Tehnici Web
CURSUL 5

Semestrul II, 2021-2022

Carmen Chirita

https://sites.google.com/site/fmitehniciweb/

JavaScript-istoric

- Inventat de Brendan Eich în 1995, la Netscape (denumit initial Mocha și LiveScript);
- Implementat de browserul Netscape Navigator sub numele de JavaScript;
- Adaptat de Microsoft în 1996 și denumit Jscript;
- Standardizat în 1997 de asociatia ECMA (European Computer Manufacturer's Association) sub numele de ECMAScript;
- Ultima versiune suportată de browsere: ECMAScript6

JavaScript-caracteristici

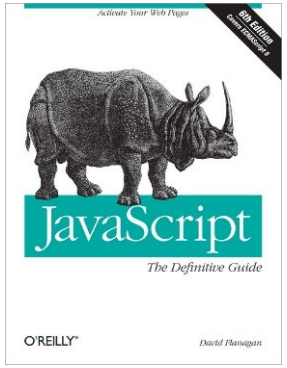
- Este un limbaj de scripting pentru pagini web

  (pe partea de client)
- Este un limbaj care combina mai multe paradigme: functionala, orientata pe obiecte și bazata pe prototipuri, bazata pe evenimente
- Este un limbaj interpretat (scriptul este executat direct, fără compilare prealabila)
- Este „loosely typed”

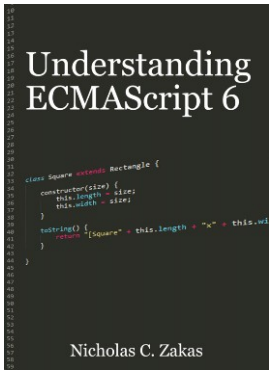
JavaScript

https://developer.mozilla.org/en-US/docs/Web/JavaScript

https://drive.google.com/drive/u/0/folders/1uJsUzLkT4CCAxathxLruI7k9dE7U8Yc9



JavaScript The Definitive Guide, David Flanagan



Understanding ECMAScript 6, Nicholas C. Zakas

JavaScript și HTML

- Orice pagina Web este reprezentata în memorie ca un arbore de elemente (obiecte)

  (Document Object Model – DOM)

- JavaScript poate interactiona cu documentul HTML prin intermediul DOM-ului.

element HTML → Obiect în JavaScript

Atribut al unui element HTML → Proprietate a obiectului în JavaScript

HTML	JavaScript
element HTML →	obiect JavaScript
<h1 id=”titlu”>...</h1>	ob=document.getElementById(”titlu”)
atribut al unui element HTML →	proprietate a obiectului JavaScript
<h1 id=”titlu”class=”special”>...	ob.id, ob.className
<img src=”poza.jpg”>	ob.src
atributul style – proprietăți CSS →	proprietatea style -> obiectul style – proprietati de stilizare CSS (ex. backgroundColor)
<h1 style="color:blue;text-align:center;">	ob.style.color, ob.style.textAlign

Codul JavaScript poate fi plasat:

Oriunde in documentul HTML, folosind tagul script (varianta inline)

```
<head>
<script type="text/javascript">
/* cod JavaScript */
</script>
<head>
```

Într-un fisier extern (NumeFisier.js) care este importat in documentul HTML (varianta recomandata)

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

Atribute suplimentare care funcționează doar pentru scripturi externe

defer : întârzie executarea codului până documentul HTML se încarcă complet;

async: permite încărcarea codului asincron față de document

Orice tab al unui browser contine un obiect **window** (din clasa Window)

Metodele prompt si alert

prompt(text, default-text): afiseaza o caseta de dialog care cere utilizatorului sa introduca informatii

alert(mesaj): afiseaza o caseta de alertare care contine un mesaj și un buton OK

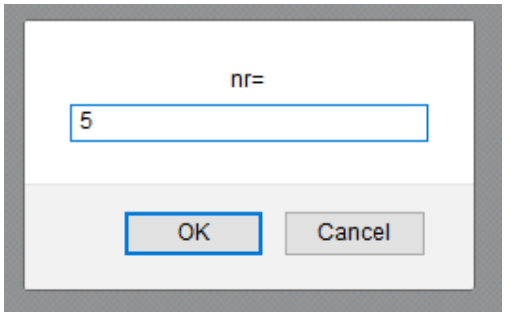
```
var x = prompt("nr1");
var y = prompt("nr2");
alert(typeof(x));
alert(x+y);
alert(parseInt(x)+parseInt(y));
```

