

Tehnici Web CURSUL 12 Recapitulare

Semestrul II, 2022-2023
Carmen Chirita

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

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

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;  
}
```

Exemple

Sa se puna culoarea de background mov pe elementele de tip b care sunt ultimul fiu al unui paragraf si au un element de tip i imediat inaintea lor.

```
p > i + b:last-child {background-color: purple;}
```

La inceputul fiecarui prim copil al unui element se va adauga mesajul "Eu sunt primul!!" scris italic si cu culoare roşie.

```
:first-child::before {content:'Eu sunt primul!!';  
                        color:red;  
                        font-style:italic;  
                        }
```

Exemple

La intrarea cu mouse-ul pe un element li din ul, elementele li de deasupra, impreuna cu el sa aiba background rosu iar toate elementele li de sub el sa aiba background albastru.

```
ul:hover li {background-color:red;}
```

```
ul li:hover {background-color:red;}
```

```
ul li:hover ~li {background-color:blue;}
```


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” sa nu mai aibă border.

```
@media screen and (max-width:500px){  
  
    #container div{  
        display:block;  
        width:50vw;  
    }  
    div#d3{  
        border: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-template-areas

grid-column-gap

grid-row-gap

grid items:

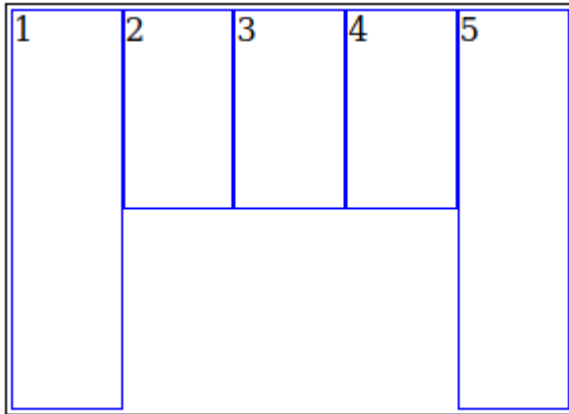
grid-row

grid-column

grid-area

Exercitiu

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



HTML

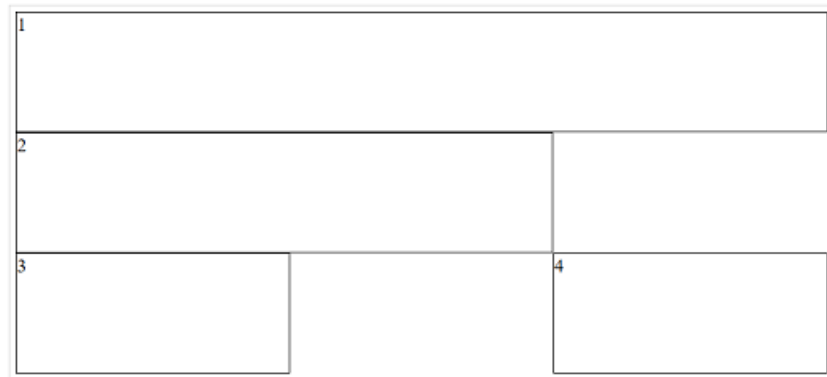
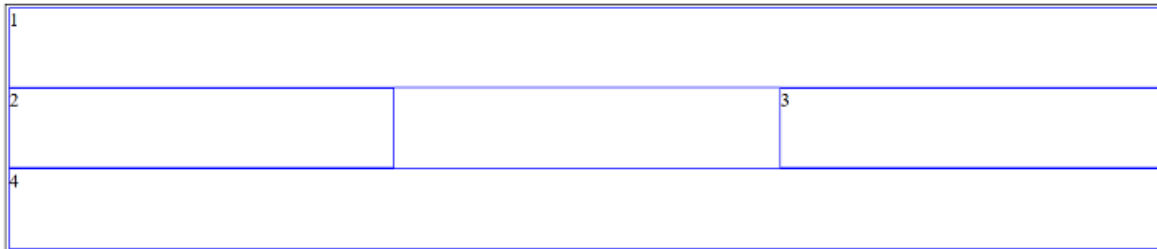
```
<div id="container">  
    <div id="d1">1</div>  
    <div id="d2">2</div>  
    <div id="d3">3</div>  
    <div id="d4">4</div>  
    <div id="d5">5</div>  
</div>
```

