# R4.A.10 Complément Web Partie 2

## Introduction

Dans cette seconde partie, vous allez vous focaliser sur la présentation des données à l'écran par manipulation du DOM.

Évidemment, 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 **`Country`**, **`Currency`** et **`Language`**.

Rappels :

- N'oubliez pas de commenter votre code.
- Respectez les noms s'ils sont spécifiés (nom des pages, nom des variables etc.)
- Lisez avant de commencer à coder.

## Modèle

Créez une page **`template.html`** qui doit afficher en **`HTML`** brut, sans faire référence aux données JSON (i.e. en y plaçant les valeurs en dur dans la page), une **`<table>`** contenant un seul pays de votre choix. Vous pouvez utiliser un peu de CSS pour embellir votre table si vous voulez, mais n'y passez pas plus de quelques minutes, ce n'est pas l'objet de ce projet.

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

La table doit afficher uniquement les informations suivantes :

- Nom en français
- Population
- Surface
- Densité de population
- Continent d'appartenance
- Drapeau (en taille mini, adaptée à la hauteur des autres champs d'information)

Cette page va vous servir de modèle pour construire la page complète de tous les pays (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.

Si certaines informations sont absentes ou non calculables, vous afficherez **N/A**.

# Tous les pays

À partir de **`template.html`**, créez une page **`countries_v1.html`** (+ **`script_v1.js`**) qui affiche la liste de tous les pays provenant du tableau **`all_countries`**. Cette fois-ci, les **`<tr>`** de la **`<table>`** vont être construits dynamiquement, en énumérant les données **`JSON`** et en manipulant le **`DOM`** pour ajouter vos **`<tr>`**.

Conseil : étudiez l'utilisation d'une **`map`** au lieu d'une boucle traditionnelle pour parcourir le tableau **`all_countries`** et en générer les lignes **`<tr>`**.

# Pagination

À partir de **`countries_v1.html`**, créez une page **`countries_v2.html`** (+ **`script_v2.js`**) qui va afficher la liste des pays par page de 25.

Tous les pays continuent d'être chargés en mémoire dès le départ mais seuls les 25 premiers sont affichés.

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 aussi quelque part sur votre page.

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 bas) dans un cookie.

# Détails

À partir de **`countries_v2.html`**, créez une page **`countries_v3.html`** (+ **`script_v3.js`**) dans laquelle vous ajouterez une zone de détails qui est masquée par défaut et qui s'affiche, par dessus la liste des pays par exemple, sur un événement "clic sur une ligne d'un pays". Prévoir un moyen de refermer cette zone pour pouvoir accéder de nouveau à la liste masquée derrière.

Attention, un clic sur le drapeau n'affiche pas la zone de détail du pays, mais affiche le drapeau en grand (par dessus la liste aussi).

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

# Filtrage

À partir de **`countries_v3.html`**, créez une page **`countries_v4.html`** (+ **`script_v4.js`**) dans laquelle vous ajouterez une zone de filtrage au dessus de la **`<table>`** et dans laquelle on pourra filtrer la liste des pays avant leur affichage (toujours par page de 25).

Le filtrage doit proposer les filtres suivants :

- **`Continent`** : liste déroulante construite dynamiquement à partir des données. Prévoir un choix vide pour ne pas utiliser ce filtre.
- **`Langue`** : idem que **`Continent`**
- **`Pays`** : champ de texte libre servant à filtrer les pays dont le nom français ou anglais “contient” le texte saisi (si non vide) et sans tenir compte ni de la casse ni des accents.

Le filtrage et donc l’affichage, doivent être faits en temps réel, au fur et à mesure de la saisie.

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

# Tri

À partir de **`countries_v4.html`**, créez une page **`countries_v5.html`** (+ **`script_v5.js`**) dans laquelle, sur clic d’un en-tête de colonne (sauf “drapeau”), vous devrez trier votre liste de pays. En cas d’égalité de valeur (ex : 2 pays du même continent dans un tri par continent), vous départagerez les 2 pays sur leur nom français.

Vous mettrez l’en-tête de colonne en gras pour indiquer sur laquelle se fait le tri.

Un 1er clic sur un intitulé de colonne tri ces données par ordre croissant (numérique ou alphabétique). Un second clic sur la même colonne provoque l'inversion du sens du tri.

# Participation

Vous devez nous fournir un fichier **`participation.txt`** contenant la part de contribution de chaque membre sous forme d’un pourcentage global pour tout le projet (pas de détail à la question).

# Livrables

```
+ html/
|   + data/
|   |   + class_country.js
|   |   + class_currency.js
|   |   + class_language.js
|   |   + countries.js
|   + jquery-3.6.3.min.js
|   + test/
|   |   + test.html
|   |   + test.js
|   + v1/
|   |   + countries_v1.html
|   |   + script_v1.js
|   + v2/
|   |   + countries_v2.html
|   |   + script_v2.js
|   + v3/
|   |   + countries_v3.html
|   |   + script_v3.js
|   + v4/
|   |   + countries_v4.html
|   |   + script_v4.js
|   + v5/
|   |   + countries_v5.html
|   |   + script_v5.js
+ participation.txt
```