

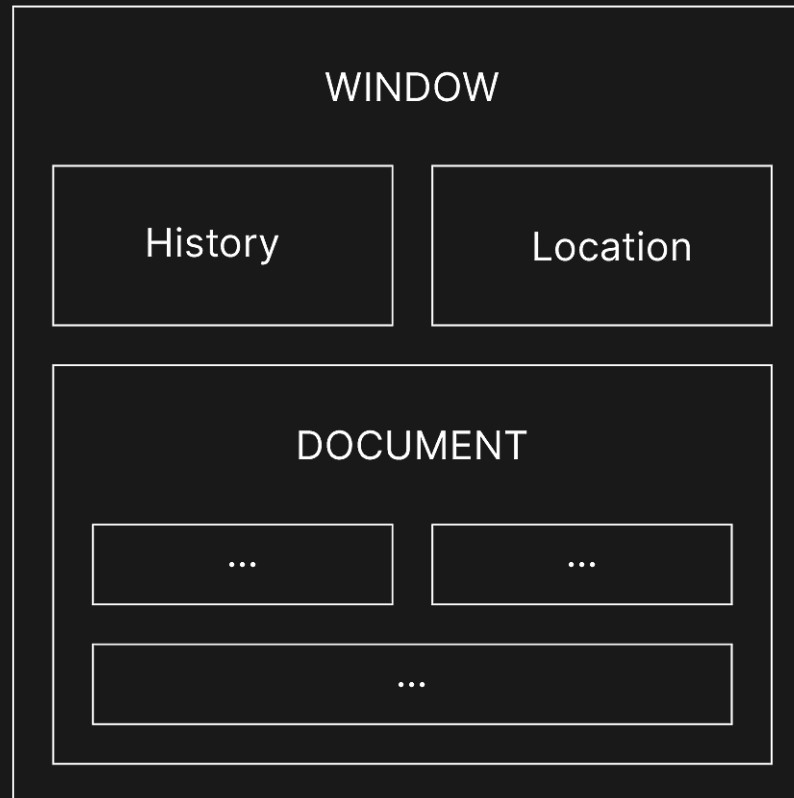
TEHNICI WEB

DOM

Claudia Chiriță . 2024/2025

WINDOW

- orice tab al unui browser conține un obiect *window* al interfeței *Window*



WINDOW

- proprietatea *document* a obiectului *window* are ca valoare obiectul document asociat paginii web