CSS - grid

```
#container{  
    display:grid;  
    grid-template-columns: auto auto auto auto auto;  
}  
  
#d1, #d5{  
    grid-row:1 / 3;  
}  
  
#d2, #d3, #d4{  
    grid-row:1 / 2;  
}
```

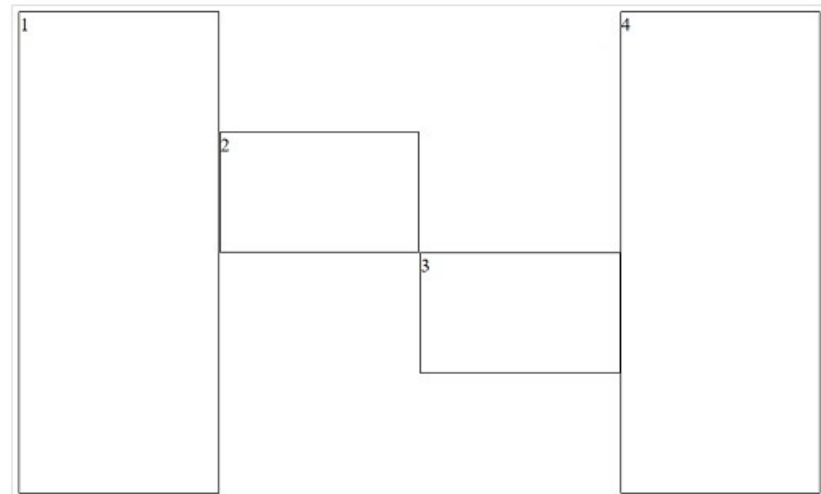
Exercitii

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



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>
```



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 aceeași zona  
o2.prop = "2";  
// se modifica si o1 si o2  
alert(o1.prop); // "2"
```

```
var v1=[1,2,3];  
var v2=v1;  
//vor referi aceeași 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 = "astazi este soare";  
  
var a = s.split(" ");  
    // a = ["astazi","este","soare"]  
  
a.reverse();  
    // a = ["soare","este","astazi"]  
  
var s= a.join('/');  
    // s="soare/este/astazi"
```


Tipul string

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

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

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

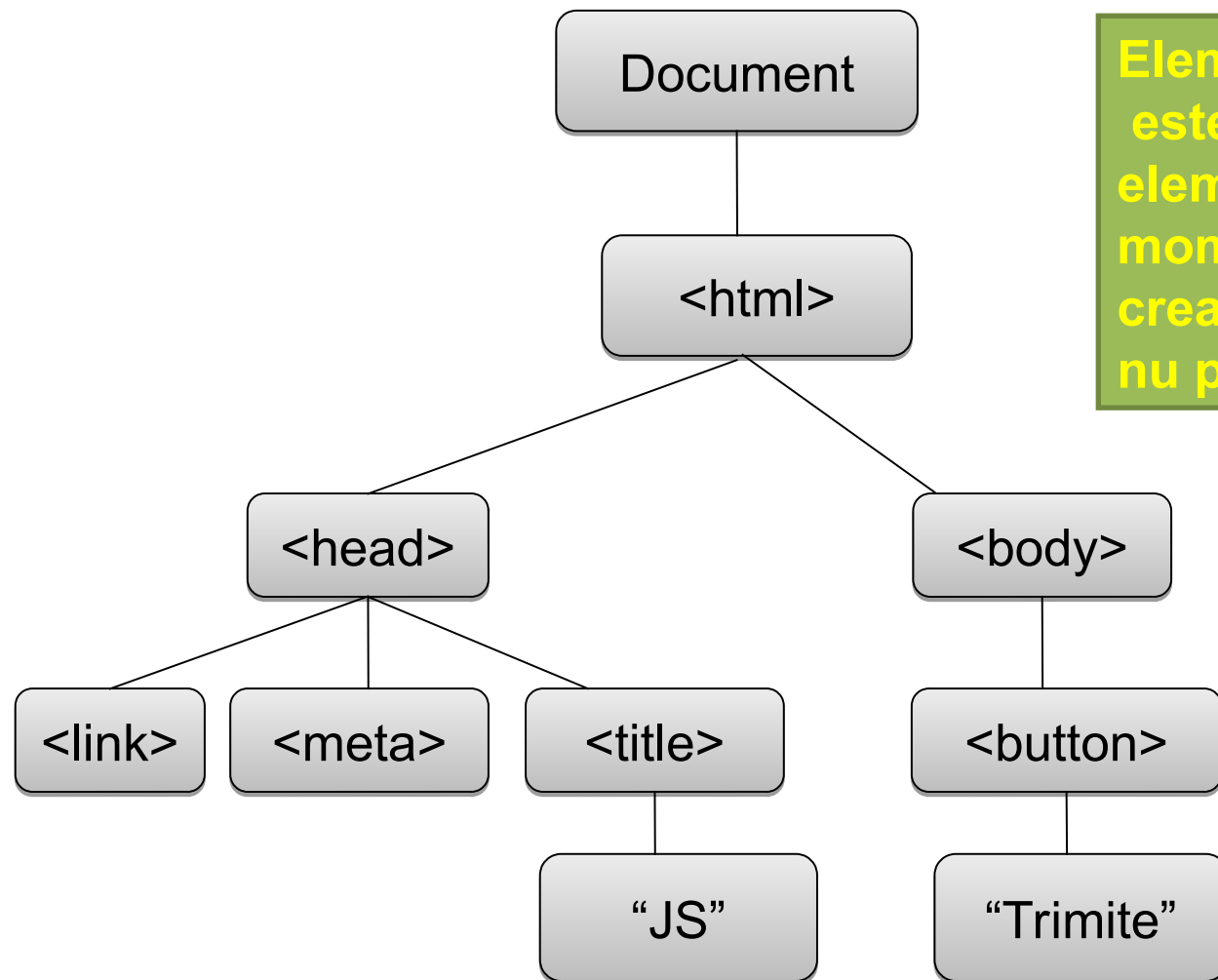
Caractere speciale: `\' \' \' \' \n \t \& \\\`

