1 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 .

Écrivez le module `temperatures.js` qui exporte la fonction `temperaturesFromTable(t)` qui renvoie l'objet des températures écrites dans l'élément `t` représentant une `<table>` 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.

Écrivez le module `view.js` représentant la vue de l'application qui exporte par défaut la fonction `temperaturesOfPage()` qui renvoie les températures du tableau `<table id="tempvilles">`. **Testez** (affichez les températures sur la console).

Tracé du diagramme

On veut tracer le diagramme des températures du tableau en utilisant Chart.js **Chargez** le script `chart.js` dans la page HTML, la classe `Chart` sera alors dans l'espace global.

On récupère le contexte graphique du canevas :

```
const ctx = document.getElementById('myChart')
                    .getContext('2d');
```

Ensuite on crée le diagramme :

```
let myChart = new Chart(ctx, {
 type:"line",
 data:{
 labels:[ liste mois ],
 datasets:[
   {  //un objet par ville
     label: "Ville",
     data:[ températures ],
     borderColor: couleur
   } ,…
 ]
 },
 options:{
   title: { display: true, text: "description" },
   datasets: { line: { lineTension:0, fill: false }},
   responsive: false
 }
});
```

Ajoutez dans le module `view.js` la donnée `colors` qui représente un tableau (initialement vide) de couleurs données sous la forme de chaînes `"hsl(`*H*`,`*S*`%,`*L*`%)"`. **Ajoutez** la fonction `newColors(n)` qui renvoie un tableau de `n` couleurs (*H*=*i*×360/*n*, *S*=97 et *L*=42).

Écrivez le module `diagram.js` qui contient la fonction `temperaturesDatasetsFrom(t, colors)` qui à partir des températures `t` et du tableau de couleurs `colors` renvoie un tableau d'objets (un par ville) ayant les propriétés `label`, nom de la ville, `data`, tableau des températures et `borderColor` la couleur correspondante. **Ajoutez** et exportez par défaut la fonction `temperaturesChartFrom(t, ctx, colors)` qui crée et renvoie le diagramme des températures `t` associée au contexte `ctx` avec les couleurs `colors`.

Ajoutez dans le module `view.js` 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 au préalable l'ancien diagramme en appelant sa méthode `destroy()`).

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

Amélioration de la table

Pour 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. **Ajoutez** dans le module `view.js` la fonction `colorizeTable(table, colors)` qui prend en paramètre un élément table `table` et un tableau de couleurs `colors` : il faut, pour chaque ligne du corps de la table, changer dans son style en-ligne la couleur d'affichage (propriété `color`) avec celle correspondante du tableau de couleurs. **Faites** en sorte qu'une fois la page chargée les lignes de la table de températures soient colorées, **testez**.

2 Gestion des températures

On veut transformer la page en une application permettant de charger, afficher, modifier et sauver des températures. Créez un nouveau dossier et copiez-y les modules écrits dans la partie précédente. **Récupérez** la page `temperatures2.html` qui contient la page HTML servant de base à l'application. Faites-lui charger `button.css` qui définit des styles pour la barre de menu et la classe `button` pour tout ce qui doit ressembler à un bouton.

Affichage des températures

Cette fois-ci on aura d'abord des températures et il faudra créer le HTML du tableau correspondant et l'insérer dans la page en utilisant `innerHTML`. Pour cela vous pouvez vous aider de la méthode `map` des tableaux : `t.map(f)` renvoie le tableau `[f(t[0]), f(t[1]), … ]` contenant les valeurs de `t` transformées par `f`. Vous pouvez aussi vous aider de la méthode `join` qui concatène les éléments du tableau en une chaîne et utiliser les chaînes interpolées.

Créez le module `temperatureshtml.js` contenant la fonction `temperaturesHTMLTR(temp)` qui renvoie la chaîne contenant le HTML de la ligne `<tr>` des températures `temp` d'une ville et la fonction `temperaturesHTMLTable(t)` exportée par défaut qui renvoie une chaîne contenant le HTML du tableau des températures de `t`. **Ajoutez** et **exportez** dans le module `view.js` la fonction `displayTemperaturesTable(t)` qui affiche le tableau des températures de `t` dans l'élément `<div id="temperature-table">`.

