

Programmation Web pour la Visualisation

SD BUT3 R5.VCOD.07
Yacine Ouzrout & Boubou Thiam Niang

ORGANISATION DU COURS

7 TD et 10 TP

- **Partie 1 :** Approfondissement Javascript
 - Programmation Objet en JS
 - Structuration des pages Web (DOM)
- **Partie 2 :** Traitement des données Json
 - Les formats de transfert données XML, Json...
 - Automatisation du traitement des données avec Json
 - Exploitation des données en JS
- **Partie 3 : Visualisation avec D3.js**
 - Les graphiques en html : Canvas & SVG
 - La librairie D3.js pour la visualisation
 - Exemples de graphiques avec D3.js

PARTIE 3 : VISUALISATION AVEC D3.JS

3.1 Les graphiques en HTML

1. Les graphiques en HTML

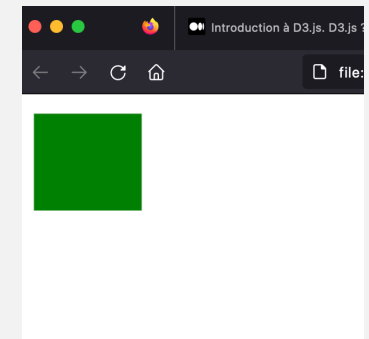
Pour faire des graphiques en html on utilise deux éléments particuliers : **<canvas>** ou **<svg>**

- **CANVAS** : on utilise l'élément html **<canvas>** avec l'API [canvas](#) ou l'API [WebGL](#) (ensemble de propriétés et de méthodes) pour dessiner des graphiques et des animations.

Exemple :

```
//HTML
<canvas id = "IdCanvas">
  Balise canvas définie dans la page web
</canvas>
```

```
//JavaScript
<script>
  var varCan =
  document.getElementById("IdCanvas");
  var vCtx = varCan.getContext("2d");
  vCtx.fillStyle = "green";
  vCtx.fillRect(10, 10, 100, 100);
</script>
```



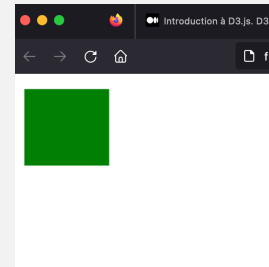
Canvas Tutorial → https://developer.mozilla.org/fr/docs/Web/API/Canvas_API/Tutorial

1. Les graphiques en HTML

- **SVG** (Scalable Vector Graphics) : SVG est un langage qui permet de décrire des images vectorielles (i.e. qui décrivent des lignes et des formes).
- Une image vectorielle n'est pas pixelisée, comme avec des images dites matricielles (tableaux de carrés colorés), elle est donc plus nette quel que soit la résolution de l'écran.
- Les formes de base de SVG sont : *circle*, *rect*, *ellipse*, *line*, *polyline*, *polygon* et *path*

Ex. 1

```
//Dessiner un rectangle vert sur la page
<svg width="300" height="200">
  <rect width="100%" height="100%" fill="green" />
</svg>
```



Ex. 2

```
<svg width="300" height="200" >
  <rect width="100%" height="100%" fill="red" />
  <circle cx="150" cy="100" r="80" fill="green" />
  <text x="150" y="125" font-size="60" text-anchor="middle" fill="white">SVG</text>
</svg>
```



SVG Tutorial →

<https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial>

https://edutechwiki.unige.ch/fr/Tutoriel_SVG_avec_HTML5

SVG Formes de base →

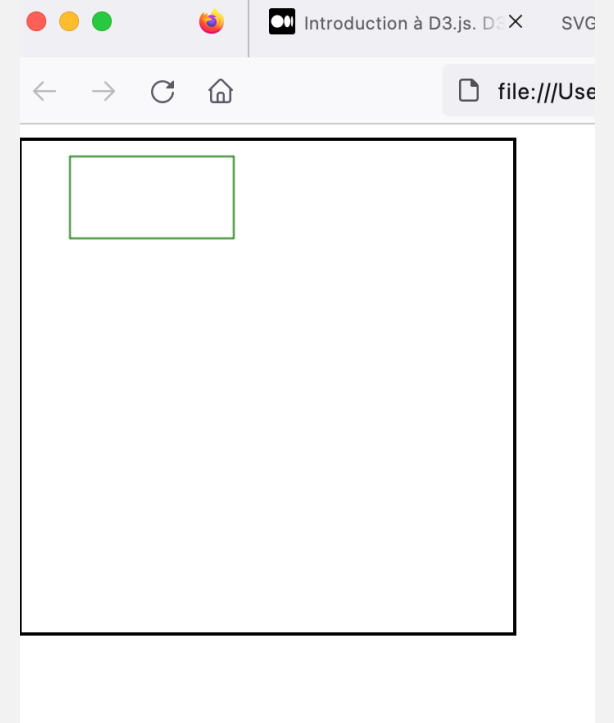
https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial/Basic_Shapes

Exemples de graphiques avec CANVAS

Exemple 1 : canvas

```
<body>
<canvas id="canv" width="300" height="300" style="border:2px solid black;"> </canvas>
<script>
  var canvas = document.getElementById("canv");
  var c = canvas.getContext("2d");

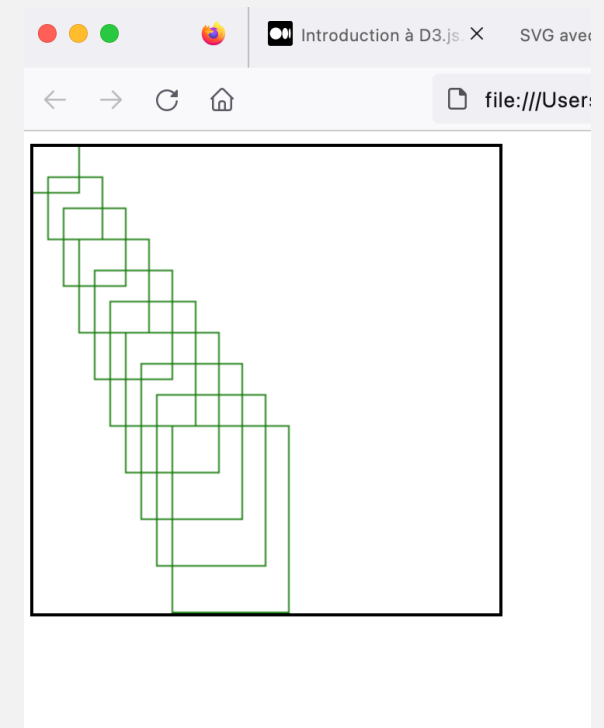
  var draw = function() {
    c.clearRect(0, 0, canvas.width, canvas.height);
    c.strokeStyle = "green";
    c.strokeRect(30, 10, 100, 50);
  }
  draw(); //appel de la fonction
</script>
</body>
```



