

# Tehnici Web

## CURSUL 13

Semestrul I, 2019-2020  
Carmen Chirita

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

**HTML** - limbajul de marcare prin care definim structura și conținutul unui document web

**CSS** - limbajul prin care definim stilul (font, culoare, dimensiune, spațiere) și modul de aranjare a elementelor într-un document web; se pot defini și animații folosind CSS

**JavaScript** - limbajul de scripting care permite interacțiunea cu paginile web

# CSS-limbajul de stilizare a paginilor web

## Regula CSS-exemplu

**selector**

h1 {

color: blue;

text-align: center;

}

**proprietate**

**valoare**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="fisier_stil.css">
</head>
<body>
  <h1 style="color:blue; text-align:center;">CSS</h1>
  <p>CSS este un limbaj de stilizare a paginilor web</p>
</body>
</html>
```

in fisier extern

în fișierul HTML

inline

# Selectori CSS

Selector CSS	CSS	HTML
numele tagului	<pre>h1 {   color: red; }</pre>	<code>&lt;h1&gt;La mulți ani! &lt;/h1&gt;</code>
id	<pre>#p20 {   font-weight: bold; }</pre>	<code>&lt;p id="p20"&gt;...&lt;/p&gt;</code>
clasa	<pre>.mare {   font-size: 16pt; }</pre>	<code>&lt;p class="mare"&gt;...&lt;/p&gt;</code>
atribut	<pre>a[target=_blank] {   background-color: yellow; }</pre>	<code>&lt;a href=".." target="_blank"&gt;...</code>
universal	<pre>* {   border: 2px solid black; }</pre>	toate elementele

## Exemple

### HTML

```
.....  
<div class="a b">div</div>  
<p class="a b">p1</p>  
<p class="a">p2</p>  
<p>p3</p>  
<p class="b">p4</p>  
<p>p5</p>  
<p class="a b"> p6</p>  
.....
```

### CSS

```
p.a.b {  
    font-size:3em;  
}  
p.a, p.b {  
    border: 1px solid  
    black;  
}  
p.a:not(.b){  
    color: red;  
}  
p.b:not(.a){  
    color: blue;  
}
```

div

p1

p2

p3

p4

p5

p6

# Selectori combinati

**div p {**  
background-color: yellow;}

► descendentii

**div > p {**  
font-weight:bold;}

► copiii

**div + p {**  
border:1px solid blue;}

► următorul frate adjacent

**div ~ p {**  
color:red;}

► următorii frati

<body>

<div>

<p>Paragraph 1 in the div.</p>

<p>Paragraph 2 in the div.</p>

<section><p>Paragraph 3 in the div.</p></section>

</div>

<p>Paragraph 4. Not in a div.</p>

<p>Paragraph 5. Not in a div.</p>

</body>

**Paragraph 1 in the div.**

**Paragraph 2 in the div.**

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

## Pseudo-clase CSS

```
p:hover, a:hover {  
  background-color: yellow;}  
}
```

➔ mouse-ul este deasupra elementului

```
a:visited {  
  color:green;}  
}
```

➔ linkul a fost vizitat

```
a:link {  
  color:blue;}  
}
```

➔ linkul nu a fost vizitat

```
:not(p){  
  color:blue;}  
}
```

➔ elementele care nu sunt de tipul **p**

```
:first-child / :last-child  
:nth-child(n)  
:nth-of-type(n)
```

➔ elementul specificat care primul/ultimul copil

➔ al n-lea copil al altui element

➔ al n-lea copil de tipul specificat

## Pseudo-elemente CSS

```
:after  
:before  
:first-line  
:first-letter
```

```
p:after{  
  display:inline-block;  
  width:10px;  
  height:10px;  
  content:"";  
  background:blue;  
}
```

# Example

## HTML

```
<body>
<p>Ion are o <b>pisica</b>, un <b>catel</b>, un <b> papagal</b>. </p>
<p>George are o <b>bicicleta</b> si un <b>scuter</b>.</p>
<p>Ana are <b>mere</b> si <b>pere</b>.</p>
</body>
```

p:first-child b {  
color: blue;  
}

Ion are o **pisica**, un **catel**, un **papagal**.  
George are o **bicicleta** si un **scuter**.  
Ana are **mere** si **pere**.

p b:first-child {  
color: blue;  
}

Ion are o **pisica**, un **catel**, un **papagal**.  
George are o **bicicleta** si un **scuter**.  
Ana are **mere** si **pere**.



# Exemple

Cand se vine cu cursorul pe un paragraf, toate elementele de dupa el din pagina (frați) care au clasa „ceva” cat si elementele din subarborii fratilor care au clasa „altceva” sa aiba font-size-ul de 2 ori mai mare decat containerul

