UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI
Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Interacțiunea om-calculator

26. Proiectare si implementare de interfete in JavaScript

JAVASCRIPT & AJAX

1. Introducere

Ce este JavaScript? JavaScript este un limbaj de scripting folosit în general ca limbaj de scripting client-side pentru aplicaţii web (codul Javascript este inclus in pagini şi este interpretat de browserul clientului). JavaScript poate fi folosit si in alte contexte in afara de acesta (exista si o versiune JavaScript server-side) dar îşi datorează popularitatea versiunii client-side.

JavaScript a fost creat în 1995 de Brendan Eich (Netscape) pentru a aduce un plus de dinamism paginilor de web. Numele iniţial al limbajului a fost LiveScript şi a fost creat cu scopul de a oferi un limbaj simplu de scripting pentru a aduce modificări dinamic paginilor web html pe partea de client (de unde şi „Live"). Deoarece înainte de lansarea limbajului, Java era la apogeul popularităţii, s-a decis schimbarea numelui din LiveScript în JavaScript din motive comerciale. Totuşi, Java şi JavaScript au puţine lucruri în comun.

Faptul că browserul este responsabil de interpretarea codului JavaScript a adus unele dezavantaje. În primii ani de viaţă utilizarea JavaScript era foarte dificilă, datorită lipsei unui standard, datorită evoluţiei rapide în lumea browserelor şi mai ales datorită evoluţiei pe drumuri divergente a principalilor lor producători (Microsoft, Netscape şi Opera). Codul scris în JavaScript pentru Internet Explorer, Netscape şi Opera nu producea decât rareori acelaşi rezultat. Existau deseori diferenţe mari şi între rezultatele produse de versiuni succesive ale aceluiaşi browser. Din aceste motive, majoritatea scripturilor trebuiau scrise în mai multe variante similare ceea ce era deosebit de neplăcut pentru dezvoltatori. De asemenea, a durat mult timp până au apărut unele medii de dezvoltare şi depanare avansate. Astfel a durat destul de mult până când tehnologia a ajuns la maturitate şi a devenit destul de avansată pentru a fi folosită pe scară largă.Odată cu apariţia Web 2.0 şi AJAX, JavaScript a crescut în popularitate şi astăzi, o gamă largă de aplicaţii web (Google Mail, Yahoo Mail) folosesc această tehnologie.

2. JavaScript

Scripturile Javascript se execută de către browser şi sunt incluse deci în pagina HTML ce se afişează pe calculatorul clientului. Scripturile pot fi incluse complet în pagina HTML sau pot fi stocate în fişiere separate şi referite în pagina HTML. În ambele cazuri, marcajul HTML folosit este <script>, exemplele de includere fiind următoarele:

- Script inclus în pagina HTML
  <script type="text/javascript">
  //cod script
  </script>
- Script păstrat într-un fişier extern
  <script src="fisier_sursa.js"></script>

Marcajul <script> poate fi inclus atât în interiorul marcajului <head>, cât şi în cadrul marcajului <body>. Diferenţa este că în primul caz scriptul se execută la încărcarea paginii, în timp ce în al doilea caz se execută în momentul întâlnirii marcajului. Din acest motiv, în secţiunea <head> sunt incluse funcţiile ce vor fi folosite în restul paginii iar în <body> sunt în general apelurile funcţiilor.

Un script JavaScript poate conţine definiţii de funcţii, definiţii de clase (cu menţiunea că JavaScript nu este un limbaj orientat obiect în adevăratul sens al cuvântului neavând o mare parte din mecanismele unui limbaj orientat obiect), apeluri ale funcţiilor definite sau ale funcţiilor oferite de browser.Sintaxa JavaScript este foarte asemănătoare cu sintaxa Java. Cuvintele cheie sunt cu mici excepţii aceleaşi, diferenţele majore de sintaxă fiind evidenţiate în prima parte a acestei secţiuni. Pentru o referinţă completă a elementelor limbajului JavaScript consultaţi [1].

Javascript este un limbaj cu tipare dinamică, verificarea tipului datelor efectuându-se la rulare. Astfel, la declararea unei variabile nu se va specifica tipul acesteia ci doar că este vorba de o variabilă. Se foloseşte cuvântul cheie ***var***:

```
var x=2, y=5;
```

Declararea unei funcţii se face folosind cuvântul cheie ***function*** urmat de numele funcţiei, de lista de parametri şi de un bloc ce conţine codul funcţiei. Ca şi în Java, cuvântul cheie ***return*** este folosit pentru a întoarce rezultatul funcţiei.