Exemple 2 : canvas

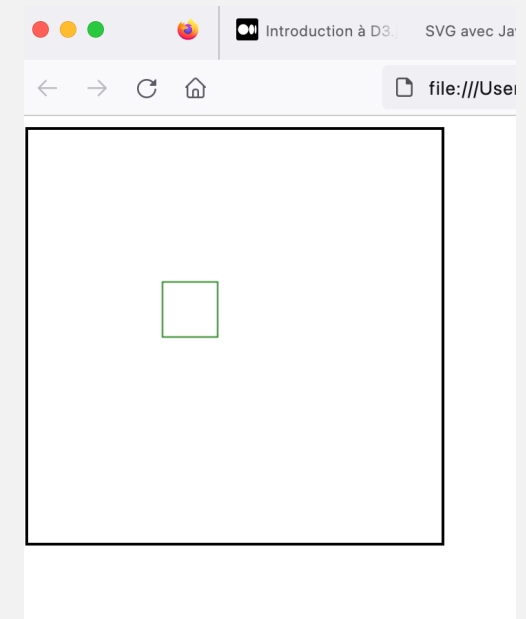
```
<body>
<canvas id="canv" width="300" height="300" style="border:2px solid black;"> </canvas>
<script>
  var canvas = document.getElementById("canv");
  var c = canvas.getContext("2d");

  var draw = function() {
    c.clearRect(0, 0, canvas.width, canvas.height);
    c.strokeStyle = "green";
    for ( x = 0; x < 10; x=x+1 ) {
      c.strokeRect(10*x, 20*x, 30+5*x, 30+10*x);
    }
  }
  draw(); //appel de la fonction
</script>
</body>
```



Exemple 3 : canvas

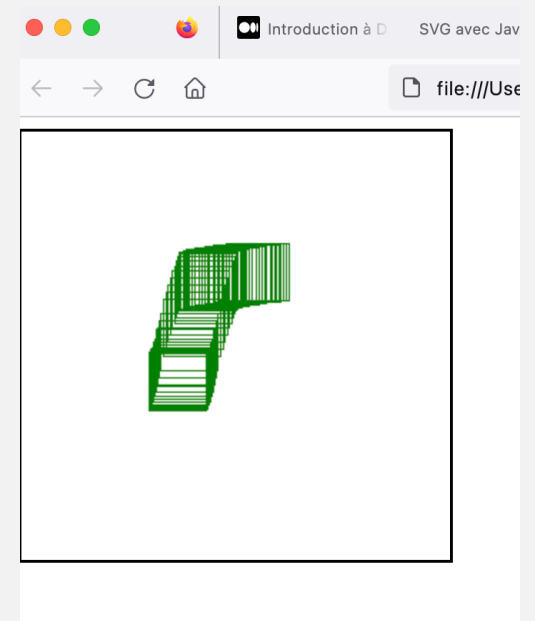
```
<body>
<canvas id="canv" width="300" height="300" style="border:2px solid black;"> </canvas>
<script>
  var canvas = document.getElementById("canv");
  var c = canvas.getContext("2d");
  var souris_x = 0;
  var souris_y = 0;
  var redessiner = function() {
    c.clearRect( 0, 0, canvas.width, canvas.height );
    c.strokeStyle = "green";
    c.strokeRect( souris_x-20, souris_y-20, 40, 40 );
  }
  var mouvementDeSouris = function(e) {
    var rectangle = canvas.getBoundingClientRect();
    souris_x = e.clientX - rectangle.left;
    souris_y = e.clientY - rectangle.top;
    redessiner();
  }
  canvas.addEventListener( 'mousemove',mouvementDeSouris);
</script>
</body>
```



Exemple 4 : canvas

```
<body>
<canvas id="canv" width="300" height="300" style="border:2px solid black;"> </canvas>
<script>
  var canvas = document.getElementById("canv");
  var c = canvas.getContext("2d");
  var t = []; // tableau de coordonnées de souris

  var redessiner = function() {
    c.clearRect( 0, 0, canvas.width, canvas.height );
    c.strokeStyle = "green";
    for ( i = 0; i < t.length; i=i+1 ) {
      var x = t[i][0];
      var y = t[i][1];
      c.strokeRect( x-20, y-20, 40, 40 );
    }
  }
  var mouvementDeSouris = function(e) {
    var rectangle = canvas.getBoundingClientRect();
    var x = e.clientX - rectangle.left;
    var y = e.clientY - rectangle.top;
    t.push( [ x, y ] ); // ajouter un élément à la fin du tableau
    if ( t.length > 50 ) {
      t.shift(); // enlever le premier élément du tableau
    }
    redessiner();
  }
  canvas.addEventListener('mousemove',mouvementDeSouris);
</script>
</body>
```



Exemples de graphiques avec SVG

Exemple 1 : SVG

<body>

```
<svg width="6cm" height="5cm" viewBox="0 0 600 500">
```

```
<!-- Fonction qui change le rayon à chaque click -->
```

```
<script>
```

```
function circle_click(evt) {
```

```
    var circle = evt.target;
```

```
    var currentRadius = circle.getAttribute("r");
```

```
    if (currentRadius == 100)
```

```
        circle.setAttribute("r", currentRadius*2);
```

```
    else
```

```
        circle.setAttribute("r", currentRadius*0.5);
```

```
}
```

```
</script>
```

```
<!-- Contour en bleu de l'aire du dessin -->
```

```
<rect x="1" y="1" width="598" height="498" fill="none" stroke="blue"/>
```

```
<!-- Appel à la fonction qui s'exécute à chaque click -->
```

```
<circle onclick="circle_click(evt)" cx="300" cy="225" r="100" fill="red"/>
```

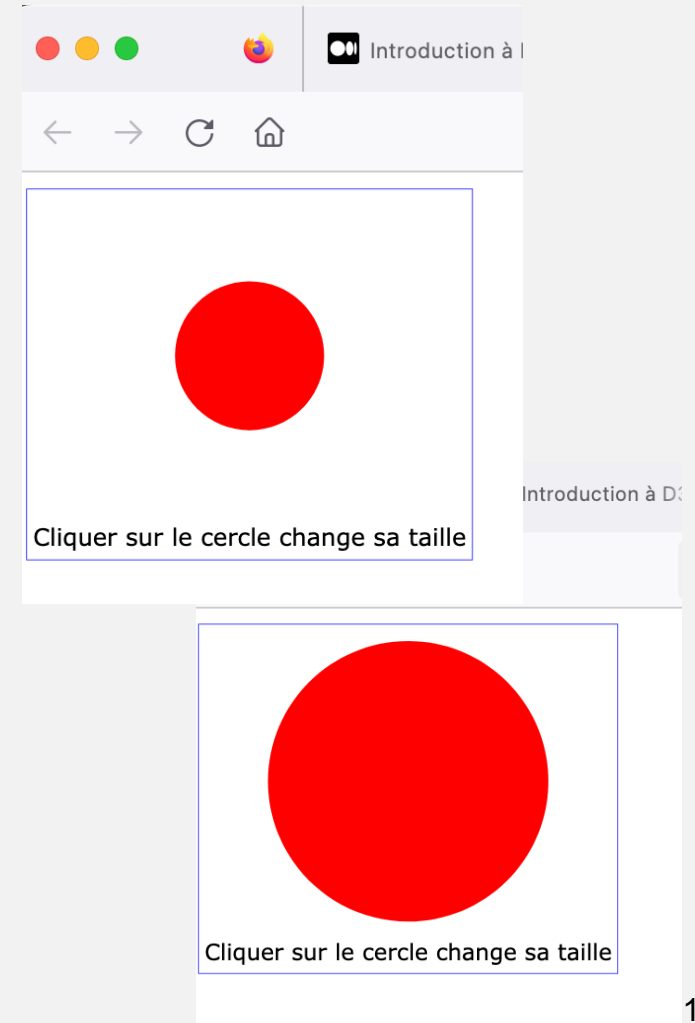
```
<text x="300" y="480" font-family="Verdana" font-size="32" text-anchor="middle">
```

```
    Cliquer sur le cercle change sa taille
```

```
</text>
```

