Année 2021-2022

RAPPORT DE TP – ENSIBS Cybersécurité du Logiciel – 3e année

TP WEB « Todo list »

TP réalisé par

Lucas CHAPRON

TP encadré par

Mazen EL-SAYED

Table des matières

Table des figures

I. INTRODUCTION

Dans le cadre de ma première année du cycle ingénieur en Cybersécurité du Logiciel à l'ENSIBS, il nous est proposé un TP nous permettant de mettre en pratique nos connaissances et nos compétences pour le développement d'une première page web à l'aide d'HTML5, CSS3 et JS.

II. DÉVELOPPEMENT

1. CONTEXTE

Le but de ce TP est de réaliser une page WEB Todo list qui utilise l'objet localStorage pour stocker les tâches effectuées ou à faire.

Mon site Web comporte :

- Un menu de navigation (accueil, contact, aide) un pied de page.
- Une interface qui affiche la liste des tâches avec la possibilité d'ajouter une nouvelle tâche, de supprimer une tâche ou de marquer une tâche comme achevée.
- Je peux afficher toutes les tâches, les tâches achevées ou les tâches non achevées.

Dans ce TP, je vais :

- Utiliser l'objet localStorage
- Ajouter des nouveaux éléments dynamiquement à la page en utilisant javascript
- Appliquer dynamiquement des styles CSS aux éléments de la page

Autres restrictions :

- Les navigateurs les plus connus devront pouvoir afficher votre site Web.
- Le site devra être « *responsive* » pour un affichage sur ordinateur, mais aussi sur plate-forme mobile.

2. PRÉSENTATION GLOBALE

Ma page WEB possède les fonctionnalités suivantes :

- Ajouter une tâche à faire
- Marquer une tâche comme achevée
- Supprimer une tâche
- Afficher suivant la sélection : toutes les tâches, les tâches achevées ou les tâches non achevées
- Un menu de navigation (accueil, contact, aide) mais qui ne mène à aucune autre page car je ne les ait pas faites.

Elle a pour apparence globale :

