2 - Présentation des données des Pokémons

Introduction

Dans cette deuxième partie, vous allez vous focaliser sur la présentation des données à l'écran par la manipulation du DOM. Puisqu'il s'agit de la partie 2 du projet, vous devrez vous appuyer sur ce qui a été réalisé dans la partie 1, à savoir le chargement de la source JSON et la création des objets **Pokemon**, **Type** et **Attack**.

Structure du rendu

Voici la structure (qui complète celle donnée dans la partie 1) imposée que vous devez respecter (attention au malus 😌) :

```
+ html/
|   + webp/
|   |   + images/
|   |   + sprites/  (si besoin)
|   |   + thumbnails/
|   + css/
|   + data/
|   |   + pokemons.js
|   |   + generation.js
|   |   + ...
|   |   + class_pokemon.js
|   |   + class_type.js
|   |   + class_attack.js
|   + test/
|   |   + test.html  (voir sujet partie 1)
|   |   + test.js
|   + v1/
|   |   + pokemons_v1.html
|   |   + script_v1.js
|   + v2/
|   |   + pokemons_v2.html
|   |   + script_v2.js
|   ...
```

Vous ferez une archive ZIP du dossier racine (**`html/`**) que vous nommerez de vos NOMS par ordre alphabétique. Exemple **`NOM1_NOM2.zip`**

Rappels

- Commitez dans Git, toutes les étapes de votre avancement dans le projet, et ceci de façon individuelle.
- N'oubliez pas de commenter votre code.
- Respectez les noms s'ils sont spécifiés (nom des pages, nom des variables, etc.).
- Lisez l'énoncé avant de commencer à coder

Modèle de page

Créez une page **`template.html`** qui doit afficher en HTML brut, **sans faire référence aux données JSON**, une **`<table>`** contenant un seul Pokémon de votre choix. Vous utiliserez du CSS pour embellir votre table.

La table doit avoir un **`<thead>`** pour identifier le nom des colonnes.

La table doit afficher uniquement les informations suivantes :

- identifiant unique
- nom
- génération
- types (liste des noms des types du Pokémon)
- endurance (stamina)
- points d'attaque de base
- points de défense de base
- image (en taille miniature, adaptée à la hauteur des autres champs d'information)

Cette page va vous servir de modèle pour construire la page complète de tous les Pokémons (question suivante). On commence par cette page-modèle car il est plus aisé de mettre au point un montage de page statique que de coder la page complète (avec ses données) directement.

Tous les Pokémons

Sur la base de cette page template.html, créez une page **`pokemons_v1.html`** (qui sera associée à un **`script_v1.js`**) qui doit afficher la liste de tous les Pokémons provenant de la source JSON. Cette fois-ci les **`<tr>`** de la **`<table>`** vont être construits

dynamiquement, en énumérant (boucle) les données JSON et en manipulant le DOM pour ajouter vos **<tr>**.

Vous l'aurez compris, la ligne **<tr>** va servir de modèle (ou de template) pour construire toutes les lignes de votre **<table>**.

Pour ce faire, avant de commencer, vous pouvez peut-être commencer par remplacer les informations des **<td>** de votre ligne d'entête en y mettant des labels facilement identifiables, comme par exemple **ID**, pour l'identifiant, **NOM** pour le nom, **GEN** pour la génération, **URL** pour l'adresse de l'image miniature, qu'il vous sera alors simple de rechercher/remplacer pour générer vos **<tr>** finaux.

Pagination

Sur la base de **pokemons_v1.html**, créez une page **pokemons_v2.html** (+ **script_v2.js**) qui va afficher la liste des Pokémons par page de 25.

Tous les Pokémons continuent d'être chargés en mémoire dès le départ mais seuls les 25 premiers sont affichés au chargement de la page.

Votre page HTML doit afficher deux boutons **"PRÉC"** et **"SUIV"** qui vont permettre de naviguer de page en page. Vous devez donc coder les événements pour rafraîchir votre liste en tenant compte du numéro de page que vous afficherez sur votre page.

Vous devrez indiquer le numéro de page et le nombre total de pages et gérer l'état des boutons en fonction de l'existence ou non d'une page suivante/précédente. Les boutons doivent toujours être visibles.

NB : si on rafraîchit la page du navigateur manuellement, on perd la mémoire du numéro de page, c'est acceptable. Quand vous aurez terminé toutes les questions, s'il vous reste du temps vous pourrez revenir améliorer ce code en conservant le numéro de page et la colonne de tri (voir plus loin) dans un cookie.

Détails

Sur la base de **pokemons_v2.html**, créez une page **pokemons_v3.html** (+ **script_v3.js**) dans laquelle vous ajouterez une zone de détails qui est masquée par défaut et s'affiche, en mode "popup" (en surimpression de la liste), sur un événement "clic sur la ligne d'un Pokémon". Prévoir un moyen de refermer cette zone pour pouvoir accéder de nouveau à la liste masquée derrière.

Le détail sur les Pokémons apportera des informations sur les attaques notamment car elles ne figurent pas dans la liste générale des Pokémons. Dans un premier temps, vous ne mentionnerez que les noms des attaques classées par type (**charged** et **fast**). Vous compléterez ensuite avec les caractéristiques des attaques.

Un survol de la miniature du Pokémon affichera l'image du Pokémon en grande taille, aussi en mode "popup". Prévoir le masquage en fin de survol.

Astuce: prévoyez un attribut spécial (de votre choix) sur chaque **`<tr>`** que vous pourrez ensuite récupérer par manipulation du DOM pour identifier le Pokémon cliqué.

Filtrage

Sur la base de **`pokemons_v3.html`**, créez une page **`pokemons_v4.html`** (+ **`script_v4.js`**) dans laquelle vous ajouterez une zone filtrage au dessus de la **`<table>`** et dans laquelle on pourra filtrer la liste des Pokémons affichés (toujours par page de 25).

Le filtrage doit proposer les filtre suivants :

- **Génération** : liste déroulante construite dynamiquement à partir des données. Prévoir un choix vide pour ne pas utiliser le filtre.
- **Type** : idem que **Génération**
- **Nom** : champ de texte libre servant à filtrer les Pokémons dont le nom "contient" le texte saisi (si non-vide).

Tous les filtres doivent s'appliquer en cours de saisie ou dès la sélection d'un choix dans une liste (pas de bouton de validation !).

Les filtres doivent pouvoir être combinés entre eux et, évidemment, ce filtrage doit aussi être combiné à la pagination, codée précédemment.

Tout filtrage réinitialise la pagination (affichage en page 1).

Tri

Sur la base de **`pokemons_v4.html`**, créez une page **`pokemons_v5.html`** (+ **`script_v5.js`**) dans laquelle, sur clic d'un entête de colonne (sauf image), vous devez trier votre liste de Pokémons. En cas d'égalité de valeur (ex : deux Pokémons de même génération sur un tri par génération), vous départagerez les deux Pokémons sur leur nom.

Vous mettrez l'entête de colonne en gras pour indiquer le critère de tri appliqué..

Le tri se fait par ordre alphabétique lors du premier clic sur l'entête d'une colonne puis l'ordre est inversé au clic suivant.