Licence Informatique
parcours Science des Données

Visualisation de données

TP1 – Introduction à SVG et à d3.js

1 SVG

SVG est l'accronyme de Scalable Vector Graphics. Il s'agit d'un langage de description de graphiques vectoriel hérité de XML (eXtensible Markup Language) et reconnu par le World Wide Web Consortium.

1.1 Très brève introduction à XML

XML est un langage de balisage extensible.

XML est construit sur la base de balise délimitant des éléments. La balise l'ouverture d'un élément nommé « `element` » est `<element>`. La balise indiquant la fermeture du même élément est `</element>`. Entre la balise d'ouverture et la balise de fermeture se trouve le contenu de l'élément.

Une balise fermante doit exister pour toute balise ouvrante. La balise fermante succède à la balise ouvrante à laquelle elle correspond. Il est cependant possible d'indiquer un élément vide avec la syntaxe suivante `<element />` équivalente à `<element></element>`.

Un élément peut être inclus dans un autre, ce qui signifie que sa balise d'ouverture et sa balise de fermeture doivent intervenir entre les balises d'ouverture et de fermeture de l'élément qui l'inclut.

La syntaxe suivante est correcte :

```
<element1>
  <element2>
  </element2>
  <element3>
  </element3
</element1>
```

La syntaxe suivante n'est pas correcte :

```
<element1>
  <element2>
  <element3>
  </element2>
  </element3
</element1>
```

De l'imbrication des différents éléments reésulte le fait qu'un document XML est un arbre.

Il est possible de spécifier des attributs pour un élément. Ces attributs sont indiqués au sein de la balise ouvrante par des associations clé-valeur. Dans l'exemple suivant, on donne deux attributs à l'élément, l'attribut nommé `att1` a pour valeur `val1` et l'attribut nommé `att2` a pour valeur `val2`.

```
<element val1="att1" val2="att2">
</element>
```

Et la notation équivalente

```
<element val1="att1" val2="att2" />
```

Un document XML qui respecte les principes énoncés plus haut est dit bien formé.

« Extensible » signifie qu'on peut construire d'autres langage à partir de XML. La spécification d'un langage XML revient à exprimer les différents éléments qui peuvent être utilisés au sein de ce langage et la syntaxe de ces éléments, c'est-à-dire quelle doit être la composition des éléments, quels éléments sont inclus les uns dans les autres, lesquels sont obligatoires, lesquels sont optionnels, leurs nombres minimal et maximal d'occurrences, et pour chacun d'eux la liste et le type de leurs attributs ainsi que des indications sur le caracttères obligatoire ou optionnel de ces attributs et leur valeur par défaut.

L'ensemble des spécifications d'un langage XML est spécifié au sein d'une DTD (Document Type Definition) ou d'un schéma.

Lorsqu'un document XML est conforme aux spécifications définies dans son schéma ou sa DTD, on dit qu'il est valide.

S'il est possible de définir son propre type de documents, un certain nombre de schémas sont déjà largement adoptés par des communautés. Ainsi, même si le langage HTML est historiquement plus ancien que XML, ses versions les plus récentes sont définies comme des déclinaisons de XML. On peut également évoquer la liste non exhaustive de schémas dont certains sont normalisés par le W3C (World Wide Web Consortium) :

XSL : eXtensible Stylesheet Language

XSD : eXtensible Schema Description

CML : Chemical Markup Language

MathML : Mathmatical Markup Language

DocBook : Schéma utilisé pour la documentation libre

Il est a noter également que les formats ouverts de description de documents tels que ODT (Open Document Text) ou ODS (Open Document Spreadsheet) utilisées notamment par LibreOffice et OpenOffice reposent sur XML.

SVG (normalisé par le W3C) est un langage de description de graphiques vectoriels décliné de XML.

1.2 Les éléments SVG

Nous donnons ici un aperçu rapide des différents éléments SVG permettant de définir des primitives graphiques. Le document de référence décrivant SVG est accessible via `https://www.w3.org/TR/SVG11/`, mais ses fonctionalités sont également décrites en profondeur à partir de `https://www.w3schools.com/graphics/svg_intro.asp`.