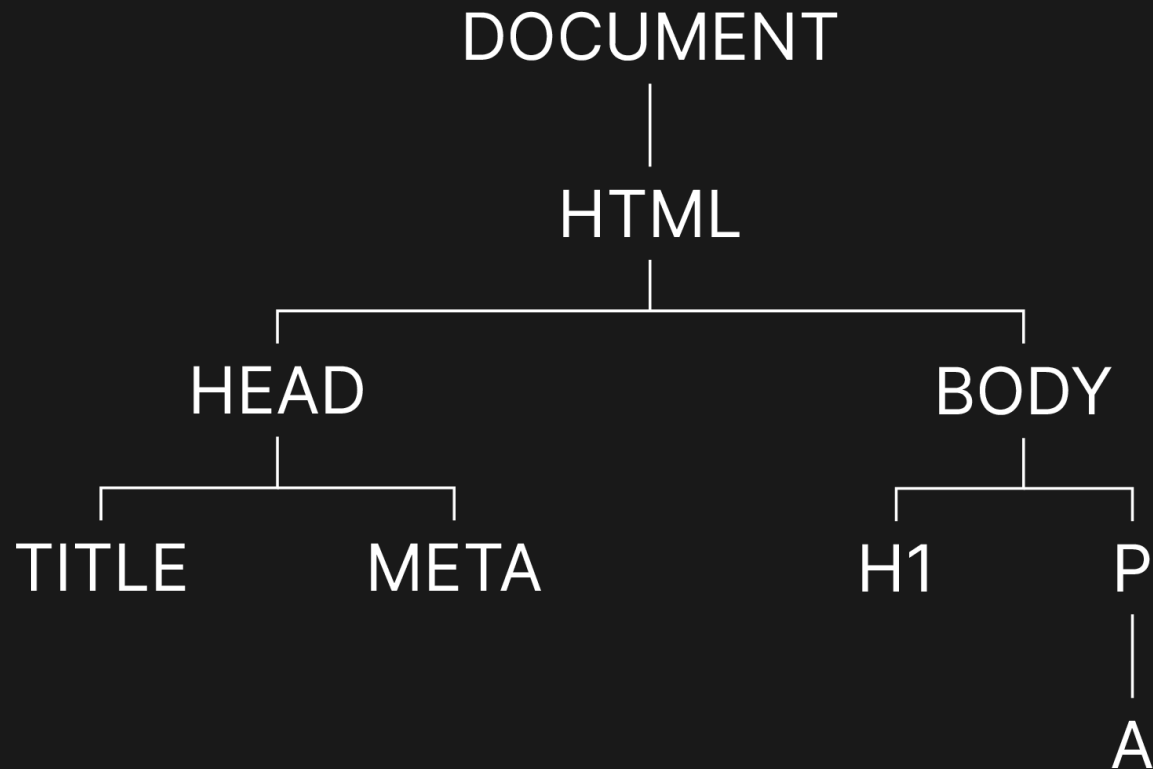
```
window.document // referința
```

- alte proprietăți: history, location, console, localStorage, sessionStorage

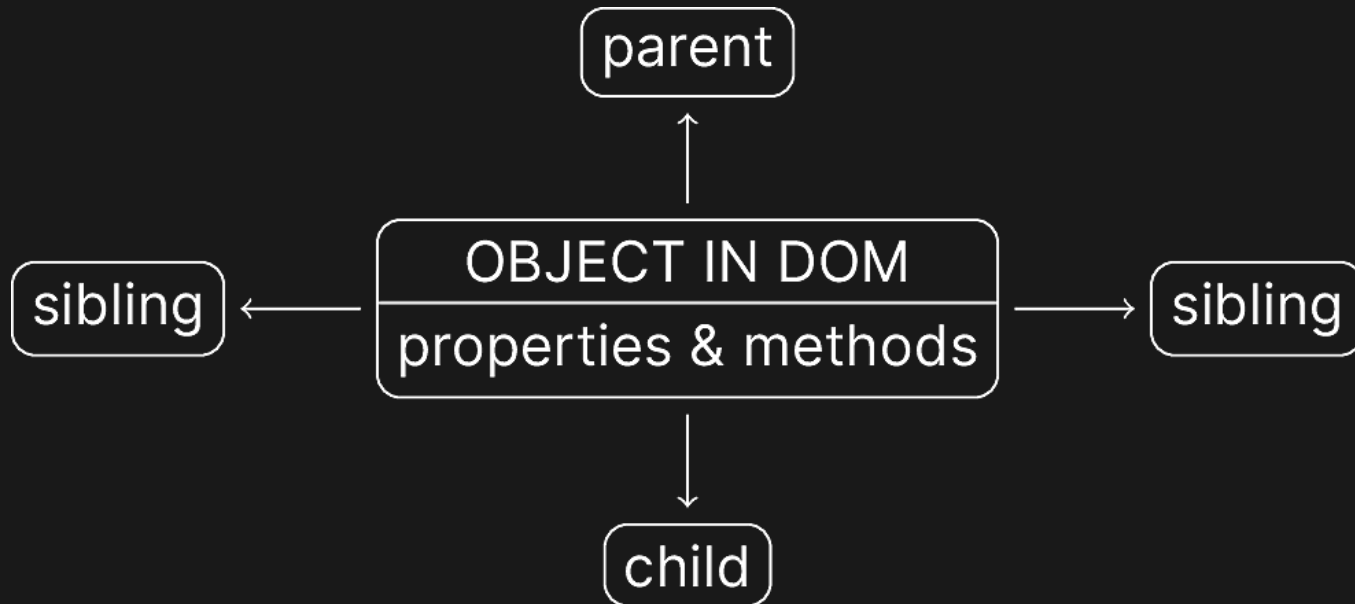
DOM (DOCUMENT OBJECT MODEL)

