Iris Dataset Report

Answers and Code for Questions 2-10

2. map():

Added a random color from a list to each iris object.

Code:

```
irisesWithColors = irises.map(iris => ({
    ...iris,
    color: possibleColors[Math.floor(Math.random() * possibleColors.length)]
}));
```

3. filter():

Removed iris objects with sepalWidth >= 4.

Code:

```
filteredIrises = irisesWithColors.filter(iris => iris.sepalWidth < 4);
```

4. reduce():

Calculated average petalLength.

Code:

```
const totalPetalLength = irisesWithColors.reduce((acc, iris) => acc + iris.petalLength, 0);
const averagePetalLength = totalPetalLength / irisesWithColors.length;
```

5. find():

Found the first iris with petalWidth > 1.0.

Code:

```
const widePetal = irisesWithColors.find(iris => iris.petalWidth > 1.0);
```

Iris Dataset Report

6. some():

- Checked if any iris has petalLength > 10.

- Checked if any iris has petalLength === 4.2.

7. every():

- Verified all petalWidths are < 3.

- Verified all sepalWidths are > 1.2.

8. toSorted():

Sorted iris objects by petalWidth.

Code:

irisesWithColorsSorted = irisesWithColors.toSorted((a, b) => a.petalWidth - b.petalWidth);

9. Visualization Setup:

Used canvas API to draw each iris as a colored circle, with interactive hover effects showing species.

The Iris class handles drawing, positioning, and interaction.

10. Observations:

- petalLength values range sensibly.

- no iris had petalLength > 10.

- All petalWidths are below 3.

Extra Note:

=> does the same as calling a function and the return is automatically implemented in and called if needed.

11. Visualization Summary

The intention of the visualization was to explore the Iris dataset in a playful and interactive way using canvas graphics.

Each iris is represented by a colored circle, with the color randomly assigned from a palette. The size of the circle is proportional to the petal length of the flower. When hovering over a circle, the species name appears to help identify it.

This allows quick pattern recognition and data engagement. The visualization maps complex data into accessible visual form and showcases JS array methods like map, filter, reduce, find, some, every, and toSorted.