Leaflet

2D3D.GIS Organisme de formation
1 bis Perspective de l'Océan 17000 La Rochelle France

2

TÉLÉCHARGEMENT DE LEAFLET

PREMIÈRE CARTE

- Téléchargement de Leaflet
- Utilisation de Leaflet
- Création d'une première carte

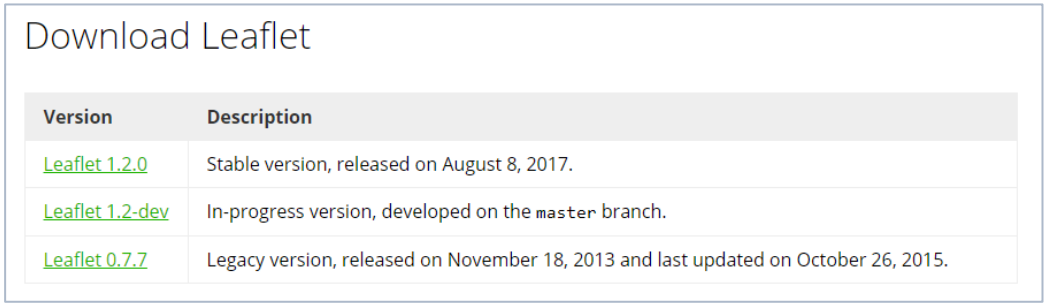
COURS

DÉBUTER AVEC LEAFLET

TÉLÉCHARGER LEAFLET

Leaflet est au même titre que jQuery une librairie de développement, il n'y a donc **pas d'installation logicielle** à réaliser pour l'utiliser mais simplement un téléchargement de fichiers qu'il suffira de placer dans un répertoire **js** pour une utilisation dans les pages HTML.

- Pour télécharger la **dernière version** de Leaflet**,** rendez-vous sur le site officiel www.leafletjs.com, puis sur le lien **Download** :



- Téléchargez la version stable en cliquant sur le premier lien, cela lancera le téléchargement d'un **fichier zip**.

UTILISER LEAFLET

Le zip téléchargé contient un ensemble de fichiers javascript, css et images. Pour les utiliser dans une page HTML, il faut les extraire dans le répertoire que nous avons créé dans les exercices précédents.

- **Dézippez** le répertoire téléchargé dans un dossier **leaflet** et placez ce dossier dans votre dossier **leaflet_intro/js** :

CRÉER UNE CARTE AVEC LEAFLET

Pour utiliser Leaflet et créer une carte dans une page web, il va falloir utiliser à minima 3 langages : **HTML**, **CSS** et **javascript**. Nous allons repartir d'un nouveau fichier HTML pour créer la carte.

- Créez un nouveau fichier HTML nommé **carte.html** dans le répertoire **leaflet_intro** pour y ajouter les chemins vers les librairies javascript **jQuery**, **Leaflet** et les fichiers de style Leaflet et personnel **style.css**.

```
<html>
<head>
  <title>Carte Leaflet</title>
  <script type="text/javascript" language="javascript" src="js/jquery-3.2.1.min.js"></script>
  <script type="text/javascript" language="javascript" src="js/leaflet/leaflet.js"></script>
  <script type="text/javascript" language="javascript" src="js/carte.js"></script>
  <link rel="stylesheet" type="text/css" href="css/style.css">
  <link rel="stylesheet" type="text/css" href="js/leaflet/leaflet.css">
</head>
<body>
  <div id="map"></div>
</body>
</html>
```

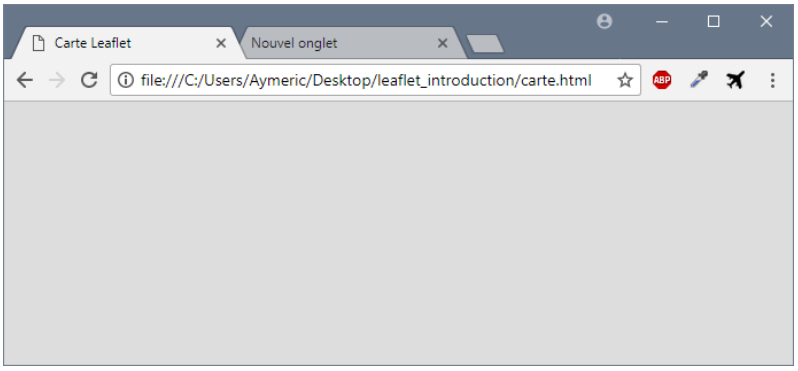
- Créez un nouveau fichier javascript vide (pour l'instant) nommé **carte.js** dans le dossier **js**
- Créez également un dossier **css** et à l'intérieur un fichier **style.css** et écrivez les lignes suivantes :

```
body, html {
  width:100%;
  height:100%;
  margin:0px;
  padding:0px;
}

#map {
    width:100%;
    height:100%;
    background:#DDD;
}
```

- La page HTML aura une hauteur et une largeur de 100% de la taille de la fenêtre
- Pas de marges extérieures ni intérieures dans la page
- La carte aura aussi une hauteur et une largeur de 100%
- Un fond gris sera placé dans la div de la carte

- Ouvrez la page carte.html dans un navigateur web et vérifiez que vous avez bien un fond gris sur l'ensemble de la page.



Si ce n'est pas le cas c'est que le fichier **css** n'est pas chargé ou qu'il y a une erreur dans les balises.

Maintenant, nous allons dire en javascript que Leaflet doit placer un composant cartographique dans notre **div map** à la place de ce fond gris.

