Guide TP2

Version JavaScript

Technologies

Pour ces labs, nous utilisons:

- HTML / SVG / CSS
- JavaScript + D3.js

Bien que les TPs peuvent être accomplis avec d’autres outils, nous nous attendons à ce que vous utiliser ceux-ci lorsque possible - D3 en particulier!

Technologies

HTML

“Hypertext Markup Language” est utilisé pour structurer du contenu web.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Mon titre</title>
    </head>
    <body>
        <h1>Introduction</h1>
        <p>Ceci est un paragraphe intéressant..</p>
    </body>
</html>
```

- <...> ouvrant, </…> fermant
- DOM - Document Object Model (structure hiérarchique du HTML)
  - <...> </…> est un élément
  - Relations:
    - Enfants (h1 et p)
    - Parents (body)

Technologies

SVG

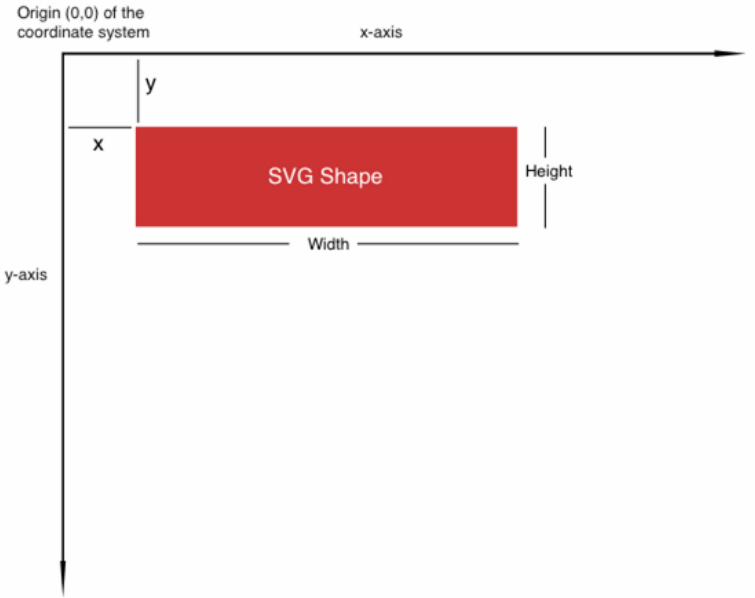
- “Scalable Vector Graphic”
- Élément HTML utile pour les visualisations de données

Exemples :

- <circle>
  - Cercles avec attributs : cx, cy, r
- <rectangle>
  - Rectangle avec attributs : x, y, width, height

Attention! Le système de coordonnées commence en haut à gauche.

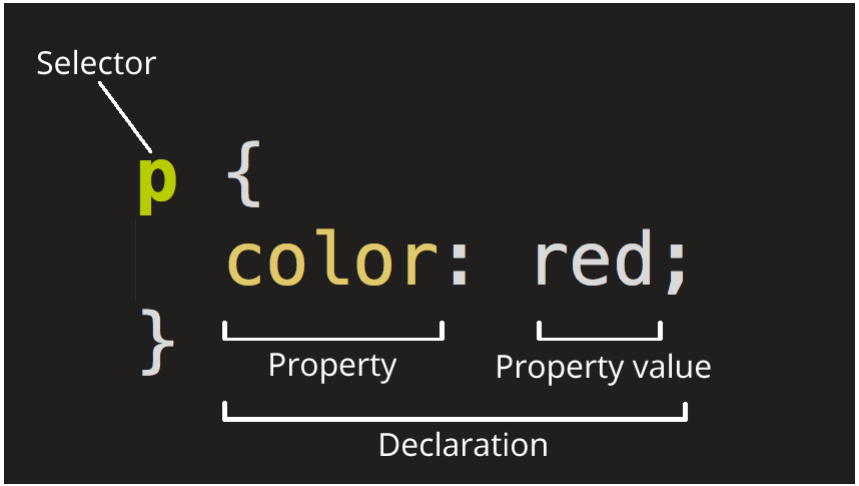
Rectangle SVG et système de coordonnées (source)



Technologies

CSS

- Décrit la présentation visuelle d'une page HTML
- Utilise des **sélecteurs** (« *selectors* ») and et déclarations (« *declarations* »)



CSS (source)

Technologies

CSS : Sélecteurs et déclarations

- Exemples de sélecteurs
  - Par type d'élément

```css
p {
  font-weight: bold;
}
```

Tous les éléments <p> seront en gras (« bold »)

  - Sur un élément : **par id**

