# Introducere în limbajul JavaScript

1. **HTML** - folosit pentru a defini conținutul paginilor web

2. **CSS** - folosit pentru a specifica aspectul paginilor web

3. **JavaScript** - folosit pentru a programa comportamentul paginilor web

- utilizarea scripturilor în JavaScript în vederea realizării de pagini dinamice
- utilizarea PHP pentru realizarea paginilor Web dinamice
- utilizarea bazelor de date MySQL pentru realizarea de pagini Web complexe

# *Diferența dintre Front-End(client) şi Back-End (server)*

## Front-End:

- Implică tot ceea ce este vizibil pentru utilizatorul final în browser.
- De obicei dezvoltatorii de Front-End au cunoștințe de HTML, CSS, **JavaScript**, jQuery, Photoshop.
- În această arie găsim:

– Web Designer (stilizează aplicația), User Experience (teste de utilizare), Front-End Developer (implementare de pagini statice)

## Back-End:

- Reprezintă tot ceea ce utilizatorul final nu vede în aplicație: baza de date, server.
- De obicei dezvoltatorii de Back-End au cunoștințe de HTML, CSS, JavaScript, dar nu prioritar. Pentru ei, limbajele de programare: **PHP**, .Net, Python, Java, precum și cunoștințele legate de **baze de date** sunt mai des folosite
- În această arie găsim:

– Back-End Developer: preocupat in principal de securitatea aplicației, structura, gestionarea datelor.

**Aplicații statice/dinamice:**

- Un site static nu necesită Back-End, nu are interacțiune cu baze de date, nu își schimbă conținutul.

- Un site dinamic este mult mai complex decât unul static.

# Ce este JavaScript?

- Limbaj de programare(1995), folosit pentru:

– Calcul, manipulare și validare date.

– Modificare conținut HTML și CSS.

- **HTML** - folosit pentru a defini conținutul paginilor web
- **CSS** - folosit pentru a specifica aspectul paginilor web
- **JavaScript** - folosit pentru a programa comportamentul paginilor web- - codul JavaScript din aceste pagini fiind rulat de către browser

# Caracteristici-avantaje-utilizare

- **JavaScript poate fi introdus in HTML**
- **JavaScript este dependent de mediu** - JavaScript este un limbaj de scriptare; software-ul care rulează de fapt programul este browser-ul web
- **JavaScript este un limbaj in totalitate interpretat-** codul scriptului va fi interpretat de browser înainte de a fi executat
- **JavaScript este un limbaj flexibil**
- **JavaScript este bazat pe obiecte -** modelul de obiect JavaScript este bazat pe instanță și nu pe moștenire
- **JavaScript este condus de evenimente -** mare parte a codului JavaScript răspunde la evenimente generate de utilizator sau de sistem
- **JavaScript nu este Java**
- **JavaScript este multifuncțional** - rezolva diferite probleme: grafice, matematice, etc

# Aplicații ale limbajului JavaScript

- Oferă dinamică paginilor web statice
- Dezvoltarea de aplicații pentru
  - Telefoane
  - Desktop
- Dezvoltarea de jocuri

**Limbajul JavaScript poate servi la:**

- generarea paginilor Web personalizate şi modificarea dinamică a prezentării lor;
- realizarea calculelor matematice;
- validarea conţinutului unui formular;
- 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

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

➢ sintaxa JavaScript

Sintaxa defineşte un ansamblu de reguli care trebuie respectate când se scrie cod JavaScript.

➢ DOM-ul (Document Object Model – modelul obiectelor documentului).

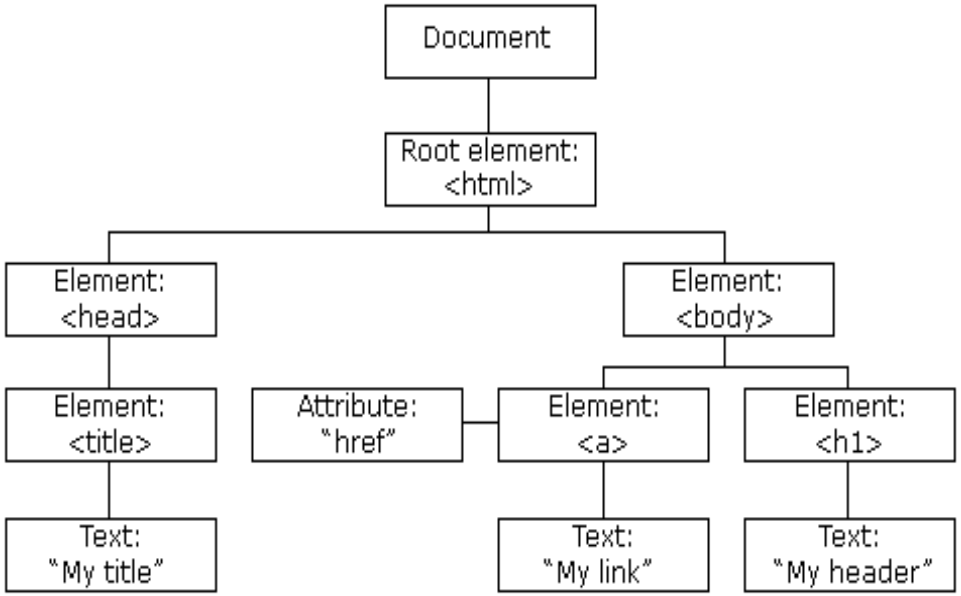
DOM-ul se referă la componentul paginii Web, obiectele care se pot accesa şi manipula cu ajutorul limbajului JavaScript.

https://www.w3schools.com/Js/js_htmldom.asp

HTML DOM este un model de obiect standard și o interfață de programare pentru HTML. Acesta definește:

- Elementele HTML ca obiecte
- Proprietățile tuturor elementelor HTML
- Metodele de accesare a tuturor elementelor HTML
- Evenimentele pentru toate elementele HTML

Când o pagină web este încărcată, browserul creează un DOM al paginii.




- JavaScript poate schimba toate elementeleHTML din pagină
- JavaScript poate schimba toate atributele HTML în pagină
- JavaScript poate schimba toate stilurile CSS în pagină
- JavaScript poate elimina elemente HTML și atribute existente
- JavaScript poate adăuga noi elemente HTML și atribute
- JavaScript poate reacționa la toate evenimentele HTML existente in pagină
- JavaScript poate crea noi evenimente HTML în pagină

JavaScript poate să apară:

1. Într-un document HTML:

<head> </head> sau <body></body>

doar dacă este însoţit de eticheta

**<script>** *…cod…* **</script>**

Obs.

- atributul // **language="JavaScript“** sau
- atributul // **type="text/javascript”**

2. Fișier extern cu extensia **".js"** - *nu putem folosi etichete HTML, ci numai instrucțiuni JavaScript*.

<script src=“nume_fișier.js”> </script>

- Atributul src specifică locaţia unde se află codul JavaScript.

```
<html>
<head>
</head>
<body>
   <script>
    ...
   </script>
</body>
</html>
```

script în body

```
<html>
<head>
   <script>
    ...
   </script>
</head>
<body>
</body>
</html>
```

script în head

# Ex.

```
<html><head>
<meta charset="utf-8" />
<title>Pagina HTML cu JavaScript</title>
<style type="text/css">
.rosu{color:#FF0000;}
</style> </head>
<body>
<br><p class="rosu" >
Acesta este un document HTML conținând JavaScript. </p>
<script>
document.write("Acesta este JavaScript!")
</script>
<br><p class="rosu" >
Din nou text și cod HTML.
</body>
</html>
```

**Obs.**

- **document** este un obiect, iar **write()** este o metodă.
- Un obiect poate conţine alte obiecte care pot fi considerate proprietăţi ale acestuia.

