PROIECT FINAL

CIPRIAN COJOCARIU

09 Septembrie 2023

HTML

- **Tag**-urile sunt marcaje ce conțin simboluri sau cuvinte predefinite, pe care limbajul HTML le foloseşte, pentru a transmite browser-ului cum să afişeze conţinutul paginii web. Sunt încadrate de parantezele unghiulare „<” și „>”, între care scriem numele specific al tag-ului. Tag-ul arată începutul sau sfârșitul unui element HTML (p - paragraf, în cazul de față).
- **Head**—ul <head> este un element care nu adaugă nimic conținutului paginii, dar oferă diverse informații despre pagină. În interiorul său apar elemente prin care transmitem browser-ului: titlul paginii <title>, calea către fisierele CSS sau framework-urile utilizate (ex. Bootstrap), anumite librării (ex. Font Awesome), cum ar trebui citit documentul de browser (char set) sau o descriere a conținutului paginii <meta>, dpv SEO.
- **Body**-ul <body> este elementul în interiorul căruia este cuprins tot conținutul paginii. Împreună cu <head>, este al 2-lea copil al elementului <html>, care este elementul principal al paginii. În <body> vom pune absolut toate elementele care dorim să fie afișate de către browser.

CSS

- **Specificitatea** este un scor/rang, care decide ce declarație de stil trebuie utilizată pentru un element. Cu cât o regulă dispune de mai mulți selectori HTML, de clasă și de id, cu atât scorul ei crește. În cazul unui conflict între reguli, regula cu scorul cel mai mare se va aplica elementului.
  - Id-urile au valoarea 100
  - Clasele au valoarea 10
  - Selectorii HTML au valoarea 1.
- **Clasa** se folosește în CSS pentru a selecta mai multe elemente HTML. Clasa apare în HTML ca și atribut al elementelor ce vor fi selectate. Ex.: .imgStyle {… CSS code…} selectează elementele HTML care au atributul class="imgStyle".
- **Id**-ul se folosește în CSS pentru a selecta un singur element, iar în HTML id-ul este un atribut al elementului respectiv. Ex.: #bestProduct {… CSS code…} selectează doar elementul HTML care are atributul id="bestProduct".

CSS – Tipuri de selectori

- **Selectori simpli** - selectează folosindu-se doar de:
  - *element* - p, a, h1
  - *clasă*: .clasa1, .textTitlu
  - *id*: #target
- **Selectori combinatori** - selectează pe baza unei relații între elementele care apar în criteriul de selecție:
  - *descendant* selector (space)
  - *child* selector (>)
  - *adjacent* sibling selector (+)
  - *general* sibling selector (~)
- **Pseudo-selectori** - selectează pe baza unei stări a elementului, sau doar o anumită parte a acestuia:
  - *de clasă* – a:hover, input:checked, a.highlight:hover - selecteaza pe baza stării
  - *de element* – p::first-line, p.intro::first-letter - selectează doar o anumită parte din element
- **Selectori de atribut**: – selectează în funcție de:
  - *atribut* - a[href]
  - *valoarea unui atribut* - a[target=„..."]

JAVASCRIPT - Variabile

- **Variabilele** sunt locații de memorie în care pot fi stocate date. În funcție de tipul de date asociat unei variabile, este alocată o anumită cantitate de memorie, pentru a stoca în ea valoarea variabilei.

Tipuri de variabile

1. Din punct de vedere al domeniului de acces („scope”) la conținutul lor, variabilele sunt:
   - ***Locale*** – sunt vizibile și pot fi accesate doar în „scope”-ul în care au fost declarate (bloc de cod, funcție etc)
   - ***Globale*** - sunt vizibile și pot fi accesate în tot codul
2. Din punct de vedere al posibilității de modificare a valorii inițializate, variabilele sunt:
   - ***Mutabile*** - se declară cu „let variableName”, iar valoarea lor poate fi modificată în cod, după inițializare. Se poate folosi și declararea (nerecomandată) cu „var variableName”, declarând variabila ca și globală (risc de bug-uri), fără scope.
   - ***Constante*** - se declară cu „const variableName”, iar valoarea lor nu mai poate fi modificată în cod, după inițializare.