```
p:hover ~ .ceva, p:hover ~ * .altceva{  
    font-size:2em;  
}
```

# CSS-tranzitii și animatii

Tranzitie: Daca se ajunge cu mouse-ul pe un li dintr-o sublista, daca li-ul este pe o pozitie para, isi schimba culoarea de background treptat pe parcursul a doua secunde de la transparent la verde (si invers, cand se ia cursorul de pe li)

```
li li:nth-child(even) {  
    transition: background-color 2s;  
}  
  
li li:nth-child(even):hover {  
    background-color:green;  
}
```

Animatie: Ultimul element al divului cu id-ul „parinte” să-și schimbe opacitatea de la 1 la 0.5 in timp de 4s, aceasta repetandu-se la infinit

```
div#parinte > :last-child {  
    animation-name: myanimation;  
    animation-duration: 4s;  
    animation-iteration-count:infinite;  
}
```

```
@keyframes myanimation {  
    from {opacity: 1;}  
    to {opacity: 0.5;}  
}
```

# CSS-media query

Permite definirea unui cod css care se va aplica doar in anumite conditii specificate de query

Exemplu:

La latimea paginii sub 500px, divurile continute în elementul cu id-ul „container” trebuie sa se aseze unele sub altele si sa aiba latimea egala cu jumatate din latimea vieportului (nu a containerului).

De asemenea divul cu id-ul „d3” trebuie sa nu se mai afiseze.

```
@media screen and (max-width:500px){  
  
    #container div{  
        display:block;  
        width:50vw;  
    }  
    div#d3{  
        display:none;  
    }  
}
```

# CSS-layout

Proprietatea **position**:

**static (implicit)**

**relative**

**absolute**

**fixed**

**sticky**

**left**  
**right**  
**top**  
**bottom**

Proprietatea **display**:

**none**

**inline**

**block**

**inline-block**

Proprietatea **visibility**:

**hidden**

**visible**

**CSS-flex**

**flex container:**

**display:flex /inline-flex;**

**flex-direction:**

**row/row-reverse/column/ column-reverse**

**flex-wrap: nowrap | wrap | wrap-reverse**

**CSS-grid**

**grid container:**

**display:grid /inline-grid;**

**grid-template-columns**  
**grid-template-rows**

**grid-column-gap**  
**grid-row-gap**

**grid items:**

**grid-colum**

**grid-row**

## Exercitiu

**Scrieți cod CSS astfel încât divurile din documentul HTML să fie aranjate ca în figura de mai jos**

1	2	3
		4
		5

## HTML

```
<div id="container">  
    <div id="d1">1</div>  
    <div id="d2">2</div>  
    <div id="d3">3</div>  
    <div id="d4">4</div>  
    <div id="d5">5</div>  
</div>
```

## CSS -grid

```
#container{  
    display:grid;  
    grid-template-columns: auto auto auto;  
}  
  
#d1, #d2{  
    grid-row:1 / 4;  
}
```

# JavaScript-tipuri de date

Toate datele de tipuri primitive (string, number, boolean) sunt transmise prin valoare.

Datele de tip object (Object, Array, Date, Math, String,..) sunt transmise prin referinta.

```
var s1 = "1"; // string
var s2 = s1;
// s2 copiaza val lui s1
s2= "2"
// se modifica doar s2
alert(s1) // "1"
```

```
var o1 = {prop: "1"} // object  
var o2 = o1;  
// o1 si o2 refera aceeaasi zona  
o2.prop = "2";  
// se modifica si o1 si o2  
alert(o1.prop); // "2"
```

```
var v1=[1,2,3];  
var v2=v1;  
//vor referi aceeaasi zona  
v1[0]=4;  
alert(v2[0]);// afiseaza 4
```

# Array