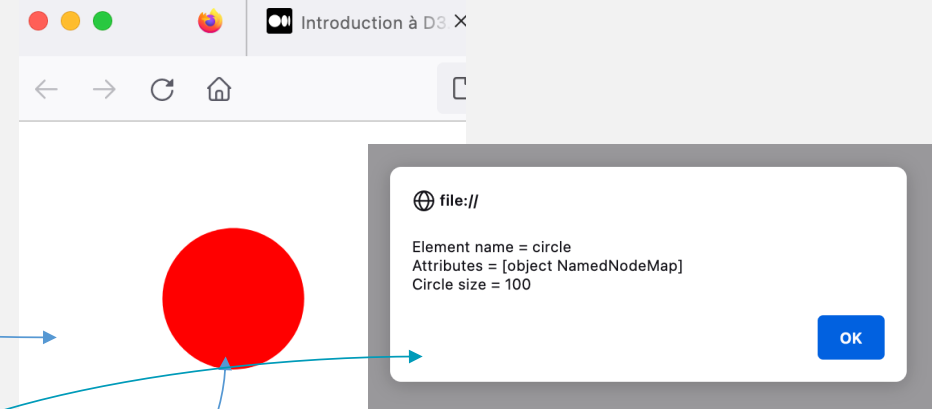
```
</svg>
```

</body>



Exemple 2 : SVG

```
<body>
<svg width="7cm" height="6cm" viewBox="0 0 600 500">
  <script>
    function infos(evt) {
      var element = evt.target;
      el_name = element.nodeName;
      el_attributes = element.attributes;
      circle_size = element.getAttribute("r")
      display_text = "Element name = " + el_name;
      display_text += " \nAttributes = " + el_attributes;
      display_text += " \nCircle size = " + circle_size;
      alert(display_text);
      element.setAttribute("fill", "yellow");
    }
  </script>
  <!-- S'exécute a chaque clic -->
  <circle onclick="infos(evt)" cx="300" cy="225" r="100" fill="red"/>
  <text x="300" y="480" font-family="Verdana" font-size="32" text-anchor="middle">
    Cliquer sur le cercle et lire le popup ...
  </text>
</svg>
</body>
```



Cliquer sur le cercle et lire le popup ..



Cliquer sur le cercle et lire le popup ..

Rappel : Fonctions en javascript

Les fonctions fléchées

- Les fonctions fléchées sont une nouvelle syntaxe pour écrire des fonctions JavaScript.
- Elles font gagner du temps aux développeurs et simplifient la portée des fonctions

Fonctions classiques :

- **function** somme(n1,n2) {
 document.write(n1+n2);
}

- maFonction = **function**(a, b) { return a+b; }

Fonctions fléchées :

- somme = (n1 , n2) =>
 document.write(n1 + n2);

- maFonction = (a, b) => { return a+b; }
- OU**
- maFonction = (a, b) => a+b;

PARTIE 3 : VISUALISATION AVEC D3.JS

3.2 Le Framework (librairie) D3.js

LIBRAIRIES DE CRÉATION DE PAGES WEB

Des librairies, *frameworks*, **CSS** gratuits ou payants :

- Bootstrap
- Foundation
- Materialize
- Bilma
- Ukit
- Skeleton
- ...

<https://www.codeur.com/blog/frameworks-css/>

Des librairies, *frameworks*, **Javascript** gratuits ou payants :

- Angular.js (*création de pages*)
- React.js (*faire des interfaces*)
- Vue.js (*faire des interfaces*)
- Jquery (*manipuler les données*)
- **D3.js** (*visualisation des données*)
- Chart.js (*faire des graphiques*)
- Anime.js (*faire des animations*)
- Bideo.js (*intégrer des vidéos*)
- ...

<https://kinsta.com/fr/blog/bibliotheques-javascript/>

<https://alokai-com.translate.goog/blog/vue-vs-react? x tr sl=en& x tr tl=fr& x tr hl=fr& x tr pto=rq>

Des **CMS** (Content Management System) :

- WordPress (*open source*)
- Joomla (*open source*)
- Drupal (*open source*)
- Jimdo
- Site123
- ...

<https://www.iphon.fr/hebergeur/fag/cms>

La librairie D3.JS

D3 : Data Driven Document

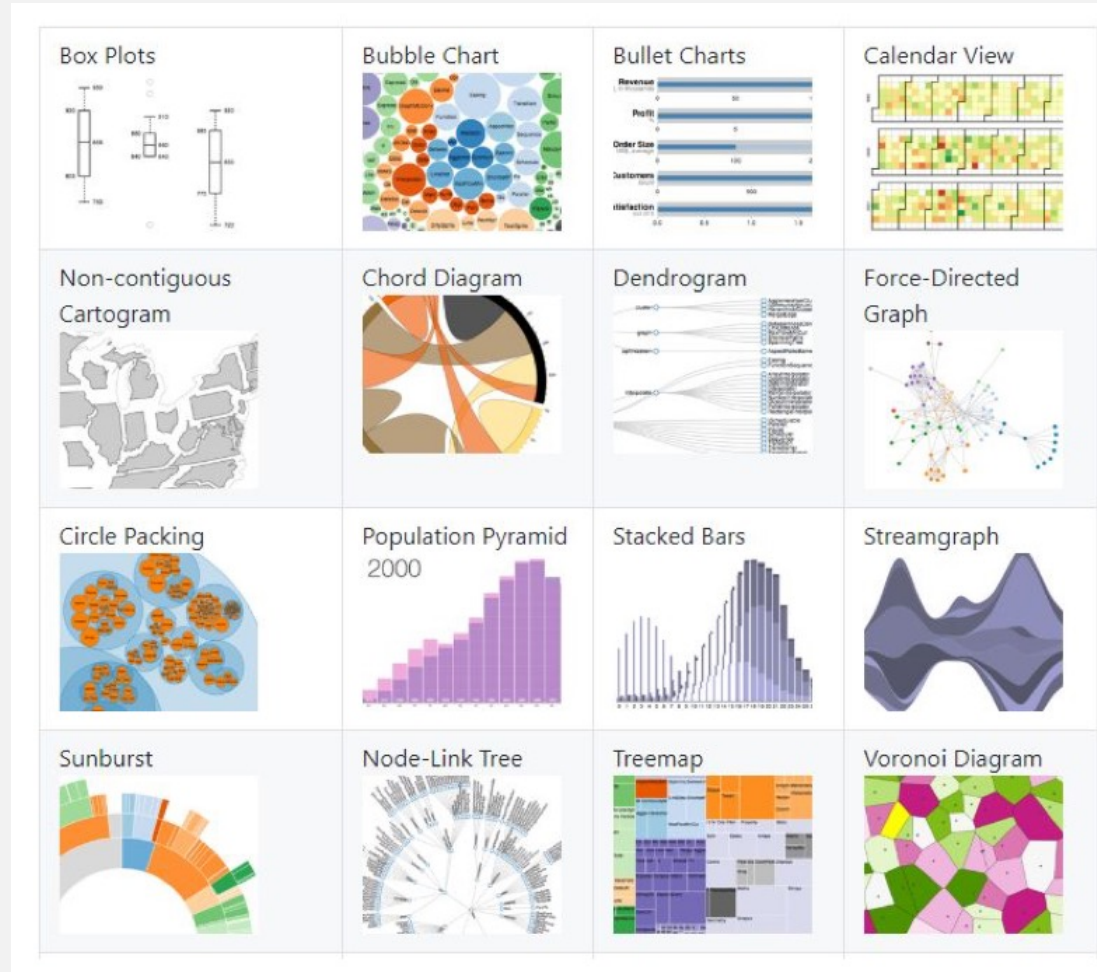
- Framework (librairie) JavaScript permettant de manipuler des documents (DOM) basés sur des données.
- D3 permet d'afficher des données numériques sous une forme graphique et dynamique en s'appuyant principalement sur HTML, SVG et CSS
- D3 permet de dessiner des graphiques, visualiser des données et développer des tableaux de bord.
- Il existe d'autres Framework du même type : *Chart.js*, *Plotly*, *paper* (<http://paperjs.org/>), *snapsvg* (<http://snapsvg.io/>),...

