Cartographie / Javascript et SVG

Les données :

Déclarées comme tableau :

```
donnees= new Array(
["?",-3703,-350,0,0,0],
["Alsace",2483,2003,1734145,1624372,209],
["Lorraine",1833,1753,2310376,2305726,98],
…..
)
```

Ainsi pour récupérer le nom d'une région : donnees[1][0] donnera « Alsace »
Les coordonnées du centre de la région sont dans donnees[i][1] et donnees[i][2]
Les autres données suivent, ici par exemple population 1999, population 1990 et densité 1999
Le premier enregistrement correspond au rectangle contenant la carte, les coordonnées sont celles de l'angle gauche supérieur

```
<rect id="reg00" x="-3703" y="-500" width="10806" height="7427" style="fill:silver"/>
<path id="reg01" style="stroke:#1E1916;fill:#B7DDC8;stroke-width:3" d="….z"/>
```

Les polygones représentant les régions seront nommés « reg01 » ……

Colorier et afficher le nom de la région au passage de la souris

```
<g onmousemove="nommer(evt)">
<rect id="reg00" x="-3703" y="0" width="10806" height="7427" style="fill:silver"/>
<path id="reg01" style="stroke:#1E1916;fill:#B7DDC8;stroke-width:3" d="…" />
……
<text id="nom" x="-3703" y="0" style="text-anchor:middle;font-size:8pt;font-family:Times Roman;fill:black">?</text>
```

La fonction nommer(evt)

```
var num_choisi=0,cible0="reg00",old_style="";
var svgdoc=evt.getTarget().getOwnerDocument();

function nommer(evt)
{cible=evt.getTarget().getAttribute("id");
if (cible.substring(0,3)=="reg")
{numero=parseInt(cible.substring(3,5),10);
if (numero!=num_choisi)
{obj=svgdoc.getElementById("nom");
obj.setAttribute("x",donnees[numero][1]);obj.setAttribute("y",donnees[numero][2]);
child=obj.getFirstChild();child.setData(donnees[numero][0]);
if (cible0!="reg00") {obj=svgdoc.getElementById(cible0);
obj.setAttribute("style",old_style)};
if (cible!="reg00") {obj=svgdoc.getElementById(cible);
old_style=obj.getAttribute("style");obj.getStyle().setProperty("fill","red");
obj.getStyle().setProperty("fill-opacity","1.0");};
num_choisi=numero;cible0=cible}}}
```

Remplissage en fonction d'une valeur

Un exemple de remplissage en 3 nuances d'une couleur obtenues avec fill-opacity en fonction des valeurs d'une donnée relative à une autre

Déclarer le nombre de régions

```
var nb_regions=22;
```

Paramètres de la fonction :
col1 : donnée à traiter, col2 : valeur de référence, q1 : premier quartile, q2 : troisième quartile, couleur : couleur de remplissage

```
function remplir(evt,col1,col2,q1,q2,couleur)
{for(i=1;i<=nb_regions;i++)
{ if (i<=9) {region="reg0"+i} else {region="reg"+i};
obj=svgdoc.getElementById(region);
if ((donnees[i][col1]/donnees[i][col2])>=q2)
    { obj.getStyle().setProperty("fill",couleur);obj.getStyle().setProperty("fill-opacity","1.0")};
  if ((donnees[i][col1]/donnees[i][col2])<q1)
    { obj.getStyle().setProperty("fill",couleur);obj.getStyle().setProperty("fill-opacity","0.2")};
  if (((donnees[i][col1]/donnees[i][col2])>=q1)&&((donnees[i][col1]/donnees[i][col2])<q2))
    { obj.getStyle().setProperty("fill",couleur);obj.getStyle().setProperty("fill-opacity","0.5")}}}
```

Le remplissage peut être accompagné d'une légende qui reprend les significations des différentes couleurs

```
function legendage(evt,couleur)
{obj=svgdoc.getElementById("rectleg1");
     obj.getStyle().setProperty("fill",couleur);
     obj.getStyle().setProperty("fill-opacity","0.1");
obj=svgdoc.getElementById("rectleg2");
     obj.getStyle().setProperty("fill",couleur);
     obj.getStyle().setProperty("fill-opacity","0.5");
obj=svgdoc.getElementById("rectleg3");
     obj.getStyle().setProperty("fill",couleur);
     obj.getStyle().setProperty("fill-opacity","1.0")}
```

Et d'un titre explicatif

```
obj=svgdoc.getElementById("texlegdensite");
obj.getStyle().setProperty("visibility","visible");
```

