Notions

- **Fonction asynchrone** : c'est une fonction qui se déroule de façon asynchrone et utilisant le système de « promesse »
- **Synchrone :** Le programme se met en attente d'une tâche synchrone qui se termine pour continuer son déroulement
- **Asynchrone** : Le programme continue sont déroulement sans attendre la fin de la tâche asynchrone.
- **Promesse** : les promesses permettent de réaliser des traitements asynchrones, par l'intermédiaire de la fonction « then() ». Les fonctions « catch » et « finally » sont également disponibles ES9
- **« async »** : Permet d'indiquer qu'une fonction est asynchrone
- « **await** » : permet de mettre en attente le retour d'une fonction

Exemples

```
const promise1 = new Promise((resolve, reject) => {
    console.log("Etape 1");
    resolve("coucou");
    console.log("Etape 2");
});

promise1.then((value) => {
    console.log("Etape 3");
    console.log(value);
});
```

```
Etape 1
Etape 2
Etape 3
coucou
```

```
const promise1 = new Promise((resolve, reject) => {
    console.log("Etape 1");
    setTimeout(() => {
        resolve('coucou');
    }, 2000);
    console.log("Etape 2");
});

promise1.then((value) => {
    console.log("Etape 3");
    console.log(value);
});
```

```
Etape 1
Etape 2
```

Après 2 s

```
Etape 1
Etape 2
Etape 3
coucou
```

La fonction « then » s'exécute après l'autre partie

ES8

```
function tempo(){
    return new Promise(function(resolve,reject){
        console.log("en cours");
        setTimeout(() => {
            resolve("coucou");
            console.log("fin du traitement");
        },2000)
    });
}

async function ditbonjour() {
    console.log("début");
    const affichageResultat = await tempo();
    console.log(affichageResultat);
    console.log("fin du programme");
}

ditbonjour();
```

```
début
en cours
```

Après 2 s

```
fin du traitement
coucou
fin du programme
```