JAVASCRIPT – Tipuri de date

- **Tipurile de date** se împart în 2 mari categorii:

PRIMITIVE

- ***String***, folosit pt. șiruri de caractere alfanumerice și simboluri
- ***Number***, folosit pt. numere întregi sau cu zecimale
- ***BigInt***, folosit pt. numere întregi mai mari, ce nu pot fi reprezentate cu ***Number***
- ***Boolean***, folosit pt. valori logice (true/false, 0/1, on/off)
- ***Undefined***, folosit în cazul unei variabile care nu este inițializată
- ***Null***, folosit pentru a exprima un conținut „gol” al unei variabile sau „nimic” și care se atribuie prin cod de către developer, nu implicit
- ***Symbol***, un identificator unic și imutabil ce poate fi folosit ca și cheie pt. o proprietate a unui Object

NON-PRIMITIVE

- ***Object*** este o colecție de date, folosit la reprezentarea structurilor de date în forma perechilor „cheie:valoare”.
- ***Array*** este o colecție de date, folosit pentru a grupa mai multe date într-o singură variabilă. Este un tip special de “Object”.
- ***Date*** este folosit pentru a reprezenta o dată calendaristică, fiind tot un tip special de “Object”.

JAVASCRIPT – Condiționalul “if else”

- **If else** se folosește în situația în care dorim ca un anumit bloc de cod să fie executat, doar dacă este îndeplinită o condiție logică.

SINTAXA

```
if (condiție1) {
bloc de cod 1
} else if (condiție2) {
bloc de cod 2
}
else {
bloc de cod 3
}
```

SECVENȚA DE EXECUȚIE

- Se testează valoarea de adevăr a *“condiție1”*.
  1. Dacă este ”true”, se execută “bloc de cod 1” din ramura ”if”.
  2. Dacă este “false”, se trece la evaluarea “condiție2”. Ramura “else if” este opțională.
- Se testează valoarea de adevăr a “condiție2”.
  1. Dacă este ”true”, se execută “bloc de cod 2” din ramura ” else if”.
  2. Dacă este “false”, se execută “bloc de cod 3” din ramura ”else”. Ramura “else” este opțională.

JAVASCRIPT – Cicluri repetitive – *FOR (FOR IN, FOR OF)*

- **Ciclurile repetitive** (buclele – "loops") se utilizează când dorim să executăm un bloc de cod în mod repetat (cu număr de iterații cunoscut, sau nu). La fiecare iterație se evaluează valoarea de adevăr („true" sau „false") din condiția de execuție a buclei, iar codul din corpul buclei se execută atâta timp cât valoarea condiției este „true".
- **Tipuri de cicluri**:
  - cu număr de iterații cunoscut – ***FOR***
  - cu număr de iterații necunoscut – ***WHILE, DO WHILE***

SINTAXA

for (expresie1;
condiție;
expresie2;)
{ *bloc de cod* }

SECVENȚA DE EXECUȚIE

1. *"expresie1"* este pasul de inițializare a buclei, cu această valoare se pregătește execuția primei iterații. Se execută o singură dată.
2. se testează valoarea de adevăr din *"condiție"*. Doar dacă este "true", *bloc de cod* va fi executat. În caz contrar, se iese din buclă și programul continuă execuția cu liniile de cod care urmează buclei *for*.
3. dacă iterația anterioară a fost executată, se execută *"expresie2"*, prin care se pregătește execuția următoarei iterații. Apoi se reia pasul 2 și bucla continuă secvențial, reluând pașii 2 si 3.

FOR IN

for (key ***in*** object)
{ *bloc de cod* }

FOR OF

for (value ***of*** object)
{ *bloc de cod* }

EXPLICAȚII

1. *"for in"* parcurge proprietățile unui obiect sau array. Fiecare iterație returnează o cheie, care este folosită pentru a accesa valoarea ce îi corespunde.
2. *"for of"* parcurge valorile unui obiect *iterabil*, de ex Array, String, Map. La fiecare iterație, valoarea următoarei proprietăți este atribuită variabilei "value".