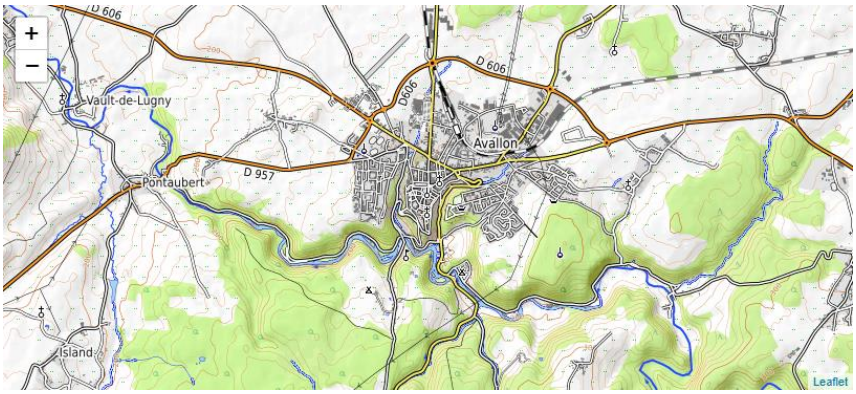
- Éditez le fichier **carte.js** pour y ajouter les lignes suivantes :

```
$(document).ready(function() {
    /* Création de la carte avec coordonnées GPS latitude, longitude et niveau de zoom */
  map = L.map('map').setView([47.48822, 3.90772], 13);

    /* Ajout d'un fond de plan Topo style IGN Scan25 */
  fondTopo = L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {
    maxZoom: 17
  }).addTo(map);
});
```

- Rafraichissez la page **carte.html** pour voir le résultat du script.

  Vous devriez visualiser la carte centrée sur la ville d'**Avallon**.

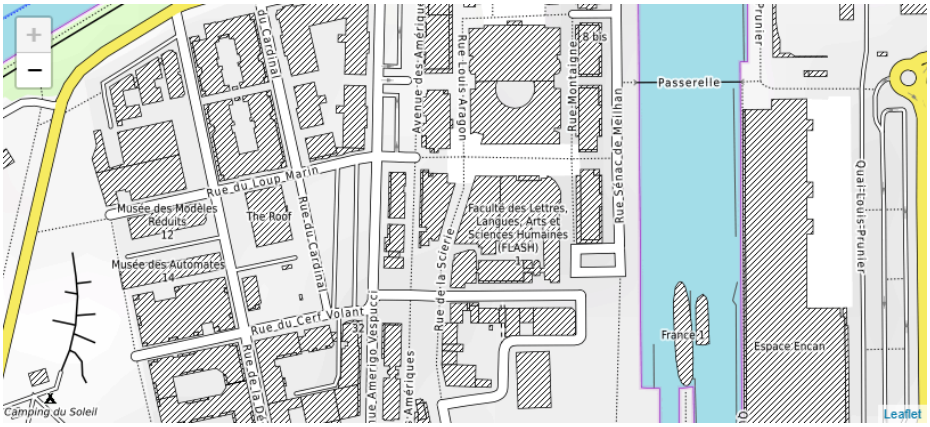


Pour centrer la carte sur une autre ville, il suffit de modifier les 2 coordonnées géographiques en **WGS84** de la première ligne et éventuellement ajuster le seuil de zoom **13**.

Les niveaux de zoom vont de 0 à x :

- 0 = Vue du monde entier
- 17 = Vue d'une rue
- x = En fonction du fond de plan utilisé on peut aller à un niveau plus précis.

Essayez par exemple avec les coordonnées géographiques **46.15197**, **-1.15385** et au zoom **17** pour constater la différence d'affichage de la carte.



3

DÉVELOPPER UNE CARTE

LES DONNÉES

1 bis Perspective de l'Océan, BP 63019, 17030 La Rochelle – France
+33 5 46 34 07 71
information@2d3d-gis.com

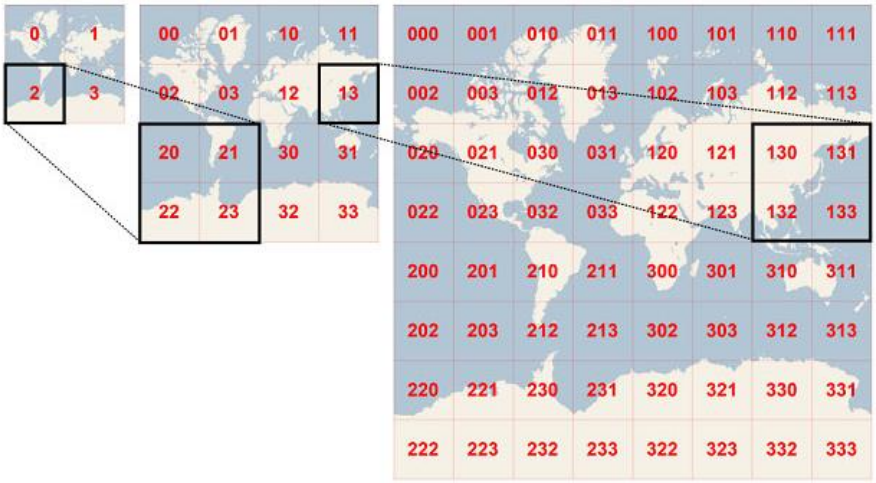
- Ajouter des données rasters
- Ajouter des données vecteurs
- Créer un gestionnaire de couches

COURS

TUILAGE RASTER

PLUGIN QGIS

Un tuilage raster est une succession de dalles de 256x256 pixels à différents seuil de zoom, une pyramide ou une image est décomposée en 4 images à un niveau de zoom supérieur.



