despre raport
                        subrap.WRITE 'Restantieri la o Materie'
                        generaza html pentru forma f2
IF1                     IF (clic pe butonul Executa)
                                DO EXECUTACR
IF1                      ENDIF
AFIS_SUBRAP4   END

EXECUTACR     BEGIN
                        //determinarea indexului materiei selectate
                        imat=smaterii.selectedIndex;
                        subrap.WRITE 'CAUTARE RESTANTIERI LA MATERIA:'
                        subrap.WRITE materii[imat]
                         DO afis_restantieri(imat);
EXECUTACR     END
```

Figura 12.35

❑ **Codificarea în limbajul JavaScript**

În figurile 12.36, 12.37, 12.38, 12.39, 12.40 sunt prezentate, pentru fiecare cadru în parte, documentele HTML complete.

Codul HTML pentru interfaţa cu utilizatorul

```
<html>
<head>
<title>Intranet baza de date cu studenti</title>
<script language="JavaScript">
<!---
//CREAREA TABELULUI CU MATERII
var materii = new Array("Algebra superioara", "Fizica","Chimie","Informatica", "Analiza
Functionala");
//CREAREA TABELULUI DE STUDENTI
var lista_stud=new Array();
//DEFINIREA CONSTRUCTORULUI OBIECTULUI PERSONALIZAT
function def_student(Nume,Prenume,Note){
 this.Nume=Nume;
 this.Prenume=Prenume;
 this.Note=Note;
}

//CREEAZA LISTA DE STUDENTI
var note1=new Array(5,6,7,10,10);
lista_stud[0]=new def_student("Avram","Nicolae", note1);
note1=new Array(4,9,5,7,8);
lista_stud[1]=new def_student("Berbecaru","Catalin", note1);
note1=new Array(10,9,8,10,10);
lista_stud[2]=new def_student("Botea","Ion", note1);
note1=new Array(6,8,7,7,5);
lista_stud[3]=new def_student("Cazacu","Marin", note1);
note1=new Array(9,8,0,7,8);
lista_stud[4]=new def_student("Ciufu","Gheorge", note1);
```

Figura 12.36

```
note1=new Array(8,6,8,6,5);
lista_stud[5]=new def_student("Codescu","Petre", note1);
note1=new Array(4,4,0,5,5);
lista_stud[6]=new def_student("Dinu","Mihaela", note1);
note1=new Array(5,6,5,4,4);
lista_stud[7]=new def_student("Dobre","Magdalena", note1);
note1=new Array(9,4,7,0,5);
lista_stud[8]=new def_student("Georgescu","Dragos", note1);
note1=new Array(6,8,9,6,6);
lista_stud[9]=new def_student("Ionescu","Ludovic", note1);
note1=new Array(4,0,4,4,5);
lista_stud[10]=new def_student("Mihail","Gheorghe", note1);
note1=new Array(8,7,7,8,6);
lista_stud[11]=new def_student("Nastase","Stefan", note1);
note1=new Array(7,5,8,7,8);
lista_stud[12]=new def_student("Neagu","Constanta", note1);
note1=new Array(9,9,8,8,9);
lista_stud[13]=new def_student("Oprea","Tudor", note1);
note1=new Array(8,8,6,9,9);
lista_stud[14]=new def_student("Panait","Gabriel Andrei", note1);
note1=new Array(5,8,5,6,7);
lista_stud[15]=new def_student("Patrichi","Mihai Ioan", note1);
note1=new Array(9,8,6,5,8);
lista_stud[16]=new def_student("Radoiu","Alin", note1);
note1=new Array(6,5,8,6,8);
lista_stud[17]=new def_student("Savu","Ion Mihut", note1);
note1=new Array(7,5,5,5,7);
lista_stud[18]=new def_student("Stanescu","Stelian", note1);
note1=new Array(8,5,5,4,6);
lista_stud[19]=new def_student("Stoian","Daniel", note1);
note1=new Array(6,9,8,7,8);
lista_stud[20]=new def_student("Trifu","Mihaela Elena", note1);
note1=new Array(7,5,8,7,6);
lista_stud[21]=new def_student("Ungureanu","Eliza", note1);
note1=new Array(6,6,5,6,7); lista_stud[22]=new def_student("Zaharescu","Danut", note1);

function rotunjeste(x)     {
    var s=""+x;
     i=s.indexOf(".");
     if(i!=-1){
       s=s.substring(0,i+3);
       }
     return s;
     }