De exemplu, **document** conţine alte obiecte, cum ar fi **title**. Acest obiect se va identifica cu **document.title** şi reprezintă titlul paginii.

Ex. obiectul **submit** care este într-un formular din pagină va fi adresat **document.form.submit**.

- Iată câteva exemple de obiecte HTML şi corespondenţele în JavaScript:

| **Obiectul** | **Tag-ul HTML** | **Corespondent JavaScript** |
|---|---|---|
| Pagina Web | <body> .... </body> | document |
| Formular HTML | <form name="formular"> ... </form> | document.formular |
| Buton | <INPUT TYPE="button" name="buton"> | document.formular.buton |
| Imagine | <IMG name="imagine"> | document.imagine |

- Javascript nu are încorporat funcții de tipărire sau afișare

**JavaScript** poate afişa date:

1. scriind într-o casetă de dialog de tip alert folosind metoda **window.alert()**

https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_alert

2. scriind într-un fișier HTML folosind metoda **document.write()**

https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write

3. scriind într-un element HTML folosind proprietatea **innerHTML**

https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_dom

4. scriind în consola browser: **console.log()**

https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_console

```
<!DOCTYPE html>
<html> <body>
<h1>Prima pagina Web </h1>
<p> <FONT COLOR="#FF0000"> Primul paragraf. </FONT> </p>
<script>
document.write("adunam 9+6 = ")
document.write(9 + 6);
</script>
</body> </html>
```

Pentru a avea acces la un element HTML, JavaScript poate utiliza metoda document.getElementById (id) - atributul **id** definește elementul HTML, iar proprietatea **innerHTML** definește conținutul HTML.

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8" /> </head>
<body>
<h1>Prima pagină Web </h1>
<p>
<FONT COLOR="#FF0000">
Primul paragraf. </FONT>
</p>
<p id="demo"></p>
<script>
document.getElementById("demo"). innerHTML = 9 + 6;
</script>
</body>
</html>
```

# Comentarii în JavaScript

Comentariile pot fi adăugate pentru a explica codul sau a-l face mai uşor de citit.

- Comentariile tip linie

//

- Comentarii multi-linie

/*

...

*/

**Obs.** Dacă browserul nu recunoaşte JavaScript, liniile de cod vor fi afişate ca atare în pagină. Pentru a evita acest lucru, scriptul trebuie „ascuns” în taguri de comentariu.

```
<html> <body>
<script>
<!--
document.write("Bine ati venit!");
// -->
</script>
</body> </html>
```

# Elementele fundamentale ale limbajului JavaScript

**Convenții de sintaxa**

**1. Case-sensitive** –

**2. Punct și virgula (;)** - Toate declarațiile trebuie să se termine cu un caracter "punct și virgulă" (;)

**3. Spațiile libere** – ignorate

**4. Ghilimelele** - Ghilimelele simple (') și duble ("") sunt folosite pentru a delimita șirurile de caractere (string).

**5. Caractere speciale**

**\b** - backspace
**\f** - indică o pagină nouă
**\n** - indică o linie nouă
**\r** - indică un retur de car
**\t** - indică o apăsare a tastei TAB
**** - indică un caracter backslash
**\'** - indică un apostrof (ghilimele simple)
**\"** - indică ghilimele duble

**Ex**: **Limbajul "JavaScript"** se va scrie **"Limbajul \"JavaScript\" “**

**6. Numele variabilelor şi funcțiilor -**

# Elementele fundamentale ale limbajului JavaScript

- JavaScript utilizează setul de caractere **Unicode** UTF-8 - Unicode Transformation Format
- identificatori, cuvinte-cheie, elemente literale şi operatori
- tipuri de date:

❑ numerice întregi;

❑ numerice în virgulă flotantă;

❑ caracter;

❑ Boolean;

❑ Array;

❑ Obiecte;

Limbajul JavaScript permite specificarea datelor numerice în patru formate diferite: întreg, virgulă flotantă, octal şi hexazecimal.

În JavaScript există 5 tipuri de date diferite care pot conține valori:

- Number
- String
- Boolean
- Object
- Function

Există 3 tipuri de obiecte:

- Object
- Date
- Array

și 2 tipuri de date care nu pot conține valori:

- Null
- Undefined

# Valorile speciale JavaScript

- **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.

ex: 1.79769313486231E+308 este limita pozitivă, orice valoare peste acest număr este reprezentat ca “infinity”

- **NaN** (Not a Number) este o valoare specială furnizată ca rezultat de anumite operaţii aritmetice

ex: var x = “numar” / 2;

- **null** este o valoare specială care indică absenţa valorii

ex: var x = 5 + null;

- **undefined** este o valoare specială nedefinită

ex: var x;

- constante şi variabile

const nume = valoare;

var nume [= valoare];

**Ex.**

const X=‘Java Script’;

var i, j, k;

var mesaj;

**Obs.** JavaScript nu cere să se definească tipul de date şi se pot atribui diferite tipuri de date aceleiaşi variabile**.**

**Ex1.**

var x;

x = 4 + 5;

document.writeln(x);

x = “sir de caractere";

document.writeln(x);

# OPERATORII IN JS

1. **Operatori aritmetici: adunare (+), scadere (-), inmultire (*), impartire (/), modulul (%), incrementare (++), decrementare (--)**
2. **Operatori de atribuire : =**

   **+=  -=  *=  /=  %=**
3. **Operatori de comparare: ==, ===, !=, !==, <, >, <=, >=**
4. **Operatori logici : &&, ||, !**
5. **Operatori pentru șiruri: +**

   **t1 = "Astazi este o zi" ;**
   **t2 = " frumoasa ";**
   **t3 = t1+t2;**
6. **Operatorul typeof** - întoarce tipul de date conţinut la momentul respectiv de operandul său. Este util pentru a determina dacă o variabilă a fost definită
7. **Operator condițional ?**:

var result = (nr == 100) ? "Egal" : "Diferit";

6. **Operatori pentru funcții** () ,
7. **Operatori pentru structuri de date . [ ]**

# Precedența operatorilor

| Operator | Nume operator |
|---|---|
| () [] . | de apelare, pt. structuri de date |
| ! ++ -- | de negare, incrementare/decrementare |
| * / % | de inmultire, impartire, rest |
| + - | de adunare, scadere |
| < <= > >= | de comparatie |
| == != | de egalitate |
| && | SI logic |
| \|\| | SAU logic |
| ? : | conditional |
| = += -= *= /= %= | de atribuire |
| , | virgula |

# Operatorul +

- poate fi utilizat pentru a concatena variabile tip şir de caractere (string sau text)

**Regulă:** Dacă se adună un număr cu un şir de caractere, se obţine un şir de caractere.

Ex.

1. var x = 5 + 2 + 3;
2. var x =  " Curs" + " " +  " JavaScript";
3. var x = "5" + 2 + 3;
4. var x = 5 + 2 + ”3”;

