Utilisez la librairie pour enrichir votre projet

5–7 minutes

Placez les balises script et canvas dans le fichier HTML

La documentation nous indique qu'il faut ajouter une balise script dans notre fichier HTML pour importer la librairie dans notre page. Comme nous l'avons vu au chapitre précédent, les paquets téléchargés avec la commande NPM s'installent dans le répertoire node_modules. En explorant ce répertoire, on peut trouver l'emplacement du fichier à importer : node_modules/chart.js /dist/chart.umd.js.

Nous ajoutons donc la balise script avec le bon chemin, dans le fichier index.html :

```
<head>
    ...
    <script src="node_modules/chart.js
/dist/chart.umd.js" defer></script>
</head>
```

Continuons de suivre la documentation. Sur la page Get Started, on nous indique qu'il faut ajouter une **balise canvas**. Cette balise permet de dessiner librement à l'écran, en utilisant des lignes et des formes géométriques.

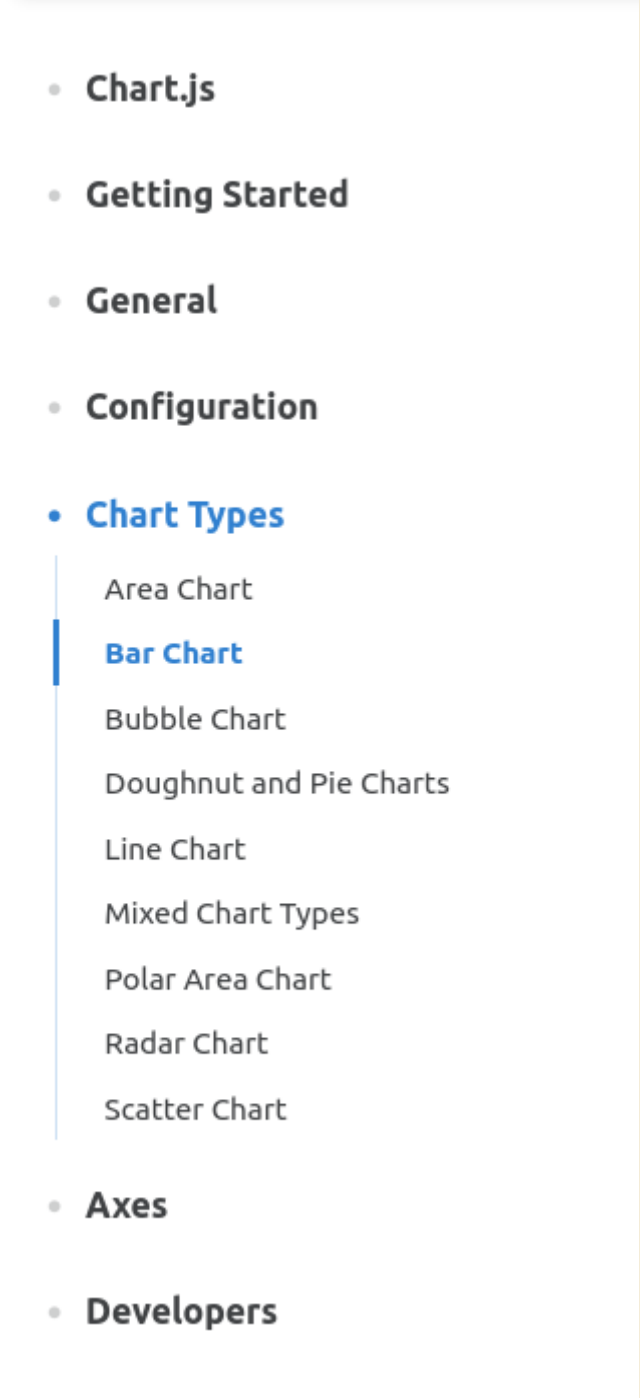
Ajoutons la balise canvas dans la balise section, après le bouton de mise à jour des pièces :

```
<h3>Aperçu des avis</h3>

<canvas id="graphique-avis" width="300" height="300">
</canvas>
```

Calculez les données à afficher

Il faut maintenant trouver un type de graphique adapté à notre besoin. La documentation de Chart.js en propose déjà plusieurs sous le titre “Chart Types”. Parmi les liens proposés, l’option “Bar Chart” est probablement celle qui nous correspond.



