JavaScript Objects Exercise: RPG Game

Exercise 1:

Create an object named `character` that represents details of a game character. The object should contain the properties `name` (string), `class` (string), `level` (number), `abilities` (array of strings), and `stats` (object with properties like `strength` , `dexterity` , `intelligence` , all numbers). Also, add a method `getOverallStrength` that returns the sum of the strength, dexterity, and intelligence stats.

Exercise 2:

- Add a new property `equipment` to the `character` object created in Exercise 1 without modifying the initial object declaration. The `equipment` should be an object with the properties `weapon` (string), `armor` (string), and `accessories` (array of strings).
- Log the first ability in the `abilities` array using both dot notation and bracket notation.
- Log the type of armor the character is wearing using both dot notation and bracket notation.

Exercise 3:

- Update the `level` of the `character` object to a higher level.
- Add a new ability to the `abilities` array.
- Delete the `class` property from the `character` object.
- Modify the `weapon` property in the `equipment` object using both dot and bracket notations.
- Log if the `character` object contains the `stats` property or not.

Exercise 4:

You have below an array of character objects. Each character object contains a nested object `attributes` with properties `health` (number) and `mana` (number).

- Log the `health` of the first character in the array.

- Write a function to find the average health of all characters in the array.

```
  const characters = [
    { name: "Eldrin", attributes: { health: 100, mana: 50
}},
    { name: "Mira", attributes: { health: 85, mana: 60 }}
  ];
```

Exercise 5:

- Merge two character objects into one.
- Write a code snippet that prevents further changes to a character object.
- Write a code snippet that prevents new properties from being added to a character object, but values of existing properties can still be changed.
- Log if a character object is sealed.
- Log if a character object is frozen.

Exercise 6:

Given the object `characterStats` below, write a function that logs all property names and their values separately. Use `Object.keys()` in your solution.

```
const characterStats = {
    name: "Eldrin",
    class: "Mage",
    level: 7,
    health: 100,
    mana: 200
};
```

Exercise 7:

Assume you have the `gameScores` object below where each property is a character name and its value is the character's experience points. Write a function to increase the experience points of a specific character by 100, but only if the character's current experience points are less than 1000. Use `Object.entries()` to find the character and modify its experience points.

```
const gameScores = {
    "Eldrin": 950,
    "Mira": 1200,
    "Thorn": 800
};
```

Exercise 8:

Given the object `quests` below where each key is a quest ID and each value is an object containing `name`, `difficulty`, and `reward`, write a function that returns an array of quests with a difficulty level higher than 'Medium'. Each array element should be an object with all original properties plus an additional `isHard` property set to true. Utilize `Object.entries()` in your solution.

```
const quests = {
    1: { name: "Find the Lost Sword", difficulty: "Easy",
reward: "100 gold" },
    2: { name: "Defeat the Dragon", difficulty: "Hard",
reward: "500 gold" },
    3: { name: "Escort the Merchant", difficulty:
"Medium", reward: "250 gold" }
};
```

Exercise 9:

You have the `skillLevels` object below that contains character classes as keys and arrays of numbers (skill levels) as values. Write a function that calculates the average skill level for each class and logs a summary. The summary should include the class name and its average skill level. Implement your solution using `Object.keys()`.

```
const skillLevels = {
    Ranger: [8, 9, 7, 10, 8.5],
    Mage: [8.5, 8, 9, 9.5, 7.5],
    Warrior: [7, 7.5, 8, 8.5, 9],
};
```

Here's how you can adapt the given exercise to fit an RPG game theme, focusing on quests and their rewards:

Exercise 10:

Given the object `questRewards` below where keys are quest names and values are the types of rewards they grant, write a function that creates a new object where the keys are the types of rewards and the values are arrays of quests that grant that type of reward. Use `Object.entries()` to traverse the original object and construct the new one.

```
const questRewards = {
    "Find the Lost Sword": "Legendary Weapon",
    "Defeat the Dragon": "Epic Armor",
    "Escort the Merchant": "Gold",
    "Discover the Ancient Ruins": "Legendary Weapon"
};
```