```
<!DOCTYPE html>
<html>
<head> <meta charset="utf-8" />
</head>
<body>
<h1> Variabile JavaScript</h1>
<p>Aduna "Curs" + " " +
"JavaScript" și afișează rezultatul:
</p>
<p id="demo"></p>
<script>
var x = "Curs" + " " + "JavaScript";
document.getElementById("demo")
.innerHTML = x;
</script>
</body> </html>
```

Rezultat

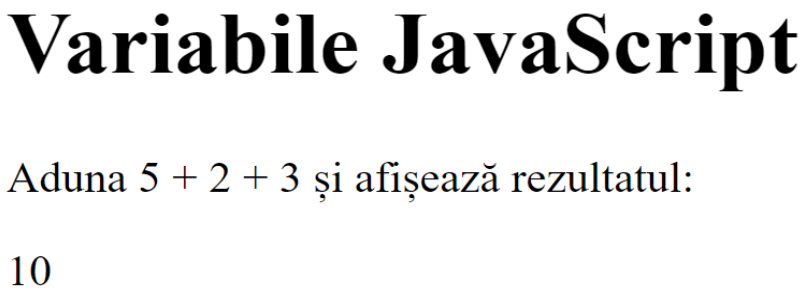
# Variabile JavaScript

Aduna "Curs" + " " + "JavaScript" și afișează rezultatul:

Curs JavaScript

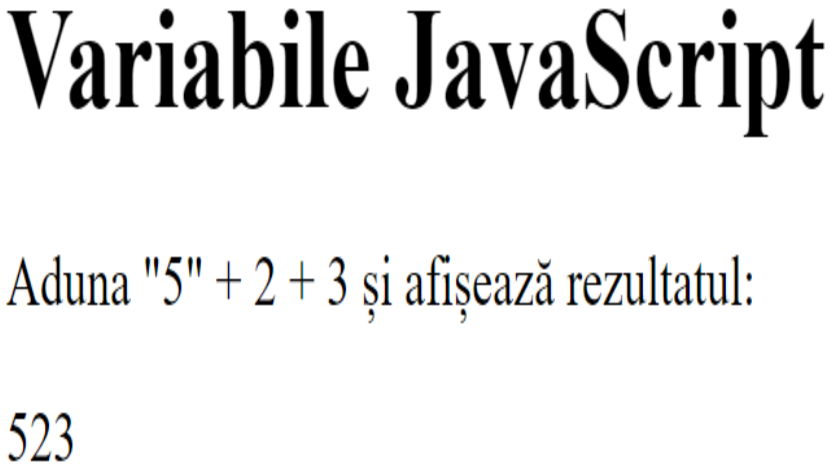
```
<!DOCTYPE html>
<html>
<head> <meta charset="utf-8" />
</head>
<body>
<h1> Variabile JavaScript</h1>
<p>Aduna 5 + 2 + 3 și afișează rezultatul:</p>
<p id="demo"></p>
<script>
var x = 5 + 2 + 3;
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

Rezultat



```
<!DOCTYPE html>
<html>
<head>  <meta charset="utf-8" />
</head>
<body>
<h1> Variabile JavaScript</h1>
<p>Aduna "5" + 2 + 3 și afișează rezultatul:</p>
<p id="demo"></p>
<script>
var x = "5" + 2 + 3;
document.getElementById("demo")
.innerHTML = x;
</script>
</body>
</html>
```

Rezultat



- Pentru conversia tipurilor de date, JavaScript are 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 nu este găsit la începutul şirului de caractere nici un număr, funcţiile returnează valoarea NaN (Not a Number).

```
var a = parseInt("10") ; // 10
var b = parseInt("10.33") ; //10
var c = parseInt("34 45 66"); //34
var d = parseInt(" 60 "); //60
var e = parseInt("40 years"); //40
var f = parseInt("He was 40") //NaN
```

# Instrucţiuni

- **Instrucțiuni condiționale: if, switch**

if (condiţie) {

*// codul ce se va executa dacă condiţia este adevarată…*

} else {

*// codul ce se va executa dacă condiţia este falsă…*

}

**ex_if.html** **ex_switch.html**

switch(*expresie*) {

case *x*:

*// cod … când valoarea expresiei este egală cu valoarea cazului x*

break;

case *y*:

*// cod … când valoarea expresiei este egală cu valoarea cazului y*

break;

default:

*// cod … când valoarea expresiei nu este egală cu nici o valoare a unui caz*

}

- **Instructiuni ciclice (repetitive)**

**for -** ex_for.html

**for ... in**

– utilizată pentru a parcurge elementele unui tablou sau a enumera proprietăţile unui obiect

- execută câte un set de instrucțiuni pentru fiecare proprietate dintr-un obiect

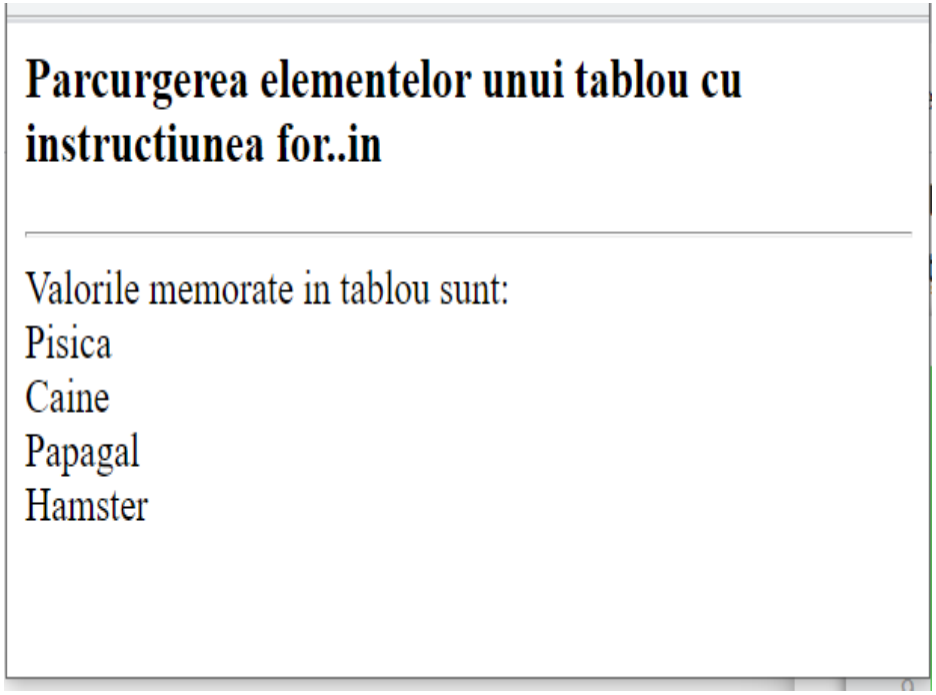
**for (*nume_proprietate* in *obiect*) {**
**instructiuni**
**}**

Ex. var a = [1,2,3,4,5];
for (x in a){..
…..
}

ex_for_in.html

```
<html>
<body>
<h3>Parcurgerea elementelor unui tablou cu instructiunea for..in</h3>
<hr/>
<script>
var x;
var pets = new Array();
pets[0] = "Pisica";
pets[1] = "Caine";
pets[2] = "Papagal";
pets[3] = "Hamster";
document.write("Valorile memorate in tablou sunt:"+"<br/>");
for (x in pets)
{  document.write(pets[x] + "<br />");
}
</script>  </body>  </html>
```

**Rezultat**



**while –** repetă codul atâta timp cât o anumită condiție este adevărată

ex_while.html

**do ... while** - execută o dată codul apoi il repetă atâta timp cât o anumită condiție este adevărată

ex_do_while.html

## Alte instrucțiuni

- **break** - intrerupe definitiv execuția unui ciclu

ex_break.html

- **continue** - întrerupe execuţia iteraţiei curente şi sare la următoarea iteraţie ex_continue.html.
- **with** - se foloseste pentru a fi evitată specificarea repetată la referirea unui obiect, când accesăm metodele sau proprietățile acestuia ex_with.html