Menu du site Chart.js montrant les différents types de graphiques

Cependant, les barres sont verticales, alors que nous les voudrions à l'horizontale. Heureusement, en bas de la page, la documentation nous explique quelle option utiliser pour passer en mode horizontal. C'est parfait ! Nous avons trouvé le graphique adapté, et nous avons des exemples pour nous aider.

Avant d'afficher le graphique à l'écran, nous avons besoin de calculer les données qui seront affichées. La réponse de l'API HTTP à la requête GET /avis contient tous les avis, avec le nombre d'étoiles attribuées par chaque commentaire. Nous utilisons donc les données des avis pour calculer une liste de nombres correspondant à la quantité de commentaires avec une étoile, avec deux étoiles, et ainsi de suite.

On rajoute ce code de calcul à la fin du fichier avis.js dans une fonction asynchrone afficherGraphiqueAvis() :

```
export async function afficherGraphiqueAvis(){

// Votre code ici

}

// Calcul du nombre total de commentaires par quantité
d'étoiles attribuées

const avis = await fetch("http://localhost:8081
/avis").then(avis => avis.json());

const nb_commentaires = [0, 0, 0, 0, 0];

for (let commentaire of avis) {

  nb_commentaires[commentaire.nbEtoiles - 1]++;

}
```

Après la boucle, on se retrouve avec la constante nb_commentaires qui contient une liste de nombres, [1, 1, 3, 6, 9]. C'est-à-dire qu'il y a neuf commentaires qui ont attribué 5 étoiles, six commentaires qui ont attribué 4 étoiles, et ainsi de suite.

Nous avons calculé les données à afficher, intéressons-nous

maintenant à l'utilisation de la librairie à proprement parler. 😉

Configurez la librairie pour afficher le graphique

Dans un premier temps, on configure une liste qui servira de légende aux barres horizontales. Le premier élément de la liste sera la barre la plus haute sur le graphique.

```
// Légende qui s'affichera sur la gauche à côté de la barre horizontale
const labels = ["5", "4", "3", "2", "1"];
```

Dans un deuxième temps, on rassemble toutes les données visibles sur le graphique dans un objet data. On y retrouve deux propriétés : **labels** et **datasets**. La première a comme valeur la liste labels que l'on a définie juste avant, et la deuxième est une liste d'objets.

```
// Données et personnalisation du graphique
const data = {
   labels: labels,
   datasets: [{
    label: "Étoiles attribuées",
    data: nb_commentaires.reverse(),
     backgroundColor: "rgba(255, 230, 0, 1)", // couleur jaune
   }],
};
```

Revenons ensemble sur la propriété datasets. Un objet de cette liste peut contenir, à son tour, plusieurs propriétés :

- label : le nom de la série de données. Elle s'affiche en haut du graphique avec la couleur correspondante utilisée pour les barres ;
- data : les données à mettre en forme ;

- backgroundColor : une chaîne de caractères spécifiant le code couleur pour la série de données.

  Pour le moment, la liste nb_commentaires présente le nombre de commentaires par ordre croissant d'étoiles (de 1 à 5 étoiles). Nous devons donc inverser l'ordre de la liste, de manière à afficher les données par ordre décroissant, de 5 à 1 étoiles.

  La couleur choisie correspond à un jaune doré. La valeur de la propriété backgroundColor est une chaîne de caractères qui reprend le format de la fonction CSS `rgba(rouge, vert, bleu, transparence)` .

  Dans un troisième temps, on crée un objet de configuration qui indique le type de graphique, les données et une option pour rendre les barres horizontales.

```
// Objet de configuration final
const config = {
   type: "bar",
   data: data,
   options: {
      indexAxis: "y",
   },
};
```

  Et enfin, il ne nous reste plus qu'à déclencher le rendu du graphique dans l'élément canvas.

```
// Rendu du graphique dans l'élément canvas
const graphiqueAvis = new Chart(
   document.querySelector("#graphique-avis"),
   config,
);
```

Il ne nous reste plus qu'à importer la fonction afficherGraphique avis dans pieces.js et à l'appeler à la fin du fichier.

Et voilà 😅 ! Notre joli graphique s'affiche sur notre page web !

Graphique montrant la quantité de commentaires ayant attribué 5, 4, 3, 2 et 1 étoiles