Curs 8 - Javascript

JS → limbaj de front-end (client-side) → la început
→ Node JS → back-end

ActionScript → animațiile Flash și, mai târziu, Adobe

JS → limbaj interpretat de browser
→ limbaj orientat obiect, orientat eveniment, case sensitive
→ permite: * interacțiuni cu pagina
* modificare dinamică: adaugă / șterge tag în pag.
* create events
* validare date → pentru o interfață friendly
acestea sunt opționale! ↳ nu sunt 100% de încredere
validare de pe back-end e OBLIGATORIE! pt. că userul are acces la cod

→ weakly typed → o variabilă poate lua orice valoare

<script type="text/javascript"> ... cod JavaScript... </script>
<script type="text/javascript" src="myscript.js"></script>
TREBE!

<noscript> Se execută doar dacă browser-ul nu poate executa cod JavaScript </noscript>

* în mod tradițional: tag-ul <script> se află în secțiunea <head>
* el se poate pune oriunde

! Cât timp se execută codul JS, nu se fac alte acțiuni
°
=> user-ul nu poate da click și pagina nu se randează

* funcții → declarate cu „function", ex: function sum(a,b)
{ return a+b; }

→ există și funcții anonime, ex: d.onclick() = function() {...}

→ o funcție anonimă poate fi apelată fix după definire

⇒ economie de memorie: (function (a,b) {...}) (2,3); apelul fct.

* variabile: var i = 7; i = 'Ana'; i = "mere"; i = 3,1415;

↳ number, string, boolean, object, function, undefined

→ operatorul „typeof" returnează tipul elementului

Ex: typeof "1.5" → "string", typeof eval("1,5") → "number"

↳ execută cod JS

1/0 → Infinity, typeof Infinity → "number", typeof 1/0 → NaN

* array: var y = new Array(5,6,7,8); → array cu 4 elem.

var z = new Array(11); → array cu 11 elem. undefined

↳ pot avea elemente cu tipuri diferite

↳ indexate de la 0

↳ push(list.elem), pop(), shift() ..., se pot face sortări custom

* obiect: colecție neordonată de valori, care au împreună o semantică

Ex: datele și acțiunile unei persoane

↳ se pot defini atribute și după declarare (sau funcții) și delete

↳ atributele se pot itera cu for

↳ este corect person.name și person["name"], dar și person[2]

"==" compară strict valori; "===" compară și tipul

strpos → returnează poziția unde se găsește, grija la poz 0!

- funcțiile definite ajung membri pe window (global)
- alert, document -> sunt deja în browser

* DOM: document.getElementById("un id") apelul de funcție
  - prima dată trebuie să apară în cod acel id, apoi codul JS
  - pe select există funcția add (adaugare de option), la fel și appendChild, diferența e că appendChild merge și pe div

* evenimente: se pot asocia evenimente butoanelor ((1 sau mai multe)
→ .addEventListener('click', doSmt()); este și delete