# FERESTRE POP-UP

- ALERT
- PROMPT
- CONFIRM

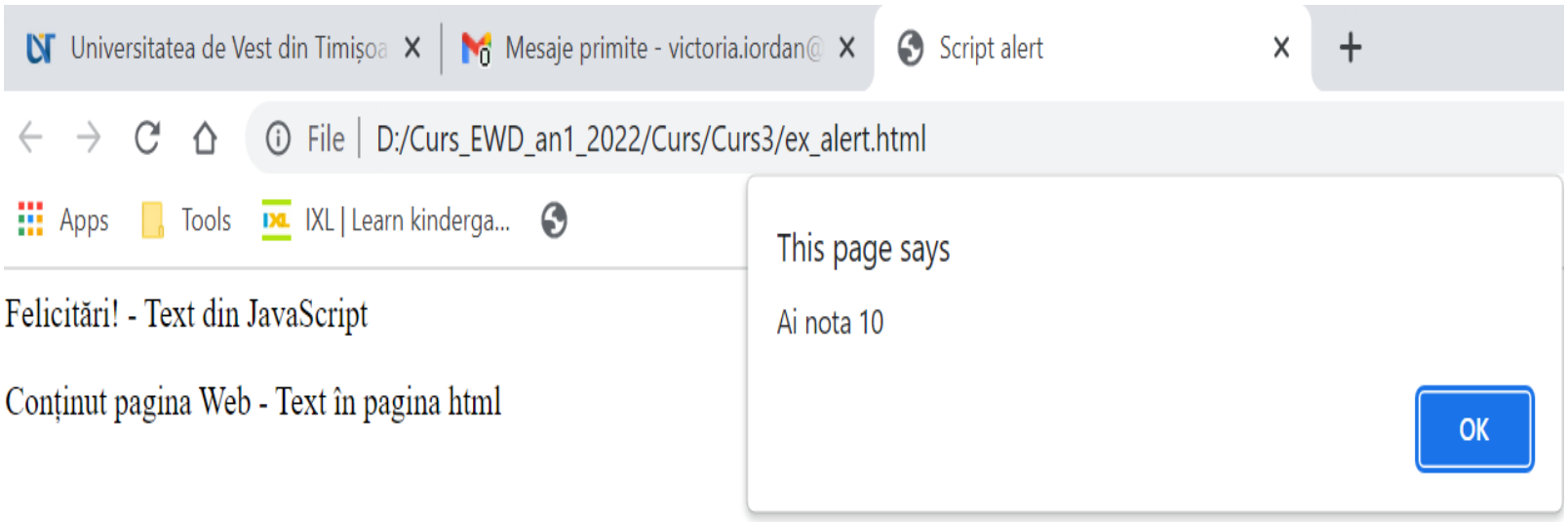
# Fereastra Alert

- este o fereastră predefinită de dialog și aparține direct obiectului "Window"
- O fereastră ALERT este folosită atunci când dorim să informăm utilizatorul printr-un mesaj. Când o fereastră ALERT apare, utilizatorul trebuie să acționeze butonul "***OK***".
- Sintaxa:

**window.alert("mesaj")**

ex_alert.html

```
<!DOCTYPE html>
<html><head><meta charset="utf-8" />
<title> Script alert</title>
<script>
    var nota=10;
    if (nota==10){
            alert("Ai nota 10");
            }
    document.write("Felicitări! - Text din JavaScript");
</script> </head>
<body>
<p>
<p>Conținut pagina Web - Text în pagina html
</body> </html>
```



# Fereastra Prompt

Fereastra PROMPT este utilizată atunci când doriți ca utilizatorul să introducă o anumită valoare inainte de a accesa pagina. Când o fereastră PROMPT apare, utilizatorul poate să modifice valoarea de intrare, apoi să acționeze unul dintre cele două butoane: “OK” sau “Cancel”

- Butonul "***OK***" returnează **valoarea de intrare**
- Butonul "***Cancel***" returnează **null**.

## Sintaxa:

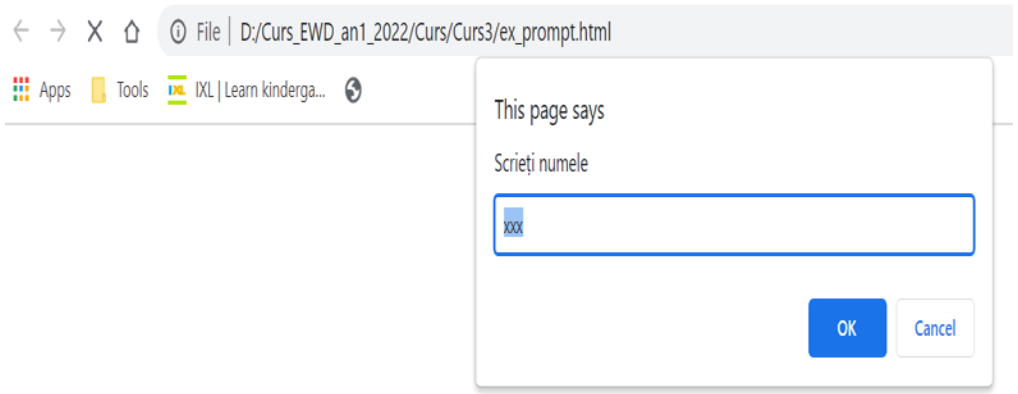
window.prompt("*mesaj* ", " default ");

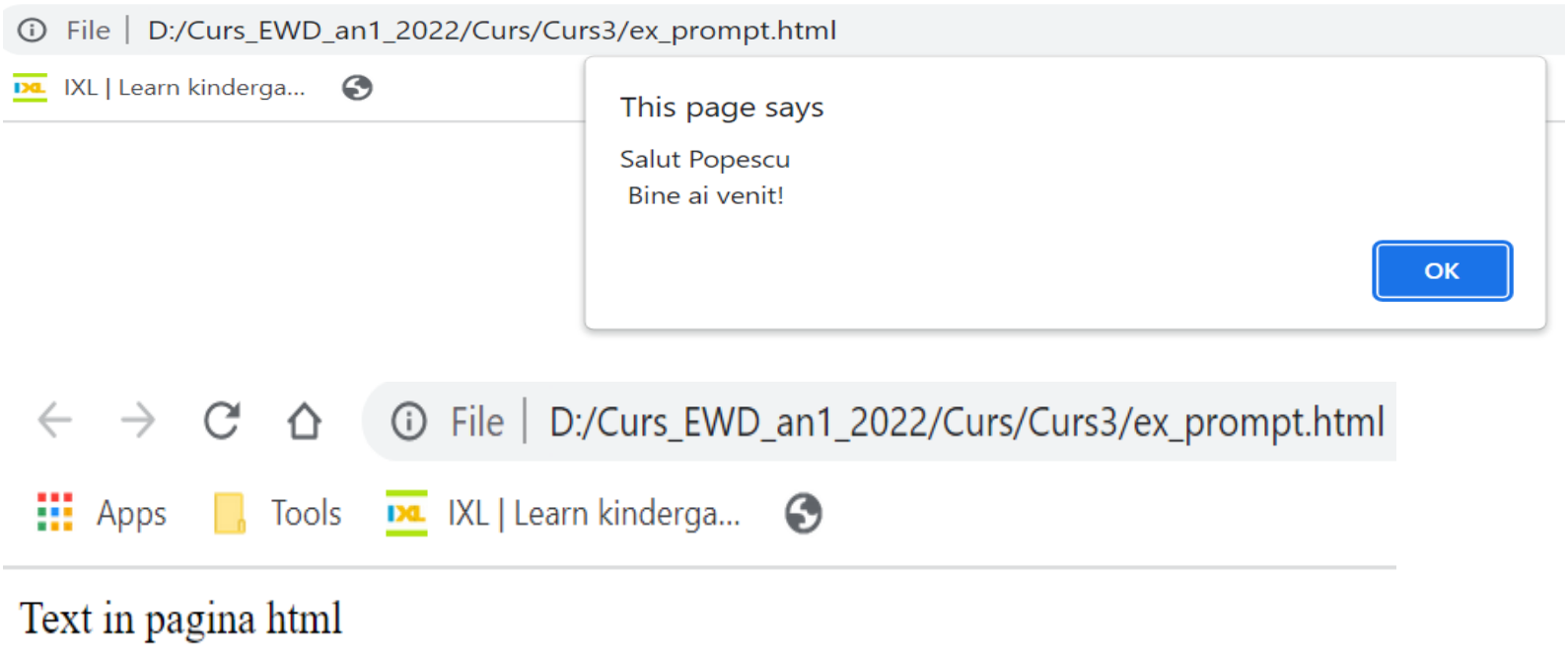
**unde "mesaj" este un text care va apare in fereastră, deasupra unei căsuțe de text input; iar "default" este textul care va apare in căsuța input**

