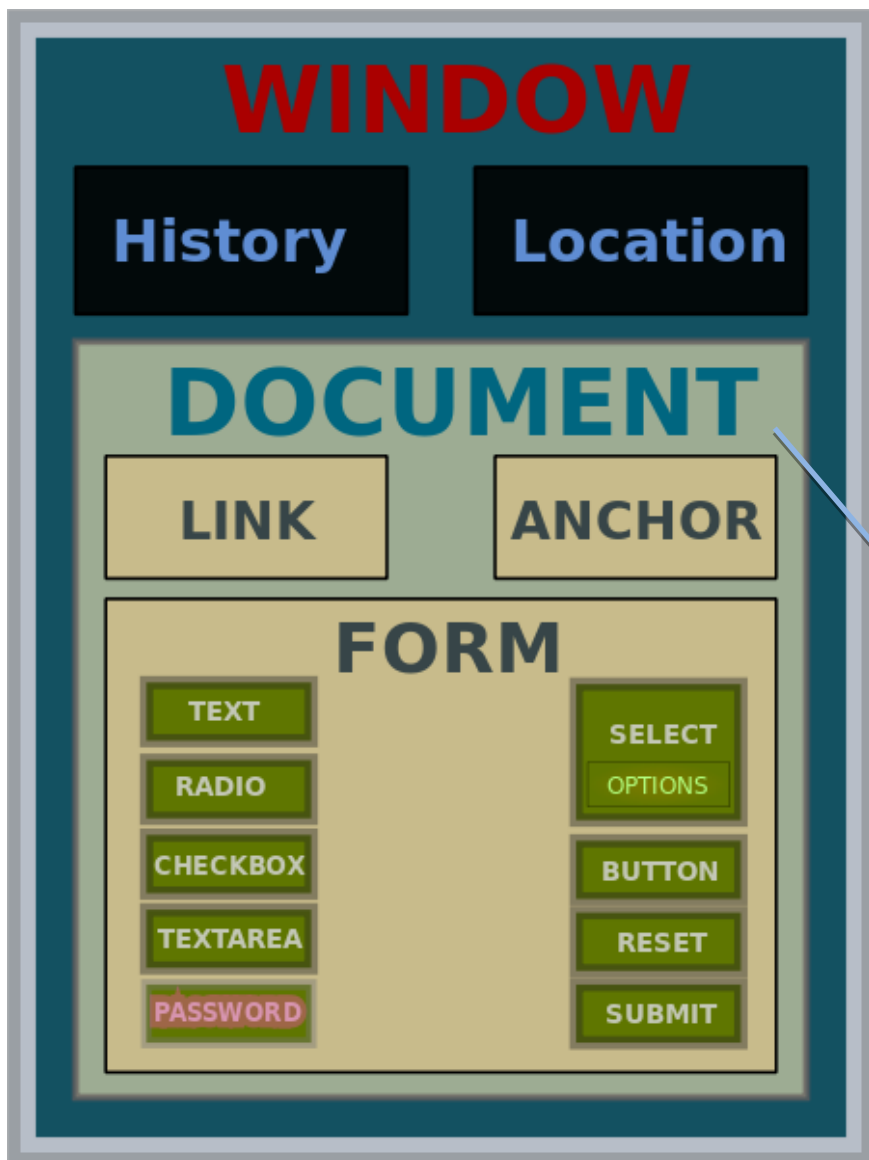


# Tehnici Web

## CURSUL 6

Semestrul II, 2021-2022  
Carmen Chirita

<https://sites.google.com/site/fmitehnicweb/>



Orice tab al unui browser contine un obiect `window`, care apartine clasei `Window`; proprietatea `document` a obiectului `window` are ca valoare obiectul `document` asociat paginii web.

Referinta completa ar fi `window.document`  
Alte proprietati: `history`, `location`, `console`, `localStorage`, `sessionStorage`

**DOM**

Sursa imaginii

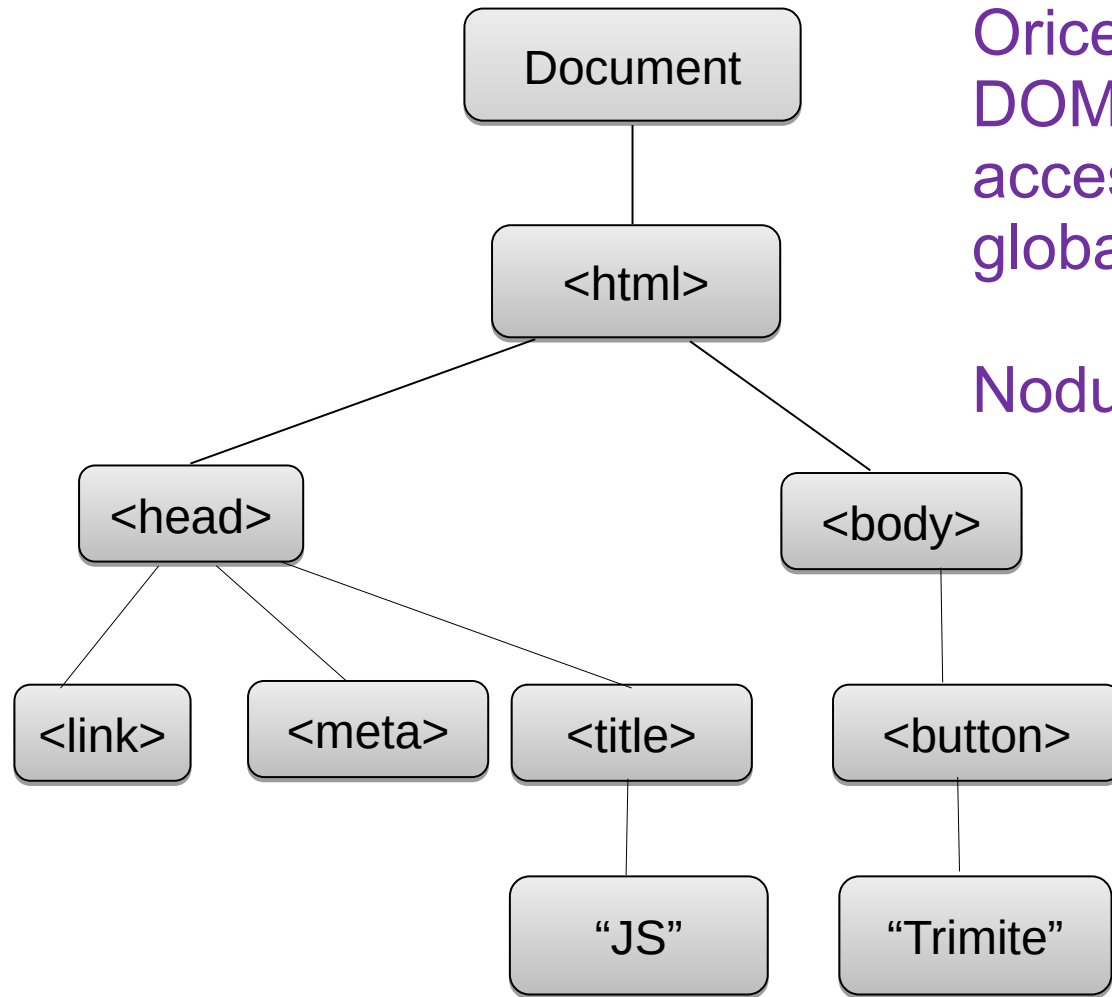
<https://developer.mozilla.org/en-US/docs/Web/API/Window>