```css
#my-pretty-text {
  font-family: 'Times New Roman';
}
```

La police de l'élément "my-pretty-text" sera "Times New Roman"

```html
<p id="my-pretty-text">Hello!</p>
```

Élément HTML correspondant

  - Sur plusieurs éléments

```css
.another-text{
  font-size: 12px;
}
```

La police des éléments avec la classe "another-text" sera de 12px

```html
<p class="another-text">Goodbye!</p>
```

Élément HTML correspondant

Technologies

CSS : Sélecteurs et déclarations

- Déclarations
  - Déterminent l'apparence visuelle du ou des éléments sélectionnés
  - Exemples…

- **width, height** : La hauteur et la largeur du ou des éléments
- **fill** : couleur du ou des éléments
- **margin** : marge autour du ou des éléments

Technologies

JavaScript

Langage de script qui peut, entre autre, rendre les pages dynamiques en manipulant le DOM.

Il est interprété et faiblement typé.

Points à souligner:

- Objets et tableaux: Des structures de données qui peuvent aider à traiter les données.
- JSON: Syntaxe spécifique pour organiser les données sous forme d'objets JavaScript. Peut être plus facile à analyser et meilleur pour les travaux D3.
- Fonctions: Prennent des arguments ou des paramètres en entrée, puis retournez des valeurs en sortie.

Technologies

JavaScript

Pour l'utiliser :

- Directement dans le HTML, entre deux tags s*cripts* :

```
<body>
    <script type="text/javascript">
        alert("Hello, world!");
    </script>
</body>
```

- Inclus dans le HTML en référant vers un fichier avec une extension **.js**

```
<head>
    <title>Page Title</title>
    <script type="text/javascript" src="myscript.js"></script>
</head>
```

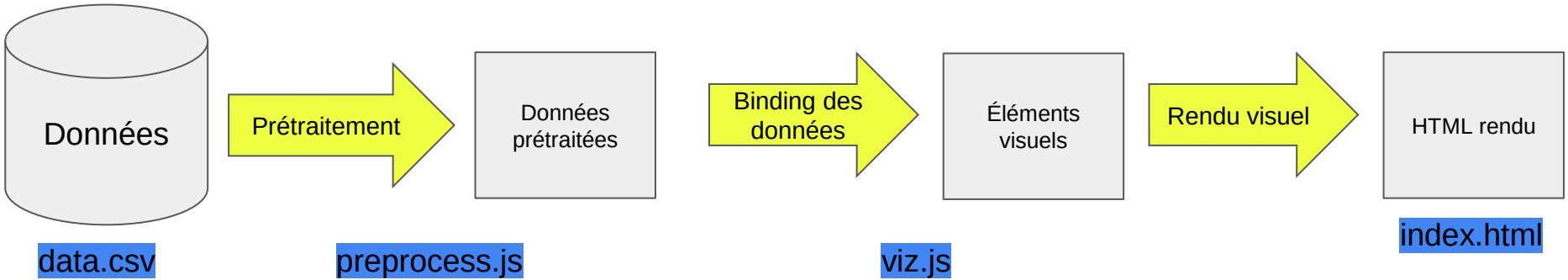
Utilisé dans les TPs

Technologies

D3.js

- D3 réfère à « document basé sur les données »
- Bibliothèque JS
- Nous utiliserons la version 5 (**Important** : lors de la recherche d'exemples, assurez-vous qu'ils sont > v4, sinon ils pourraient être assez différents!)
- D3 peut être utilisé pour traiter et visualiser des données, ex:
  - `d3.select ("svg"). append ("circle"). attr ("r", 10);`
    - Sélectionne le premier élément avec la balise «svg» et y ajoute un cercle de rayon 10

Étapes typiques dans les TPs



Pistes de solutions

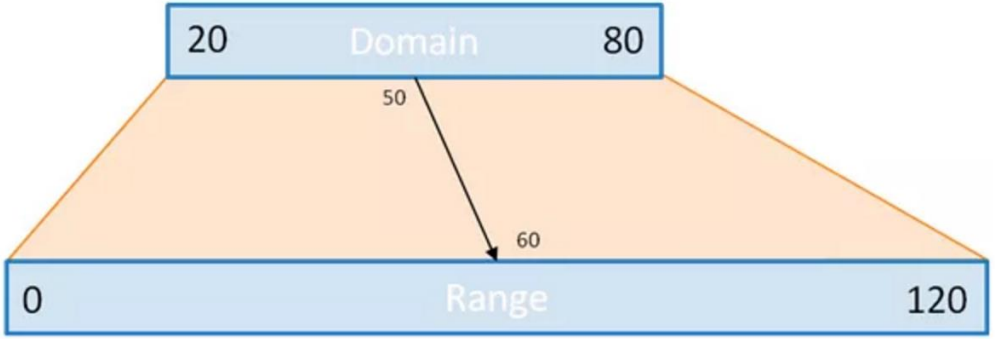
D3 Scales

- Dans **viz.js**, vous devez déterminer le domaine et l'image d'échelles D3

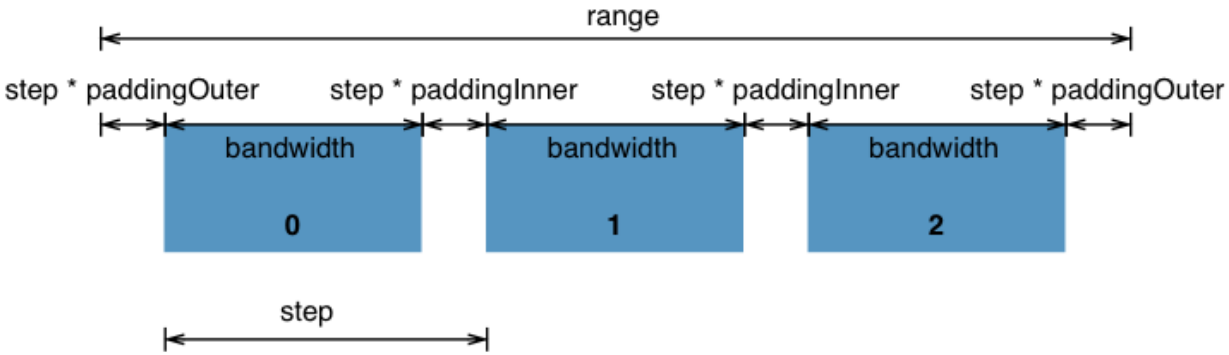
- Dans ce TP, les échelles aident pour tracer les axes et dessiner les groupes de rectangles
  - Une échelle de couleurs, 2 pour la position en X (une pour placer les groupes et l'autre les sous-éléments de ceux-ci) et une pour la longueur en Y

D3 scales

Visuellement



Source : https://medium.com/@sahilaug/line-graphs-using-d3-drawing-the-axes-8ffc0076a8be



Source : https://github.com/d3/d3-scale

Pistes de solutions

Data Binding

- Dans ce TP, vous allez utiliser les fonctionnalités de **data binding** de D3
- 2 structures :

1. Approche classique :

