ENONCE – EXERCICES JAVASCRIPT

Objectif :

L'objectif est de réaliser une page web dynamique permettant d'afficher des albums d'artistes en fonction du nombre de vue/d'écoute. Les données qui serviront à tester notre logique sont fictives et seront issues depuis un site qui s'appelle **Jsonplaceholder**. Cette technique qui consiste à récupérer temporairement des données venant d'un système qui simule un comportement se nomme par le terme « franglais » **mocker**.
Nous utiliserons différents outils JavaScript pour afficher notre page en manipulant successivement des données en **Json**, sous forme de **String**, dans des **Array**. Nous utiliserons le **DOM** pour récupérer les **event** et proposer l'affichage correspondant.

Ressources indispensables :

- *https://developer.mozilla.org/en-US/docs/Web/JavaScript*
- *https://www.w3schools.com/js/default.asp*
- *https://quickref.me/javascript*

Etape 1 : Une jolie collection

Un dépôt du projet est à récupérer sur Github. Vous pouvez récupérer le squelette du code avec la commande **git clone https://github.com/laurentgiustignano/exercicesjavascript** dans votre répertoire de travail. Ou bien depuis **PhpStorm**, en choisissant : **Get from Version Control** avec la même URL.

Le projet comporte 4 fichiers :

- **index.html** qui correspond à la page d'accueil de notre site.
- **style.css** permet d'améliorer le rendu de la page **index.html**
- **functions/api.js** qui contient une fonction permettant d'aller chercher des données sur **Jsonplaceholder** et de les proposer au format **Json** dans le fichier **app.js**
- **app.js** qui contiendra le code JavaScript de notre

Remarque : les trois premiers fichiers n'ont pas besoin d'être modifié. L'ensemble de votre travail devra se trouver dans le fichier **app.js**.

En chargeant la page **index.html** dans votre navigateur, le message « Chargement en cours… » devrait disparaître et laisser visible 3 boutons permettant de manipuler les données. Sinon, à sa place, sera visible un message en rouge « Impossible de contacter la ressource distance ».

- En premier lieu, glissez une erreur dans l'URL de l'appel de la fonction **fetchJson()** à la ligne 9, en enlevant ou rajoutant une lettre par exemple. Si le message rouge s'affiche, cela signifie que le traitement fournis est correct. Vous pouvez remettre l'URL d'origine.

```
8   try {
9       const albums = await fetchJson( url: 'https://jsonplace.code.com/albums')
10      console.log(albums)
11      // À compléter à partir d'ici
```

Figure 1- Exemple d'URL erronée pour afficher le message d'alerte

- Un appel de fonction du DOM **console.log()** permet d'afficher une message dans la console de l'interpréteur JavaScript. Ici, il s'agit de votre navigateur. Faites un click-droit dans la fenetre du navigateur et choisissez « inspecter ». Une fois la fenêtre ouverte, allez jusqu'à l'onglet console.
- Un objet complexe est visible. Vous pouvez le déplier pour observer son contenu. Déterminez son type.

- Maintenant que nous confirmons la présence des données, c'est qu'elles sont chargées. Le message du **loader** « Chargement en cours… » n'est plus nécessaire, nous allons le supprimer avec la fonction **remove()** appartenant au DOM.

Etape 2 – C'est vu !

La constante **albums** contient des données représentant des albums avec un **userId** qui pourrait représenter l'auteur, un **title** pour le titre de l'album, ainsi qu'un **id** pour différentier tous les éléments de manière unique. Il nous faudrait rajouter un champ à chaque objet qui contiendrait le nombre de vue (ou d'écoute) pour chacun des albums. Pour cette fois-ci, nous allons tricher un peu en générant de manière aléatoire un nombre de vues compris entre 100 et 10000, mais qui augmente de 100 à chaque palier (100, 200, 300, …, 9800, 9900, 10000). Ce nombre sera ajouté dans le nouveau champs vue de chaque album.

- Trouver l'expression permettant de générer un nombre entier dans l'intervalle demandé. Les fonctions **Math.floor()** et **Math.random()** sont nécessaire.
- Complétez la fonction **ajouterVues()** qui s'occupera de rajouter le champs vue à chaque élément de albums. Effectuez son appel et enregistrez le retour dans la variable **albumsAvecVues**.

Conseil : pour vous aider, la fonction **map()** *des objets* **Array** *permet de parcourir l'ensemble d'un tableau et pour chacun de ses éléments d'y exécuter un traitement. Cette fonction ne va pas modifier le tableau d'origine, mais retournera un nouveau tableau, de même dimension, avec les modifications. L'autre outil à utiliser est le* **spread operator** *(…). Cet opérateur permet de désigner tous les éléments d'un itérable, tout en les traitants de manière individuelle.*

```
const lettres = ['a', 'e', 'i']
const voyelles = [...lettres, 'o', 'u', 'y']
console.log(voyelles)   // affiche ['a', 'e', 'i', 'o', 'u', 'y']
```

Figure 2- Exemple d'utilisation du Spread Operator

Étape 3 – « attrapez les tous ! »

Maintenant, nous allons afficher tous les albums dans une structure **ul/li**. Cet affichage dois se déclencher lorsque l'utilisateur appuis sur le bouton contenant la chaine « **tous les titres** ». Il faut relier l'évènement du clic sur ce bouton vers la fonction **tous()**. Ensuite, il faut créer l'élément **ul**, et autant d'élément **li** que d'albums contenu dans **albumsAvecVues**. Ne pas oublier que pour chaque nouvelle référence de **HTMLElement**, il faut l'associer au DOM avec la fonction **prepend()** ou **append()**, si nous voulons les insérer, respectivement, au début ou à la fin de l'élément associé.

- A l'aide de la fonction **addEventListener()**, effectuez la liaison entre le bouton et la fonction **tous()**.
- A l'aide de la fonction **document.createElement()**, ajouter un élément **ul** ainsi que plusieurs **li** pour vérifier le comportement.
- A l'aide de la fonction **Object.entries()**, parcourrez **albumsAvecVues** pour récupérer notamment les valeurs et insérez dans chaque **li** le titre et le nombre de vues.

Étape 4 – « Simply the best ! »

A présent, nous travaillerons sur le bouton **BestOf** qui nous affichera uniquement les albums qui ont plus de 9000 vues. Le traitement est semblable à la fonction **tous()**, à la différence qu'il ne doit contenir qu'une sélection de **albumsAvecVues**.

- Reliez le bouton « Le Best Of » à la fonction **bestOf()**, puis à l'aide de la fonction **filter()**, ne conserver que les éléments respectant la condition du nombre de vues désirée.

Étape 5 – « Il n'en restera … que 3 »

Pour terminer, le bouton **top 3** doit présenter uniquement les 3 albums qui ont le plus de vues. Il est évident de penser que nous devons effectuer en premier lieu un tri sur l'ensemble des albums, puis en conserver que les 3

premiers. Deux fonctions vont vous aider : **sort()** qui permet de trier un ensemble, et **slice()** qui permet de couper un ensemble.

Conseil : lisez attentivement la documentation pour la fonction **sort()**. *Son comportement diffère des autres fonctions que vous avez observés sur les tableaux. On dit qu'elle a un traitement «* **in place** *».*

- Reliez le bouton « Top 3 » à la fonction **top3()**, puis compléter cette fonction pour qu'elle affiche uniquement les « 3 meilleurs albums ».

Bon courage…