BUT3 VCOD R5.VCOD.07
Prog. Web Visualisation
TP 7 JavaScript & D3.JS

« Exercices du cours de l'ICOM Lyon 2 – D3.JS Initiation – Niveau A. MARCHAL, R. JACQUOT, D. NGUYEN & H. BOUJEDDAYN »

Exercice 1. Création d'un diagramme en barres

Sujet : Le but de l'exercice est de créer un diagramme en barres à partir de données simples.

Pour cela, copier le code html suivant dans un fichier que vous appellerez « TP7-Exo1-D3.html » :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <!-- Ici on utilise d3js -->
  <script src="https://d3js.org/d3.v5.min.js"></script>
  </head>
 <body>

 </body>
</html>
```

et le code CSS de base à copier dans un fichier « TP7-D3.css » :

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

1. Étape 1 : Création d'une table de données

Il faut commencer par créer le jeu de données avec lequel nous allons travailler :
Dans la partie JavaScript, créez une table de données avec les valeurs 15, 36, 18, 29, 23, 38, 12, 30, et 11. Utilisez des [] pour délimiter la table.

2. Étape 2 : Création d'un svg

Avant de commencer à créer les barres, il faut créer l'élément svg qui servira de canevas. Pour cela, créez la variable « svg » à l'aide de la commande `d3.select()` en sélectionnant le « body » du html et en y ajoutant (`append()`) le « svg ».
Ajoutez à cette variable des attributs de hauteur et de largeur, à l'aide de :
`".attr("height", "pourcentage de la page à utiliser en hauteur")"`
et `".attr("width","pourcentage de la page à utiliser en largeur")"`
Ici on souhaite que le svg prenne la totalité de la page, c'est à dire 100%, que l'on note **"100%"** dans l'attribut.

3. Étape 3 : Création des rectangles

Il va maintenant falloir créer les barres sur le svg créé plus haut :
À partir de la variable svg, sélectionnez un ensemble de rectangles « `rect` » avec `selectAll()`, une fonction de d3.js.

S'il trouve des rectangles, javascript va les renvoyer, sinon ça sera vide. Ici on sélectionne l'élément avant de travailler dessus.

Joignez-y les données à l'aide de `.data(vos-donnees).enter().append("rect")`
Enter `.enter()` *et* `.append(« rect »)` *permet d'ajouter un rectangle pour chaque valeur dans les données.*

Comme pour l'étape 2, attribuez des valeurs de hauteur et largeur (par ex. 250 en hauteur et 40 en largeur) et des coordonnées x et y (par exemple 25 et 50) :
("x", "distance par rapport au côté gauche en nombre de pixels")
("y", "distance par rapport au haut en nombre de pixels")

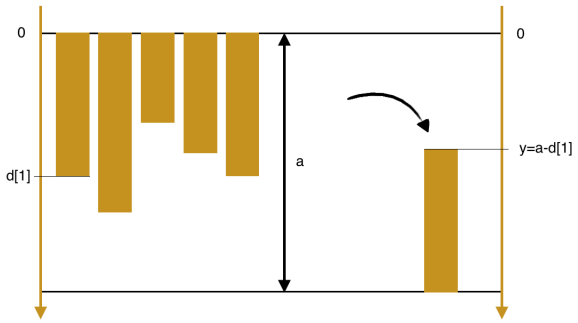
- **Visualiser votre graphique.** Il ne semble afficher qu'une seule barre. En réalité, elles sont superposées les unes sur les autres, car il n'y a qu'une seule valeur de position x pour toutes les barres.

Pour changer cela, il faut remplacer les valeurs de "x" par une fonction de deux paramètres : d (la donnée) et i (l'index de la donnée).
Une fonction prend la forme `function(d, i) {return --vos-valeurs--;}`.
Ici, on souhaite que la fonction renvoie pour chaque index des données la valeur de x ou l'on doit placer le rectangle. Il faut donc que la fonction renvoie i multiplié par une valeur de votre choix (pour jouer sur l'écartement entre les rectangles du diagramme). On peut faire en plus de cette multiplication une addition pour éviter que le premier rectangle soit collé au côté gauche du svg. Trouvez donc une fonction adéquate pour bien visualiser les 9 rectangles.

- On observe maintenant 9 rectangles de même hauteur : la hauteur des rectangles ne correspond pas aux données. Il faut donc dans l'attribut hauteur passer par une fonction qui renvoie cette fois tout simplement d.

On remarque que les valeurs de d ne sont pas assez grandes comparativement à la page. Il suffit donc de multiplier d par un facteur un facteur de votre choix dans la fonction (pour que tous les rectangles entrent dans votre fenêtre et soient lisibles).

Reste un autre problème : comme le navigateur lit les pages depuis le haut à gauche jusqu'en bas à droite, les diagrammes sont "à l'envers" par rapport à notre sens de lecture habituel. Pour régler ce problème, il faut jouer sur la position par rapport au haut de la page, c'est à dire sur "y". Pour "y", on entre une nouvelle fonction dépendant des données d. Le schéma suivant peut vous donner un indice quant à la fonction à indiquer. Le paramètre a est un nombre de pixels à trouver pour que cela corresponde à votre fenêtre d'affichage.



4. Étape 4 : Mise en page du graphe

- Pour mettre en page le graphe, il faut passer par le CSS. Avant cela, il faut nommer les rectangles dans le javascript pour simplifier la démarche. On ajoute donc un nouvel attribut dans le svg : `.attr("class", "barre")`
  On nomme ainsi les rectangles “barre”.

- On souhaite changer la couleur des barres. On utilise pour cela la partie CSS. Voici le code à ajouter :

