programmation web en js

TD n°4 : mini-projet fonctions-tableaux-dom

Introduction

L'objectif de ce mini-projet est de construire une application javascript vous permettant d'utiliser et d'appliquer les notions abordées jusqu'ici : bases du langage, fonctions, objets et tableaux, DOM et événements.
Le projet peut être réalisé en binôme. Il se compose de 4 étapes.

Description globale

L'objectif est de construire une application javascript permettant de prendre des notes et de mémoriser ces notes.
Une note est composée d'un titre, d'un contenu et d'une date de création. L'application permet de créer des notes, de modifier des notes existantes et de supprimer des notes.
La liste des notes déjà créées est affichée ; elle permet de sélectionner une note à afficher.
Un exemple d'affichage est fourni en annexe. Une vidéo de démonstration des fonctionnalités est également disponible.

Le contenu d'une note peut être formaté en utilisant une syntaxe Markdown. Pour être affiché, la note est convertie en un fragment de code html à l'aide de la librairie showdown.js (http://showdownjs.com/) qui vous est fournie.

Pour que la liste de notes soit persistante, c'est à dire qu'elle soit conservée d'une exécution à l'autre, elle sera stockée dans le localStorage du navigateur. Il s'agit d'un espace de stockage géré par le navigateur et associé à une page web quelconque. plus d'information disponible ici : https://developer.mozilla.org/fr/docs/Web/API/Window/localStorage

Etape 1 : Créer une note et l'afficher

A partir du squelette html-css fourni, l'objectif de l'étape 1 est :

- lorsque l'utilisateur clique sur l'option "+" du menu, afficher un formulaire de saisie d'une note,
- lorsque l'utilisateur valide, la note est créée et affichée en dessous du formulaire.

Dans cette étape, la note courante (celle qui est affichée) est stockée dans une variable partagée.
A réaliser :

- créer *la classe* `Note()` dont le constructeur reçoit en paramètre le titre et le contenu de la note, et initialise la note. Une note possède en plus une propriété `date_creation` initialisée à la date courante par le constructeur. Prévoir les méthodes `setTitre`() et `setContenu`() pour modifier une note.

- créer *la classe* `NoteView` chargée de gérer l'affichage d'une note. Le constructeur reçoit une Note. La classe propose les méthodes suivantes :

  - une méthode pour convertir une note en texte html, en utilisant la librairie showdown :

```
let conv = new showdown.Converter();
let htmlText = conv.makeHtml(markdownText);
```

  - une méthode pour afficher (en insérant dans le dom) la note courante convertie en html. Pour l'insertion dans le DOM, on utilisera Element.innerHTML auquel on affectera le texte de la note converti en HTML.

- créer *l'objet* `noteFormView` qui est chargé de gérer le formulaire de saisie d'une note. Il comporte les méthodes :
  - `display`() : rend visible le formulaire en modifiant la liste de classes css,
  - `hide`() : rend le formulaire caché en modifiant la liste de classes css,
  - `validate`() : utilisé comme handler de validation du formulaire : récupère les données saisies, créée une note et la stocke dans la variable partagée, puis l'affiche en créant un objet `noteView.`

- créer l'objet mainMenuView chargé de gérer le menu général. Il comprend les méthodes :
  - `addHandler`() : utilisée pour traiter le click sur l'option "+" en affichant le formulaire de saisie,
  - `init`() : initialise le menu en ajoutant les listeners sur les événements attendus ("click" sur "+")
- créer un objet de gestion de l'état global de l'application, qui comprend :
  - la variable stockant la note courante,
  - une méthode d'initialisation générale et de démarrage de l'application, utilisée comme handler de l'événement load sur window. A cette étape, elle se contente d'initialiser le menu.

Etape 2 : Gérer une liste de notes

L'objectif de l'étape est de gérer une liste de notes. Lorsqu'une note est créée, elle est ajoutée à la liste. La liste de note s'affiche en colonne dans l'application. Lorsqu'un item est sélectionné (click) dans cette colonne, la note correspondante est affichée.
Dans cette étape, la note courante est repérée par son indice dans la liste. Cette indice de la note courante est stocké dans une variable partagée.

- créer la classe `NoteList` chargé de gérer la liste de notes ; elle comprend :
  - une propriété tableau pour stocker la liste, initialisée à [ ]
  - une méthode `addNote(note)` qui reçoit une note et l'ajoute dans la liste ; elle retourne l'indice de la nouvelle note,
  - une méthode `getNoteById(i)` qui retourne la note d'indice *i,*
  - une méthode `getList()` qui retourne la liste de notes.
- transformer l'objet de gestion de l'état global de l'application : il doit stocker une liste de notes et l'index de la note courante puis compléter la création d'une note (validation du formulaire) pour qu'elle soit ajoutée dans la liste et affichée.