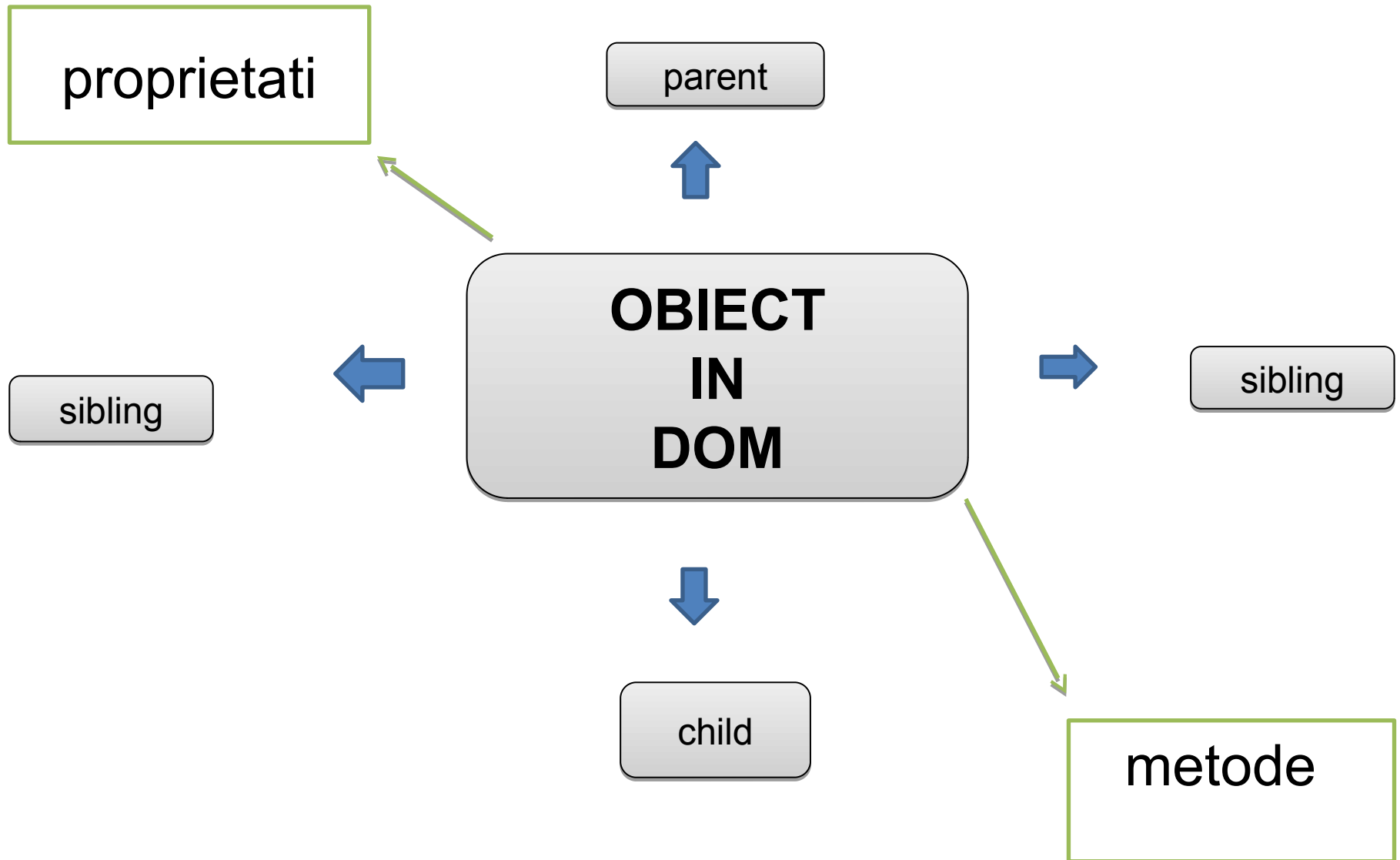
# DOM = Document Object Model

## JavaScript folosește DOM pentru accesarea și modificarea documentelor HTML

Orice pagina web este reprezentată în DOM ca un arbore de obiecte și este accesată folosind variabila globală `document`.