Il existe dans QGIS un plugin permettant de générer un tuilage de type **xyz**, lu par la plupart des composants cartographiques comme Leaflet. Ce plugin se nomme **QMetaTiles**.

Il est possible de l'installer simplement et ensuite d'ouvrir le raster à tuiler.

Cela peut être une **superposition de données raster et vecteur**. QMetaTiles conservera tous les styles car il se contente de transformer en raster ce qui est affiché dans la carte QGIS courante) à tous les niveaux de zoom nécessaires spécifiés par l'utilisateur.

Le plugin permet de spécifier le zoom minimal et zoom maximal entre lesquels on souhaite générer les tuiles et il est **TRES** important de le faire en plusieurs étapes si on ne connait pas le nombre de tuiles approximatif qu'il y aura en sortie. Un tuilage d'un raster au niveau de zoom 21 peut générer des **millions de tuiles**.

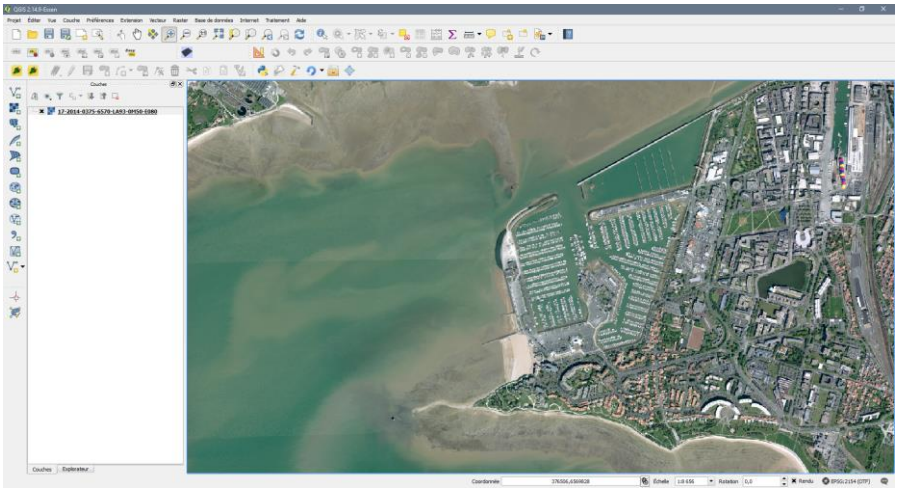
EXERCICE

AJOUTER UN RASTER TUILÉ DANS LA CARTE

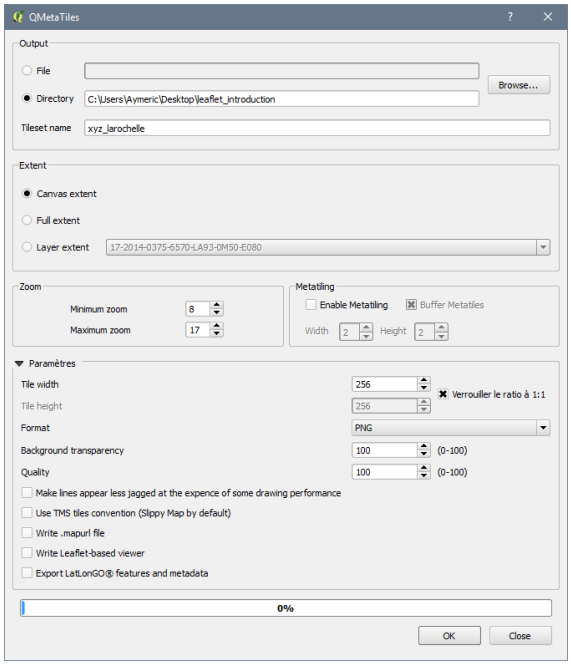
CRÉER UN TUILAGE A PARTIR DE QGIS

Nous allons créer un tuilage raster sur une BD Ortho de l'IGN sur le quartier des Minimes dans la ville de La Rochelle. Tout d'abord, il faut **créer le projet QGIS** et y **ajouter la photographie aérienne**.

- **Placez-vous à l'échelle** sur la photo de manière à vous fixer sur la zone exacte de la photo sur laquelle le tuilage devra être généré :



- Ouvrez le plugin **QMetaTiles** (installez-le si ce n'est pas encore fait) et renseignez les paramètres suivants :



Directory : Répertoire dans lequel sera généré le dossier résultat du tuilage, choisissez votre dossier **leaflet_intro**.

Tileset name : Nom du dossier dans lequel sera créer le tuilage, nommez le **xyz_larochelle**.

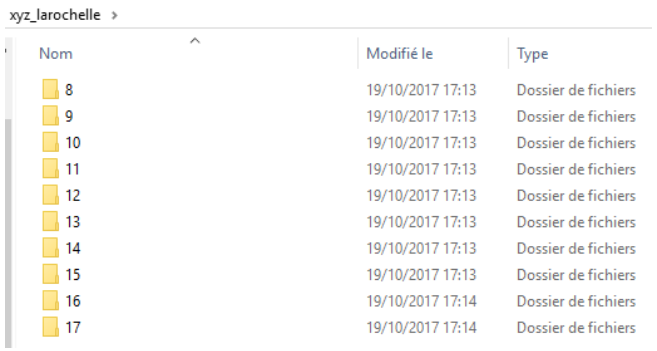
Cochez **Canvas extent** pour prendre l'emprise courante de la carte dans QGIS.