**ex_prompt.html**

...

```
<script>
var nume = window.prompt("Scrieți numele  ", "xxx");
alert("Salut "+nume+"\n Bine ai venit!");
</script>
...
<body> Text in pagina html /body>
```



# Fereastra Confirm

- Fereastra CONFIRM este folosită, dacă doriţi ca utilizatorul să verifice sau să accepte un mesaj.
- Utilizatorul poate să folosească unul dintre butoanele “OK” sau “Cancel”.

Butonul "***OK*** " returnează **true**.

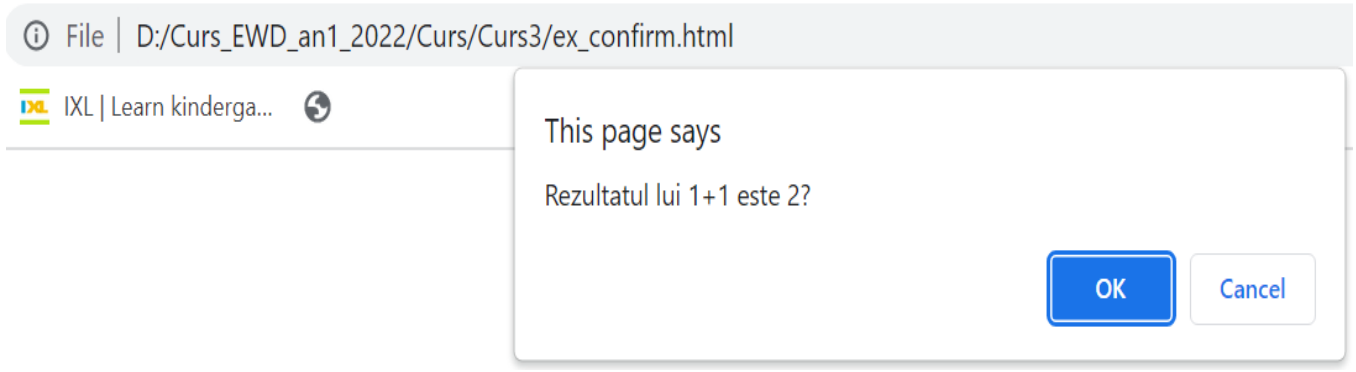
Butonul "***Cancel***" returnează **false**.

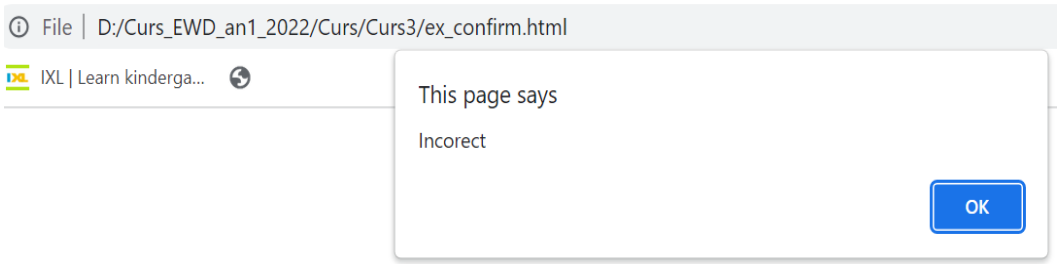
- Sintaxa:

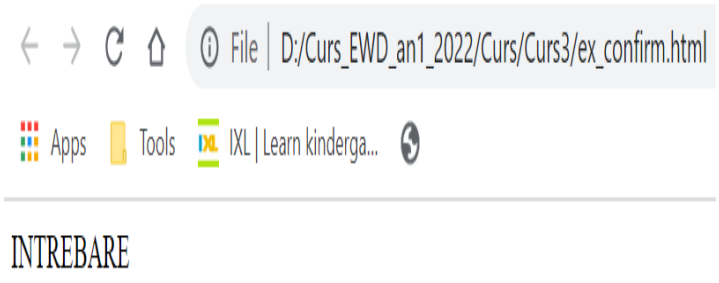
**window.confirm("mesaj")**

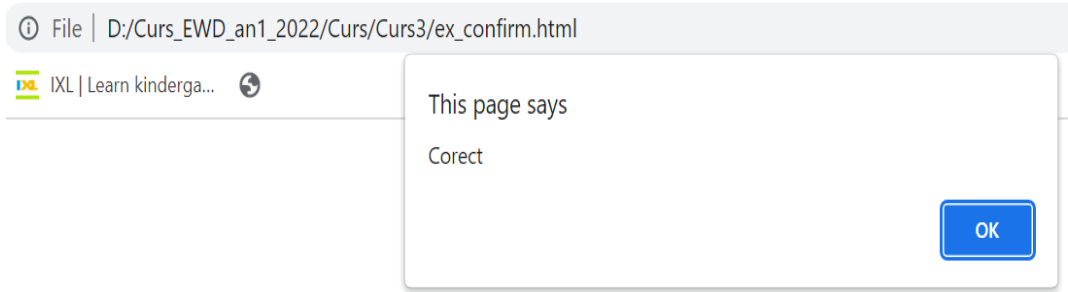
**in fereastra de confirmare va apare textul "mesaj" și două butoane "*OK*" și "*Cancel*"** ex_confirm.html

```
<script>
var intrebare="incorect";
while (intrebare != "corect")
{  intrebare = window.confirm("Rezultatul lui 1+1 este 2?");
    if (intrebare)
    { alert("Corect");
      break;
    }
   else
   alert("Incorect");
}
</script>
<body> INTREBARE </body>
```



# FUNCŢII

✓ O funcţie reprezintă o secvenţă de cod ce îndeplineşte o anumită funcţionalitate.

✓ O funcție se defineşte într-un fișier HTML în interiorul etichetelor <script>…</script> sau într-un fişier .js

✓ O funcţie se poate apela de câte ori este nevoie, cu un număr diferit de argumente sau valori ale acestora, pentru a produce diferite rezultate.