Un exemplu de definire şi de apel al unei funcţii ar fi următorul:

```
function sum(x,y)
{
var rez=x+y;
return rez;
}
var suma=sum(2,5);
```

Metodele de iterare în Javascript sunt aproape identice cu cele din Java. Sintaxa pentru instrucţiunile ***for, while*** şi ***do..while*** este identică cu cea din Java. În plus faţă de

Java, Javascript oferă instrucţiunea ***foreach*** ce iterează pe proprietăţile unui obiect spre deosebire de alte limbaje de programare (C#, PHP) unde ***foreach*** iterează pe elementele unei colecţii.

Următorul exemplu ilustrează definirea unei funcţii în care se iterează proprietăţile unui obiect şi în care se şi definesc şi se setează valori pentru un obiect.

```
function printNote(stud)
{
var nota
document.write("note pentru studentul "+stud.nume+":<br>");
for each (nota in stud.note)
  document.write(nota+" ");
}

var student={nume: "ion", an:2, note:{mate:9, pc:8}};
document.write(student.nume +"<br>" );
document.write(student.an +"<br>");
printNote(student)
```

Instrucţiunea document.write afişează un şir de caractere în documentul HTML curent (în care se află script-ul). Despre obiectul document vom discuta mai pe larg în secţiunea următoare.

Javascript pune la dispoziţie şi un număr mare de obiecte ce pot fi folosite în marea majoritate a browserelor, printre care ***Array, Math, Date, String***. Mai multe detalii despre aceste obiecte pot fi găsite la [1].

Un exemplu rapid de afişare a datei curente este următorul:

```
var data=new Date();// creez un obiect de tip Date
document.write("Documentul s-a incarcat ultima oara la data: " + data.getDate()
+":"+data.getMonth()+ ":"+ data.getFullYear());// folosind metodele asociate acestui obiect, afisez
ziua, luna si anul
```

Observaţie! Data afişată de scriptul de mai sus este dependentă de data setată pe calculatorul pe care se execută scriptul.

3. JavaScript şi DOM

Javascript este utilizat în special pentru a modifica modul de afişare sau conţinutul unei pagini web.

Cum anume putem modifica conţinutul unei pagini web?După încărcarea paginii de pe server, codul JavaScript din pagină poate accesa şi modifica structura documentului afişat. În acest scop este utilizat DOM (v. Lab4).

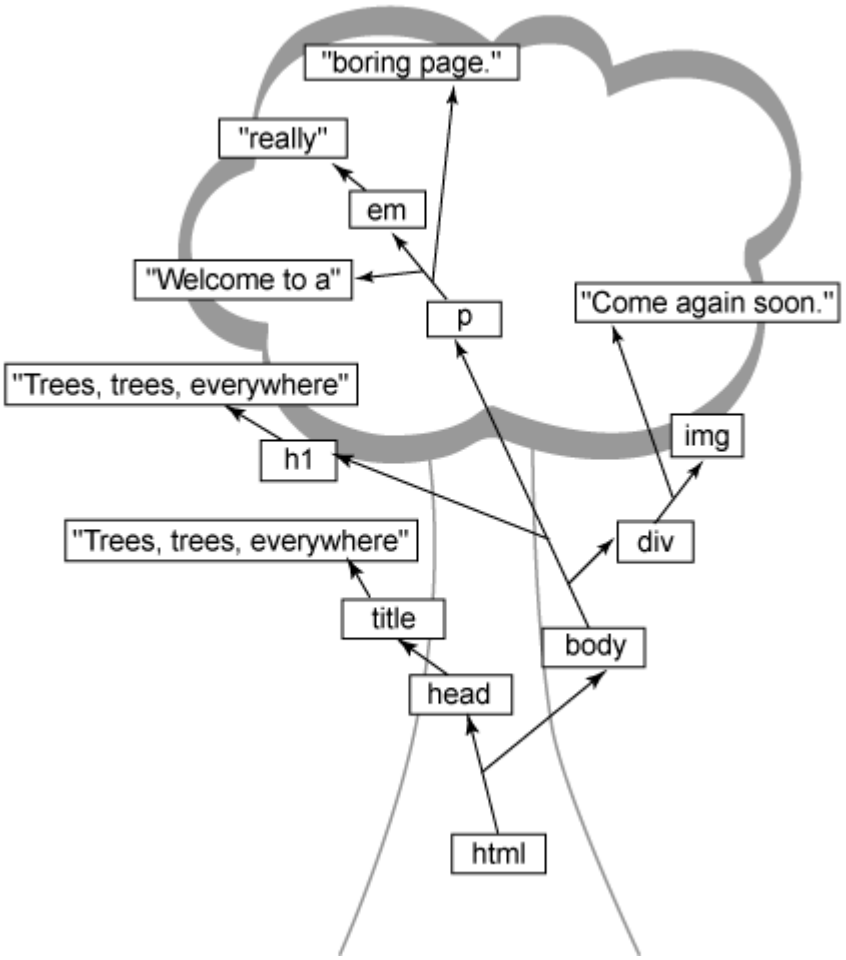
DOM (Document Object Model) reprezintă un API standardizat de W3C pentru a manipula documente HTML sau XML valide [2].

