Limbajul JavaScript

Mihai Gabroveanu

Ce este JavaScript?

- JavaScript este un limbaj de scripting client-side (ruleaza la nivelul browser-ului)
- JavaScript a fost dezvoltat pentru a prelucra informatiile din formulare si a adauga dinamism paginilor web
- JavaScript este un limbaj interpretat
- JavaScript este incorporat de regula in paginile HTML
- JavaScript **nu este** JAVA, denumirea initiala a fost *LiveScript*
- JavaScript are o sintaxa asemanatoare limbajului C/Java
- JavaScript este **dependent de mediu** – software-ul care ruleaza de fapt programul este browser-ul web (Firefox, Opera, Netscape Navigator, Internet Explorer, Safari, etc.)

Ce putem face cu JavaScript?

- **JavaScript ofera designerilor HTML posibilitarea de executa scripturi la nivelul browerului**
- **JavaScript poate reactiona la evenimente -** Un JavaScript poate fi configurat sa execute o anumita operatie atunci cand se intampla ceva (e.g. s-a terminat de incarcat pagina, userul a facut click pe un element HTML)
- **JavaScript poate citi si scrie elemente HTML –** Un JavaScript poate citi si schimba continutul unui element HTML
- **JavaScript poate fi utilizat pentru a valida datele din formulare inainte de a fi trimise catre server**
- **JavaScript poate fi utilizat pentru a detecta tipul browserului** - in functie de acesta putem incarca o pagina sau alta
- **JavaScript poate fi utilizat pentru a manipula cookie-uri** - Un JavaScript poate fi utilizat pentru a seta cookie-uri si obtine valoarea acestora

Inserarea codului JavaScript in pagina

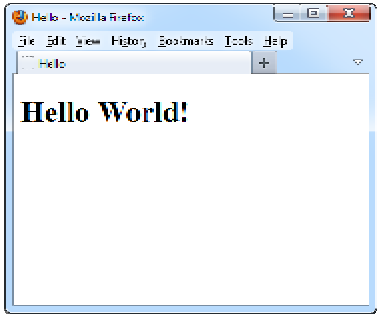
- Inserarea de cod JavaScript intr-o pagina HTML se face prin intermediul tagului <script>:

```
<script type="text/javascript">
  . . .
  instructiuni
  . . .
</script>
```

- unde:
  - Atributul type stabileste limbajul de scripting utilizat (implicit JavaScript)

Exemplu

```
1.  <html>
2.  <head>
3.    <title>Hello</title>
4.  </head>
5.  <body>
6.   <script type="text/javascript">
7.     document.write("<h1>Hello World!</h1>");
8.   </script>
9.  </body>
10. </html>
```



JavaScript nu e intotdeauna disponibil

- Unele browsere mai vechi nu recunosc tag-ul <script>:
  - Acestea vor ignora tag-ul script dar vor *afisa codul* JavaScript inclus
  - Pentru a evita acest lucru folosim urmatoarea sintaxa:

```
<script type="text/javascript">
<!--
   instructiuni
//-->
</script>
```

  - <!-- este secventa de start a unui comentariu HTML
  - Pentru ca motorul de JavaScript sa ignore secventa de sfarsit a comentariului HTML, -->, utilizam secventa // care indica un cometariu JavaScript
- Uneori utilizatori pot dezactiva de a nivelul browserului executia JavaScript-urilor
  - In acest caz putem afisa un mesaj foslosind urmatoarea secventa:

```
<noscript>mesaj afisat dacat JavaScript-ul e dezactivat </noscript>
```


Exemplu

```
1.  <html>
2.  <head>
3.    <title>Hello</title>
4.  </head>
5.  <body>
6.  <script type="text/javascript">
7.   <!--
8.    document.write("<h1>Hello World!</h1>");
9.    //-->
10. </script>
11. <noscript>Limbajul JavaScript nu e disponibil</noscript>
12. </body>
13. </html>
```


Utlizarea codului JavaScript

- Putem utiliza JavaScript atât în <head> cât şi în <body>
  - Funcţiile JavaScript se definesc de regulă în <head>
    - Aceasta ne asigura că acea funcţie este încărcată înainte de fi nevoie de ea
  - Codul JavaScript din <body> se va executa în momentul in care este incarcata pagina
- Scripturile JavaScript pot fi definite intr-un fisier extern, fisier ce are extensia .js
  - Avataj: fisierul poate fi inclus in mai multe pagini HTML
- Includerea unui fisier .js intr-o pagina HTML se face de regula in tag-ul <head> printr-o constructie de forma:

```
<script type="text/javascript" src="/cale/numefiser.js"></script>
```