- créer l'objet `noteListMenuView` chargé de gérer l'affichage de la liste de notes dans la colonne (section `.note_list` dans le document html). L'objet comporte la méthode
  - `displayItem(note)` : reçoit une note et ajoute un item dans la liste affichée. Le nouvel item est ajouté en fin de liste. l'affichage d'un item correspond au titre de la note + date de création.

- faire en sorte que l'on puisse sélectionner une note dans la colonne pour l'afficher. Il faut pour cela traiter l'événement "click" sur les items de la liste ; le handler de cet événement :
  - calcule l'indice de la note sélectionnée, qui devient donc l'indice de la note courante,
  - change l'apparence de l'item sélectionné en utilisant la classe css `.note_list_item-selected`
  - affiche la nouvelle note courante en créant un objet `noteView`.
  - indication : utiliser le bubbling des événements pour déclarer un seul listener sur la

section "`noteListMenu`", lors de l'initialisation de l'application.

Etape 3 : une liste de notes persistante

L'objectif de l'étape est de rendre la liste de notes persistante, c'est-à-dire conservée d'une exécution à l'autre. Au démarrage de l'application, si des notes ont été créées, elles sont chargées et la liste est initialisée.
Pour cela on utilise le localStorage, qui est un mécanisme de stockage géré par le navigateur et associé à une page web, basé sur des enregistrements clé-valeur. l'api localStorage propose 4 méthodes :

```
localStorage.setItem( 'key', 'value' ) ;
localStorage.getItem( 'key' ) ;
localStorage.removeItem( 'key') ;
localStorage.clear() ;
```

Les clés et les valeurs sont des chaines de caractères. Pour stocker des valeurs quelconque (par exemple un tableau de Note), il faut donc les sérialiser pour les stocker, et les désérialiser pour les extraire. En javascript, l'habitude est de les sérialiser en json :

```
let jsonString = JSON.stringify( value ) ;
let value = JSON.parse(  jsonString ) ;
```

La gestion de la sauvegarde de la liste de notes dans lelocalStorage est encapsulée dans la classe NoteList, de façon à ne pas impacter et modifier le code de l'application existante.

- ajouter dans la classe `NoteList` la méthode `save()` qui enregistre la liste dans le `localStorage`. On utilise une clé unique dont la valeur sera le tableau de notes sérialisé.
- utiliser cette méthode dans la méthode d'ajout d'une note : chaque fois que la liste est modifiée, elle est sauvegardée.
- ajouter dans la classe `NoteList` la méthode `load()` qui charge la liste de notes stockées dans le localStorage dans le tableau de notes.
- On utilisera cette méthode à l'initialisation de l'application.

Il faut maintenant initialiser la liste affichée en colonne au démarrage de l'application, lorsqu'une liste de notes non vide est trouvée dans le localStorage.
Pour cela, compléter l'objet `noteListMenuView` avec une méthode qui reçoit une liste de note et créé la liste d'items en utilisant la méthode `displayItem()`.

Etape 4 : supprimer des notes dans la liste, modifier des notes existantes

L'objectif de cette dernière étape est de terminer l'application en ajoutant une fonctionnalité de suppression de notes dans la liste.
Cette fonctionnalité sont accessibles grâce à l'option "Del" du menu. Elle s'applique toujours à la note courante repérée par son indice.

Réaliser d'abord la fonctionnalité de suppression de la note courante. Il faut pour cela :

- ajouter une méthode de supression dans la liste de note (classe `NoteList`),
- ajouter une méthode de suppression dans la colonne,
- ajouter le handler sur l'option "del" du menu.

Réaliser ensuite la fonctionnalité de modification de la note courante. Pour cela il faut :

- ajouter une méthode pour modifier un élément de la liste de note (classe `NoteList`). La méthode reçoit l'indice de la note modifiée et la nouvelle valeur. Penser à sauvegarder la liste dans le localStorage.
- ajouter une méthode d'affichage du formulaire pour l'édition. Le formulaire est initialisé avec les valeurs (titre et contenu) de la note qui doit être modifiée. Le handler de validation ne crée pas de nouvelle note mais modifie la note dans la liste.
- ajouter la fonctionnalité dans l'application en réalisant le handler sur le "click" sur l'option "edit" du menu.

Annexe