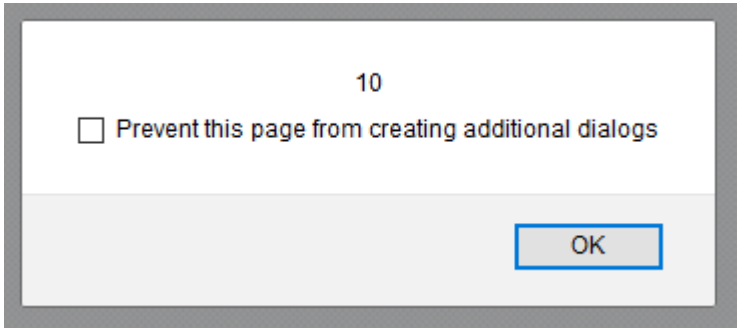
```
<script
type="text/javascript" >

var n=prompt("nr= ");
alert(suma(parseInt(n)));

function suma(x) {
var i;
var s = 0;
for (i=1; i< x ; i++) s=s+i;
return s;};

</script>
```





JavaScript este CASE SENSITIVE și folosește setul de caractere Unicode

Identificatorii:

- denumesc variabile, cuvinte cheie, funcții, etichete;
- formati din: cifre, litere, _, $;
  primul caracter: litera, _, $

; separator (ex. a = 3; b = 4;)

{ } bloc de instructiuni

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

Tipuri de date în JavaScript:

primitive: number, string, boolean, null, undefined, symbol (ES2015)

obiecte (tipul object)

obiecte predefinite:

Array, String, Number, Boolean, Math, Date, Set, Function, Error, RegExp,etc.

Instrucțiuni:

=, if/else, for, switch, while, return, { inst1; inst2;}

Funcții:

function nume(param1,param2,..) {corpul functiei}

Tipuri de date în JavaScript

In Javascript **nu** este necesara precizarea tipului de date, ca în alte limbaje de programare (ex. C/C++, Java)

Browserul realizeaza singur identificarea tipului de date

```
x=23;   //number
y=”abc”;  //string
z= true;  //boolean
var a;  //undefined
persoana={nume:”Ana”, vârsta:20};  //obiect
```

Variabile în JavaScript

Declarare

- variabilele declarate în interiorul functiilor sunt locale pentru acele funcții

- variabilele declarate în afara oricărei funcții se numesc variabile globale

- variabilele declarate într-un domeniu (scope) sunt accesibile în funcțiile copil

Initializare

- se poate atribui o valoare inițială la declarare

- pentru atribuire fără declarare se caută numele variabilei în scope-ul curent, continuând în arborele de părinți

- dacă nu se găsește declarația atunci se va creea o variabilă globală cu acea valoare

Variabile în JavaScript

- Variabilele pot fi declarate explicit folosind cuvântul cheie var; optional variabila poate fi initializata cu o valoare;

  - are scope-ul global sau la nivel de funcție; poate fi accesata din afara blocului de instructiuni

```
var x=0; //var globala
function f(){
      var y=1; //var locala
      var x=5; //var locala
      function g(){
            y++;
            z=1; //var globala
            console.log(y);
            }
      g();
}

f();  //va afisa 2
console.log(x); //va afisa 0
console.log(y); //va genera eroare
console.log(z); //va afisa 1
```

Variabile în JavaScript

- Variabilele se mai pot declara folosind cuvântul cheie let; acesta declara o variabila locala vizibila **doar** în blocul, instrucțiunea sau expresia în care este folosită.

ex.1

```
var g = 0;
    {
     let g = 1;
    }
console.log (g);  // 0
```

ex.2

```
function letTest() {
  let x = 1;
  if (true) {
    let x = 2;
console.log(x); // 2
  }
  console.log(x);// 1
}
```

In ECMAScript 6 au fost introduse declaratii de constante

```
const Max =100;
```
- se initializeaza la declarare
- nu se pot modifica pe parcursul executiei programului

Variabilele au tipuri dinamice
- pot fi declarate (și pot primi valori) cu orice tip de date.