```
var v = [];  
v[0] = "a";  
var v = [6,4,7,3]  
  
v.length  
  
v.push(10); // => v=[6,4,7,3,10]  
v.pop();    // => [6,4,7,3]  
  
v.shift();  // => [ 4,7,3]  
v.unshift(10); // => [10,4,7,3]  
  
v.sort();   // => [3,4,6,7]
```

```
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"
```



# Tipul string

```
var s="hello"
```

Metode: `charAt`, `indexOf`, `lastIndexOf`, `replace`, `split`,  
`toLowerCase`, `toUpperCase`,

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

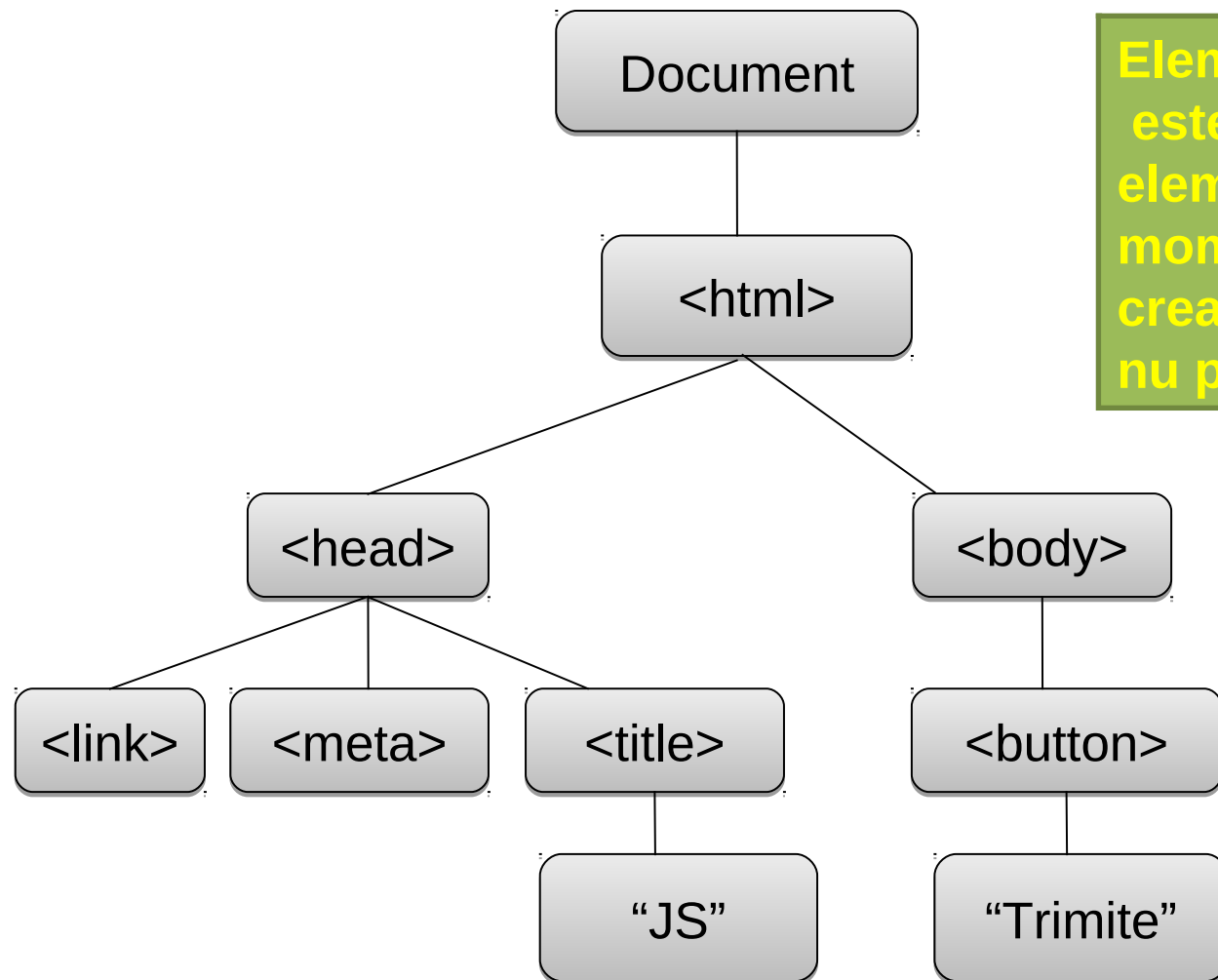
Caractere speciale: `'\'` `"` `\n` `\t` `\&` `\\`

Accesarea unui caracter: `s[0]`, `s.charAt(0)`, `s.charAt(s.length-1)`