`svg :` Il s'agit de l'élément racine permettant de définir un graphique. Un tel élément peut être inclus au sein d'une page HTML. Les deux arguments `width` et `height` permettent de définir la largeur et la hauteur du graphique.

`line :` permet de définir un segment de droite avec les attributs `x1`, `y1`, `x2` et `y2` qui définissent les abscisses et ordonnées des deux extrémités.

`rect :` permet de définir un rectangle grâce aux attributs `x`, `y`, `width` et `height` qui définissent position et dimension du rectangle.

`circle :` permet de définir un cercle grâce aux attributs `cx` et `cy` qui définissent la position du centre du cercle, et de l'attribut `r` qui définit son rayon.

`ellips :` permet de définir une ellipse grâce aux attributs `cx` et `cy` qui définissent la position du centre de l'ellipse, et des attributs `rx` et `ry` qui définissent la taille des rayons horizontaux et verticaux.

`polyline :` permet la définition d'une ligne brisée grâce à l'atttribut `points` permettant d'indiquer la liste des couples abscisse-ordonnée des différents sommets

`polygon :` permet la définition d'un polygone grâce à l'atttribut `points` permettant d'indiquer la liste des couples abscisse-ordonnée des sommets

`path :` permet la définition d'un tracé. L'attribut `d` permet la définition du tracé grâce à une grammaire composée de commandes :

M	position de départ du tracé
L	un segment
H	un segment horizontal
V	un segment vertical
C	une courbe
S	une courbe lisse
Q	une courbe de Bezier quadratique
T	une courbe de Bezier quadratique lisse
A	un arc d'ellipse
Z	indique la fermeture du tracé

`text :` définit un élément textuel dont la position est indiquée par les attributs `x` et `y`. Le texte de l'élément doit être indiqué entre les balises `<text>` et `</text>`.

À l'ensemble de ces éléments, il est possible de spécifier des attributs de mise en forme. Cette mise en forme peut être exprimée de différente façons :

— directement via des attributs optionnels des éléments. Par exemple, l'attribul `stroke` permet d'indiquer la couleur de la bordure, `stroke-width` permet de préciser la largeur de la bordure, `fill` permet de préciser la couleur ou le motif du remplisaque pour les formes fermées.

— en indiquant dans l'attribut `style` l'ensemble des directives de mise en forme. Par exemple `fill="stroke:blue; stroke-width:3pt; fill: red;"` précise que la bordure est bleue et de largeur 3 pt et que la couleur de remplissage est rouge.

— en spécifiant les attributs de mise en forme via des directives CSS (voir plus loin).

Il est possible d'appliquer des transformations géométriques aux différents éléments grâce à l'attribut `transform`. Cet attribut prend pour valeur une série de commandes telles que `translate()`, `rotate()`, `skewX()`, `skewY()`, `scale()` ou encore `matrix()`.

Enfin, il est possible de spécifier d'animer les éléments les éléments SVG en spécifiant une modification de leurs attributs géométriques ou de mise en forme.

2 CSS

Nous présentons dans les paragraphes suivants un très bref aperçu de CSS. CSS (Cascading Style Sheets) est un langage de description des mises en forme adopté par le W3C. Il est documenté sur de nombreux sites dont en particulier celui du W3C `https://www.w3.org/Style/CSS/`. Le mécanisme des feulles de style CSS permet de séparer la couche logique d'un document de sa couche de présentation et aide ainsi à la définition de charte graphique cohérente puisque tous les éléments de même type se verront appliquer la même mise en forme. La modification de cette charte graphique opérée sur la feuille de style s'appliquera de façon identique à tous les documents et élements concerné par la modification.

Une feuille de style CSS est une suite de directives de mise en forme. Une directive est composée d'un sélecteur suivi de la définition de la mise en forme. Le sélecteur permet d'identifier les éléments auxquels la mise en forme dont la définition suit va s'appliquer. La définition de la mise en forme est une suite de couple attribut-valeur indiquant la valeur à attribuer à l'attribut.