- JavaScript poate fi utilizat si in obiecte de tip *form*, de exemplu butoane
  - Acest cod JavaScript se va executa cand obiectul este folosit

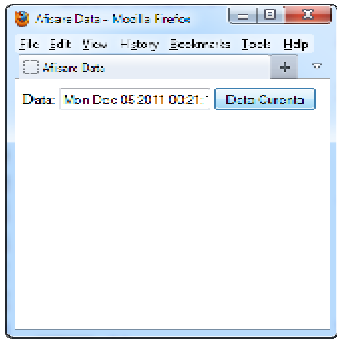
Exemplu

date.js

```
1.  function displayDate(id) {
2.  document.getElementById(id).value=Date();
3.  }
```

date.html

```
<html>
2.<head>
3.  <title>Afisare Data</title>
4.  <script type="text/javascript" src="date.js"></script>
5.</head>
6.<body>
7. <form>
8.  Data: <input type="text" id="data" value="">
9.  <input type="button" onclick="displayDate('data')"
value="Data Curenta">
10. </form>
11.</body>
12.</html>
```



Tipuri primare in JavaScript

- JavaScript are trei tipuri primare: number, string, si boolean
  - Orice altceva este obiect
- **Numerele** sunt memorate intodeauna in virgula flotanta
  - Numerele hexazecimale incep cu 0x
  - Numerele in baza 8 incep cu 0 (nu toate browserele suporta)
- **Stringurile** sunt secvente de caractere cuprinse intre ghilimele sau apostroafe
  - Stringurile pot contine \n (newline), \" (ghilimele), etc.
- Valorile **boolene** sunt true sau false
  - 0, "0", stringurile vide, undefined, null, si NaN sunt considerate false , orice altceva este considerat true

Variabile

- Sintaxa declararii unei variabile este urmatoarea
  ```
  var idVar;
  var idVar = val_initiala;
  idVar = val_initiala;
  ```
- Numele variabilelor sunt case-sensitive si trebuie sa inceapa cu o litera
- Variabilele nu au tip (ele pot retine orice valoare)
- Cuvantul var este optional (daca nu se specifica automat variabila e considerata globala)
- Variabilele declarate intr-o functie sunt locale acelei functii
  - La declararea variabilelor locale trebuie obligatoriu var
- Variabilele declarate in afara oricarei functii sunt globale (sunt accesibile oriunde in pagina)

Operatori (I)

- Aritmetici:
  `+  -  *  /  %  ++  --`
- Operatori de comparatie:
  `<  <=  ==  !=  >=  >`
- Operatori logici:
  `&&  ||  !`
- Operatori pe biti:
  `&  |  ^  ~  <<  >>  >>>`
- Operatori de atribuire:
  `=  +=  -=  *=  /=  %=  <<=  >>=  >>>=  &=  ^=  |=`

Operatori (II)

- Concatenarea stringurilor:
  `+`
- Operatorul conditional :
  conditie ? *val_if_true* : *val_if_false*
- Testarea egalitatii:
  - == si != incearca sa converteasca ambii operanzi la acelasi tip inainte de a efectua testul
  - === si !== considera valorile *inegale* daca au tipuri diferite
- Alti operatori
  `new` `typeof` `delete`

Comentarii

- Comentariile in JavaScript sunt similare celor din C++ sau Java:

```
// comentariu pe o singura linie
/* comentarii pe
   mai multe linii */
```


Instructiuni

- Atribuire
  `idVar = expresie;`
- Instructiunea compusa

```
{
 instructiune1;
 instructiune2;
    . . .
 instructiune n;
}
```


Instructiuni de decizie

- Selectie simpla:

```
if (conditie)
      instructiune;
if (conditie)
      instructiune1;
else
      instructiune2;
```

- Selectie multipla:

```
switch(n) {
case constanta1:
  bloc instructiuni 1
  break;
 ...
case constantaN:
  bloc instructiuni N
  break;
default:
  bloc instructiuni n+1
}
```

Instructiuni de ciclare (I)