La Librairie D3.js

- **D3.js permet de :**
 - Créer sa propre dataviz à partir de mockups (maquettes)
 - Adapter de nombreux exemples D3.js disponibles (<https://d3js.org>)
 - Contribuer à une bibliothèque de visualisation basée sur D3.js
 - ...
- **D3.js est caractérisé par :**
 - Facilité de manipulation du DOM d'une page web, aussi bien en SVG qu'en Canvas
 - Facilité de créer des axes, labels et autres finitions pour une dataviz
 - Interpolation des différents états du DOM pour faire des transitions animées
 - ...

La Librairie D3.js

- Exemples de diagrammes
- Beaucoup d'exemples D3 à disposition : <https://d3-graph-gallery.com/>



La Librairie D3.js

Installation

- Soit par un **chargement direct à partir de d3js.org** : en l'insérant directement dans votre page HTML à partir du réseau de diffusion de contenu (CDN). Le CDN est un réseau de serveurs où les fichiers sont hébergés et livrés à l'utilisateur.

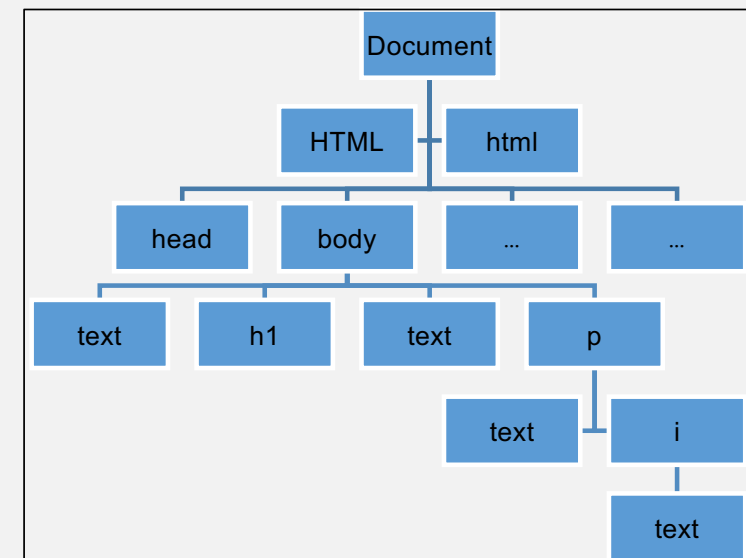
```
<head>  
  <script src = https://d3js.org/d3.v7.js> </script>  
</head>
```

- Soit en **téléchargeant sur votre ordinateur la librairie D3.js** :
 - Télécharger la dernière version de D3.js (.zip)
 - Décompresser le dossier et rechercher le fichier « **d3.min.js** » que vous copierez dans votre dossier de travail, puis intégrer le code suivant dans votre fichier html :

```
<head>  
  <script src = "chemin/d3.min.js"> </script>  
</head>  
<body>  
  <script> // écrire votre code javascript </script>  
</body>
```

La librairie D3.js – Principes (1/10)

- Lier les données au DOM (Document Object Model)
- Appliquer des transformations au document, basées sur les données.
- Principaux éléments du Framework D3 :
 1. **Sélection** : comment pointer sur les éléments de la page ?
 2. **Modification** : comment modifier ces éléments ?
 3. **Ajout** : comment ajouter des éléments ?



La librairie D3.js – Principes (2/10)

1. Sélection Comment pointer sur les éléments de la page web ?

- Un seul élément : **select()** le premier s'il y en a plusieurs
- Plusieurs éléments similaires : **selectAll()**

La sélection fonctionne comme pour le CSS :

- "balise" → <div>, <h1>, «body»...
- ".vClasse" → <balise **class**="vClasse">
- "#monId" → <balise **id**='monId'>

Exemple

```
var vCorps = d3.select("body");  
// la variable vCorps contient l'objet <body>
```

Exemple

```
var vld = d3.selectAll('vClass');  
// la variable vld contient tous les objets qui ont  
// comme attribut class = "vClass"
```

La librairie D3.js – Principes (3/10)

2. Modification Comment modifier des éléments d'une page web ?

- Informations sur la sélection :
 - **size()** taille de la sélection
 - **empty()** sélection vide ou non
- Modifier les éléments sélectionnés :
 - **style()** appliquer des règles CSS
 - **html()** modifier le contenu de la balise
- Exemple : mettre le texte en rouge pour tout le corps de la page

```
var vCorps = d3.select("body");  
vCorps.style("color", "red");
```

La librairie D3.js – Principes (4/10)

3. Ajout comment ajouter des éléments à une page web ?

- Deux fonctions :
 - Ajouter un élément HTML fils à la fin : **append()**
 - Insérer au début : **insert()**

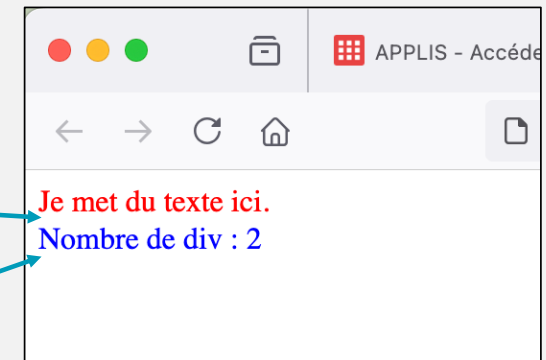
- Exemple :

```
// Sélection de la balise body
var corps = d3.select("body");

// Ajout de deux balises div au corps (child de <body>)
var div1 = corps.append("div");
var div2 = corps.append("div");

// Définition du contenu des deux balises
div1.html("Je met du texte ici.");
div2.html("Nombre de div : " + d3.selectAll("div").size());

// Modification de la couleur de la police
corps.style("color", "red"); // rouge pour tout le corps de la page
div2.style("color", "blue"); // bleu pour le 2ème div créé
```



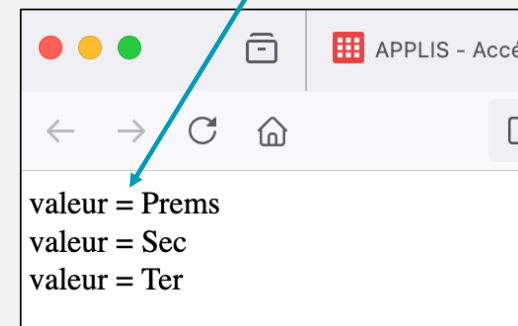
La librairie D3.js – Principes (5/10)

Lier des données au document (data binding)