Tipul variabilei nu este specificat explicit, dar poate fi aflat cu `typeof(x)`

Tipul unei variabile nedeclarate este undefined.

```
typeof(null)    // “object”
```

```
typeof(undefined)    // “undefined”
```

```
> x=2
  2
> typeof(x)
  "number"
> x="Hello"
  "Hello"
> typeof(x)
  "string"
> typeof(y)
  "undefined"
```

Scopul variabilelor: zona din program în care sunt declarate variabilele;
Trei tipuri de scopuri: scop global, scop functie (local), scop bloc

```
// scopul  global
 function fA (){
     //scopul A
          function fB () {
                  // scopul B
                              }
                 }
```

Toate variabilele/obiectele/functiile declarate de o functie parinte sunt vizibile in descendentii ei

```
function calculeaza(an){
        var varsta=2021- an;

       function afiseaza(){
         const str= nume + " are " + varsta + "
                    ani, nascuta in "  + an;
          if (varsta>=18) {
           const majora=true;
          }
         console.log(str);
         //console.log(majora); va genera eroare
        }
       afiseaza();

}
const nume="Maria";
calculeaza(1990);
```

Hoisting in JavaScript

Domeniul de vizibilitate al unei variabile coincide cu functia in interiorul careia a fost definita;

Inainte de a fi executat, codul JavaScript este parsat si "rearanjat" a.i. toate declaratiile de variabile (nu si operatiile de atribuire) sunt mutate (ridicate) la inceputul zonei de vizibilitate (adica la inceputul functiei).

```
var x = 5; //globala
function host() {
    if (x == 5) {
        var x = 10;
       }
alert(x);
}
host(); // va afisa undefined
alert(x); // va afisa 5
```

echivalent cu

```
var x = 5; //globala
function host() {
   var x; // variabila locala
    if (x == 5) { // x este undefined
        x = 10;
     }
alert(x);
}
host(); // va afisa undefined
alert(x); // va afisa 5
```

Tipul number (reprezentare binara pe 64 biti)

```
var a = 4;
var r = 34.7;
```

Operatorii aritmetici specifici:
+ - * / % ++ --

Conversia de tip automata

```
x = “2” * 7; // 14
y = “2” + 7; // "27"
z = parseInt(“2”) + 7; // 9
t = “2” * “7”; // 14
u = 2 + 3 + “4”; // "54"
z = “2” + 3 + 4; // "234"
```

Obiectul Math
Math.PI //=> 3.14
Math.pow(2,3) //=> 8
Math.round(4.7) //=>5
Math.random() // intre 0 si 1
Math.sqrt(-1) // => NaN

conține proprietăți si metode

```
> x=5
  5
> y=3.4
  3.4
> typeof(x)
  "number"
> typeof(y)
  "number"
> parseInt("5hello")
  5
> parseFloat("3.4Hello")
  3.4
> parseInt("3.4Hello")
  3
```

Tipul string (sir de caractere scris intre ‘ ‘ , ` ` sau " ")

```
var s = "Ana Popescu";
var t = 'Ana Popescu';
var  pnume = s.slice (0, s.indexOf(" "));  //’Ana’
var  fnume=s.slice(s.lastIndexOf(" ")+1,s.length);  //’Popescu’
```

Proprietăți și metode: `length, charAt(), indexOf(), lastIndexOf(), replace(), split(), toLowerCase(), toUpperCase(), concat(),..`

Concatenarea: "numarul" + "1", "id"+1

Caractere speciale: \’ \” \n \t \v \b \\

Accesarea unui caracter: `s[pozitie], s.charAt(pozitie)`

```
var x =  “abcde”;
alert(x[0]);
x[0] = 'v';
alert(x[0]); // => a
```

Un string nu este un array de caractere

Tipul boolean : true si false

Orice valoare poate fi convertita explicit folosind obiectul predefinit Boolean

```
var nume =  Boolean(valoare);

true ===  Boolean("adevarat") // => true
false === Boolean("") // => true
```

Operatori logici pentru tipuri primitive: > < <= >= && || ! == === != !==

verifica si tipul operanzilor

Putem utiliza și alte tipuri de date în context logic:

