Coding Challenge 1: Dog Age Study

Micheal and Omotayo are conducting a study on dogs. Each of them surveyed 5 dog owners about their dog's age, storing the data in separate arrays. The study aims to determine if a dog is an **adult** or a **puppy**. A dog is classified as:

- **Adult**: At least 3 years old.
- **Puppy**: Less than 3 years old.

Your task is to create a function `checkDogs` that does the following:

Requirements

1. **Correct Micheal's Data**:
    - Micheal discovered that the owners of the **first** and **last two dogs** have cats instead of dogs.
    - Create a shallow copy of Micheal's array and remove the cat ages (to avoid mutating the original array).
2. **Combine the Data**:
    - Merge Micheal's corrected data with Omotayo's data into a single array.
3. **Analyze Each Dog**:
    - For each dog in the combined data:
        - If the dog is an adult, log:
          `Dog number 1 is an adult, and is 5 years old`
        - If the dog is a puppy, log:
          `Dog number 2 is still a puppy` 🐶
4. **Test the Function**:
    - Run the function using the test datasets below.

Test Data

1. **Dataset 1**:
    - Micheal's data: `[3, 5, 2, 12, 7]`
    - Omotayo's data: `[4, 1, 15, 8, 3]`
2. **Dataset 2**:
    - Micheal's data: `[9, 16, 6, 8, 3]`
    - Omotayo's data: `[10, 5, 6, 1, 4]`

Solution

```
const checkDogs = function (dogsMicheal, dogsOmotayo) {
  // Step 1: Create a shallow copy of Micheal's array and remove the first and
last two elements
  const dogsMichealCorrected = dogsMicheal.slice(); // Copy the array
  dogsMichealCorrected.splice(0, 1); // Remove the first element
  dogsMichealCorrected.splice(-2); // Remove the last two elements
  console.log("Corrected Micheal's Data:", dogsMichealCorrected);

  // Step 2: Combine Micheal's corrected data with Omotayo's data
  const dogs = dogsMichealCorrected.concat(dogsOmotayo);
  console.log("Combined Data:", dogs);

  // Step 3: Log whether each dog is an adult or a puppy
  dogs.forEach(function (dog, i) {
    if (dog >= 3) {
      console.log(`Dog number ${i + 1} is an adult, and is ${dog} years old`);
    } else {
      console.log(`Dog number ${i + 1} is still a puppy 🐶`);
    }
  });
};

// Test Data 1
console.log("TEST DATA 1:");
checkDogs([3, 5, 2, 12, 7], [4, 1, 15, 8, 3]);

// Test Data 2
console.log("\nTEST DATA 2:");
checkDogs([9, 16, 6, 8, 3], [10, 5, 6, 1, 4]);
```

Step-by-Step Breakdown

1. **Correcting Micheal's Data**

- Use `slice()` to create a shallow copy of Micheal's array.
- Use `splice()` to:
  - Remove the first element: `dogsMichealCorrected.splice(0, 1)`
  - Remove the last two elements: `dogsMichealCorrected.splice(-2)`
- This ensures the original array remains unaltered.

2. **Combining the Data**

- Use `concat()` to merge Micheal's corrected data and Omotayo's data into one array.

3. **Analyzing the Data**

- Use `forEach()` to loop through the combined array:
  - Check if each dog's age is `>= 3` (adult) or `< 3` (puppy).

- Log the appropriate message with the dog's index and age using template literals.

4. **Testing the Function**

- Test the function with the provided datasets and verify the output.

Expected Output

Test Data 1

```
Corrected Micheal's Data: [5, 2]
Combined Data: [5, 2, 4, 1, 15, 8, 3]
Dog number 1 is an adult, and is 5 years old
Dog number 2 is still a puppy 🐶
Dog number 3 is an adult, and is 4 years old
Dog number 4 is still a puppy 🐶
Dog number 5 is an adult, and is 15 years old
Dog number 6 is an adult, and is 8 years old
Dog number 7 is an adult, and is 3 years old
```

Test Data 2

```
Corrected Micheal's Data: [16, 6]
Combined Data: [16, 6, 10, 5, 6, 1, 4]
Dog number 1 is an adult, and is 16 years old
Dog number 2 is an adult, and is 6 years old
Dog number 3 is an adult, and is 10 years old
Dog number 4 is an adult, and is 5 years old
Dog number 5 is an adult, and is 6 years old
Dog number 6 is still a puppy 🐶
Dog number 7 is an adult, and is 4 years old
```

Key Takeaways

1. **Avoiding Mutation**: Using `slice()` prevents changes to the original array, promoting good practices in functional programming.
2. **Efficient Array Operations**: Combining arrays with `concat()` and filtering elements with `splice()` are efficient and easy to use.
3. **Dynamic Output**: Template literals enhance clarity and readability when logging results dynamically.
4. **Testing for Robustness**: Running tests with diverse datasets ensures the function handles different inputs reliably.