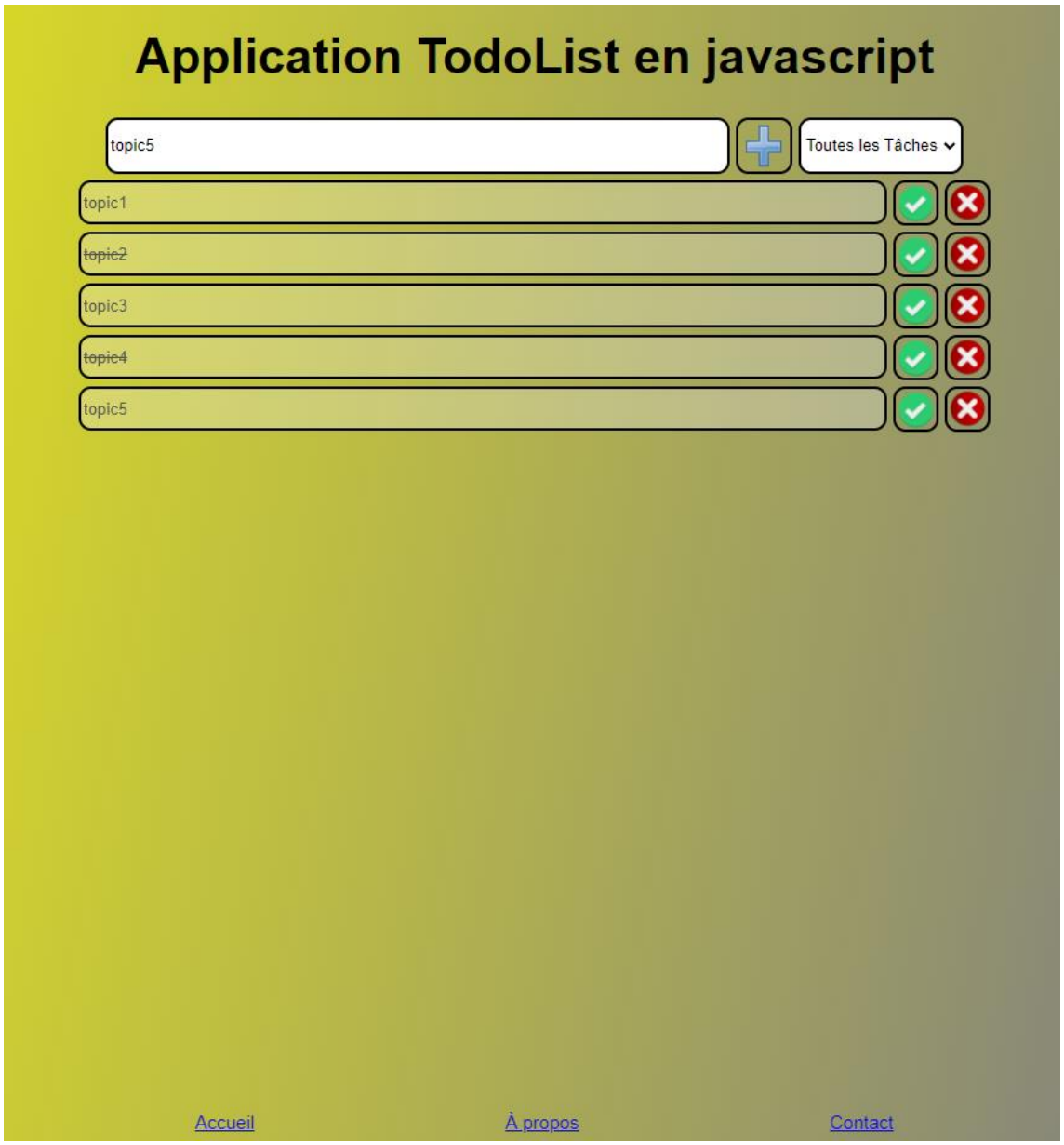

Figure 1 : Apparence globale

3. PRÉSENTATION HTML

Tout d'abord il faut réaliser le code HTML pour la page. Il nous faut un titre, une entrée (intitulé des tâches), un bouton pour ajouter une liste qui a pour valeur l'entrée, un menu déroulant qui permet de choisir l'affichage des tâches, un endroit pour 'afficher' les tâches qu'on a ajouté et aussi 2 boutons un 'check' pour marquer une tâche comme fini et un 'delete' pour retirer une tâche. Les 3 derniers éléments ne sont pas présents dans le code html à la génération de la page car ils sont générés dynamiquement à l'aide de javascript.

Le code HTML se présente tel que :

```html
<!DOCTYPE html>
<html>

    <head>
        <meta charset="UTF-8">
        <title> TodoList </title>
        <link rel="stylesheet" href="CSS/style.css">
    </head>

    <body>
        <h1>Application TodoList en javascript</h1>
        <div class="entree">
            <input type="text" id="entreetopic" placeholder="Tâche">
            <input type="image" id="Add" onclick="newElement()" src="./Image/Add.png">
            <select id="menuderoul" name="menuderoul">
                <option id="all" value="all">Toutes les Tâches</option>
                <option id="todo" value="todo">À faire</option>
                <option id="completed" value="completed">Fait</option>
            </select>
        </div>

        <div id="uljs">
        </div>

        <footer>
            <nav class="menu">
                <ul>
                    <li><a href="#Accueil">Accueil</a></li>
                    <li><a href="#Apropos">À propos</a></li>
                    <li><a href="#Contact">Contact</a></li>
                </ul>
            </nav>
        </footer>

    </body>
    <script src="JS/script.js"></script>
</html>
```

Figure 2 : Code HTML

Explication de l'HTML :

J'appelle la feuille de style (CSS) dans le header car c'est une 'norme' et est stocké dans le dossier CSS et à pour nom *style.css*. Quant au javascript est inséré à la fin du document car lors de l'appel à celui-ci j'ai besoin que l'HTML soit chargé (à cause de la fonctionnalité de localStorage) donc si je le mets au début mes éléments ne sont pas encore crées. Dans mon body, j'ai un titre (h1), tout ce qui concerne l'ajout/affichage (div entree), tout ce qui concerne les tâches que l'utilisateur à rentrée (div uljs) et mon pied de page (footer).

Pour la div entree :

J'ai mon input de valeur (id entreetopic), le bouton pour ajouter un élement (id Add) avec une image stockée dans le dossier Image et à pour nom *Add.png*. Le menu déroulant permettant de sélectionner

les tâches à afficher avec 3 options *all*, *todo* et *completed* qui permettent respectivement d'afficher toutes les tâches, celles restantes et celles finies.

Pour la div uljs :

Celle-ci est gérer dynamiquement comme dis précédemment donc au chargement elle est vide. Lorsqu'une tâche est ajoutée il est créé une nouvelle div contenant la tâche (des li). Il est ajouté la div suivante où seul la value de la li est modifié en fonction de l'entrée de l'utilisateur :

```
<div id="listeli">
    <li><input type="text" id="sortietopic" value="NomTopicEntrée" disabled></li>
    <input type="image" onclick="done(this)" id="buttondone" src="./Image/done.png"></input>
    <input type="image" onclick="remove(this)" id="buttonremove" src="./Image/delete.png"></input>
</div>
```

Figure 3 : HTML div Tâche ajoutée

Pour afficher sur la page les tâches j'ai utilisé une input que j'ai disabled même si je sais que ce n'est pas l'optimum niveau sécurité j'ai tout de même laissé ce dernier car au vu de mon niveau en WEB c'était une solution correcte. Pour le bouton *check* et *delete* j'ai refait comme pour le bouton Add avec des images dans le même dossier.