Concret, ce înseamnă acest lucru?

Să presupunem că în pagina HTML avem următorul cod

```
<html>
 <head>
  <title>Trees, trees, everywhere</title>
 </head>
 <body>
  <h1>Trees, trees, everywhere</h1>
  <p>Welcome to a <em>really</em> boring page.</p>
  <div>
    Come again soon.
    <img src="come-again.gif" />
  </div>
 </body>
</html>
```

Interpretarea internă a browserului pentru acest cod va fi o structură arborescentă similară cu cea de mai jos, cunoscută sub numele de arbore DOM al paginii.



Fiecare element al arborelui DOM are asociată o listă de atribute şi evenimente (acţiuni).

Prin intermediul Javascript putem accesa dinamic şi manipula arborele DOM al paginilor web , având posibilitatea de a modifica proprietăţile şi comportamentul fiecărui element din acesta.

Obiectul rădăcină al arborelui DOM este ***document.*** Prin intermediul acestui obiect putem accesa orice alt obiect sau marcaj din document. Lista completă a metodelor şi proprietăţilor acestui obiect poate fi găsită la [3].

În secţiunea precedentă am folosit metoda ***write*** pentru a scrie un şir de caractere în locaţia curentă a documentului. O altă metodă foarte des utilizată a acestui obiect este ***getElementById(id)***. Această metodă întoarce nodul (marcajul) care are marcajul ***id.*** În cazul în care sunt folosite id-uri în document putem găsi foarte rapid un anumit nod. O dată găsit acest nod putem să-i accesăm sau să-i schimbăm proprietăţile.

Exemplul următor arată cum putem schimba textul din interiorul unui element de tip ***div*** şi, de asemenea, cum putem apela cod Javascript în momentul declanşării unui eveniment de către utilizator.

```
<script>
function f1()
{
var nodDiv=document.getElementById("a1");// selectez elementul DOM cu atributul id = a1
nodDiv.innerHTML="text schimbat";// setez textul din interiorul acestui element

}
function f2()
{
var nodDiv=document.getElementById("a1");");// selectez elementul DOM cu atributul id = a1

nodDiv.innerHTML="text";// setez textul din interiorul acestui element

}
</script>

// atasez cele 2 functii ca evenimente de mouse
<div id="a1" onmouseover="f1()" onmouseout="f2()"> text </div>
```

Toate elementele unei pagini web au asociate o listă de evenimente pe care le pot recepţiona. O listă exhaustivă a acestor evenimente poate fi găsită la [4]. Un element oarecare nu poate recepţiona toate aceste evenimente. Pentru a vedea ce evenimente pot fi gestionate de un anumit element trebuie consultată referinţa în DOM pentru elementul respectiv.

Două exemple de evenimente destul de comune sunt ***onmouseover*** care se declanşează atunci când cursorul mouse-ului intră în zona elementului şi ***onmouseout*** care se declanşează când cursorul mouse-ului părăseşte zona elementului. Fiecărui astfel de tip de eveniment i se poate asocia un cod Javascript. În exemplul de mai sus, fiecărui eveniment i-a fost asociată o funcţie care îi modifică valoarea. În cele două funcţii se accesează elementul căutat folosindu-i-se id-ul şi i se modifică valoarea conţinutului HTML.

Metoda folosită în acest caz nu este foarte bună deoarece funcţiile ***f1*** şi ***f2*** nu pot fi folosite decât pentru un singur element (cel cu id-ul „a1”). Exemplul poate fi modificat pentru a trimite prin lista de parametri elementul ce trebuie modificat de funcţie.

```
<script>
function f1(nodDiv)
{
nodDiv.innerHTML="text schimbat";// setez textul din interiorul elementului primit ca parametru

}
function f2()
{
nodDiv.innerHTML="text";
}
</script>

// atasez cele 2 functii ca evenimente de mouse , avand ca parametru elementul curent
<div id="a1" onmouseover="f1(this)" onmouseout="f2(this)"> text </div>
```

Se observă folosirea cuvântului cheie ***this*** în apelul funcţiilor. ***This***, ca şi în Java este o referinţă la elementul curent şi apelând funcţia cu parametrul ***this*** se transmite spre prelucrare obiectul asociat marcajului curent.