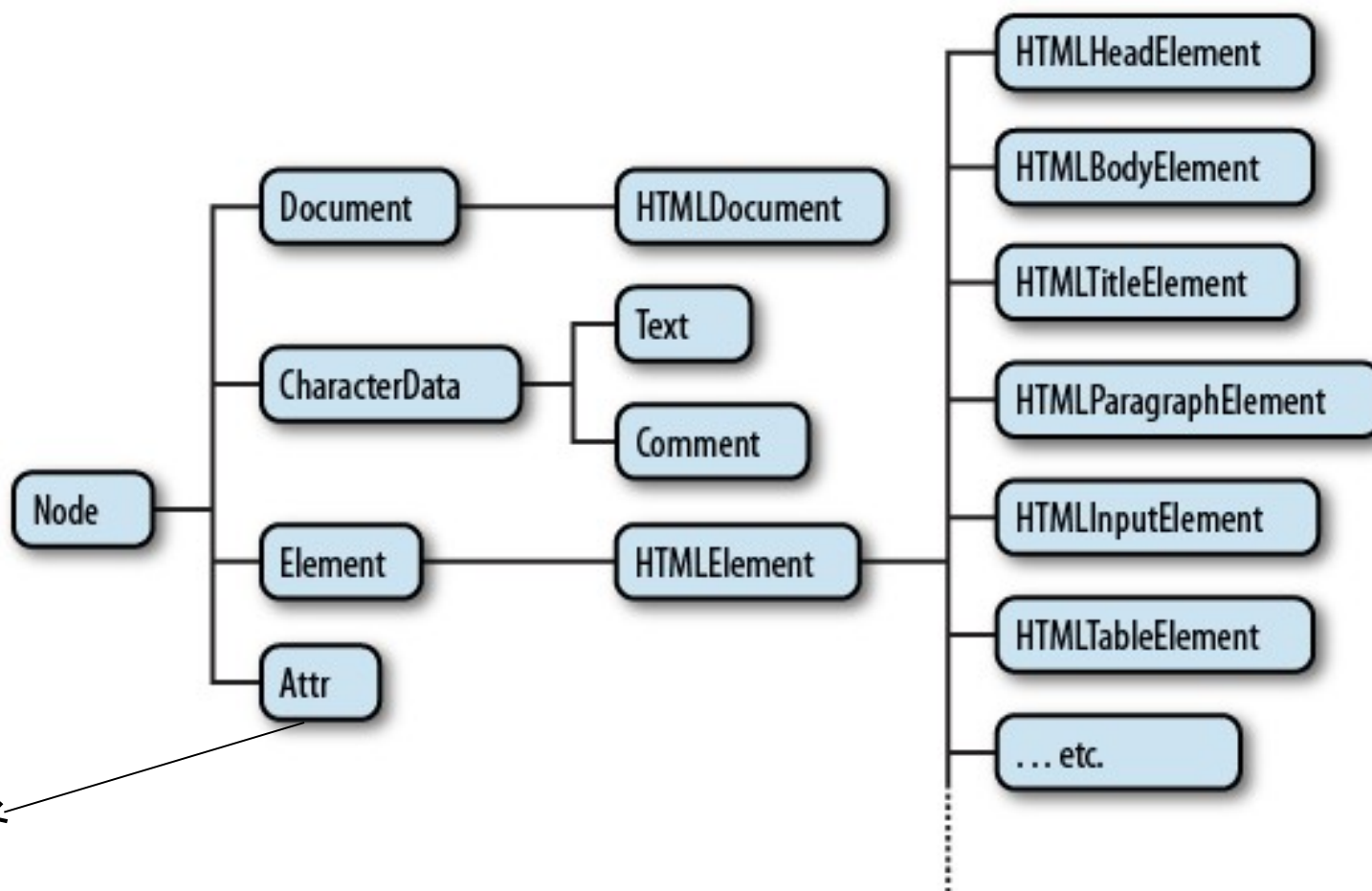
Nodurile conțin taguri HTML sau text.





Orice nod din arborele DOM are un tip (clasa); in functie de elementul pe care il reprezinta, obiectul corespunzator are proprietati si metode specifice.

## Ierarhia claselor pentru nodurile din DOM



Atributele sunt manevrate prin proprietati si metode ale clasei Element (nu ca obiecte din clasa Attr)

# DOM

unui tag HTML ii corespunde un obiect (derivata din Element)

unui atribut al tag HTML ii corespunde o proprietate a obiectului

atributele(generale) HTML **id, class, style**

corespund

proprietatilor **id, className, style** (ale obiectului corespondent)

proprietatile obiectului style (din clasa Style) asociat atributului style

corespund

proprietatilor de stilizare CSS pentru elementele HTML

**element.style.proprietateCSS=valoare**

**el.style.color="red"**

**el.style.backgroundColor="blue"**

## EXEMPLU

<p> Imagine cu DOM



</p>

Obiect Element  
in DOM

tagName: "img"  
src: "dom.jpg"  
alt: "dom"  
id: "myimg"

```
function schimb(){  
    var x=document.getElementById('myimg'); //obiect  
    x.alt = "sunt dom";  
    x.style.border="2px solid red";  
}
```

Elementul `<script>` din `<head>` este procesat inaintea elementului `<body>`;

in acel moment arborele DOM nu este creat si elementele lui nu pot fi accesate.