JAVASCRIPT – Cicluri repetitive

WHILE

SINTAXA

while

(condiție)

{ *bloc de cod* }

SECVENȚA DE EXECUȚIE

1. se testează valoarea de adevăr din *“condiție”*. Doar dacă este “true”, *bloc de cod* va fi executat. În caz contrar, se iese din buclă și programul continuă execuția cu liniile de cod care urmează buclei *while*.
2. conform celor menționate anterior, bucla se poate executa o dată, de mai multe ori, niciodată, la infinit.
3. pentru a evita ”bucla infinită” - rularea la infinit a *bloc de cod*, acesta va trebui ca, în conținutul său, să determine modificarea valorii *“condiție”* la “false”.

DO WHILE

SINTAXA

do

{ *bloc de cod* }

while (condiție);

SECVENȚA DE EXECUȚIE

1. funcționează similar cu bucla *while*, cu singura deosebire că *bloc de cod* va fi executat cel puțin o dată, indiferent de valoarea de adevăr din *“condiție”*.
2. începând cu a 2-a iterație, funcționează exact ca o buclă *while*. Va fi executată conform 1 de mai sus, însă poate fi executată de mai multe ori sau la infinit.

JAVASCRIPT – Funcții, parametri

FUNCȚII

- ***funcția*** este un corp de cod, care realizează o anumită funcționalitate.
- putem executa acest corp de cod ori de câte ori dorim să îl folosim, fără a-l scrie în întregime, din nou.
- funcția poate fi definită în program ca având nevoie de anumite date de intrare, numite ***parametri***. Valoarea parametrilor determină valoarea returnată de funcție.

PARAMETRI

- ***parametrii*** reprezintă datele de intrare de care are nevoie funcția, pentru a fi executată corect. Valorile parametrilor nu se cunosc în momentul definirii funcției, dar vor fi transmise acesteia la momentul apelării ei.
- un parametru „ține locul” unei valori de care are nevoie funcția, atunci când este apelată. La apelul ei, valoarea parametrului este transmisă funcției, iar cu această valoare se va executa corpul funcției.

PROIECT PRACTIC – Site prezentare portofoliu

https://github.com/cipric2005/portfolio-website
https://cipric2005.github.io/

https://github.com/cipric2005/portfolio-website
https://cipric2005.github.io/

DESCRIERE

- Aplicația este un mini-proiect de prezentare servicii și portofoliu proiecte. În acest moment apropiat terminării cursului de WD, nu am finalizate lucrari pentru portofoliu, dar ele vor fi adăugate pe măsura realizării lor.
- Website-ul este responsive pentru mai multe tipuri de dispozitive (mărimi de ecrane), funcționalitate obținută în special cu ajutorul framework-ului Bootstrap, dar și cu cod CSS.

STRUCTURA APLICAȚIEI

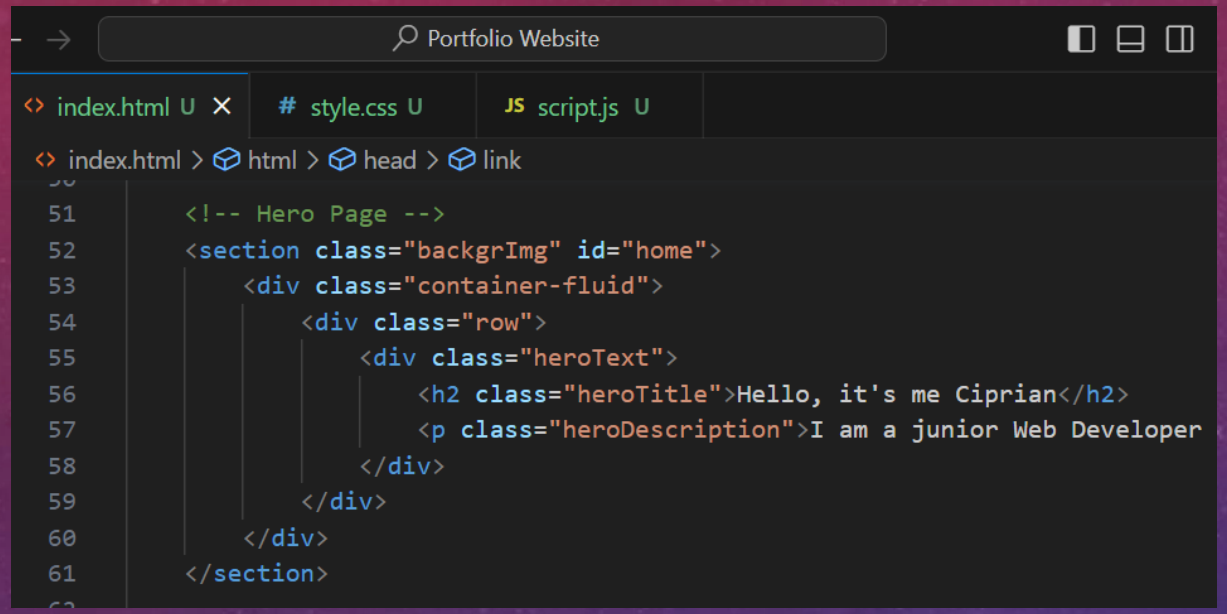
Aplicația este un ***website de prezentare pe o singură pagină***, având următoarele elemente:

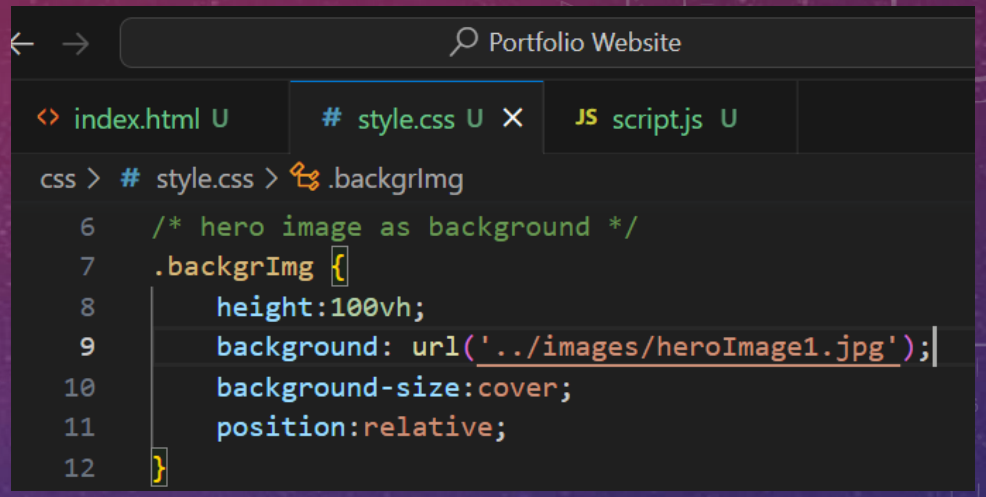
- Meniul de navigare: Home, About, Services, Portfolio, Contact
- Pagina de imagine „***Home***” (landing page) de tipul „Hero Page”.
- Secțiunea „***About***”, care conține informații profesionale, din experiența mea, relevante pentru rolul de Web Developer
- Secțiunea „***Services***”, în care sunt prezentate serviciile oferite de mine
- Secțiunea „***Portfolio***”, în care urmează să prezint proiectele realizate, ulterior finalizării lor
- Secțiunea „***Contact***” care dă posibilitatea celor interesați să trimită un mesaj de contact
- Secțiunea „***Footer***”, care conține profil LinkedIn și modalități de contact – e-mail și tel.

Limbaje și framework-uri folosite:

