Soutenance Projet 3 Parcours Développeur Web

Créez une page web dynamique avec JavaScript

Camille SIESSE-CUSCUSA

Date de début de projet : 03/07/23

OPENCLASSROOMS

Plan de soutenance

- 1. Présentation globale
- 2. Appel à l'API et récupérations de données
- 3. Fonctionnement des filtres de projets dans la galerie
- 4. Gestion de la connexion de l'utilisateur et différence login / log out
- 5. Fonctionnalité ajout de nouveau projet (et suppression projets existants)

1.Présentation Globale : le projet

- Projet

  → **Intégration de fonctionnalités et d'affichage d'une page web dynamique avec JavaScript**

  → Livrables : **deux fichiers HTML** qui codent la structure de base du site, **deux fichier JS** qui codent pour le contenu dynamique du site et un **dossier CSS** qui contient les **fichiers CSS** pour organiser le style de certains éléments du site

- Destinataires, interlocuteurs et utilisateurs

  Dans l'entreprise ArchiWebos : **Charlotte Chef de Projet , Fatima Dev Backend**

  L'utilisateurs : **usager** souhaitant en savoir plus sur les travaux de Sophie Bluel ou **Sophie Bluel** elle même voulant gérer ses projets sur son site.

- Site

  Deux pages : accueil et page de connexion

  Fonctionnalités principales : filtres de projets, connexion, ajout/suppression de nouveaux projets

1. Présentation Globale : organisation du code

Fichiers JS

→ **projects.js**

→ **login.js**

Fichiers HTML

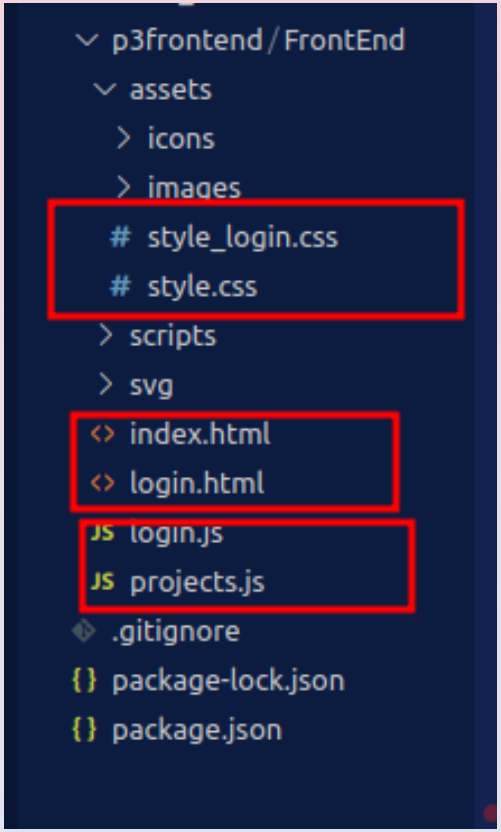
→ **index.html**

→ **login.html**

Fichiers CSS

→ **style.css**

→ **style_login.css**



1. Présentation Globale : organisation code projects.js

1 Importation et stockage des data (l1-10)

2 Constantes header (l12-23)

3 Galerie et génération dynamique des projets (l24-90)

4 **Filtres (l93 -148)**

5 Boucle **Login**/ Log-out **IF (token enregistré) : (l158-848)**

a- modifications graphiques de la page d'accueil (l158- 252)

b- modale 1 (galerie modale et suppression de projets) (253-433)

c- modale 2 (formulaire et ajout d'un nouveau projet + fonction de mise à jour de la page sans avoir besoin de réactualiser celle-ci) (433- 845)

SINON (si pas de token) : alors page d'accueil habituelle.

2. Appel à l'API et récupérations de données

Appel à l'API

API : SWAGGER

→ requête avec la fonction **fetch**

→ a laquelle on peut ajouter des infos sur la « **method** » (**get, post, delete, put**), sur le « **body** » de notre requête ou sur le contenu de son « **header** »

→ Voir utilisation dans le code pour le login, la suppression de projets ou l'ajout de nouveau projets

Récupération de données (+ galerie)

Stocker les réponses de l'API à nos requêtes dans des constantes

Traduire la réponse en un format qu'on peut lire et manipuler : **le JSON**

→ Voir utilisation dans le code pour l'importation des données utiles pour la création de la galerie de projets

→ Fonction générant les projet

3. Fonctionnement des filtres de projets dans la galerie

Général

l93-146

Deux type de « buttons »

Event listener

→Event listener qui, ici au « click »,

→ Instructions pour filtrer les projets et ne retourner que les projets dont l'ID de catégorie correspond bien à la catégorie sélectionnée

Chaque catégorie

Générer des boutons « Filtres » par catégorie automatiquement en parcourant la liste des différentes catégories

