MÉTAPHORE POUR COMPRENDRE LES PROMISES

Imagine que tu commandes une **pizza** dans une pizzeria.

Étape 1 : Tu passes la commande

Tu demandes une pizza. Le cuisinier dit :
"Ta pizza sera prête dans 20 minutes."
Tu ne l'as pas **immédiatement**, mais tu sais qu'elle va arriver **plus tard**.
En JavaScript, cette commande est comme **une Promise**.

```
const pizzaPromise = commanderPizza();
```

Étape 2 : Le cuisinier te fait une promesse

Il te promet soit :

- "Tu auras ta pizza" → `resolve()`
- "Il y a eu un problème (four cassé...)" → `reject()`

```
const pizzaPromise = new Promise((resolve, reject) => {
 let fourEnMarche = true;

 if (fourEnMarche) {
   setTimeout(() => {
     resolve("Voici ta pizza  !");
   }, 2000); // après 2 sec
 } else {
   reject("Le four est cassé ");
 }
});
```

Étape 3 : Tu attends la pizza

Tu peux faire autre chose pendant que la pizza est en préparation.

Puis, quand elle est prête, tu veux **la manger** (faire une action).

```
pizzaPromise
 .then(pizza => {
   console.log("Miam  : ", pizza);
 })
 .catch(erreur => {
   console.log("Oh non  :", erreur);
 });
```

Mot	Signification
`new Promise()`	La commande
`resolve()`	Tout s'est bien passé
`reject()`	Erreur ou échec
`.then()`	Que faire si ça marche
`.catch()`	Que faire si ça rate

Exercices Corrigés :

Exercice 1 : Crée une promesse qui renvoie "Bonjour" après 1 seconde

```
function direBonjour() {
 return new Promise((resolve) => {
  setTimeout(() => {
   resolve("Bonjour ");
  }, 1000);
 });
}

direBonjour().then(msg => console.log(msg)); // Affiche "Bonjour " après 1 sec
```

Exercice 2 : Crée une promesse qui donne une note si l'étudiant a >10, sinon elle échoue

```
function verifierNote(note) {
 return new Promise((resolve, reject) => {
  if (note >= 10) {
   resolve("Réussi ");
  } else {
   reject("Échec ");
  }
 });
}

// test
verifierNote(15).then(console.log).catch(console.error);  // Réussi
verifierNote(8).then(console.log).catch(console.error);   // Échec
```

Exercice 3 : Simule un téléchargement avec un message "Téléchargement terminé" après 2 secondes

```
function telechargerFichier() {
 return new Promise((resolve) => {
  setTimeout(() => {
   resolve("Téléchargement terminé ");
  }, 2000);
 });
}

telechargerFichier().then(msg => console.log(msg));
```

Exercice 4 :

```
function commanderPizza() {
 return new Promise((resolve, reject) => {
  const fourEnMarche = true;

  if (!fourEnMarche) {
   return reject("Erreur : le four est cassé ");
  }

  console.log("Commande passée. Préparation en cours...");
  let secondes = 0;

  const interval = setInterval(() => {
   secondes++;
   console.log(`Pizza en cuisson... ${secondes}s`);
  }, 1000);

  setTimeout(() => {
   clearInterval(interval);
   resolve("Votre pizza est prête  !");
  }, 5000);
 });
}

commanderPizza()
 .then((message) => {
  console.log(" Succès :", message);
 })
 .catch((erreur) => {
  console.error(" Erreur :", erreur);
 });
```

Même exemple avec await/async

```
function commanderPizza() {
 return new Promise((resolve, reject) => {
  const fourEnMarche = true;

  if (!fourEnMarche) {
   return reject("Erreur : le four est cassé ");
  }

  console.log("Commande passée. Préparation en cours...");
  let secondes = 0;

  const interval = setInterval(() => {
   secondes++;
   console.log(`Pizza en cuisson... ${secondes}s`);
  }, 1000);
```

```
   setTimeout(() => {
     clearInterval(interval);
     resolve("Votre pizza est prête  !");
   }, 5000);
  });
}

async function lancerCommande() {
  try {
    const resultat = await commanderPizza();
    console.log(" Succès :", resultat);
  } catch (erreur) {
    console.error(" Erreur :", erreur);
  }
}

lancerCommande();
```

Exercice 5 : Lire un fichier json

Soit le fichier json :

```
[
  { "id": 1, "name": "ALI" },
  { "id": 2, "name": "SARA" },
  { "id": 3, "name": "RIDA" }
]
```

Version1 : utiliser fetch avec then et catch

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => {
    if (!response.ok) {
      throw new Error("Erreur lors du chargement du fichier JSON.");
    }
    return response.json();
  })
  .then(data => {
    console.log("Liste des utilisateurs :");
    data.forEach(user => console.log(user.name));
  })
  .catch(error => {
    console.error("❌ Problème :", error.message);
  });
```

Version2 : utiliser fetch avec async et await

```
async function lireUtilisateurs() {
  try {
    const response = await fetch("https://jsonplaceholder.typicode.com/users");
    if (!response.ok) {
      throw new Error("Erreur lors du chargement du fichier JSON.");
    }

    const data = await response.json();
    console.log("Liste des utilisateurs :");
    data.forEach(user => console.log(user.name));
  } catch (error) {
    console.error("❌ Problème :", error.message);
  }
}

lireUtilisateurs();
```