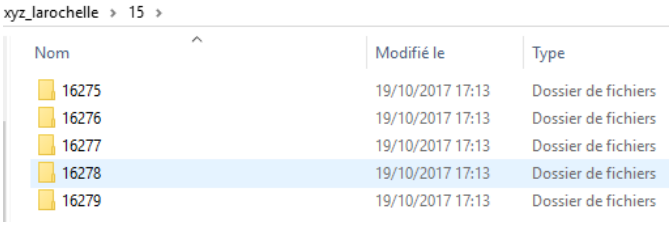
Minimum zoom : **8**

Maximum zoom : **17**

Laissez les autres paramètres par défaut (dalles de 256 pixels en PNG avec la transparence)

Une barre de chargement vous affichera la progression de la génération du tuilage ainsi que le nombre de fichiers qui seront créés pour composer le tuilage. **Vérifiez le contenu** du répertoire :





Chaque arborescence de dossier correspond à un nombre **z/x/y.**

AJOUTER LE TUILAGE A LA CARTE

Le tuilage maintenant généré, il va falloir ajouter les lignes de code nécessaires à l'appel de dossier de fichiers rasters. Pour cela, il va suffire de copier/coller l'instruction d'ajout du **fondTopo** déjà présent.

- **Ajoutez les lignes de codes** suivantes et créez la couche **fondOrtho** à la suite du premier fond de plan :

```
fondOrtho = L.tileLayer('xyz_larochelle/{z}/{x}/{y}.png', {
  maxZoom: 17
}).addTo(map);
```

- Rafraichissez la page pour constater le résultat. Votre orthophoto devrait être superposée au fond de plan topo.

COURS

LE FORMAT GEOJSON

GeoJSON (de l'anglais Geographic JSON) est un format open source de stockage d'ensemble de données géographiques **vectorielles** utilisant la norme javascript JSON.

Il permet de décrire des données par leur **géométrie**, de type point, ligne ou polygone, ainsi que l'ensemble des **attributs** qui y sont rattachés.

Il est de plus en plus utilisé dans les technologies web et notamment par Leaflet et OpenLayers. Ce format est facilement lisible et parcourable par le langage javascript.

```
{ "type": "Polygon",
    "coordinates": [
        [[30, 10], [40, 40], [20, 40], [10, 20], [30, 10]]
    ]
}
```

QGIS permet de convertir n'importe quel format SIG vecteur (ESRI Shapefile, Mapinfo Tab, PostGIS, etc.) en **GeoJSON** nativement, sans installation de plugin préalable. Un simple **clic droit/Enregistrer sous** sur la couche suffit pour convertir en GeoJSON.

EXERCICE

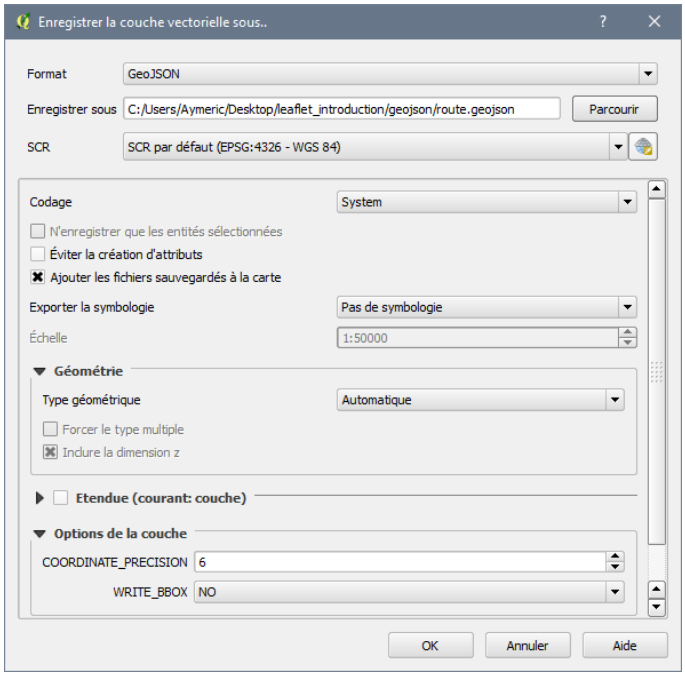
AJOUTER UNE COUCHE VECTEUR DANS LA CARTE

CRÉER UN FICHIER GEOJSON A PARTIR DE QGIS

Nous allons créer **3 couches vecteur** au format GeoJSON à partir de QGIS, une de chaque type (point, ligne, polygone). La procédure est identique pour les trois. Nous allons commencer par la couche de lignes :

- Ouvrez le fichier **ROUTE_MINIMES.SHP** dans QGIS

- Faites un **clic droit** sur la couche puis **Enregistrer sous…** et remplissez les options suivantes :



Format : **GeoJSON**

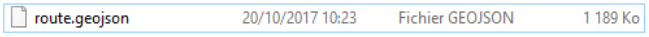
Emplacement : dossier **leaflet_intro/geojson/route.geojson**

SCR : **WGS84 (4326)**

COORDINATE_PRECISION : **6**

NB : La **précision des coordonnées** correspond au nombre de chiffres après la virgule pour chaque coordonnée x et y. Plus elle est précise plus le fichier sera volumineux.

Plus le fichier sera volumineux, plus il sera long à charger dans la Leaflet.



