Small Report of “task week7-8”

Hubert Sia and Nadia Abdul Aziz

Solution for #2 map():

```
const possibleColors = ["#5d3fd3","#a73fd3","#d33fb5","#d35d3f","#d3a73f"];
const irisesWithColors = originalData.map(iris => ({
  ...iris,
  color: possibleColors[Math.floor(Math.random() * possibleColors.length)]
}));
```

- Creates new array irisesWithColors
- Copies all original properties with spread operator
- Adds new color property with random color selection

Solution for #3 filter():

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

- Returns new array with only irises where sepalWidth < 4
- Removes 6 records (144 remain**)**

Solution for #4 reduce():

```
const sum = irisesWithColors.reduce((total, iris) => total + iris.petalLength, 0);
const averagePetalLength = sum / irisesWithColors.length; // 3.76
```

- Accumulates sum of all petalLengths
- Divides by number of items for average

Solution for #5 find():

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

- Returns first iris where petalWidth > 1.0
- Returns object (e.g., petalWidth: 1.4) or undefined if none found

Solution for #6 and #7 some():

```
#6: const hasVeryLongPetal = irisesWithColors.some(iris => iris.petalLength > 10); // false
#7: const hasExactLength = irisesWithColors.some(iris => iris.petalLength === 4.2); // true
```

- First check returns false (max length is 6.9)
- Second check returns true (4.2 exists in data)

Solution for #8 and #9 every():

#8: const allNarrowPetals = irisesWithColors.every(iris => iris.petalWidth < 3); // true

#9: const allWideSepals = irisesWithColors.every(iris => iris.sepalWidth > 1.2); // true

- Both return true:
    - Max petalWidth is 2.5 (< 3)
    - Min sepalWidth is 2.0 (> 1.2)

Solution for #10 toSorted():

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

- Creates new sorted array
- Orders from smallest (0.1) to largest (2.5) petalWidth
- Uses modern toSorted() instead of mutating sort()

Brief Summary of Visualization (#11):

We decide to create an interactive floral garden where each iris datum becomes a dynamic flower.

The visualization:

1. **Accurately models** biological features:
    - Sepal size → Green base ellipse dimensions
    - Petal size → Colored petal shapes (4 per flower)
    - Species → Center color (gold/red/purple)
2. **Encodes data** through:
    - **Size scaling**: Measurements directly control element proportions
    - **Color coding**: Species instantly recognizable
    - **Sorting**: Flowers arranged by petal width (small→large)
3. **Interaction features**:
    - Hover reveals species/details
    - Toggleable gentle rotation animation
    - Responsive grid layout
4. **Design choices** prioritize:
    - Visual clarity over strict realism
    - Immediate pattern recognition
    - Engaging exploration of dataset relationships

The result is an interactive and intuitive botanical illustration that presents distributions and species differences at-a-glance with visual inspection but allows
closer inspection with organic interaction.