- Faire correspondre une balise à un objet
- Modifier les balises en fonction des données associées
- Fonction **data()** sur une sélection :
 - Lier les données au DOM :
 - Données passées en paramètres
 - Doit être un tableau
- **Exemple** : affecter les données du tableau (qui contient du texte) à chaque div du body → on affecte un contenu en prenant comme valeur celle contenu dans le tableau.

```
<div> </div>  
<div> </div>  
<div> </div>
```

```
<script>  
  var vSelec = d3.selectAll("div");  
  vSelec.data(["Prem", "Sec", "Ter"]);  
  vSelec.html(function(d) {  
    return "valeur = " + d; })  
</script>
```



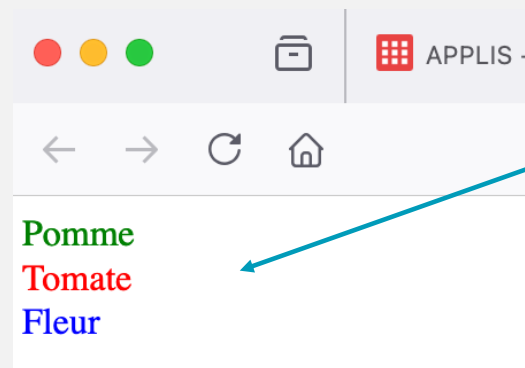
La librairie D3.js – Principes (6/10)

Lier des données au document (data binding)

- **Exemple 2** : affecter les données du tableau (qui contient des couleurs) à chaque div du body → on affecte un style CSS (la couleur), en prenant comme valeur celle contenu dans le tableau.

```
<div>Pomme</div>  
<div>Tomate</div>  
<div>Fleur</div>
```

```
<script>  
  var vdiv = d3.selectAll("div");  
  vdiv.data(["green", "red", "blue"]);  
  vdiv.style("color", function (d) { return d; });  
</script>
```



La librairie D3.js – Principes (7/10)

Propriétés dynamiques

- Sur chaque sélection, modifications de style ou de contenu, en fonction des données sont liées au DOM
- Utilisation d'une fonction anonyme, dont les paramètres peuvent être, dans cet ordre :
 - Valeur de l'élément du tableau affectée à l'élément
 - Position de la valeur dans le tableau
- Il est possible de n'utiliser que la valeur, voire aucun paramètre si besoin

```
<div> </div>
```

```
<div> </div>
```

```
<div> </div>
```

```
<script>
```

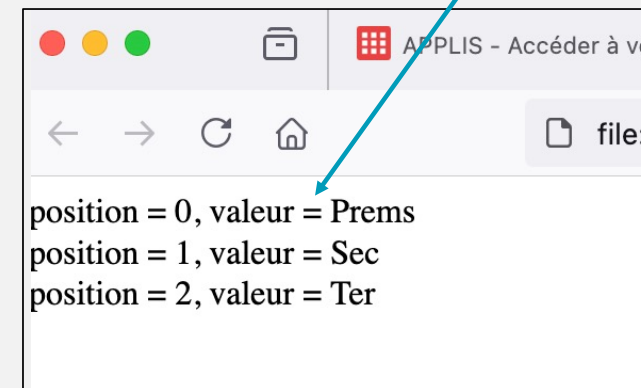
```
var vdiv = d3.selectAll("div");
```

```
vdiv.data("Prem", "Sec", "Ter");
```

```
vdiv.html(function(d, i)
```

```
{ return "position = " + i + ", valeur = " + d; })
```

```
</script>
```



La librairie D3.js – Principes (8/10)

Ajout de données + Propriété dynamique

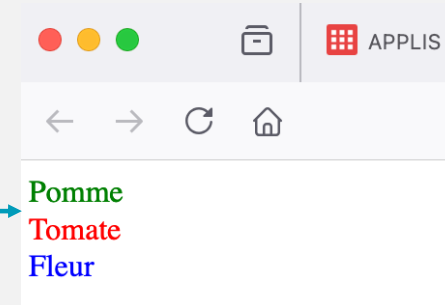
Données du tableau (des noms de couleurs ici) associées à chaque div du body

- Couleur css prenant comme valeur celle contenue dans le tableau

```
<div>Pomme</div>
<div>Tomate</div>
<div>Fleur</div>

<script>
  var vdiv = d3.selectAll("div");
  vdiv.data(["green", "red", "blue"]);
  vdiv.style("color", function (d) { return d; });
</script>
```

Cet exemple permet de modifier la couleur du premier div en vert, le 2^{ème} en bleu et le 3^{ème} en rouge.



Affecter les données du tableau data (qui contient des couleurs) à chaque div du body

→ on affecte un style CSS (la couleur), en prenant comme valeur celle contenu dans le tableau.

La librairie D3.js – Principes (9/10)

S'il y a des différences entre la taille de la sélection `selectAll` et la taille du tableau passé en paramètre `data` :

- `enter()` : permet de gérer les éléments du tableau en plus
- `exit()` : permet de gérer les éléments de la sélection en plus

```
<div> Pomme </div>
<div> Tomate </div>
<div> Fleur </div>
```

Fonction `enter()`

Tableau passé en paramètre de `data()` plus long que la sélection

- Sélection des valeurs supplémentaires avec `enter()`
- Ajout de nouveaux div (avec `append()`) pour celles-ci
- Modification de la couleur à faire

```
//vdiv : liste des balises div présentes dans le HTML
var vdiv = d3.select("body").selectAll("div");

//vdata : tableau des div avec les données associées
var vdata = vdiv.data(["green", "red", "blue", "orange", "black"]);

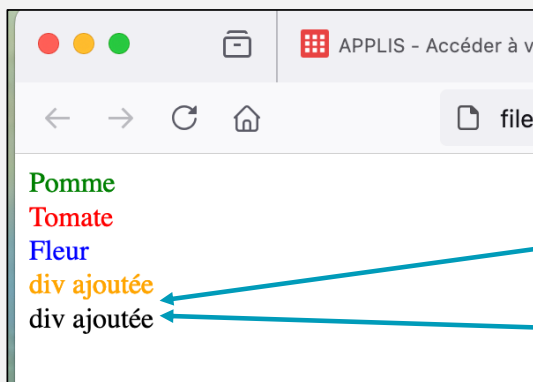
//affectation de la couleur à chaque div
vdiv.style("color", function (d) { return d; });

//venter : tableau avec les données en trop
var venter = vdata.enter();

//vnv : nouvelles div ajoutées à partir des données en trop
var vnv = venter.append("div");

//définition du contenu des nouvelles div
vnv.html("div ajoutée");

//affectation de la couleur à chaque nouvelle div
vnv.style("color", function (d) { return d; });
```



La librairie D3.js – Principes (10/10)

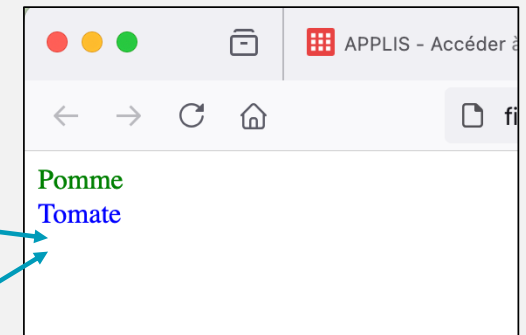
Fonction exit()

Tableau est plus petit que la sélection

- Sélection des éléments en trop avec **exit()**
- Suppression avec **remove()**

```
<div> Pomme </div>  
<div> Tomate </div>  
<div> Fleur </div>
```

```
var vDiv = d3.selectAll("div");  
//une autre variable avec les données associées  
var vData = vDiv.data(["green", "blue"]);  
//affectation de la couleur à chaque div  
vDiv.style("color", function (x) { return x; });  
//vExit : les balises div en trop  
var vExit = vData.exit();  
//suppression des div en trop  
vExit.remove();
```