- Maintenant que le fichier GeoJSON est créé, **ouvrez-le** avec votre éditeur de texte pour constater la composition du fichier

Vous remarquerez que l'ensemble des **attributs** sont présents, ainsi que la **géométrie**.

NB : Il est très important de ne faire figurer que les attributs qui seront utiles dans l'application cartographique !! Pour un soucis de rapidité d'affichage de la donnée.

Dans l'exemple ci-dessus, nous avons l'ensemble des attributs avec des géométries à 6 décimales, le fichier fait : **1 189 Ko**.

Le même fichier avec uniquement **un identifiant** et **le nom de la voie** fait : **406 Ko**

Si une couche dépasse un certain poids, par exemple **1Mo**, il est conseillé de passer en base de données spatiale type PostGIS plutôt qu'en mode fichiers comme nous le faisons ici car **1Mo** peut mettre **2sec à charger** voire plus en fonction du débit.

Cela permettra de **ne charger que les données de l'emprise cartographique** à visualiser au lieu de toute la couche.

- Créez une version **route_light.geojson** en supprimant les colonnes inutiles du fichier shape d'origine et en exportant à nouveau la couche en GeoJSON.

COURS

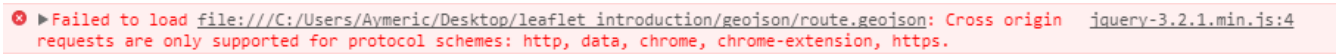
MÉTHODE AJAX

Pour ouvrir le fichier GeoJSON dans la carte Leaflet, il va falloir stocker le contenu du fichier dans une variable javascript. Pour se faire, il faut accéder au fichier en **Ajax**.

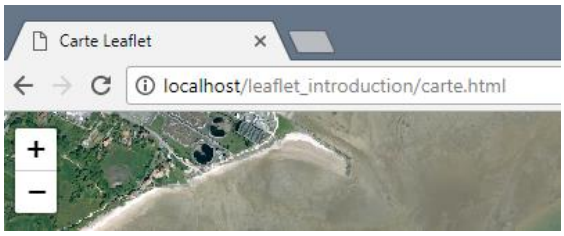
Ajax est une méthode informatique utilisée dans le web avec javascript pour interroger des fichiers de manière asynchrone. C'est-à-dire que notre fichier geoJSON va se stocker dans une variable grâce à une interrogation server au cours du chargement de la page.

Pour pouvoir faire des appels Ajax, notre **application ne peut pas être appelée en mode fichier** directement par le disque dur avec une adresse de type file:///C:/Users/, il faut obligatoirement passer par un serveur web pour avoir une adresse http://.

Voici l'erreur qui apparait lorsque l'on tente d'ouvrir un fichier geoJSON en local :



Pour pouvoir faire de l'Ajax, placez votre dossier **leaflet_intro** sur un serveur web local, par exemple : **uWamp**, **easyPHP**, **Wamp**. Dans le dossier www et accédez-y par l'adresse du serveur :



EXERCICE

AJOUTER UNE COUCHE VECTEUR DANS LA CARTE (SUITE)

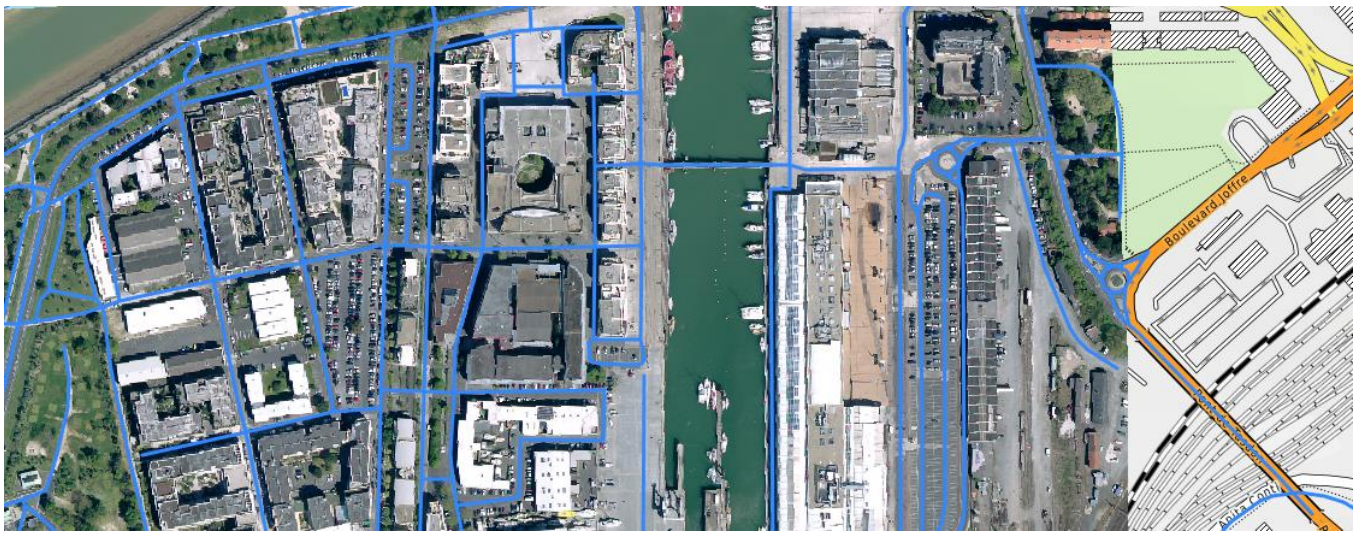
AFFICHER UNE COUCHE DE LIGNES

Maintenant que votre serveur est prêt pour pointer vers des fichiers **GeoJSON**, il va falloir créer la couche dans le code javascript pour faire l'appel Ajax qui ira récupérer les données.