```
var s="hello"  
s[0] = 'v'; //s[0] nu se modifica  
alert(s[0]);
```

Un string nu este un array de caractere

Orice tab al unui browser contine un obiect window (din clasa Window)  
Proprietatea document a obiectului window e obiectul document al paginii web  
(apartine clasei Document)  
Orice pagina web e reprezentata in DOM ca un arbore de obiecte;



**Elementul <script> din <head> este procesat inaintea elementului <body>; in acel moment arborele DOM nu este creat si elementele lui nu pot fi accesate.**

Pentru a putea accesa proprietatea document a obiectului window trebuie ca pagina sa fie incarcata:

```
window.onload=function()  
  {var el=document.getElementById("i1");}
```

```
window.onload=myMain;  
function myMain()  
{var el=document.getElementById("i1");}
```

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

# Selectarea elementelor in DOM (I)

`document.getElementById( id)`

`document.querySelector(selectorCss) //primul`

*-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);
```

Exercitiu: inlocuiti toate elementele de tip <p> din document cu elemente de tip <div> si invers.

Solutie:

```
var pars=document.querySelectorAll('p');
var divs=document.querySelectorAll('div');

for(var i=0;i<pars.length;i++) {
    var div=document.createElement("div");
    div.innerHTML=pars[i].innerHTML;
    document.body.insertBefore(div,pars[i].nextElementSibling);
    document.body.removeChild(pars[i]);
}

for(var i=0;i<divs.length;i++) {
    var par=document.createElement("p");
    par.innerHTML=divs[i].innerHTML;
    document.body.insertBefore(par,divs[i].nextElementSibling);
    document.body.removeChild(divs[i]);
}
```

## Selectarea elementelor in element

`element.getElementsByTagName(umeClasa)`

`element.getElementsByTagName(umeTag)`

`element.querySelector(selectorCss)`

`element.querySelectorAll(selectorCss)`

Exemplu

```
var list=document.getElementById("lista1");  
var elem=list.getElementsByClassName("click");
```



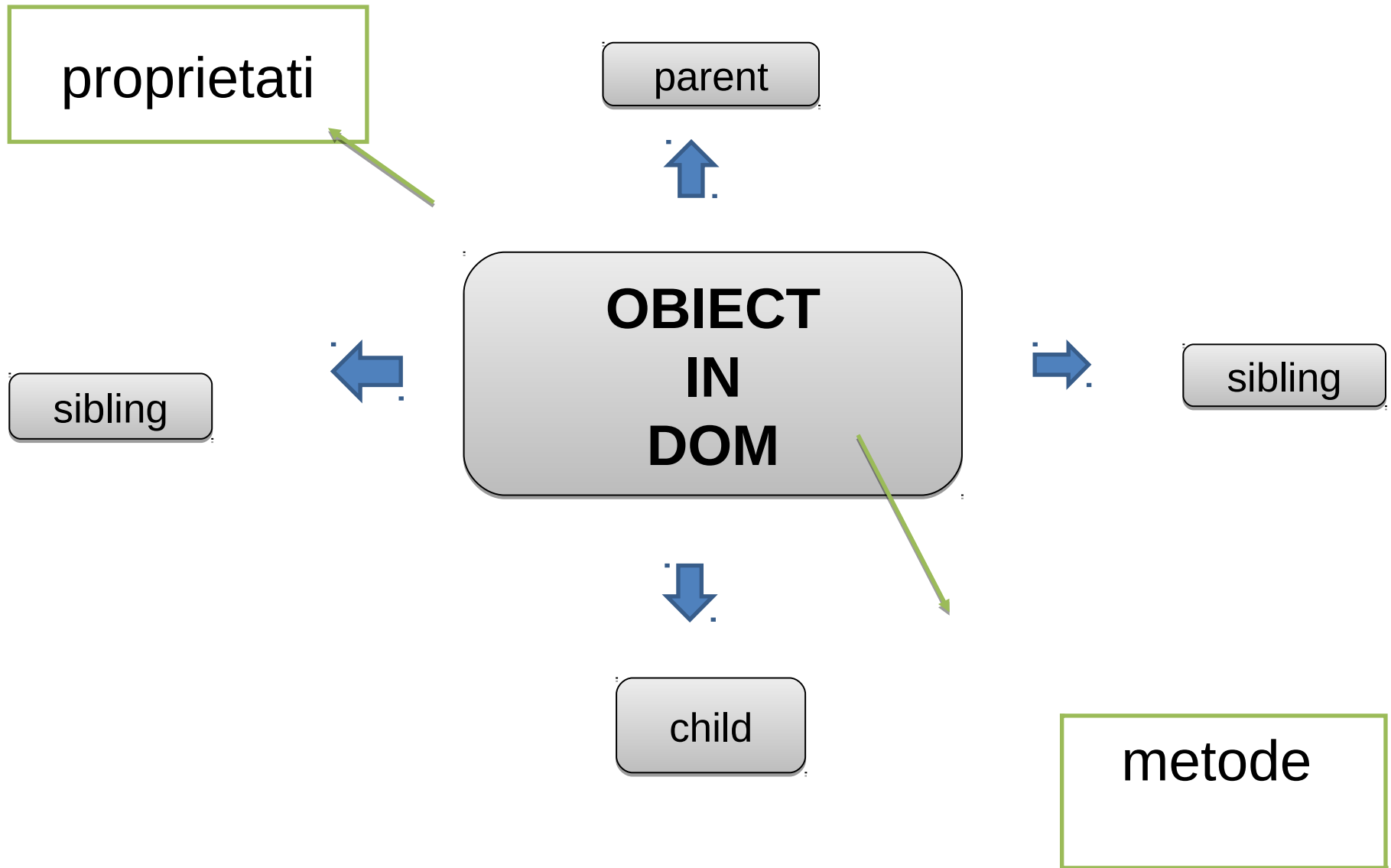
## Proprietati pentru obiecte de tip Node

