1.

```
const response = await fetch('data/iris.json');
const data = await response.json();
```

2.

```
let possibleColor = ["#5d3fd3","#a73fd3","#d33fb5","#d35d3f","#d3a73f"];

let irisesWithColors = data.map(iris => {
    let color = possibleColor[Math.floor(Math.random() * possibleColor.length)];
    return {
        ...iris,
        color: color
    };
});
```

3.

```
let filteredIrises = irisesWithColors.filter(
    function (iris) {
        return iris.sepalWidth < 4;
    }
);
```

4.

```
const sumPetalLength = irisesWithColors.reduce(
    function (acc, iris) {
        return acc + iris.petalLength;
    }, 0
);

const averagePetalLength = sumPetalLength / irisesWithColors.length;
```

5.

```
const foundIris = irisesWithColors.find(function (iris) {
    return iris.petalWidth > 1.0;
});
```

6.

```
const hasLongPetal = irisesWithColors.some(function (iris) {
    return iris.petalLength > 10;
});
```

7.

```
const hasSpecificPetal = irisesWithColors.some(function (iris) {
    return iris.petalLength === 4.2;
});
```

8.

```
const allPetalWidthsLessThan3 = irisesWithColors.every(function (iris) {
    return iris.petalWidth < 3;
```

```
});
```

9.

```
const allSepalWidthsGreaterThan1_2 = irisesWithColors.every(function (iris) {
    return iris.sepalWidth > 1.2;
});
```

10.

```
const irisesWithColorsSorted = irisesWithColors.toSorted(function(a, b) {
    return a.petalWidth - b.petalWidth;
});
```

11.

My initial idea for the visualization was to make a kaleidoscope out of the data, but there are too many factors in the data to apply and I wanted to use as many of them as I could. I ended up making a galaxy style visualization that is more abstract, where I could implement more of (but unfortunately not all) the variables without sticking to the flower motif. Once I finished the placement and styling of the “stars,” I added a simple animation that would make them spread out slowly. I still wasn’t satisfied with it because it felt too stiff, so I played with some interaction ideas and landed on making the mouse cursor apply an outward force to the objects. The variables in the constructor can be changed to make slightly different effects, and I’m really happy with it now.