//FUNCTIA CARE REALIZEAZA RAPORTUL GENERAL
function executaRG(){
 VAR I,J,K;
 //AFISARE REZULTATE PENTRU RAPORT GENERAL
fraport.document.open();
fraport.document.writeln("<center><p><b>RAPORT GENERAL </b></p></center>");
fraport.document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
fraport.document.writeln("<td bgcolor=#8bd3e9><b>NR.</b>",
```

Figura 12.36 *(continuare)*

```
"<td bgcolor=#8bd3e9><b>Nume Prenume </b></td>");
 for(i=0;i<materii.length;i++) {
    fraport.document.writeln("<td bgcolor=#8bd3e9><b>"+ materii[i]+" </b></td>");
 }
 for(i=0;i<lista_stud.length;i++) {
  k=i+1;
  if(i%2){bgc="#c0e7ed";}else{bgc="#d9f0f4";}
  fraport.document.writeln("<tr><td bgcolor="+bgc+"><b>"+k+" </b></td>");
  fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Nume+"
"+lista_stud[i].Prenume+"</b></td>");
 for(j=0;j<materii.length;j++)
  {fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Note[j]+" </b></td>");
  }
 }
 fraport.document.writeln("</tr></table>");fraport.document.close();
}
//FUNCTIA CE REALIZEAZA RAPORTUL PE MATERII
function executaRM(){
 var i,j,k;
 //AFISARE REZULTATE PENTRU RAPORT MATERIE
var imat=subrap.f2.smaterii.selectedIndex;
var iord=subrap.f2.sordine.selectedIndex;
fraport.document.open();
fraport.document.writeln("<center><p><b>RAPORT NOTE MATERIA:");
fraport.document.writeln(materii[imat],"  </b></p></center>");
 vord=new Array(lista_stud.length);
 for(i=0;i<lista_stud.length;i++)vord[i]=i;
    if(iord==1) {
 //ORDONEAZA VECTORUL DUPA NOTE
 neordonat=true;
 while (neordonat) {
   neordonat=false;
   for(i=0;i<lista_stud.length-1;i++)
    if(lista_stud[vord[i]].Note[imat]< lista_stud[vord[i+1]].Note[imat]) {
      j=vord[i]; vord[i]=vord[i+1];
      vord[i+1]=j;
      neordonat=true;
     }
  }
}
fraport.document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
fraport.document.writeln("<td bgcolor=#8bd3e9><b>NR.</b>",
 "<td bgcolor=#8bd3e9><b>Nume Prenume </b></td>");
 fraport.document.writeln("<td bgcolor=#8bd3e9><b> Nota </b></td>");
  for(i=0;i<lista_stud.length;i++) {
  k=i+1;
  if(i%2){bgc="#c0e7ed";}else{bgc="#d9f0f4";}
  fraport.document.writeln("<tr><td bgcolor="+bgc+"><b>"+k+" </b></td>");
  fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[vord[i]].Nume+"
"+lista_stud[vord[i]].Prenume+"</b></td>");
  fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[vord[i]].Note[imat]+" </b></td>");
  }
```

Figura 12.36 *(continuare)*

