TP : Tests unitaires et fonctionnels en Node.js

Objectifs pédagogiques

- Comprendre la différence entre test unitaire, test d'intégration et test fonctionnel.
- Savoir écrire et exécuter des tests en Node.js sans framework externe.
- Apprendre à valider automatiquement le bon fonctionnement d'un programme.

Contexte du TP

Nous développons une application d'e-commerce.
Il est possible d'acheter des produits grâce à un porte-monnaie virtuel inclus dans l'application.
Le code suivant définit les principales fonctions de l'application et leur utilisation :

Code de l'application

```
// On définit une classe représentant un panier
class Basket {
 constructor(items = [], totalPrice = 0) {
  this.items = items;
  this.totalPrice = totalPrice;
 }
}

function addToBasket(basket, item) {
 basket.items.push(item);
 basket.totalPrice = basket.totalPrice + item.price;
}

function removeFromBasket(basket, item) {
 for (let i = 0; i < basket.items.length; i++) {
  if (JSON.stringify(item) === JSON.stringify(basket.items[i])) {
   basket.items.splice(i, 1);
   basket.totalPrice = basket.totalPrice - item.price;
   break;
  }
 }
}
```

```
function transactionAllowed(userAccount, priceToPay) {
  if (userAccount.balance >= priceToPay) {
    return true;
  }
  return false;
}

function payBasket(userAccount, basket) {
  if (transactionAllowed(userAccount, basket.totalPrice)) {
    userAccount.balance = userAccount.balance - basket.totalPrice;
    console.log('Paiement du panier réussi');
  } else {
    console.log('Paiement du panier échoué');
  }
}

// Exemple d'utilisation
const currentBasket = new Basket();

const item1 = { name: 'Carte mère', price: 100 };
const item2 = { name: 'Carte graphique', price: 300 };
const user = { name: 'Perceval', balance: 500 };

addToBasket(currentBasket, item1);
addToBasket(currentBasket, item2);

// Suppression d'un produit du panier
removeFromBasket(currentBasket, item1);
console.log(currentBasket);

// Paiement du panier
payBasket(user, currentBasket);
console.log(user);
// Perceval n'a plus que 200 euros
```

Partie 1 — Tests unitaires et fonctionnels

Les tests seront développés dans un fichier séparé (par exemple testEcommerce.js).
Chaque test retournera true si le résultat attendu est obtenu, et false sinon.

🧪 Question 1 — Test unitaire : testAdd()

Développer un test unitaire qui ajoute un produit au panier et vérifie que le montant total est correct.
Indice :
- Créer un panier vide.
- Ajouter un produit (ex : { name: "Carte mère", price: 100 }).
- Vérifier que basket.totalPrice vaut 100 après l'ajout.

🧪 Question 2 — Test unitaire : testRemove()

Développer un test unitaire qui supprime un produit du panier et vérifie que le montant total est correct.
Indice :
- Créer un panier vide.
- Ajouter un produit, puis le retirer.
- Vérifier que le panier est de nouveau vide et que basket.totalPrice vaut 0.

🧪 Question 3 — Test factorisé : testAddRemove()

On constate qu'il est possible de factoriser les tests unitaires précédents.
Donner un test unitaire unique qui vérifie successivement :
- l'ajout d'un produit au panier,
- puis sa suppression, dans une même fonction.
Indice :
Le test de retrait dépend de l'ajout. Il est donc possible d'enchaîner les deux opérations.

💰 Question 4 — Test unitaire : testTransactionAllowed()

Développer un test unitaire qui teste complètement la fonction transactionAllowed().
Chaque branche de la condition devra être vérifiée.
Indice :
- Créer un utilisateur avec un solde de 500 €.
- Vérifier qu'une transaction de 400 € est autorisée.
- Vérifier qu'une transaction de 600 € est refusée.

📋 Question 5 — Test fonctionnel : testPayBasket()

Écrire un test fonctionnel complet pour vérifier le bon déroulement d'un paiement.
Indice :

Formateur: DIALLO ALIMOU

- Créer un utilisateur avec un solde de 500 €.
- Créer un panier avec un produit coûtant 300 €.
- Effectuer le paiement une première fois : la transaction doit réussir et le solde passer à 200 €.
- Tenter un second paiement : la transaction doit échouer et le solde rester à 200 €.

🚀 Question 6 — Fonction principale : testAppEcommerce()

Écrire une fonction testAppEcommerce() qui lance successivement tous les tests précédents.
Cette fonction affichera "OK" si tous les tests ont réussi, et "ERREUR" sinon.
Indice :
Tous les tests renvoient true ou false.
On peut chaîner les résultats avec l'opérateur logique && :

```
function testAppEcommerce() {
  let success = testAdd();
  success = success && testRemove();
  success = success && testAddRemove();
  success = success && testTransactionAllowed();
  success = success && testPayBasket();

  if (success) {
    console.log("OK");
  } else {
    console.log("ERREUR");
  }
}
```