R4.A.10 Complément Web 1 - Modèle pour gestion des Pokémons

Avant-propos

Ce projet va vous permettre de mettre en œuvre, en 3 étapes, les concepts vus en TD et lors des premières séances de TP.

L'objectif de ce projet est de construire une WebApp qui affichera des informations sur les Pokémons, et permettra de rechercher, trier, filtrer les Pokémons et leurs informations en fonction de divers critères qui vous seront détaillés ultérieurement. Cette WebApp permettra en outre de simuler des affrontements entre Pokémons dans une version simplifiée.

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'un API Web Service.

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.

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 des fichiers contenant des structures JSON. Nous verrons dans une partie future comment récupérer ces informations depuis un WebService.

Les fichiers de données

Récupérez les fichiers sur Teams :

- **`pokemon.json`** : les caractéristiques de base des Pokémons.
- **`pokemon_type.json`** : les types de Pokémons.
- **`pokemon_moves.json`** : les attaques (qu'on appelle aussi mouvements) que peut effectuer chaque Pokémon, en fonction de sa forme.
  Ces mouvements sont des attaques soit "rapides", soit "chargées".
- **`fast_moves.json`** : le détail des attaques rapides, dont leurs types.
- **`charged_moves.json`** : le détail des attaques chargées, dont leurs types.
- **`type_effectiveness.json`** : le coefficient multiplicateur appliqué à la force d'une attaque en fonction de son type et du type du Pokémon qui subit l'attaque.
- **`cp_multiplier.json`** : le coefficient multiplicateur appliqué aux caractéristiques de base d'un Pokémon en fonction de son niveau.
- **`generation.json`** : la génération du Pokémon. Jusque fin 2021, il y a eu 8 générations de Pokémons. Chaque génération apporte un ensemble de nouveaux Pokémons. Annoncée lors du Pokémon Presents du 27 février 2022, la neuvième génération vient de sortir, mais n'est pas utilisée ici.

Afin de vous simplifier l'utilisation de ces fichiers **`JSON`**, nous vous invitons à les adapter légèrement de la façon suivante :

- Renommer-les en **`.js`** au lieu de **`.json`**, ce qui les rendra donc exécutables.
- Adaptez le début de chaque fichier en affectant à une variable nommée comme le fichier (sans le **`.js`**) l'objet **`JSON`** qu'il contient. Exemple avec **`fast_moves.json`** qui devient donc **`fast_moves.js`**, son contenu actuel :

```
[
   {
      "duration": 400,
      "energy_delta": 6,
etc...
```

devient :

```
let fast_moves = [
   {
      "duration": 400,
      "energy_delta": 6,
etc...
```

- Enfin, importez ces fichiers **JS** dans votre page HTML comme suit :

```
<script src="fast_moves.js"></script>
```

Les Pokémons

Cet ensemble de données contient la description de 898 Pokémons des 8 premières générations. Analysez la structure **JSON** d'un Pokémon décrite dans le fichier **pokemon.json**.

On notera la présence des attributs suivants : **pokemon_id**, **pokemon_name**, **form**, **base_attack**, **base_defense** et **base_stamina**.

Le fichier **generation.json** détermine la génération d'un Pokémon.

Ces données décrivent les caractéristiques de base d'un Pokémon. Certaines d'entre elles pourront évoluer au fil du jeu. Cette évolution est décrite dans le fichier **cp_multiplier.json**. Ces caractéristiques de base sont utiles pour déterminer le déroulement d'un affrontement entre deux Pokémons.

Comme il est prévu, à terme, d'afficher les informations sur les Pokémons dans la WebApp, une archive contenant les images des Pokémons est disponible sur Teams. Elle contient des images au format **webp**, classées dans des dossiers en fonction des types d'images. Le numéro du Pokémon est codé sur 3 chiffres :

- **images/<num_pokemon>.webp** : les images complètes des Pokémons.
- **thumbnails/<num_pokemon>.webp** : les images miniatures des Pokémons.

.

Les types de Pokémons

Le fichier **pokemon_type.json** décrit les différents types de Pokémons en fonction de leur forme. La forme permet de distinguer des variantes d'un même Pokémon (par exemple : **Normal**, **Fall_2019** ou encore **Alola**).

Un Pokémon d'une certaine forme possède 1 ou 2 types maximum. Si on analyse ce fichier, on peut dénombrer 18 types différents pour les Pokémons. Ces types s'appliquent également aux attaques.

Ce type va conditionner les attaques que chaque Pokémon pourra effectuer, mais aussi la résistance aux attaques qu'il subira.