Les rectangles recevant les couleurs sont déclarés :

```
<rect id="rectleg1" x="500" y="75" width="60" height="15" style="stroke:black;fill:none;fill-opacity:1"/>
<rect id="rectleg2" x="560" y="75" width="60" height="15" style="stroke:black;fill:none;fill-opacity:1"/>
<rect id="rectleg3" x="620" y="75" width="60" height="15" style="stroke:black;fill:none;fill-opacity:1"/>
```

ainsi que le texte explicatif

```
<text id="texlegdensite" x="530" y="70" style="visibility:hidden;text-anchor:middle;font-size:8pt;font-family:Times
Roman;fill:black">+100h/km<tspan dy="-4" style="font-size:6pt">2</tspan><tspan x="590" dy="4" style="font-
size:8pt">20-99h/km<tspan dy="-4" style="font-size:6pt">2</tspan></tspan><tspan x="650" dy="4" style="font-
size:8pt">-20h/km<tspan dy="-4" style="font-size:6pt">2</tspan></tspan></text>
```

(Utilisation de <tspan> et de text-anchor :middle pour que les différents éléments soient correctement positionnés et utilisation de dy pour mettre 2 en exposant.

La fonction qui déclenche le remplissage devient

```
var legende= "" ;
function densite(evt)
{if (legende!= "densite")
{legende="densite" ;legendage(evt,"#cc9933");
obj=svgdoc.getElementById("texlegdensite");
obj.getStyle().setProperty("visibility","visible");
remplir(evt,3,4,20,100,"#cc9933")}}
```

En supposant que donnees[i][3] est la population de la région i et donnees[i][4] sa superficie.

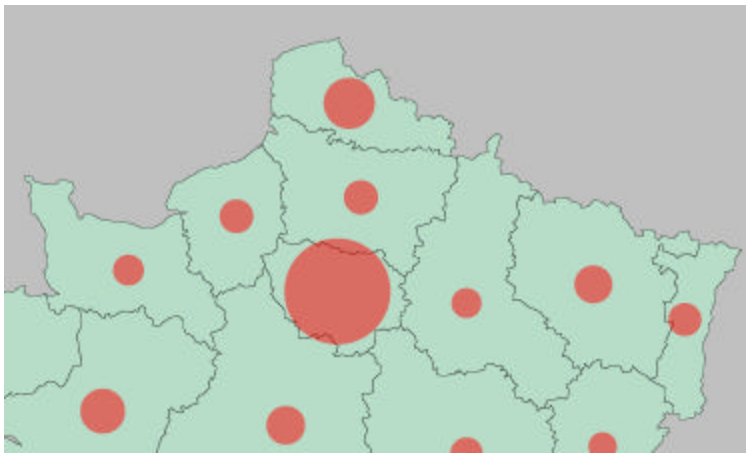
Affecter à une région un cercle dont la taille sera fonction d'une valeur

Ici nous allons cloner un cercle pour chaque région et lui donner un rayon plus ou moins grand selon la valeur de la population

Un objet est déclaré avec un rayon nul pour qu'il ne soit pas visible

```
<g id="cercles">
<circle id="cercle" cx="-3703" cy="0" r="0" style="fill:red;fill-opacity:0.5" />
</g>
```
La fonction
```
function population(evt)
{if (legende!="population")
{var target=svgdoc.getElementById("cercle");
var  contents = svgdoc.getElementById ('cercles');
mini=donnees[1][3];maxi=donnees[1][3];
for (i=2;i<=nb_regions;i++)
{if (donnees[i][3]<mini) {mini=donnees[i][3]};
if (donnees[i][3]>maxi) {maxi=donnees[i][3]}};
for (i=1;i<=nb_regions;i++)
{var newnode = target.cloneNode(false);
etiq="cercle"+i.toString();newnode.setAttribute ('id', etiq);
newnode.setAttribute("cx",donnees[i][1].toString());
newnode.setAttribute("cy",donnees[i][2].toString());
rayon=75+Math.round(300*(donnees[i][3]-mini)/(maxi-mini));
newnode.setAttribute("r",rayon.toString());
newnode = contents.appendChild (newnode)};legende="population"}}
```

Nous recherchons le minimum et le maximum des valeurs, dans cet exemple le rayon des cercles ira de 75 à 375
Nous affectons un identificateur aux cercles créés pour pouvoir les effacer :



```
function efface_population(evt)
{ for (i=1;i<=nb_regions;i++)
{cible="cercle"+i;obj=svgdoc.getElementById(cibl
e);
var  contents = svgdoc.getElementById
('cercles');
contents.removeChild (obj)}};
```