```
d3
  .selectAll('rect')
  .data(myData)
  .enter()
  .append('rect')
```

2. Approche plus nouvelle :

```
d3
  .selectAll('rect')
  .data(myData)
  .join('rect')
```

- Voir aussi la doc : https://github.com/d3/d3-selection

Data binding en D3

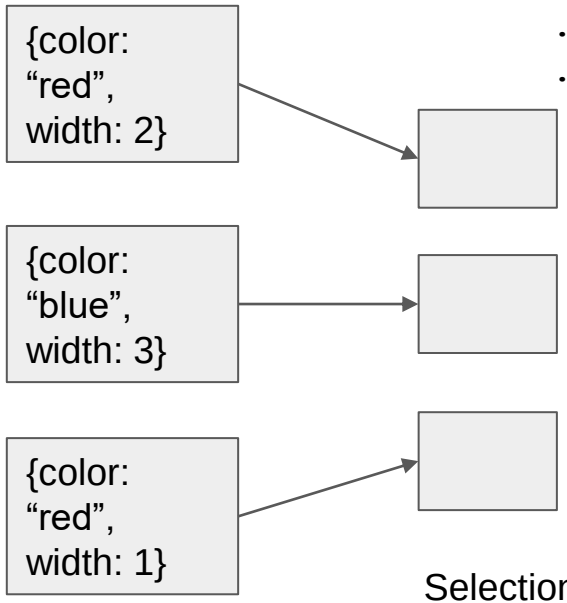
rectangles.js