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

Solutie:

```
var colectie = document.querySelectorAll("section:nth-of-type(2) p");  
for(var i=0; i<colectie.length;i++) {alert(colectie[i].innerHTML);}  
// for(p of colectie) { alert(p.innerHTML);}
```

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");
```

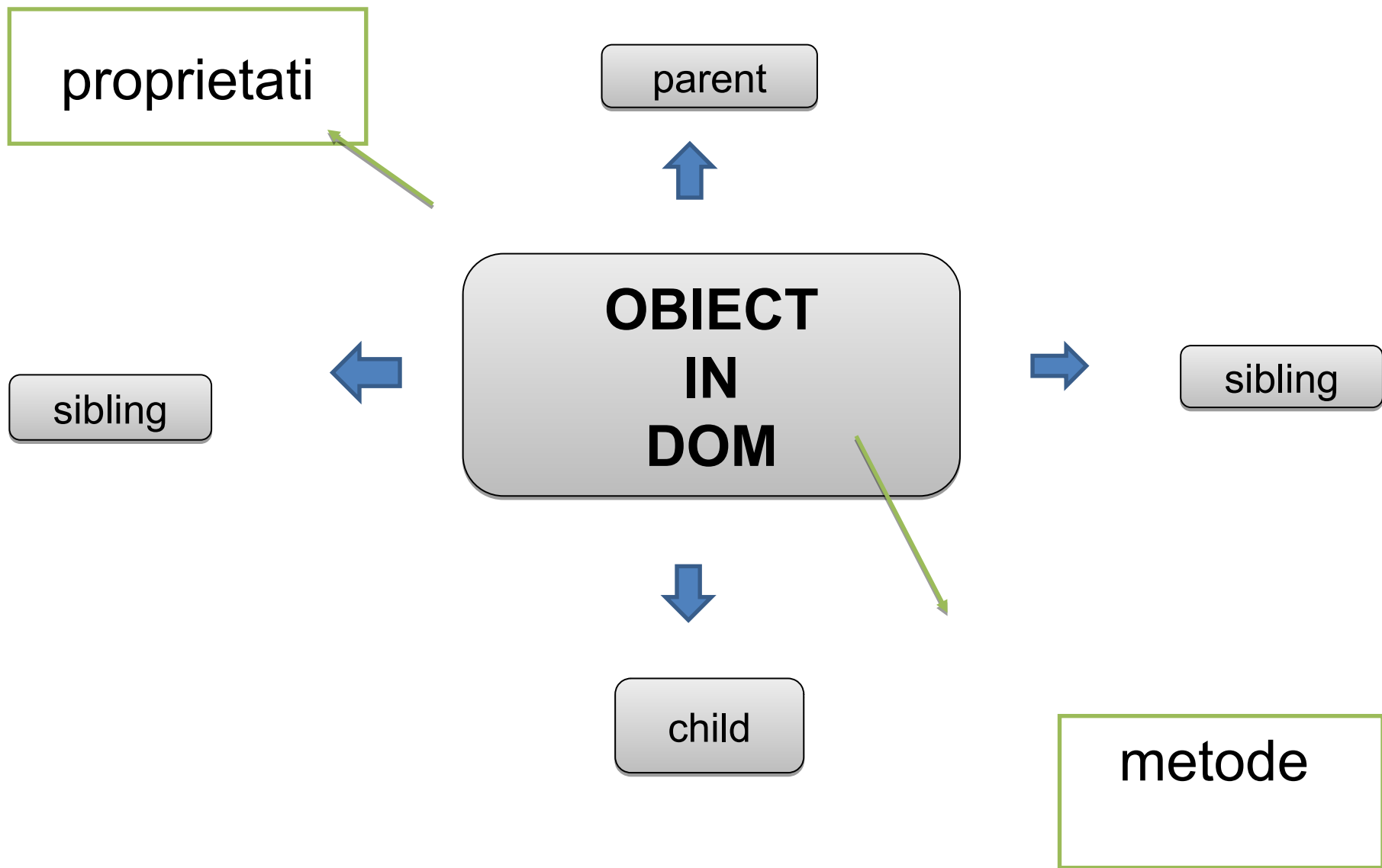

Proprietati pentru obiecte de tip Node

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)
```

