Tema 3 Tehnologii Web Țuluca Maria Laura Grupa 322AA

Proiectul meu implementează procesarea unei imagini preluate dintr-un API, aplicând transformări logaritmice ale nivelului de gri și un efect de oglindire.

**1. HTML-ul** definește structura paginii web care conține butoane pentru generarea și aplicarea efectelor pe imagine, secțiuni pentru a arăta imaginea originală și imaginea procesată, precum și afișarea datelor de procesare.

* **<div class="container">**: containerul principal pentru pagina de aplicație.
* **<h1>**: titlul aplicației.
* **<button id="generateImageButton">**: buton pentru generarea imaginii.
* **<button id="applyEffectsButton">**: buton pentru aplicarea efectelor.
* **<div id="jsonDataContainer">**: secțiune pentru afișarea datelor JSON preluate de la API.
* **<div id="imageContainer">**: container pentru imaginile originală și procesată.
* **<div id="timingContainer">**: afișează timpii de procesare pentru fiecare etapă.

**2. CSS-ul** se ocupă cu aspectul vizual al aplicației. Principalele secțiuni sunt:

* **body**: stilizează fundalul și centrarea conținutului.
* **.container**: setează aspectul general al containerului, inclusiv margini, umbră și dimensiune.
* **#imageContainer**: împărțirea imaginii în două secțiuni pentru a afișa imaginea originală și procesată.
* **img**: stilizarea imaginilor (mărimea maximă, colțuri rotunjite, umbră).
* **button**: stiluri pentru butoane, inclusiv schimbarea culorii la hover.
* **#timingContainer**: afișarea timpurilor de procesare.

3. Scriptul **JavaScript** se ocupă cu preluarea imaginii de la API, aplicarea transformărilor asupra imaginii și gestionarea evenimentelor. Codul include atât funcții sincrone, cât și asincrone pentru a manipula imaginea pas cu pas.

*Explicație detaliată a funcțiilor:*

1. **fetchDogImage**: preia o imagine aleatorie de pe API-ul DOG și imediat ce se obține răspunsul de la API, URL-ul imaginii este stocat într-o variabilă globală (imageURL). În cazul unei erori la obținerea imaginii, este afișat un mesaj de eroare în consolă.
2. **logarithmicGrayLevelTransform**: aplică o transformare logaritmică a nivelului de gri asupra imaginii; calculează valoarea medie a nivelului de gri folosind coeficienți pentru canalele de culoare RGB. Apoi, se aplică o transformare logaritmică asupra valorii medii a griului pentru a modifica intensitatea acesteia. Rezultatul este aplicat asupra fiecărui pixel din imagine.
3. **applyMirrorEffect**: creează un canvas temporar și aplică un efect de oglindire imaginii, adică parcurge fiecare coloană de pixeli și schimbă valorile pixelilor la stânga cu cei de la dreapta imaginii, efectiv oglindind imaginea pe axa verticală.
4. **generateImage:** generează imaginea prin apelarea funcției *fetchDogImage*. După ce imaginea este preluată, aceasta este afișată pe pagină prin actualizarea atributului *src* al elementului *img* cu id-ul *originalImage*.
5. **processImage**: parcurge etapele de procesare ale imaginii (oglindire și transformare logaritmică) și le aplică succesiv, adăugând un delay între fiecare pas. La final, imaginea procesată este afișată într-un container, iar timpii de procesare sunt afișați pe pagină.
6. **Event Listener** pentru buton: sunt asociate butoanelor de pe pagină. La apăsarea butonului „Generare Imagine”, se apelează funcția *generateImage*. La apăsarea butonului „Aplică Efecte”, se apelează funcția *processImage* pentru a aplica transformările asupra imaginii.

În implementare, am folosit ***setTimeout*** pentru a introduce întârzieri între etape, ceea ce face ca fiecare efect să fie aplicat secvențial cu o întârziere între ele. ***setTimeout*** este util pentru a simula întârzieri sau pentru a controla momentul în care se aplică un anumit efect. În contextul procesării imaginilor, o întârziere poate fi folosită pentru a oferi utilizatorului un timp vizibil de tranziție între diferitele etape.

Funcția *fetchDogImage* este o funcție asincronă care utilizează ***async*/*await*** pentru a aștepta răspunsul de la API-ul DOG și a obține URL-ul imaginii. Folosirea funcției ***await*** în proiectul meu asigură că funcțiile vor aștepta până când răspunsul de la *fetch*() și *response*.*json*() sunt complet procesate înainte de a trece la următoarea linie de cod. Acest lucru ajută la evitarea unor comportamente de tip "callback hell" și face codul mai ușor de urmărit și mai ușor de gestionat. Avantajele folosirii *async/await:*

* Pot permite să faci procesarea imaginii mai fluidă și mai organizată, fără a folosi multiple callback-uri sau a risca blocarea aplicației.
* Permit să aștepți ca fiecare etapă de procesare a imaginii să se termine înainte de a trece la următoarea, ceea ce îmbunătățește experiența utilizatorului.

*Bibliografie:*

* <https://www.tutorialspoint.com/dip/gray_level_transformations.htm>
* <https://www.w3schools.com/js/js_async.asp>
* <https://www.w3schools.com/js/js_asynchronous.asp>
* <https://stackoverflow.com/questions/30130149/processing-mirroring-or-flipping-an-image-without-affecting-others/30131192#30131192>