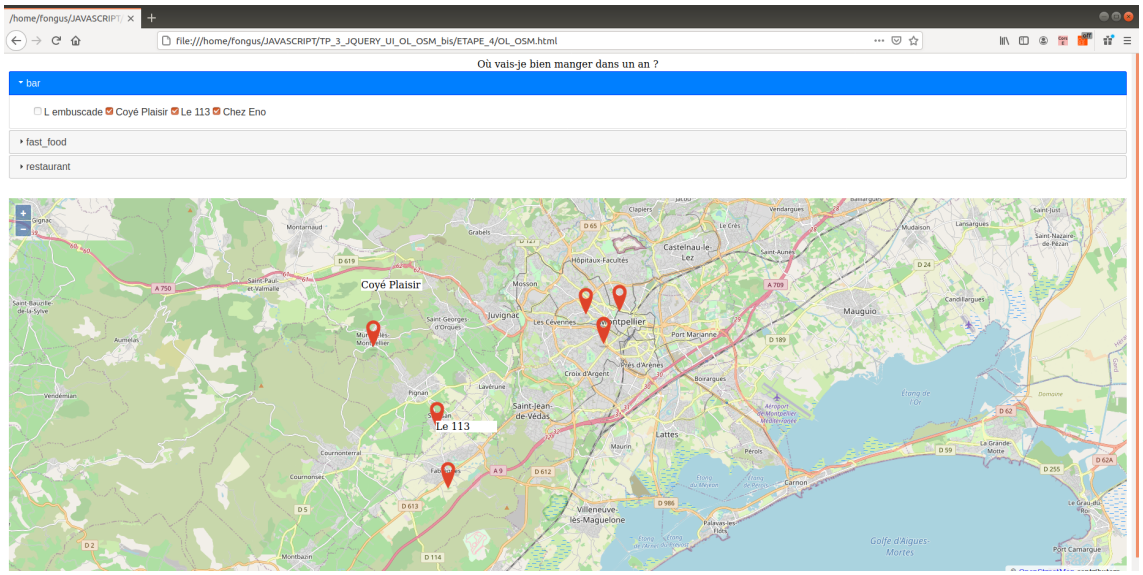
TD/TP JavaScript 3
Affichage d'une carte avec OpenLayers et OpenStreetMaps
Utilisation d'une widget jQuery-ui

–

Pierre Pompidor

Introduction au TP :

Ce TP a pour but d'afficher une **carte** avec la bibliothèque **OpenLayers** interfaçant la base de données cartographiques **OpenStreetMaps**. Sur cette carte des marqueurs correspondront à des points d'intérêt (dans notre exemple, des bars, des restaurants...). Ces points d'intérêt seront construits à partir d'informations délivrées par un serveur, répertoriés dans un fascinant composant graphique (une widget), et seront représentés sur la carte par des marqueurs qui, sélectionnés, afficheront leurs noms.



Vous devez faire ce TP en quatre étapes :

- Première étape : mise en place des pré-requis (notamment du serveur de données géolocalisées)
- Deuxième étape : affichage de cases à cocher correspondant aux points d'intérêt, puis d'un accordéon pour les mettre en musique
- Troisième étape : affichage d'une carte centrée sur Montpellier, puis des marqueurs correspondant aux points d'intérêt
- Quatrième étape : lien entre les cases à cocher et les marqueurs, affichage de popups sur la sélection des marqueurs

Première étape - mise en place des pré-requis :

Sur le Moodle, récupérez l'archive contenant les ressources nécessaires à ce TP (serveur Node/express pour fournir les données, le fichier JSON que va exploiter le serveur, l'image d'un marqueur, le fichier HTML et d'autres ressources pour les bibliothèques *JQuery* et *Open Layers*) et dézipez-la.

Mise en œuvre du serveur Node/express

Le fichier *OSM_Metropole_restauration_bar.json* contient un fragment des données géolocalisées sur le domaine de la restauration et des bars de l'opendata de la métropole de Montpellier.