- pentru fals: `0, "" , NaN, null, undefined`
- pentru adevarat: orice alta valoare

Operatorul conditional

conditie ? expr1 : expr2

```
function fact(n){
return (n <= 2) ? n: n*fact(n-1);
}
```

Tipurile undefined si null

variabilele care nu au primit încă o valoare au tipul undefined

```
var x
x == undefined // true
typeof(x) // undefined
```

null lipsa unei valori (intentionata)

```
var x=null
typeof(x) //object
null == undefined // true
null === undefined // false
```

Tipul object

Un obiect este o colectie de perechi nume-valoare. Daca valoarea este o functie atunci proprietatea se numeste metoda.

```
var ob = {prop1: val1, prop2: val2, … ,prop-n: val-n};

Accesarea proprietatilor:

ob.prop1; // val1

ob["prop1"]; // val1
```

Exemple:

```
var complex = { re: 5, im: 3}; \\ complex.re, complex.im
var student = {nume: "Ionescu",
               nota1:9,
               nota2:10,
               media:function(){
                       return (this.nota1 + this.nota2)/2;
                               }
               }
```

```
student.nume    \\ Ionescu
student.nota1   \\ 9
student.nota2   \\ 10
student.media() \\ 9.5
student.media \\ functia
```

this este obiectul cu care se va apela functia

Tipul object

Toate datele din tipurile primitive in afara de tipul object sunt transmise prin valoare.

Datele de tip object sunt transmise prin referinta.

```
var a = {nume: “Ana”} // object
var b = a;  // a și b refera aceeași zona
b.nume = b.nume + “ Popescu”; // se modifica și b și a
alert(a.nume ); //  “Ana Popescu”
```

```
var s  = “Ana”; // string
var t = s; // t copiaza valoarea lui s
t= t + “ Popescu” // se modifica doar t
alert(s) // => “Ana”
```

Crearea obiectelor

- Prin obiect literal:

  proprietatile, metodele, împreuna cu valorile lor sunt enumerate intre acolade; se creaza un singur obiect.

```
var pers= {nume:"Popescu",prenume:"Andrei”,vârsta:20}
```

Crearea obiectelor

- Cu ajutorul obiectului generic

  se apeleaza constructorul new Object() și se adauga apoi proprietatile și metodele; se creeaza un singur obiect

```
var pers= new Object();
pers.nume="Popescu";
pers.prenume="Andrei”;
pers.varsta=20;
```

Crearea obiectelor

- Cu ajutorul unui constructor de obiecte

Se defineste o funcție constructor(parametrii) care apoi va fi apelata cu new constructor(parametrii) pentru fiecare obiect care va fi creat

```
function pers(n,p,v) { this.nume=n;
                       this.prenume=p;
                       this.varsta=v;
                     }

var p1=new pers(”Popescu”,”Andrei”,20);
var p2=new pers(”Ionescu”,”Bogdan”,20);
```

Proprietăți și metode globale

Pot fi folosite împreuna cu orice variabila și obiect creat în JavaScript

Proprietate	Descriere
`Infinity`	O valoare numerică care reprezintă infinitiv pozitiv/negativ
`NaN`	O valoare "Not-a-Number"
`undefined`	Indică o variabilă căreia nu i-a fost atribuită o valoare

Metode

isNaN() // Determină dacă valoarea este un număr invalid
parseInt() //converteste un sir într-un intreg
parseFloat() //converteste un sir într-un numar zecimal
Number() //converteste un obiect într-un numar
String() //converteste un obiect într-un sir

Obiecte predefinite în JavaScript

Obiecte corespunzatoare tipurilor primitive

Boolean, String, Number

se pot crea obiecte noi cu new

Object
Array, Set, Map
Function
RegExp
Date
se pot crea obiecte noi cu new

```
var y = new Number(123);
typeof(y) // object

var ob = new Object();
ob.x =1; ob.y=2;    // ob ={x:1, y:2}

var d= new Date(2015,3,1);
alert(d.getUTCDay()); // 2 (ziua din
 săptămâna (0-6))
```