- Ajoutez la création d'une couche vecteur **vRoute** en écrivant les lignes de code suivantes à la suite des deux fonds rasters :

```
$.getJSON(
  "geojson/route_light.geojson",
  {},
  function(data) {
    vRoute = L.geoJSON(data);
    vRoute.addTo(map);
  }
);
```

- Testez le bon fonctionnement de la couche vecteur en chargeant votre page sur le serveur **localhost**, vous devriez visualiser la couche de routes en bleu :

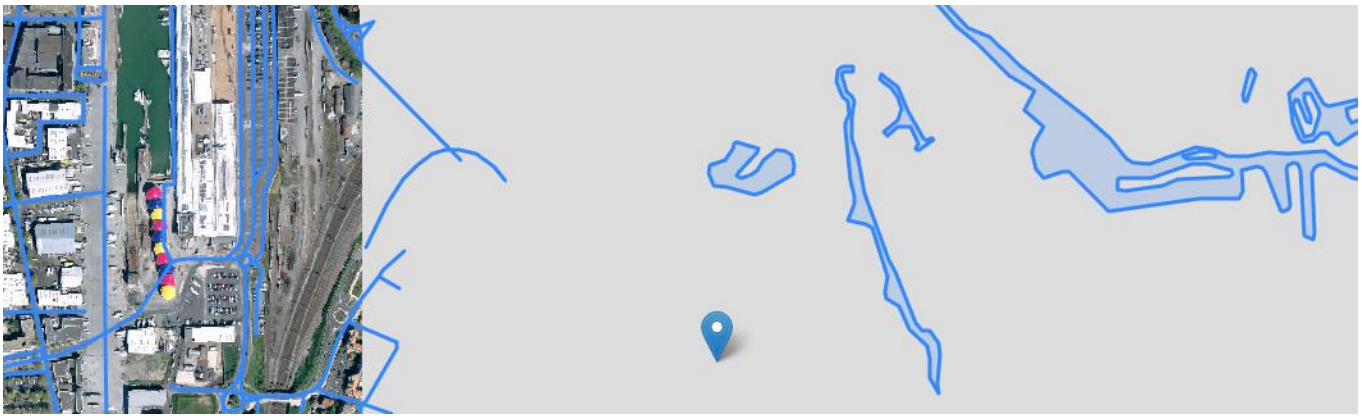


La **couleur bleue** est mise **par défaut** par Leaflet, il est bien entendu possible de modifier le style de toutes les couches vecteurs. Nous verrons cela dans le prochain chapitre.

AJOUTER DES COUCHES DE POINTS ET POLYGONES

Faites le même travail que précédemment pour ajouter les couches de points **LIEU_DIT.SHP** et **SURFACE_EAU.SHP**.

Vous devriez avoir le résultat suivant sur la carte une fois les trois couches vecteurs ajoutées :



Les styles des points sont représentés avec une icône bleue par défaut également, tout comme les polygones avec un contour et un remplissage bleu avec de la transparence.

Voici l'état du code source du fichier **carte.js** à ce niveau de l'exercice.

```
$(document).ready(function() {
    /* Création de la carte avec coordonnées GPS latitude, longitude et niveau de zoom */
  map = L.map('map').setView([46.15197, -1.15385], 17);

    /* Ajout d'un fond de plan Topo style IGN Scan25 */
  fondTopo = L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {
    maxZoom: 17
  }).addTo(map);

    /* Fond de plan Ortho */
  fondOrtho = L.tileLayer('xyz_larochelle/{z}/{x}/{y}.png', {
    maxZoom: 17
  }).addTo(map);

    /* Couche de lignes : route */
  $.getJSON(
    "geojson/route_light.geojson",
    {},
    function(data) {
      vRoute = L.geoJSON(data);
      vRoute.addTo(map);
    }
  );

    /* Couche de points : lieu-dit */
  $.getJSON(
    "geojson/lieu_dit.geojson",
    {},
    function(data) {
      vLieudit = L.geoJSON(data);
      vLieudit.addTo(map);
    }
  );

    /* Couche de polygones : surface en eau */
  $.getJSON(
    "geojson/surface_eau.geojson",
    {},
    function(data) {
      vSurfaceEau = L.geoJSON(data);
      vSurfaceEau.addTo(map);
    }
  );

});
```

CRÉATION D'UN GESTIONNAIRE DE COUCHES

Un **gestionnaire de couches** va permettre de faire apparaitre sur la carte, une liste de toutes les couches présentes et à l'utilisateur de pouvoir **cocher/décocher les données** qu'il veut voir apparaitre.

Leaflet embarque un composant qu'il appelle **Layer control** qui permet de définir les couches ainsi que leur nom dans la légende.