Sélecteur	Signification
*	tout élement
`E`	tous les éléments `<E>`
`E F`	tous les éléments `<F>` fils d'un élément `<E>`
`E > F`	tous les éléments `<F>` descendant d'un élément `<E>`
`E + F`	tous les éléments `<F>` précédés d'un élément `<E>`
`E[att]`	tous les éléments `<E>` disposant d'un attribut `att`
`E[att="val"]`	tous les éléments `<E>` disposant d'un attribut `att` dont la valeur est `val`
`E.val`	tous les éléments `<E>` dont l'attribut `class` a pour valeur `val`
`E#val`	tous les éléments `<E>` dont l'attribut `id` a pour valeur `val`

```
p {
  color: red;
  text-align: justify;
}

h1, h2, h3 {
  font-family: Helvetica, Geneva, Arial, sans-serif;
  font-weight: bold;
}

.emph {
  font-style: italic;
```

```
}
```

L'exemple précédent indique que tous les éléments `<p>` seront écrits en rouge et avec un alignement à gauche, que tous les éléments `<h1>`, `<h2>` et `<h3>` seront écrits dans les polices de caractères indiquées avec l'ordre de priorité indiqué (selon disponibilité sur le système) en gras, et qu'enfin tous les éléments dont l'attribut `class` vaut `emph` seront écrit en italic.

CSS permet le positionnement des éléments grâce un modèle de boîtes. Chaque boîte dispose d'une marge (espacement externe), d'une bordure, d'une marge de remplissage (décalage entre l'intérieur de la bordure et le contenu de la boîte). L'épaisseur de ces différentes dimensions peuvent être distinguées (en haut, en bas, à gauche et à droite). Il est notamment possible d'indiquer les couleurs, épaisseurs, style, pointillés.

Les attributs de mise en forme peuvent concerner le texte, par exemple polices de caractères, alignement, graisse, décoration (barré, italique...), hauteur de ligne, hauteur d'interligne, espacement entre les caractères...

Il n'est pas possible d'énoncer ici l'ensemble des fonctionnalités de CSS. Un aperçu relativement complet est accessible sur `https://www.w3schools.com/css/default.asp`. On peut notamment citer les différentes façons de définir les couleurs notaments via les systèmes RGB ou HSL associé à l'opacité (ou transparence). Il est également possible de définir des gradients de couleurs ou des filtres de transformation.

Il est également possible de spécifier des animations ou des transformations géométriques (rotation, translation, mise à l'échelle).

Les feuilles de style peuvent s'appliquer aux éléments d'une page HTML, dont en particulier les éléments SVG intégrés au sein des page HTML.

La spécification des directives CSS peut être spécifié entre les balises `<style>` et `</style>` au sein de l'élément `head` d'une page HTML. Elle peut également spécifiée dans un document externe auquel il est fait référence via l'élément `link` inclus dans la balise head de la façon suivante.

```
<link rel="stylesheet" type="text/css" href="style.css" />
```

Un directive de mise en forme peut également être indiquée dans l'attribut style d'un élément.

3 d3.js

d3.js (pour Data Driven Documents) est une bibliothèque de traitement pour la visualisation de données écrite en langage JavaScript documentée sur `http://d3js.org` qui permet via d'ajouter des éléments à un document HTML via DOM (Document Object Model). Les éléments ajoutés sont des éléments SVG dont les attributs de mise en forme peuvent être fonction des données à représenter.

On lie une page HTML à la bibliothèque d3.js en intégrant une balise `<script/>` dont l'attribut `src` désigne le fichier `d3.v4.mis.js` disponible en ligne à l'adresse `http://d3js.org/d3.v4.min.js`. Notez que le fichier peut également être téléchargé pour être référencé localement. Il est ensuite possible de faire appel aux fonctions de la bibliothèque d3.js dans du code JavaScript intégré dans une page HTML.

JavaScript est un langage objet qui permet notament d'interagir avec les éléments de la page HTML au sein de laquelle le code est insérer via l'arbre DOM, arbre des éléments XML. d3.js exploite ces caractéristiques via des méthodes qui renvoient le plus souvent l'objet appelant, modifié par la méthode, lui-même. Il est ainsi possible d'enchainer les appels aux méthodes.