```
svg.selectAll('rect')
.data(data)
.enter()
.append('rect')
.attr('fill', (d) =>  d.color)
.attr('width', (d) => d.width)
.attr('height' 25)
.attr('y', (d, i) => i*50)
```

Visuellement

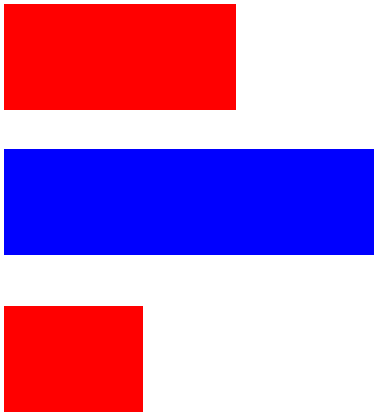
data.csv

color, width
red, 2
blue, 3
red, 1



Data

Selection



Résultat

Data binding en D3

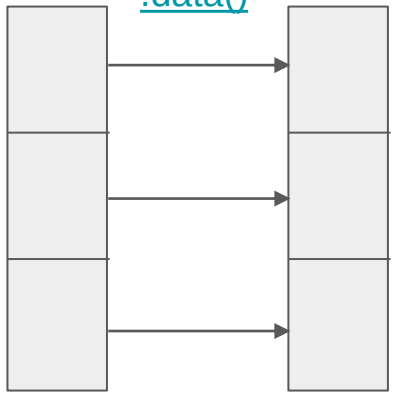
.data() VS .datum()

Deux fonctions avec des noms semblables, mais comportements différents :

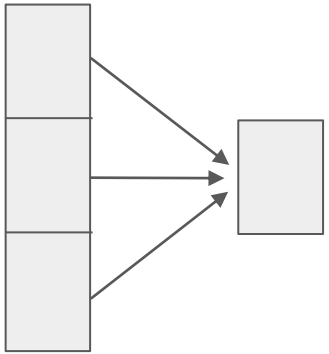
- **.data()** “bind” (ou « associe ») un élément des données par élément dans la sélection
- **.datum()** “bind” (ou « associe ») tous les éléments des données à la sélection

Dans les TP, il est recommandé d’utiliser .data() pour éviter les erreurs conceptuelles

.data()



.datum()



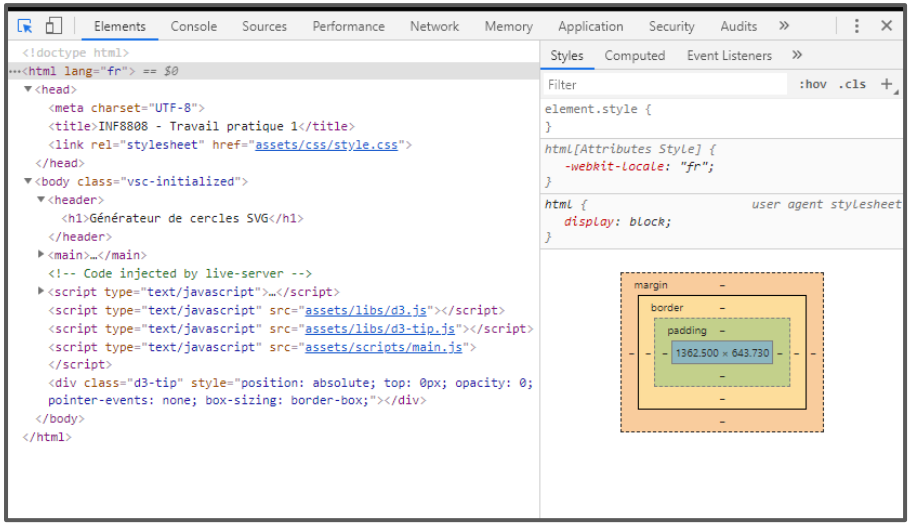
Conseils généraux de débogage

Débogage avec Chrome

Notez que d'autres fureteurs offrent des outils semblables.

1. Cliquez avec le bouton droit n'importe où sur la page et sélectionnez «*Inspect*» OU Ctrl + Maj + I
2. L'inspecteur s'ouvre, ce qui est utile pour le débogage
3. Les onglets «Éléments», «Console» et «Sources» seront les plus utiles pour ces TP (voir les diapositives suivantes)

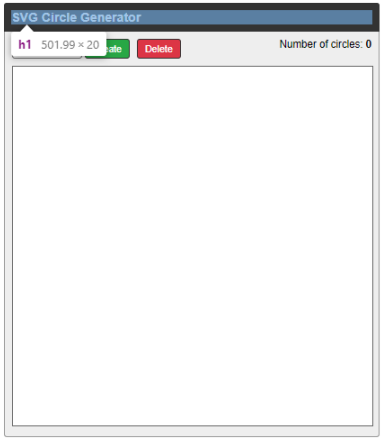
Inspecteur Chrome



Débogage avec Chrome

Inspection d'élément

- Affiche la structure HTML (DOM) de votre code
- Met en évidence l'élément actuellement survolé sur la page
- Utilisez-le pour vérifier que votre code D3 génère correctement des éléments HTML
- Vous permet de tester directement des modifications au HTML et CSS



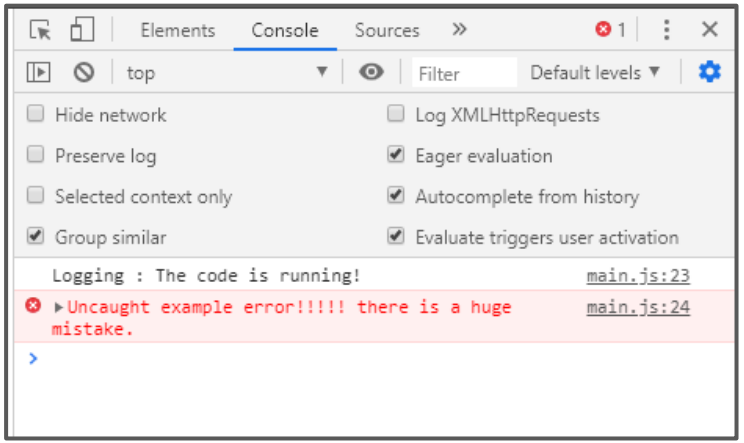
Inspection d'un élément h1

Débogage avec Chrome

Console

- Dans la console, vous verrez les sorties de votre code
- Celles-ci peuvent inclure des messages d'erreur et des « *logs* »
- Si quelque chose ne fonctionne pas, c'est le premier endroit où vous devriez regarder