- Dans le fichier **carte.js** à la suite de la création de la carte, ajoutez un Layer control nommé **lControl** :

```
map = L.map('map').setView([46.15197, -1.15385], 17);
lControl = L.control.layers().addTo(map);
```

Dans le Layer control de Leaflet, chaque couche peut être définie soit comme un **baseLayer** soit comme un **overlay**. Rasters ou vecteurs peuvent tous être l'un ou l'autre, cela dépend du choix d'affichage que l'on souhaite faire. Pour bien comprendre la différence, nous allons faire les deux, en mettant les **rasters en baseLayer** et les **vecteurs en overlay**

- Après chaque **création de couche** la liste des couches du layer control doit être mise à jour grâce aux instructions suivantes :

```
fondTopo = L.tileLayer('https://{s}.tile.opent
  maxZoom: 17
});
lControl.addBaseLayer(fondTopo, "Carte topo");
```

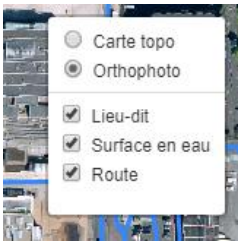
```
lControl.addBaseLayer(fondOrtho, "Orthophoto");
```

```
function(data) {
  vLieudit = L.geoJSON(data);
  vLieudit.addTo(map);
  lControl.addOverlay(vLieudit, "Lieu-dit");
}
```

```
lControl.addOverlay(vRoute, "Route");
```

```
lControl.addOverlay(vSurfaceEau, "Surface en eau");
```

- Testez l'exécution du script pour constater l'affichage d'une nouvelle icône dans le coin haut / gauche de l'écran permettant de choisir les couches à afficher



Vous pouvez remarquer la différence entre les **baseLayers** et les **overlays**. Les **baseLayers** sont des **radio boutons**, on ne peut en activer qu'un seul à la fois. Alors que les **overlays** sont des **cases à cocher**.

4

STYLE DES DONNEES

MARKERS ET ICONES

- Choisir le style des couches vecteurs
- Attribuer des icônes, markers aux points

COURS

STYLE DES COUCHES VECTEURS

COUCHE DE LIGNES

Les couches de lignes possèdent différentes propriétés de style, à savoir :

- La couleur
- L'épaisseur
- L'opacité
- Le type de ligne (tiret, point, etc.)

COUCHE DE POLYGONES

Les couches de polygones possèdent différentes propriétés de style, à savoir :

- La couleur du remplissage
- La couleur du contour
- L'épaisseur du contour
- L'opacité du remplissage
- L'opacité du contour
- Le type de contour (tiret, point, etc.)

COUCHE DE POINTS

Les couches de points ne possèdent pas directement de propriétés de style. En effet, un point n'a pas de représentation dans l'espace, il doit être associé à un symbole. Il est possible de le remplacer par un cercle sur la carte possédant des propriétés :

- Le rayon du cercle
- La couleur du remplissage
- La couleur du contour
- L'épaisseur du contour
- L'opacité du remplissage
- L'opacité du contour

On peut également le symboliser par une **icône de type image** avec elle aussi des propriétés différentes :

- Le chemin vers l'image (URL)
- Le point d'ancrage de l'image aux coordonnées sur la carte
- La taille de l'image
- Etc.

EXERCICE

STYLISER LES COUCHES VECTEUR DANS LA CARTE

COUCHE DE LIGNES

Modifier le code permettant l'affichage de la couche des routes en ajoutant des **options** {} après le passage du paramètre data contenant le geoJSON, le deuxième paramètre contiendra le style comme suit :

```
vRoute = L.geoJSON(data, {
  style: function() {
    return {
      color:"#FF0000",
      weight: 2,
      opacity: 0.7,
      dashArray: '15,8'
    };
  }
});
vRoute.addTo(map);
```

La couleur (color) est une chaine de caractère du code couleur en hexadécimal.

L'épaisseur (weight) est un nombre entier en valeur de pixels

L'opacité (opacity) est un nombre décimal compris entre 0 et 1 (0 = transparent, 1 opaque)

Le style de trait (dashArray) est une chaine de caractère donnant la longueur du trait et l'espacement en pixels séparés par une virugle. Si l'on souhaite **une ligne en trait plein**, il suffit de **retirer la ligne dashArray.**

COUCHE DE POLYGONES

Pour les couches de polygone, le principe est le même, ajoutez les propriétés de style telles qu'indiquées pour la couche de ligne et ajoutez également les propriétés manquantes, à savoir :

```
color:"#a9d1f5",
fillColor: "#a9d1f5",
weight: 0, // 0 = Pas de contour
opacity: 1,
fillOpacity: 1
```

Il suffit de mettre la valeur 0 à la propriété weight pour ne pas afficher de contour sur une couche de polygones, la propriété opacity s'en trouve inutile dans ce cas-là.

COUCHE DE POINTS

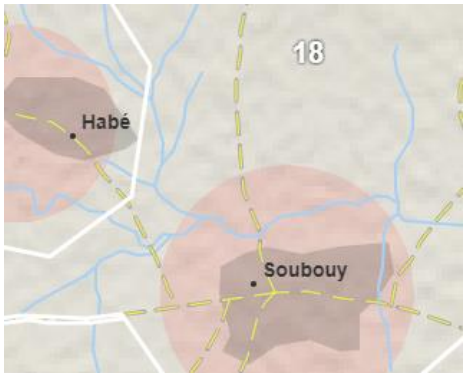
Pour les couches de points, comme vu dans le cours précédemment, par défaut un marker bleu est positionné car les points n'ont pas de représentation dans l'espace. Il faut donc y associer un symbole ou une icône.

La procédure est donc un peu plus complexe, notamment par l'utilisation de la propriété **pointToLayer**.