L'exemple suivant indique une page HTML vide.

```
<!doctype html>
<html>
  <head>
    <title>Page de test</title>
    <script src="http://d3js.org/d3.v4.min.js" />
  </head>
  <body>
  </body>
</html>
```

Le code suivant va créer dynamiquement un élément `svg` au sein de l'élément `body`. À cet élément `svg` dont les dimensions sont indiquées, sera également ajouté un rectangle lui aussi affecté d'attributs. Le résultat peut-être visualisé via un navigateur et détaillé en utilisant la fonctionnalité « inspecteur » de celui-ci.

```
<!doctype html>
<html>
  <head>
    <title>Page de test</title>
    <script src="http://d3js.org/d3.v4.min.js"></script>
  </head>

  <body>
    <script>
      var canvas = d3.select("body")
      .append("svg")
      .attr("width",500)
      .attr("height",500)

      var rect = canvas.append("rect")
      .attr("x",200)
      .attr("y",200)
      .attr("width",200)
      .attr("height",100)
      .classed("rounded",true)
      .style("fill","linen")
      .style("stroke","black")
      .style("stroke-width","2pt")
      .style("stroke-dasharray","10,5,30,5")
    </script>
  </body>
</html>
```

D'autres formes plus complexes peuvent être intégrées au moyen des fonctions décrites dans `https://github.com/d3/d3/wiki`. Il est notamment possible de générer des formes dont les attributs de mise en forme sont liés à des données à représenter, comme dans l'exemple suivant.

```
<!doctype html>
<html>
  <head>
    <title>Page de test</title>
    <script src="http://d3js.org/d3.v4.min.js"></script>
  </head>

  <body>
    <script>

      var data = [
        {"x": 100, "y": 10, "number": 200, "class":"C1", "intensity":0.4},
        {"x": 130, "y": 120, "number": 150, "class":"C1", "intensity":0.6},
        {"x": 200, "y": 150, "number": 250, "class":"C2", "intensity":0.7},
        {"x": 400, "y": 200, "number": 100, "class":"C2", "intensity":0.3},
        {"x": 300, "y": 400, "number": 120, "class":"C3", "intensity":0.7},
        {"x": 200, "y": 350, "number": 80, "class":"C3", "intensity":0.4},
        {"x": 350, "y": 100, "number": 200, "class":"C3", "intensity":0.8},
      ]

      var canvas = d3.select("body")
      .append("svg")
      .attr("width",500)
      .attr("height",500)

      canvas.selectAll("circle")
      .data(data)
      .enter()
      .append("circle")
      .attr("cx", function(d) {return d.x})
      .attr("cy", function(d) {return d.y})
      .attr("r", 50)
      .attr("stroke", "black")
      .attr("fill", "cyan")

      canvas.selectAll("text")
      .data(data)
      .enter()
      .append("text")
      .attr("x", function(d) {return d.x})
      .attr("y", function(d) {return d.y})
      .text(function (d,i) {return i})
    </script>
  </body>
</html>
```

4 Manipulations

1. Créez une page contenant un élément SVG déclaré de façon statique (pas de code JavaScript) décrivant un histogramme avec une barre pour chacune des données de l'exemple précédent. La surface de la barre doit être proportionnelle au champ `number`.
2. Créez ce même histogramme de façon dynamique avec la bibliothèque d3.js. Chaque rectangle décrivant une barre aura un champ `class` dont la valeur sera `C1`, `C2` ou `C3` selon les cas. Des couleurs différentes seront attribuées à chacune des classes.
3. Modifiez éventuellement le code précédent de sorte que l'association entre la couleur et la classe soient indiquée dans un élément `style` au moyen de directives CSS.
4. On souhaite que chacune des données soit dorénavant décrite par une forme dont la position est définie par les coordonnées `x` et `y`. La surface de la forme est proportionnelle à l'attribut `number`, la couleur de remplissage correspond à l'association précédente, l'opacité de ce remplissage est défini par l'attribut `intensity` de la donnée. Enfin, la forme doit être un cercle si le type de la donnée est `A` et un carré si le type de la donnée est `B`.
5. Intégrez des fonctionnalités d'animation et de transformation lors du survol de l'élément avec la souris.