Următorul exemplu, ceva mai practic şi puţin mai complicat, exemplifică validarea unui formular cu ajutorul Javascript.

```
<!—— script de validare ——>
<script type="text/javascript">
function f1()
{
var nodForm=document.getElementById("f1");// selectez elementul DOM cu atributul id = f1

var nodDiv=document.getElementById("d1");// selectez elementul DOM cu atributul id = d1

if(nodForm.mesaj.value.length<5) // daca lungimea textului din elementul cu id-ul mesaj din
interiorul formului selectat este mai mica de 5 caractere, se afiseaza un mesaj de eroare
// altfel, se trimite formularul la server

   nodDiv.innerHTML="mesajul trebuie sa aiba minim 5 caractere"

else {nodForm.action="test2.php"; nodForm.submit();}
}
</script>

<!—— formular simplu, cu un input si un buton ——>
<form id="f1" method="get">
<input type="text" length="20" id="mesaj">
<input type="button" value="trimite" onclick="f1()">
</form>

<!—— zona de afisare a erorilor ——>
<div id="d1"></div>
```

În secţiunea de cod HTML adăugăm un formular simplu ce conţine un câmp de tip text şi un buton. Evenimentului ***onclick*** al butonului îi asociem o funcţie f1 ce va valida câmpul mesaj al formularului şi în cazul în care acesta este valid (are cel puţin 5 caractere) formularul se trimite mai departe. Altfel, formularul nu se trimite şi se afişează un mesaj de eroare.

4. YUI

Există pe internet un mare număr de componente Javascript open-source ce oferă numeroase soluţii pentru adăugarea de funcţionalităţi suplimentare paginilor web sau doar un plus de interactivitate sau de dinamism. Printre cele mai cunoscute frameworkuri JavaScript se numară prototype, mootools, script.aculo.us, jQuery, YUI.

Ne vom opri puţin asupra funcţionalităţilor oferite de Yahoo User Interface Library (YUI, [5]). YUI oferă un mare număr de componente (împreună cu o documentaţie consistentă) pentru creşterii vitezei de dezvoltare a unei pagini web interactive. Câteva din funcţionalităţile oferite de componentele YUI care vă pot uşura eforturile depuse pentru obţinerea unor funcţionalităţi uzuale pentru paginile voastre web sunt

- Drag and drop
- Paginator
- Rich Text Editor
- Color Picker
- Image Loader

YUI pune la dispozitia dezvoltatorilor diferite clase care implementează aceste funcţionalităţi precum şi o documentaţie consistentă (foarte utilă - YUI fiind destul de complex, nu este şi cel mai intuitiv framework).

Următoarele exemple ilustrează câteva din facilităţile pe care le oferă YUI.

Exemplul 1: Joaca de-a şoarecele şi pisica într-o pagină web [6].

```
<script type="text/javascript">
function YahooAnimate(offL,offT)
{
  if(offL>250)
  {offL=-300; offT=-300;}

     var attributes = {
     points: { to: [offL+300, offT+300]}
  };
  var anim = new YAHOO.util.Motion('tinta', attributes,0.5);
  anim.animate();
}
</script>

<div id="tinta" onmouseover="YahooAnimate(this.offsetLeft,this.offsetTop);" >apasa aici!</div>
```

Scriptul de mai sus animează un element de tip div cu textul „apasă aici” ce are id-ul „ţintă” astfel încât să fugă de mouse. Animaţia este foarte simplă – se apelează la declanşarea evenimentului **onmouseover** şi primeşte ca parametrii poziţia elementului curent şi lansează o nouă animaţie spre altă poziţie de pe ecran.

Tot ce trebuie să specifice programatorul ce utilizează această bibliotecă este id-ul elementului ce trebuie animat, parametrii animaţiei din vectorul attribute şi durata animaţiei. Acest exemplu necesită includerea următoarelor fişiere din bibliotecă:

- yahoo/yahoo.js
- dom/dom.js
- event/event.js
- animation/animation

Exemplul 2: afişarea unui calendar într-o pagină web [7]

```
<div id="container"></div>
<script type="text/javascript">
  function calendarInit() {

  var schedule=new Array();//initializam un vector cu evenimente
  var d0=new Date();//initializam un obiect de tip Date
  d0.setFullYear(2007,11,15);
  schedule[0]={data:d0, eveniment:"partial IE"};

  calendar = new YAHOO.widget.Calendar("calendar","container");
//functia care creeaza calendarul si-l plaseaza in elementul html //cu id-ul dat ca al doilea parametru

  var mySelectHandler = function(type,args,obj) {
//functia care se apeleaza la selectarea de catre utilizator a unei //date din calendar

                var dates = args[0];
                var date=dates[0];
                var nodDiv=document.getElementById("eveniment");
                nodDiv.innerHTML=date[0]+":"+date[1]+":"+date[2];

                for(i=0;i<schedule.length;i++)
                {
                        var ev=schedule[i];

  if(ev.data.getFullYear()==date[0]&&ev.data.getMonth()==date[1]&&ev.data.getDate()==date[2])
//daca am gasit vreun eveniment il tiparesc
                nodDiv.innerHTML+=": "+ev.eveniment;
                        }
                };
                calendar.selectEvent.subscribe(mySelectHandler, calendar, true);
                calendar.render();
  }
  YAHOO.util.Event.onDOMReady(calendarInit);
</script>

<!—— zona de afisare a evenimentelor ——>
<div id="eveniment"></div>
```