O funcţie se defineşte astfel:

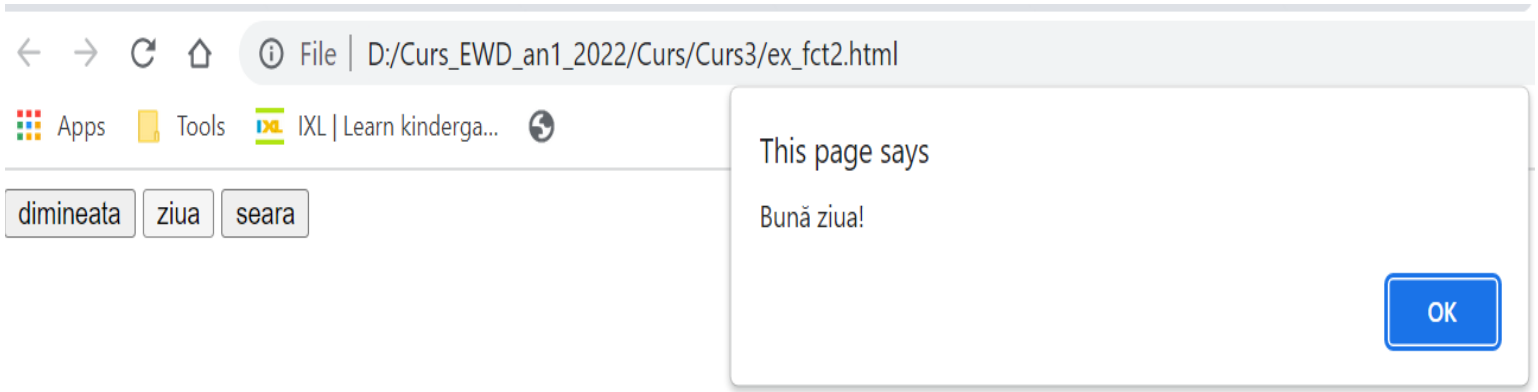
```
function nume_functie([argument1, argument2,
   ...])
 {
    // codul ce va fi executat
  }
```

## Apelul funcțiilor

```
nume_functie()
nume_functie(argument1, argument2, ...)
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
    <script>
        function exemplu1() {
            return document.write("Bine ați venit!")
        }
    </script>
</head>
<body>
    <script>
        exemplu1()
    </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html> <head> <meta charset="utf-8" />
    <script type="text/javascript">
        function exemplu2(text) {
            alert(text);
        }
    </script> </head>
<body>
  <form>
  <input type="button" onclick="exemplu2('Bună dimineața!')" value="dimineata" />
  <input type="button" onclick="exemplu2('Bună ziua!')" value="ziua" />
  <input type="button" onclick="exemplu2('Bună seara!')" value="seara" />
  </form>
</body>
</html>
```



```
<!DOCTYPE html> <html> <head>
<script>
 function suma(a,b)
  {
  return a+b;
  }
</script>
</head> <body>
<h3>Suma următoare este calculată și returnată de o funcție</h3> <hr/>
<script>
document.write("7+9="+suma(7,9));
</script> </body> </html>
```

**Suma următoare este calculată și returnată de o funcție**

---

7+9=16

```
<!DOCTYPE html> <html> <head>
<script>
   function inm(x,y)
   {  var rez=x*y;
      return rez;
   }
</script> </head><body> <b>Tabla inmulțirii cu 9:</b><br>
<script>
  for(var i=1;i<=10;i++)
   { document.write("9 x"+i+"= "+inm(9,i));
     document.write("<br>");
   }
</script>
</body> </html>
```

**Tabla inmulțirii cu 9:**

9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

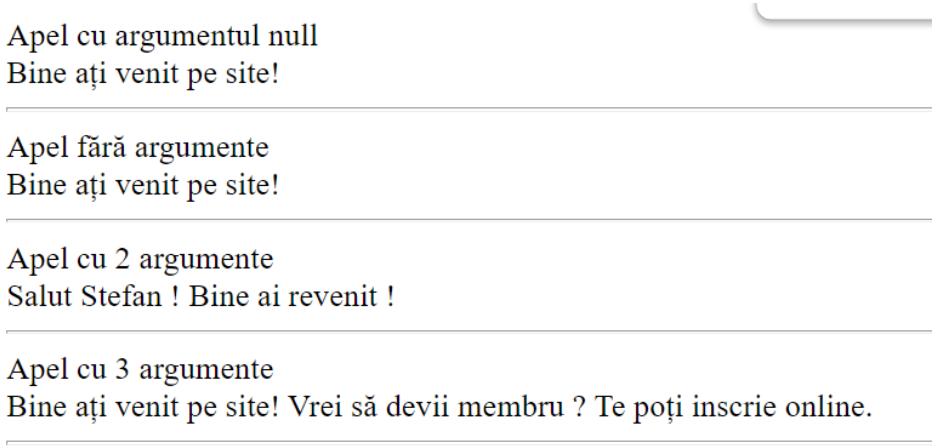
- **Modificarea numărului argumentelor**
- **Mai multe argumente decât sunt specificate – tablou numit “arguments”**
- **Funcții recursive**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<script >
    function mesaj(utilizator) {
       if (utilizator!=null) {
document.writeln("Salut " + utilizator);
          }
         else {
 document.writeln("Bine ați venit pe site!");
          }
       }
</script> </head>
<body>
<script >
document.writeln("Prima apelare a funcției mesaj() cu 1 argument <br />");
   mesaj("Studenți!");
document.writeln("<br />A doua apelare a funcției mesaj() fără argumente <br />");
   mesaj();
</script>
</body>
</html>
```

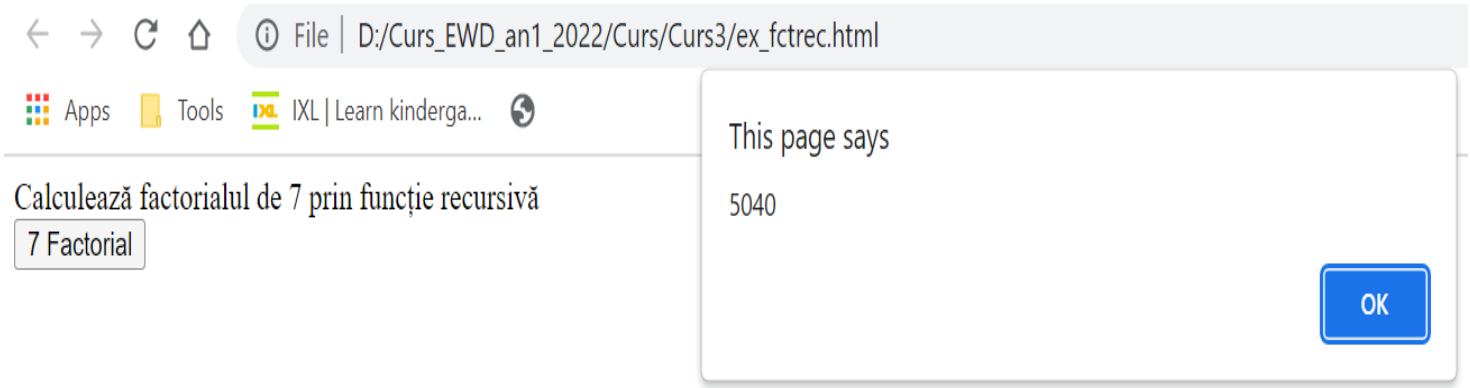
***Prima apelare a funcției mesaj() cu 1 argument***
***Salut Studenți!***
***A doua apelare a funcției mesaj() fără argumente***
***Bine ați venit pe site!***

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<script >
function mesaj(utilizator) {
    if (utilizator!=null) {
 document.writeln("Salut "+utilizator);
    }
    else {
  document.writeln("Bine ați venit pe site!");
    }
numarArgumente=mesaj.arguments.length;
    if(numarArgumente>1) {
        for (var i=1; i<numarArgumente; i++) {
   document.writeln(mesaj.arguments[i]);
        }    } }
</script>
</head>
```

```
<body>
<script >
var utilizator0=null;
document.writeln(" Apel cu argumentul null  <br />");
mesaj(utilizator0);
document.writeln("<hr>");
document.writeln("Apel fără argumente <br />");
mesaj();
document.writeln("<hr>");
document.writeln("Apel cu 2 argumente <br />");
var utilizator="Stefan !   ",
extraMesaj="Bine ai revenit !";
mesaj(utilizator,extraMesaj);
document.writeln("<hr>");
document.writeln("Apel cu 3 argumente <br />");
var utilizator1=null;
var extraMesaj1="Vrei să devii membru ?";
var extraMesaj2="Te poți inscrie online.";
mesaj(utilizator1,extraMesaj1,extraMesaj2);
document.writeln("<hr>");
</script>
</body>
</html>
```



```
<!DOCTYPE html>
<html> <head>
<script>
document.writeln("Calculează
factorialul de 7 prin funcție recursivă")
function factorial(n) {
var rezultat;
if (n>0) { rezultat = n*factorial(n-1);
    }
    else if (n==0) {  rezultat = 1;
            }
return(rezultat)
}
</script> </head>
<body>
<form>
   <input type = "button" value = "7
Factorial" onclick="alert(factorial(7))"
>
</form>
</body> </html>
```