PARTIE 3 : VISUALISATION AVEC D3.JS

3.3 Exemples D3.js

La Librairie D3.js – Exemple de code

```
<html>
```

```
<head>
```

```
<meta charset= "utf-8">
```

```
<title> Fichier de base D3 </title>
```

```
<script src = https://d3js.org/d3.v7.js> </script>
```

La page html se charge, lance la récupération des données D3.js, puis exécute le code de réalisation des graphiques :

1. Appel de la bibliothèque D3.js

```
<script type = "text/javascript">
```

```
  function graph1(data) {
```

```
    // le code des graphiques sera placé ici
```

```
    // ...
```

```
  }
```

```
</script>
```

```
</head>
```

3. Définition d'une fonction dans laquelle on créera des graphiques

```
<body>
```

```
<script type = "text/javascript">
```

```
  d3.json("data/data.json", graph1);
```

```
</script>
```

```
</body>
```

2. Le script de chargement des données json, qui lancera le graphique (fonction *graph1*) une fois terminée.

```
</html>
```

La Librairie D3.js – Exemple de code

```
<html>
....
<body>
  <script type = "text/javascript">
    d3.json("data/data.json", graph1);
  </script>
</body>
</html>
```

Chargement des données à partir d'un fichier externe

Les données peuvent être chargées à partir de nombreux types de fichiers externes à l'aide de commandes telles que **d3.csv** ou **d3.json**

- **D3** prend en charge des fonctions qui permettent de traiter les données ou les erreurs résultantes.

```
<script type = "text/javascript">
  var vData;

  d3.json("fichier.json", function(error, data) {
    if (error) {
      alert(error);
    } else {
      vData = data;
      alert(vData);
    }
  });
</script>
```

La Librairie D3.js

Exemple de création d'un graphique SVG (1/3)

```
<html>
....
<script type = "text/javascript">

  // Définition de la longueur et largeur du graphe
  function graphe(data) {
    var vLarg = 500;
    var vHaut = 100;
    var vPadding = 1; //Padding des barres du graphique
  }
  // Création de l'élément SVG
  var svg = d3.select("body")
    .append("svg")
    .attr("width", vLarg)
    .attr("height", vHaut)
    ...

</script>
</html>
```

→ La fonction *graphe* va permettre de définir les valeurs de variables : largeur et hauteur du graphe, largeur de l'intervalle entre deux barres (padding).

→ Création d'un élément de type svg au niveau du <body> de la page en précisant ses dimensions.

La Librairie D3.js

Exemple de création d'un graphique SVG (2/3)

```
<html>
....
<script type = "text/javascript">
    var vData = [5, 10, 15, 20, 25];
    // Lier les données à des balises
    d3.select("body") // trouver la balise <body> dans le DOM
      .selectAll ("p") // sélectionner tous les paragraphes du DOM
      .data (vData)    // affecter les données
      .enter ()        // créer un nouvel élément lié aux données
      .append ("p")    // ajouter un élément paragraphe dans le body
      .text ("Nouveau paragraphe") // ajouter du texte au paragraphe
    ...

</script>

</html>
```

La Librairie D3.js

Exemple de création d'un graphique SVG (3/3)

Spécification des échelles D3

- Les échelles sont des fonctions qui font correspondre un domaine en entrée à une plage de sortie.
- Les générateurs d'échelles de D3 sont accessibles avec **d3.scale** suivi du type d'échelle :

```
var vScale = d3.scale.linear();
```

- Les échelles peuvent aussi être ordinales et temporelles.
- Exemples de gestion des Axes d'un graphique : <https://d3js.org/d3-axis>

La Librairie D3.js – Exemple 1 – Diagramme à barres

```
html>
<head>
  <script src="https://d3js.org/d3.v7.js"></script>
  <style>
    svg rect {
      fill: red;
    }
    svg text {
      fill:white;
      font: 10px sans-serif;
      text-anchor: end;
    }
  </style>
</head>

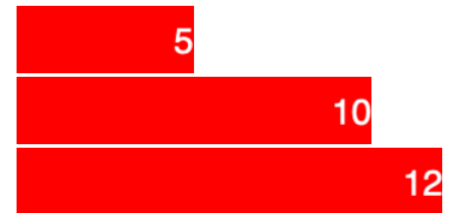
<body>
<script>
  //définition du tableau de données
  var data = [5, 10, 12];
  var width = 200; // longueur du graphique
  var scaleFactor = 10; //Echelle
  var barHeight = 20; //hauteur de la barre
  //ajout d'une balise <svg> dans le body de la page
  //création d'un conteneur pour notre graphique
  var vgraph = d3.select("body")
    .append("svg")
    .attr("width", width)
    .attr("height", barHeight * data.length);
```

```
//Lier les données pour créer les barres – la 1ère ligne crée
//autant de barres qu'il y a de données dans le tableau
var bar = vgraph.selectAll("g")
  .data(data)
  .enter()
  .append("g")
  .attr("transform", function(d, i) {
    return "translate(0," + i * barHeight + ")";
  });

//affichage des barres du graphique (un rect pour chaque g)
bar.append("rect")
  .attr("width", function(d) { return d * scaleFactor; })
  .attr("height", barHeight - 1);

//affichage des valeurs dans le graphique (text)
bar.append("text")
  .attr("x", function(d) { return (d*scaleFactor); })
  .attr("y", barHeight / 2)
  .attr("dy", ".35em")
  .text(function(d) { return d; });

</script>
```



Source : tutorialsteacher.com <https://www.tutorialsteacher.com/d3js/scales-in-d3>

La Librairie D3.js – Exemple 2 – Diagramme à barres avec échelle

```
<html>
<head>
  <script src="https://d3js.org/d3.v7.js"></script>
</head>
```

```
<body>
<script>
  //définition du tableau de données
  var data = [100, 400, 300, 900, 850, 1000];

  //définition de variables pour les dimensions des barres
  var width = 600; //longueur du graphique
  var barHeight = 20; //hauteur de la barre
  var margin = 1; //marge entre les barres

  //définition de l'échelle - mise à l'échelle des axes)
  // axe de 100 à 1000
  var scale = d3.scaleLinear()
    .domain([d3.min(data), d3.max(data)])
    .range([50, 500]);

  //ajout d'une balise <svg> dans le body de la page
  //création d'un conteneur pour notre graphique
  var svg = d3.select("body")
    .append("svg")
    .attr("width", width)
    .attr("height", barHeight * data.length)
    .style('background-color', 'lightyellow');
```

```
//Lier les données pour créer les barres – la 1ère ligne crée
//autant de barres qu'il y a de données dans le tableau
var g = svg.selectAll("g")
  .data(data)
  .enter()
  .append("g")
  .attr("transform", function (d, i) {
    return "translate(0," + i * barHeight + ")";
  });

//affichage des barres du graphique (un rect pour chaque g)
g.append("rect")
  .attr("width", function (d) {
    return scale(d);
  })
  .attr("height", barHeight - margin)
  .style('fill', 'red'); //couleur de fond des barres du graphique

//affichage des valeurs dans le graphique (text)
g.append("text")
  .attr("x", function (d) { return (scale(d)); })
  .attr("y", barHeight / 2)
  .attr("dy", ".35em") //permet de centrer la valeur affichée
  .text(function (d) { return d; });
```

```
</script>
```