Sites

Voir fonctionnement sur le site de Sophie Bluel

```
93  // ----------------------------------Boutons Filtres
94
95  // Création du bouton "Tous" pour afficher tous les projets + ajout listener d'événement sur le bouton "Tous" pour afficher tous les projets
96
97  const buttonFilterAll = document.createElement("button");
98  buttonFilterAll.innerText = "Tous";
99  buttonFilterAll.classList.add("filters");
100 › buttonFilterAll.setAttribute(…
103 );
104 buttonFilters.appendChild(buttonFilterAll);
105 buttonFilterAll.addEventListener("click", function (e) {
106   e.preventDefault();
107   e.stopPropagation();
108   const piecesFiltres = projects.filter(function (project) {
109     return project;
110   });
111   document.querySelector(".gallery").innerHTML = "";
112   generateProjects(piecesFiltres);
113 });
114
115 // Création des boutons de filtre pour chaque catégorie
116
117 // Création d'un ensemble Set pour stocker les noms uniques des catégories
118 const categorySet = new Set(categories.map((category) => category.name));
119
120 // Parcours de chaque nom de catégorie dans l'ensemble Set et créations des buttons
121 categorySet.forEach((categoryName) => {
122   // création d'un bouton de filtre pour chaque catégorie
123   const buttonFilter = document.createElement("button");
124   buttonFilter.innerText = categoryName;
125   buttonFilter.classList.add("filters");
126 ›   buttonFilter.setAttribute(…
129   );
130
131   buttonFilters.appendChild(buttonFilter);
132
```

4. Gestion de la connexion de l'utilisateur et différence login / logout

Page login organisation code js

- Création formulaire
- Création bouton de connexion avec event-listener
- Envoi d'un requête au serveur
- Boucle gérant le succès ou l'échec de la connexion

→ Voir détails étape requête/ boucle login, stockage token et redirection

→ Changements graphiques si login dans la page d'accueil grâce à une boucle if/else

projet_developper_portefolio_architecte > p3frontend > FrontEnd > JS login.js > buttonConnexion.addEventListener('click') callback

You, 5 days ago | 1 author (You)

```
1  // page de connexion
2  // selection de la zone div de formulaire
3
4  const sectionLogIn = document.querySelector(".logIn");
5
6  // création du formulaire
7
8  const formulaire = document.createElement("form");
9  formulaire.innerText = "";
10 sectionLogIn.appendChild(formulaire);
11
12 const organisationForm = document.createElement("fieldset");
13 organisationForm.innerText = "";
14 formulaire.appendChild(organisationForm);
15
16 // titre formulaire
17
18 const legend = document.createElement("legend");
19 legend.innerText = "Log In";
20 organisationForm.appendChild(legend);
21
22 // zone label/ input email et mot de passe
23
24 const labelEmail = document.createElement("label");
25 labelEmail.innerText = "E-mail";
26 organisationForm.appendChild(labelEmail);
27
28 const email = document.createElement("input");
29 email.innerText = "E-mail";
30 organisationForm.appendChild(email);
31
32 const labelMdp = document.createElement("label");
33 labelMdp.innerText = "Mot de passe";
34 organisationForm.appendChild(labelMdp);
35
```

5. Fonctionnalité ajout de nouveau projet

Ajout de nouveau projet

l664-847

Event-Listener Bouton

Création d'une constante catId qui associe la catégorie sélectionnée par l'utilisateur à son Id

Vérification du formulaire avant de le soumettre

Création d'un objet form Data

Envoi de la requête avec en corps de requête l'objet form data

Appel de la fonction d'ajout de nouveau projet au DOM

→ Voir détails requête et création fonction d'ajout au DOM

```
// -------------------------------- Bouton de soumission du formulaire en envoyant une requete à l'API

const buttonValidationModal2 = document.createElement("button");
buttonValidationModal2.innerText = "Valider";
buttonValidationModal2.setAttribute(…
);
modalWrapper2.appendChild(buttonValidationModal2);

// Fonction ajout d'un nouveau projet

buttonValidationModal2.addEventListener("click", async function (e) {
  //gestion des erreurs
  e.preventDefault();
  e.stopPropagation();

  // stokage des data rentrées par l'utilisateur dans trois contantes (image, title, categorie)
  const image = buttonAjoutPhotoM2.files[0];
  const title = titlePhotoModal2.value;
  const category = categoryPhotoModal2.value;

  // création et initialisation d'une variable catId
  let catId = null;

  // associations noms des catagories dans select à id de la catégorie correspondant et stockage dans la viraible catId
  if (category === "Objets") {
    catId = 1;
  } else if (category === "Appartements") {
    catId = 2;
  } else if (category === "Hôtels et restaurants") {
    catId = 3;
  }
```

Bilan

Difficultés rencontrées, limites et faiblesses du projet

→ Difficultés majeures rencontrées : Requête multi CORS, Formats envoi/réponses requêtes, modification du DOM sans rechargements

→ Limite et faiblesse du code : non-optimisation du code (redondances possibles), pas assez d'automatisation

Forces et intérêt du projet

→ Bonne cohérence **maquette / livrables/ fonctionnalités**

→ Fonctionnalités efficientes

→ Quasiment tout le contenu de la page est en js (pas de CSS/ html) et donc dynamique

Bilan général

Projet très intéressant / **requêtes API** rôle centrale dans le projet / découverte fonctionnalité de **login** / découverte et approfondissement du langage **JavaScript (et du format JSON)** et de ses très nombreuses possibilités / utilisation de **Git et Github**