Cours développement Web

Niveau Intermédiaire

Activité Numéro 2

Objectifs : Cette activité vise à :

- Renforcer la compréhension des API HTML avancées, des éléments CSS avancés et de la création de graphiques à l'aide de l'élément Canvas.
- Encourager la recherche autonome et l'application pratique de ces concepts.

Ressources :

- Les ressources du module, y compris les notes de cours et les exemples.
- Recherche sur Internet pour des informations complémentaires et des tutoriels.

Type de l'activité : Activité individuelle.

Outils de communication : Forum (pour poser des questions ou partager des ressources avec le tuteur et les autres étudiants).

Résultat attendu : Un fichier compressé qui contient les différents fichiers de code de cette activité.

Durée : 04 heures.

Évaluation : L'évaluation sera effectuée par le tuteur en fonction de la qualité de la réalisation et de la compréhension démontrée.

Consignes : Dans cette activité, vous allez appliquer vos connaissances sur les API HTML avancées, les éléments CSS avancés et la création de graphiques à l'aide de l'élément Canvas pour créer une application web interactive. Choisissez deux API à appliquer : l'API **de Stockage Web** détaillée en annexe de cette activité et une autre API de votre choix parmi la liste ci-dessous :

1. **API de Géolocalisation :** Utilisez l'API de géolocalisation de HTML pour créer une application web qui permet à l'utilisateur de voir sa position actuelle sur une carte. Vous pouvez utiliser des bibliothèques externes si nécessaire, mais assurez-vous de comprendre le fonctionnement de l'API de géolocalisation.

2. **API de Glisser-Déposer :** Créez une fonctionnalité de glisser-déposer (drag-and-drop) dans votre application web. Les utilisateurs devraient pouvoir faire glisser des éléments d'une zone à une autre de la page.
3. **API Web Workers :** Créez un Web Worker pour effectuer une tâche en arrière-plan, par exemple, des calculs complexes ou des mises à jour périodiques.
4. **Canvas Graphics de HTML :** Utilisez l'élément Canvas pour dessiner un graphique ou une animation personnalisée. Vous pouvez choisir le sujet de votre choix, mais assurez-vous de montrer une utilisation avancée des fonctionnalités Canvas.

Vous pouvez appliquer aussi les concepts avancés des éléments CSS, tels que les pseudo-classes et les animations, pour améliorer l'apparence et l'interaction de votre application web.

Compression des Fichiers : Rassemblez tous les fichiers de code et les ressources nécessaires dans un dossier selon l'arborescence proposée dans les énoncés, puis compressez l'ensemble dans un fichier ZIP.

Une fois terminé, soumettez votre fichier ZIP contenant tous les fichiers de code avec un document explicatif à travers l'espace de dépôt réservé.

Le tuteur évaluera votre travail en fonction de la qualité de la mise en œuvre et de votre compréhension des concepts avancés.

Annexe : Activité API Local Storage

Introduction

L'API Local Storage introduit par le HTML 5 permet de stocker des données localement sans avoir à se connecter à une source de données externe. Les données stockées sont assez sécurisées du moment que l'accès aux données est contrôlé. En effet, uniquement les pages HTML qui appartiennent à une même source (nom de domaine et protocole) sont autorisées à partager les données.

L'espace de stockage est de quelques méga octets. Une taille suffisante pour les cas d'utilisation classiques. Il faut noter que le local storage permet de stocker des données sous forme de couples : clé et valeur. La clé et la valeur correspondante sont des chaînes de caractères. La clé est utilisé pour manipuler (accéder, modifier et supprimer) la valeur correspondante.

Exemple :

Clé	Valeur
"theme"	"nuit"

Tous les navigateurs modernes supportent l'API local storage de HTML 5.

Objets de l'API

L'API local storage offre deux objets qui peuvent être manipulés via le langage JavaScript :

- window.localStorage : pour stocker les données sans date d'expiration.

- window.sessionStorage : pour stocker les données pour une session (donc disparition dès la fermeture du navigateur).

Par la suite nous allons donner des exemples avec l'objet window.localStorage mais vous pouvez les utiliser également avec l'objet window.sessionStorage.

Manipulation

Présence de l'API

Tout d'abord, il est toujours prudent de vérifier que votre navigateur (ou Webview) supporte cet API HTML5. Le code suivant permet d'afficher un message d'alerte au cas où vous n'avez pas le support de l'API local Storage :

```
if (typeof(Storage) !== "undefined") {
```

alert ("Votre navigateur support l'API local storage !");

} else {

alert ("Votre navigateur ne support pas l'API local storage !");

}

Insertion de données

Maintenant pour insérer un couple clé/valeur, il y a deux façons équivalentes :

localStorage.setItem("theme", "nuit");

ou

localStorage.theme = "nuit";

Il faut noter que le fait d'insérer un couple clé/valeur avec une clé qui existe déjà, ceci va engendrer le remplacement de l'ancien couple par le nouveau couple. Donc c'est l'équivalent d'une mise-à-jour de la valeur.

