Génération de diagramme

On veut à partir d'un tableau d'une page HTML créer et insérer un diagramme affichant les courbes correspondant aux données. Récupérez le fichier `temperatures.html` qui contient une page HTML dont il faut afficher les données.

Tableau des données

On veut récupérer les données du tableau HTML dans un objet contenant les propriétés `description`, intitulé (textuel) du titre, `months` tableau des mois de mesures (tableaux de chaînes correspondant aux intitulés de colonnes du tableau HTML) et `temperatures` tableaux d'objets ayant les propriétés `location`, contenant le nom de la ville, et `temperatures` tableau des températures de la ville .

Créez le module `view` (fichier `view.js`) représentant la vue de l'application. **Ajoutez**-y la fonction `temperaturesFromTable(t)` qui renvoie l'objet des températures écrites dans l'élément `<table>` indiqué par `t` sous le format suivant :

```
<table>
  <caption>Description</caption>
  <thead>
    <tr><td></td>
      <th>Month 1</th><th>Month 2</th>…
    </tr>
  </thead>
  <tbody>
    <tr><th>Location 1</th>
        <td>temp</td><td>temp</td>…
    </tr>
    <tr> … </tr> …
  </tbody>
</table>
```

La ligne de l'en-tête du tableau contient une cellule `<td>` vide puis des cellules en-tête `<th>` contenant les intitulés des colonnes. Les lignes du corps contiennent une cellule en-tête `<th>` contenant l'intitulé de la ligne puis des cellules `<td>` contenant les données de la ligne.

Ajoutez et **exportez** la fonction `temperatures()` qui renvoie les températures du tableau `<table id="tempvilles">`.

Tracé du diagramme

On veut tracer le diagramme des températures du tableau en utilisant Chart.js

Pour cela on récupère le canevas :

```
const ctx = document.getElementById('temp');
```

Ensuite on crée le diagramme avec~ :

```
let myChart = new Chart(ctx, {
 type:"line",
 data:{
 labels:[ liste mois ],
 datasets:[
 {
   label: "Ville 1",
   data:[ températures ],
   borderColor: couleur
 } ,… ]
 },
 options:{
  datasets: { line: {
  lineTension:0, fill: false
 }},
  responsive: false
 }
});
```

Ajoutez dans le module `view` la donnée `colors` qui représente un tableau (initialement vide) de couleurs au format `"hsl(`*H*`,`*S*`%,`*L*`%)"` où *H* est compris entre 0 et 360, et *S* et *L* const compris entre 0 et 100. **Ajoutez** la fonction `randomColors(n)` qui renvoie un tableau de `n` couleurs au hasard (tirer les composantes *H* *S* et *L* au hasard). **Ajoutez** la fonction `temperaturesDatasets(t, col)` qui à partir des températures `t` et du tableau de couleurs `col` renvoie un tableau d'objets (un par ville) ayant les propriétés `label`, nom de la ville, `data`, tableau des températures et `borderColor` le couleur correspondante. **Ajoutez** la fonction `temperaturesChartFrom(t, ctx, col)` qui crée et renvoie le diagramme des températures `t` associée au contexte `ctx` avec les couleurs `col`. **Ajoutez** la donnée `temperaturesChart` (initialement nulle) représentant le diagramme des températures **Ajoutez** et exportez la fonction `displayTemperaturesChart(t)` qui affiche le diagramme des températures `t` (s'il n'y a pas assez de couleurs, il faudra en retirer au hasard) ; en détruisant l'ancien en appelant sa méthode `destroy()`).

Faites en sorte qu'une fois la page chargée, les températures soient aussi affichée dans le diagramme. **Testez**.

Amélioration de la table

Pour mieux faire le lien entre la table HTML et le diagramme correspondant, on veut que les noms de lignes soient affichés dans la table avec la même couleur que la courbe qui les représente dans le diagramme. **Écrivez** la fonction `colorizeTable` qui prend en paramètre l'identifiant d'un élément table et un tableau de couleurs : il faut changer dans le style en-ligne des cellules en-tête des lignes la couleur d'affichage (propriété `color`) avec celles du tableau de couleurs. **Testez**.