Pour le footer :

J'ai utilisé une nav (menu) qui permet de faire comme des liens hypertexts. J'ai utilisé des li que je mettrai en ligne avec du CSS.

4. PRÉSENTATION CSS

Je ne vois pas l'utilité de le justifié. Dire que j'utilise text-align : center pour centrer le texte ne me semble pas utile. Certes je pense qu'il y a des trucs en double ou inutile mais j'ai fait ce que je voulais faire avec ma page. Cf dossier *CSS/style.css*.

5. PRÉSENTATION JS

Pour le javascript j'ai mis plein de commentaire à coté des étapes fonctions donc je n'expliquerai pas plus en détail que ce que j'ai marqué.

Fonction newElement() :

```
/**
 * Création d'une tâche lors du clique sur le '+'
 * @returns void
 */
function newElement() {
  let inputValue = document.getElementById('entreetopic').value; //Get la valeur rentrée

  if (inputValue === "") { //Vérification que l'entrée ne soit pas null
    alert("Rentrez une tâche non vide"); //On averti l'utilisateur qu'il faut rentrée au moins une valeur
    return; //Pour sortir de la fonction et ne pas executer la suite
  }

  var content = localStorage.getItem("stock"); //Get les valeurs stockée
  content = JSON.parse(content); //Pour avoir un tableau et non juste un string
  content.push({ desc: inputValue, done: false }); //Ajout au storage le nouvel element
  localStorage.setItem("stock", JSON.stringify(content)); //Stockage

  let syntaxelist1 = '<li><input type="text" id="sortietopic" value="'; //Début de la syntaxe pour la création des li
  let syntaxelist2 = '" disabled></li>'; //Fin de la syntaxe pour la création des li
  let done = '<input type="image" onclick="done(this)" id="buttondone" src="./Image/done.png"></input>'; //Syntaxe pour le bouton done lors de la création des li
  let remove = '<input type="image" onclick="remove(this)" id="buttonremove" src="./Image/delete.png"></input>'; //Syntaxe pour le bouton remove lors de la création des li

  document.getElementById("uljs").innerHTML += '<div id="listeli">' + syntaxelist1 + inputValue + syntaxelist2 + done + remove + '</div>'; //Création d'un li quand il y a un click sur le '+'
}
```

Figure 4 : JS newElement()

Cette fonction permet d'ajouté dans la div *uljs* la div *entreetopic* qui représente une tâche avec ses 2 boutons.

Fonction done(element) :

```
/**
 * Permet de changer l'id d'une tâche lors de l'appuie sur le 'check' pour dire qu'une tâche est fini
 * @param {*} element
 */
function done(element) {
  var content = localStorage.getItem("stock"); //Get les valeurs stockée
  content = JSON.parse(content); //Pour avoir un tableau et non juste un string

  let input = element.parentNode.firstChild.firstChild; //J'accède à la li concerné
  if (input.id == "sortietopic") { //Si le topic n'étais pas encore fini alors
    input.id = "sortietopicdone"; //On change l'id de la li pour l'afficher correctement
  }
  else {
    input.id = "sortietopic"; //On change l'id de la li pour l'afficher correct
  }

  for (let i = 0; i < content.length; i++) { //Parcours de toutes les valeurs stockées pour gérer le stockage des taches accomplies
    if (input.value == content[i].desc) { //Si la valeur de la li concerné est égale au contenu d'une des valeurs stockées
      content[i].done = true; //On change dans le contenu de localStorage la valeur de done
    }
  }
  localStorage.setItem("stock", JSON.stringify(content)); //Stockage
}
```

Figure 5 : JS done(element)

Cette fonction permet de changer l'id d'une tâche lors de l'appuie sur le 'check' pour marquer une tâche comme fini.

Fonction remove(element) :

```
/**
 * Permet de changer l'id d'une tâche lors de l'appuie sur le 'remove' pour dire qu'une tâche n'est plus d'actualité
 * @param {*} element
 */
function remove(element) {
  var content = localStorage.getItem("stock"); //Get les valeurs stockée
  content = JSON.parse(content); //Pour avoir un tableau et non juste un string
  let input = element.parentNode; //On accède à la div

  for (let i = 0; i < content.length; i++) { //Parcours de toutes les valeurs stockées pour gérer le stockage des taches remove
    if (content[i].desc == input.firstChild.firstChild.value) { //Si la value de la li est la meme que la description stockées alors
      content.splice(i, 1); //On supprime du stockage la div contenant cette li
    }
  }
  input.remove(); //On supprime de la page la div

  localStorage.setItem("stock", JSON.stringify(content)); //Stockage
}
```

Figure 6 : JS remove(element)

Cette fonction permet de supprimer une tâche lors de l'appuie sur le 'remove'.