nodeValue // pentru noduri Text, Comment

nodeName, tagName // numele tagului

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

Node poate fi: document, element, text, atribut



## Selectarea elementelor in DOM (II)

Node.parentNode / Node.parentElement

Node.childNodes(live) / Node.children(live)

Node.firstChild / Node.firstElementChild

Node.lastChild / Node.lastElementChild

Node.nextSibling / Node.nextElementSibling

Node.previousSibling / Node.previousElementSibling

Exemplu: afisati toate nodurile de tip element din document (numele tagurilor) si numarul lor

```
var elem=document.getElementsByTagName("*");  
alert(elem.length);  
for(var i=0; i<elem.length;i++)  
alert(elem[i].nodeName);
```

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 unui nod

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

## ➤ Inserarea unui nod

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

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

## ➤ Stergerea / Inlocuirea unui nod

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

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

innerHTML si textContent

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

innerHTML

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

textContent

Un text simplu si colorat.

Exemplu: adaugarea de elemente noi unui părinte folosind innerHTML va recrea arborele descendentilor în alte variabile

```
<script>
window.onload=function()
{
var sectiune= document.getElementById("container");
var p1=document.getElementById("p1");

p1.onclick = function(){ //click pe primul copil al sectiunii
    alert(p1.id);}

sectiune.innerHTML+="
```



# Modificarea atributelor

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

➤ **metode:**

el.getAttribute()

el.setAttribute("class", "numeclassa")

el.hasAttribute()

el.removeAttribute()

**Adaugare de proprietati noi:**

el.proprietateNoua=valoare

Exemplu:

```
var i=document.getElementById("i1");
```

```
/*i refera aceeaasi zona ca obiectul corespunzator  
tagului */
```

```
alert(i.src);
```

```
// proprietatea src corespunde atributului src
```

```
i.src="s2.jpg";
```

```
/* modificarea proprietatii lui i modifica  
proprietatea obiectului corespondent tagului  
<img> ceea ce modifica atributul src */
```

# JavaScript si CSS

Pentru a determina stilul efectiv aplicat unui element folosim metoda

`window.getComputedStyle(element, ":first-letter")`



este obiect din clasa `CSSStyleDeclaration`  
este **read-only**



pseudo-clasa sau null

```
var oStil = window.getComputedStyle(ob,null) ;  
var x=oStil.color; // proprietatea css ob.style.color
```

Se poate schimba clasa unui element:

```
element.className="clasanoua"
```

```
element.classList.add(clasa1,clasa2)
```

```
element.classList.remove(clasa1,clasa2)
```

```
element.classList.contains(clasa)
```

# Evenimente

## Atribut eveniment

```
<p onclick="codJavascript">
```

## Proprietate eveniment

```
element.onclick= numeFuncție
```

```
element.onclick=function(){}
```

Gresit : element.onclick=**numeFuncție()**;

