JS Conditions multiples, boucles variées et objets en JS

Exercice 1 – Compter les multiples de 3 ou 5

Objectifs : boucle `for`, condition combinée `||`, opérateur `!`

1. Crée un tableau vide `multiples`.
2. À l'aide d'une boucle `for` qui parcourt les entiers de 1 à 100 inclus :
   - Si le nombre est **multiple de 3 ou de 5** (`nombre % 3 === 0 || nombre % 5 === 0`), fais `multiples.push(nombre)`.
   - Sinon, ajoute à `multiples` la chaîne `"!" + nombre` (ex. `"!7"`).
3. Affiche le tableau final dans la console.

Exercice 2 – Filtrer les notes valides

Objectifs : boucle `while`, condition `&&`, opérateur `!`, méthode `push`

1. Déclare `const notes = [15, -3, 12, 0, 19, 22, 8]` et un tableau vide `valides`.
2. Parcours `notes` avec une boucle `while` (index `i`) ; ajoute dans `valides` chaque note **strictement comprise entre 0 et 20** (`note > 0 && note < 20`).
3. En sortie, log `valides` et le nombre d'éléments écartés (`notes.length - valides.length`).

Exercice 3 – Parcourir un objet d'utilisateurs

Objectifs : boucle `for in`, opérateur `!`, condition composée

1. Crée l'objet :

```
const users = {
  alice: 18,
  bob: 25,
  charlie: 17,
```

```
  dave: 30
};
```

2. À l'aide de `for in`, affiche :
   - Le nom de l'utilisateur **si** son âge est **≥ 18 et ≠ 25** (`age >= 18 && age !== 25`).
   - Sinon affiche `"Accès refusé : " + nom`.
3. Termine par le nombre total d'accès accordés.

Exercice 4 – Addition sélective dans un tableau

Objectifs : boucle `for of`, `push`, conditions multiples

1. Déclare `const nombres = [2, 7, 10, 21, 14, 3];` et un tableau `resultats = [];`.
2. Parcours `nombres` avec `for of`.
   - Si un nombre est **pair ET > 5** (`n % 2 === 0 && n > 5`) ou **impair ET < 10** (`n % 2 !== 0 && n < 10`), additionne-le à la variable `somme` et fais `resultats.push(n)`.
3. Affiche `somme` et `resultats`.

Exercice 5 – Inventaire en boucle `while`

Objectifs : objet, `while`, `push`, opérateurs logiques

1. Déclare `const stock = { pommes: 4, bananes: 0, poires: 3, mangues: 1 };`
2. Crée un tableau vide `rupture`.
3. Utilise `for in` pour parcourir `stock` ; **tant qu'une** quantité est **> 0**, décrémente-la et affiche `${fruit}: ${stock[fruit]}`.
4. Quand la quantité devient 0, ajoute le nom du fruit à `rupture` **une seule fois** (condition avec `&& !rupture.includes(fruit)`).
5. Affiche la liste `rupture` à la fin.

Exercice 6 – Tableau de nombres aléatoires et bonus

Objectifs : boucle `for`, `push`, `||`, `!`

1. Crée un tableau vide `nums`. Remplis-le de 20 nombres entiers aléatoires de 1 à 30 via une boucle `for`.

2. Parcours de nouveau le tableau :
   - Si le nombre est **multiple de 4 ou de 6**, multiplie-le par 2 (→ `nums[i] *= 2`).
   - Sinon, si le nombre **n'est pas pair** (`n % 2 !== 0`), remplace-le par `1`.
3. Log le tableau final.

💡 **DANGER ZONE - BONUS**

Exercice 7 – Fusion d'objets sans fonction

Objectifs : `for in`, `&&`, opérateur `!`

1. Déclare deux objets :

```
const o1 = { a: 1, b: 2, c: 3 };
const o2 = { b: 4, c: 3, d: 5 };
const fusion = {};
```

2. Copie dans `fusion` les paires clé-valeur de `o1`.
3. Parcours `o2` avec `for in` :
   - **Si** la clé n'existe **pas encore** dans `fusion` (`!(key in fusion)`) **OU** si la valeur diffère (`fusion[key] !== o2[key]`), assigne `fusion[key] = o2[key]`.
4. Affiche l'objet `fusion`.

Exercice 8 – Recherche dans un tableau d'objets

Objectifs : tableau d'objets, boucle `for`, conditions composées

1. Déclare :

```
const produits = [
  { nom: "stylo", prix: 1.2, dispo: true },
  { nom: "cahier", prix: 2.5, dispo: false },
  { nom: "crayon", prix: 0.8, dispo: true },
```

```
  { nom: "gomme", prix: 0.5, dispo: true }
];
```

2. Parcours avec `for` :
   - Si `dispo` est **true ET** `prix < 2`, pousse le nom dans `achats`.
   - Sinon, si `!dispo || prix >= 2`, ajoute le nom dans `ignores`.
3. Affiche `achats` et `ignores`.

Exercice 9 – Palindrome simplifié (sans fonction)

Objectifs : `for`, `push`, `!`, conditions multiples

1. Lis une chaîne dans `const mot = "kayak";` (ou toute autre).
2. Copie chaque caractère dans `const chars = [];` avec `for` + `push`.
3. Crée `const inverse = [];` et remplis-le en parcourant `chars` **en sens inverse**.
4. Compare : si `chars[i] !== inverse[i]` **OU** `!chars[i]`, détermine que ce n'est **pas** un palindrome.
5. Affiche le résultat.

Exercice 10 – Somme conditionnelle avec `for of`

Objectifs : tableau à 2 dimensions, `for of`, `&&`, `||`

1. Déclare `const grilles = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];`
2. Initialise `let somme = 0;`
3. Parcours chaque sous-tableau avec `for of`, puis chaque nombre :
   - Ajoute auxomme **si** le nombre est **pair ET ≥ 4 OU impair ET ≤ 5**.
4. Log la somme.