Les attaques

Le fichier **`pokemon_moves.json`** décrit les différents types de mouvements (attaques) que les Pokémons peuvent effectuer en fonction de leur identifiant et de leur forme. On retrouve les types de ces attaques dans les deux fichiers **`fast_moves.json`** et **`charged_moves.json`**.

Le principe d'un combat de Pokémons est que chaque Pokémon attaque à son tour celui auquel il fait face. Son attaque, en fonction de son type et du type de son adversaire, va avoir une certaine efficacité (elle peut même être totalement inefficace). Cette efficacité est décrite dans le fichier **`type_effectiveness.json`**. La clé de chacun des éléments de cette structure de données est le type de l'attaque lancée, et en fonction du type du Pokémon défenseur donne un coefficient multiplicateur.

Réalisation du Modèle

L'objectif est donc dans un premier temps d'implanter un modèle de données en JavaScript à partir de la description des données faites dans les différents fichiers que nous venons de citer. L'étape suivante sera d'utiliser les classes et leurs méthodes pour afficher les données à la demande dans la WebApp.

La classe Pokemon

Q1. A partir de la description faite plus haut et de votre analyse des structures dans les fichiers **`JSON`**, proposez une classe **`Pokemon`**. Prévoyez une méthode **`toString()`** synthétique. Pour simplifier l'indexation, on ne désire modéliser que les Pokémons dont la forme (attribut **`form`**) a pour valeur **`Normal`**.

Q2. Concevez une fonction **`import_pokemon()`** qui lit la source de données et crée des objets **`Pokemon`** que vous stockerez dans un objet JS dont les clés sont les **`id`** des Pokémons. Cet objet JS sera une variable de classe nommée **`all_pokemons`**.

La classe Type

Q1. À partir des fichiers **`pokemon_type.json`** et **`type_effectiveness.json`**, proposez une classe **`Type`**. Ce type permettra donc de connaître l'efficacité d'un type d'attaque contre un type de défenseur. N'oubliez pas la méthode **`toString()`**. De même que pour les Pokémons, on ne gardera que les types des formes dont la valeur est **`Normal`**. Une variable de classes nommée **`all_types`** contiendra l'ensemble des types de Pokémons et sera indexé par le nom du type.

Q2. Complétez la fonction **`import_pokemon()`** qui lit la source de données et crée des objets **`Type`** au fur et à mesure que les Pokémons sont importés, si ce type n'est pas déjà dans **`all_types`**. Dans la classe **`Pokemon`**, vous devrez faire le lien avec les types.

Q3. Complétez la classe **`Pokemon`** avec une méthode **`getTypes()`** qui retourne la liste des types (des objets **`Type`**, pas des **`id`** ou du JSON !).

La classe Attack

Q1. À partir de **`pokemon_moves.json`**, **`fast_moves.json`** et **`charged_moves.json`**, proposez une classe **`Attack`** et complétez la classe **`Pokemon`**. N'oubliez pas une méthode **`toString()`**. Une variable de classe **`all_attacks`** contiendra toutes les attaques existantes, indexées par l'identifiant de l'attaque.

Q2. Complétez la fonction **`import_pokemon()`** qui lit la source de données et crée des objets **`Attack`** au fur et à mesure que les Pokémons sont importés, si cette attaque n'est pas déjà dans **`all_attacks`**.

Q3. Complétez la classe **`Pokemon`** avec une méthode **`getAttacks()`** qui donne la liste des attaques (des objets **`Attack`**, pas des **`id`** ou du JSON !).

Tests

Écrivez les fonctions qui permettent de répondre aux questions suivantes. Vous testerez (afficherez) dans la console d'un navigateur ou dans la page HTML, à votre convenance.

Q1. Fonction **`getPokemonsByType(typeName)`** donnant la liste des Pokémons par type, celui-ci étant passé en argument.

Q2. Fonction **`getPokemonsByAttack(attackName)`** donnant la liste des Pokémons par attaque, celle-ci étant passée en argument.

Q3. Fonction **`getAttacksByType(typeName)`** donnant la liste des attaques par type, celui-ci étant passé en argument.

Q4. Fonction **`sortPokemonByName()`** donnant la liste des Pokémons triés par nom dans l'ordre alphabétique.

Q5. Fonction **`sortPokemonByStamina()`** donnant la liste des Pokémons triés dans l'ordre décroissant d'endurance (stamina).

Q6. Fonction **`getWeakestEnemies(attack)`** qui retourne la liste des Pokémons qui ont le moins de résistance à une attaque donnée choisie parmi les attaques du Pokémon.

Q7. Fonction **`getStrongestEnemies(attack)`** qui retourne la liste des Pokémons qui ont le plus de résistance à une attaque donnée choisie parmi les attaques du Pokémon.