Incarcarea paginii declanseaza evenimentul `load` care are ca tinta obiectul `window`; un handler pentru evenimentul `load` se inregistreaza setand proprietatea `onload` pentru obiectul `window`.

```
window.onload=function()  
{cod JavaScript;}
```

sau

```
window.onload=myMain;  
function myMain()  
{cod JavaScript;}
```



# EXEMPLU

```
window.onload = schimb;
```

```
function schimb(){  
    var x=document.getElementById('myimg');  
    x.alt = "sunt dom";  
    x.style.border="2px solid red";  
}
```

script.js

```
<!DOCTYPE html>  
<html lang="ro">  
<head>  
<script type="text/javascript" src="script.js"></script>  
<meta charset="utf-8">  
<title> JS si CSS</title>  
</head>  
<body>  
....  
  
.....  
</body>  
</html>
```

ex\_dom.html

## Selectarea elementelor in document

`document.getElementById( numId) // un obiect`  
`document.querySelector(selectorCss) //primul obiect`

-colectii “live”:

`document.getElementsByClassName(umeClasa)`

`document.getElementsByTagName(umeTag)`

`document.getElementsByName(ume)`

-colectii “static”

`document.querySelectorAll(selectorCss)`

Colectii= organizare ca Array  
au proprietatea – length  
nu pot invoca direct –metodele Array

Exemplu: afisati numarul elementelor <h1> care sunt  
descendenti directi (copii) ai elementelor <section> cu  
clasa "special"

Solutie:

```
var colectie = document.querySelectorAll("section.special > h1");  
alert(colectie.length);
```

Ex: colecție live

```
<script>
window.onload=function() {

var c=document.getElementsByClassName("abc");
var v=[];
for(var i=0;i<c.length;i++) v[i]=c[i];
alert("lungimea colectiei: " + c.length + '\n' +
"lungimea vectorului: " + v.length);

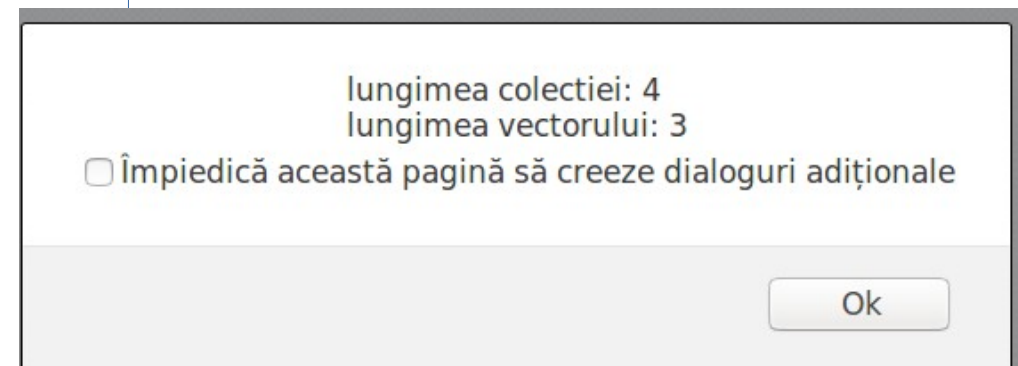
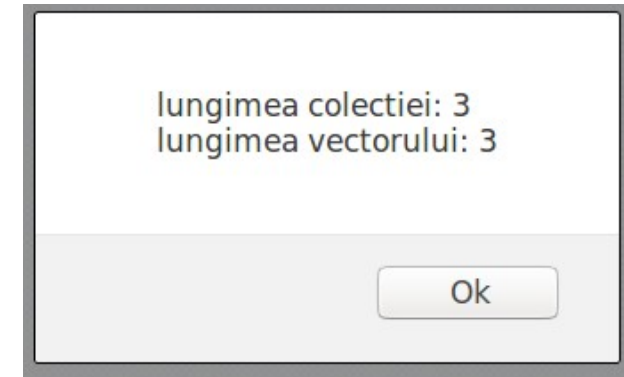
document.getElementById("p1").className="abc";

alert("lungimea colectiei: " + c.length + '\n' +
"lungimea vectorului: " + v.length);

}
</script>
</head>

<body>
<p id="p1">Paragraful 1</p>
<p class="abc">Paragraful 2</p>
<p class="abc">Paragraful 3</p>
<p class="abc">Paragraful 4</p>

</body>
```



Ex: colecție static

```
<script>
window.onload=function() {

var c=document.querySelectorAll(".abc");
var v=[];
for(var i=0;i<c.length;i++) v[i]=c[i];
alert("lungimea colectiei: " + c.length + '\n' +
"lungimea vectorului: " + v.length);

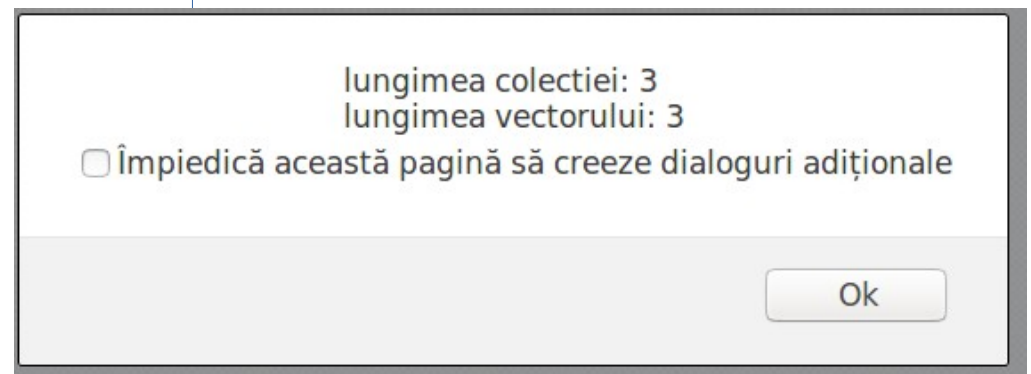
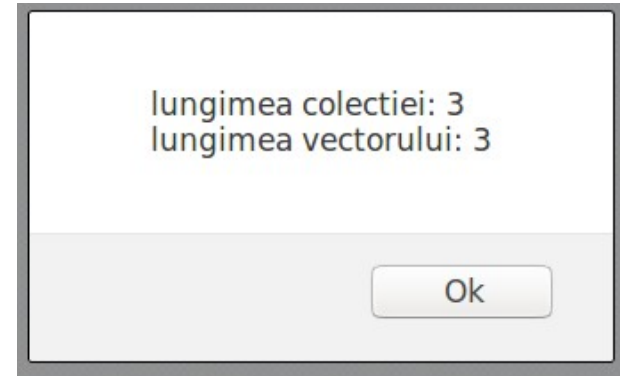
document.getElementById("p1").className="abc";

alert("lungimea colectiei: " + c.length + '\n' +
"lungimea vectorului: " + v.length);

}
</script>
</head>

<body>
<p id="p1">Paragraful 1</p>
<p class="abc">Paragraful 2</p>
<p class="abc">Paragraful 3</p>
<p class="abc">Paragraful 4</p>

</body>
```