```
.barre {
 fill : ?
}
```

Dans “fill”, il suffit d’indiquer en lettres la couleur : red, pink, purple, blue… Une liste complète des couleurs se trouve sur ce site http://stylescss.free.fr/couleurs.php
On peut également mettre une couleur qui change lorsque l’on passe la souris sur une barre, en ajoutant en dessous le code suivant :

```
.barre:hover {
 fill: ?
}
```

On choisit de la même manière la couleur en la renseignant dans fill.

Comme on peut le voir, cela change la couleur en prenant celle désignée dans le CSS. Il est aussi possible de sélectionner la couleur directement dans le code js avec `.attr(« fill », « -couleur choisie- »)`.

5. Étape 5 : Ajout des étiquettes de données

La dernière étape consiste à ajouter les étiquettes de valeurs aux barres afin de rendre le diagramme lisible fonctionnellement.

Comme dans le cas de l’affichage du rectangle, sur le svg il faut sélectionner un array « text » avec d3.selectAll() et y joindre les données à l’aide de

`.data(-vos-donnees-).enter().append("text")`

Comme pour la création des rectangles, sauf qu’au lieu de “`rect`”, on a “`text`”. Copiez-collez donc à la suite tout le code de création des rectangles (étape 3) en remplaçant “rect” par “text”, et “barre” par “texte”.

Il faut ensuite préciser le texte que l'on souhaite écrire, à l'aide du code suivant : `.text(fonction)`
On souhaite renvoyer les valeurs des données, il faut donc passer par une fonction qui prend en entrée les données et qui renvoie ces mêmes données.

Pour mieux centrer les étiquettes sur les rectangles, il suffit de jouer avec la valeur du décalage des x. On peut reprendre les mêmes valeurs que celles utilisées pour les rectangles, ou les changer un peu pour plus de lisibilité.

Si l'on veut changer le style du texte, comme on a pu le faire pour les rectangles, on procède exactement comme on l'a fait pour les rectangles au début de l'étape 4. On a nommé le texte "texte", on peut donc comme auparavant ajouter le code dans CSS pour le changement de couleur.

Exercice 2. Création d'un diagramme en secteurs

Le tableau suivant représente la répartition des types de missions du BUT SD en 2024-25 :

Type de mission	Nombre d'étudiants
Statistiques études	8
Informatique	1
BI/ Décisionnel	15
Autres	2

L'objectif de l'exercice est de représenter cette répartition sous forme d'un diagramme circulaire ou diagramme en secteurs (camembert).

1. Étape 1 : Création du tableau de données et d'un svg

Dans un nouveau fichier, « TP7-Exo2-D3.html », Reprendre le code de création du tableau de données et du SVG présentés dans l'exercice précédent en choisissant les attributs de hauteur et de largeur correspondants ainsi que la couleur du background de votre choix.

2. Étape 2 : Création du tableau des segments

Le but est de construire le diagramme circulaire "Pie chart" traduisant la répartition des alternants par type de mission. Pour cela, il faut d'abord créer une variable **data** qui va contenir les données :

```
var data = [{Type_de_mission : "", Nombre_etudiants: valeur}, ...]
```

Ensuite, il faut utiliser le Layout standard Pie chart. Contrairement à ce que l'on peut penser, la fonction pie ne trace pas directement un graphe mais elle fournit des informations pour le tracer. Elle va ajouter des informations à notre dataset : l'angle de départ et l'angle de fin. Il faut donc encore tracer tous les arcs, leur assigner un texte, une couleur, etc.

Pour ce faire il faut définir les 3 variables suivantes :

- La variable **base_diagramme** pour avoir l'angle de départ et l'angle de fin, en utilisant la fonction de génération du diagramme de type Pie :

  var base_diagramme = d3.pie()

Les données sont de type "object", donc on va utiliser la fonction value pour définir une fonction **d** qui va retourner la valeur des nombres d'étudiant du tableau de données.