Source : tutorialsteacher.com <https://www.tutorialsteacher.com/d3js/scales-in-d3>

La Librairie D3.js – Exemple 2 – Diagramme à barres avec échelle



Source : tutorialteacher.com <https://www.tutorialteacher.com/d3js/scales-in-d3>

La Librairie D3.js – Exemple 3 – Diagramme à barres avec échelle + axe des x

```
<html>
<head>
  <script src="https://d3js.org/d3.v7.js"></script>
</head>
```

```
<body>
<script>
```

//définition du tableau de données

```
var data = [100, 400, 300, 900, 850, 1000];
```

//définition de variables pour les dimensions des barres

```
var width = 600; //longueur du graphique
```

```
var barHeight = 20; //hauteur de la barre
```

```
var margin = 1; //marge entre les barres
```

//définition de l'échelle - mise à l'échelle des axes)

// axe de 100 à 1000

```
var scale = d3.scaleLinear()
```

```
  .domain([d3.min(data), d3.max(data)])
```

```
  .range([50, 500]);
```

//ajout d'une balise <svg> dans le body de la page

//création d'un conteneur pour notre graphique

```
var svg = d3.select("body")
```

```
  .append("svg")
```

```
  .attr("width", width)
```

```
  .attr("height", barHeight * data.length)
```

```
  .style("background-color", 'lightyellow');
```

//Lier les données pour créer les barres – la 1^{ère} ligne crée

//autant de barres qu'il y a de données dans le tableau

```
var g = svg.selectAll("g")
```

```
  .data(data)
```

```
  .enter()
```

```
  .append("g")
```

```
  .attr("transform", function (d, i) {
```

```
    return "translate(0," + i * barHeight + ")";
```

```
  });
```

//affichage des barres du graphique (un rect pour chaque g)

```
g.append("rect")
```

```
  .attr("width", function (d) {
```

```
    return scale(d);
```

```
  })
```

```
  .attr("height", barHeight - margin)
```

```
  .style('fill', 'red'); //couleur de fond des barres du graphique
```

//affichage des valeurs dans le graphique (text)

```
g.append("text")
```

```
  .attr("x", function (d) { return (scale(d)); });
```

```
  .attr("y", barHeight / 2)
```

```
  .attr("dy", ".35em") //permet de centrer la valeur affichée
```

```
  .text(function (d) { return d; });
```

//Ajout de l'axe des x en bas du graphique

var xEchelle = d3.scaleLinear() //création d'une échelle pour l'axe des x

```
  .domain([0, 1000])
```

```
  .range([5, 500]);
```

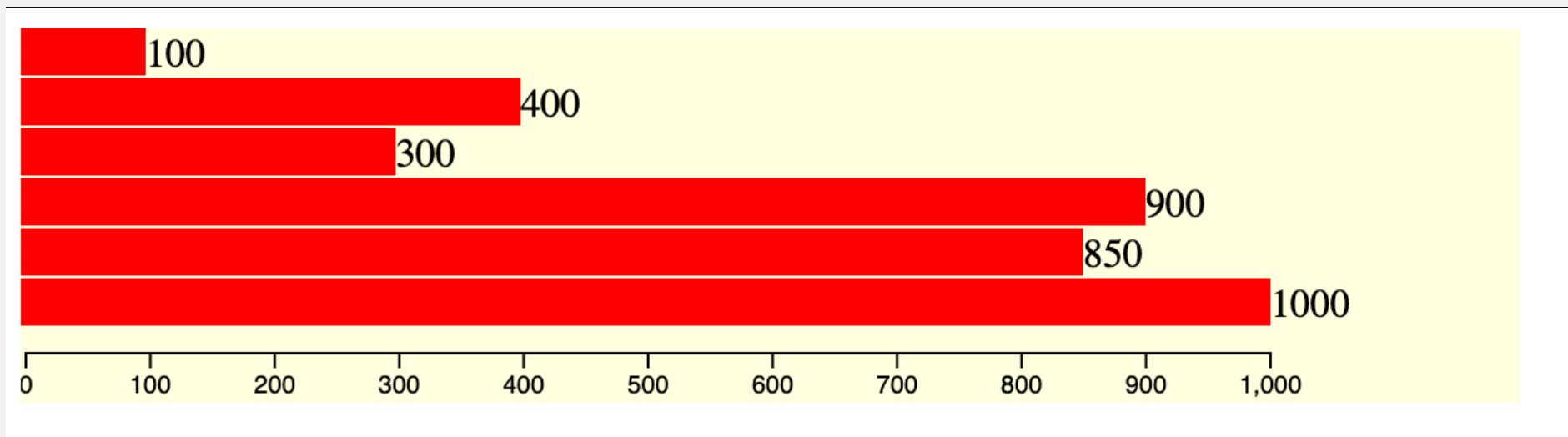
```
var vg = svg.append("g")
```

```
  .attr('transform', 'translate(0, 130)') //pour mettre l'axe en bas du graphique
```

```
  .call(d3.axisBottom(xEchelle)); //la fonction axisBottom() crée l'axe
```

```
</script>
```

La Librairie D3.js – Exemple 3 – Diagramme à barres avec échelle + axe des x



PARTIE 3 : VISUALISATION AVEC D3.JS

3.4 Les principales fonctions SVG utilisées en D3.js

Rappel : Fonctions en javascript

Les fonctions fléchées

- Les fonctions fléchées sont une nouvelle syntaxe pour écrire des fonctions JavaScript.
- Elles font gagner du temps aux développeurs et simplifient la portée des fonctions

Fonctions classiques :

- **function** somme(n1,n2) {
 document.write(n1+n2);
}

Fonctions fléchées :

- somme = (n1 , n2) => document.write(n1 + n2);

- maFonction = **function**(a, b) { return a+b; }
- maFonction = **function**(a) { return a*a; }

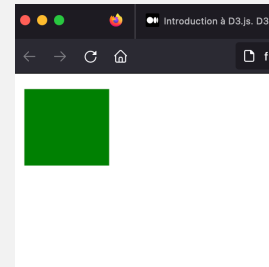
- maFonction = (a, b) => a+b;
- maFonction = a => a*a;

1. Les graphiques en HTML

- **SVG** (Scalable Vector Graphics) : SVG est un langage qui permet de décrire des images vectorielles (i.e. qui décrivent des lignes et des formes).
- Une image vectorielle n'est pas pixelisée, comme avec des images dites matricielles (tableaux de carrés colorés), elle est donc plus nette quel que soit la résolution de l'écran.
- Les formes de base de SVG sont : *circle*, *rect*, *ellipse*, *line*, *polyline*, *polygon* et *path*

Ex. 1

```
//Dessiner un rectangle vert sur la page
<svg width="300" height="200">
  <rect width="100%" height="100%" fill="green" />
</svg>
```



Ex. 2

```
<svg width="300" height="200" >
  <rect width="100%" height="100%" fill="red" />
  <circle cx="150" cy="100" r="80" fill="green" />
  <text x="150" y="125" font-size="60" text-anchor="middle" fill="white">SVG</text>
</svg>
```



SVG Tutorial →

<https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial>

https://edutechwiki.unige.ch/fr/Tutoriel_SVG_avec_HTML5

SVG Formes de base →

https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial/Basic_Shapes

Attributs des éléments <SVG> <https://developer.mozilla.org/fr/docs/Web/SVG/Attribute>