```
function update() {
  console.log("Logging : The code is running!")
  throw "example error!!!!! there is a huge mistake."
}
```

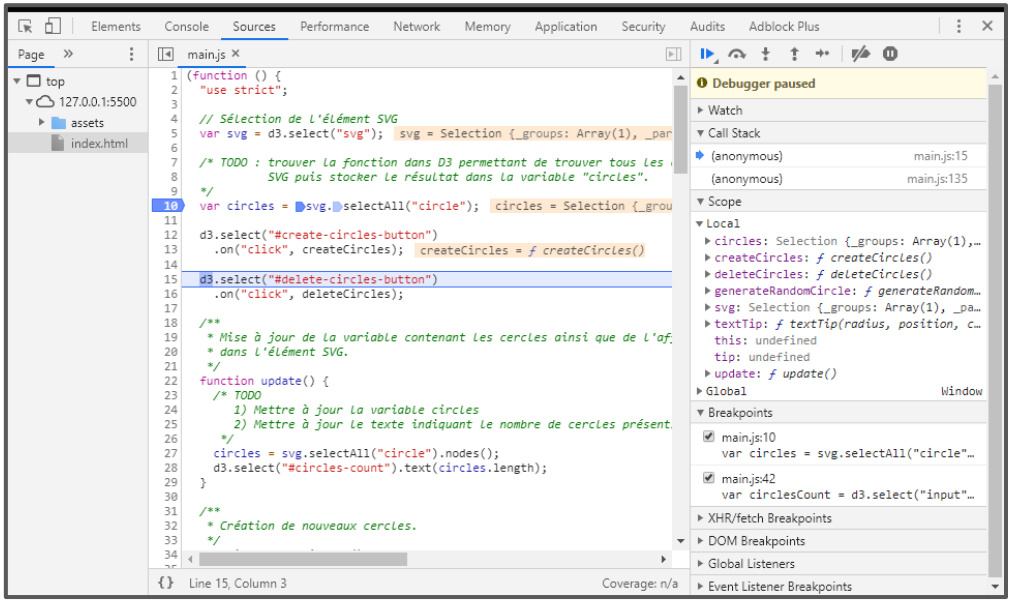


Des messages d'erreur et de « logs » apparaissent dans la console

Débogage avec Chrome

Sources inspection tool

- Dans cet onglet, vous pouvez voir votre code source et tester sa modification
- Vous pouvez également ajouter des points d'arrêt, où l'exécution s'arrêtera
- À partir d'un point d'arrêt, vous pouvez voir la valeur de chaque variable et parcourir le code ligne par ligne

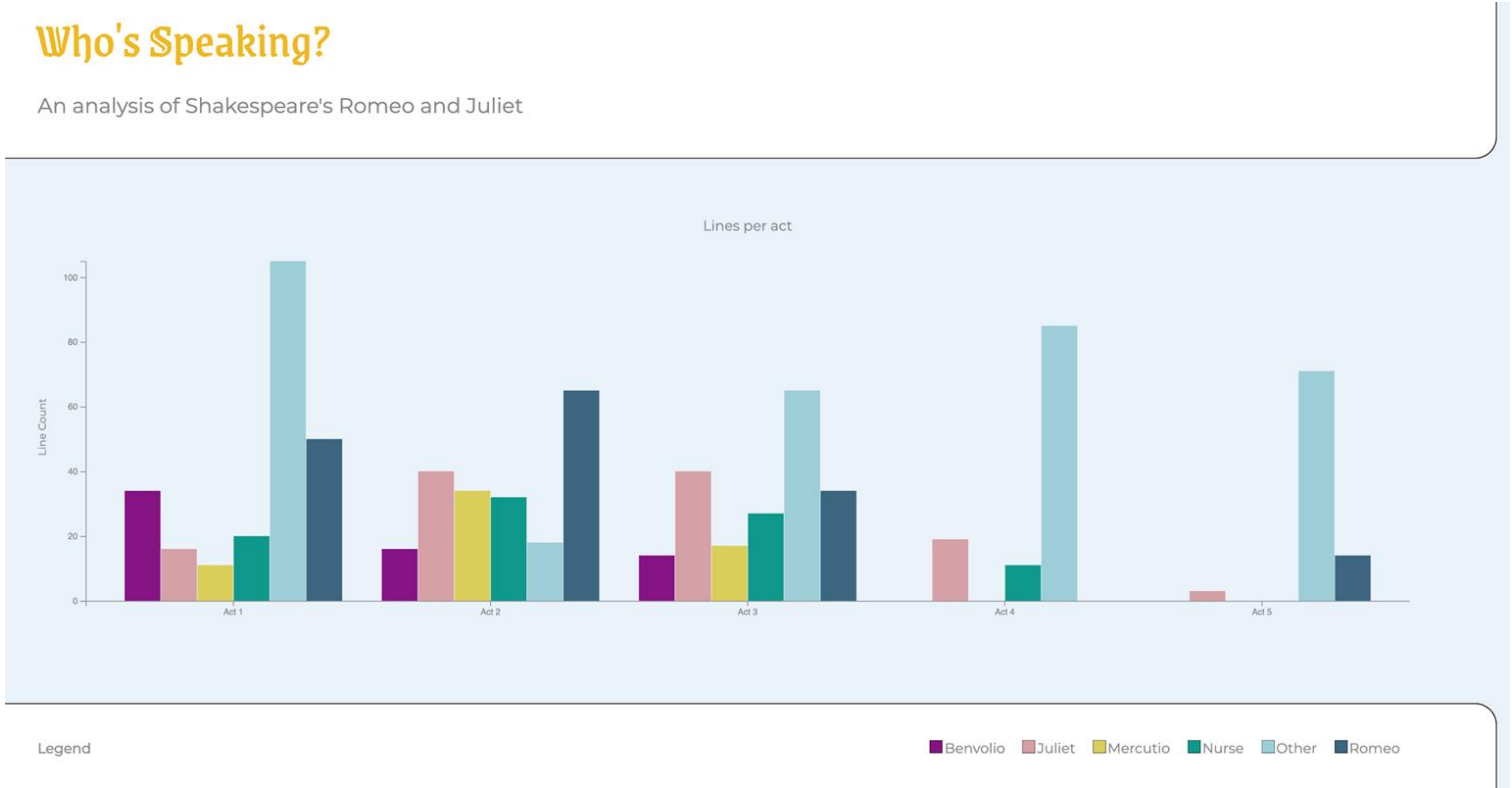


Parcours de code D3

TP2

Introduction au TP2

Dans le TP2, vous allez créer un diagramme à bandes (« *bar chart* ») à partir de données tirées de la pièce de théâtre *Roméo et Juliette* de Shakespeare.



Rouler le code

- Dans un terminal, au même niveau que *package.json:*

