Design Pattern
TP n° 4 : Promesses, listes

Dans ce TP, vous allez voir comment les scripts permettent d'interagir avec le DOM d'une page HTML.

Promesses

1. Définissez une fonction `delay(time)` qui renvoie une promesse qui se résout au bout de `time` ms.
2. Utilisez cette fonction pour écrire un script qui attend une seconde et demande à l'utilisateur d'entrer une phrase. Cette phrase est ensuite affichée au bout d'une seconde. Si l'utilisateur a annulé la saisie, une erreur doit être lancée immédiatement.
3. Ajoutez une gestion de l'erreur pour afficher un message et quitter proprement.

De JavaScript à HTML

Un exemple du résultat attendu est donné sur la page du cours.
On se donne la classe suivante pour représenter des arbres.

```
class Tree {
        // name is a string, children are instances of Tree
        constructor(name, ...children) {
          this._name = name;
          this._children = children;
        }

        get name() {  return this._name;  }
        get children() { return Array.from(this._children); } // ensure immutability
        isLeaf() { return this._children.length == 0; }
      }
```

1. Définissez une fonction `treeToUl(tree)` qui renvoie une représentation de l'arbre `tree` sous la forme d'un élément du DOM. Cet élément est une liste (tag `ul`). L'étiquette d'un nœud est stocké dans un élément `li`. S'il a des enfants, ils apparaissent dans une sous-liste à la suite.

   ```
   let tree = new Tree(
     "a",
     new Tree("b", new Tree("c")),
     new Tree("d"));
   ```

   va produire un élément qui représente l'arbre HTML

   ```
   <ul>
     <li>a</li>
     <ul>
       <li>b</li>
       <ul><li>c</li></ul>
       <li>d</li>
     </ul>
   </ul>
   ```

2. Définissez une fonction `setCollapsable(ul)` qui prend en argument une liste représentant un arbre. Elle rend la liste déroulante : chaque nœud interne peut afficher/cacher ses enfants via un clic sur son entête. On ne doit pas pouvoir interagir avec les feuilles de l'arbre. L'usage de style CSS et de classes est fortement recommandé. Le sélecteur `+` peut s'avérer utile.

3. Définissez une fonction `setColorable(ul)` qui prend en argument une liste représentant un arbre. Si on clique sur une entrée en appuyant sur shift, l'entrée doit se colorer en rouge. Elle retourne dans son état initial après la même manipulation.
4. Est-il possible de faire cohabiter les fonctions `setCollapsable` et `setColorable` sur une même liste ? En particulier, shift+clic doit colorer une entrée sans la plier/déplier.