CART 263

Prof. Sabine Rosenberg

Nat Nina, Breina Kelly

Task 7-8 Report

1. Fetch API

```
window.onload = async function () {
   console.log("task 7-8");

   try {

      let response = await fetch('../data/iris.json'); //response
      let parsedResultJS = await response.json();
      console.log(parsedResultJS);

      mapIrisesColors(parsedResultJS);

   }
   catch (err) {

      console.log(err)
   }
}
```

2. Map Irises

```
function mapIrisesColors(irisArray) {
   const possibleColor = ["#5d3fd3", "#a73fd3", "#d33fb5", "#d35d3f", "#d3a73f"];

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

   console.log("iris with random colors", irisesWithColors);
```

3. Filter irises with sepal width >= 4

```
   const filteredIrises = irisesWithColors.filter(sepalWidthFour);

   function sepalWidthFour(iris) {
      return iris.sepalWidth >= 4;
   }

   console.log("filtered irises", filteredIrises);
```

4. reduce to find average petal length

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

```
console.log("total petal length", totalPetalLength);

const averagePetalLength = totalPetalLength / irisesWithColors.length;

console.log("average petal length", averagePetalLength);
```

5. find to find petals with width > 1

```
const petalWidthOne = irisesWithColors.find(
   function (iris) {
      return iris.petalWidth > 1
   });

console.log("irises with petal width > 1", petalWidthOne);
```

6. some to find object with petal length > 10

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

console.log("irises with petal length > 10", petalLengthTen);
```

7. some to find object with petal length == 4.2

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

```
  console.log("irises with petal length == 4.2", petalLengthFourPointTwo);
```

8. every: check if all petalWidths are less than 3

```
  const allPetalWidthsUnderThree = irisesWithColors.every(
    function (iris) {
      return iris.petalWidth < 3;
  });
console.log("irises with petal width < 3?", allPetalWidthsUnderThree);
```

9. every: check if all sepalWidths are greater than 1.2

```
  const allSepalWidthsAboveOnePointTwo = irisesWithColors.every(
    function (iris) {
      return iris.sepalWidth > 1.2;
  });
console.log("irises with sepal width > 1.2?", allSepalWidthsAboveOnePointTwo);
```

10. toSorted: sort by petalWidth

```
  const irisesWithColorsSorted = irisesWithColors.toSorted((a, b) => a.petalWidth - b.petalWidth);
  console.log("irises sorted by petal width", irisesWithColorsSorted);

  const shuffledIrises = shuffleArray(irisesWithColors);
  renderIrisShapes(shuffledIrises);
}
```

11. Visualization

```
function clamp(value, min, max) {
  return Math.min(Math.max(value, min), max);
```

```
}

function shuffleArray(array) {
    for (let i = array.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [array[i], array[j]] = [array[j], array[i]];
    }
    return array;
}

function renderIrisShapes(allPetals) {
    const existing = document.getElementById("iris-flower");
    if (existing) existing.remove();

    const container = document.createElement("div");
    container.id = "iris-flower";
    container.style.width = "1000px";
    container.style.height = "1000px";
    container.style.margin = "50px auto";
    container.style.transform = "rotate(0deg)";
    document.body.appendChild(container);

    const petalCount = allPetals.length;

    allPetals.forEach((iris, i) => {
        const angle = (360 / petalCount) * i;

        // wrapper for each petal
        const wrapper = document.createElement("div");
```

```
    wrapper.style.position = "absolute";
    wrapper.style.top = "20%";
    wrapper.style.left = "50%";
    wrapper.style.transform = `rotate(${angle}deg)`;

    // petal
    const petal = document.createElement("div");
    const scaledWidth = clamp(iris.petalWidth *50, 10, 50);
    const scaledLength = clamp(iris.petalLength *50, 20, 170);
    const petalOffset = 20;
    petal.style.position = "absolute";
    petal.style.top = `-${scaledLength + petalOffset}px`;
    petal.style.width = `${scaledWidth}px`;
    petal.style.height = `${scaledLength}px`;
    petal.style.backgroundColor = iris.color;
    petal.style.borderRadius = "50%";
    //attach
    wrapper.appendChild(petal);
    container.appendChild(wrapper);

    console.log("I am showing");
});

// center
const center = document.createElement("div");
center.style.position = "absolute";
center.style.top = "20%";
center.style.left = "50%";
center.style.width = "100px";
```

```
    center.style.height = "100px";
    center.style.backgroundColor = "#d3a73f";
    center.style.borderRadius = "50%";
    center.style.transform = "translate(-50%, -50%)";
    container.appendChild(center);
}
```

12. Visualisation summary:

For the visualisation, I wanted to render the petals into a full flower. I started by creating a center, and then used it as an “anchor point” to know where to locate the petals, and what I wanted them to revolve around.

I made a clamp function because the petal sizes were too varied and ended up offsetting one another. I am also scaling the petals within the clamp function, because I found them too small at the beginning.

I randomnized the order of the petals with a function because the small ones were originally showing up in a cluster.

There is a wrapper around the petals, because I found it difficult to manipulate the petals together. Now it holds everything in place and allows it to rotate cohesively.

Data like the colors of the petals and the height and width of the petals was used to visualize the flower, and it made the process more impressive and easier than if I had just written out all the data for the individual petals.