```
fraport.document.writeln("</tr></table>");
fraport.document.close();
}

// FUNCTIA CE REALIZEAZA AFISAREA RESTANTIERILOR LA O MATERIE
function afis_restantieri(imat){
var i,j,k,nrest;
 nrest=0;
 //DETERMINA NUMARUL DE RESTANTIERI
 for(i=0;i<lista_stud.length;i++)
   if(lista_stud[i].Note[imat]<5)nrest++;
 if(nrest>0) {
 //AFISEAZA IN TABEL STUDENTII RESTANTIERI
   fraport.document.writeln("<center>Numar restantieri:",nrest,"<br></center>");
   fraport.document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
   fraport.document.writeln("<td bgcolor=#8bd3e9><b>NR.</b>",
       "<td bgcolor=#8bd3e9><b>Nume Prenume </b></td>");
   fraport.document.writeln("<td bgcolor=#8bd3e9><b> Nota </b></td>");
   k=0;
  for(i=0;i<lista_stud.length;i++)
   if(lista_stud[i].Note[imat]<5) {
    if(k%2){bgc="#c0e7ed";}else{bgc="#d9f0f4";}
    k++;
    fraport.document.writeln("<tr><td bgcolor="+bgc+"><b>"+k+" </b></td>");
    fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Nume+"
"+lista_stud[i].Prenume+"</b></td>");
   if(lista_stud[i].Note[imat]==0)
      fraport.document.writeln("<td bgcolor="+bgc+"><b> absent </b></td>");
    else
      fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Note[imat]+" </b></td>");
   }
 fraport.document.writeln("</tr></table>");
}
 else  fraport.document.writeln("<p> Nu esista restantieri");
}

//RAPORT RESTANTIERI
function executaRR(){
 var i,j,k;
 var nrest;
fraport.document.open();
fraport.document.writeln("<center><p><b>RAPORT RESTANTIERI </b></p></center>");
for(j=0;j<materii.length;j++){
 fraport.document.writeln("<center><p><b>Materia:",materii[j]," </b></p></center>");
 afis_restantieri(j);
 }//de la for j
 fraport.document.close();
}

//CAUTARE DUPA STUDENT
function executaCS(){
 var i,j,k,nr;
```

Figura 12.36 *(continuare)*

```
 var sn=subrap.f2.snume.value;
 var pn=subrap.f2.spnume.value;
 fraport.document.open();
 fraport.document.writeln("<center><p><b>Cautare rezultate student </b></p></center>");
if(sn!=""){
 if(pn!="") {
  nr=0;
  for(i=0;i<lista_stud.length;i++)
   if((lista_stud[i].Nume==sn)&&(lista_stud[i].Prenume==pn))nr++;
  if(nr>0)  {
    fraport.document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
    fraport.document.writeln("<td bgcolor=#8bd3e9><b>NR.</b>",
       "<td bgcolor=#8bd3e9><b>Nume Prenume </b></td>");
    for(i=0;i<materii.length;i++)
     fraport.document.writeln("<td bgcolor=#8bd3e9><b>"+ materii[i]+" </b></td>");
    k=0;
  for(i=0;i<lista_stud.length;i++)
     if((lista_stud[i].Nume==sn)&&(lista_stud[i].Prenume==pn))     {
       if(k%2)  bgc="#c0e7ed";  else bgc="#d9f0f4";
       k++;
       fraport.document.writeln("<tr><td bgcolor="+bgc+"><b>"+k+" </b></td>");
      fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Nume+"
"+lista_stud[i].Prenume+"</b></td>");
      for(j=0;j<materii.length;j++)
      fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Note[j]+" </b></td>");
    }
   fraport.document.writeln("</tr></table>");
  }
  else
  fraport.document.writeln("<center><p>Nu exista nici un student cu numele:", sn,"
Prenumele:",pn);
 }
 else {     //DETERMINA NUMARUL DE STUDENTI CU NUMELE SN
  nr=0;
  for(i=0;i<lista_stud.length;i++)
   if(lista_stud[i].Nume==sn)nr++;
  if(nr>0)  {
    fraport.document.writeln("<center>Numar Studenti:",nr);
    fraport.document.writeln("<center><table border=1 bgcolor=#efefff><tr>");
    fraport.document.writeln("<td bgcolor=#8bd3e9><b>NR.</b>",
       "<td bgcolor=#8bd3e9><b>Nume Prenume </b></td>");
    for(i=0;i<materii.length;i++)
     fraport.document.writeln("<td bgcolor=#8bd3e9><b>"+ materii[i]+" </b></td>");
    k=0;
    for(i=0;i<lista_stud.length;i++)
     if(lista_stud[i].Nume==sn) {
       if(k%2)bgc="#c0e7ed";else bgc="#d9f0f4";
       k++;
       fraport.document.writeln("<tr><td bgcolor="+bgc+"><b>"+k+" </b></td>");
      fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Nume+"
"+lista_stud[i].Prenume+"</b></td>");
      for(j=0;j<materii.length;j++)
      fraport.document.writeln("<td bgcolor="+bgc+"><b>"+lista_stud[i].Note[j]+" </b></td>");
```

Figura 12.36 *(continuare)*

