TD – D3.js avancé

Objectif : la séance a pour objectif de mettre en place une carte interactive des gares SNCF en France affichant des informations sur leur fréquentation.

Les jeux de données :

Départements.json : https://raw.githubusercontent.com/Lawiss/d3js/master/departments.json

gares.json : https://raw.githubusercontent.com/Lawiss/d3js/master/gares.json

---

Rendez vous sur le site (trame de départ) : https://codepen.io/idriss_/pen/mzzdMr.
Cliquez sur "Fork" pour commencer à coder.

Dans la partie HTML :

1. Insérer une balise script dans le header du code HTML pour insérer le lien de la bibliothéque d3.js : *https://d3js.org/d3.v5.min.js*
2. Ajouter dans le corps du fichier une balise div avec comme id: *carte*

A présent, on ne travaillera que dans la partie Javascript :

3. Créer un élément SVG à l'aide de D3 avec les valeurs de width et height que vous aurez définis au préalable.

Partie 1 : Affichage de la carte de France

1. Le code suivant définit les objets nécessaires pour configurer la projection géographique de nos données sur le plan 2D de notre page. A l'aide des méthodes center, définissez le centre de la projection à la latitude et la longitude suivante : [2.454071, 46.279229] afin que notre projection soit centrée sur la France. Vous pouvez également régler l'échelle avec la méthode scale.

```
//
const pathD3 = d3.geoPath();
const projection = d3.geoConicConformal()
```

2. Assigner la projection à l'objet pathD3 avec la méthode du path : projection()

3. Ajouter un élément *g* (groupe) au svg créé précédemment : append()

4. Charger les données geojson du fichier departement.json : d3.json() et dessiner les données des départements. Chaque départements est défini dans le JSON par un path (polygone complexe) en coordonnées géographique.
   A l'aide de la fonction pathD3 définie précédemment, on peut projeter ces coordonnées dans un plan 2D pour pouvoir ensuite les attribuer aux éléments path de notre SVG.

5. Customiser le rendu de la carte avec les attributs : fill et stroke.

Partie 2 : Affichage des gares sur la carte

Le fichier gares.json contient les coordonnées et informations de plusieurs milliers de gares françaises. Les informations de la gare sont contenues dans "properties" tandis que les coordonnées sont dans "geometry" puis "coordinates".

1. Ajouter un élément *g* (groupe) au svg créé précédemment : append()

2. Charger les données geoJSON : gares.json.

3. Projeter ces données. Utilisez cette fois la méthode projection() définies précédemment pour obtenir les coordonnées de chaque cercle. Le rayon d'un cercle est défini par l'attribut "r".

Partie 3 : Affichage des informations au survol

1. Insérer un élément div de classe **tooltip** dans le corps HTML en passant par D3.js et stockez-le dans une variable nommée div. Utilisez la méthode style() pour rendre invisible ce div:

```
 .style("display", "none");
```

2. Écrire une fonction qui prend en entrée un tableau contenant la fréquentation d'une gare sur trois ans et retournant la moyenne de la fréquentation.

3. Créer une fonction showInfo(gare) qui prend en entrée un objet gare du geoJSON, qui calcule la moyenne de sa fréquentation et qui remplit l'élément div créé précédemment avec le nom de la gare, sa fréquentation moyenne et enfin qui fait apparaître l'élément. Pour vous aider :

```
// Change le style de l'élément "div" pour le faire apparaître
div.style("display","block")

//Permet de modifier le contenu HTML de l'élément
div.html("Nom de la gare : Lyon Vaise")
```

4. Créer une autre fonction hideInfo() qui fait disparaître l'élément div.

5. Dans l'instruction qui affiche les points ajouter deux évènements de type: mouseover qui appelle showInfo et mouseout qui appelle hideInfo().

6. Modifier votre fonction showInfo() pour positionner l'élément div par rapport à la position du curseur. Pour vous aider:

```
// Positionne l'élément à une distance de 50 pixel du haut de la page et
30 pixel de la gauche de la page
.style("left", "30px")
.style("top", "50px")
// Pour récupérer la position sur X et Y du curseur de la souris au
moment où l'évènement est lancé:
d3.event.pageX
d3.event.pageY
```

- **Bonus** : ajoutez la possibilité de zoomer sur la carte en appelant la fonction d3.zoom.on("zoom",function) à l'aide de la méthode call() au moment de la création du SVG. Function est votre fonction callback qui va venir modifier les départements et les cercles des gares. Pour vous aider :

```
// Pour récupérer les information à propos du zoom et de la translation
au moment où l'événement est levé sous la forme
{x:value,y:value,k:value}:
d3.event.transform
// Pour obtenir seulement le facteur multiplicateur du zoom:
d3.event.transform.k
//applique une transformation à l'élément (translation de 10 px selon x,
20 selon y et multiplication de la taille par 2):
.attr("transform","translate(10 20)  scale(2)")
```

Partie 4 : Affichage d'une fenêtre modale au clic

Dans cette partie vous allez manipuler les éléments suivants qui se trouvent déjà dans trame HTML :

```
<div class="modal-contenu">
      <span class="close">&times;</span>

      <div id="nom-gare"></div>
      <div id="conteneur-svg-modal"></div>
```

```
</div>
```

1. Créer une fonction openModalWindow() qui prends en paramètre un objet JSON gare. Cette fonction devra changer le style de l'élément ayant pour id "modal" pour le faire apparaître. Elle devra également modifier l'élément ayant pour id "nom-gare" pour y insérer le nom de la gare.

2. Créer une fonction closeModalWindow() qui changera le style de de l'élément ayant pour id "modal" pour le faire disparaître.

3. Ajouter un événement au clic sur l'élément ayant pour classe "close" appelant la fonction closeModalWindow().

4. Créer une fonction drawChart() qui prends pour paramètre un objet JSON gare et qui va créer un diagramme à barre représentant la fréquentation des voyageurs pour la gare sur les années 2014,2015 et 2016. Pour cela, définir un nouvel élément svg dans l'élément ayant pour id "conteneur-svg-modal". Utilisez les dimensions suivantes :

```
//Dimensions du SVG:
svgWidth=960;
svgHeight = 500;

//Les dimensions suivantes serviront pour définir la longueur des axes à
l'aide de la méthode range.
margin = { top: 50, right: 50, bottom: 30, left: 70 };
widthChart = svgWidth - margin.left - margin.right;
heightChart = svgHeight - margin.top - margin.bottom;

//Domaine de définition de l'axe x:
annees=["2014","2015","2016"]
```

   a. Créer les échelles des axes : axe x avec une échelle d3.scaleBand() et axe y avec une échelle d3.scaleLinear(). N'oubliez pas de définir un range() et un domaine de définition avec domain().

   b. Utiliser les méthodes d3.axisBottom() et d3.axisLeft() pour créer les axes avec les échelles définies précédemment.

   c. Ajoutez les axes au SVG en appliquant les transformations suivantes :

```
modalSVG.append("g")
      .attr("transform","translate(0,"+heightChart+")")
      .call(xAxis)

modalSVG.append("g")
      .call(yAxis)
      .attr("transform","translate("+margin.left+",0)")
```

d. Créer et ajoutez au SVG les trois rectangles à partir des données contenues dans l'attribut .properties.VOYAGEURS. Pour positionner le rectangle et régler sa taille utilisez les axes x et y créés précédemment :

```
.attr("x",function(d,i){ return x(annees[i]);})
.attr("y",function(d,i){return y(d)})
.attr("height",function(d,i){return Math.abs(heightChart-y(d))})
```