PLATYPUS

TP 3
EFREI 2A WEB

Louis Cherel - 05/2019

Préambule

Merci de lire **toutes les consignes**, elles sont écrites pour être lues.

Les TP sont tous ramassés

Tous les TP sont ramassés et doivent être rendus au plus tard à 23h59 le soir de la séance sur le **formulaire de dépôt** prévu à cet effet.
Chaque jour de retard vous retire 1 point sur votre note de TP, avec une limite à 3 jours de retard, ce qui, passé ce délai, vous vaudra une notre de 0 sur votre TP.

Seul un des TPs sera noté, mais ne pas rendre les deux autres vous vaudra quand même 0.

Les TP et le projet sont à réaliser en binôme ou trinômes, et doivent être enregistrés sur le **formulaire d'enregistrement d'équipes** avant la fin de la première séance.
Sans votre enregistrement d'équipe, vous ne pourrez pas rendre votre travail.

Il n'y a pas de DE. Votre note de module sera constituée à 50% de la note de TP et 50% de la note de projet.

Cf les consignes du TP 1 pour le but et le séquencement des TP.

Les ressources conseillées

Vos armes seront donc le Mozilla Developer Network (MDN), Stack Overflow, et les recherches en anglais. **Évitez au maximum** les forums (pas les cours) des sites comme OpenClassrooms ou Comment Ça Marche, qui proposent des solutions souvent peu fiables. W3Schools peut être une bonne ressource, mais les bonnes pratiques et les standards ne sont pas leur fort. Leur préférer le MDN lorsque possible.

Quantité de blogs sont très bons, notamment alsacreations (en français).

Lectures préliminaires

Les lectures **obligatoires** avant de commencer le TP:

- Qu'est-ce que le JavaScript ? (MDN)
- Comment ajouter du JavaScript à votre page ? (MDN)
- Notre premier code JavaScript (MDN)

Certains exercices nécessitent des lectures annexes, prenez soin de les lire avant de commencer le TP, même si les questions vous semblent simples.

Exercices

Partez de votre code du TP précédent.

Exercice 1 (🕒10 min lecture + 🕒 30 min pratique)

Lectures préalables nécessaires:

- setInterval (MDN)
- Date.now() (MDN)

1) Reprenez votre code du TP précédent.
2) Faites en sorte que l'utilisateur ne puisse valider le formulaire que 10 secondes après l'ouverture de la page ou 10 secondes après la dernière validation du formulaire
3) Affichez le décompte dans le bouton "Ajouter" entre parenthèses, il doit s'actualiser toutes les secondes
4) Faites en sorte que le bouton aie l'air "désactivé" lorsque l'utilisateur doit attendre
5) Si l'utilisateur clique avant les dix secondes imparties, utilisez la fonction alert() afin de l'avertir de son mauvais comportement

Exercice 2 (🕒 30 min pratique)

Nous allons ajouter une nouvelle section à notre page. Cette section va contenir une liste de tâches, récupérées depuis internet. Il va donc nous falloir un menu pour indiquer quelle section doit être affichée.

1) Entre le logo et le formulaire de votre page, ajoutez un menu de navigation à l'aide de la balise <nav>. Ce menu doit contenir deux boutons, "Utilisateurs" et "Tâches".

   À vous de définir leur style, voici une suggestion:

2) Créez un nouveau tableau HTML (qui nous servira pour les tâches) qui réutilise le style que vous aviez défini dans le TP précédent (à l'aide d'une classe CSS), mais qui comprend les colonnes suivantes: "USER ID", "ID", "TITRE", "TERMINÉ"

3) Au-dessus de ce tableau, ajoutez un titre "Tâches". Ajoutez également un titre "Utilisateur" au-dessus de votre tableau précédent, afin de bien les distinguer

4) Concevez deux fonctions javascript showUsers() et showTasks()

5) showUsers() doit masquer le tableau HTML de tâches, et afficher le tableau HTML qui liste les utilisateurs ainsi que le formulaire (celui du TP précédent)