- Les éléments **SVG** peuvent être modifiés en utilisant des attributs qui spécifient comment les éléments doivent être traités ou présentés
- Les principaux attributs :
 - **class** l'attribut class permet de donner le nom d'une classe à un élément
 - **color** l'attribut color est utilisé pour définir une valeur potentielle pour les attributs fill...
 - **cx** l'attribut cx définit la coordonnée de l'axe x pour le point central d'un élément, utilisé pour les éléments : **<circle>**, **<ellipse>**, et **<radialGradient>**
 - **cy** l'attribut cy définit la coordonnée de l'axe y pour le point central d'un élément (idem cx)
 - **r** l'attribut r définit le rayon, il est utilisé pour les éléments : **<circle>**, **<radialGradient>**
 - **font-family, font-size, font-style...** Utilisés pour la mise en forme d'éléments **<text>**
 - **height** l'attribut height définit la longueur verticale d'un élément
 - **point** l'attribut point définit une liste de points. Chaque point est défini par deux nombres représentant les coordonnées X et Y
 - **transform** l'attribut transform définit une liste de définitions de transformation qui sont appliquées à l'élément ainsi qu'à ses éléments fils

Attribut Transform

- Les éléments SVG peuvent être manipulés à l'aide de fonctions de transformation.
- L'attribut **transform** peut être utilisé avec n'importe quel élément SVG.
- L'attribut **transform** définit une liste de **fonctions** de transformation qui peuvent être appliquées à un élément (balise) et à ses enfants
 - **translate()** : La fonction *translate()* est utilisée pour déplacer un objet de x et y.
 - **scale()** : La fonction *scale()* permet de mettre un objet à l'échelle en fonction de x et y
 - **rotate()** : La fonction *rotate()* est utilisée pour faire pivoter un objet d'un degré
 - **skewX()** : La fonction *skewX()* est utilisée pour incliner d'un degré un objet le long de l'axe des x
 - **skewY()** : La fonction *skewY()* est utilisée pour incliner d'un degré un objet le long de l'axe des y

Exemple de transformations : https://www.w3schools.com/graphics/svg_transformations.asp

Les transformations de base : https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial/Basic_Transformations

Fonction Translate()

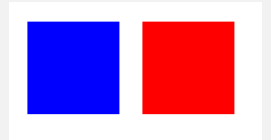
- La fonction `translate()` est utilisée pour déplacer un objet de `x` et `y`.
- Supposons qu'un objet soit placé en `x=« 5 »` et `y=« 5 »`. Ensuite, un autre objet contient `transform=`**"`translate(50 0)`"**, ce qui signifie que l'autre objet sera placé à la position `x = 55 (5 + 50)` et à la position `y = 5 (5 + 0)`.
- Exemple : Dans cet exemple, le rectangle rouge est translaté/déplacé au point (55,5) au lieu de (5,5) :

```
<body>

<h2>SVG la fonction translate() </h2>

<svg width="200" height="100" xmlns="http://www.w3.org/2000/svg">
  <rect x="5" y="5" width="40" height="40" fill="blue" />
  <rect x="5" y="5" width="40" height="40" fill="red" transform="translate(50,0)" />
</svg>

</body>
```



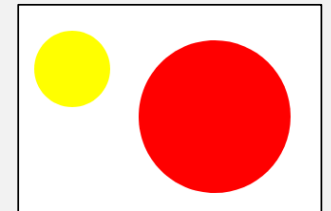
Fonction scale()

- La fonction **scale()** permet de mettre un objet à l'échelle en fonction de x et y (si y n'est pas fourni, il est égal à x).
- La fonction **scale()** permet de modifier la taille d'un objet. Elle prend 2 nombres : le facteur d'échelle x et le facteur d'échelle y. x et y sont considérés comme le rapport entre la dimension transformée et la dimension originale. Par exemple, 0,5 réduit l'objet de 50 %.
- Dans cet exemple, le cercle rouge est redimensionné au double de sa taille à l'aide de la fonction scale().

```
<body>

  <svg width="200" height="100">
    <circle cx="25" cy="25" r="20" fill="yellow" />
    <circle cx="50" cy="25" r="20" fill="red" transform="scale(2)" />
  </svg>

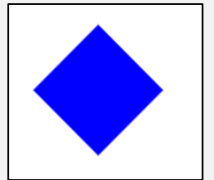
</body>
```



Fonctions rotate(), skewX(), skewY() et matrix()

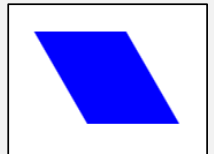
- La fonction rotate() est utilisée pour faire pivoter un objet de 45 degrés

```
<svg width="200" height="100">  
  <rect x="50" y="5" width="40" height="40" fill="blue" transform="rotate(45)" />  
</svg>
```



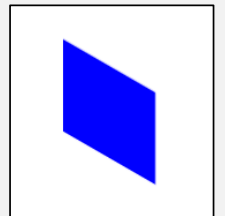
- La fonction skewX() est utilisée pour incliner de 30 degrés un objet le long de l'axe des x

```
<svg width="200" height="50">  
  <rect x="5" y="5" width="40" height="40" fill="blue" transform="skewX(30)" />  
</svg>
```



- La fonction skewY() est utilisée pour incliner de 30 degrés un objet le long de l'axe des y

```
<svg width="200" height="50">  
  <rect x="5" y="5" width="40" height="40" fill="blue" transform="skewY(30)" />  
</svg>
```



Javascript D3.js

Pour en savoir plus

Canvas & SVG

- **Canvas Tutorial** → https://developer.mozilla.org/fr/docs/Web/API/Canvas_API/Tutorial
- **SVG Tutorial** → <https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial>
https://edutechwiki.unige.ch/fr/Tutoriel_SVG_avec_HTML5
- **SVG Formes de base** → https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial/Basic_Shapes

D3.js

- **Framework D3.js Exemples de graphes** : https://d3-graph-gallery.com/intro_d3js.html
- **Introduction à D3.js et aux documents pilotés par les données** :
<https://www.miximum.fr/blog/introduction-a-d3-js-et-aux-documents-pilotes-par-les-donnees/>
- **Tutorial 1 : D3.js premiers graphiques** : <https://grafikart.fr/tutoriels/graph-d3js-2215>
- **Tutorial 2: Visualisation des données D3.js** : <https://fxjollois.github.io/cours-2019-2020/du-dataviz--d3js.html>
- **Tutorial 3 : Introduction à D3.js** : <https://medium.com/@OVHUXLabs/introduction-%C3%A0-d3-js-ccaed9dfde03>
- **Tutorial 4 : D3.js Différents types de graphes** : <https://www.tutorialsteacher.com/d3js/create-bar-chart-using-d3js>

Exemple 1 :

- A partir d'un fichier HTML vierge, créez dynamiquement une page web avec les éléments suivants :
 - Un titre H1
 - Un 1^{er} paragraphe
 - Un titre H2
 - Un 2^{ème} paragraphe
 - Une liste ordonnée avec 3 éléments
- Modifier la couleur du texte dans le 1^{er} paragraphe
- ...

```
<!DOCTYPE html>
<head>
  <meta charset= "utf-8">
  <script src = https://d3js.org/d3.v7.js> </script>
</head>

<html>
  <body>

    </body>
</html>
```

Exemple 2 :

- Reprendre l'exemple de la diapo 37 (diagramme à barre)
- Faire saisir les 3 valeurs du graphique par l'utilisateur (utiliser prompt)
- Afficher le graphique avec les valeurs saisies
- Faites en sorte que la couleur soit différente pour chacune des barres du graphique