582-21F-MA Programmation d’interface Web 1

TP 2

Description

Travail pratique permettant l'intégration des éléments d'apprentissage.

Pondération

25%

Modalité particulière

Travail individuel

Sujet

Gestionnaire complet de liste de tâche (*to-do-list*)

Date de remise

Cours 20 - 31 octobre 2022

Modalités de remise

- Le dossier zippé du projet, à votre nom (nomdefamille-prenom) doit m'être remis sur Léa avant le 31 octobre 2022 23h59.
- Le site doit être en ligne avant le 31 octobre 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

La commande consiste à créer une page Web (une seule page *HTML*) qui permet à un usager de faire la gestion d'une liste de tâches à accomplir (*to-do-list*).

Comme pour le TP1, ce n'est pas un travail de rendu de maquette. Vous aurez évidemment quelques définitions *CSS* à écrire pour réaliser les comportements demandés, mais la mise en page est minimale et le *responsive* ne sera pas évalué. Assurez-vous toutefois de soigner votre structure *CSS* et de ne pas faire d'erreurs.

Vous devez faire ce projet en programmation orientée objet. Notez que les fonctions fléchées et les librairies externes (par exemple *jQuery*) sont interdites. Vous êtes responsable de l'organisation de vos scripts. Je vous invite à suivre les consignes, celles-ci donnent plusieurs stratégies de développement.

Consultez les références données pour les contenus textuels des titres et intitulés.

Consignes

Créer le *DOM* statique

- La première image donnée en référence illustre la structure *HTML* statique du projet. Après l'habituel `<header>`, il y a trois `<section>`, soit :
  - un formulaire d'ajout de tâche ;
  - la liste des tâches ajoutées ;
  - le détail d'une tâche.

Consignes (suite)

Ajouter une tâche

- Le formulaire pour ajouter une tâche a :
  - un `input` de type texte (obligatoire) pour le nom de la tâche ;
  - un `input` de type texte pour la description de la tâche ;
  - trois `input` de type radio pour donner une importance
  - (obligatoire) à la tâche (Haute, Moyenne ou Basse).
- Au chargement de la page, instanciez la classe *ES6* nommée (par exemple) `Form` - en prenant soin de passer son propre élément *DOM* en argument - afin de faire la gestion du formulaire.
- Au clic du bouton, faites d'abord la validation des deux contrôles obligatoires (Nom et Importance).
- Si le formulaire est valide, gardez en mémoire les valeurs saisies. Une bonne stratégie est de les placer à l'intérieur d'un objet littéral, que vous ajoutez ensuite comme dernier index d'un tableau. Exemple :

```
let object = {
    tache:              // valeur du champ 'Nouvelle tâche'
    description:        // valeur du champ 'Description'
    importance:         // valeur du champ radio coché
}
toDoList.push(object);
```

- Pour avoir un accès global dans tout le projet au tableau `toDoList`, celui-ci est déclaré dans le premier fichier *JS* chargé dans le `<head>`. Exemple :

```
let toDoList = [];
```

Consignes (suite)

- Toujours si le formulaire est valide, assurez-vous de nettoyer tous ses champs afin que l'usager puisse saisir une nouvelle tâche sans avoir à effacer la précédente.
- Ensuite, vous pourrez injecter dynamiquement la nouvelle tâche, son importance et ses deux boutons dans la section 'Liste des tâches'. Aidez-vous en injectant son index dans un attribut `data`.
- La beauté de *ES6* est que vous pouvez faire l'instanciation de nouveaux objets à n'importe quel moment. Ainsi, après avoir injecté la tâche, vous pourrez récupérer son nœud *DOM* afin d'instancier une classe *ES6* nommée (par exemple) `Task`, ceci toujours en prenant soin de passer son propre élément *DOM* en paramètre. Cette classe prendra en charge les comportements et fonctionnalités associés à chaque tâche.

Gérer chaque tâche

- La classe `Task` instanciée pour chaque tâche a essentiellement deux fonctionnalités :
  - afficher le détail d'une tâche ;
  - supprimer une tâche.
- Au clic du bouton 'Afficher le détail' d'une tâche, vous devez injecter dynamiquement les détails de cette tâche dans la section 'Détail d'une tâche'.
- À cette étape, rappelez-vous que le champ description n'était pas obligatoire, si cette valeur dans son objet littéral du tableau `toDoList` est vide, affichez : 'Aucune description disponible.'

Consignes (suite)

- Au clic du bouton ‘Supprimer‘ d’une tâche, vous devrez la supprimer de la liste. Pour assurer la séquence d’affichage des tâches avec le tableau d’objets littéraux `toDoList`, il faudra supprimer la tâche du tableau `toDoList` puis réinjecter dans le *DOM* toutes les tâches toujours présentes. N’oubliez pas que c’est du nouveau *DOM*, il faudra alors ré-instancier la classe `Task` pour chaque tâche réinjectée.

Filtrer les tâches

- La section ‘Liste des tâches‘ a deux boutons pour les filtrer :
  - par ordre alphabétique ;
  - par ordre d’importance.
- Pour cette fonctionnalité, je vous suggère de créer une nouvelle classe nommée (par exemple) `SortTasks` en prenant soin de passer son propre élément *DOM* en paramètre, celle-ci instanciée au chargement de la page (en même temps que la classe `Form`).
- Nous avons vu ensemble la méthode `sort()` sur les tableaux, mais voici une référence qui vous aidera :
  Sort array by firstname (alphabetically) in Javascript [duplicate]
- Comme pour la suppression d’une tâche, il vous faudra réinjecter dans le *DOM* toutes les tâches, mais suivant le nouvel ordre du tableau. Encore une fois, n’oubliez pas que c’est du nouveau *DOM*, il faudra alors ré-instancier la classe `Task` pour chaque tâche réinjectée.

Bonis

- Deux points bonis sont offerts pour l'animation en accordéon de la section affichant le détail de la tâche. Je rappelle qu'il est interdit d'utiliser *jQuery*.
- Pour le chevron, je vous invite à dessiner en *CSS* ce genre d'icône simple et commun, ça évite d'importer un *SVG*.

Exigences techniques et critères d'évaluation

- Fonctionnement conforme aux exigences
- Réussite des différentes fonctionnalités
- Réussite de la fonctionnalité d'ajout d'une tâche (validée) une semaine après la remise de l'énoncé
- *HTML* sémantique
- Utilisation de la programmation orientée objet
- 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. Cela-dit, vous devez citer les extraits de code qui dépassent une ligne ou deux et suivre un tutoriel (en tout ou en partie) 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 tous, mais faites attention à ne pas partager ou récupérer des fonctionnalités complètes, ce n'est pas vous aider. 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*.

Si vous êtes vraiment bloqué, contactez-moi.