Exemplu: schimbarea unei clase “vechi” in clasa “nou”.

```
var colectie=document.getElementsByClassName("vechi");  
for(var i=0; i<colectie.length;i++)  
    colectie[i].className="nou";  
    //nu functioneaza; se modifica colectia
```

Solutii:

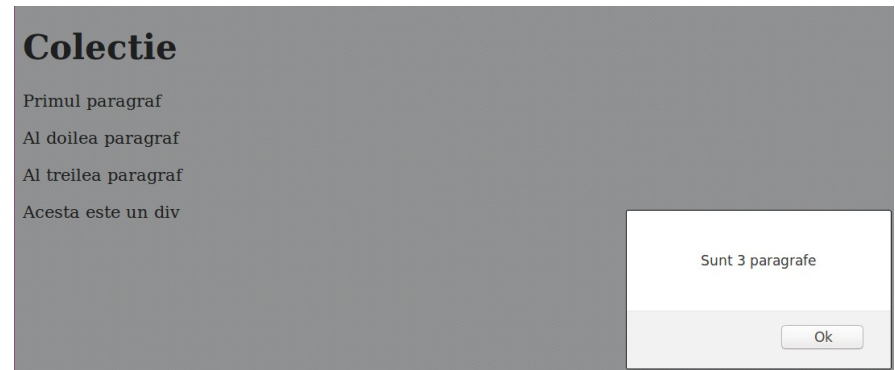
```
var colectie=document.querySelectorAll(".vechi");  
for(var i=0; i<colectie.length;i++)  
    colectie[i].className="nou";
```

sau

```
var vector=[];  
var colectie=document.getElementsByClassName("vechi");  
for(var i=0;i<colectie.length;i++)  
    vector[i]=colectie[i];  
for(var i=0;i<vector.length;i++)  
    vector[i].className="nou";
```

Exemplu: selectarea tuturor elementelor de tip „p” (paragraf) din document, afisarea numarului lor și colorarea lor în culoarea roșie (culoarea textului)

```
<script>
window.onload=function()
{
var
lista=document.getElementsByTagName("p");
alert("Sunt "+lista.length+" paragrafe");
for(var p of lista)
    {p.style.color="red";}
}
</script>
</head>
<body>
<h1>Colectie</h1>
<p>Primul paragraf</p>
<p>Al doilea paragraf</p>
<p>Al treilea paragraf</p>
<div>Acesta este un div</div>
</body>
```



## Colectie

Primul paragraf

Al doilea paragraf

Al treilea paragraf

Acesta este un div

Continutul unui element poate fi accesat si modificat ca String folosind proprietatile:

**innerHTML**

intoarce un text HTML,  
adica un text cu marcate;  
la setarea proprietatii  
browserul interpreteaza textul;

**si** **textContent**

intoarce text fara marcate;  
are ca rezultat concatenarea  
continuturilor descendente de  
tip Text.

`<p>Un text <i> simplu </i> si colorat. </p>`

obiect.innerHTML

Un text *simplu* si colorat.

obiect.textContent

Un text simplu si colorat.



textContent

```
<head>
<script>
  window.onload = function(){

    var el1 = document.getElementById("text1");
    var el2 = document.getElementById("text2");
    var temp = el1.textContent;
    el1.textContent = el2.textContent;
    el2.textContent = temp; }

</script>
</head>
<body>
<p id="text1"> Continutul paragrafului </p>
<div id="text2"> Continutul divului </div>
</body>
```

# innerHTML

```
<script>
window.onload=function()
{
var x=document.getElementById("text2");
var continut = [ '<ul class="myclass">',
  '<li class="item">item1</li>',
  '<li class="item">item2</li>',
  '<li class="item">item3</li>',
  '<li class="item">item4</li>',
  '<li class="item">item5</li>',
  '</ul>' ].join(" ");
x.innerHTML += continut;
}
</script>
</head>
<body>
<p id="text1"> Continutul paragrafului</p>
<div id="text2"> Continutul divului </div>
</body>
```

Continutul paragrafului

Continutul divului

- item1
- item2
- item3
- item4
- item5

# DOM - o structură ierarhică bazată pe noduri

## Tipuri de noduri:

- Un nod special numit **document**
- Noduri de tip **Element** (modeleaza tagurile <p>, ...)
- Noduri de tip **Text** (modeleaza textul)

# Manipularea informatiei din noduri

nodeValue // pentru noduri Text, Comment,  
pentru noduri Element = null

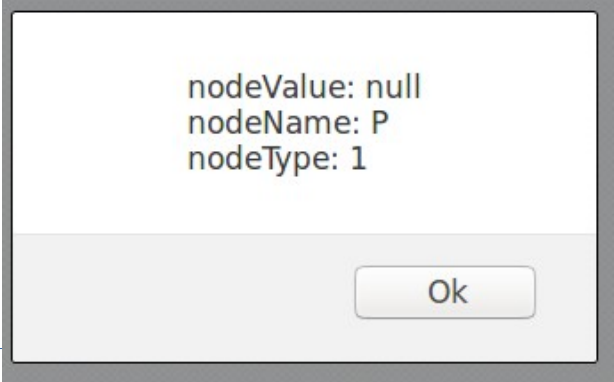
nodeName // numele tagului

nodeType /\* Document=9, Element=1,  
Text=3, Comment=8 \*/

```
<script>
window.onload = function(){

    var p = document.getElementById("par");
    alert("nodeValue: " + p.nodeValue + '\n' + "nodeName: " + p.nodeName
    + '\n' + "nodeType: " + p.nodeType);
}

</script>
</head>
<body>
<p id="par">Continutul paragrafului</p>
</body>
```



