Exo 1 — Déstructuration & Rest/Spread sur un tableau de stats

On dispose d'un tableau d'objets de la forme `{ name, score, city }`.

```
const stats = [
  { name: "Alice", score: 92, city: "Lyon" },
  { name: "Bob",   score: 75, city: "Paris" },
  { name: "Eve",   score: 88, city: "Lille" },
  { name: "Dan",   score: 66, city: "Nice" },
  { name: "Cara",  score: 81, city: "Bordeaux" }
];
```

Objectif

1. Obtenir le **top 3** des éléments avec les meilleurs `score` (ordre décroissant) **sans modifier** le tableau d'origine.
2. Calculer la **moyenne des scores du reste** (tous les éléments en dehors du top 3).
3. Construire un objet `resume` avec le **spread** et produire un **message final** via **template literal**.

Contraintes

- Utiliser la **déstructuration de tableau** pour extraire le top 3
- **Pas de boucles impératives** inutiles (`for`, `while`) : privilégier `map`, `filter`, `reduce`.
- Préserver l'**immutabilité** (ne pas trier en place l'original).

Sortie attendue
Un objet `resume` de la forme :

```
{
  top3: [
    { name: 'Alice', score: 92, city: 'Lyon' },
    { name: 'Eve', score: 88, city: 'Lille' },
    { name: 'Cara', score: 81, city: 'Bordeaux' }
  ],
  moyenneDesAutres: 70.5,
  message: 'Top 3: Alice, Eve, Cara — Moyenne des autres: 70.50'
}
```

Exo 2 — Set / Map : nettoyer & indexer

On dispose :

- d'une **liste de tags** (chaînes) potentiellement dupliqués et mal formatés (majuscules/minuscules, espaces).

```
const rawTags = [" JS", "React", "js", "react ", "Node", "node", " JS "];
```

- d'une **liste d'utilisateurs** de la forme :

```
const users = [
  { id: 1, name: "Alice", email: "alice@mail.com", address: { city: "Lyon" } },
  { id: 2, name: "Bob",   email: "bob@mail.com",   address: { city: "Paris" } },
  { id: 3, name: "Eve",   email: "eve@mail.com" } // city optionnelle
];
```

Objectifs

1. **Nettoyer** les tags : `trim()` + passer en **minuscule**, puis **dédupliquer** avec un `Set`.
   - Sortie : un **tableau** `uniqueTags` (pas un Set) pour l'affichage.
2. **Indexer** les utilisateurs dans un `Map` dont la clé est `user.id` et la valeur est un **profil** minimal :

```
{ id, name, email, city }
```

Montrer l'accès avec `map.get(id)`, l'insertion avec `map.set(key, value)`, et une **itération `for…of`**.

Exo 3 — Prototype → Classe (refactor OO)

On part d'un code "ancienne école" (fonction constructeur + méthode sur le prototype) et on le **refactore en classe ES6** avec :

- un **constructor** pour les propriétés d'instance,
- une **méthode d'instance** (définie via la syntaxe de classe → placée sur le prototype),
- **Variante** : une méthode **statique** `from(dto)` pour construire depuis un objet.

Données de départ (à refactorer)

```
function User(name, age) {
  this.name = name;
  this.age = age;
}

User.prototype.describe = function () {
  return `${this.name} (${this.age} ans)`;
};
```