TP 003 : Ajax, Json, Geoloc et SVG

Dans un premier temps téléchargez l'archive base, ce sont les fichiers dans lesquels vous allez travailler.

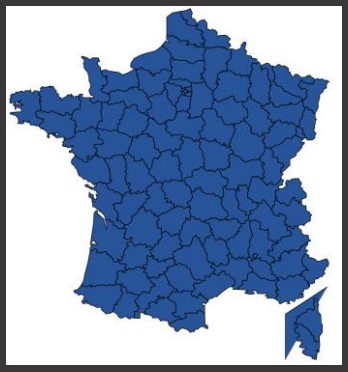
Dans cet archive vous trouverez un fichier .svg, c'est un format d'image dit vectoriel. Pour faire simple, ce sont des images définies par un ensemble de traits ; contrairement aux images classiques qui sont définies par un ensemble de pixel.

On peut facilement trouver sur le web la description en svg des différents pays. Le site du gouvernement propose une extraction en svg tirée du site openstreepmap, l'équivalent opensource de googlemap.

Copiez le contenu de ce fichier et mettez-le dans le body du fichier html.

La page reste blanche parce que nos dessins n'ont pas de couleurs.

Dans le CSS, grâce à l'identifier de l'élément svg, donnez-lui une largeur de 100% et une hauteur maximale de 100%.
Puis, grâce aux propriétés fill et stroke, colorez tous les éléments path du svg en bleu avec une bordure noire.



Vous devriez arriver à ce résultat.

Exercice 1 : Geolocalisation

Dans votre fichier javascript, créez une variable latitude et longitude.

Pour la suite vous trouverez un très bon exemple sur w3schools, qu'il faudra adaptez avec les instructions suivantes (https://www.w3schools.com/html/html5_geolocation.asp) :

- Créez une fonction activeLocation. Dans cette fonction appelez la geolocalisation du navigateur. La méthode getCurrentPosition prendra en paramètre une fonction nommée showLocation.

- Créez la fonction showLocation qui prend un paramètre. Ce paramètre sera automatiquement rempli par la méthode getCurrentPosition de la geolocation.
- Dans cette fonction sauvegardez la latitude et la longitude dans les variables latitude et longitude créées précédemment.
- Vérifiez que le bon fonctionnement avec un affichage console. Pensez à faire un appel de la fonction adéquate au début du fichier JS.

Exercice 2 : Ajax

L'ajax est une technologie qui va vous permettre de communiquer en JavaScript avec des serveurs web.

Pour communiquer avec un serveur web on utilise le protocole HTTP, puis on donne l'adresse du fichier sur ce serveur. Par exemple, quand j'écris http://www.monsite.fr/index.html dans la barre d'adresse de mon navigateur, cela va générer une requête HTTP vers le serveur monsite.fr pour récupérer le fichier index.html.

Certains serveurs web mette à disposition ce qu'on appelle une API. C'est-à-dire qu'on va faire appelle à une fonction du serveur web via des requêtes HTTP. Par exemple, openstreetmap met à disposition un ensemble d'API dont l'une d'elle permet de récupérer toutes les informations géographiques d'un point gps donné.

Dans l'exercice 1 vous avez récupéré votre position GPS, il va donc falloir fournir ces informations au serveur openstreetmap pour qu'ils puissent vous délivrer son savoir.

Commencez par créer une fonction ajaxOpenSM.

Dans cette fonction, vous allez travailler avec un objet XMLHttpRequest. Cet objet va vous permettre de construire facilement une requête HTTP, ce que fait habituellement le navigateur web sans que vous ne le voyiez.

```
maRequete = new XMLHttpRequest();
```

Je ne détaille pas cette instruction, c'est de la programmation objet que vous verrez le semestre prochain.

Ensuite, 3 choses :

- *On doit indiquer l'adresse à laquelle envoyer la requête*

```
maRequete.open('GET','https://nominatim.openstreetmap.org/
reverse?format=json&lat=49.24&lon=2.9', true);
```

- *Donnez une fonction de callback à la requête (pour traiter la réponse)*

```
maRequete.onreadystatechange = callB;
```

- *Envoyer la requête*

```
maRequete.send();
```

Vous avez donc tout mis en place pour interroger le serveur openstreetmap sur la position 49.24 ;4.9.

Cependant la fonction de callback, callB, n'exsite pas.

Créez-la.

Dans cette fonction, mettez les instructions suivantes :

```
if (maRequete.readyState === XMLHttpRequest.DONE) {
    if (maRequete.status === 200) {
        console.log(maRequete.responseText);
    } else {
        alert(maRequete.status);
    }
}
```

La première condition regarde si la requête a bien été envoyée. La deuxième condition vérifie que la requête a bien été traitée (200 = OK). Puis, on affiche dans la console ce que nous a renvoyé le serveur.

A la fin de la fonction saveLocation, appelez la fonction ajaxOpenSM et observez le résultat dans la console. Le serveur vous a renvoyé du texte au format JSON. Dans ce texte vous pouvez voir que les coordonnées fournies correspondent à la ville de Senlis dans l'Oise et dont le code postal est 60800.

Dans l'adresse de destination, remplacez ''reverse'' par ''notgood''. Observez le résultat.
Remettez reverse. Grâce à la concaténation, remplacez 49.24 par votre variable latitude et 2.9 par votre variable longitude. Cette fois, la réponse devrait vous indiquez la ville dans laquelle vous vous trouvez.

Exercice 3 : JSON.

On veut maintenant extraire de la réponse, le code postal. Cet exercice nécessite de la recherche de votre part !

Créez une fonction getCP, qui prend un paramètre nommé response.
Dans la fonction callB, après l'affichage console, appelez la fonction getCP et passez lui en paramètre la réponse de la requête.

Dans la fonction getCP, affichez le paramètre response pour vérifier que tout se déroule bien.

Trouvez une manière d'extraire les deux premiers chiffres du code postal, soit par manipulation de chaînes de caractères, soit grâce à son format JSON. Mettez en place cette extraction dans la fonction getCP et retournez le résultat (par exemple la fonction retourne 51 si vous êtes dans la marne).

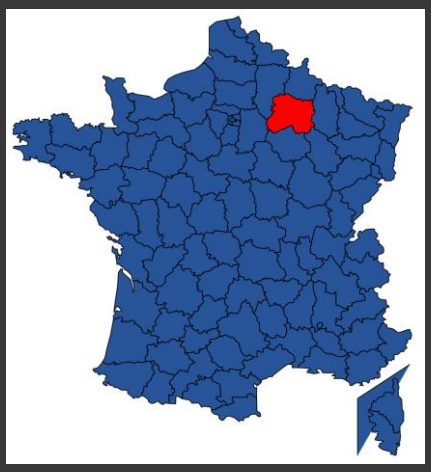
Dans la fonction callB, stockez le retour de la fonction dans une variable nommé codeP.

Il ne manque plus qu'à colorez votre département sur la carte. Si vous observez le svg, vous verrez que chaque path a un attribut data-num qui est le numéro du département.

Créez une fonction colorDPT, qui prend un paramètre dpt.

Grâce à querySelector, récupérez le path qui a pour attribut data-num égale au paramètre dpt. Modifiez sa propriété de style fill pour qu'elle soit rouge.

Appelez la fonction colorDPT au bon endroit.



Vous avez fini !