# Obiectele de tip Event

evenimentul= un obiect din clasa Event

-se poate referi in HTML : event

<p onclick="f(event)">                      sau

-e parametrul implicit al functiei apelate de eveniment

element.onclick=function(e){}

## Proprietati

event.target, event.currentTarget, event.type

## Metode

event.preventDefault()

event.stopPropagation()

event.stopImmediatePropagation()

## Obiectele MouseEvent au proprietati speciale

event.button // 0(stanga)1(mijloc) 2(dreapta)  
event.clientX, event.clientY// pozitia in fereastra  
event.pageX, event.pageY//pozitia in document  
event.screenX, event.screenY//pozitia in ecran

## Obiectele KeyboardEvent au proprietati speciale

event.key //numele tastei  
event.keyCode //deprecated  
event.which //deprecated

# Event listeners

un eveniment poate avea atasate mai multe functii handler sau i se pot sterge unele dintre functiile handler

```
el.addEventListener("click", handleClick, false)
```

nume eveniment

functie

faza de  
executie

```
el.removeEventListener("click",handleClick,true)
```



# Captarea evenimentelor: obiectul event este transmis ca primul argument al handler-ului

```
elem.onclick = myfct;  
  
function myfct (ev) {alert(ev.type);} }  
  
_____
```

```
elem.addEventListener("click", myfct);  
  
function myfct (ev) {alert(ev.type);} }  
  
_____
```

```
elem.onclick = function (ev) {  
  
    alert(ev.type);} }  
  
_____
```

```
elem.addEventListener("click",  
    function (ev) {alert(ev.type);} )  
  
}  
  
_____
```

Setare faza\_exec pentru PARINTE

-false– Bubbling (implicit)

-true - Capturing

Modele de executie:

Parinte true (capturing):

intai handler **parinte** apoi handler **copil**

Parinte false(bubbling) :

intai handler **copil** apoi handler **parinte**

La aparitia unui eveniment se executa:

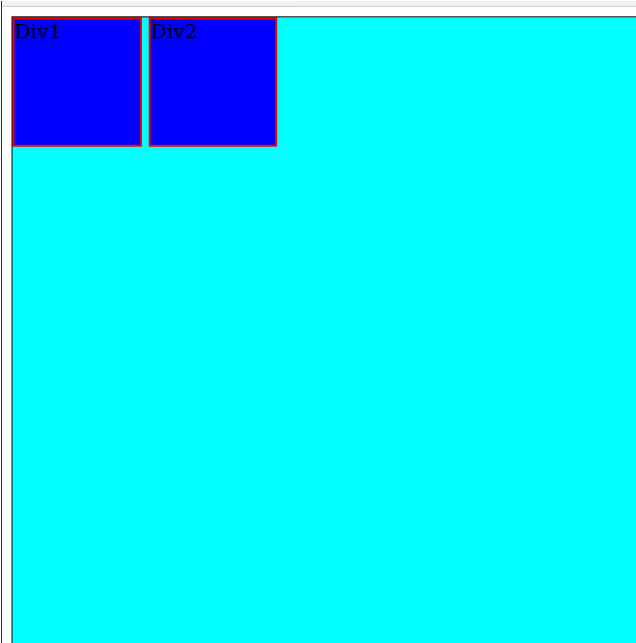
1. handlerele setate pe CAPTURE (true) ale stramosilor tinte

2. handlerul tinte

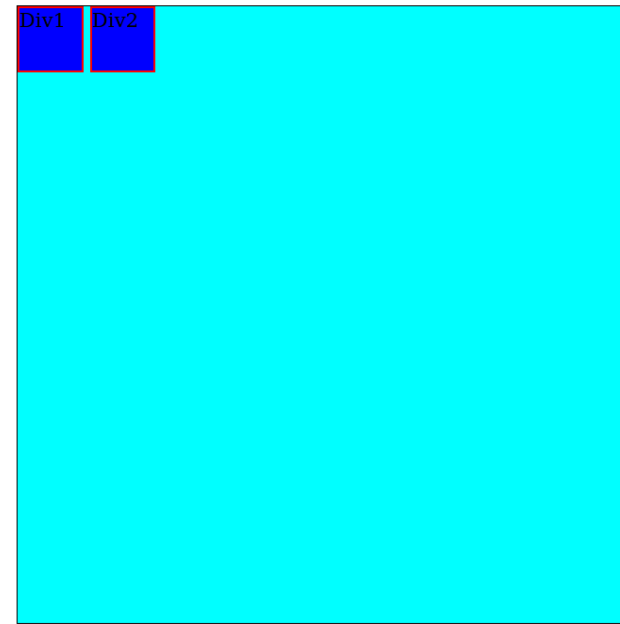
3. handlerele setate pe BUBBLE (false) ale stramosilor tinte