```
var base_diagramme = d3.pie().value(function(d){return
d.Nombre_etudiants;})
                (data);
```

- La variable **segments** pour définir un arc qui sert à segmenter le nombre d'étudiants par type de mission, en utilisant la fonction d3.arc()

Ajouter à cette fonction des attributs du diagramme (rayon, angle...). Tester plusieurs valeurs pour voir les changements.

Ajouter à cette fonction des attributs du diagramme (rayon, angle...). Tester plusieurs valeurs pour voir les changements.

```
var segments =
d3.arc()
.innerRadius(0)
.outerRadius(200)
.padAngle(.05)
.padRadius(50);
```

- La variable **sections** : Pour regrouper les différents segments et définir l'emplacement du diagramme sur le SVG. Pour cela il faut créer un groupe g et ajouter des paths :

```
var sections = svg.append("g").attr("transform", "translate(250,
250)")
                .selectAll("path").data(base_diagramme);
sections.enter().append("path")
              .attr("d", segments);
```

3. Étape 3 : Segmentation par couleur

Le diagramme obtenu représente l'ensemble des segments avec la même couleur, on veut maintenant modifier le diagramme de telle sorte que chaque segment soit représenté par une couleur spécifique.

Il faut d'abord définir la variable **couleur** en utilisant l'échelle ordinale des couleurs prédéfinies :

```
var couleur = d3.scaleOrdinal(d3.schemeDark2);
```

Pour choisir le nom de la palette, vous pouvez utiliser le code de la palette des couleurs prédéfinies :

* Pour des couleurs sombres :
  `d3.schemeDark2`
* Pour des couleurs claires :
  `d3.schemePastel1`
* Pour plus de codes couleurs, vous pouvez consulter le lien suivant :
  https://bl.ocks.org/pstuffa/3393ff2711a53975040077b7453781a9

Le but étant de remplir les segments avec les couleurs spécifiques, en fonction du nombre des étudiants, pour cela il faut rajouter un attribut fill à la variable section pour récupérer les valeurs de la variable **couleur** précédemment définie en la plaçant en tant qu'argument de la fonction attr().

```
var sections = svg.append("g").attr("transform", "translate(□,□)")
                  .selectAll("path").data(base_diagramme);
sections.enter().append("path")
                .attr("d", segments).attr("fill", function(d){return
couleur(--Compléter ici--);});
```

4. Étape 4 : Libellé des segments

Nous avons certes les segments représentés par des couleurs, par contre, nous n'arrivons pas à associer chaque couleur à sa variable. Il faut donc afficher sur le diagramme, le nombre d'étudiants correspondant à chaque segment.

Pour cela, il faut définir une variable **libelle** qui va contenir les libellés des segments, dans notre cas le nombre d'étudiant, et une variable **center** pour le positionnement des étiquettes dans leur bon emplacement.

```
var libelle = d3.select("g").selectAll("text").data(base_diagramme);
       libelle.enter().append("text").classed("inside",
true).each(function(d){
   var center = segments.centroid(d);
   d3.select(this).attr("x", center[0]).attr("y",
center[1]).text(d.data.Nombre_etudiants);});
```

Vous pouvez utiliser la balise <style> du code HTML pour la modification des attributs des caractères (police, taille, couleur...) comme l'exemple qui suit :

```
<style>
        text {
         font-size:20px;
         font-weight:bold;
         fill:white;
        }
 </style>
```

5. Étape 5 : Création de la légende

Afin de créer une légende pour le diagramme, copier le code ci-dessous et compléter les valeurs manquantes (□) avec les tailles et les variables correspondantes :

- Définir la position de la légende à afficher

```
var legends = svg.append("g").attr("transform", "translate(□,□)")
.selectAll(".legends").data(□);
```

```
var legend = legends.enter().append("g").classed("label",
true).attr      ("transform", function(d,i){return "translate(0," +
(i+1)*50 + ")";});
```

- Créer des symboles de la légende

```
legend.append("rect").attr("width",   20).attr("height",   20).attr("fill",   function(d){return
couleur(d.data.Nombre_etudiants);});
```

- Afficher le texte qui correspond à chaque rectangle

```
legend.append("text").classed("label",true).text(function(d){retur d.data.Type_de_mission;})
  .attr("fill", function(d){return couleur(d.data.Nombre_etudiants);})
  .attr("x", 30)
  .attr("y", 15);
```

- Ajoutez un titre à votre diagramme et placez le où vous le souhaitez, de manière à ce que le diagramme global soit lisible. Veuillez noter que vous pouvez changer sa couleur avec des attributs tels que attr("fill", "votre couleur") et modifier le style du texte avec "font-size" et "text-decoration".

```
svg.append("text")
        .attr("x", □)
        .attr("y", □)
        .attr("text-anchor", "middle")
        .style("font-size", "18px")
        .style("text-decoration", "underline")
        .text("□");
```

Vous devez obtenir le graphique suivant :