Le code du serveur Node/express qui se trouve dans l'archive et nommé *serveur_points_d_interet_à_compléter.js* est à compléter. Pour qu'il puisse délivrer les données géolocalisées, vous devez créer un objet JavaScript nommé *pis* qui aura :

- pour clefs les différents types d'établissements (*amenity*)
- pour valeurs une collection (une liste d'objets), chaque objet ayant trois propriétés :
  - *nom* : le nom de l'établissement
  - *long* : la longitude
  - *lat* : la latitude

Vous devez rajouter du code là où il y a marqué "A COMPLETER", et seulement là !

```
var express = require("express");
var fs = require('fs');
var app = express();
app.listen(8888);

let pis = {}
let PI = JSON.parse(fs.readFileSync('JSON/OSM_Metropole_restauration_bar.json','utf8'));
>>>>>> A COMPLETER <<<<<<
console.dir(pis)

// Renvoi de la page HTML
app.get('/', function(request, response) {
    console.log('/');
    response.sendFile('OL_OSM.html', {root: __dirname});
});

// Renvoi des différents types d'établissements
app.get('/types', function(request, response) {
    console.log('/types');
    let types = [];
    for (let type in pis) types.push(type);
    response.setHeader("Content-type", "application/json");
    response.end(JSON.stringify(types));
});

// Renvoi des établissements d'un type donné
app.get('/type/:type', function(request, response) {
    console.log("/type/"+request.params.type);
    response.setHeader("Content-type", "application/json");
    response.end(JSON.stringify(pis[request.params.type]));
});

...
```

de telle sorte que :
`http://localhost:8888/types` renvoie la listes des différents types d'établissements (*amenity*) possibles :
`["bar", "fast_food", "restaurant"]`

`http://localhost:8888/type/restaurant` renvoie les informations sur quatre restaurants :


```
[{"name":"Pizza Mario","long":3.763083484826347,"lat":43.582574597739864},
 {"name":"Le Parc","long":3.763229627338037,"lat":43.58198519510649}],
 {"name":"Brasserie Le XV","long":3.776585934045546,"lat":43.551899463347695},
 {"name":"Fabrègaüda","long":3.77765443722869,"lat":43.54777021067066}]
```


N'oubliez pas que la méthode *hasOwnProperty()* permet de tester l'existence d'une propriété d'un objet.

Le serveur doit être évidemment exécuté par Node (et avant cela, renommé *serveur_points_d_interet.js*) :

```
node serveur_points_d_interet.js
```

Un coup d'œil sur la page HTML

Voici le contenu de la page HTML (qui ne sera pas à modifier) :

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="https://code.jquery.com/jquery-3.5.1.min.js" integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/fa
    <script src="http://localhost:8888/fichier/jquery-ui.min.js"></script>
    <link rel="stylesheet" href="http://localhost:8888/fichier/jquery-ui.min.css"></link>
    <script src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v6.0.1/build/ol.js"></scri
    <link rel="stylesheet" href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v6.0.1/css
  </head>

  <body>
    <center> <h7> Où vais-je bien boire ou manger ? </h7> </center>
    <div id="points_interet" class="fullwidth" style="height:25%"></div>
    <div id="map" class="fullwidth" style="height:75%"></div>
    <img id="markerProto" style="display: none; width: 100%;" src="http://localhost:8888/fichier/marker.png"
    <div id="popupProto" style="display: none; width: 100%; color:black; width:100px; height:100px"></div>
    <script src="http://localhost:8888/fichier/OL_OSM.js"></script>
  </body>
</html>
```

La bibliothèque *jQuery* sera utilisée pour importer du serveur Node.js les données au format JSON.
Une widget de la bibliothèque *jQuery-ui* permettra de créer un **accordéon** qui regroupera dans ses soufflets les restaurants, les fastfood et les bars..

La division *points_interet* contiendra les cases à cocher correspondant aux points d'intérêt.
La division *map* contiendra la carte.
L'image *markerProto* et la division *popupProto* seront utilisées par la bibliothèque *openlayers* pour placer les marqueurs et les informations associées sur une carte fournie par le fournisseur de cartes *OpenStreetMap*.
OL_OSM.js contiendra notre code JavaScript côté client qui va importer les données, les afficher dans une widget JQuery-ui et gérer la carte.

Mais le plus important à comprendre est que cette page HTML devra être chargée dans le navigateur via le serveur (pour éviter une alerte de sécurité CORS) :