```
npm install

npm start
```

- Puis, consultez localhost:8080 dans votre fureteur

Résultat :

Ensemble de données

Les données représentent le texte de la pièce de théâtre. Elles se trouvent dans le fichier *./src/assets/data/romeo_and_juliet.csv*

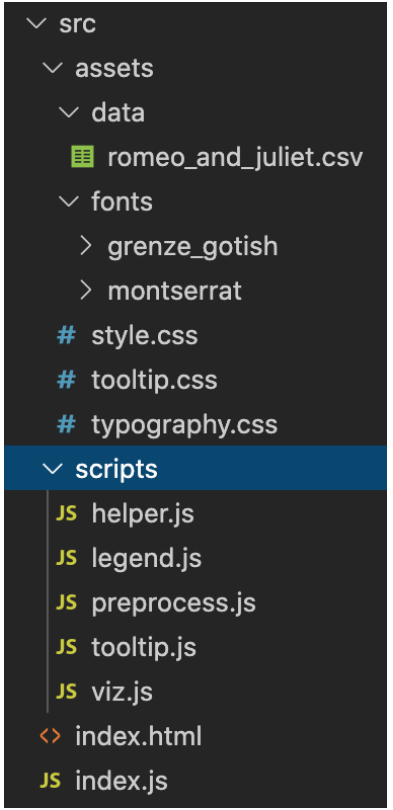
- **Act:** Cette colonne représente l'acte dans lequel la ligne est prononcée.
- **Scene:** Cette colonne représente la scène dans laquelle la ligne est prononcée.
- **Line:** Pour une *n*-ième ligne dans une scène donnée, cette colonne contient la valeur *n*.
- **Player:** Cette colonne contient le nom du joueur qui a prononcé la ligne.
- **PlayerLine:** Cette colonne contient le contenu prononcé par le joueur pour cette ligne.