Array

```
var v = new Array();
v[0] = “a”; v[1]= “b”;

var v = new Array(“a”,”b”);

var v = [6,4,7,3];
```

creare

Proprietati si metode

```
v=[6,4,7,3];
v.length // 4
v.push(10);    // =>v=[6,4,7,3,10]
v.pop();       // =>v= [6,4,7,3]
v.shift();  // => v=[ 4,7,3]
v.unshift(10);  // =>v= [10,4,7,3]
v.sort();     // => v= [3,4,6,7]
```

tipul elementelor nu e fixat

```
v=["hi",2,[5,7]]
```

```
var s = “azi este joi”;

var a = s.split(“ “);
    // a = [“azi”,”este”,”joi”]

a.reverse();
    // a = [“joi”,”este”,” azi”]

var s= a.join(‘/’);
        // s=“joi/este/azi”
```

Exemplu

Generarea random a unei culori dintr-un vector de culori și colorarea body-ului în culoarea respectiva

```
function getRandomInt(max) {    \\generarea random a unui întreg intre 0 și max-1
  return Math.floor(Math.random() * max);
}
var culori=["blue","green","yellow","black","red","orange","white","pink"];
document.body.style.backgroundColor=culori[getRandomInt(culori.length)];
```

random(), floor() : metode ale obiectului **Math**

length: proprietate a obiectului **Array**

Exemplu

Generarea random a unei culori dintr-un vector de culori și colorarea body-ului în culoarea respectiva

```
function getRandomInt(max) {    \\generarea random a unui întreg intre 0 și max-1
  return Math.floor(Math.random() * max);
}
var culori=["blue","green","yellow","black","red","orange","white","pink"];
document.body.style.backgroundColor=culori[getRandomInt(culori.length)];
```

random(), floor() : metode ale obiectului **Math**

length: proprietate a obiectului **Array**

Exemplu

Generarea random a unei culori dintr-un vector de culori și colorarea body-ului în culoarea respectiva

```
function getRandomInt(max) {    \\generarea random a unui întreg intre 0 și max-1
  return Math.floor(Math.random() * max);
}
var culori=["blue","green","yellow","black","red","orange","white","pink"];
document.body.style.backgroundColor=culori[getRandomInt(culori.length)];
```

random(), floor() : metode ale obiectului **Math**

length: proprietate a obiectului **Array**

Set (introdus in ECMAScript6)

Metode: add(val), has(val), size(),values(), keys(),
delete(val), clear()

```
var m = new Set();
m.add(1).add(2).add(3);
if (m.has(3)) {let s=0;
                for (e of m.values()) s=s+e;
                alert(s);}
```

Instructiuni: for (exista 2 variante noi ale instructiunii for)

```
for (initializare; conditie; update) {
 instructiuni;
}
```

```
for (var i = 0; i < 9; i++) {
   console.log(i);}

 //afiseaza la consola nr. de la  0-8
```

```
for (variabila in obiect) {
instructiuni
}
```

```
var pers = {nume:"Popescu", prenume:"Andrei", varsta:30};
   var text = " ";
   var x;
   for (x in pers) {//iteram printre proprietatile obiectului pers
       text += pers[x] + " "; //retinem valoarea proprietatii
   }
     //text va conține „Popescu Andrei 30”
```

```
for (item  of iteratii) {
instructiuni
}
```

```
var tablou = [10, 20, 30];
for (var x of tablou) {//iteram printre
elementele tabloului
  x += 1;
  console.log(x);
}//se va afisa la consola 11, 21, 31
```

Instrucțiuni: while, do, if, switch (întâlnite și în alte limbaje de programare)

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

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

```
if (conditie) instructiune;
```

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

```
switch (expresie)
{
 case 1:
        bloc 1
        break;
case 2:
        bloc 2
        break;
.....
default : bloc
}
```

Funcții

Sintaxa:

```
function nume(arg1, arg2,.., argn) {
    instructiuni;
    return valoare; // nu neaparat
}
```

În Java Script o funcție poate fi apelata cu un numar variabil de parametrii

```
function suma(a,b) {
 return a+b;
}

suma(2,3); // 5
suma(); // NaN
suma(2); // NaN
suma(3,4,1,5,6,7) // 7
```

```
var x = 0;
var y = {n:0};
function f(a,b){
                a = 1;
                b.n = 1;}
f(x,y);
alert(x); //=> 0
alert(y.n);   //=> 1
```

Parametrii de tip primitiv se transmit prin valoare;
Parametrii de tip obiect se transmit prin referinta.