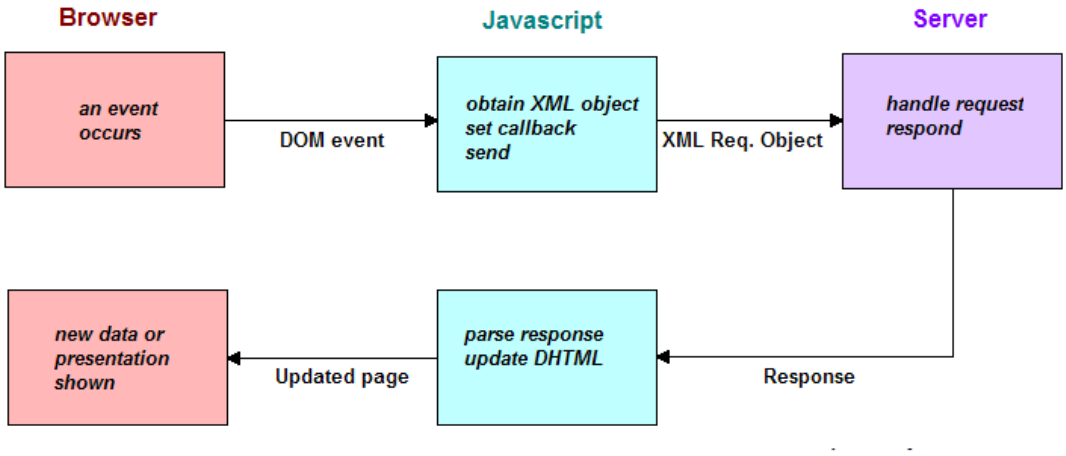
Scriptul afişează un calendar în cadrul marcajului div cu id-ul „container”. În plus, având o listă de evenimente date într-un vector de evenimente (schedule), la click pe o anumită dată se verifică dacă există vreun eveniment în ziua respectivă şi dacă există se afişează evenimentul. În capitolul următor vom vedea cum putem să afişăm acest program în mod dinamic pe baza înregistrărilor dintr-o bază de date.

5. AJAX

AJAX – Asynchronous Javascript and XML este o tehnică de programare web care permite efectuarea unor cereri http către serverul web, prin intermediul cărora se poate actualiza o pagină web fără a se efectua reîncărcarea sa completă.
AJAX foloseşte o combinaţie de :

- CSS, pentru stilul de afişare al elementlor
- Document Object Model acesat prin intermediului unui limbaj de scripting client-side precum JavaScript pentru a interacţiunea cu pagina şi afişarea dinamică a informaţiei în pagină
- Obiectul XMLHttpRequest pentru schimbul asincron de date cu serverul web
- XML ca format de transfer al datelor trimise în comunicaţia client-server (nu este însă unicul format suportat)

Cum funcţionează? La anumite evenimente din pagină sau la un anumit interval de timp, se apelează un obiect de tip XMLHttpRequest care va cere anumite date de la server iar la primirea acestora, le va trimite pentru a fi afişate în pagină fără ca aceasta să se reîncarce, doar prin modificarea unor elemente ale paginii. Formatul în care sunt primite şi trimise datele poate fi XML, JSON sau chiar text.
Posibilitatea de a reîncărca datele din pagină oferă o experienţă utilizator mult îmbunătăţită prin minimizarea timpului de aşteptare.



Obiectul Javascript care permite efectuarea acestor cereri asincrone se numeşte XMLHttpRequest şi specificaţiile sale pot fi găsite la [8], unde W3C încearcă să definească un standard pentru acest obiect.

Deşi comportarea acestui obiect este standard în fiecare browser, iniţializarea sa este diferită pentru fiecare browser. Iată codul standard pentru creearea unui obiect de tip XMLHttpRequest aşa cum este prezentat în [9] şi [10].

```
function ajaxFunction()
 {
 var xmlHttp;
 try
  {
  // Firefox, Opera 8.0+, Safari
  xmlHttp=new XMLHttpRequest();
  }
 catch (e)
  {
  // Internet Explorer
  try
   {
   xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
   }
  catch (e)
   {
   try
    {
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
   catch (e)
    {
    alert("Your browser does not support AJAX!");
    return false;
    }
   }
  }
 }
```

Principalele metode, proprietăţi şi evenimente oferite de XMLHttpRequest sunt:

- open – creează o conexiune GET sau POST către un url dat ca parametru
- send – efectuează cererea către server
- onreadystatechange – evenimentul care este declanşat de schimbarea valorii proprietăţii readystate, proprietate ce poate avea următoarele valori (conform standardului):
  - UNSENT = 0
  - OPENED = 1
  - HEADERS_RECEIVED = 2
  - LOADING = 3
  - DONE = 4

Vom încerca să exemplificăm funcţionarea Ajax în 2 cazuri şi anume în cazul în care cererea HTTP este declanşată de un eveniment din pagina web (selectarea unei date a calendarului) şi de cazul în care cererea HTTP se face periodic pentru a verifica de

exemplu dacă s-a modificat ceva în baza de date de pe server şi trebuie notificat utilizatorul.

