TP n° 04 – AJAX et Fetch Ressource R209

AJAX et Fetch sont 2 techniques permettant de récupérer des données de façon asynchrone depuis un serveur distant.

1 Données issues d'un service WEB de comptage de vélos :

- 1. Le site https://data.montpellier3m.fr/ permet de consulter les données partagées par l'agglomération de Montpellier. Trouver sur le site, l'URL donnant le nombre de vélos comptés au point X2H19070220.
- 2. Tester l'accès aux données depuis un terminal Linux à l'aide de la commande système curl.
- 3. Tester ensuite le code fourni sur moodle, en adaptant la valeur de l'URL. Quel message d'erreur obtient-on tout de même lorsque l'on clique sur le bouton (voir la console WEB du navigateur) ?
- 4. En fait le serveur ne fourni pas d'entête CORS (Cross-Origin Resource Sharing) autorisant l'accès aux données depuis n'importe quel domaine. On peut le vérifier en ajoutant l'option –v à la commande curl du §1.2 et en cherchant l'entête Access-Control-Allow-Origin.

Pour s'affranchir de ce problème, on peut créer un script local récupérant les données et ajoutant la bonne entête (voir json.php).

```
<?php
header('Access-Control-Allow-Origin: *');
header('Content-Type: application/json; charset=utf-8');
$r=@file_get_contents('https://data.montpellier3m.fr/...');
echo ($r)?$r:'{"error":"NOT FOUND !"}';
exit(0);
?>
```

Mais surtout, il faut le placer sur un serveur WEB auquel on accès (par exemple celui utilisé en R207: https://r207.borelly.net/~uXXXXX/ où XXXXX est votre numéro d'étudiant, voir TP n° 02 de R109 pour l'accès). Adapter l'URL du service WEB indiquée à la ligne 4 et copier le script PHP dans le sous répertoire r209-tp4 de la partie web de votre compte utilisateur et tester celui-ci avec curl:

```
curl -v https://r207.borelly.net/~uXXXXX/r209-tp4/json.php
```

- Changer l'URL dans le script §1.3 et vérifier que cela fonctionne bien maintenant.
- 6. On peut tester ensuite avec la méthode ajax de JQuery:

```
$.ajax({
   url : "https://r207.borelly.net/~uXXXXX/r209-tp4/json.php",
   dataType : "json"
}).done(function(json0bj) {
   cblog("AJAX OK...<br/>"+JSON.stringify(json0bj));
}).fail(function() {
   cblog("AJAX ERROR !!!");
});
```

NB: Sachant que l'on veut récupérer seulement des données par GET, on peut aussi se servir des méthodes simplifiées de JQuery :

```
.load( url [, data ] [, complete ] )
jQuery.get( url [, data ] [, success ] [, dataType ] )
jQuery.getJSON( url [, data ] [, success ] )
```

7. Il est aussi possible d'utiliser l'API Fetch des navigateurs « modernes » :

```
fetch('https://r207.borelly.net/~uXXXXX/r209-tp4/json.php')
  .then(response => response.json())
  .then(json0bj => {cblog('Fetch OK...<br/>'+JSON.stringify(json0bj));})
  .catch(error => {cblog('Erreur Fetch...<br/>'+error)});
```

8. Comme les données sont au format JSON, il est facile d'accéder au contenu des données en JavaScript avec la notation objet. Afficher alors les coordonnées du point de mesure, la date et la valeur du nombre de vélos.

Nombre de vélos: 89

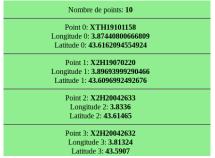
Date: **2021-03-15T00:00:00**Longitude: **3.8956288**Latitude: **43.6266977**

2 Affichage dynamique sur une carte :

On veut maintenant dessiner une carte référençant tous les points de mesure de comptage de vélo.

1. Récupérer tous les points de mesure disponibles avec une nouvelle requête (créer par exemple le script jsonGetPoints.php). Voir par exemple l'URL:

https://data.montpellier3m.fr/sites/default/files/ressources/MMM MMM GeolocCompteurs.geojson



- 2. Créer ensuite une carte OpenStreetMap centrée sur Montpellier avec un facteur de zoom de 12 (Voir TP n° 03).
- 3. Ajouter alors un marqueur simple pour chaque point trouvé au §2.1.
- 4. On veut maintenant pouvoir cliquer sur chaque point pour visualiser le nombre de vélos comptés. Il faut donc faire une requête spécifique pour chaque point de mesure. Adapter alors le script PHP de l'exercice 1 pour utiliser un paramètre (le numéro de série du point de mesure par exemple) et créer la fonction asynchrone async function getValueForPoint(ref).

Réf: XTH19101158 => 1794 Réf: X2H19070220 => 1339 Réf: X2H20042633 => 677 Réf: X2H20042632 => 244 Réf: X2H20063161 => 317 Réf: X2H20063163 => 737 Réf: X2H20063163 => 747 Réf: X2H20063162 => 1460 Réf: X2H20042634 => 89 Réf: X2H20042635 => 460

5. Créer enfin la fonction async function addPoints(points) permettant d'ajouter à la carte tous les points trouvés au §2.1. Ne pas oublier d'utiliser la méthode bindPopup() sur les marqueurs pour ajouter les « info-bulles ».