nodeValue: null  
nodeName: P  
nodeType: 1

Ok

# Navigarea prin ierarhia de noduri

`node.parentNode` // un obiect

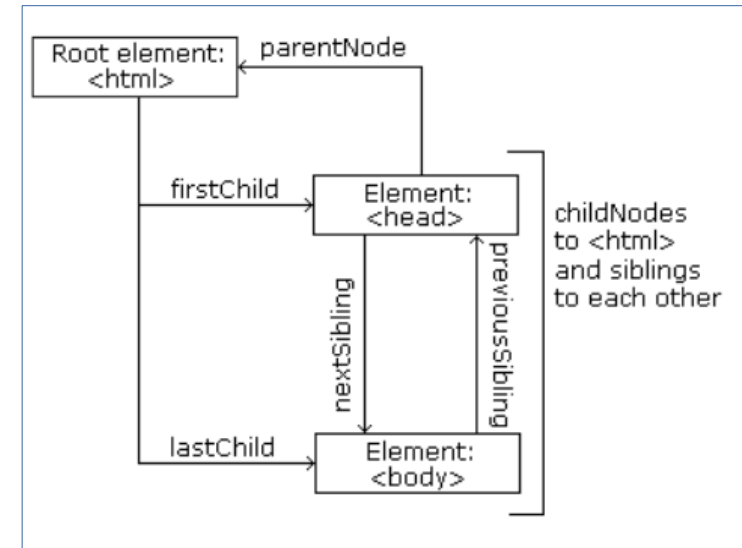
`node.childNodes` // NodeList obiect

`node.firstChild` // un obiect

`node.lastChild` // un obiect

`node.nextSibling` // un obiect

`node.previousSibling` // un obiect



```

<script>
window.onload=function() {
    var c = document.body.childNodes;
    var txt = "";
    var i;
    for (i = 0; i < c.length; i++) {
        txt = txt + c[i].nodeName + "<br>";
    }
    document.getElementById("demo").innerHTML += txt;
    alert(document.getElementById("demo").childNodes[0].nodeName);
}
</script>
</head>
<body><!-- Comentariu -->
<p>Un paragraf</p>
<div>Un div</div>
<p>O imagine</p><br>

<p id="demo"><strong>document.body.childNodes:</strong><br>
</p>
</body>

```

Un paragraf

Un div

O imagine



**document.body.childNodes:**

#comment

#text

P

#text

DIV

#text

P

BR

#text

IMG

#text

P

#text

STRONG

Ok

## Se poate parcurge documentul ca arbore de Elemente

node.parentNode / node.parentElement  
node.childNodes / node.children  
node.firstChild / node.firstElementChild  
node.lastChild / node.lastElementChild  
node.nextSibling / node.nextElementSibling  
node.previousSibling / node.previousElementSibling

se pot defini metode noi

```
function secondChild(e){  
  return e.firstElementChild.nextElementSibling;  
};
```

```

<script>
window.onload=function() {
    var c = document.body.children;
    var txt = "";
    var i;
    for (i = 0; i < c.length; i++) {
        txt = txt + c[i].nodeName + "<br>";
    }

    document.getElementById("demo").innerHTML += txt;
    alert(document.getElementById("demo").children[0].nodeName);
}
</script>
</head>

<body><!-- Comentariu -->
<p>Un paragraf</p>
<div>Un div</div>
<p>O imagine</p><br>

<p id="demo"><strong>document.body.children</strong><br></p>
</body>

```

Un paragraf

Un div

O imagine



**document.body.children**

P

DIV

P

BR

IMG

P

STRONG

Ok



```
<head>
```

```
<script>
```

```
window.onload = function(){
```

```
var b = document.getElementsByTagName("body");
```

```
var p = b[0].firstElementChild.nextElementSibling.nodeName;
```

```
  alert(p); }
```

```
</script>
```

b este Array-like

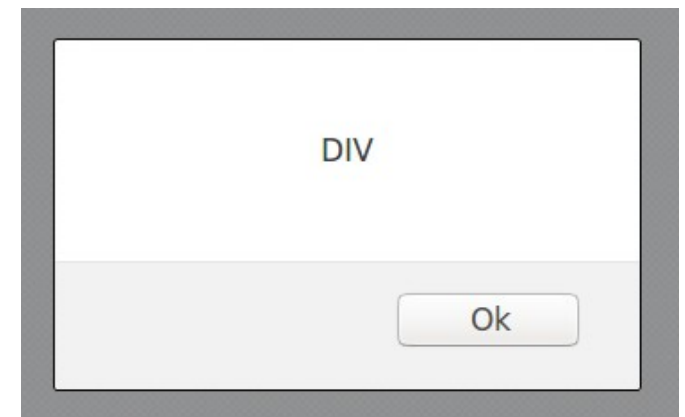
```
</head>
```

```
<body>
```

```
<p class="par"> Continutul paragrafului </p>
```

```
<div class="par"> Continutul paragrafului </div>
```

```
</body>
```



Exemplu:

O funcție `parinte(a, stil)` care pentru un element `a` din DOM, întoarce primul stramos care conține clasa `stil`.

```
function parinte(a, stil) {  
    var b = a.parentElement;  
    while (b != document.documentElement) {  
        if(b.classList.contains(stil))  
            return b;  
        b = b.parentElement;  
    }  
    return null;  
}
```



Nodul corespunzător  
tagului <html>

Exemplu:

O funcție `frati(a)` care pentru un element `a` din DOM, calculează numărul fraților lui care sunt de același tip cu el.

```
function frati(a) {  
    var nr=-1;  
    var b = a.parentElement;  
    var copii=b.children;  
    for(var i=0;i<copii.length;i++)  
    {  
        if(a.nodeName == copii[i].nodeName) nr++;  
    }  
    return nr;  
}
```