Exemplul 1: calendar ce obţine datele referitoare la programul utilizatorului de pe server şi actualizează dinamic pagina web.
În plus faţă de exemplul precedent, care foloseşte componenta calendar a YUI ,în acest script se iniţializează un obiect XMLHttpRequest care deschide o conexiune asincronă către un script php căruia îi trimite data selectată de utilizator.
În scriptul php se extrage din baza de date (de exemplu) programul utilizatorului pentru ziua primită parametru şi acest program se întoarce formatat într-un fişier XML. Acest rezultat este apoi afişat în pagina web când se detectează finalizarea cererii.

```
var xmlHttp;
   function calendarInit() {
calendar = new YAHOO.widget.Calendar("calendar","container");
//functia care se apeleaza la selectarea de catre utilizator a unei date din calendar
   var mySelectHandler = function(type,args,obj) {
                        var dates = args[0];
                        var date=dates[0];
                        xmlCall(date[0],date[1],date[2]);
            };
            calendar.selectEvent.subscribe(mySelectHandler, calendar, true);
            calendar.render();
   }
   YAHOO.util.Event.onDOMReady(calendarInit);


   function xmlCall(year, month, day){
//primim parametrul ziua, luna si anul selectate si cerem de pe //server, programul utilizatorului
pentru ziua respectiva
xmlHttp=createXMLHttp();//creem obiectul necesar comunicarii cu serverul
   var
connString="http://localhost/testAjax2/getCalendarEvent.php?year="+year+"&month="+month+"&
day="+day; //url-ul scriptului php folosit+parametrii necesari pt executia //scriptului

//functia ce va fi apelata la schimbarea starii obiectului xmlHttp
   xmlHttp.onreadystatechange=displayChange;
/*deschidem conexiunea; true semnifica faptul ca este o conexiune asincrona si ca scriptul javascript
isi continua executia fara a astepta raspunsul; cand vine raspunsul de la scriptul php apelat se
apeleaza functia displayChange*/
   xmlHttp.open("GET",connString,true);
   xmlHttp.send(null);
   }

   function displayChange()
   {
   var nodDiv=document.getElementById("eveniment");
   if (xmlHttp.readyState==4) /*daca rezultatul cererii s-a primit complet*/
            {
                        if(xmlHttp.status==200) /*daca raspunsul cererii http a fost ok*/
                        {
                                    var eveniment=xmlHttp.responseText; /*afisam rezultatul*/
                                    nodDiv.innerHTML="eveniment:" +eveniment;
```

```
                }
            }
        }
```

Scriptul php ar putea arăta astfel:

```
<? function getEvent($data)
 {
   // select din DB
   return "rezultat";/*intorc un text oarecare pt ca exemplul sa fie functional ☺*/
 }

 header('Content-Type: text/xml');
 echo'<?xml version="1.0">'; /*construim un document XML*/
 echo'<response>';
 $data=mktime(0,0,0,$_GET['month'],$_GET['day'],$_GET['year']);
 echo getEvent($data); /*obtinem evenimentul asociat datei primite si-l tiparim*/
 echo'</response>';?>
```

Exemplul 2: Actualizarea „live” a unei liste de mesaje folosind Ajax.