```
    }
   fraport.document.writeln("</tr></table>");
  }
  else
  fraport.document.writeln("<center><p>Nu exista nici un student cu numele:", sn);
 }
}// if sn

fraport.document.close();
}

// CAUTAREA RESTANTIERILOR LA O MATERIE
function executaCR(){
 var i,j,k;
 var nrest;
var imat=subrap.f2.smaterii.selectedIndex;
fraport.document.open();
fraport.document.writeln("<center><p><b>CAUTARE RESTANTIERI LA MATERIA: ");
fraport.document.writeln(materii[imat]," </b></p></center>");
afis_restantieri(imat);
 fraport.document.close();
}
// -->
</script>
</head>
 <frameset rows="10%,30%,60%">
   <frame scrolling="no" noresize src="btitlu.html" name="titlu">
   <frameset cols="30%,70%">
      <frame scrolling="no" noresize src="tipraport.html" name="inputd">
      <frame src="btiprap0.html" name="subrap">
    </frameset>
   <frame src="bsit.html" name="fraport">
</frameset>
</html>
```

Figura 12.36 *(continuare)*

Codul HTML pentru cadrul titlu

```
<html>
<head>
</head>
<body>
   <center>
   <font size=+3 color=green> Intranet- Baza de Date cu Studenti</font>
   </center>
</body>
</html>
```

Figura 12.37

Codul HTML pentru cadrul inputd

```
<html>
<head>
<script language="JavaScript">
<!---
var sel=0;
function afis_subrap0(){
parent.subrap.document.open();
//AFISAREA DATELOR GENERALE
parent.subrap.document.writeln("<font color=green size=+1>Raport general</font>");
parent.subrap.document.writeln("<hr>");
parent.subrap.document.writeln("<p>Afiseaza toata grupa si toate notele</p>");
parent.subrap.document.writeln("<p>");
//FORMAREA ELEMENTELOR DE INTERFATA -BUTONUL DE EXECUTIE AL RAPORTULUI
parent.subrap.document.writeln("<form name=\"f2\">");
parent.subrap.document.writeln("<p><input type=\"button\" value=\"Executa\"
onClick=\"parent.executaRG();\">");
parent.subrap.document.writeln("</form>");
parent.subrap.document.close();
}
function afis_subrap1(){
parent.subrap.document.open();
parent.subrap.document.writeln("<font color=green size=+1>Raport pe Materii</font>");
parent.subrap.document.writeln("<hr>");
parent.subrap.document.writeln("<table border=\"0\"><tr><td>Selecteaza materia");
parent.subrap.document.writeln(" <td> Ordonare rezultate dupa</td></tr>");
//FORMA F2 DE INTERFATA CU UTILIZATORUL
parent.subrap.document.writeln("<form name=\"f2\">");
parent.subrap.document.writeln("<tr><td><select size=\"1\" name=\"smaterii\">");
for(i=0;i<parent.materii.length;i++)
parent.subrap.document.writeln("<option>",parent.materii[i],"</option>");
parent.subrap.document.writeln("</select>");
parent.subrap.document.writeln("<td><select size=\"1\" name=\"sordine\">");
parent.subrap.document.writeln("<option>Alfabetica studenti</option>");
parent.subrap.document.writeln("<option> Descrescatoare Note</option>");
parent.subrap.document.writeln("</select>");
parent.subrap.document.writeln("<tr><td colspan=2><input type=\"button\" value=\"Executa\"
onClick=\"parent.executaRM();\">");
parent.subrap.document.writeln("</form>");
parent.subrap.document.writeln("</table>");
parent.subrap.document.close();
}

function afis_subrap2(){
parent.subrap.document.open();
parent.subrap.document.writeln("<font color=green size=+1>Raport Restantieri</font>");
parent.subrap.document.writeln("<hr>");
parent.subrap.document.writeln("<p>Afiseaza Studentii Restantieri din Grupa</p>");
parent.subrap.document.writeln("<p>");
parent.subrap.document.writeln("<form name=\"f2\">");
parent.subrap.document.writeln("<p><input type=\"button\" value=\"Executa\"
onClick=\"parent.executaRR();\">");
parent.subrap.document.writeln("</form>");
```

Figura 12.38