## Exemplu: addEventListener

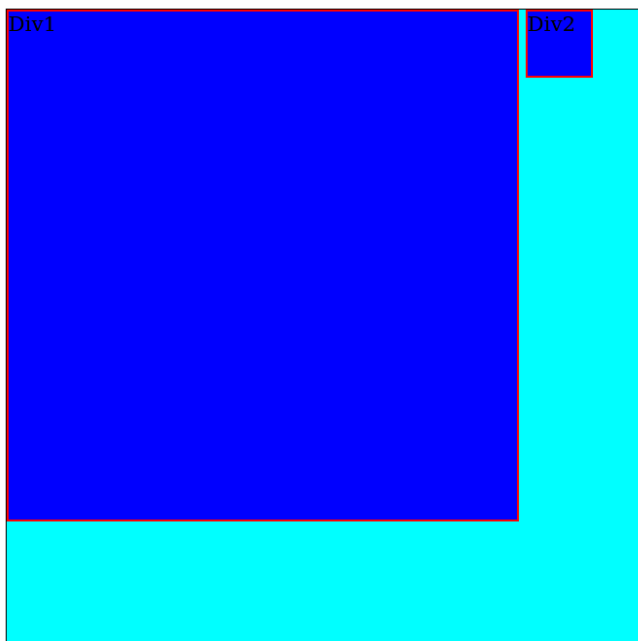
Initial



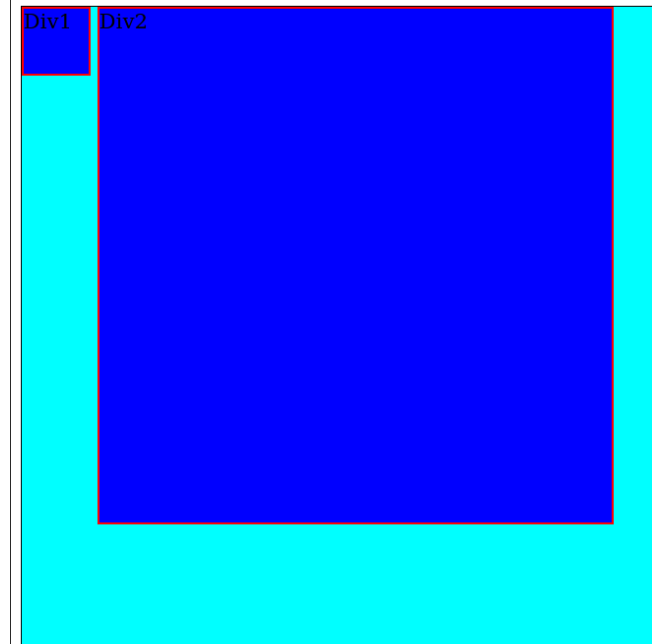
Click pe container



Click pe div1



Click pe div2



## JavaScript

```
var parinte = document.getElementById("container");
var copii=document.querySelectorAll(".copil");
for(var i=0;i<copii.length;i++)
copii[i].classList.add("initial");

parinte.addEventListener("click",function(){
for(var i=0;i<copii.length;i++) {
copii[i].classList.remove("initial","mare");
copii[i].classList.add("mica");
}},true);

for(let i=0;i<copii.length;i++)
{
copii[i].addEventListener("click",function(){
    copii[i].classList.remove("initial");
    copii[i].classList.add("mare");});
}}
```

addEvent.html

## HTML

```
<body>
<div id="container">
  <div class="copil">Div1</div>
  <div class="copil">Div2</div>
</div>
</body>
```

## CSS

```
#container{border: 1px solid black;
width: 500px;height:500px;
background-color:cyan;}

.copil{ border: 2px solid red;
display:inline-block;
background-color:blue;}

.initial{ width: 100px;
height: 100px;}

.mica{ width: 50px;
height: 50px;}

.mare{width: 400px;
height: 400px;}
```

