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 - mercredi 24 août 2022

Modalités de remise

- Le dossier zippé, à votre nom (nomdefamille-prenom) doit m'être remis sur Léa avant le 24 août 23h59.
- Le site doit être en ligne avant le 24 août 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 et vous êtes rendu à 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 ajouter la table `taches` dans la *db* nouvellement créée.
- Inspirez-vous des exemples vus en classes pour développer vos instructions *PHP* pour adresser 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 certains algorithmes. Aussi, vous ne devez pas utiliser de fonctions fléchées (*arrow functions*).

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.
- Profitez-en pour placer 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.
- L’affichage de chaque nouvelle tâche doit être l’instance d’un fragment `<template>`.
- Chaque tâche listée a comme comportement : un bouton pour afficher son détail et un bouton pour la supprimer.
- 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 supprimer-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 et les afficher via l’instance d’un fragment `<template>`. Notez que, si la tâche n’a pas de description, affichez : ‘Aucune description disponible’.
- Si un usager saisi directement l’*URL* avec un identifiant de tâche valide, cette tâche a son détail affiché. Toutefois, si l’identifiant n’existe pas, la section détail est vidée (si celle-ci affichait le détail d’une tâche) et l’*URL* est nettoyée (`replaceState`).

Consignes (suite)

- Au clic des boutons ‘Trier par ordre alphabétique‘ et ‘Trier par importance’, vous devrez toujours faire le tri des tâches, mais celles-ci seront `order by` côté *PHP* lors de l’appel asynchrone puis réinjectés dans le *DOM* à l’aide du fragment `<template>` déjà défini précédemment pour l’ajout d’une nouvelle tâche.

Dossier à remettre

Le dossier Web complet avec - bien sûr - les fichiers *JavaScript*, mais aussi tous les fichiers *PHP* nécessaires ainsi que les styles (vous pouvez récupérer les miens). À la racine du dossier, créez un fichier `readme.txt` et écrivez l’*URL* par lequel je pourrai accéder à votre site.

Critères d’évaluation

- Fonctionnement conforme aux exigences
- Réussite des différentes fonctionnalités
- *HTML* sémantique
- Utilisation de la programmation orientée objet de type *Module*
- Absence de fonctions fléchées (*arrow functions*)
- Qualité des algorithmes
- Qualité du code source
- Clarté, organisation et commentaires du code
- 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. Pour les extraits de code qui dépasse une ligne ou deux, prenez tout de même l'initiative de placer en commentaire l'*URL* de votre source.

Il s'agit d'un travail strictement individuel. Vous pouvez vous aider quelque peu entre vous tous, mais faites attention à ne pas partager ou récupérer des fonctionnalités complètes, ce n'est pas vous aider. En cas de plagiat, je devrai faire le suivi conséquent. Je rappelle encore une fois qu'un des objectifs principaux de la formation est de développer votre autonomie, c'est important.

Si vous êtes vraiment bloqué, contactez-moi.