**JavaScript**

Adaugare cod JS in pagina HTML

<script> Code JS </script>

Sau <script src=”path\_catre\_code.js”> </script>

Acesta se afla de obicei intre tag-urile <head> </head>. Dar daca scriptul genereaza niste continut pentru pagina de HTML, atunci scriptul trebuie pus intre tag-urile <body> </body> la pozitia unde este nevoie de el.

Puteti citi mai multe despre pozitionarea <script> [aici](https://www.digitalocean.com/community/tutorials/how-to-add-javascript-to-html)

**Let, const, var**

Pentru a declara variabile in JS se foloseste let. Var este folosit pentru a declara scopuri globale, dar nu este o buna practica. Const este pentru a declara variabile constante, care nu isi modifica valoarea.

Puteti citi [aici](https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/) mai multe despre diferenta dintre cele 3 si de ce e recomandat sa folositi **let** si **const**

Also, **const** are scope de **block**, la fel ca si **let**, **nu** este **global**.

**Array si object**

Pentru declararea lor se foloseste tot **let**

Let array1 = [1, 2, ‘3’, false, “Ana”] // poate contine int-uri, string-uri, bool, obiecte, etc.

Let object1 = {

“nume”: “Bogdan”,

“email”: “[bogdan.nitica99@gmail.com](mailto:bogdan.nitica99@gmail.com)”,

“note”: {

“tehniciWeb”: 4,

“poo”: 4.3

}

}

Pentru a acesa un element dintr-un array se utilizeaza [ indice ]

Pentru a acesa un element dintr-un obiect se utilizeaza `.` object1.nume sau object1.note.tehniciWeb.

**== vs ===**

== verifica valoarea fara sa tina cont de tipul de date

5 == 5 **true**

“5” == 5 **true**

=== verifica valoarea, inclusiv tipul de date

5 === 5 **true**

“5” === 5 **false**

Pentru == pot aparea situatii dubioase:

0 == false **true**

0 == [ ] **true**

0 == { } **false**

Pentru un tabel de diferente intre == si === (si situatiile dubioase de la == ) puteti citi [aici](https://dorey.github.io/JavaScript-Equality-Table/)

Cum spune si link-ul de mai sus, “Always use 3 equals unless you have a good reason to use 2.”

**Functions**

Structura unei functii in JS. Keyword-ul pentru definita unei functii este **function** urmat de numele functiei. Exemplu de functie

function adunareDouaNumere(a, b) {

return a+b;

}

Parametrii functiei pot primi valori default ex: function adunare(a=10, b=5). In cazul in care la apelarea functiei nu sunt date valori pentru parametrii rezultatul o sa fie 15.

**getElementById**

Retunreaza obiectul HTML cu id-ul respectiv. Sintaxa: document.getElementById(“id1”)

Pentru mai multe exemple puteti citi [aici](https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById)

**innerHTML**

Pentru a modifica text-ul unui element HTML se foloseste proprietatea innerHTML. Pentru a obtine obiectul HTML ne folosim de getElementByID. Exemplu, daca vrem sa modificam textul unui paragraf cu id-ul “paragraf”:

let pObject = document.getElementById(“paragraf”)

pObject.innerHTML = “Alt text”

Sau intr-o singura linie: document.getElementById(“paragraf”).innerHTML = “Alt text”

Pentru a aplica aceasta schimbare de text este nevoie ca o functie sa apeleze acest cod JS, altfel compilatorul nu o sa ruleze codul JS. Functia poate fi apelata cu ajutorul unui buton din HTML cu atributul onClick. Exemplu

<buton onClick=”schimbaText()”>Click aici</buton>

**Exercitiul 1**

Scrieti un cod JS care sa modifice textului unui paragraf HTML in “Numele meu este: insert\_your\_name\_here”. Numele trebuie sa fie un parametru al functiei care sa fie transmis de catre apelarea din buton.

**Exercitiul 2**

Scoateti duplicatele dintr-o matrice JS de tipul [5, 5, ‘5’, 3, 2, ‘A’, ‘B’, ‘A’, 3]. Rezultatul trebuie sa fie [‘5’, 2, ‘B’]. Afisati noua matrice in consola. Rezolvati in 2 moduri, unul cu 2 for-uri si unul cu filtru.

**Tema**

Deadline **26.03.2022,** rezolvati exercitiile de mai sus.