JavaScript / Web Development - TP1

Introduction

Dans ce TP, nous allons développer une page Web dynamique capable d'afficher la météo pour une ville française.

A la fin, l'utilisateur est capable de renseigner une ville par son nom et d'obtenir la météo pour les 5 prochains jours.

Pour cela, nous allons utiliser différents concepts de JavaScript :

- XMLHttpRequest
- API fetch
- Manipulation de chaînes de caractères
- Manipulation d'objets
- Manipulation de tableaux
- Modification du DOM

Le symbole indique que vous devez faire quelque chose.

Prérequis

Outils nécessaires :

- un navigateur récent (Firefox, Chromium)
- un éditeur de texte

Pour ce TP, nous allons utiliser une API web pour récupérer des informations de météo. Une API est un service exposé par un site pour mettre à disposition des données (ici météorologiques).

Dans notre cas, l'élu est https://openweathermap.org/api. Vous trouverez la documentation de l'API sur le site https://openweathermap.org/forecast5. Plusieurs API sont disponibles, nous utiliserons **5 day / 3 hour forecast**.

Pour utiliser cette API, nous devons utiliser une clé. Cette clé permet de faire 60 appels / minutes.

En haut du fichier script.js présent dans le ZIP d'initialisation, vous trouverez deux variables définies contenant des clés. Merci d'utiliser une des deux.

```
const apiKey = 'd367d2b32c6effdcff8407f35c63c1d7';
// const apiKey = '7aa887c381f3021f19462c39c14c98ee';
```

Démarrage

Dans le ZIP d'initialisation, vous trouverez :

- un fichier `index.html` contenant le code **HTML** de la page
- un fichier `script.js` contenant une structure vide de code **JavaScript**

Nous n'utiliserons pas de CSS (style) spécifique mais la librairie **Bootstrap** qui permet d'avoir rapidement un rendu professionnel.

En ouvrant le fichier `index.html`, vous pouvez observer que notre fichier `script.js` est appelé à la fin du fichier. Ceci permet de charger la page sans attendre l'exécution du contenu du script et donc d'avoir un rendu visuel plus rapide.

Découvrir l'API OpenWeatherMap

Le premier objectif de ce TP est de se familiariser avec l'utilisation de l'API et les données reçues.

Essayez de construire une URL avec les paramètres nécessaires pour récupérer les données météorologiques pour Talence. N'oubliez pas de lire la documentation La clé doit être transmise dans le paramètre `appid`
L'enchainement de deux paramètres se fait avec le symbole `&`

Observez la structure des données reçues (au format JSON), nous en aurons besoin plus tard.

Récupérer la ville de l'utilisateur

Dans le code du fichier `index.html`, vous pouvez observer que le champ ville a un **id**. Ce nom permettra d'identifier le champ en JavaScript.

Ajouter un nom de fonction dans l'attribut `onclick` du bouton (en passant en paramètre `event` qui représente le clic) :

```
<button type="submit" class="btn btn-primary col-2"
onclick="searchWeather(event)">Rechercher</button>
```

A vous maintenant de compléter le contenu de la fonction `searchWeather` dans le fichier `script.js` :

- utiliser preventDefault pour ne pas rafraichir la page
- utilisez document pour récupérer la ville saisie par l'utilisateur
- essayer de vous familiariser avec les outils de Développement de Chrome ou de Firefox pour arrêter l'exécution du code sur une ligne ou pour afficher la valeur d'une variable.

Appeler l'API d'openweathermap - XMLHttpRequest

Dans cette étape, vous allez devoir :

- construire l'url de l'API à appeler à partir de l'url de base et de la ville saisie par l'utilisateur
- utiliser XMLHttpRequest pour déclencher l'appel : https://flaviocopes.com/xhr/#an-example-xhr-request
- afficher le résultat dans la console (`console.log()`)

Ajoutez toujours dans la fonction `searchWeather` votre code.

La réponse reçue depuis openweathermap est une longue chaîne de caractère contenant du JSON. Pour pouvoir exploiter plus facilement ces données, nous allons les convertir en un objet JavaScript :

```
let data = JSON.parse(xhr.responseText);
```

Appeler l'API d'openweathermap - fetch

Maintenant, voyons comment serait cet appel avec la norme ES6 qui introduit la fonction `fetch`
https://flaviocopes.com/xhr/#comparison-with-fetch

Commentez la première partie de code utilisant XMLHttpRequest et ajoutez les lignes nécessaires pour appeler la même url mais avec `fetch` .

Toujours pour convertir les données en un objet :

```
.then(data => data.json())
```

Traitement des données

Nous voulons maintenant traiter les données reçues pour en extraire uniquement le contenu intéressant avant de l'afficher.