Eventimente

Atribut eveniment

<p onclick="codJavascript">

Proprietate eveniment

element.onclick= numeFunctie

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

```

Gresit : `element.onclick=`**`numeFunctie()`**;

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:

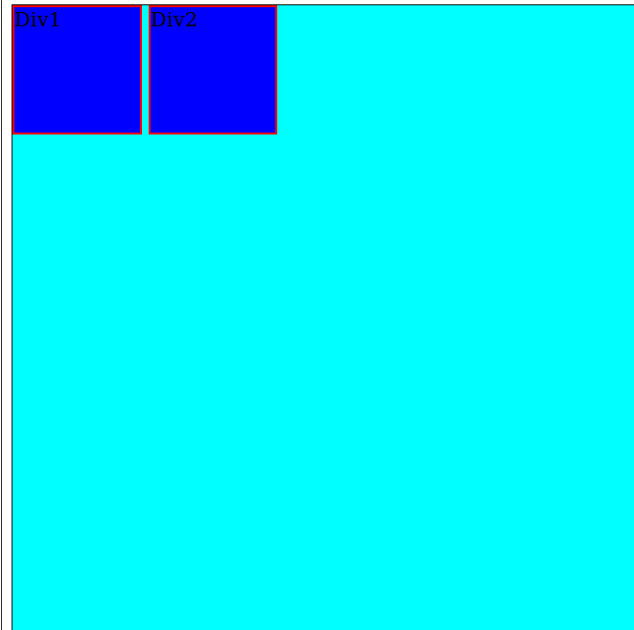
1. handlerele setate pe CAPTURE (true) ale stramosilor tinte

2. handlerul tinte

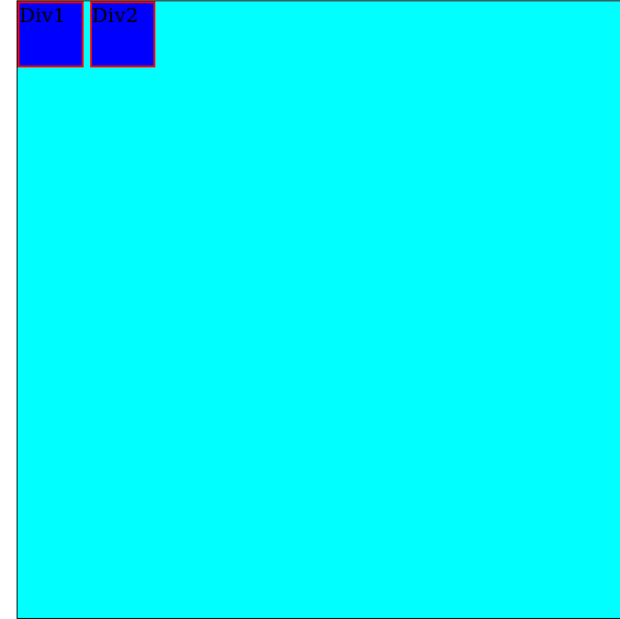
3. handlerele setate pe BUBBLE (false) ale stramosilor tinte

Exemplu: addEventListener

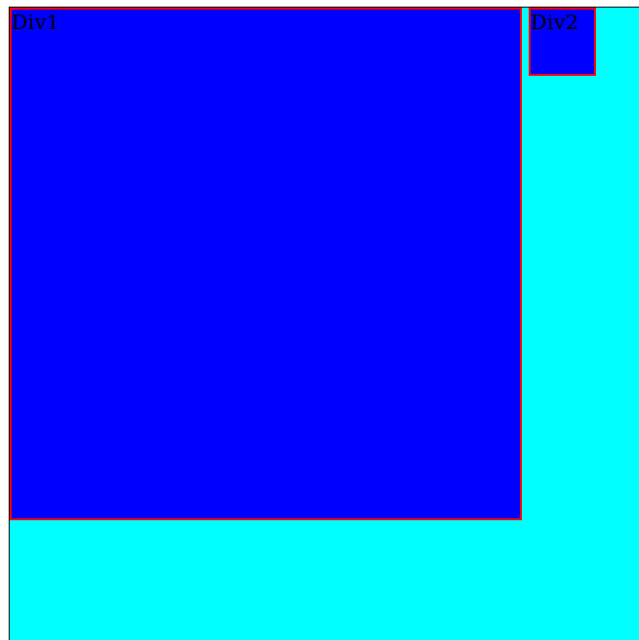
Initial



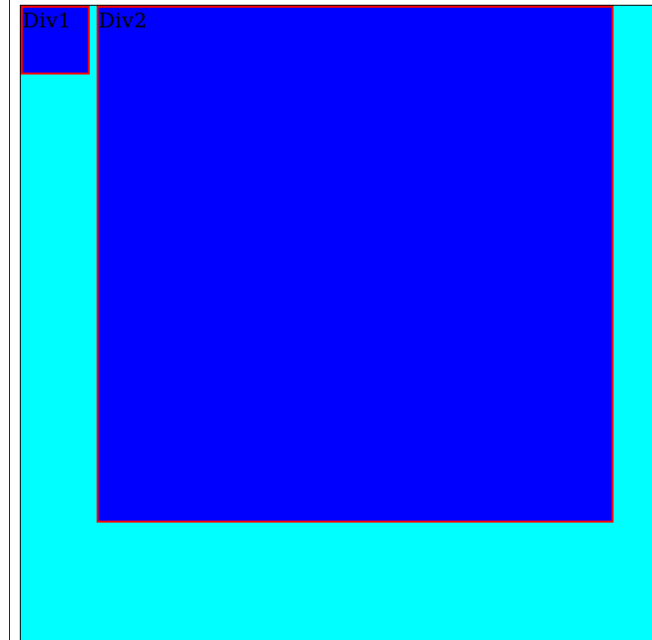
Click pe container



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 aceluasi 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 ancora din document sa se afiseze url-ul ancorei

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



Soluție:

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

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

Subiecte date la examen

S1. Sa se implementeze urmatoarea functionalitate:
la click pe orice paragraf se construiesc o lista cu continutul elementelor ****
care sunt copii ai paragrafului, lista fiind adaugata in document imediat dupa paragraf;
urmatoarele clickuri pe paragraf nu vor avea nici un efect.

Ex: daca p este

`<p id="pgf">Ion are o pisica, un catel, un papagal. </p>`
atunci lista va contine: **pisica, catel, papagal.**

Ion are o **pisica**, un **catel**, un **papagal**.

George are o **bicicleta** si un **scuter**.

Ana are **mere**, cateva **pere**, si multe **nuci**.

Vine **vacanta**, de Craciun voi primi **cadouri** si voi rezolva **exercitii** la **Tehnici Web** pentru **examen**.

A venit vacanta!

Ion are o **pisica**, un **catel**, un **papagal**.

- pisica
- catel
- papagal

George are o **bicicleta** si un **scuter**.

Ana are **mere**, cateva **pere**, si multe **nuci**.

- mere
- pere
- nuci

Vine **vacanta**, de Craciun voi primi **cadouri** si voi rezolva **exercitii** la **Tehnici Web** pentru **examen**.

- vacanta
- cadouri
- exercitii
- Tehnici Web
- examen

A venit vacanta!

liste.html

