|  |  |
| --- | --- |
| SLAM5 | TP JSON AJAX API |

**Pré-requis**

* Visual Studio 2019, installé

Connaissance de base des concepts et des commandes de Git, comme les dépôts, les duplications (fork), les clones, les branches, l’indexation et la désindexation des changements, et l’envoi (push) de commits

* Un compte GitHub
* Avoir des connaissances de bases du DOM et de JSON

**Travail à réaliser**

Le travail à réaliser ci-dessous doit être référencé dans git et synchronisé sur github.

FredericBazile doit être mis en collaborateur du projet.

**Quelques recherches :**

Qu’est-ce que Ajax ?

Asynchronous JavaScript and XML, sert à modifier partiellement la page web d’un client sans avoir à en créer de nouvelles.

Qu’est-ce que le mode asynchrone ?

Le mode asynchrone est une communication qui se réalise en différé.

Qu’est-ce qu’une API ?

Qu’est-ce qu’une API  web?

**Consommer une API**

Pour pouvoir consommer (utiliser) une API web, il faut connaître son adresse et son mode de fonctionnement. La plupart des API web sont accessibles via une URL et utilisent le format JSON pour les échanges de données.

**Construire et envoyer une requête HTTP avec AJAX**

Créer une requête HTTP avec la méthode GET, afin de récupérer des données :

var request = new XMLHttpRequest();

request.open("GET", "http://url-service-web.com/api/users");

request.send();

Ce code nous permet d'envoyer une requête HTTP de type GET au service web se trouvant à l'adresse http://url-service-web.com/api/users .

Voici ce que fait le code, ligne par ligne :

* Ligne 1 : on crée un nouvel objet de type XMLHttpRequest qui correspond à notre objet AJAX. C'est grâce à lui qu'on va créer et envoyer notre requête ;
* Ligne 2 : on demande à ouvrir une connexion vers un service web. C'est ici que l'on précise quelle méthode HTTP on souhaite, ainsi que l'URL du service web ;
* Ligne 3 : on envoie finalement la requête au service web.

**Récupérez le résultat de la requête**

Pour cela, nous allons devoir utiliser la **propriété**  onreadystatechange  en lui passant une fonction. Cette fonction sera appelée à chaque fois que l'état de la requête évolue.

Voici les différents états possibles :

* UNSENT  (code 0) : l'objet est prêt mais la méthode  open()  n'a pas encore été appelée ;
* OPENED  (code 1) :  open()  a été appelé ;
* HEADERS\_RECEIVED  (code 2) :  send()  a été appelé, les headers et  status  sont disponibles au sein de l'objet ;
* LOADING  (code 3) : réception en cours, les données reçues sont partielles ;
* DONE  (code 4) : requête terminée.

C'est  DONE  qui va nous intéresser car c'est à ce moment-là que la requête est terminée et que nous venons de recevoir le résultat du service web. Pour récupérer l'état actuel de la requête, la fonction que l'on passe à  onreadystatechange  contiendra un objet  this  directement accessible dans la fonction, et qui nous permettra d'accéder aux propriétés suivantes :

* readyState  : qui contient l'état de la requête ;
* status  : qui contient le code de statut de la requête (souvenez-vous, *2xx* quand ça s'est bien passé, *3xx* pour les redirections, *4xx* pour les erreurs...) ;
* responseText  : qui contient la réponse du service web au format texte. Ainsi, si le texte que l'on attend est au format JSON, il va falloir le transformer en objet JavaScript avec la fonction  JSON.parse(texteJSON)

Voici comment procéder pour récupérer la météo actuelle sur Paris avec l'API fournie par www.prevision-meteo.ch (il existe bien entendu d’autres sites permettant de récupérer des informations météo mais ce site a l’avantage de ne pas demander de clé).

var request = new XMLHttpRequest();

request.onreadystatechange = function() {

if (this.readyState == XMLHttpRequest.DONE && this.status == 200) {

var response = JSON.parse(this.responseText);

console.log(response.current\_condition.condition);

}

};

request.open("GET", "https://www.prevision-meteo.ch/services/json/paris");

request.send();

**Travail à faire**

Dans un premier temps, créer une page web qui interroge l’API de météo (Le document nommé SLAM5\_TP\_JSON AJAX API-recuperation-donnees-meteo.pdf vous informe complètement sur l’utilisation de l’API.)

**Capturer l’évènement submit d’un formulaire :**

L’événement submit est émis lorsqu’un formulaire est soumis au serveur.

Notez que l’événement submit se déclenche uniquement sur l’élement form, et pas sur les éléments button ou input submit. (Les formulaires sont soumis, pas les boutons.)

L’exemple ci-dessous utilise EventTarget.addEventListener() pour écouter l’évènement submit du formulaire. Il affiche dans la console l’event.timeStamp et prévient l’action par défaut de l’évènement submit du formulaire.

function logSubmit(event) {

log.textContent = `Form Submitted! Time stamp: ${event.timeStamp}`;

event.preventDefault();

}

const form = document.getElementById('form');

const log = document.getElementById('log');

form.addEventListener('submit', logSubmit);

**Travail à faire**

Dans un deuxième temps, créer un formulaire permettant à l’utilisateur de choisir la ville dont il souhaite obtenir les informations et les afficher en utilisant le DOM.

Les informations s’afficheront en dessous du formulaire. Les informations à afficher concerne la localisation, le temps et le vent.

Idéalement une image permettra d’illustrer le temps.