Fonction Load() :

```
/**
 * Permet de gérer localstorage au lancement de la page
 */
function Load() {
  if (localStorage.getItem("stock") == null) { //S'il n'y a pas encore de stock alors on le crée
    localStorage.setItem("stock", JSON.stringify([])); //On crée un stock d'une liste vide
  }

  var content = localStorage.getItem("stock"); //Get les valeurs stockée
  content = JSON.parse(content); //Pour avoir un tableau et non juste un string

  if (content.length == 0) { //S'il n'y a pas de taches de stockées
    alert("Pas de liste à charger"); //On averti l'utilisateur qu'il n'y a pas de liste
    return; //Pour sortir de la fonction
  }

  let syntaxelistnotdone = '<li><input type="text" id="sortietopic" value="'; //Début de la syntaxe pour la création des li qui n'est pas fini
  let syntaxelistdone = '<li><input type="text" id="sortietopicdone" value="'; //Début de la syntaxe pour la création des li qui est fini
  let syntaxelist2 = '" disabled></li>'; //Fin de la syntaxe pour la création des li
  let done = '<input type="image" onclick="done(this)" id="buttondone" src="./Image/done.png"></input>'; //Syntaxe pour le bouton done lors de la création des li
  let remove = '<input type="image" onclick="remove(this)" id="buttonremove" src="./Image/delete.png"></input>'; //Syntaxe pour le bouton remove lors de la création des li

  for (var i = 0; i < content.length; i++) { //Parcours de toutes les taches stockées pour gérer la bonne création
    if (content[i].done === true) { //Si la tache avait été coché comme faite alors
      document.getElementById("uljs").innerHTML += '<div id="listeli">' + syntaxelistdone + content[i].desc + syntaxelist2 + done + remove + '</div>'; //Création d'un li
    }
    else {
      document.getElementById("uljs").innerHTML += '<div id="listeli">' + syntaxelistnotdone + content[i].desc + syntaxelist2 + done + remove + '</div>'; //Création d'un li
    }
  }
}
```

Figure 7 : JS Load()

Cette fonction permet de gérer localStorage au chargement de la page.

Fonction filter(event) :

```
/**
 * Permet de gérer l'affichage des tâches lors d'une selection à l'aide du menu
 * @param {*} event
 */
function filterTodo(event) //Filtrer les todos
{
  const taches = listeli; //On récupere les div avec les li
  for (item of taches) { //On parcours les div
    switch (event.target.value) //Vérifier le filtre
    {
      case "all": //Tous les todos
        item.style.display = "flex"; //Afficher le todo
        break;
      case "completed": //Tous les todos terminés
        if (item.firstChild.firstChild.id == "sortietopicdone") { //Vérifier si le todo est terminé
          item.style.display = "flex"; //Afficher le todo
        } else { //Sinon
          item.style.display = "none"; //Cacher le todo
        }
        break;
      case "todo": //Tous les todos non-terminés
        if (item.firstChild.firstChild.id == "sortietopic") { //Vérifier si le todo n'est pas terminé
          item.style.display = "flex"; //Afficher le todo
        } else { //Sinon
          item.style.display = "none"; //Cacher le todo
        }
        break;
    }
  }
}
```

Figure 8 : JS filterTodo(event)

Cette fonction permet de gérer l'affichage des tâches lors d'une sélection à l'aide du menu.

Instructions éxecutées au lancement :

```
const filterOption = document.querySelector("#menuderoul");
filterOption.addEventListener("click", filterTodo); //Execute filterTodo dès que l'utilisateur change la valeur du menu
Load(); //Pour charger les valeurs précédentes s'il y a
```

Figure 9 : JS Instructions lancement

III. CONCLUSION

Ce projet m'a permis de mieux assimiler plusieurs points liés au JS car les autres parties comme le HTML et le CSS m'étais déjà un peu familié. Si je devais refaire ce TP j'utiliserais un autre moyen d'afficher les tâches que celui utilisé (input disabled), je ne ferai également pas le même déroulement que ce que j'ai fait (HTML, CSS puis JS) je ferai HTML, JS puis CSS. Ce fût un TP assez peu intéressant par rapport aux fonctionnalités mais le fond n'était pas si mal. Cependant pour une personne qui n'a pas trop fait de WEB, la durée du projet est mal évaluée (~12h de passé) et ceux ayant déjà fait plein de WEB se sont ennuyés donc pourquoi ne pas faire 2 TP différents suivant les niveaux et laisser plus de temps qu'une semaine.