```
<script>
function copiiB(elem){ \creeaza lista cu continutul copiilor de tip b ai elementului elem
    var copii=elem.getElementsByTagName("b");
    var ul=document.createElement("ul");
    for(var i=0;i<copii.length;i++){
        var li=document.createElement("li");
        li.innerHTML=copii[i].innerHTML;
        ul.appendChild(li);
    }
    var frate=elem.nextElementSibling;
    document.body.insertBefore(ul,frate); \inserare lista imediat după elem
}

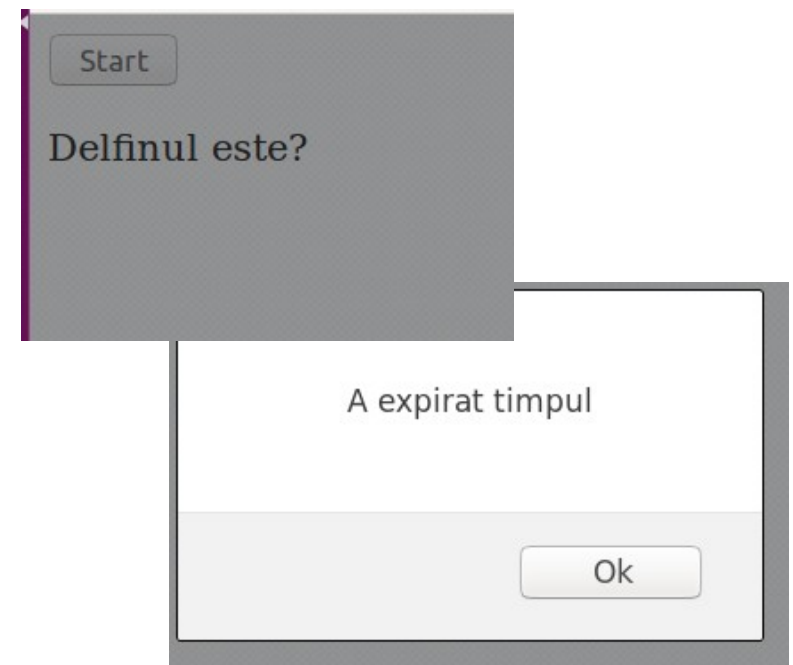
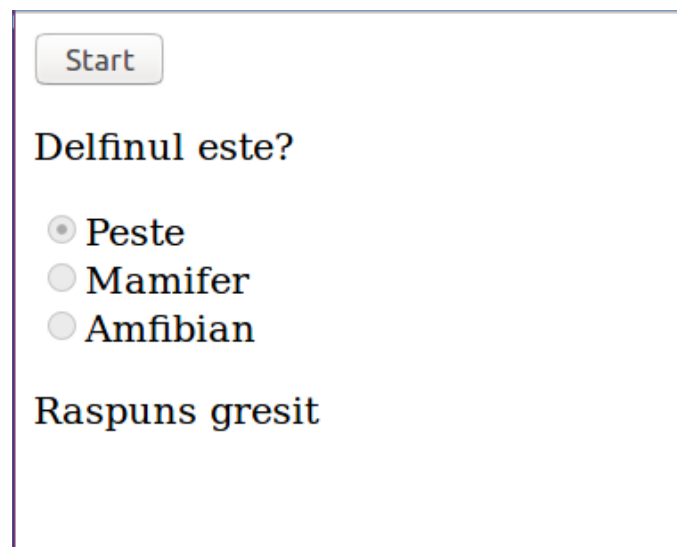
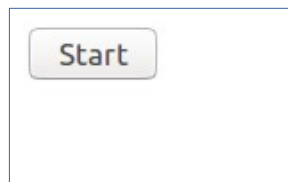
window.onload=function(){
    var colectie=document.getElementsByTagName("p"); \selectarea paragrafelor
    for(var i=0;i<colectie.length;i++){
        let k=i;
        colectie[i].nr=0; \proprietate noua care reține nr. clickului
        colectie[i].onclick = function(){ \clickul pe paragrafe cu apelarea functiei copiiB
            if(colectie[k].nr==0){
                copiiB(colectie[k]);
                colectie[k].nr=1;
            }
        }
    }
}
</script>
```

S2. Un fișier HTML conține: un `<button>` cu id-ul "start", un paragraf cu id-ul "răspuns", un paragraf cu id-ul "intrebare" care conține o întrebare, 3 butoane radio reprezentând variantele de răspuns, cu valorile "corect" sau "gresit".
Inițial întrebarea și butoanele radio nu sunt vizibile.

La click pe butonul "start" întrebarea și butoanele radio devin vizibile și utilizatorul are la dispoziție 5s pentru a selecta un răspuns.

Dacă utilizatorul a selectat un răspuns, în paragraful "răspuns" se afișează textul "Răspuns corect" sau "Răspuns gresit" în funcție de valoarea răspunsului.

După 5s, dacă nu s-a răspuns la întrebare, butoanele dispar și se afișează într-o fereastră alert "Timpul a expirat".