- HTML
- CSS
- Bootstrap5 (framework de CSS)
- Javascript

DETALII COD HTML, CSS

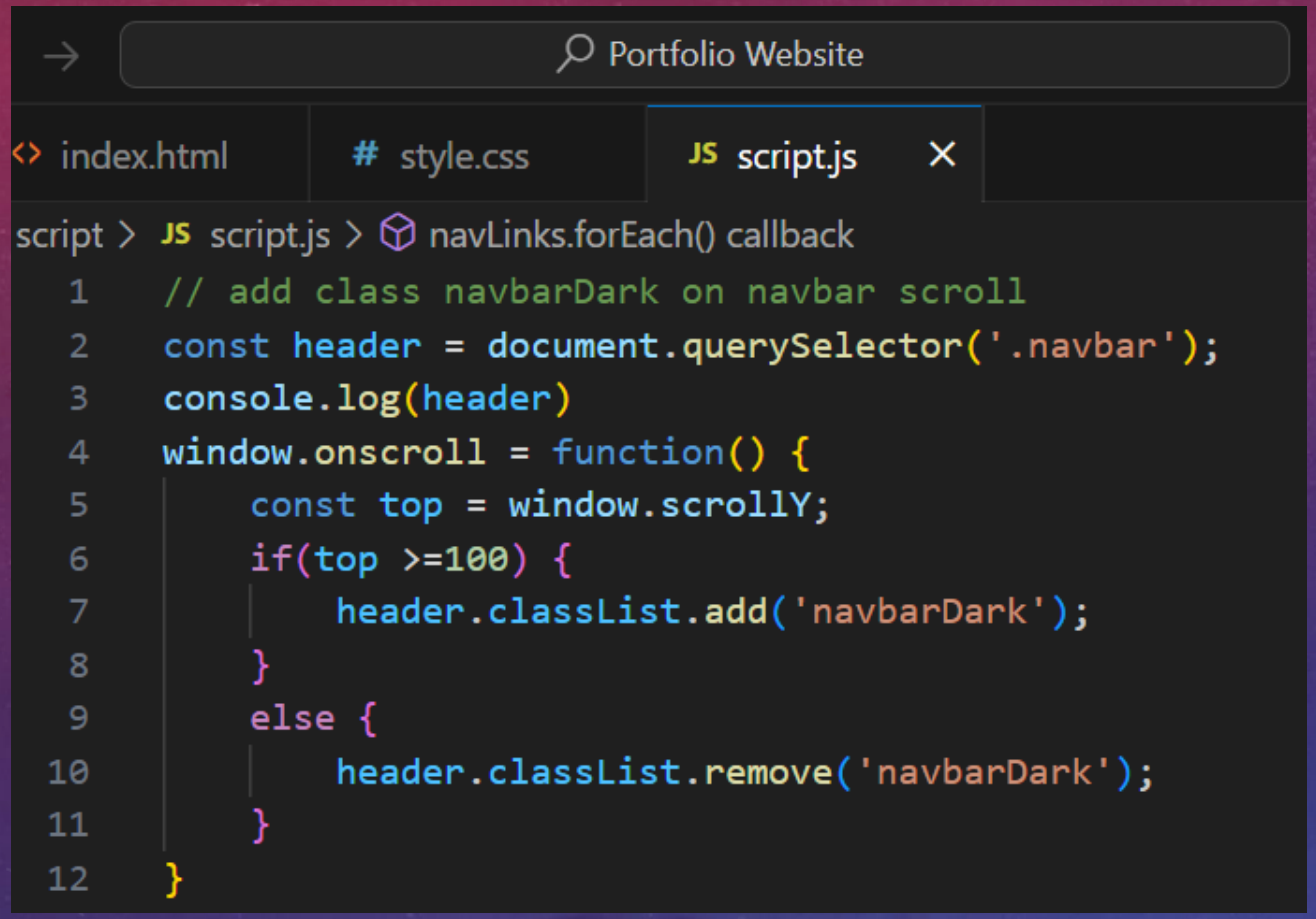




Tag-ul <section class="backgrImg" id="home"> este un tag semantic, ce definește și conține imaginea „Hero” care va fi afișată pe tot ecranul, indiferent de dispozitiv (class=”container-fluid”) împreună cu textul conținut în elementul . Tag-ul <section> mai conține o clasă și un id pentru stilizare CSS, respectiv "backgrImg" și "home".

.backgrImg selectează după clasă, elementul <section class="backgrImg"> asupra căruia va aplica stilizarea definită. Astfel, heroImage1.jpg din linia 9, va ocupa întreg viewport-ul (height:100vh;). Linia 10 va ajusta automat imaginea la mărimea viewport-ului (eventual ajustând mărimea, sau prin întindere/îngustare).

DETALII COD JAVASCRIPT



Codul din liniile 2-12 are rolul de a modifica fundalul navbar-ului pe negru, în timpul derulării paginii pe verticală. Asta deoarece meniul nu ar fi suficient de vizibil la scroll (lipsă contrast), în lipsa acestei modificări. În linia 2 selectăm meniul - elementul navbar - și îl reținem în constanta „header”. Apoi definim o funcție „onscroll” pe obiectul „window”, care verifică derularea pe verticală (scroll) a paginii, iar la un scroll în jos de minim 100 de pixeli, clasa „navbarDark” este adăugată elementului navbar (linia 7) și ulterior se aplică stilizarea CSS (linia 3 din fișierul style.css).

VĂ MULȚUMESC!