```
Act,Scene,Line,Player,PlayerLine
1,0,1,RICHMOND,"Two households, both alike in dignity, / In fair
1,1,1,SAMPSON,"Gregory, o' my word, we'll not carry coals."
1,1,2,GREGORY,"No, for then we should be colliers."
1,1,3,SAMPSON,"I mean, an we be in choler, we'll draw."
1,1,4,GREGORY,"Ay, while you live, draw your neck out o' the coll
1,1,5,SAMPSON,"I strike quickly, being moved."
1,1,6,GREGORY,But thou art not quickly moved to strike.
1,1,7,SAMPSON,A dog of the house of Montague moves me.
1,1,8,GREGORY,"To move is to stir, and to be valiant is to stand:
```

Tâches à accomplir :

1. **Prétraitement des données**
   - Fichier : ./src/scripts/*preprocess.js*
1. **Création du diagramme à bandes**
   - Fichier : ./src/scripts/*viz.js*
1. **Completion de la légende**
   - Fichier : ./src/scripts/*legend.js*
1. **Ajout d'une info-bulle**
   - Fichier : ./src/scripts/*legend.js*

Les autres fichiers ne doivent pas être modifiés.



1. Prétraitement des données

4 fonctions dans le fichier **preprocess.js** à remplir pour prétraiter les données :

1. cleanName
2. getTopPlayers
3. summarizeLines
4. replaceOthers

Le résultat sera une structure de données telle que celle à droite :

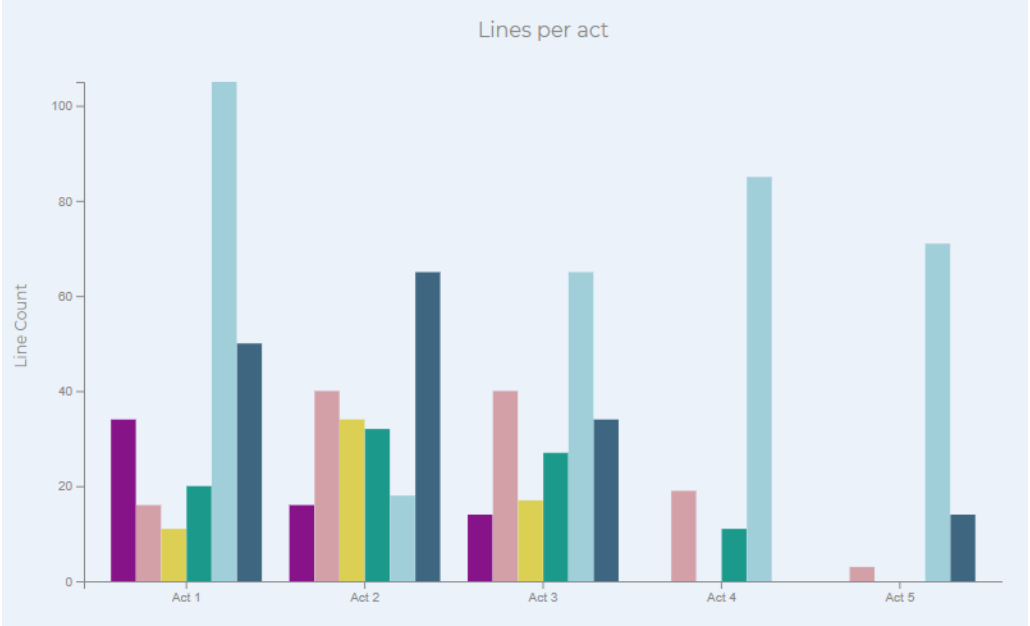