- Obiectele din JavaScript :

  **- Obiecte esențiale**

  **- Obiecte pe partea de client**

  **- Obiecte pe partea de server**

# Obiecte esenţiale

1 - Obiectul String
2 - Obiectul Array
3 - Obiectul Date
4 - Obiectul Math
5 - Obiecte de tip Global

# Obiectul String

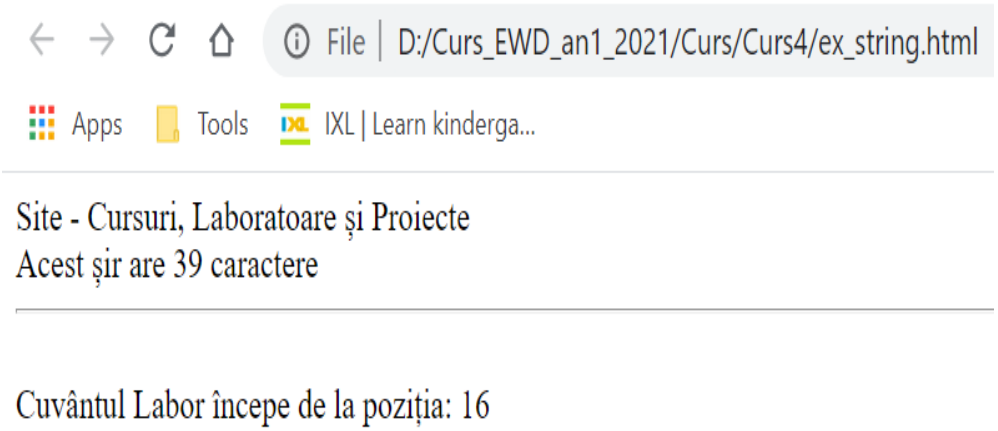
**String** (sau șir) se foloseşte pentru a prelua text.

**Proprietatea acestui obiect** este:

- **length** - returnează numărul de caractere dintr-un sir (string)

**Metodele obiectului string** sunt următoarele: metode_string.pdf

```
<!DOCTYPE html>
<html> <head>
<script >
var str="Site - Cursuri, Laboratoare și Proiecte"
document.write(str+"<br />")
document.writeln("Acest șir are "+ str.length + " caractere")
document.writeln("<hr>");
document.writeln("<br>");
var cuv="Labor";
var pos=str.indexOf(cuv);
if (pos>=0) {
document.write("Cuvântul "+ cuv +
" începe de la poziția: ")
document.write(pos + "<br />")
}
 else
 { document.write("Cuvântul "+cuv+"  nu a fost găsit!")
 }
</script>  </head> </html>
```



# Obiectul Array

## var nume=new Array(dimensiune);
## var nume=Array(val1,val2,…,valn) ;
## var nume = [ ];

**Ex.**

```
var pets=new Array();
pets[0]="Caine";
pets[1]="Pisica";
pets[2]="Papagal";
```

```
var pets=new Array("Caine","Pisica",
    "Papagal");
```

```
var pets=["Caine","Pisica",
    "Papagal"];
```

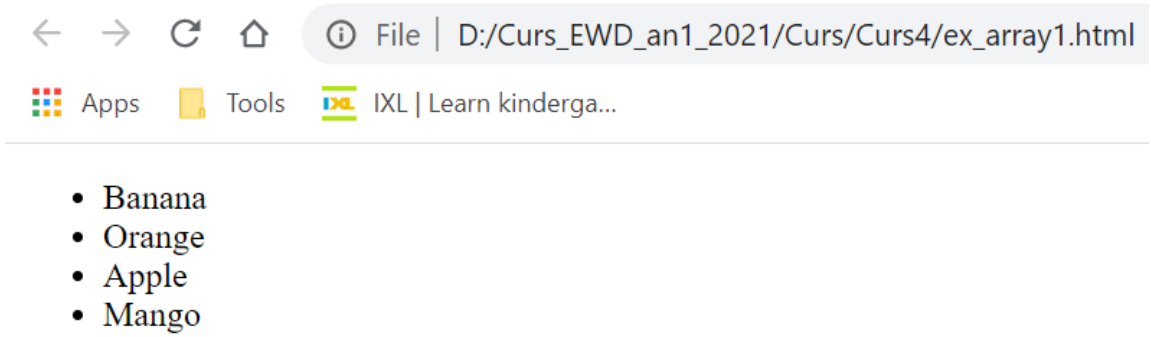
```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var cars =["Toyota ","Volvo "," BMW"];
document.getElementById("demo").inner
HTML = cars;
</script>
</body>
</html>
```

**REZULTAT**

Toyota ,Volvo , BMW

- **Proprietatea : length**

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
    text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
document.getElementById("demo").innerHTML = text;
</script> </body>
</html>
```



# Obiectul Array are următoarele metode:

- **concat()** - Returnează un tablou rezultat din concatenarea a două tablouri
- **join()** - Returnează un string format din toate elementele unui tablou concatenat

Ex.

```
<html> <body>
<p>metoda join() </p>
<p id="demo"></p>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var ex=[1,2,3];
document.getElementById("demo").innerHTML = fruits.join(ex);
</script>
</body> </html>
```

Banana1,2,3Orange1,2,3Apple1,2,3Mango

- **splice()** – adaugă/îndepărtează elemente
- **slice()** - returnează o parte specificată a unui tablou
- **sort()** - returnează tabloul ordonat
- **reverse()** - returnează inversul unui tablou
- **pop()** - elimină ultimul element dintr-un array
- **push() -** adaugă un element în array (la sfârșit)
- **shift() –** elimină primul element
- **unshift() –** adaugă la început un element

# Obiectul Date

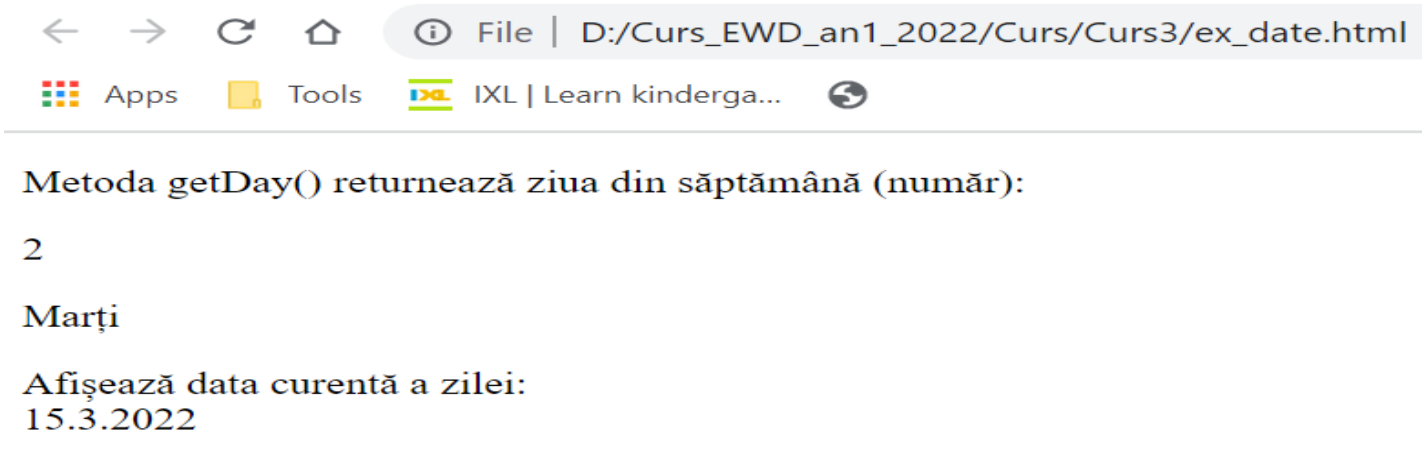
1. Data iniţială (de referinţă) este 1-01-1970
2. Când se creează un obiect "Date", ora folosită de obiect este aceea de pe calculatorul client

- **var data = new Date()**

  **new Date("Month dd, yyyy hh:mm:ss")**
  **new Date("Month dd, yyyy")**
  **new Date(yy,mm,dd,hh,mm,ss)**
  **new Date(yy,mm,dd)**
  **new Date(milliseconds)**

**Metodele obiectului Date:** metode_date.pdf