Într-o pagină web avem un tabel ce conţine ştiri extrase dintr-o bază de date. Baza de date poate fi actualizată cu ştiri noi în timp ce utilizatorul vizitează pagina. Scopul exemplului este ca şi lista de ştiri „noi” să fie actualizată pe pagină fără ca utilizatorul să reîncarce pagina.
Aplicaţia constă într-un script Javascript, un script php şi o pagină HTML.

```
/*Scriptul js*/
var xmlHttp;
xmlHttp=createXMLHttp(); //creem obiectul necesar comunicarii cu serverul


   function xmlCall()
   {
            if (xmlHttp.readyState==4 || xmlHttp.readyState==0)
            {/*deschid conexiune catre pagina php ce imi va construi rezultatul*/
                     var connString="http://localhost/testAjax2/latestNews.php";
                     xmlHttp.onreadystatechange=displayChange;
                     xmlHttp.open("GET",connString,true);
                     xmlHttp.send(null);
                     }
            /*mai verific stiri noi peste 10 secunde; se poate seta acest timeout la orice valoare in
functie de aplicatia pe care o scriem si de cat de des ni se pot actualiza datele*/
            setTimeout('xmlCall()',10000);
   }

   function displayChange()
   {
   /*tabelul cu date pe care trebuie sa-l completez*/
   var nodT=document.getElementById("t1");
   /*sterg randurile existente in tabel cu exceptia headerului*/
   while(nodT.rows.length>1)
            nodT.deleteRow(nodT.rows.length-1);
   if (xmlHttp.readyState==4) /*daca rezultatul cererii s-a primit complet*/
            {
                     if(xmlHttp.status==200) /*daca raspunsul cererii http a fost ok*/
```

```
                        {
                                var response=xmlHttp.responseXML;
                                var newsList=response.getElementsByTagName("newsItem");
                                /*prelucrez fisierul xml primit ca rezultat si adaug inregistrarile in
tabel*/
                                for (i=0;i<newsList.length;i++)
                                {
                                        var noRows=nodT.rows.length;
                                        nodT.insertRow(noRows);
                                        var currentRow=nodT.rows[noRows];
                                        var cell=currentRow.insertCell(0);
                                        cell.innerHTML=newsList[i].firstChild.nodeValue;
                                        cell=currentRow.insertCell(1);
                                        cell.innerHTML=new Date();
                                }
                        }
                }
   }
/*scriptul php – latestNews.php*/
function make_seed()
{/*functie preluata de pe php.net [12]*/
  list($usec, $sec) = explode(' ', microtime());
  return (float) $sec + ((float) $usec * 100000);
}
function getNewsItems()
{/* generez un vector de stiri – in mod normal aceste stiri se iau din baza de date dar preluarea
acestora din baza de date nu face obiectul acestui document*/
   $newsArray=Array();
   $newsArray[0]['title']="Bulgaria-Romania 1-0";
   $newsArray[1]['title']="Scotia-Italia 1-2";
   srand(make_seed());
   $goalsB=rand(1,5);
   $goalsA=rand(0,3);
/*generez si o stire care sa se schimbe de fiecare data pentru a fi vizibile modificari in pagina*/
   $newsArray[2]['title']="Brazilia-Anglia ".$goalsB."-".$goalsA;
   return $newsArray;
}


header('Content-Type: text/xml');
echo'<?xml version="1.0"?>';
echo'<response>';
$news=getNewsItems();
foreach ($news as $newsItem)
   {/* construiesc un fisier xml rezultat*/
            echo '<newsItem>'.$newsItem['title'].'</newsItem>';
   }
echo'</response>';
?>
```

În fişierul HTML se vor adăuga următoarele linii:

```
<body onload="xmlCall();">

<!—— Tabel cu 2 coloane, una pentru continut, alta pt data ——>
```

```
<table id="t1" border="1">
  <tr>
        <td>Continut stire</td>
        <td>Data aparitiei</td>
  </tr>
  </table>

<input type="button" onclick="xmlCall();" value="Refresh">
```

Prima linie specifică faptul că funcţia se va apela la încărcarea paginii. În tabelul cu id-ul „t1" se vor adăuga datele primite de la server. Elementul input permite reîncărcarea tabelului la dorinţa utilizatorului prin apelarea funcţiei XmlCall (în cazul în care nu este mulţumit de intervalul de timp la care se face reîncărcarea). Pe acest principiu funcţionează şi noul Yahoo Mail Beta – e-mailurile se verifică la un interval stabilit de timp folosind AJAX sau la apăsarea butonului „Check Mail".

Aceste două exemple ilustrează puterea tehnologiei AJAX precum şi posibilitatea de a integra componente externe (YUI) cu AJAX.

6. Instrumente de dezvoltare

Instrumentele de dezvoltare sunt aceleaşi cu cele alese pentru dezvoltare web – Eclipse, Dreamweaver, Notepad, etc. Instrumentele ce par însă esenţiale pentru dezvoltarea unei aplicaţii ce foloseşte Javascript sunt depanatoarele (debuggerele). Pentru că este un limbaj de scripting cu tipare dinamică şi datorită diferenţelor încă existente între browsere, erorile se strecoară destul de uşor şi sunt relativ greu de detectat fără instrumentele potrivite. În timp ce IE are asociat Microsoft Script Debugger, care poate ajuta pentru detectarea erorilor ce apar sub IE, pentru Firefox trebuie instalat Firebug [13]. În schimb, spre deosebire de Script Debugger, Firebug rulează în browser şi pe lângă posibilitatea de depanare (se pot seta breakpoints, watches, etc.) afişează în timpul rulării sau în timpul afişării unei pagini scriptului o serie de date importante cum ar fi structura DOM a paginii curente, arborele de elemente HTML, clasele CSS, erorile de sintaxa Javascript într-o consolă, precum şi conexiunile efectuate împreună cu timpul de încărcare.