- API web
- JavaScript folosește DOM pentru accesarea și modificarea documentelor HTML
- orice pagină web este reprezentată în DOM ca un arbore de obiecte și este accesată folosind variabila globală *document*

DOM (DOCUMENT OBJECT MODEL)



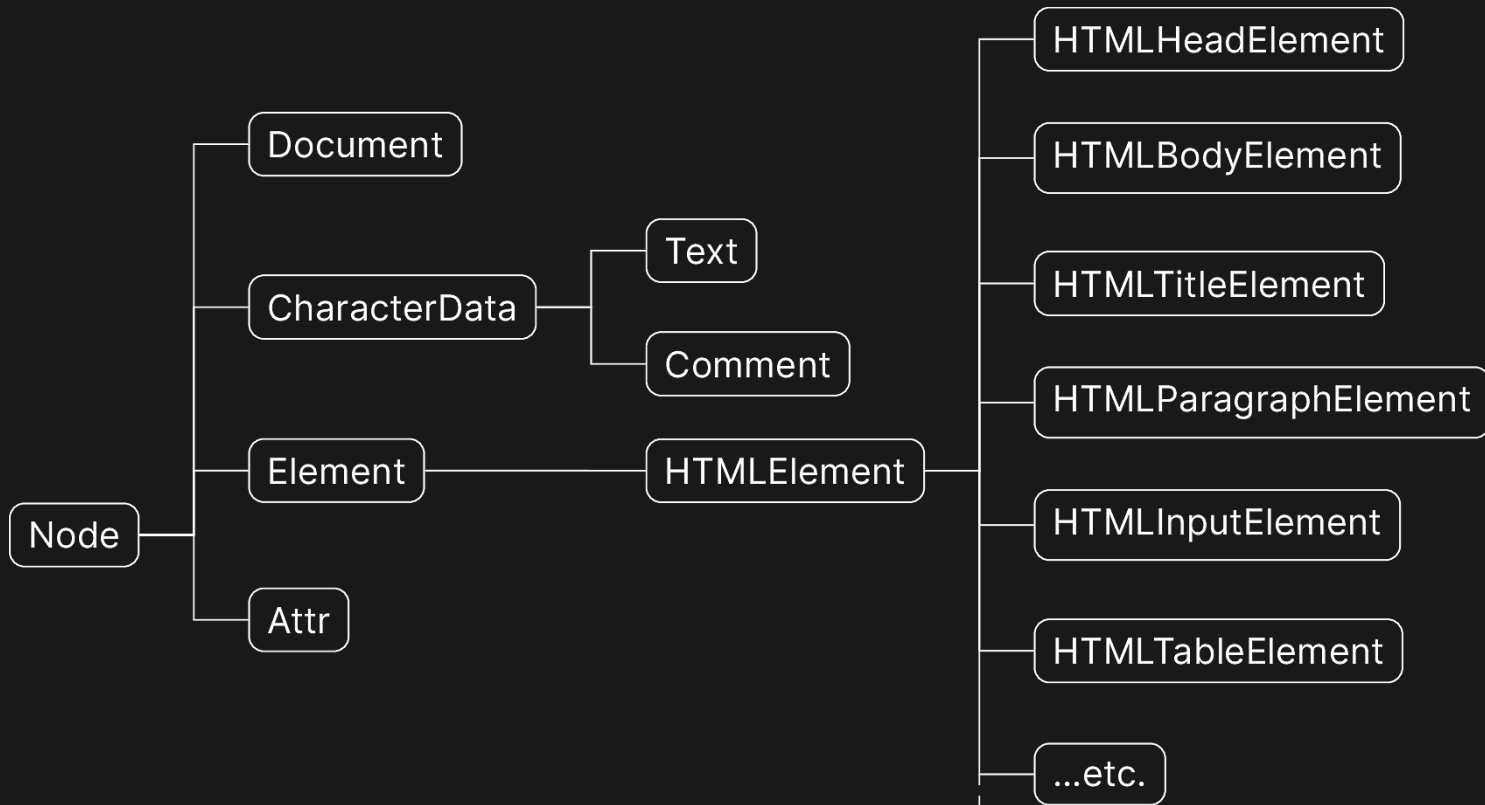
DOM - NODURI



DOM - NODURI

- orice nod din arborele DOM are un tip
- în funcție de elementul pe care îl reprezintă, obiectul corespunzător are proprietăți și metode specifice