Affecter à une région un symbole

Créer le symbole :

```
<g id="symboles">
<g id="maison" transform="matrix(1 0 0 1 0 0)" style="stroke:black;visibility:hidden;fill-opacity:0.7">
<path d="M-50 -50l100 0 -50 -30 z" style="fill:red" />
<path d="M-45 -50l90 0 0 100 -90 0 z" style="fill:yellow" />
<path d="M-15 50l30 0 0 -40 -30 0 z M-30 -10l20 0 0 -20 -20 0 z M30 -10l-20 0 0 -20 20 0 z" style="fill:white" />
</g>
</g>
```

Attention : définir un groupe de symboles qui gérera la transformation, la visibilité, l'opacité et l'identification.
Le symbole sera cloné puis placé et dimensionné avec une transformation définie par une matrice.
Il faut impérativement qu'il soit centré en 0 ; 0, car le centre de l'homothétie sera obligatoirement 0 ; 0.
Au départ, ce symbole est caché.

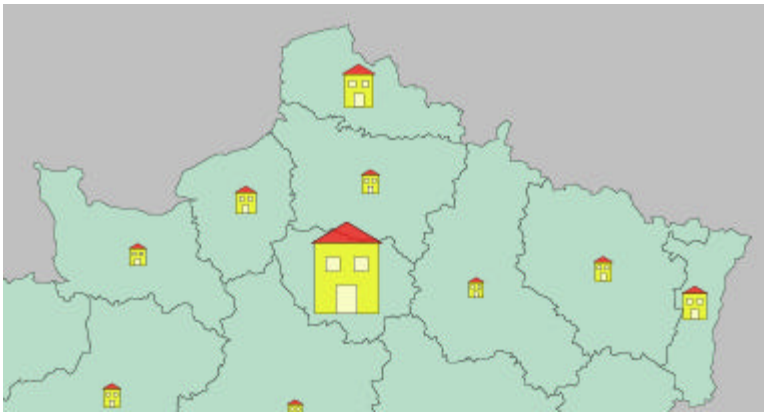
```
function densite(evt)
{if (legende!="densite")
{var target=svgdoc.getElementById("maison");
var  contents = svgdoc.getElementById ('symboles');
mini=donnees[1][5];maxi=donnees[1][5];
for (i=2;i<=nb_regions;i++)
{if (donnees[i][5]<mini) {mini=donnees[i][5]};
if (donnees[i][5]>maxi) {maxi=donnees[i][5]}};
for (i=1;i<=nb_regiions;i++)
{var newnode = target.cloneNode(false);
```

```
etiq="maison"+i.toString();
newnode.setAttribute ('id', etiq);
taille=1+Math.round(40*(donnees[i][5]-mini)/(maxi-mini))/10;
matrice="matrix("+taille+" 0 0 "+taille+" "+donnees[i][1].toString()+" "+donnees[i][2].toString()+")";
newnode.getStyle().setProperty("visibility","visible");
newnode.setAttribute("transform",matrice);
newnode = contents.appendChild (newnode)};legende="densite"}}
```

La taille k varie de 1 à 5, le centre du symbole créé sera au centre de la région.
La matrice de transformation est donc (k 0 0 k x_centre y_centre)
Pour effacer ces symboles, la procédure est la même que pour les cercles.

```
function efface_densite(evt)
{ for (i=1;i<=nb_regions;i++)
{cible="maison"+i;obj=svgdoc.getElementById(cible);
var  contents = svgdoc.getElementById ('symboles');
contents.removeChild (obj)}};
```

On peut définir une fonction qui efface une série de symboles quelconques en envoyant comme paramètres le nom du groupe contenant les symboles et le nom générique du symbole



```
function
efface(evt,groupe_symboles,nom_symbole)
{ for (i=1;i<=nb_regions;i++)
{cible=nom_symbole+i;obj=svgdoc.getElementBy
Id(cible);
var  contents = svgdoc.getElementById
(groupe_symboles);
contents.removeChild (obj)}};
```

Pour effacer la densité précédente
`efface(evt, 'symboles','maison'`

Affecter à une région un camembert

Nous devons définir chaque région du camembert pour lui donner une couleur de remplissage différente

```
<g id="pie" style="visibility:visible;stroke:gray;fill-opacity:0.5">
<path id="pie1" d="M0 0" style="fill:#cc9933" />
<path id="pie2" d="M0 0" style="fill:#ff9900" />
<path id="pie3" d="M0 0" style="fill:#ffff00" />
<path id="pie4" d="M0 0" style="fill:#ffffff" />
</g>
```