```
http://localhost:8888/
```

Deuxième étape : affichage de cases à cocher correspondant aux points d'intérêt

Des cases à cocher doivent être insérées dans la division *points_interet*. Elles doivent correspondre aux noms des points d'intérêt que nous voulons faire afficher sur la carte (bars, restaurants, peut-être même des distributeurs de donuts...). Les informations concernant ces points d'intérêt sont délivrées par le serveur. Comme lors du TP précédent, nous allons utiliser la bibliothèque jQuery, et plus précisément `$.getJSON()`, pour importer ces informations.
Pour cela, créez le code JavaScript *OL_OSM.js* (le début de ce code sera présenté ci-après)..

La récupération des données doit se faire en deux temps : Il faut d'abord invoquer le serveur pour que celui-ci nous renvoie la liste des différents types de points d'intérêt (bar, restaurant...), et puis pour chaque type, le réinvoquer pour avoir la liste des points d'intérêt.

Le début du code JavaScript :

Le code JavaScript doit tout d'abord interroger le serveur pour que celui-ci lui donne les différents types d'établissements :

```
$().ready(function(){
  $.getJSON("http://localhost:8888/types", function(data) {
    ...
  });
});
```

`$()` sélectionne l'objet *document* du DOM et la fonction de rappel que gère la méthode *ready()* ne sera appelée que lorsque tous les élements du DOM auront été alloués.

La méthode *getJSON()* de l'objet `$` invoque le serveur et la fonction de rappel ne sera appelée que lorsque le serveur aura renvoyé les données (stockée dans la liste *data*).

La création des cases à cocher :

Attention, voici quelques contraintes de "fabrication" (elles sont requises pour la génération ultérieure d'un accordéon) :

- le nom du type doit apparaître entre deux balises ouvrante/fermante `<h3>` avant chaque série de cases à cocher ;
- les séries de cases à cocher doivent être insérées dans une division qui a pour *id* le nom du type ;

Par ailleurs les cases à cocher doivent avoir comme attribut *name* un numéro entier initialisé à 0 et incrémenté après chaque création de case à cocher.

Voici le code HTML **qui devrait être créé si nous codions cela en statique** pour deux bars et deux restaurants **(et bien sûr, ce n'est pas le code que vous allez écrire)** :

```
<div id="points_interet">
     <h3> bar </h3>
     <div id='bar'>
          <input type='checkbox' name='1'> restaurant1 </input>
          <input type='checkbox' name='2'> restaurant2 </input>
     </div>
     <h3> restaurant </h3>
     <div id='restaurant'>
          <input type='checkbox' name='3'> hotel1 </input>
          <input type='checkbox' name='4'> hotel2 </input>
     </div>
</div>
```

Voici le code HTML **qui devrait être créé si nous codions cela en statique** pour deux bars et deux restaurants **(et bien sûr, ce n'est pas le code que vous allez écrire)** :

```
<div id="points_interet">
     <h3> bar </h3>
     <div id='bar'>
          <input type='checkbox' name='1'> restaurant1 </input>
          <input type='checkbox' name='2'> restaurant2 </input>
     </div>
     <h3> restaurant </h3>
     <div id='restaurant'>
          <input type='checkbox' name='3'> hotel1 </input>
          <input type='checkbox' name='4'> hotel2 </input>
     </div>
</div>
```

La création de l'accordéon

Les cases à cocher doivent être gérées par un **accordéon** qui est une widget (composant graphique) de la bibliothèque **jQuery-ui**. Pour cela appliquez la méthode *accordion()* à la division *points_interet* comme suit :