```
<!DOCTYPE html>
<html> <head>
<title>ex obiect Date</title> </head>
<body>
<p>Metoda getDay() returnează ziua din săptămână (numar):</p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
var d = new Date();
document.getElementById("demo1").innerHTML = d.getDay();
var days = ["Duminică","Luni","Marți","Miercuri","Joi","Vineri","Sâmbătă"];
document.getElementById("demo2").innerHTML = days[d.getDay()];
document.write("Afișează data curentă a zilei: <br /> ");
document.write(d.getDate());
document.write(".");
document.write(d.getMonth() + 1);
document.write(".");
document.write(d.getFullYear());
</script>
</body> </html>
```



# Obiectul Math

- Acest obiect include constante matematice și functii.
- Nu este nevoie să fie creat (instanțiat) un **obiect Math** înainte de a fi folosit.
- Proprietățile obiectului Math **E** - Returnează constanta lui Euler (2.7182.....)

**LN2** - Returnează logaritm natural din 2

**LN10** - Returnează logaritm natural din 10

**LOG2E** - Returnează logaritm in baza 2 din E

**LOG10E** - Returnează logaritm in baza10 din E

**PI** - Returnează PI

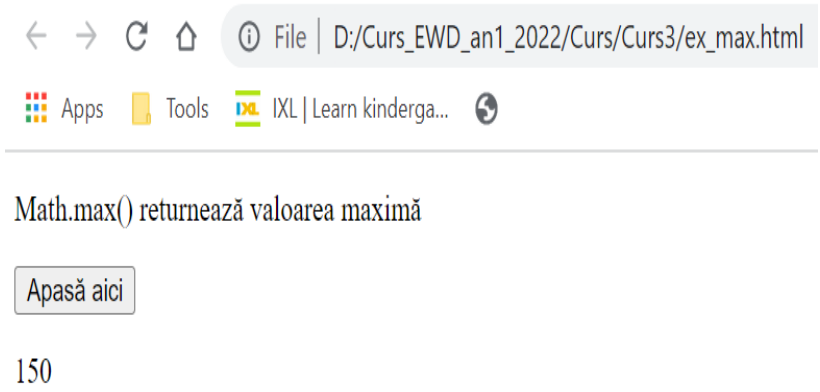
**SQRT1_2** - Returnează radical din 0.5

**SQRT2** - Returnează radical din 2

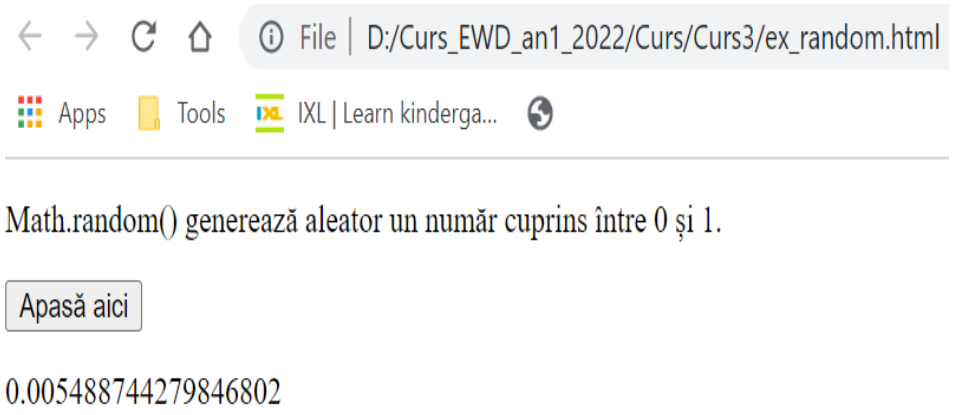
- Metodele obiectului Math:

**abs(x)** - Returnează valoarea absolută din x
**acos(x)** - Returnează arccosinus din x
**asin(x)** - Returnează arcsinus din x
**atan(x)** - Returnează arctangenta din x
**atan2(y,x)** - Returnează unghiul dintre axa și un punct (x,y)
**ceil(x)** - Returnează cel mai apropiat intreg mai mare sau egal cu x
**cos(x)** - Returnează cosinus din x
**exp(x)** - Returnează valoarea lui E la puterea x
**floor(x)** - Returnează cel mai apropiat intreg mai mic sau egal cu x
**log(x)** - Returnează log natural din x
**max(x,y)** - Returnează maximul dintre x și y
**min(x,y)** - Returnează minimul dintre x și y
**pow(x,y)** - Returnează valoarea lui x la puterea y
**random()** - Returnează un număr aleator intre 0 si 1
**round(x)** - Rotunjeste pe x la cel mai apropiat intreg
**sin(x)** - Returnează sinus din x
**sqrt(x)** - Returnează radical din x
**tan(x)** - Returnează tangenta din x

```
<!DOCTYPE html> <html><body>
<p>Math.max() returnează valoarea maximă</p>
<button onclick="myFunction()">
Apasa aici</button>
<p id="demo"></p>
<script>
function myFunction() {
document.getElementById("demo").innerHTML =
    Math.max(0, 150, 30, 20, -8, -200);
}
</script></body></html>
```




```
<!DOCTYPE html><html> <body>
<p>Math.random() generează aleator un număr cuprins între 0 și 1.</p>
<button onclick="myFunction()">Apasă aici</button>
<p id="demo"></p>
<script>
function myFunction() {
document.getElementById("demo").innerHTML = Math.random();
}
</script> </body></html>
```



# Obiecte de tip Global

**Obiectul Global** are trei proprietăţi:

- **Infinity** - cuvânt-cheie care reprezintă plus sau minus infinit
- **NaN** - reprezintă un obiect null, care nu are vreo valoare
- **undefined** - indică dacă o variabilă a fost sau nu definită

Metodele acestui obiect (care pot fi considerate și funcții JavaScript):

- **eval()** - acceptă un șir de instructiuni Java Script și il evaluează ca fiind cod sursă
- **isFinite()** - determină dacă o variabilă are limite finite
- **isNaN()** - determină dacă o variabilă este sau nu un număr.
- **Number()** – convertește valoarea unui obiect in număr
- **parseFloat()** - transformă un șir intr-un număr de tip float (cu virgula)
- **parseInt()** - transformă un șir intr-un număr intreg
- **String()** – convertește valoarea unui obiect in string(de tip șir)