R4.A.10 Complément Web Partie 1

Avant-propos

Ce projet va vous permettre de mettre en œuvre, en 2 étapes, les concepts vus en TD et dans les premières séances de TP.

L'objectif de ce projet est de construire une WebApp qui affichera des informations sur les *Pays du Monde*, et permettra de rechercher, trier, filtrer les pays et leurs informations en fonction de divers critères qui vous seront détaillés ultérieurement.

Une *WebApp* est une page Web qui se comporte comme une application traditionnelle, à savoir un rafraîchissement dynamique du contenu affiché sans faire appel à des chargements de nouvelles pages HTML. Une telle application porte aussi le nom d'*Application Web Monopage* (Single Page App). Vous aurez à manipuler le DOM.

Les données qui serviront à remplir votre application proviendront d'une API WebService.

Consignes importantes

1) Vous coderez en **JS** et vous pourrez utiliser **JQuery 3.x** que vous devrez télécharger si besoin (pas de CDN). Pas d'autre librairie, ni framework.
2) Vous mettrez vos pages en forme avec du **CSS** et, si vous le souhaitez, avec du **SASS**. Pas de framework CSS (Bootstrap ou autre).
3) Vos pages devront être **responsives** Desktop vs Mobile.
4) Vous devrez nommer vos scripts, vos classes, etc. comme indiqué dans le sujet.
5) Vous devrez rendre un détail de la participation de chaque membre en pourcentage d'implication sur chaque partie.
6) Utilisez **`console.table()`** au lieu de **`console.log()`** pour afficher les objets.

Le non respect des consignes sera pris en compte dans la note finale (points en moins, non correction, etc.)

Introduction

Dans cette partie, vous allez vous focaliser sur la lecture des données directement depuis un fichier contenant une structure JSON.

Le fichier de données

Récupérez le fichier **`countries.js`** qui accompagne le sujet de ce projet.

Ce fichier contient 248 pays et beaucoup de données pour chacun d'entre eux. Analysez la structure JSON d'un pays (pour vous aider à localiser les données : le 1er est Afghanistan, le 2ème est Albanie...).

Nous n'allons pas utiliser toutes les informations issues de ce fichier. Voici celles qui nous intéressent et que vous devez prendre en compte (à vous d'identifier les noms des champs) :

- Code Alpha 3
- Nom (en français, anglais, allemand, espagnol et italien), attention il y a un piège !
- Capitale
- Continent
- Population
- Superficie
- Tableau des codes Alpha 3 des pays frontaliers, attention il y a des pièges (cherchez Tuvalu)
- Gentilé
- Tableau des codes des monnaies
- Tableau des codes iso639_2 des langues
- Tableau des Top Level Domains (Internet)
- Drapeau (celui au format SVG). Attention, ce ne sera pas l'image du drapeau qui sera stockée dans l'objet mais une information permettant une affichage du drapeau (de type URL ou partie d'URL par exemple)

La classe *Currency*

Vous allez d'abord traiter les monnaies.

Q1 - A partir de votre analyse de la structure JSON d'une monnaie, créez une classe **`Currency`** pour y stocker vos données de travail. Prévoyez une fonction **`toString()`** synthétique qui retourne une chaîne (sans **`\n`**) contenant :

- Le code
- Le nom (anglais)
- Le symbole

Exemple à respecter :

`EUR, Euro, €`

Q2 - Écrivez une fonction **`fill_currencies()`** qui, à partir de la source de données, crée des objets **`Currency`** que vous stockez dans un tableau associatif (variable de classe) nommé **`all_currencies`** dont les clés sont les valeurs trouvées dans le champ **`code`** de la propriété **`currencies`**, et les valeurs sont des objets **`Currency`**.

La classe *Language*

Vous allez maintenant traiter les langues, de façon similaire aux monnaies.

On s'intéresse au code **`iso639_2`** et au nom (en anglais).