- Instructiunea for

```
for (expr_i;conditie; expresie_reitializare)
    instructiune
```

- Exemplu

```
<script type="text/javascript">
var i=0, s=0;
for (i=0;i<=5;i++) {
    s+=i;
}
document.write(" Suma este " + s);
</script>
```

- Instructiunea for ... in

```
for (variabla in object)  {
   instructiuni
}
```

- Exemplu

```
<script type="text/javascript">
var persoana={nume:"Gabroveanu",
    prenume:"Mihai", varsta:25};

for (x in persoana){
document.write(persoana[x] + " ");
}
</script>
```


Instructiuni de ciclare (II)

- Instructiunea while

```
while (conditie)
    instructiune
```

- Exemplu

```
<script type="text/javascript">
var i=0, s=0;
while (i<=5) {
    s+=i;
     i++
}
document.write(" Suma este " + s);
</script>
```

- Instructiunea do ... while

```
do  {
   instructiuni
} while (conditie);
```

- Exemplu

```
<script type="text/javascript">
var i=0, s=0;
do {
    s+=i;
     i++
} while (i<=5);
document.write(" Suma este " + s);
</script>
```


Exceptii

- JavaScript dispune de un mecanism de tratare a exceptiilor *aproape* identic cu cel din Java
- Aruncarea unei exceptii

```
throw expresie
```

  - unde *expresie* este valoarea exceptiei, si poate fi de **orice tip** (cel mai adesea un string)
- Prinderea unei exceptii

```
try {
   instructiuni
} catch (e) {    // Observatie: nu se specifica/declara tipul lui e
   instructiuni de tratare a exceptiei e
} finally {       // otional
   cod ce se executa la final
}
```


Functii

- Se definesc de regula in <head>
- Sintaxa definirii unei functii este urmatoarea:

```
function numeFunctie(p1, …, pN) {
   //delcaratii de variabile locale (se utilizeaza var)
   instructiuni
}
```

- O functie poate returna o valoare cu return *valoare*;
- Sintaxa apelului unei functii

```
numeFunctie(vp1, …, vpN)
```

- Parametri simpli sunt transmisi prin valoare, obiectele prin referinta

Functii - Exemplu

```
1.  <html>
2.  <head>
3.   <title>Instructiuni</title>
4.   <script type="text/javascript">
5.   function factorial(n){
6.    var i, p=1;
7.    for (i=1;i<=n;i++) {
8.      p*=i;
9.    }
10.  return p;
11.  }
12. </script>
13. </head>
14. <body>
15. <h1>Functii</h1>
16. <script type="text/javascript">
17. var n=3;
18. document.write( n + "!=" +
    factorial(n));
19. </script>
20. </body>
21. </html>
```


Tablouri

```
var masini= new Array();
masini[0]="Audi";
masini[1]="Logan";
var masini= new Array("Audi", "Logan"); //tablou condensat
var masini= ["Audi", "Logan"]; //tablou de literali
var masini= ["Audi", "Logan", "Ford"]; //tablou de literali
masini.sort();
var i=0;
for(i=0;i<masini.length;i++){
 document.write(masini[i]);
}
masini.reverse();
document.write(masini.toString());
```


Crearea Obiectelor

- Utilizand obiecte literal:

```
var curs = {"I3502", "Tehnologii Web"}
```

- Cream un obiect ”gol” cu ajutorul operatorului new, apoi adaugam proprietati:

```
var curs = new Object();
curs.id = " I3502 ";
curs.nume = " Tehnologii Web";
```

- Cu aujutorul costructorilor:

```
function Curs(id, nume){
  this.id = id;            //cuvantul cheie this e obligatoriu
  this.nume = nume;
}
var curs =new Curs("I3502", "Tehnologii Web");
```


Adaugarea de metode la obiecte

```
1.  function Triunghi(a, b, c){
2.     this.a = a;
3.     this.b = b;
4.     this.c = c;
5.     this.getPerimetru = function (){
6.          return this.a + this.b + this.c;
7.     }
8.  }
9.  var t = new Triunghi(3,4,5);
10. document.write(t.getPerimetru());
```

Evenimente

- **Evenimentele** sunt actiuni care pot fi detectate de JavaScript. Putem configura executia de actiuni la detectarea de evenimente
- Exemple de evenimente:
  - S-a efectuat click pe un buton
  - S-a terminat de incacat pagina