Remplacez le `console.log()` qui permettait d'afficher les données par l'appel à la fonction `handleData` en passant en paramètre les données sous forme d'objet.

Il faut maintenant traiter les données dans la fonction `handleData` avec pour objectif :

- stocker le nom de la ville dans une variable `cityName`
- parcourir la liste des données reçues
- récupérer les informations suivantes
  - date (qui nous permettra de déduire le jour)
  - température
  - l'icône représentant la météo. Utilisez la variable `iconUrl` pour calculer l'url de l'icône à afficher.
  - le pourcentage de nuages
- stocker le tout sous forme d'objets dans un tableau `weatherToShow`

Voici un exemple de résultat attendu :

```
console.log(weatherToShow);
{date: 1541451600, icon: "http://openweathermap.org/img/w/04n.png", temp: 11, clouds:
76}
{date: 1541538000, icon: "http://openweathermap.org/img/w/10n.png", temp: 10, clouds:
56}
{date: 1541624400, icon: "http://openweathermap.org/img/w/02n.png", temp: 11, clouds:
8}
{date: 1541710800, icon: "http://openweathermap.org/img/w/10n.png", temp: 11, clouds:
100}
{date: 1541797200, icon: "http://openweathermap.org/img/w/10n.png", temp: 15, clouds:
44}
```

La réponse de l'API est une liste de 40 données météorologiques (toutes les 3 heures pour 5 jours). Pour simplifier le TP, nous allons conserver uniquement une donnée par jour.

Pour cela, utilisez l'opérateur **modulo** % sur l'itérateur dans votre boucle. Exemple :

```
if (i % 8 === 0) {
    // je traite et stocke la donnée
} else {
    // je ne fais rien
}
```

Gestion de la date

La date est dans un format Unix Timestamp et difficilement lisible par un humain. Nous voudrions convertir cette date en jour (Lundi, Mardi, ...).

Complétez la fonction `computeDayFromDatetime` en utilisant la fonction getDay et la variable `days`.

Appelez cette fonction dans la création de l'objet dans `handleData`.

Afficher les données

Vous devriez avoir deux variables `cityName` et `weatherToShow` contenant 5 objets avec des données météorologiques. Nous voulons maintenant les afficher dans la page. Appelez la fonction `render` et la suite du code sera à écrire dans cette fonction.

Utilisez `document.getElementById()` et `innerHTML` pour modifier la valeur de la balise `<span id="city-title"></span>`

Pour les données météo, il faut manipuler le DOM (Document Object Model) qui représente la structure de la page. Nous allons utiliser deux fonctions :

createElement()

createElement permet de créer un élément HTML. Par exemple :

`document.createElement('p')` crée un paragraphe `<p></p>`

appendChild()

appendChild permet d'ajouter un élement à l'intérieur d'un autre. Par exemple, si nous partons de cet élément :

```
<div id="render">
</div>
```

Le code suivant

```
let parentDiv = document.getElementById('render');
let paragraph = document.createElement('p');
paragraph.innerHTML = 'Bonjour';
parentDiv.appendChild(paragraph);
```

aura pour effet :

```
<div id="render">
    <p>Bonjour</p>
</div>
```

Utilisez les données reçues en paramètre (boucle à faire) pour créer 5 blocs identiques à celui-ci :

```
<div class="col-sm">
    <h2>Mardi</h2>
    <img src="http://openweathermap.org/img/w/04n.png">
    <p>13°C</p>
    <p>76% (nuages)</p>
</div>
```

et ajoutez ces blocs à la balise `<div id="render"></div>`

Petite astuce pour un meilleur affichage, ajoutez la classe `col-sm` à chaque bloc.

Résultat

Vous devriez maintenant avoir une page web où, lorsque l'utilisateur saisie une ville et clique sur Recherche, une partie de la page se rafraichit en affichant la météo pour les 5 jours à venir. Bravo

Conclusion

Dans ce TP, nous avons vu :

- comment JavaScript peut utiliser le protocole HTTP pour récupérer des informations
- comment nous pouvons manipuler des données (transformation, création d'objets, parcours de listes)
- comment interagir avec le rendu HTML et donc le rendu visuel

J'ai fini plus tôt ou je veux en faire plus

Nous avons choisi de ne garder qu'une donnée météo par jour en ne gardant qu'une donnée sur 8. Ce n'est pas très représentatif de la météo de la journée.

Pourquoi ne pas stocker toutes les informations sur une journée et calculer une sorte de moyenne ?

A vous de jouer !!!