```

<script>
window.onload=function(){
    var intrebare=document.getElementById("intrebare");
    var forma=document.getElementById("form");
    var butoane=document.getElementsByTagName("input");

    document.getElementById("start").onclick=function(){ //click pe start
        intrebare.style.visibility="visible";
        forma.style.visibility="visible";

        var t=setTimeout(function(){
            document.body.removeChild(forma);
            alert("A expirat timpul");},5000);

        forma.onchange=function(){ //daca s-a selectat un buton
            for(var i=0;i<butoane.length;i++){
                butoane[i].disabled=true; //butoanele sunt dezactivate după
                selecția unui buton
                if(butoane[i].checked)
                    document.getElementById("raspuns").innerHTML=
                    "Raspuns"+ ' '+ butoane[i].value;
                    \generarea raspunsului
            }
            clearTimeout(t);
        }
    }
}

```

quizz.html

```

<body>
<button id="start">Start</button>
<p id="intrebare">Delfinul este?</p>
<form id="form">
<input type="radio" name="q1" value="gresit">Peste<br>
<input type="radio" name="q1" value="corect">Mamifer <br>
<input type="radio" name="q1" value="gresit">Amfibian<br>
</form>
<p id="raspuns"></p>
</body>

```

S3. Avem o pagina cu urmatorul body:

```
<ul id="imagini">  
  <li>img1.jpg</li>  
  .....  
  <li>imgk.jpg</li></ul>  
<button id="start">start</button>
```

La click pe butonul "start" numele fisierelor sunt inlocuite cu imaginile respective (src-ul fiind chiar numele fisierului) astfel: primul fisier este inlocuit cu imaginea care este afisata 3s, dupa 3s prima imagine este inlocuita cu numele fisierului, iar al doilea nume este inlocuit cu a doua imagine pe durata a 3s si asa mai departe.

- img1.jpg
- img2.jpg
- img3.jpg
- img4.jpg
- img5.jpg

start

- img1.jpg
- img2.jpg
- img3.jpg



-
- img5.jpg

start

imagini.html

```
<script>
```

```
window.onload=function(){
    document.getElementById("start").onclick=schimba;
    var img=document.getElementById("imagini").children;
    function imagine(i){ //creeaza o imagine și schimba fișierul i cu imaginea
        var sursa=img[i].innerHTML;
        var imagine=document.createElement("img");
        imagine.src=sursa;
        imagine.alt="eroare";
        imagine.fisier=sursa; //proprietate noua
        img[i].innerHTML="";
        img[i].appendChild(imagine);
    }
    function inapoi(i){ //schimba imaginea cu numele
                        fisierului
        var fisier=img[i].children[0].fisier;
        img[i].innerHTML=fisier;
    }

    function schimba(){
        imagine(0); //prima schimbare
        for(let i=1;i<=img.length;i++){
            setTimeout(function(){ //urmatoarele schimbări la fiecare 3s
                inapoi(i-1);
                if (i < img.length) imagine(i);},3000*i);
        }
    }
}
</script>
```

