Notion Promise Async Await

Les mots clé async et await est ajouté au JavaScript pour nous permettre d'écrire du code asynchrone : ils n'ajoutent aucune fonctionnalité mais fournissent une syntaxe plus intuitive et plus claire pour définir des fonctions asynchrones et utiliser des promesses.

```
async function bonjour(){
   return 'Bonjour';
 }

 /*Commentez le code pour l'exécuter
 //La valeur retournée par bonjour() est enveloppée dans une promesse
 bonjour().then(alert); // Bonjour
 */
```

Utiliser le mot clé async devant une fonction force la fonction à retourner une promesse et nous permet d'utiliser await dans celle-ci.

```
async function test(){
   const promise = new Promise((resolve, reject) => {
       setTimeout(() => resolve('Ok !'), 2000)
   });

   let result = await promise; //Attend que la promesse soit résolue
ou rejetée
   alert(result);
}
```

En utilisant le mot clé await devant une promesse, on oblige le JavaScript à attendre que la promesse soit consommée. Si la promesse

est résolue, le résultat est retourné. Si elle est rompue, une erreur (exception) est générée.

Utiliser async / await permet ainsi d'écrire du code asynchrone qui ressemble dans sa structure à du code synchrone auquel nous sommes habitués et nous permet notamment de nous passer de then() et de catch() (qu'on va tout de même pouvoir utiliser si le besoin s'en ressent).

```
function loadScript(src){
    return new Promise((resolve, reject) => {
        let script = document.createElement('script');
        script.src = src;
        document.head.append(script);
        script.onload = () => resolve('Fichier ' + src + ' bien
chargé');
        script.onerror = () => reject(new Error('Echec de chargement de
' + src));
    });
}

async function test(){
    try{
        const test1 = await loadScript('boucle.js');
        alert(test1);
        const test2 = await loadScript('blblbl.js');
        alert(test2);
        const test3 = await loadScript('cdcdcd.js');
        alert(test3);
    }catch(err){
        alert(err);
        let script = document.head.lastChild;
        script.remove(); //Supprime le script ajouté qui n'a pas pu
être lu
    }
}
```