## Proprietati obiect event:

event.target: tinta **initiala** a evenimentului

event.currentTarget : tinta **curenta** a evenimentului (la capturare sau propagare)

event.type : numele evenimentului

# Metode obiect event:

`event.stopPropagation()`

opreste propagarea evenimentului in DOM;

`event.stopImmediatePropagation()`

daca mai multe functii listener sunt atasate aceluiasi element iar una contine `event.stopImmediatePropagation()` functiile listener urmatoare nu mai sunt apelate

`event.preventDefault()`

se anuleaza actiunea implicita a elementului

## Exemplu:

Un document HTML conține un container cu imagini;  
La click pe imagine să se afișeze sursa imaginii iar la click pe container să se afișeze numărul de imagini din container

```
var imagini=document.getElementsByTagName("img");
var container= document.getElementById("container");

for(var i=0;i<imagini.length;i++)
    imagini[i].onclick=function(event){
        event.stopPropagation(); //opreste propagarea spre parinte
        alert(event.target.src);
    }

container.onclick=function(){
    alert( container.getElementsByTagName("img").length);
}
```



## Window metode

```
vt=setTimeout(umeFunctie, intarziere, param);  
vt=setTimeout(function(){}, intarziere, param);  
clearTimeout(vt) ;- anulare functie lansata
```

```
vt=setInterval(umeFunctie, interval, parametrii);  
vt=setInterval(function(){}, interval, parametrii);  
clearInterval(vt) ;- anulare functie lansata
```

vt –globala pentru a fi vazuta de clearTimeout (clearInterval)  
parametrii sunt ai functiei care se va executa (umeFunctie sau anonima)

Exemplu:

La fiecare 3 secunde se schimba culoarea de background a unui element intre doua culori; la click pe element se intrerupe executia.

```
function schimba(elem, culoare1, culoare2){  
    if(elem.style.backgroundColor==culoare1)  
        elem.style.backgroundColor=culoare2;  
    else  
        elem.style.backgroundColor = culoare1;}  
  
var x=document.getElementById("p1");  
var t=setInterval(schimba,x,3000,"red","blue");  
x.onclick=function(){clearInterval(t);}
```

Exemplu:

La fiecare 3 secunde cate un buton din document isi va schimba culoarea de background intr-o culoare random. Colorarea se face circular, dupa ultimul buton se reia colorarea de la primul.

```
var buttons = document.getElementsByTagName("BUTTON");  
  
var bldx = 0;  
  
setInterval(function() {  
  
    buttons[bldx].style.backgroundColor = "rgb(" + Math.floor(Math.random() *  
256)  
  
        + "," + Math.floor(Math.random() * 256)  
  
        + "," + Math.floor(Math.random() * 256)  
  
        + ")";  
  
    bldx = (bldx + 1) % buttons.length;  
  
}, 3000);
```

Let


```
var i=0;  
el.onclick=function()  
    {alert(i); /* evaluate la click -va afisa 1*/ }  
i=1;
```

```
var i=0;  
{let il=i;// il se creaza acum  
el.onclick=function(){alert(il); /*va afisa 0*/ }  
}// se elibereaza zona il=0  
i=1;
```

## Exemplu cu let:

La click pe fiecare imagine din document sa se afiseze sursa imaginii

```
var imagini=document.getElementsByTagName("img");  
  
for(var i=0;i<imagini.length;i++)  
imagini[i].onclick=function(){alert(imagini[i].src);}  
    // nu functioneaza;  
    // i va fi egal cu imagini.length
```



Soluție:

```
for(let i=0;i<imagini.length;i++)  
imagini[i].onclick=function(){alert(imagini[i].src);}
```

## window.localStorage

- date pastrate in browserul client  
(perechi: (proprietate, valoareString))
- nu se sterg la inchiderea browserului
- toate paginile din acelasi loc(domeniu) vor pastra si accesa aceleasi date

localStorage.propNoua=valoare

localStorage.setItem("propNoua", "valoare")

localStorage.getItem("propNoua")

localStorage.removeItem("propNoua")

localStorage.clear();//elibereaza localStorage

<input>

type="text"    **value**="valoarea curentă a textului introdus  
în câmpul de text"

type="button" **value**="eticheta a butonului"

type="radio"    **checked**="true/false"

**value**="non-vizibila"

**name**=" același pt întreg grupul"

<select>

**selectedIndex**=indexul opțiunii selectate

**value**= câmpul "value" al opțiunii selectate

**options**= vectorul de opțiuni