```
[
  {
    "Act": "1",
    "Players": [
      {
        "Player": "Benvolio",
        "Count": 34
      },
      {
        "Player": "Romeo",
        "Count": 50
      },
      {
        "Player": "Nurse",
        "Count": 20
      },
      {
        "Player": "Juliet",
        "Count": 16
      },
      {
        "Player": "Mercutio",
        "Count": 11
      },
      {
        "Player": "Other",
        "Count": 105
      }
    ]
  },
```

2. Diagramme à bandes

4 fonctions dans **viz.js** pour tracer les bandes :

1. updateGroupScale
2. updateYScale
3. createGroups
4. drawBars

Le résultat de cette partie est le suivant à droite :



3. Légende

Remplissez le code dans **legend.js** pour tracer la légende.

Portez une attention particulières aux éléments HTML et aux classes CSS déjà fournis pour vous aider.

Le résultat de cette partie est ci-bas :

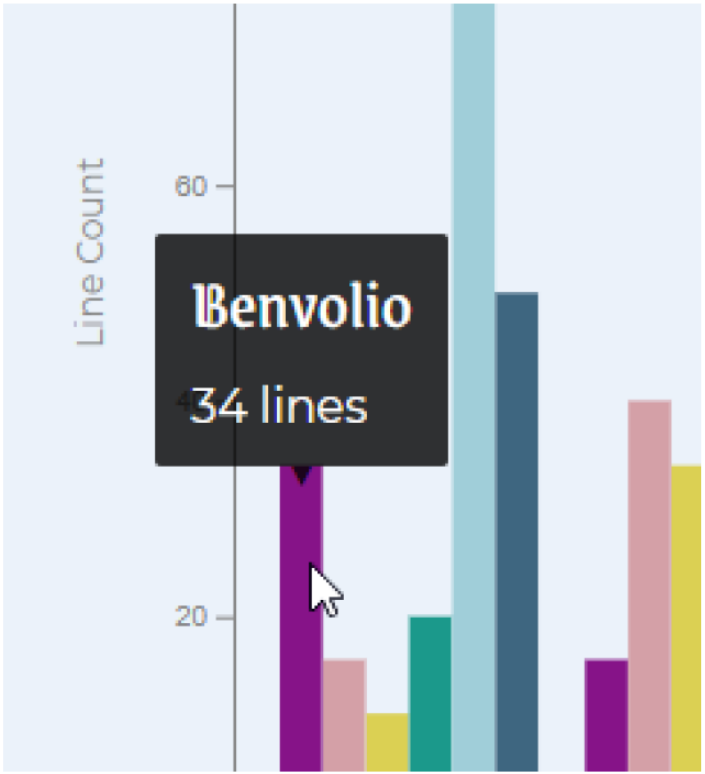
4. Info-bulle

Remplissez le code dans **tooltip.js** pour tracer l’info-bulle.

L’info-bulle contiendra le nombre de l’acte, le nom du joueur et son décompte.

Conseil : Portez une attention particulière à l’apparence visuelle de la bulle.

Le résultat est tel qu’ici à droite :



Quelques conseils pour démarrer

Prétraitement des données

- Évitez de manipuler directement les indices de données lorsque cela est possible (c'est-à-dire une boucle for index `i` → `data [i]`)
  - Cette pratique améliorera la qualité et la maintenabilité du code tout en réduisant le risque d'erreurs
- Au lieu de cela, explorez les méthodes Array de JS, telles que:
  - `.foreach()`
  - `.map()`
  - `.reduce()`
  - `.filter()`
  - `.find()`
- Utiliser `for ... of` peut aussi parfois être utile

Qualité globale et clarté de la soumission

Plus de détails

Chaque TP sera aussi noté sur la **qualité globale et clarté de la soumission**

Quelques exemples de points à surveiller:

- Essayez de structurer le code clairement
- Ne modifiez pas les signatures des fonctions existantes
- Vous pouvez ajouter de nouvelles fonctions, mais cela ne devrait pas être nécessaire
- Alignez votre code de manière cohérente
- Ajoutez des commentaires pertinents au besoin, sans laisser de commentaires inutiles
- Ne laissez pas console.log inutile
- Suivez attentivement les directives de soumission

Etc.

Directives de soumission

- Une soumission par équipe
- Soumettre dans la boîte de soumission JS sur Moodle

Directives de soumission

- Sur Moodle, soumettez un fichier .zip (pas .rar) nommé matricule1_matricule2_matricule3_.zip
- Le zip doit contenir les mêmes fichiers et la même structure que lors du téléchargé à partir de moodle
- Ne pas inclure les dossiers node_modules, .cache et dist.
- Zippez uniquement les fichiers initiaux, ne créez pas de nouveaux dossiers dans le fichier .zip.

Result example: