Sophie Sanchez

CART- 263 B

April 16, 2025

Report Task 7-8 Json & Arrays & visualize

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

try{

// data from the json file
const response= await fetch('data/iris.json');

// check the fetch
if(!response.ok)
{
throw new Error(`Response status: ${response.status}`);
}

const json= await response.json();
console.log(json);
// possible colors
let possibleColor= ["#5d3fd3","#a73fd3","#d33fb5","#d35d3f","#d3a73f"];

// map the possible colors
const irisesWithColors= json.map(item => {
const addRandomColor= possibleColor[Math.floor(Math.random()* possibleColor.length)];
return{
...item,
color: addRandomColor
};
});
console.log(" colors",irisesWithColors);



// new array and filtered irises
const filteredIrises = irisesWithColors.filter(
function (iris) {
return(iris.sepalWidth >=4);
}
);
console.log("irises:",filteredIrises);

// sum and average of the petal length
const sum =irisesWithColors.reduce(
function(accum,iris)
{
return(accum + iris.petalLength)
},0
);

// average of the petal length
const averagePetalLength= sum / irisesWithColors.length;


console.log(sum);
console.log(averagePetalLength);


// find the petalWidth on the irises
const petal= irisesWithColors.find(
function(iris) {
return( iris.petalWidth > 1.0)
}
);
console.log(petal);


// find object in the irises with >10
const theSome = irisesWithColors.some(
function (iris){
return( iris.petalLength > 10 );
}
);
console.log(theSome);


// find object in the irises with == 4.2
const theSome2 = irisesWithColors.some(
function (iris){
return( iris.petalLength == 4.2 );
}
);
console.log(theSome2);



// find the every object <3
const theEvery = irisesWithColors.every(
function (iris){
return(iris.petalWidth < 3);
});
console.log(theEvery);


// find the object on the irises >1.2
const theEvery2 = irisesWithColors.every(
function (iris){
return(iris.sepalWidth > 1.2);
});
console.log(theEvery2);
```

```
// toSorted irisesWithColors on petal width smalles to largest
const irisesWithColorsSorted = irisesWithColors.toSorted((a, b) => a.petalWidth - b.petalWidth);

console.log (irisesWithColorsSorted);

} catch (error){
console.error(error.message);
}

}
```

Summary and intentions for the #11 (the visualization)

- My intention was to create an engaging, interactive visualization that represents the Iris dataset through a creative abstraction of actual flowers. I chose to represent each iris specimen as a stylized flower where the physical attributes connect to the dimensions in the dataset.

Iris class: represents individual specimens where:

- Each of flower's petal size corresponds to the actual petal measurements
- Species is mapped to distinct colors (setosa: red, versicolor: green, virginica: blue)
- Scale is proportional to petal width

Iris visualization class: Manages the overall visualization:

- Organizes flowers into species-based clusters
- Handles animation and user interaction
- Provides spatial separation between different species

For the visualization I used color coding where each species has a different color palette, an interactive exploration where the user put the mouse over the flower and they get information on specific data points. The species are grouped in different regions