```
parent.subrap.document.close();
}

function afis_subrap3(){
parent.subrap.document.open();
parent.subrap.document.writeln("<font color=green size=+1>Cautare Rezultate Student</font>");
parent.subrap.document.writeln("<hr>");
parent.subrap.document.writeln("<p>");
parent.subrap.document.writeln("<form name=\"f2\">");
parent.subrap.document.writeln("<table border=\"0\"><tr><td>Nume student:");
parent.subrap.document.writeln("<td><input type=\"text\" name=\"snume\" size=\"20\">");
parent.subrap.document.writeln("<tr><td>Prenume student:");
parent.subrap.document.writeln("<td><input type=\"text\" name=\"spnume\" size=\"20\">");
parent.subrap.document.writeln("<tr><td colspan=2><input type=\"button\" value=\"Executa\"
onClick=\"parent.executaCS();\">");
parent.subrap.document.writeln("</form>");
parent.subrap.document.writeln("</table>");
parent.subrap.document.close();
}

function afis_subrap4(){
parent.subrap.document.open();
parent.subrap.document.writeln("<font color=green size=+1>Cautare Restantieri la o
Materie</font>");
parent.subrap.document.writeln("<hr>");
parent.subrap.document.writeln("<table border=\"0\"><tr><td>Selecteaza materia");
parent.subrap.document.writeln("</td></tr>");
parent.subrap.document.writeln("<form name=\"f2\">");
parent.subrap.document.writeln("<tr><td><select size=\"1\" name=\"smaterii\">");
for(i=0;i<parent.materii.length;i++)
 parent.subrap.document.writeln("<option>",parent.materii[i],"</option>");
parent.subrap.document.writeln("</select>");
parent.subrap.document.writeln("<tr><td colspan=2><input type=\"button\" value=\"Executa\"
onClick=\"parent.executaCR();\">");
parent.subrap.document.writeln("</form>");
parent.subrap.document.writeln("</table>");
parent.subrap.document.close();
}

function afisrap(t) {
  if(t==0)   {
   f1.tiprap.options[0].text ="General";
   f1.tiprap.options[1].text ="Pe Materii";
   f1.tiprap.options[2].text ="Restantieri";
   }
else {
   f1.tiprap.options[0].text ="Dupa Student";
   f1.tiprap.options[1].text ="Restantieri";
   f1.tiprap.options[2].text ="";
  }
  f1.tiprap.selectedIndex=0;
  afis_subopt(this);
 }
```

Figura 12.38 *(continuare)*

```
function afis_subopt(t){
 var ir=f1.tiprap.selectedIndex;
 if(f1.raport[0].checked) {  // ESTE SELECTAT RAPORT
  switch(ir)  {
   case 0:
     afis_subrap0();
     break;
   case 1:
     afis_subrap1();
     break;
   case 2:
     afis_subrap2();
  }
  window.status="Tip Raport";
 }
else {   // ESTE SELECTAT CAUTARE
     switch(ir)  {
   case 0:
     afis_subrap3();
     break;
   case 1:
     afis_subrap4();
     break;
  }
   window.status=”Tip Cautare”;
  }
}
// →
</script>
</head>

