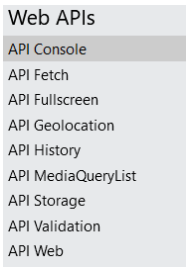
Les APIs :

Le JavaScript évolue très vite et de nombreuses fonctionnalités en béta-version sont devenues standard. On appelle ça des API (Application Programm Interface). Dans l'idée ça ressemble à des bibliothèques sauf qu'elles sont intégrées au navigateur.



On retiendra quelques API :

- {FETCH} qui remplace AJAX : requêtes asynchrones « one-shot »
- {WEBSocket} qui permet un dialogue full-duplex avec un serveur ayant le protocole WebSocket
- {SSE} (Server Sent Event) qui permet de faire comme {WEBSocket} mais uniquement dans le sens serveur→Navigateur (simplex), sans protocole particulier, ce qui est très souple pour interroger un serveur PHP.
- {IndexedDB} qui permet de stocker des données structurées (comme une table SQL) côté client. C'est un bon complément au « cache » du Navigateur. Très utilisé avec les PWA (Progressiv Web Application) et les « Service Workers »
- {Worker} qui permet de faire du JavaScript multi-thread côté client.

Fonction Promesses (*Angl : Promises*)

On a vu en TP différentes techniques : Fonctions classiques, arrow, callback, …
(Pour rappel, les TP et le site http://78.122.136.24/_accueil/opale/3-%20JavaScript%20-%20Les%20fonctions/co/0-module_Javacript-Fonctions.html) :

Une autre technique se nomme PROMISE (promesse) : ce sont des fonctions **asynchrones**.

Exemple :

```
////////////////////////////////////////////////////////////////////////////
// objet Promise (fonction asynchrone)
////////////////////////////////////////////////////////////////////////////
```

```
let calcul = new Promise ( (retour, msgErreur) => {
       // calcul long...
       let compteur = 100;
       let i = 0;
       while (compteur--) {
              i = Math.random();
              i *= 10000;
       }

      if (i < 5000)
            retour(i);
      else
          msgErreur("Erreur, le calcul n'est pas < 5000 : i=" + i);
}) ;

// Appel de la fonction :
calcul.then( resultat => console.log(resultat)).catch(erreur => console.log(erreur));

console.log("Je passe avant la fin de la promesse !");
```

Résultat : « Je passe avant la fin de la promesse » s'affiche toujours avant le résultat de la fonction : c'est asynchrone.
La méthode .then est le retour du callback « retour » et .catch est le retour du callback « msgErreur »

Façon d'importer les modules

Il est possible d'utiliser le mot clé « import » aussi côté Navigateur.
Voir le site : https://dev.to/iggredible/what-the-heck-are-cjs-amd-umd-and-esm-ikm

Les « import » côté NodeJS ou Navigateur dépendent de la façon dont les modules importés ont été « exportés ».
Le site officiel de NodeJS : https://nodejs.org/api/esm.html