Une image contenant texte, capture d’écran, ordinateur, menu

Le contenu généré par l’IA peut être incorrect.

TP3 – Applications Web

**Document d’analyse Le TP3 : HTML/JS**

**Nom 01 :** Ameur Samy-Lyes

**DA 01 :** 2430934

**Nom 02 :** Arris Yanis

**DA 02 :**2447938

**420-2CW-BB APPLICATIONS WEB**

Dans ce projet, on a créé un site web interactif qui simule la gestion d’une base de données à l’aide de fichiers JSON. On utilise uniquement des technologies qu’on a vues en cours, comme HTML, CSS et JavaScript, sans aucun backend. Le but, c’est de faire communiquer les différentes pages du site avec ces fichiers grâce à JavaScript, et d’afficher les données de façon dynamique dans des tableaux clairs et bien présentés.

Le site est divisé en plusieurs pages : **Accueil**, **À propos**, **Catégories**, **Article**, et **S’inscrire**. Chaque page est structurée en HTML et stylisée avec CSS pour assurer une bonne présentation visuelle. Le JavaScript est utilisé pour ajouter de l’interactivité, notamment en permettant de charger dynamiquement le contenu des fichiers JSON dans les pages.

Par exemple, la page **apropos** lit et affiche automatiquement le contenu des fichiers **utilisateurs.json**, **categories.json** et **articles.json** sous forme de tableaux. Quant à la page **S'inscrire**, elle contient un formulaire d’ajout d’utilisateur. Lorsqu’un nouvel utilisateur est ajouté, ses informations sont enregistrées localement dans un tableau JavaScript, et un tableau dynamique s’affiche en bas de la page pour les visualiser.

Dans notre projet, on utilise uniquement des technologies front-end, comme celles qu’on a vues en cours. Le HTML et le CSS servent à structurer et styliser les pages, et on se sert du JavaScript pour gérer les événements, manipuler le DOM, charger les fichiers JSON avec fetch() et afficher les données dynamiquement dans des tableaux.

Pour simuler une base de données, on utilise trois fichiers JSON : **utilisateurs.json**, **categories.json** et **articles.json**. On les lit avec la méthode **fetch()** dans notre script JavaScript, puis on affiche leur contenu automatiquement dans la page **apropos**, sous forme de tableaux dynamiques.

Dans la page **S’inscrire**, on a mis en place un formulaire qui permet à un utilisateur d’entrer son nom, son prénom et son adresse courriel. Quand il clique sur “Valider”, on récupère les infos avec JavaScript, on les stocke localement dans un tableau (utilisateursAjoutes) et on les affiche directement en dessous dans la même page dans un tableau HTML.

On a bien organisé notre projet pour qu’il soit propre et facile à comprendre :

- Une page Index.html afin d’accéder directement.

- Le dossier /HTML contient toutes les pages du site (S’inscrire, Àpropos, etc.).

- Le dossier /CSS contient notre feuille de style principale Style.css.

- Le dossier /JS regroupe tous nos scripts JavaScript (chargement des données, interactions).

- Le dossier /JSON contient les fichiers de données utilisés dans le site.

- Et finalement, les images et les logos sont rangés dans /IMAGES.

Grâce à cette structure, on peut lancer le projet avec Live Server dans VS Code sans modifier quoi que ce soit. Tout fonctionne directement.

Pendant le projet, on a eu quelques défis techniques, mais on a réussi à les surmonter, et ça nous a permis d’apprendre plein de choses :

* On a appris à charger un fichier .json dans une page HTML avec fetch() et then(). C’est comme ça qu’on a compris comment fonctionne AJAX côté client.
* Le tableau s’affichait mais sans style. On a ajouté la classe CSS .styled-table et ça nous a permis de mieux comprendre le lien entre du HTML généré en JavaScript et les styles CSS.
* On a galéré un peu avec le formulaire d’inscription. Au début, rien ne s’affichait. Finalement, on a compris qu’il fallait stocker les données dans un tableau avec .push() et ensuite les réafficher dans le HTML avec innerHTML. On a aussi mieux compris comment utiliser le DOM.
* On avait des erreurs dans les chemins de fichiers. Du coup, on a tout réorganisé : les dossiers, les fichiers, les liens… Résultat : maintenant, tout est bien structuré et fonctionnel.

En conclusion, nous sommes fiers de présenter un projet abouti, conforme à toutes les attentes du TP3, tant sur le plan fonctionnel que pédagogique. Tout au long du développement de TechPulse, nous avons appliqué rigoureusement les notions enseignées en classe, en respectant les bonnes pratiques de structuration, de lisibilité du code, et d’interactivité en front-end.

L’architecture du site repose uniquement sur des technologies maîtrisées en cours (HTML, CSS, JavaScript), et démontre notre capacité à manipuler le DOM, à lire des fichiers JSON dynamiquement via fetch(), ainsi qu’à créer des interfaces utilisateurs claires et réactives.

Ce travail nous a permis de consolider nos apprentissages et de gagner en autonomie dans la conception d’interfaces web dynamiques et structurées.