Affichage de températures

On veut faire une application qui permet de lire des températures dans un fichier et de les afficher dans un tableau et dans un diagramme. Récupérez le fichier `temperatures.html` qui contient une page HTML dont il faut afficher les données.

Comme on utilise les modules ES6, il faut utiliser un serveur web. Vous utiliserez le serveur écrit dans `fileserver.js` : vous lancerez le serveur dans une console node.js avec `node fileserver.js`, serveur qui tourne sur `localhost:8080`.

Températures

On mesure des températures pendant plusieurs mois dans plusieurs villes. Les températures sont stockées dans un objet contenant les propriétés `description`, description générale des températures mesurées, `mois` tableau des mois des mesures de températures (tableaux de chaînes) et `villes` tableaux d'objets ayant les propriétés `ville`, contenant le nom de la ville, et `temperatures` tableau des températures de la ville pour chaque mois.

Créez le module `exempletemperatures.js` qui exporte par défaut des `Exemples de températures`, températures mesurées en `Janvier`, … , `Décembre` pour les villes de `Brest` (températures valant 6,1, 5,8, 7,8, 9,2, 11,6, 14,4, 15,6, 16, 14,7, 12, 9 et 7 degrés) et de `Strasbourg` (températures valant 0,4, 1,5, 5,6, 9,8, 14, 17,2, 19, 18,3, 15,1, 9,5, 4,9 et 1.3 degrés).

Affichage des températures dans un tableau

On veut afficher les températures dans un tableau sous la forme suivante :

```
<table>
  <caption>Description</caption>
  <thead>
    <tr><td></td>
      <th>Mois 1</th><th>Mois 2</th>…
    </tr>
  </thead>
  <tbody>
    <tr><th>Ville 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 mois. Le corps contient les lignes correspondant aux températures des villes : une ligne contient une cellule en-tête `<th>` contenant la ville et les cellules `<td>` contenant les températures de la ville.

On veut générer la chaîne contenant le HTML du tableau. **En utilisant** les chaînes interpolées, le fait qu'elle peuvent être imbriquées (`${}` peut contenir une chaîne interpolée), en utilisant `Array.from(t,f)` qui renvoie le tableau `[f(t[0]), f(t[1]), … ]`, en utilisant les fonctions flèche et en utilisant `join()`, **écrivez** le module `temperatures.js` qui contient les fonctions `ligneMoisHTML(t)` qui renvoie la ligne d'en-tête du tableau contenant les mois correspondants aux températures `t`, `ligneVilleHTML(tville)` qui renvoie la ligne correspondant aux températures `tville` d'une ville et `temperaturesFromTable(t)` qui renvoie le HTML du tableau des températures de `t` et qui exporte la fonction `afficherTemperatures(t, id)` qui remplace le contenu de l'élément d'identifiant `id` par le tableau HTML des températures de `t`.

On veut tester. **Écrivez** le module `temperatures.js` qui contient une fonction `init()` qui affiche le tableau des températures de `exempletemperatures.js` dans l'élément d'identifiant `tableau-temperatures` et qui appelle `init()` au chargement de la page. **Testez** en faisant exécuter ce module dans la page `temperatures.html`.

Chargement des températures

On veut charger les températures depuis un fichier où elles y sont au format JSON. Pour cela on sélectionne le fichier avec l'élément `<input type="file" id="fichier-temperatures" accept="application/json">` et on clique sur le bouton `<button id="charger">Charger</button>` pour lire les températures et les afficher.

Ajoutez la variable `temperatures` représentant des températures (initialement vides : description, mois et villes vides). **Remplacez** le contenu de la fonction `init()` pour qu'elle désactive le bouton `Charger` et le bouton `Diagramme`.

Il faut réagir à chaque fois que l'on change le fichier sélectionné. **Ajoutez** le gestionnaire d'événements `onFichierChange` : il active le bouton `Charger`. **Faites en sorte** qu'il soit appelé quand l'utilisateur sélectionne un fichier (cliquer sur le `<label>` qui contient le `<input>` déclenche aussi le `<input>`, il faut attacher le gestionnaire au `<nav>` qui le contient).

Il faut réagir quand l'utilisateur demande à charger le fichier sélectionné.

Ajoutez le gestionnaire d'événements `onCharger` : il lit les températures du fichier sélectionné, affiche le tableau de ces températures et active le bouton `Diagramme` (vous pouvez vous aidez de la fonction de rappel `onLireFichier` pour lire le fichier sélectionné). **Faites en sorte** qu'il soit appelé quand quand on clique sur le bouton `Charger`. **Testez**.

L'interface est un peu moche, on peut l'améliorer avec CSS. **Complétez** le fichier `button.css` pour indiquer les sélecteurs manquants aux endroits indiqués et faites charger ce fichier par la page. **Testez**.

Tracé du diagramme

On veut tracer le diagramme des températures du tableau en utilisant Chart.js Le module `diagrammetemperatures.js` importe ce qu'il faut pour tracer des diagrammes avec Chart.js

Pour créer le diagramme des températures dans un élément `<canvas>` canevas, il faut créer un diagramme :

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

Créez le module `couleurs.js` qui exporte la fonction `nouvellesCouleurs(n)` qui renvoie un tableau de `n` couleurs, c'est-à-dire de chaînes `"hsl(`*H*`,`*S*`%,`*L*`%)"` où $H=i\times360/n$, $S=97$ et $L=42$.

Écrivez le module `diagrammetemperatures.js` qui contient la variable `diagrammeCourant` contenant le diagramme (initialement nul). **Ajoutez** la fonction `datasetsTemperatures(t, couleurs)` qui à partir des températures `t` et du tableau de couleurs `couleurs` 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** la fonction `diagrammeTemperatures(t, couleurs, canevas)` qui crée et renvoie le diagramme des températures `t` avec les couleurs `couleurs` dans le canevas `canvevas` en fansant attention à détruire l'ancien diagramme (méthode `destroy()`) s'il y en a un. **Ajoutez** et exportez la fonction `afficherDiagrammeTemperatures(t, couleurs, id)` qui trace le diagramme des températures dans le canevas d'identifiant `id`.

Ajoutez dans le module `temperatures.js` la donnée `couleurs` qui contient le tableau (initialement vide) des couleurs à utiliser. **Modifiez** la fonction `onLireFichier` pour au'elle génère des couleurs correspondant au températures qui viennent d'être chargées. **Ajoutez** le gestionnaire d'événement `onDiagramme` qui affiche le diagramme des températures dans le canevas de la page et **faites** qu'il soit appelé quand l'utilisateur clique le bouton `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** et exportez dans le module `tableautemperatures.js` la fonction `colorerTable(id, couleurs)` qui pour chaque ligne du corps de la table d'identifiant `id`, change dans son style en-ligne la couleur d'affichage (propriété `color`) avec celle correspondante du tableau de couleurs `couleurs`. **Faites** en sorte que les lignes de la table de températures soient colorées après l'affichage de la table. **Testez**.