DOM - NODURI



ierarhia tipurilor nodurilor din DOM

DOM VS. HTML

- unui tag HTML îi corespunde un obiect (care implementează interfața *Element*)
- unui atribut al unui tag HTML îi corespunde o proprietate a obiectului
- attributele (generale) HTML *id*, *class*, *style* corespund proprietăților *id*, *className*, *style* ale obiectului corespondent

DOM VS. HTML

- proprietățile obiectului *style* (care implementează interfața *Style*) asociat atributului HTML *style* corespund proprietăților de stilizare CSS pentru elementele HTML

```
/* element.style.proprietateCSS = valoare */
```

```
el.style.color = "red"
```

```
el.style.backgroundColor = "blue"
```

DOM VS. HTML - EXEMPLU

```

```

```
object-Element-în-DOM = {tagName: "img", src: "cat.png",  
                           alt: "kitty", id: "catImg"}
```

```
function change() {  
    var x = document.getElementById('catImg'); // object  
    x.alt = "image of a soft warm happy kitty";  
    x.style.border = "10px solid pink";  
}
```

LOAD

- elementul `<script>` din `<head>` este procesat înaintea elementului `<body>`
- arborele DOM este creat mai apoi și deci elementele lui nu pot fi accesate
- încărcarea paginii declanșează evenimentul *load* care are ca țintă obiectul *window*

LOAD

- un handler pentru evenimentul *load* se înregistrează setând proprietatea *onload* pentru obiectul *window*

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

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

LOAD

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* window.onload = change;
         function change(){
           var x =
document.getElementById('catImg'); // object
           x.alt = "image of a happy kitty";
           x.style.border = "10px solid pink";
         } */
    </script>
  </head>
  <body>
    
  </body>
</html>
```



SELECTAREA ELEMENTELOR

```
document.getElementById(numeId) // un obiect  
document.querySelector(selectorCSS) // primul obiect
```

colecții *live*

```
document.getElementsByClassName(numeClasa)  
document.getElementsByTagName(numeTag)  
document.getElementsByName(nume)
```

colecții *statice*

```
document.querySelectorAll(selectorCSS)
```

COLECȚII

- colecții generice similare unui Array
- au proprietatea *length*
- metodele Array nu pot fi invocate direct
- exercițiu:

Afișați numărul elementelor `<h1>` care sunt descendenți direcți ai elementelor `<section>` cu clasa "special".

COLECȚII

- colecții generice similare unui Array
- au proprietatea *length*
- metodele Array nu pot fi invocate direct
- exercițiu:

Afișați numărul elementelor <h1> care sunt descendenți direcți ai elementelor <section> cu clasa "special".

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

COLECȚII

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* window.onload=function() {
        var c =
document.getElementsByClassName("abc");
        var v = [];
        for(var i = 0; i < c.length; i++)
          v[i] = c[i];
        alert("lungimea colecției: " +
c.length + '\n' +
          "lungimea vectorului: " +
v.length);

document.getElementById("p1").className="abc";
        alert("lungimea colecției: " +
c.length + '\n' +
          "lungimea vectorului: " +
v.length);
      }*/
```

>

COLECȚII

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* window.onload=function() {
        var c =
document.querySelectorAll(".abc");
        var v = [];
        for(var i = 0; i < c.length; i++)
          v[i] = c[i];
        alert("lungimea colecției: " +
c.length + '\n' +
          "lungimea vectorului: " +
v.length);

document.getElementById("p1").className="abc";
        alert("lungimea colecției: " +
c.length + '\n' +
          "lungimea vectorului: " +
v.length);
      }*/
```

>

COLECȚII - QUIZ

Schimbarea numelui unei clase "old" în "new":