Mai jos se pot vedea 2 capturi de ecran cu secţiunea de depanare şi cu secţiunea în care se prezintă arborele DOM.

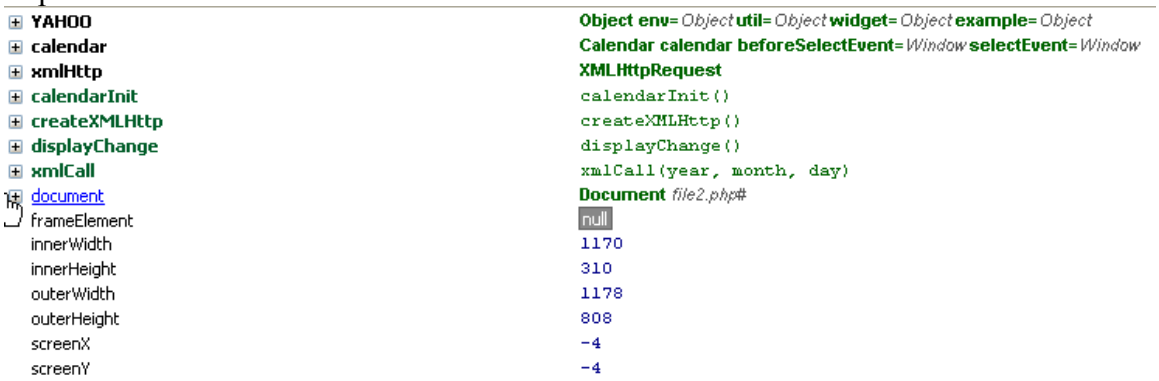


Figure 1 Arborele DOM al unei pagini web aşa cum este afişat în Firebug

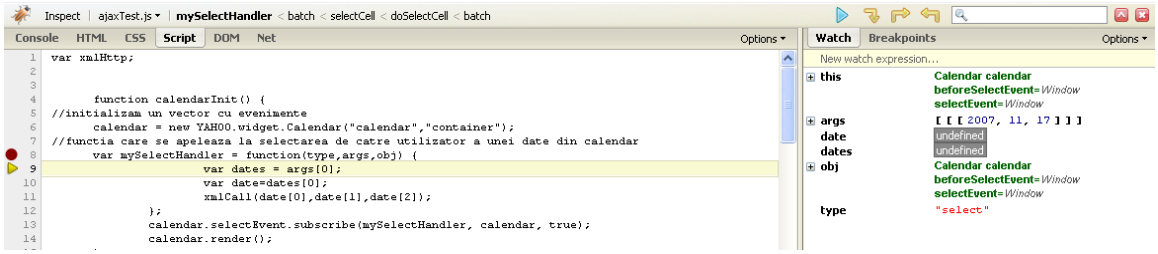


Figure 2 Fereastra de debugging pentru scripturi Javascript

7. Concluzii

Javascript reprezintă un limbaj matur pentru dezvoltarea aplicaţiilor web dinamice. Avantajele folosirii Javascript în paginile unui sit web sunt: un plus de interactivitate cu utilizatorul, posibilitatea de a actualiza informaţiile de pe pagină în timp real fără a o reîncărca (folosind Ajax). Printre dezavantajele folosirii Javascript s-ar putea număra: posibilă comportare neaşteptată în unele browsere, probleme cu motoarele de căutare care nu indexează întotdeauna paginile generate folosind Ajax.

În general este bine să se folosească Javascript având grijă:

- să se testeze paginile în cele mai importante browsere de pe piaţă şi să fie informat utilizatorul în cazul în care aplicaţia dezvoltată nu este compatibilă cu browserul lor
- să nu se actualizeze folosind Javascript şi Ajax zonele de conţinut ale site-ului ce se doresc a fi indexate de motoarele de căutare

8. Bibliografie

1. Javascript http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference
2. DOM http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/
3. DOM http://developer.mozilla.org/en/docs/DOM:document
4. DOM http://www.w3schools.com/htmldom/dom_obj_event.asp
5. YUI http://developer.yahoo.com/yui/
6. YUI animation http://developer.yahoo.com/yui/animation/
7. YUI calendar http://developer.yahoo.com/yui/calendar/
8. XMLHttpRequest http://www.w3.org/TR/XMLHttpRequest/
9. Ajax http://www.w3schools.com/ajax/ajax_browsers.asp
10. C. Darie, B. Brînzărea, Filip Cherecheş-Tosa, M. Bucică – „AJAX and PHP: Building Responsive Web Applications”. Packt Publishing 2006
11. XMLHttpRequest http://developer.mozilla.org/en/docs/XMLHttpRequest
12. PHP http://www.php.net/manual/en/function.srand.php
13. FireBug http://www.getfirebuSg.com/