La construction des camemberts

```
function pie(evt)
{if (legende!="pie")
{legende="pie";
rayon=15;var  contents = svgdoc.getElementById ('pie');
for(i=1;i<=nb_regions;i++)
{a='';xA=rayon+donnees[i][1];yA=donnees[i][2];angle=0;
```

Départ du tracé à l'Est du camembert x_centre+rayon ;y_centre
Première portion fonction du rapport donnée n°5/donnée n°3, donnée n°3 représente la somme des données.
Nous utilisons la fonction arc avec comme point de départ xA ;yA comme rayon du cercle rayon en x et rayon en y
Point d'arrivée de l'arc : xB ;yB paramètres pour le sens de l'arc 0 0 0 ou 0 1 0, et enfin retour au centre du cercle avec L x_centre y_centre :

```
angle1=360*donnees[i][5]/donnees[i][3];angle=angle+angle1;
xB=donnees[i][1]+rayon*Math.cos(angle*Math.PI/180);
```

```
yB=donnees[i][2]-rayon*Math.sin(angle*Math.PI/180);
if (angle1<=180)
{a=a+'M'+xA+' '+yA+' A '+rayon.toString()+' '+rayon.toString()+' 0 0 0 '+xB+' '+yB+ 'L'+donnees[i][1].toString()+
","+donnees[i][2].toString()+' z'} else
{a=a+'M'+xA+' '+yA+' A '+rayon.toString()+' '+rayon.toString()+' 0 1 0 '+xB+' '+yB+' L'+donnees[i][1].toString()+","
+donnees[i][2].toString()+' z'};
var target1=svgdoc.getElementById("pie1");
var newnode = target1.cloneNode(false);
etiq="pie1"+i.toString();newnode.setAttribute ('id', etiq);newnode.setAttribute("d",a);
newnode = contents.appendChild (newnode);xA=xB;yA=yB;a="";
```

Deuxième portion

```
angle1=360*donnees[i][7]/donnees[i][3];angle=angle+angle1;
xB=donnees[i][1]+rayon*Math.cos(angle*Math.PI/180);
yB=donnees[i][2]-rayon*Math.sin(angle*Math.PI/180);
if (angle1<=180)
{a=a+'M '+xA+' '+yA+' A '+rayon.toString()+' '+rayon.toString()+' 0 0 0 '+xB+' '+yB+' L' +donnees[i][1].toString()+","
+donnees[i][2].toString()+' z'} else
{a=a+'M '+xA+' '+yA+ ' A '+rayon.toString()+' '+rayon.toString()+' 0 1 0 '+xB+' '+yB+' L'+donnees[i][1].toString()+","
+donnees[i][2].toString()+' z'};
var target1=svgdoc.getElementById("pie2");
var newnode = target1.cloneNode(false);
etiq="pie2"+i.toString();newnode.setAttribute ('id', etiq);newnode.setAttribute("d",a);
newnode = contents.appendChild (newnode);xA=xB;yA=yB;a="";
```

Troisième portion

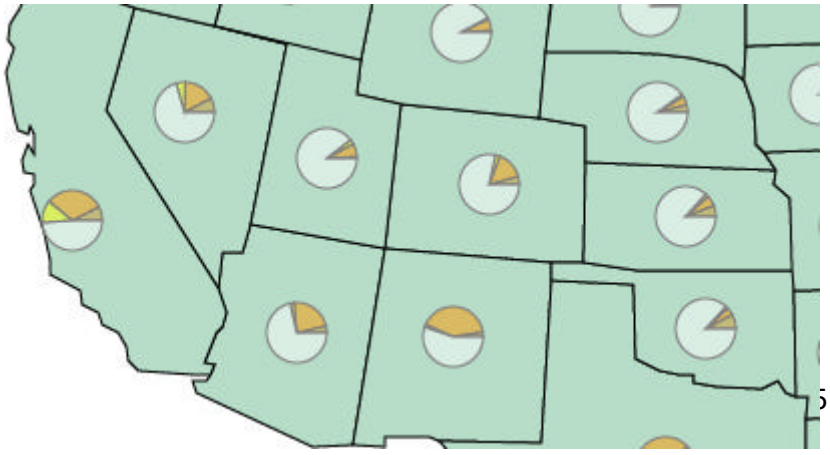
```
angle1=360*donnees[i][6]/donnees[i][3];angle=angle+angle1;
xB=donnees[i][1]+rayon*Math.cos(angle*Math.PI/180);
yB=donnees[i][2]-rayon*Math.sin(angle*Math.PI/180);
if (angle1<=180)
{a=a+'M '+xA+' '+yA+' A '+rayon.toString()+' '+rayon.toString()+' 0 0 0 '+xB+' '+yB+' L'+donnees[i][1].toString()+","
+donnees[i][2].toString()+' z'} else
{a=a+'M '+xA+' '+yA+' A '+rayon.toString()+' '+rayon.toString()+' 0 1 0 '+xB+' '+yB+' L'+donnees[i][1].toString()+","
+donnees[i][2].toString()+' z'};
var target1=svgdoc.getElementById("pie3");
var newnode = target1.cloneNode(false);
etiq="pie3"+i.toString();newnode.setAttribute ('id', etiq);newnode.setAttribute("d",a);
newnode = contents.appendChild (newnode);xA=xB;yA=yB;a="";
```