<body>
<center>
<form name=”f1”>
<p><font size=+2 color=green>Tipul Prelucrarii</font>
<p>
<table border=0>
<tr><td><input type=”radio” name=”raport” checked value=”0” onClick=”if (this.checked)
{afisrap(,0’)}” >
 <b>RAPOARTE</b>
<td>
<input type=”radio” name=”raport” value=”1” onClick=”if (this.checked) {afisrap(,1’)}” >
<b>CAUTARE</b></tr>
<tr><td colspan=2><select size=”3” name=”tiprap” onChange=”afis_subopt(this)”>
  <option selected>General</option>
  <option>Pe Materii</option>
  <option>Restantieri</option>
  </select>
</table>
</center>
</form>
</body>
</html>
```

Figura 12.38 *(continuare)*

Codul HTML pentru cadrul subrap

```
<html>
  <head>
   <title>Aplicatie</title>
  <head>
<body>
   <font color=green size=+1>Raport general</font>
  <hr>
  <p>Afiseaza toata grupa si toate notele</p>
   <form name="f2">
  <input type="button" value="Executa" onClick="parent.executaRG();">
   </form>
</body>
</html>
```

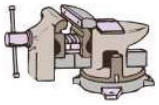
Figura 12.39

Codul HTML pentru cadrul fraport

```
<html>
<head>
</head>
<body>
  <h3>Rezultatele studentilor</h3></body>
</html>
```

Figura 12.40

Testați-vă cunoștințele



1. Cum creaţi obiecte fiu personalizate. Exemple.
2. Cum creaţi matrici de obiecte personalizate?
3. Care sunt navigatoarele care recunosc obiectele personalizate?
4. Care este cuvântul cheie JavaScript care permite crearea unei instanţe a unui obiect:
   - `instance`;
   - `object`;
   - `new`.
5. Care este semnificaţia cuvântului cheie `this`?
   - obiect curent;
   - script curent;
   - nici a, nici b.
6. Care este diferenţa dintre o funcţie constructor şi o funcţie ordinară?

7. Cum puteţi avea acces la obiectele pe care le-aţi creat?
8. Care este diferenţa dintre o metodă şi o funcţie?
9. Care sunt etapele pe care trebuie să le parcurgeţi pentru a defini o metodă pentru un obiect personalizat?

Vizitați site-urile

- ✓ http://www.wdvl.com
- ✓ http://msdn.microsoft.com/library
- ✓ http://www.databasejournal.com
- ✓ http://hotwired.lycos.com/webmonkey/programing

BIBLIOGRAFIE

1. Richard **Wagner**, R. Allen **Wyke**, *JavaScript*, traducere de Cora Radulian şi Dan Pavelescu, Editura Teora, 2001, Bucureşti
2. Jean-Paul **Mesters**, *Aide-mémoire JavaScript,* OEM-Eyrolles, Paris, 2003
3. Michael **Moncur**, *JavaScript 1.5*, CampusPress, Paris, 2002
4. Jean-Paul **Mesters**, *JavaScript, Exercices et corriges,* Collection L'Atelier, EM-Eyrolles, Paris, 2003
5. Cédric **Nilly**, Jean-Christophe **Gigniac**, *JavaScript*, MicroApplication, Paris, 2003
6. Jean-Christophe **Gigniac**, Cédric **Nilly**, *JavaScript,* e-Poche, Paris, 2002
7. Emily A. **Vander Veer**, *JavaScript pour les nuls*, Editions First Interactive, Paris, 2002
8. Steven W. **Disbrow**, *JavaScript Web Tr@ining,* OEM, Paris, 2002
9. Mike **Robertshaw**, *Web Site Design (U234)*, The Open University of Hong Kong, 2002
10. Liviu **Dumitraşcu**, *Învăţăm ... BASIC conversând cu calculatorul*, Editura Tehnică, 2 volume, 1989
11. Floarea **Năstase**, Pavel **Năstase**, *Tehnologia aplicaţiilor Web (XML-DOM-ASP)*, Editura Economică, 2002
12. Liviu **Dumitraşcu**, *(X)HTML,* Editura Universităţii din Ploieşti, 2003
13. Liviu **Dumitraşcu**, *Dreamweaver MX,* Editura Universităţii din Ploieşti, 2003
14. Liviu **Dumitraşcu**, *XML,* Editura Universităţii din Ploieşti, 2003
15. Michel **Drewfus**, *HTML 4,* Student Edition, Campus Press, Paris, 2003