582-31F-MA Programmation d’interface Web 2

TP 2

Description

Réaliser un site Web utilisant des requêtes asynchrones et un gestionnaire d'interfaces.

Pondération

25%

Modalité particulière

Travail individuel

Sujet

Gestionnaire complet d'une liste de tâche (*to-do-list*) asynchrone

Date de remise

Cours 20 - lundi 18 mars 2024

Modalités de remise

- Le dossier zippé, à votre nom (nomdefamille-prenom) doit m'être remis sur Léa avant le 18 mars 23h59.
- Le site doit être en ligne avant le 18 mars 23h59.
- Dans le dossier de remise, vous devez inclure un fichier `.txt` avec l'*URL* de votre site.

Retard

Selon les règles du collège, 5 % par jour de retard seront enlevés, jusqu'à 5 jours de retard maximum.

Énoncé

Après avoir *pitché* votre projet de *to-do-list*, vous avez reçu du financement pour développer la phase 2 ! Celle-ci reprend les mêmes fonctionnalités, mais les données doivent être enregistrées et obtenues via un gestionnaire d'interfaces (*SPA*) qui adresse une communication client-serveur asynchrone.

Consignes

- Les fonctionnalités sont déjà développées (vous pouvez démarrer de ma version ou de la vôtre), mais vous devez les adapter pour enregistrer et obtenir les données des tâches via une base de données. Important : assurez-vous de purger tout code inutile, par exemple, toute référence au *array* `toDoList` doit être supprimée.
- Créez une base de données nommée `to-do-list`.
- Dans le dossier de base, vous avez le fichier `to-do-list.sql` pour populer la base de données nouvellement créée.
- Inspirez-vous des exemples vus en classe pour développer les instructions *PHP* adressant la base de données. Idéalement, un fichier communique avec la *db* et un second agit comme un 'contrôleur' appelé par les différents appels asynchrones.
- Au chargement de la page, vous devez afficher les tâches listées dans la base de données (sans appel asynchrone).
- Les scripts *JS* doivent être orientés objets de type module. Vous aurez donc à reconsidérer plusieurs algorithmes. Aussi, les fonctions fléchées (*arrow functions*) sont interdites.

Consignes (suite)

- Au clic du bouton ‘Ajouter’, la tâche saisie est ajoutée à la base de données de façon asynchrone (*AJAX* ou *Fetch*).
- Vous ne pouvez pas utiliser *jQuery*, et ce pour tout le projet.
- Placez le script de validation dans un fichier modulaire séparé. La validation demeure la même, soit le champ ‘Nouvelle tâche’ et l’importance (`input radio`) sont obligatoires.
- La nouvelle tâche est ajoutée à la liste des tâches dans le *DOM*.
- L’injection de chaque nouvelle tâche doit utiliser la mécanique d’engin de gabarit sur le clone d’un fragment `<template>`.
- Chaque tâche listée a comme comportements : un bouton pour afficher son détail et un bouton pour la supprimer.
- Ces deux comportements doivent être fonctionnels en tout temps sans rechargement de page pour toutes les tâches listées.
- Au clic du bouton ‘Supprimer‘ d’une tâche, faites un appel asynchrone (*AJAX* et/ou *Fetch*) pour supprimer cette tâche de la base de données et supprimez-la du *DOM*.
- Au clic du bouton ‘Afficher le détail’ d’une tâche, faites un appel asynchrone (*AJAX* et/ou *Fetch*) suite à l’événement `hashchange` pour récupérer les données de cette tâche.
- Vous devez faire la gestion du système de routage à l’intérieur d’un fichier `Router.js`. Il y a deux routes, soit l’accueil (notez que le clic du titre de la page (`<h1><a href="#!">TP2</a></h1>`) ramène à l’état accueil) et le détail d’une tâche.

Consignes (suite)

- L'injection du détail d'une tâche doit utiliser la mécanique d'engin de gabarit sur le clone d'un fragment `<template>`.
- Si la tâche n'a pas de description, injectez la chaîne : 'Aucune description disponible'.
- Si un usager saisi directement l'*URL* avec un identifiant de tâche valide, le détail de cette tâche est affiché.
- À l'affichage du détail d'une tâche (au chargement de la page lorsque l'identifiant de tâche est valide ou au clic d'un bouton 'Afficher le détail'), faites descendre (*behavior smooth*) la fenêtre du navigateur à la hauteur de la section correspondante.
- S'il n'y a pas de données suite à la requête d'une tâche, injectez la chaîne : 'Cette tâche n'existe pas.' dans la zone 'Détail d'une tâche'.
- Si la route n'existe pas, le cas erreur 404 est injecté. Celui-ci offre un lien vers l'accueil (`<a href=".">Retour à l'accueil</a>`).
- Au clic des boutons 'Trier par ordre alphabétique' et 'Trier par importance', la liste des tâches triées doit être entièrement réinjecté suite à un appel asynchrone. Pour ce faire, la requête *SQL* utilise la clause `ORDER BY`.
- La réinjection de la liste réordonnée des tâches doit utiliser, pour chaque tâche, la mécanique d'engin de gabarit sur le clone d'un fragment `<template>`.
- Placez votre site en ligne (*WebDev* ou autre), notez que vous aurez à redéfinir quelques informations de connexion au serveur.

Plus-value (nice- to-haves)

- Placer son projet sur *GitHub*.
- Implémenter la mécanique d'instanciation de classes comportementales *DOM* via l'attribut `data-js-component`.
- Utiliser entièrement l'*API* `History` (méthodes `pushState()` et `replaceState()`) pour la mécanique de *routing* plutôt que la stratégie *hashbang*.
- Récupérer les différents éléments `<template>` de façon asynchrone, chacun placé à l'intérieur d'un fichier de 'vue' partiel.
- Factoriser le *PHP* en orienté objet.

Exigences techniques et critères d'évaluation

- Réussite des différentes fonctionnalités
- Utilisation de classe *ES6* de type *Module*
- Structure et optimisation du code
- Qualité des algorithmes et qualité du code source
- Utilisation du préfixe # pour toutes propriétés et méthodes privées
- Absence de fonctions fléchées (*arrow functions*)
- Aucune librairie externe
- Code sémantique
- Commentaires fréquents et pertinents
- Le site est en ligne

Plagiat

Nous travaillons dans le Web, alors les réponses s'y trouvent et il va de soi que vous allez devoir chercher sur *Google*. Dans l'esprit *open-source* du Web, vous avez tout à fait le droit de copier-coller un bout de code que vous avez trouvé. Cela-dit, vous devez citer les extraits de code qui dépassent une ligne ou deux. Évidemment, générer du code via un outil d'intelligence artificielle tel que *ChatGPT* sera considéré comme du plagiat. Pour prévenir le copier-coller direct d'Internet, aucune syntaxe autre que celles vues en classe ne sera acceptée.

Il s'agit d'un travail strictement individuel. Vous pouvez vous aider quelque peu entre vous, mais faites attention à ne pas partager ou récupérer des fonctionnalités complètes. Je rappelle encore une fois qu'un des objectifs principaux de la formation est de développer votre autonomie, c'est important. En cas de plagiat, je devrai appliquer les sanctions de la *PIEA* pour les étudiant.e.s ayant reçu et partagé le code.

Si vous êtes vraiment bloqué, contactez-moi.