Funcții

În Java Script orice funcție poate accesa un obiect notat „arguments” (asemănător unui array) care conține valorile argumentelor cu care se apeleaza functia;
arguments.length va calcula numărul argumentelor

```
function fun() {
 return arguments.length;
}

fun(2,”sss”, 5); // 3
```

```
function func1(a, b, c) {
  console.log(arguments[0]);
  // expected output: 1

  console.log(arguments[1]);
  // expected output: 2

  console.log(arguments[2]);
  // expected output: 3
}

func1(1, 2, 3);
```

Exemplu (developer.mozilla.org)

Exemplu folosind obiectul arguments:

Rescrierea functiei suma care calculează suma argumentelor functiei indiferent de numarul de argumente cu care se va apela functia

```
function suma()
 {
      var s=0;
      for(var i=0; i<arguments.length; i++)
                  s+=arguments[i];
       return s;
}

suma(); // 0
suma(5); // 5
suma(2,3,4,5); //14
```

Functii anonime

```
function (arg1,…,argn){
    instructiuni;
}

var fun = function () {
 return arguments.length;
}
```

```
function f(x){return x+1};
var f1 = f;
var x=f(3);
var x1=f1(3);


var g = function (x){return x+1};
var y = g(3);
typeof(g) // "function"

var h = (x)=>{x+1}
```

arrow functions
ECMAScript 6

Array: metoda map()

-creaza un nou array prin apelarea unei funcții (data ca parametru) pentru fiecare element din array
-nu modifica array-ul initial

Sintaxa

```
array.map(function(currentValue, index, arr), thisValue)
```

optionale

Exemplu

```
function f(x) {
  return x+1;
}

var array = [1,2,3,4];
var array1=array.map(f);
console.log(array1);      //array1=[2,3,4,5]
```

```
function g(x) {
  return x * this.a;
}
var o1={a:2}, o2={a:3};
var array = [1,2,3,4];
var array1=array.map(g,o1); //[2,4,6,8]
var array2=array.map(g,o2); //[3,6,9,12]
```

Array: metoda forEach()

- apelează o funcție pentru fiecare element dintr-un array

Sintaxa

```
array.forEach(function(currentValue, index, arr), thisValue)
```

optionale

Exemplu

```
function f(x,i) {
  alert(i+" : "+x);
}
var array = ['luni', 'marți' , 'miercuri', 'joi'];
array.forEach(f);

var d=[1,2,3,4];
d.forEach(function(v,i,a){a[i]=v+1;});
console.log(d); //[2,3,4,5];
```

Array: metoda filter()

-creaza un nou array cu elementele care verifica conditia implementata de functia data ca parametru
-nu modifica array-ul initial

Sintaxa

```
array.filter(function(currentValue, index, arr), thisValue)
```

optionale

Exemplu

```
function check(cuvant) {
  return cuvant.length > 4;
}

var cuvinte = ["pisica", "cal", "caine", "oaie"];
var rez=cuvinte.filter(check);
console.log(rez);      //rez=["pisica", "caine"]
```

Array: metoda reduce()

- executa o funcție de reducere (data ca parametru) pe fiecare element al array-ului rezultand o singura valoare de ieșire
- nu modifica array-ul initial

Sintaxa

```
array.reduce(function(total, currentValue, currentIndex, arr), initialValue)
```

optionale

Exemplu

```
function suma(total,val) {
  return total+val;
}

var array = [1,2,3,4];
var rez=array.reduce(suma);
console.log(rez);      //rez=10;
```

Exemplu: ex3-array.html

```
var clasa =[{nume:"Ion",nota:5},{nume:"Petre",nota:6},
            {nume:"Paul",nota:3},{nume:"Ana",nota:8},
            {nume:"Maria",nota:6},{nume:"Simona",nota:4}];

var copie=clasa.slice();

var rez = copie.filter((x) => (x.nota >= 5))
                .map((x) => (x.nume+" are  nota "+x.nota))
                .forEach((x)=>{alert(x)});
```