   Tip: ajoutez un ID à chaque tableau et au formulaire, pour pouvoir récupérer aisément sa référence grâce à document.querySelector("#id-du-tableau")

   sachez également que la propriété <elem>.style.display permet de définir la façon d'afficher un élément via le JS. display: none vous sera probablement utile

6) showTasks() doit évidement faire l'inverse

7) Appelez showTasks() au chargement de votre page, et vérifiez que son comportement est celui que vous attendiez. Faites ensuite la même chose avec showUsers(), en enlevant évidement l'appel à showTasks()

8) Faites en sorte que lorsque l'utilisateur clique sur le bouton "Utilisateurs", cela active la fonction showUsers(), et "Tasks" => showTask

Exercice 3 (🕒 20 min lecture + 🕒 1h pratique)

L'objectif de cet exercice est de remplir votre tableau de tâches depuis une source externe à votre page. Il vous faudra donc réutiliser la logique que vous avez appliqué au tableau précédent.

Lecture préalable nécessaire:

- Using Fetch (MDN) (**en**, la version française est obsolète)
- Array (MDN)

Code à copier-coller (Q1)

```
fetch('https://jsonplaceholder.typicode.com/todos')
.then(response => response.json())
.then(function (data) {
   console.log('data', data)
   // votre code ici
})
```

1) Pour récupérer des données depuis un serveur externe, nous allons utiliser une technologie dénommée AJAX. Concrètement, cela consiste à récupérer les données grâce au javascript, puis à les manipuler, les afficher, toujours grâce au javascript. La façon la plus simple de récupérer quelque chose depuis une URL donnée, est la fonction **fetch()** . Créez une fonction getTasks() et copiez-collez le code ci-dessus dedans

2) Ouvrez la **Console**. Appelez votre fonction getTasks() au chargement de votre page, et regardez ce qui finit par s'afficher dans votre console. Juste avant le console.log() du code ci-dessus, **ajoutez un commentaire qui explique ce que vous avez constaté**

3) Créez une fonction createTask(userId, id, title, completed) qui ajoute une ligne à votre tableau de Tâches, comme vous l'aviez fait pour le tableau du TP2. **Note**: le fait de ne pas utiliser innerHTML ici est primordial pour des raisons de sécurité

4) Appelez votre fonction createTask() avec des valeurs arbitraires au chargement de votre page, pour vérifier qu'elle fonctionne correctement. Vous pourrez retirer cet appel lorsque votre fonction sera correcte

5) Utilisez l'objet data, que vous avez reçu de l'API, afin d'appeler createTask() autant de fois qu'il y a d'éléments dans le tableau Javascript reçu

Exercice 4 (🕒 1h pratique)

1) 200 éléments dans le même tableau, ça fait beaucoup ! Trouvez un moyen de "paginer tout cela", à savoir d'afficher seulement 10 éléments à la fois, tout en permettant à l'utilisateur de naviguer dans la liste
    a) Vous aurez besoin de conserver une variable qui contient la position actuelle
    b) Vous aurez également besoin vider le contenu de votre tableau lorsqu'on change de page, mais vous avez déjà une fonction pour cela, réadaptez-la pour que cela fonctionne sur votre nouveau tableau

Pour aller plus loin (Bonus)

Exercice 5 (🕒 >10 minutes)

1) Faites en sorte que les données de votre onglet Tâches ne soient téléchargées que lorsque l'utilisateur affiche l'onglet, sinon des données inutiles sont téléchargées si l'utilisateur ne va jamais sur cette page

Exercice 6 (🕒 >30 minutes)

1) l'API https://jsonplaceholder.typicode.com/users permet de récupérer les informations des utilisateurs
2) Faites en sorte, dans la liste des tâches, de remplacer l'id de l'utilisateur par son nom, qui est fourni par l'api ci-dessus. (nécessite un nouvel appel à la fonction fetch())