```
var colectie = document.getElementsByClassName("old");  
  
for(var i = 0; i < colectie.length; i++)  
    colectie[i].className = "new";  
  
/* merge? */
```

CONȚINUTUL ELEMENTELOR

accesarea și modificarea conținutului unui element
HTML ca String folosind

- **innerHTML**
 - întoarce un text HTML, adică un text cu marcaje
 - la setarea proprietății, browserul interpretează textul

```
<p>It was <i> the best </i> of times.</p>  
<--! innerHTML -->  
It was <i> the best </i> of times.
```

CONȚINUTUL ELEMENTELOR

accesarea și modificarea conținutului unui element
HTML ca String folosind

- **textContent**
 - întoarce text fără marcaje
 - are ca rezultat concatenarea conținuturilor descendenților de tip Text

```
<p>It was <i> the best </i> of times.</p>  
<--! textContent -->  
It was the best of times.
```

NODURI DOM

tipuri de noduri:

- un nod special numit **document**
- noduri de tip **Element** (modelează tagurile HTML)
- noduri de tip **Text** (modelează textul)

NODURI DOM

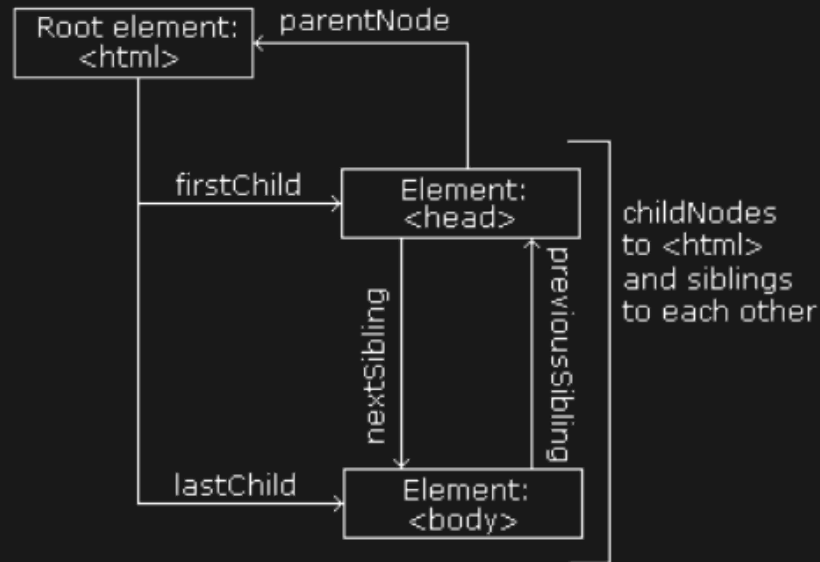
manipularea informației din noduri cu:

```
nodeValue /* pentru noduri Text, Comment
           pentru noduri Element = null */
nodeName // numele tagului
nodeType /* Document=9, Element=1,
          Text=3, Comment=8 */
```

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* window.onload=function() {
        var p =
document.getElementById("par");
        alert("nodeValue: " + p.nodeValue +
'\n' + "nodeName: "
          + p.nodeName + '\n' + "nodeType: " +
p.nodeType);
      }*/
    </script>
  </head>
  <body>
    <div id="par">Everybody wants to be a cat!
```



NAVIGAREA ÎN ARBORE



```
node.parentNode // un obiect
node.childNodes // obiect NodeList
node.firstChild // un obiect
node.lastChild  // un obiect
node.nextSibling // un obiect
node.previousSibling // un obiect
```

NAVIGAREA ÎN ARBORE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* 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("lista").innerHTML +=
txt;

        alert(document.getElementById("lista").childNodes[0].nodeName);
      }*/
    </script>
  </head>
```

>

NAVIGAREA ÎN ARBORELE DE ELEMENTE

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

putem defini metode noi

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

NAVIGAREA ÎN ARBORELE DE ELEMENTE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* 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("lista").innerHTML +=
        txt;

        alert(document.getElementById("lista").childNodes[0].nodeName);
      }*/
    </script>
  </head>
```

