05 - JavaScript : Les fonctions

Exercices pour Appréhender les Fonctions en JavaScript

Ces exercices sont conçus pour aider à comprendre et à utiliser les fonctions JavaScript, en commençant par les fondamentaux et en progressant vers des scénarios plus complexes.

Niveau 1 : Les Fondamentaux

L'objectif ici est de s'assurer que les apprenants comprennent la syntaxe de base, la déclaration, l'appel et le concept de `return`.

Exercice 1 : Salutation Simple

1. Créez une fonction nommée `saluer` qui ne prend aucun argument.
2. Cette fonction doit afficher "Bonjour le monde !" dans la console.
3. Appelez cette fonction.

Exercice 2 : Salutation Personnalisée

1. Créez une fonction nommée `saluerNom` qui prend un argument : `nom`.
2. Cette fonction doit afficher "Bonjour, [nom] !" dans la console (où `[nom]` est la valeur de l'argument).
3. Appelez cette fonction plusieurs fois avec des noms différents.

Exercice 3 : Calcul de Carré

1. Créez une fonction nommée `calculerCarre` qui prend un argument : `nombre`.
2. Cette fonction doit **retourner** le carré de ce nombre (sans l'afficher dans la console).
3. Appelez la fonction et stockez le résultat dans une variable, puis affichez cette variable dans la console.

Niveau 2 : Arguments et Portée (Scope)

Ces exercices introduisent l'idée de plusieurs arguments, les valeurs par défaut et une introduction douce à la portée des variables.

Exercice 4 : Somme de Deux Nombres

1. Créez une fonction nommée `additionner` qui prend deux arguments : `a` et `b`.
2. Cette fonction doit retourner la somme de `a` et `b`.
3. Testez la fonction avec différentes paires de nombres.

Exercice 5 : Prix Total avec Taxes

1. Créez une fonction nommée `calculerPrixTotal` qui prend deux arguments : `prixHT` (prix hors taxes) et `tauxTVA` (taux de TVA, par exemple 0.20 pour 20%).
2. Le `tauxTVA` devrait avoir une **valeur par défaut** de `0.20` si non spécifié.
3. La fonction doit retourner le prix TTC (prix hors taxes + (prix hors taxes * taux de TVA)).
4. Testez la fonction en spécifiant le taux de TVA et sans le spécifier.

Exercice 6 : Introduction à la Portée

1. Déclarez une variable `messageGlobal` en dehors de toute fonction avec la valeur "Je suis global".
2. Créez une fonction nommée `afficherMessages`.
3. À l'intérieur de cette fonction, déclarez une variable `messageLocal` avec la valeur "Je suis local".
4. À l'intérieur de la fonction, affichez `messageGlobal` et `messageLocal`.
5. En dehors de la fonction, essayez d'afficher `messageLocal`. Observez l'erreur. Expliquez pourquoi.

Niveau 3 : Fonctions comme Valeurs (Expressions de Fonctions) et Rappels (Callbacks)

Il est temps de montrer que les fonctions peuvent être assignées à des variables et passées comme arguments.

Exercice 7 : Fonction Anonyme (Expression de Fonction)

1. Créez une fonction anonyme qui multiplie deux nombres.

2. Assignez cette fonction à une variable nommée `multiplier`.
3. Appelez la fonction via la variable `multiplier` et affichez le résultat.

Exercice 8 : Fonction en Argument (Callback Simple)

1. Créez une fonction nommée `executerOperation` qui prend trois arguments : `nombre1`, `nombre2` et `operation` (où `operation` sera une autre fonction).
2. `executerOperation` doit appeler la fonction `operation` en lui passant `nombre1` et `nombre2`, et retourner le résultat.
3. Créez deux fonctions séparées : `addition` et `soustraction`, qui prennent deux nombres et retournent leur somme ou différence.
4. Utilisez `executerOperation` avec `addition` et `soustraction` comme fonctions `operation`. Affichez les résultats.

Exercice 9 : Utilisation de `forEach` avec un Callback

1. Créez un tableau de nombres (ex: `[10, 20, 30, 40]`).
2. Utilisez la méthode `forEach` sur ce tableau pour afficher chaque nombre dans la console. Le callback de `forEach` sera une fonction anonyme.
3. Modifiez le callback pour qu'il affiche "Le nombre [nombre] est à l'index [index]".

Niveau 4 : Fonctions Fléchées (Arrow Functions) et Contexte (`this`) - Introduction

Présenter les fonctions fléchées pour leur syntaxe concise et aborder subtilement la notion de `this`.

Exercice 10 : Refactorisation avec Fonctions Fléchées

1. Reprenez l'exercice 2 ("Salutation Personnalisée") et l'exercice 4 ("Somme de Deux Nombres").
2. Réécrivez les fonctions `saluerNom` et `addtionner` en utilisant la **syntaxe des fonctions fléchées**.
3. Testez pour vous assurer qu'elles fonctionnent de la même manière.

Exercice 11 : Utilisation de `setTimeout` avec une Fonction Fléchée

1. Créez un simple message qui apparaît après 2 secondes.
2. Utilisez `setTimeout` et une **fonction fléchée** comme callback.
3. Le message pourrait être "Ce message apparaît après 2 secondes !".

Projets Mini-Fonctionnels (Optionnel mais recommandé)

Ces mini-projets permettent d'intégrer plusieurs concepts.

Projet 1 : Gestionnaire de Tâches Simple (Fonctions pour ajouter/supprimer)

1. Créez un tableau vide pour stocker des tâches.
2. Écrivez une fonction `ajouterTache(description)` qui ajoute une tâche (objet `{ id: ..., description: ... }`) au tableau.
3. Écrivez une fonction `supprimerTache(id)` qui retire une tâche du tableau.
4. Écrivez une fonction `afficherTaches()` qui affiche toutes les tâches actuelles dans la console.
5. Testez en ajoutant, affichant, supprimant et affichant à nouveau.

Projet 2 : Calculatrice Basique (Fonctions pour chaque opération)

1. Écrivez quatre fonctions : `add(a, b)`, `subtract(a, b)`, `multiply(a, b)`, `divide(a, b)`.
2. Créez une fonction principale `calculer(operation, num1, num2)` qui prend l'opération souhaitée (sous forme de chaîne, ex: "add") et les deux nombres.
3. À l'intérieur de `calculer`, utilisez une structure conditionnelle (`if/else if` ou `switch`) pour appeler la bonne fonction d'opération et retourner le résultat.
4. Gérez le cas de la division par zéro.