PROMESSES JS

Objet exécuté dans le futur mais qui promet de renvoyer une des 2 valeurs **resolve** ou **reject** en fonction du résultat.

Un "code de production" qui fait quelque chose et prend du temps. Par exemple, un code qui charge les données sur un réseau.

Un "code consommateur" qui veut le résultat du "code producteur" une fois qu'il est prêt. De nombreuses fonctions peuvent avoir besoin de ce résultat.

Une promesse est un objet JavaScript spécial qui relie le "code de production" et le "code de consommation" ensemble.

Le "code producteur" prend tout le temps nécessaire pour produire le résultat promis, et la "promesse" met ce résultat à la disposition de tout le code souscrit lorsqu'il est prêt.

Les arguments resolve et reject sont des rappels fournis par JavaScript lui-même.

Notre code se trouve uniquement à l'intérieur de l'exécutor.

`Programme 1`

```
Cette promesses renvoi forcément => Résolu (resolve)


var promise = new Promise((resolve,reject) => {
    resolve('Promises are meant to be broken.');
});
console.log(promise);
```

`Programme 2`

```
var promise = new Promise((resolve,reject) => {
    resolve('Promises are meant to be broken.');
});
promise.then((result) => {
    console.log(result);
}).catch((error) => {
    console.log(error);
})

Cette promesse renvoi résolu mais est susceptible de lever
une erreur si elle n’est pas résolue
```

```
var promise1 = new Promise((resolve, reject) => {
    resolve('Inside Promise 1.');
});
var promise2 = new Promise((resolve, reject) => {
    resolve('Inside Promise 2.');
});
Promise.all([promise1, promise2])
    .then((result) => {
        console.log(result);
    }).catch((error) => {
        console.log(error);
    })
```

2 promesses tenues avant de l’exécution du résultat

```
// First Promise
var promise1 = new Promise((resolve, reject) => {
    var a = 15;
    var b = 5;
    var c = a + b;
    //Resolving the sum obtained.
    resolve(c);

});

// Second Promise
var promise2 = new Promise((resolve, reject) => {

    var x = 5;
    var sum = null;

    // Obtaining the value returned by first promise.
    promise1.then((result) => {
        sum = result + x;
        resolve(sum);

    });
```

```
});
// Obtaining the sum returned by second promise.

promise2.then((finalResult) => {
    console.log(finalResult);
}).catch((error) => {
    console.log(error);
})
</script>
```

une promesse dans une promesse

fetch est une autre fonction ajax basée sur les promesses
donc on peux utiliser .then (promesse ES6)

```
<script>

fetch('https://jsonplaceholder.typicode.com/posts')
    .then(res => {
        res.json()
            .then((data) => {
                console.log(data);
            })

    }).catch((error) => {
        console.log(error);
    })
</script>
```

```
<script>

fetch('https://jsonplaceholdertypicode.com/posts')
    .then(res => {
        res.json()
            .then((data) => {
                console.log(data);
            })

    }).catch((error) => {
        console.log(error);
    })
</script>
```

cas de l’erreur.