# Crearea/inserarea/stergerea elementelor

## Crearea unui element

```
document.createElement("tag")  
document.createTextNode("text")
```

## Inserarea unui element

```
parinte.appendChild(copil)  
parinte.insertBefore(CopilNou, CopilVechi)
```

Daca nodul copil  
exista in arbore  
atunci doar muta  
nodul (nu face  
copie)

## Stergerea / Inlocuirea unui element

```
parinte.removeChild(copil)  
parinte.replaceChild(CopilNou, CopilVechi)
```

exemplu

```
<script>
function creare(tag,text)
{
var elnou=document.createElement(tag);
var textnou=document.createTextNode(text);
elnou.appendChild(textnou);
return elnou;
}
window.onload=function()
{var list=document.getElementById("lista");

var el1=creare("li","JavaScript");
list.appendChild(el1);

var el2=creare("h2","Tehnici Web");
document.body.insertBefore(el2,list);
}
</script>
</head>
<body>
<ul id="lista">
<li id="item1">HTML</li>
<li id="item2">CSS</li>
</ul>
</body>
```

## Tehnici Web

- HTML
- CSS
- JavaScript

```
var s= document.getElementById("item1");
list.removeChild(s);
```

## Tehnici Web

- CSS
- JavaScript

# Modificarea atributelor

Atributele elementelor HTML devin proprietati ale obiectelor corespunzatoare

pot fi accesate prin

- numele direct al proprietatii
- metode specifice
- proprietatea `attributes`

# Modificarea atributelor

➤ **proprietati:** el.id, el.className, el.alt, el.href, el.src

➤ **metode:**

**el.getAttribute("nume-atribut")**

// întoarce un string (valoarea atributului specificat)

**el.setAttribute("nume-atribut", "valoare")**

// adauga un atribut și valoarea lui

**el.hasAttribute("nume-atribut")** // întoarce boolean

**el.removeAttribute("nume-atribut")** //sterge atributul specificat

**Adaugare de proprietati noi:**

**el.proprietateNoua=valoare**

## Exemplul 1

```
<script>
window.onload=function()
{
var x=document.getElementById("imag1");
alert(x.src); //sursa imaginii
x.src="iarna.jpg"; //schimb sursa imaginii
}
</script>
</head>
<body>
<h1>Doua imagini</h1>


</body>
```

## Doua imagini



## Doua imagini



file:///home/carmen/TEHNICI\_WEB\_CURSURI/EXEMPLE/vara.jpg

Ok



## Exemplul 2

```
<style>
.bright{background-color:yellow;}
</style>

<script>
window.onload=function()
{
var link=document.getElementById("pagina");
link.href="https://developer.mozilla.org/bm/docs/Web/JavaScript"; //modific atributul href
link.setAttribute("class", "bright");
alert(link.getAttribute("href"));
}
</script>
</head>
<body>
<a id="pagina" href="https://www.w3schools.com/js/default.asp">JavaScript</a>
</body>
```

JavaScript

https://developer.mozilla.org/bm/docs/Web/JavaScript

Ok

# Modificarea atributelor

- proprietatea `el.attributes` intoarce un obiect array-like cu attributele elementului

```
attrs = element.attributes;  
attrs[i].name  
attrs[i].value  
attrs.length //nr de attribute
```

### Exemplul 3 (în contextul exemplului 2)

```
var attrs = link.attributes;  
var output="";  
for (var i = 0; i< attrs.length ; i++)  
{ output += attrs[i].name + " -> " + attrs[i].value + " // ";  
}  
alert(output);  
}
```



id -> pagina // href -> https://developer.mozilla.org/bm/docs/Web/JavaScript // class -> bright //

Ok

# JavaScript si CSS

Orice obiect asociat unui element HTML (clasa `Element`) are proprietatea `style`, a carei valoare este un obiect din clasa `CSSStyleDeclaration`

Proprietatilor CSS le corespund proprietati ale obiectului `style`.

<code>background-color</code>	→	<code>backgroundColor</code>
<code>color</code>	→	<code>color</code>
<code>text-align</code>	→	<code>textAlign</code>

## Schimbarea stilului unui element HTML

`element.style.proprietate = stil nou`

## Exemplu

```
<script>
function schimbaStil(el)
{
el.style.color = "blue";
el.style.fontFamily = "Arial";
el.style.fontSize = "larger";
}
window.onload=function()
{
schimbaStil(document.getElementById("p2"));
}
</script>
</head>
<body>
<p id="p1">Paragraful 1</p>
<p id="p2">Paragraful 2</p>
<p id="p3">Paragraful 3</p>
</body>
```

Paragraful 1

Paragraful 2

Paragraful 3

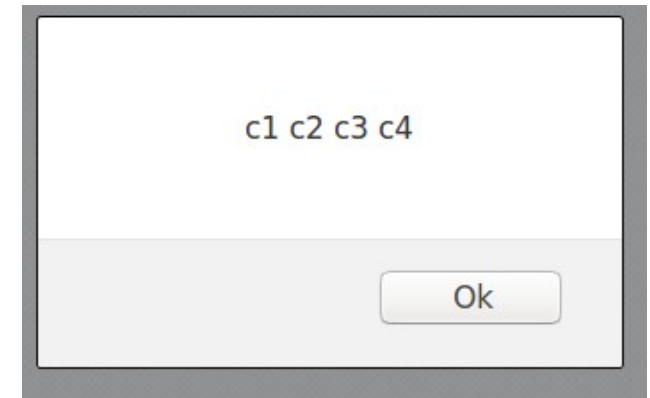
# JavaScript si CSS

Clasele asociate unui element pot fi accesate folosind proprietatea `classList`, care este un obiect din clasa `DOMTokenList`

Sintaxa: `ecl = element.classList`  
(lista claselor elementului)

```
<script>
window.onload = function(){
  var pclas = document.getElementById("par").classList;
  alert(pclas);
}
</script>

</head>
<body>
<p id="par" class="c1 c2 c3 c4"> Continutul paragrafului </p>
</body>
```