Q1 - A partir de votre analyse de la structure JSON d'une langue, créez une classe **`Language`** pour y stocker vos données de travail. Prévoyez une fonction **`toString()`** synthétique qui retourne une chaîne (sans **`\n`**) contenant :

- Le code iso639_2
- Le nom (anglais)

Exemple à respecter :

`French (fra)`

Q2 - Écrivez une fonction **`fill_languages()`** qui, à partir de la source de données, crée des objets **`Language`** que vous stockez dans un tableau associatif (variable de classe) nommé **`all_languages`** dont les clés sont les valeurs trouvées dans le champ **`iso639_2`** de la propriété **`languages`**, et les valeurs sont des objets **`Language`**.

La classe *Country*

Q1 - A partir de votre analyse de la structure JSON d'un pays, créez une classe **`Country`** pour y stocker vos données de travail. Prévoyez une fonction **`toString()`** synthétique qui retourne une chaîne (sans **`\n`**) contenant :

- Le code Alpha 3
- Le nom en français
- La capitale
- Le continent
- La population
- La liste des noms des pays voisins

Exemple à respecter :

`AND, Andorre, Andorra la Vella, Europe, 77 265 hab, (Espagne, France)`

Q2 - Ecrivez une fonction **`fill_countries()`** qui lit la source de données et crée des objets **`Country`** que vous stockez dans un tableau associatif (variable de classe) nommé

`all_countries` dont les clés sont les valeurs trouvées dans le champ **`alpha3Code`**, et les valeurs sont des objets **`Country`**.

Q3 - Complétez votre classe **`Country`** avec les méthodes suivantes :

- **`getPopDensity()`** : retourne la densité de population du pays (hab. / Km^2)
- **`getBorders()`** : retourne un tableau JS des pays frontaliers (les objets **`Country`**, pas les codes).
- **`getCurrencies()`** : retourne un tableau des monnaies (objets **`Currency`**)
- **`getLanguages()`** : un tableau des langues (objets **`Language`**)

Tests

Écrivez des fonctions permettant de répondre aux questions suivantes. Les résultats doivent s'afficher dans la console du navigateur à l'aide de **`console.table()`** si ça s'y prête bien (affichage de tableaux par exemple) ou de **`console.log()`** sinon, et rien d'autre !

Vous devez regroupez, dans le fichier **`test.js`**, les fonctions suivantes.
Vous respecterez les noms des fonctions.Vous serez évalués sur le respect de ces consignes !

Q1 - **`outsideTheContinent()`** : Tableau JS des pays (objets **`Country`**) dont au moins un pays frontalier n'est pas dans le même continent.

Q2 - **`moreNeighbors()`** : Tableau des pays ayant le plus grand nombre de voisins. Affichez aussi les voisins.

Q3 - **`neighborless()`** : Tableau des pays n'ayant aucun voisin.

Q4 - **`moreLanguages()`** : Tableau des pays parlant le plus de langues. Affichez aussi les langues (objets **`Language`**).

Q5 - **`withCommonLanguage()`** : Tableau des pays ayant au moins un voisin parlant l'une de ses langues. Affichez aussi les pays voisins (objets **`Country`**) et les langues en question (objets **`Language`**).

Q6 - **`withoutCommonCurrency()`** : Tableau des pays sans aucun voisin ayant au moins une de ses monnaies.

Q7 - **`sortingDecreasingDensity()`** : Tableau des pays triés par ordre décroissant de densité de population.

Q8 - **`moreTopLevelDomains()`** : Tableau des pays ayant plusieurs Top Level Domains Internet.

Page de test

Vous devez créer une page **`test.html`** (associée à votre script **`test.js`**, voir la structure plus loin) présentant tous les tests (Q1. à Q8.)

Sur cette page, les tests seront sous la forme de boutons qui, au clic, exécuteront le test.

Structure du rendu

Voici la structure imposée que vous devez respecter (attention au malus 😌) :

```
+ html/
|   + data/
|   |   + class_country.js
|   |   + class_currency.js
|   |   + class_language.js
|   |   + countries.js
|   + test/
|   |   + test.html
|   |   + test.js
```