Dernière portion

```
angle1=360-angle;
xB=Math.round(donnees[i][1]+rayon);yB=Math.round(donnees[i][2]);
if (angle1<=180)
{a=a+'M '+xA+' '+yA+' A '+rayon.toString()+' '+rayon.toString()+' 0 0 0 '+xB+' '+yB+' L'+donnees[i][1].toString()+","
+donnees[i][2].toString()+' z'} else
{a=a+'M '+xA+' '+yA+' A '+rayon.toString()+' '+rayon.toString()+' 0 1 0 '+xB+' '+yB+' L'+donnees[i][1].toString()+","
+donnees[i][2].toString()+' z'};
var target1=svgdoc.getElementById("pie4");
var newnode = target1.cloneNode(false);
etiq="pie4"+i.toString();newnode.setAttribute ('id', etiq);newnode.setAttribute("d",a);
newnode = contents.appendChild
(newnode)}}}
```

Pour effacer ces camemberts

```
efface('pie', "pie1") ;
efface('pie', "pie2") ;
efface('pie', "pie3") ;
efface('pie', "pie4") ;
```



Sur cet exemple, la part des minorités ethniques (noirs, hispaniques et

asiatiques) dans la population des états des USA

Affecter à une région une barre orientée

Nous aurons un trait et une barre au-dessus de ce trait pour une valeur positive et au-dessous pour une valeur négative. La hauteur de la barre étant proportionnelle à la valeur absolue de la donnée.

Déclaration des objets à cloner

```
<g id="symboles">
<rect id="barre" x="-3703" y="-500" width="300" height="0" style="fill:red;fill-opacity:0.5" />
<path id="trait" d="M-3703 -500" style="stroke:black" />
</g>

function barre(evt)
{if (legende!="barre")
{large=300;
var target1=svgdoc.getElementById("barre");
var target2=svgdoc.getElementById("trait");
var  contents = svgdoc.getElementById ('symboles');
maxi=Math.abs((donnees[1][3]-donnees[1][4])/donnees[1][4]);
for (i=2;i<=nb_regions;i++)
{if (Math.abs((donnees[i][3]-donnees[i][4])/donnees[i][4])>maxi)
{maxi=Math.abs((donnees[i][3]-donnees[i][4])/donnees[i][4])}};
for (i=1;i<=nb_regions;i++)
{var newnode = target1.cloneNode(false);
etiq="barre"+i.toString();newnode.setAttribute ('id', etiq);
newnode.setAttribute("x",Math.round(donnees[i][1]-large/2).toString());
hauteur=Math.round(300*(donnees[i][3]-donnees[i][4])/(maxi*donnees[i][4]));
if (hauteur<0) {newnode.setAttribute("y",donnees[i][2].toString());
newnode.setAttribute("height",Math.abs(hauteur).toString());
newnode.getStyle().setProperty("fill","blue")}
else
{newnode.setAttribute("y",Math.round(donnees[i][2]-hauteur).toString());
newnode.setAttribute("height",Math.abs(hauteur).toString());
newnode.getStyle().setProperty("fill","red")};
newnode = contents.appendChild (newnode);
var newnode=target2.cloneNode(false);
etiq="trait"+i.toString();newnode.setAttribute ('id', etiq);
chaine="M"+Math.round(donnees[i][1]-large).toString()+" "+donnees[i][2].toString()
+" "+Math.round(donnees[i][1]+large).toString()+" "+donnees[i][2].toString();
newnode.setAttribute("d",chaine);
newnode = contents.appendChild (newnode)};
legende="barre"}}
```

Pour effacer ces barres

```
efface('symboles', "barre") ;
efface('symboles', "trait") ;
```

Voici par exemple l'évolution de la population des régions françaises entre les recensements de 1990 et 1999.

Une barre rouge si la population a augmenté, bleue si elle a diminué.
La hauteur de la barre est proportionnelle à la variation en pourcentage par rapport à la population de la région concernée en 1990.