`ecl[i]` // read-only

ecl =element.classList

## Metode

ecl.length //nr de clase asociate elementului

ecl.item(i) //numele clasei cu indexul i (i=0,..)

ecl.add("clasa1","clasa2",...) //adauga clasa (clasele)

ecl.remove("clasa1","clasa2",...) //sterge clasa (clasele)

ecl.contains("clasa") // întoarce true sau false

ecl.toggle("clasa", expresie) //sterge clasa dacă  
aceasta exista, altfel o adauga

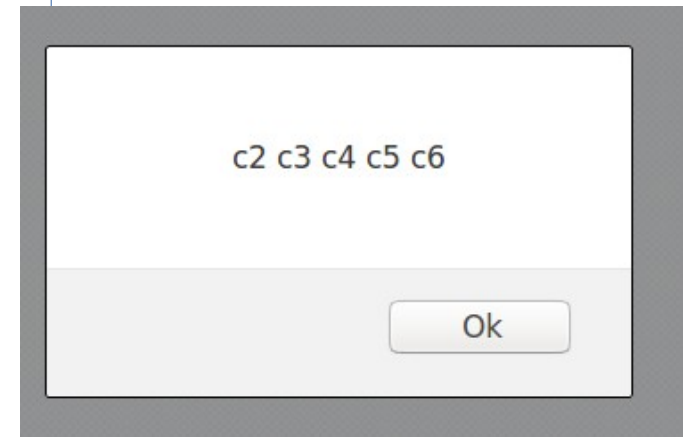
expresie = boolean (dacă este true se adauga clasa,  
dacă este false se sterge clasa)

Exemplu: add, remove, toggle, length

```
<script>
window.onload = function(){
var pclas = document.getElementById("par").classList;
// c1 c2 c3 c4

pclas.add("c5","c6"); // adaug c5 și c6
pclas.add("c3");      // adaug c3
pclas.remove("c1");   // sterge c1
var l=pclas.length;  // nr de clase (5)
pclas.toggle("c4",l==5); // expresia l==5 este true, se
încearcă adaugarea lui c4
alert(pclas); // afiseaza lista claselor
}
</script>

</head>
<body>
<p id="par" class="c1 c2 c3 c4"> Continutul paragrafului </p>
</body>
```





# Elementul HTML **button**

```
<button> continut </button>
```



## Summary

The HTML `<button>` Element represents a clickable button.

Content categories	Flow content, phrasing content, Interactive content, listed, labelable, and submittable form-associated element, palpable content.
Permitted content	Phrasing content.
Tag omission	None, both the starting and ending tag are mandatory.
Permitted parent elements	Any element that accepts phrasing content.
DOM interface	<code>HTMLButtonElement</code>
Element type	Inline

# HTML: atributul onclick

Sintaxa: `<tag atribut="cod JavaScript">`

Exemplu: la click pe button se va afisa un mesaj

```
<button onclick="alert('Hello world!')">Click Me!</button>
```

A rectangular button with rounded corners, a light gray background, and a thin orange border. The text "Click Me!" is centered on the button in a dark gray font.

An alert dialog box with a white background and a gray border. It contains the text "Hello world!" in the center. At the bottom right, there is a button labeled "Ok".

Ok

```
<!DOCTYPE html>
<html lang="ro">
<head>
<script type="text/javascript" >
function suma(x) {
var s = 0;
for (var i=1; i<= x ; i++) s=s+i;
alert(s);
return s;}
</script>
```

```
</head>
<body>
<button id="calcs" onclick="suma( 10) "> Trimate </button>
</body>
</html>
```

atribut HTML

Event Handler

Trimate

45

Ok

# JavaScript: proprietatea **onclick**

Sintaxa: `object.onclick = function(){cod JavaScript};`  
`object.onclick = nume-functie;`

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

window.onload=myMain;
function myMain()
{
document.getElementById("calcs").onclick=function(){ suma(10);}
}
function suma(x) {
var s = 0;
for (var i=1; i< x ; i++) s=s+i;
alert(s);
return s; }

</script>
```

```
<body>
<button id="calcs" > Trimite
</button>
</body>
</html>
```

# Evenimente

Un eveniment nu este un element JavaScript.  
Browserul sesizeaza evenimentul si il anunta programului.  
Unui eveniment ii sunt asociate unele  
elemente JavaScript specifice: name, target, handler



Tipuri de evenimente: form events, window events,  
mouse events, key events, ...

mouse events: onmouseover, onmouseout,...

# Evenimente care schimba stilul unui element html

la hover pe buton, paragraful cu id-ul "schimb" sa apara cu text albastru pe background galben

```
window.onload = myMain;

//selectez target-ul si setez handler-ul
function myMain() {
  document.getElementById('buton').onmouseover = schimbaStil;
}
function schimbaStil() {
  document.getElementById('schimb').style.color = "blue";
  document.getElementById('schimb').style.backgroundColor = "yellow";
}
```

```
<body >
<p id="schimb">
Lorem ipsum
</p>

<button id="buton">
Hover me
</button>
</body>
```

Lorem ipsum

Hover me

Lorem ipsum

Hover me

## Evenimente care schimba stilul unui element html

```
window.onload = myMain;
```

```
function myMain() {
    document.getElementById('buton').onmouseover = stil1;
    document.getElementById('buton').onmouseout = stil2;
}
```

```
function stil1() {  
    document.getElementById('schimb').style.backgroundColor =  
        "yellow";  
}
```

```
function stil2() {  
    document.getElementById('schimb').style.backgroundColor =  
        "white";  
}
```

## Evenimente care schimba stilul unui element html

```
window.onload = myMain;
```

```
function myMain() {  
  document.getElementById('buton').onmouseover = stil1;  
  document.getElementById('buton').onmouseout = stil2;  
}
```

```
function stil1() {  
  document.getElementById('schimb').className = "s1";  
}
```

```
function stil2() {  
  document.getElementById('schimb').className = "s2";  
}
```

```
.s1 {background-color:yellow;}  
.s2{color:red;}
```

stil.css