Ajout d'un cercle :

```
vLieudit = L.geoJSON(data, {
  pointToLayer: function (feature, latlng) {
    return L.circleMarker(latlng, {
      radius: 8,
      color:"#0000FF",
      fillColor:"#FFFFFF",
      weight: 3,
      opacity: 1,
      fillOpacity:0.8
    });
  }
});
vLieudit.addTo(map);
```

Le marker de type cercle est parfois très utile pour afficher certaines couches de points, ne nécessitant pas la création d'une image. Exemple d'utilisation des couches de points :



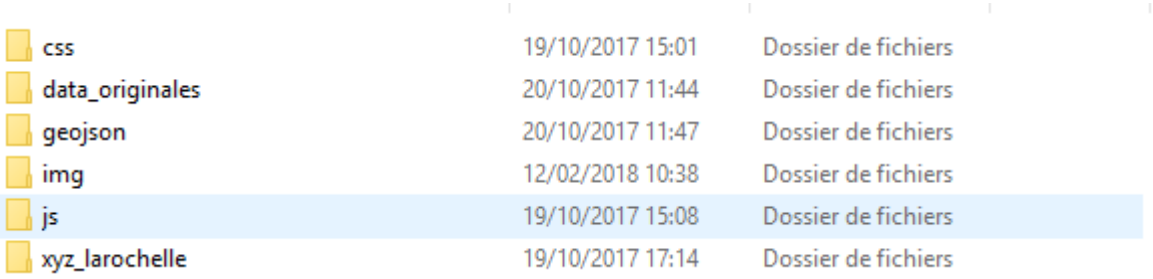
Nous verrons par la suite l'affichage d'étiquettes

Ajout d'une icône (marker) :

Allez chercher une icone de votre choix sur Google Image en tapant la recherche « marker png » ou « icon png ». De nombreux sites internet fournissent des bibliothèques d'icones comme le site http://flaticon.com/.

PNG est le format d'image permettant d'avoir des images sans compression et avec transparence. Les JPG ne sont pas conseillés.

Enregistrez l'image dans un dossier img à la racine de votre dossier leaflet.



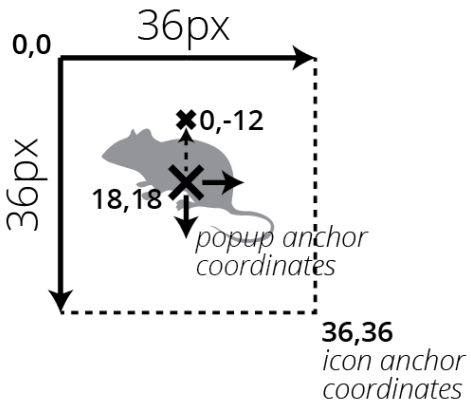
Le fichier est nommé **icon.png**

Dans le code javascript, remplacez les lignes de style de type cercle par le code ci-dessous pour afficher l'icône.

```
vLieudit = L.geoJSON(data, {
  pointToLayer: function (feature, latlng) {
    return L.marker(latlng, {
      icon: new L.Icon({
        iconUrl: './img/icon.png',
        iconSize:    [28, 28],
        iconAnchor:  [14, 14]
      })
    });
  }
});
vLieudit.addTo(map);
```

COURS

NB : L'ancre de l'icône est très importante, elle permet de bien montrer l'emplacement exact du point. Sans les bons paramètres, votre icône va « se décaler » sur la carte à chaque zoom se qui rend les données affichées **fausses** visuellement !!



Pour une icône de ce type, l'ancre est au niveau de la pointe du marker, **soit tout en bas et à la moitié de la largeur** :

Si la taille de l'icône est 50x80 pixels, le code javascript et l'ancre calculée sont les suivants :

iconSize : [50, 80],

iconAnchor : [25,80]

Pour un cercle parfait, l'ancre est donc à la moitié de la hauteur et moitié de la largeur.

COURS

ANALYSES THEMATIQUES SELON UN ATTRIBUT

Tout comme on pourrait le faire dans un logiciel SIG, il est possible de créer des analyses thématiques basées sur les valeurs d'un ou plusieurs attributs.

Les attributs enregistrés dans le geoJSON sont parcourables par le javascript, il suffit donc de faire des tests conditionnels pour renvoyer une couleur pour chaque différente valeur d'attribut.

EXERCICE

ANALYSE THEMATIQUE SUR LA COUCHE POLYGONE

La couche des **surfaces en eau** contient un attribut **REGIME** donnant la nature du régime des plans d'eau, à savoir : Intermittent ou Permanent. Nous allons afficher une couleur différente pour ces plans d'eau.

```
vSurfaceEau = L.geoJSON(data, {
  style: function(f) {
    if (f.properties.REGIME == "Intermittent") couleurR = "#0000FF";
    else if (f.properties.REGIME == "Permanent") couleurR = "#a9d1f5";
    else couleurR = "#a9d1f5";
    return {
      color: "#a9d1f5",
      fillColor:couleurR,
      weight: 0, // 0 = Pas de contour
      opacity: 1,
      fillOpacity: 1
    };
  }
});
```

En ajoutant les deux tests conditionnels, nous allons attribuer une couleur différente à la variable **couleurR**, qui sera utilisée dans la propriété fillColor du style.

Le résultat est le suivant :

EXERCICE EN AUTONOMIE

ANALYSE THEMATIQUE SUR LA COUCHE POINT

Pour cet exercice, vous allez devoir mettre une icone différente sur les lieux-dits :

- Icone sur les lieux-dits dont la NATURE est de type « Lieu-dit habité »
- Cercle de couleur noire et bordure blanche pour les lieux-dits dont la NATURE est de type « Quartier »