Lecture des températures

Écrivez le module `controller.s` qui gérera l'événementiel de la page ainsi que les données. **Ajoutez** les fonctions `load` et `unLoad` qui doivent être appelées au chargement et déchargement de la page (et faites qu'elles le soient). Les écouteurs concernant le menu devront être attachés à l'élément `<nav id="menu">` et il ne devra en avoir qu'un par type d'événement.

Ajoutez la variable `temper` représentant des températures (initialement nulle). On veut charger les températures depuis un fichier où elles y sont au format JSON. On lit le fichier des températures grâce à l'élément `<input type="file" id="temp-file" accept="application/json">` et il faut réagir à chaque fois qu'on charge un fichier. Même en ajoutant la classe `button`, c'est moche. On l'englobe dans un label, `<label class="button">Charger<input type="file" id="temp-file"></label>` et on le cache pour n'afficher que le `<label>` (dé-commentez le style pour l'input dans `button.css`).

Ajoutez dans le module `controller.js` le gestionnaire d'événements `onFileChange` : faites qu'il soit appelé quand l'utilisateur sélectionne un fichier et qu'il charge les températures qui sont au format JSON dans le fichier et affiche le tableau de ces températures. **Testez**.

Sauvegarde des températures

On veut pouvoir sauver les températures dans un fichier au format JSON. **Faites** que `Sauver` soit affiché comme un bouton et **ajoutez** dans le module `controller.js` la fonction `saveTemperatures` qui fait que les températures sont sauvées dans le fichier `temperatures.json` et le gestionnaire d'événements `onClick` qui doit être appelé quand on clique dans `<nav>` et qui quand le clic provient du bouton `Sauver` sauve les températures. **Testez**.

Affichage du diagramme dans un dialogue modal

On veut afficher le diagramme dans un dialogue modal, c'est-à-dire une page qui apparaît au-dessus. Un dialogue modal est constitué d'un `<div class="modal">` contenant un `<div class="contenu-modal">` contenant le contenu du dialogue qui figurent toujours dans la page. **Chargez** `dialog.css` qui définit le style pour le contenu modal. Pour l'instant le dialogue est affiché. **Faites** en sorte que lorsqu'un fichier de températures est lu, le diagramme des températures soit aussi affiché dans son dialogue, **testez**. **Modifiez** la feuille de style `dialog.css` pour que les éléments de classe `modal` soient positionnés de façon fixe en (0,0), prennent toute la page, soient au-dessus du contenu de la page, aient `rgba(0,0,0,0.4)`

pour couleur de fond et ne soient pas affichés, et pour que ces éléments soient affichés (en tant que bloc) quand on navigue dessus. **Testez**. **Modifiez** `Diagramme` dans le menu pour qu'il navigue sur le dialogue du diagramme. **Modifiez** la croix de fermeture du dialogue pour naviguer sur l'application principale. **Testez**.

Améliorations

Améliorez en faisant en sorte qu'on ne puisse sauver et aller sur le diagramme que si des températures sont chargées. Pour cela, **ajoutez** et **exportez** dans le module `view.js` la fonction `disableTemperatureManipulation` qui désactive le bouton `Sauver` et ajoute l'attribut `aria-disabled="true"` à ce bouton et à l'ancre `Diagramme`, et la fonction `enableTemperatureManipulation` qui active le bouton `Sauver` et enlève l'attribut `aria-disabled` de ce bouton et de l'ancre `Diagramme` ; faites en sorte que `disableTemperatureManipulation` soit appelée au chargement de la page et que `enableTemperatureManipulation` soit appelée quand un fichier de température est lu ; enfin faites en sorte que style `opacity: 0.6; pointer-events: none;` soit appliqué aux éléments ayant l'attribut `aria-disabled="true"`. **Testez**.