>

NAVIGAREA ÎN ARBORE - QUIZ

- o funcție `find(e, className)` care pentru un element `e` din DOM, întoarce primul strămoș care conține clasa `className`

```
function find(e, className) {  
    var p = e.parentElement;  
    while (p !== document.documentElement) {  
        if(p.classList.contains(className))  
            return p;  
        p = p.parentElement;  
    }  
    return null;  
}
```

NAVIGAREA ÎN ARBORE - QUIZ

- o funcție `frati(e)` care pentru un element `e` din DOM, calculează numărul fraților de același tip

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

Operații cu Elemente

crearea unui element

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

inserarea unui element

```
parinte.appendChild(copil)  
// dacă nodul copil există în arbore, doar mută nodul  
parinte.insertBefore(CopilNou, CopilVechi)
```

ștergerea / înlocuirea unui element

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

Operații cu Elemente

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* function create(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 = create("li","everything
nice");
        list.appendChild(el1);
        var el2 = create("h2","Powerpuff");

document.body.insertBefore(el2,list);
```

>

MODIFICAREA ATRIBUTELOR

- attributele elementelor HTML devin proprietăți ale obiectelor corespunzătoare
- pot fi accesate prin
 - numele proprietății
 - metode specifice
 - proprietatea `attributes`

MODIFICAREA ATRIBUTELOR

- proprietăți: `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")  
// adaugă un atribut și valoarea lui  
  
el.hasAttribute("nume-atribut") // întoarce boolean  
el.removeAttribute("nume-atribut") // șterge atributul
```

```
el.proprietateNoua = valoare // adăugare de proprietăți noi
```

MODIFICAREA ATRIBUTELOR

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* window.onload = function() {
        var x =
document.getElementById("imag1");
        x.src = "2.jpg"; // schimb sursa
imaginii
        var y =
document.getElementById("imag2");
        y.src = "3.jpg"; // schimb sursa
imaginii
      }
      */
    </script>
  </head>
  <body>
    
     " +
               attrs[i].value + " // ";
}
alert(output);
```

JAVASCRIPT + CSS = STYLE

- orice obiect asociat unui element HTML (interfața `Element`) are proprietatea `style`, a cărei valoare este un obiect care implementează `CSSStyleDeclaration`
- proprietăților CSS le corespund proprietăți ale obiectului *style*
- schimbarea stilului unui element HTML:

```
element.style.proprietate = stil nou
```

JAVASCRIPT + CSS = STYLE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* function schimbaStil(el) {
        el.style.color = "pink";
        el.style.fontFamily = "Arial";
        el.style.fontSize = "5em";
      }
      window.onload = function() {

schimbaStil(document.getElementById("par"));
      } */
    </script>
  </head>
  <body>
    <p id="par">Bravo, ai stil!</p>
  </body>
</html>
```

>

JAVASCRIPT + CSS : CLASE

- clasele asociate unui element pot fi accesate folosind proprietatea `classList`, care implementează `DOMTokenList`
- lista claselor elementului:

```
ecl = element.classList
```

JAVASCRIPT + CSS : CLASE

```
<script>
  window.onload = function() {
    var pclas = document.getElementById("par").classList;
    alert(pclas);
  }
</script>
</head>
<body>
<p id="par" class="c1 c2 c3 c4"> Hakuna Matata! </p>
```


JAVASCRIPT + CSS : CLASE

```
ecl = element.classList;

ecl.length // nr. de clase asociate elementului
ecl.item(i) // numele clasei cu indexul i (i = 0,..)
ecl.add("clasa1", "clasa2", ...)
ecl.remove("clasa1", "clasa2", ...)
ecl.contains("clasa") // returns boolean

ecl.toggle("clasa", expresie)
/* șterge clasa dacă aceasta există, altfel o adaugă;
   expresie => boolean: if true, se adaugă clasa,
                       if false, se șterge clasa */
```