```
$('#points_interet').accordion({collapsible: true, heightStyle: 'content'});
```

Mais attention l'asynchronisme de JavaScript peut rendre cette instruction inopérante si celle-ci est appliquée avant que le DOM du navigateur ne soit mis à jour (càd avant que toutes les cases à cocher n'aient été créées)....

Retenez votre émotion (surtout si cela ne marche pas).

Troisième étape : affichage d'une carte centrée sur Montpellier, puis des marqueurs correspondant aux points d'intérêt

Affichage d'une carte centrée sur Montpellier

Dans le fichier *OL_OSM.js*, rajoutez le code suivant :

```
var map = new ol.Map({
    target: 'map',
    layers: [new ol.layer.Tile({source: new ol.source.OSM()})],
    view: new ol.View({
          center: ol.proj.fromLonLat([3.876716,43.61]),
          zoom: 14
    })
});
```

Vous remarquerez avec un certain enthousiasme que :

- que la bien nommée classe **Map** crée une carte
- *target* spécifie la division d'accueil
- le fournisseur de tuiles ("tiles") est *Open Street Map*

Et rechargez la page...

Affichage sur la carte de marqueurs correspondant aux points d'intérêt

Il faut créer autant de marqueurs que de points d'intérêts. Avec *Open Layers*, cela passe par la création d'autant d'**overlays** (des objets instances de la classe **Overlay**) correspondant à des surcouches sur la carte. Et à chaque overlay sera associée une image qui sera un clone de l'image "prototype" dont nous disposons.

Nous devons donc :

- créer une liste qui contiendra nos marqueurs (en fait nos *objets overlays*) : l'indice de chaque overlay correspondrant à l'id d'une case à cocher...
- pour chaque point d'intérêt (donc dans une fonction) :
  - cloner l'image prototype et l'insérer dans le body
  - créer un objet overlay et lier l'image à celui-ci
  - associer l'overlay à la carte

Voici les éléments techniques dont nous avons besoin :

pour cloner l'image prototype :

```
let image = $("#markerProto").clone();
```

pour créer un objet overlay et lier l'image à celui-ci :

```
new ol.Overlay({position: ol.proj.fromLonLat([pi.long, pi.lat]),
                positioning: 'center-center',
                element: document.getElementById(...)});  // element fait référence à l'image
```

pour associer l'overlay à la carte :

```
map.addOverlay(...);
```

Par ailleurs n'oubliez pas de forcer l'affichage des marqueurs (car par défaut l'image prototype est désaffichée).

Quatrième étape : lien avec les cases à cocher et affichage de popups sur la sélection des marqueurs

Il nous reste à faire le lien entre les cases à cocher et les marqueurs.

Lien entre les cases à cocher et les marqueurs

Pour faire le lien entre la sélection d'une case à cocher et l'apparition ou la disparition du marqueur correspondant, nous allons mettre en place des écouteurs d'événements sur le changement de sélection des cases à cocher (svp, ne recopiez pas les ...) :

```
$('body').on("change", "input[type=checkbox]", function() {
    let valeur = $(this).attr('id');
    console.log("sélection de la case à cocher numéro "+valeur);
    if ($(this).is(':checked')) { ... }
    else { ... }
});
```

La méthode *on()* sur *body* permet d'associer des écouteurs d'événements sur de **futurs** éléments de *body* (ce n'est pas fort çà, si si, c'est fort)...

Affichage d'une popup lors de la sélection d'un marqueur

Rajoutez le code nécessaire pour qu'autant de popups (des divisions affichant dans des chaînes de caractères les noms des points d'intérêt) soient créées (mais désaffichées par défaut) sur la carte que de marqueurs. Chaque nouvelle popup doit être clonée à partir de celle importée dans la page HTML initiale.

La programmation de cette tâche est similaire à la précédente (en informatique le plaisir est si facilement renouvelable).