```
<ul id="imagini">
    <li >img1.jpg</li>
    <li >img2.jpg</li>
    <li >img3.jpg</li>
    <li >img4.jpg</li>
    <li >img5.jpg</li>
</ul>
<button id="start">start</button>
```

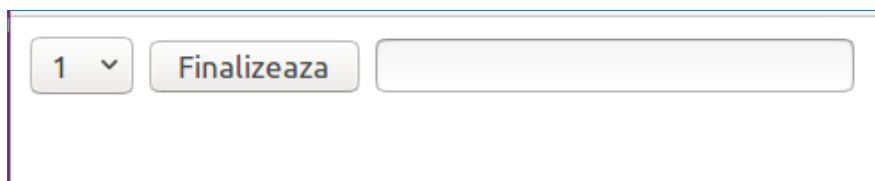
S4. Un fisier HTML contine un element `<select>` ale carui optiuni contin numerele de la 1 la 100, un `<button>` cu id-ul "finalizeaza", un element `<input>` cu id-ul "rezervari" si valoare initiala "" (sirul vid).

La selectarea optiunii n se genereaza n butoane care vor contine numerele de la 1 la n și vor avea initial culoarea verde.

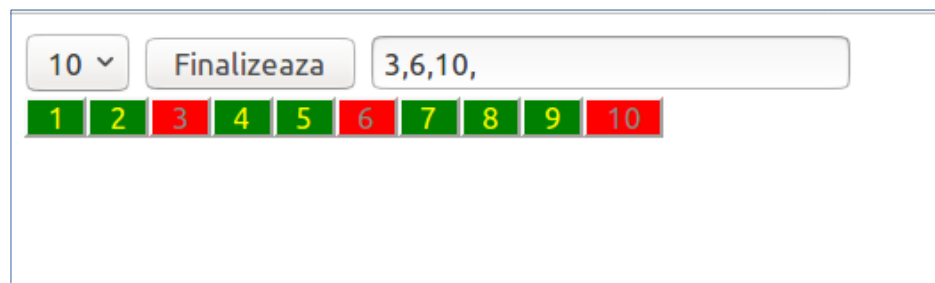
La click pe un buton verde, numarul biletului va fi retinut in input, butonul va deveni rosu, iar clickul nu va mai avea nici un efect.

La click pe butonul "finalizeaza", se va afisa alert cu biletele rezervate (continutul inputului), iar continutul inputului va fi setat la "".

Ex: daca locurile rezervate sunt 3, 50,7 atunci continutul input-ului va fi "3,50,7"



The initial user interface consists of a dropdown menu showing the number '1', a button labeled 'Finalizeaza', and an empty text input field.



After selecting the option '10' from the dropdown, the interface updates. The dropdown now shows '10'. Below it, a horizontal row of ten buttons labeled 1 through 10 is displayed. Buttons 1, 2, 4, 5, 8, and 9 are green, while buttons 3, 6, 7, and 10 are red. The 'Finalizeaza' button remains. The text input field now contains the value '3,6,10,'.



An alert dialog box is shown with the text '" 3,6,10, "' and an 'Ok' button at the bottom.

```
<script>
```

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

```
    var sel=document.getElementById("op");
```

```
    var final=document.getElementById("finalizeaza");
```

```
    var inp=document.getElementById("rezervari");
```

```
    sel.onchange = function() { \la selectarea unei opțiuni din select
```

```
        var n = sel.selectedIndex+1; \optiunea selectata
```

```
        for(var i=0; i<n;i++) { \creare butoane
```

```
            var bt=document.createElement("button");
```

```
            bt.innerHTML=i+1;
```

```
            document.body.appendChild(bt);
```

```
            bt.className="verde";
```

```
            bt.onclick=rezerva; \la click pe butoane se apeleaza functia rezerva
```

```
        }
```

```
    }
```

```
    function rezerva() {
```

```
        inp.value+=this.innerHTML + ",";
```

```
        this.className="rosu";
```

```
        this.disabled=true;
```

```
    }
```

```
    final.onclick = function(){
```

```
        alert(' ' + inp.value + ' ');
```

```
        inp.value="";
```

```
    }
```

```
}
```

```
</script>
```

```
.verde{background-color:green; color:yellow;}
```

```
.rosu{background-color:red;
```

```
<body>
```

```
<select id="op">
```

```
<option value="1">1</option>
```

```
<option value="2">2</option>
```

```
.....
```

```
</select>
```

```
<button
```

```
id="finalizeaza">Finalizeaza</button>
```

```
<input id="rezervari" value="">
```