Manipulation de données

Pour récupérer une valeur à partir de la clé il faut utiliser la méthode getItem() :

localStorage.getItem("theme")

Pour récupérer une clé à partir de son numéro d'index parmi les clés stockées il faut utiliser la méthode key() :

localStorage.key(index)

Pour connaitre combien de couples existent dans le localStorage il faut utiliser la propriété length :

localStorage(length)

Pour supprimer un couple clé/valeur il faut passer la clé à la méthode removeItem() :

localStorage.removeItem("theme")

Pour supprimer tous les couples clé/valeur il faut utiliser la méthode clear() : localStorage.clear()

Recommandation pour l'activité pratique

Objectif

L'objectif de cette activité est de créer une page Web en HTML 5 qui dispose de deux thèmes : "jour" et "nuit". Ces thèmes sont définis dans une feuille de style CSS 3.

L'utilisateur verra sa page affichée la première fois avec usage du thème "jour". Deux boutons permettent d'appliquer l'un ou l'autre thème selon le choix de l'utilisateur. L'API local storage est utilisée pour permettre à l'utilisateur de retrouver la page avec le dernier thème sélectionné, lors des prochaines visites.

Wireframing

Voici les maquettes de l'interface graphique de l'application (dessinées avec Pencil) :

HTML 5

Tout d'abord nous allons commencer par créer la page index.html dans un répertoire nommé activite_localstorage. Dans l'entête de cette page nous allons spécifier qu'il s'agit d'un encodage utf-8. Également, nous allons lier cette page à une feuille de style nommée index.css sous le répertoire css.

Dans la partie body, nous allons créer le titre de niveau 1, les deux boutons et finalement le texte du paragraphe après les boutons. Finalement, nous allons appeler le code JavaScript qui réside dans le fichier index.js sous le répertoire js. Nous devons également donner un identifiant (attribut id) au moins à la balise <body> (par exemple "maPage") et aux balises <input> (par exemple "b1" et "b2"). En effet, c'est un moyen pratique pour les manipuler par la suite.

CSS 3

Maintenant nous allons créer les thèmes dans le fichier index.css. Le premier thème, nommé jour, va se manifester sous forme d'une classe qui sera appliquée à la balise <body>. Dans cette classe on va juste indiquer que le texte est en noir et que le fond est en gris clair.

Le deuxième thème, nommé nuit, va se manifester sous forme d'une autre classe qui spécifie que le texte est en blanc et que le fond est noir.

JavaScript

Ensuite, il faut créer le code JavaScript dans index.js permettant de changer de thème et d'enregistrer le thème choisi par l'utilisateur avec l'API Local Storage pour les utilisations futures de la page index.html.

Une façon de procéder est de créer une fonction appliquerTheme() qui prend en paramètre le nom d'un thème. Celle-ci insère le thème dans le localStorage (pour les utilisations futures) puis affecte la classe CSS qui correspond au thème à la balise <body> via son identifiant (maPage).

Nous avons besoin d'une fonction AjouterEcouteursEvenements() qui va ajouter un écouteur de l'événement "click" à chaque bouton. Pour le premier, la fonction de callback qui sera appelée en cas de click est appliquerThemeJour() et pour le deuxième c'est la fonctionThemeNuit(). Chacune, appelle à son tour la fonction appliquerTheme() avec comme paramètre "jour" pour la première et "nuit" pour la deuxième.

Finalement, une fonction initialiser() va vérifier si l'utilisateur a fait déjà un choix auparavant, sinon elle va appeler la fonction appliquerTheme() avec comme paramètre "jour". Sinon, nous allons appliquer le dernier thème choisi par l'utilisateur également via un appel à la fonction appliquerTheme(). Finalement, il faut appeler la fonction ajouterEcouteursEvenements().

Puisque la fonction initialiser() est la première qui doit être exécutée, alors nous allons ajouter à la fin du fichier index.js un appel à cette fonction.

Travail Personnel

On vous demande de rendre deux fichiers compressés (format zip) contenant chacun une version différente de l'application :

Version 1 : Ajouter d'autres thèmes à ce projet Web.

Nom du fichier à déposer : Nom_Prenom_LocalStorage_v1.zip

Version 2 : Ajouter un bouton permettant d'insérer un paragraphe qui sera saisie par l'utilisateur dans un champ texte dans la page. Ce nouveau paragraphe saisi va remplacer le paragraphe qui existe déjà (Le premier paragraphe par défaut comme le montre la maquette contient le texte suivant : "Ceci est un exemple d'utilisation de l'API Local Storage de HTML5"). A chaque fois l'utilisateur ouvre la page il doit trouver le dernier thème choisi et le dernier paragraphe ajouté.

Nom du fichier à déposer : Nom_Prenom_LocalStorage_v2.zip