Source: index.js

```
/**
 * Dataset Used:
 *
https://www.kaggle.com/datasets/saurabhshahane/mu
dataset-1950-to-2019?resource=download
 * This dataset uses the Attribution 4.0
International (CC BY 4.0) license
 *
*/
function Resources() {}

/**
 * Function that updates the HTMLElements given
the id of the element
 * @typedef {{artistName: string, choice:
string}}
 */
function updateHTMLElement(artistName, choice){
    //
document.getElementById("display").innerHTML =
artistName;
    //
document.getElementById("choice").innerHTML =
choice;
    console.log(choice)
    var artistArr = display("artists.csv",
artistName, choice);
    // console.log(artistArr.length);
}

/**
 * Returns the title description of the graph
given a string ("trackName", "danceability"...)
```

Home

Classes

DanceabilityXAxis
energyXAxis
findOcc
GenresXAxis
InstrumentalnessXAxis
LoudnessXAxis
mapFilterToTrack
TracksXAxis
yAxis

Global

Resources

```
 * @typedef {{choice: string}}
 */
function filterDescription(choice){
    console.log(choice);
    if (choice == "nameOfTracks"){
        return "A list of names of the track this artist created!";
    } else if (choice == "genres"){
        return "The number of occurrences of the genre/topic for each track";
    } else if (choice == "danceability"){
        return "How danceable each track is!";
    } else if (choice == "instrumentalness"){
        return "How instrumental each track is!";
    } else if (choice == "loudness"){
        return "How loud each track is!";
    } else if (choice == "energy"){
        return "The energy level each track is!";
    }
}
/**
 * Does all the parsing of the dataset, and filters the object specified. Displays the d3 and svg graphs towards the end
 * @typedef {{csv: object, artistName: string, choice: string}}
 */
function display(csv, artistName, choice){
    artistArr = [];

    /**
     *  Handles the parsing of the data. Has functions inside of it because data can only be accessed inside the function.
     * @typedef {}
     */
    async function arr(){
        await d3.csv(csv, function(data) {
            if (data.artist_name == artistName) {
                artistArr.push(data);
            }
        });

        if (artistArr.length != 0){
            console.log(choice);
            if (choice == "nameOfTracks"){
document.getElementById("choiceTitle").innerHTML = "Name of Tracks";
            } else {
document.getElementById("choiceTitle").innerHTML = choice.charAt(0).toUpperCase() + choice.slice(1);
            }
document.getElementById("choiceDescription").inne
 = filterDescription(choice);

            myData = [];
            filter = "";
            console.log(choice)
            if (choice == "nameOfTracks"){
                filter = "track_name";
            } else if (choice == "genres"){
                filter = "topic";
            } else if (choice == "danceability"){
                filter = "danceability";
            } else if (choice == "instrumentalness"){
                filter = "instrumentalness";
            } else if (choice == "loudness"){
                filter = "loudness";
            } else if (choice == "energy"){
                filter = "energy";
            }

            myData = findOcc(artistArr, filter);

            if ((filter == "loudness") || (filter == "instrumentalness") || (filter == "danceability") || (filter == "energy")){
                myData = mapFilterToTrack(artistArr, filter);
            }

            console.log(myData);
            max = Math.max(...myData.map(o => o.occurrence));

            const width = 900;
            const height = 450;
            const margin = { top: 50, bottom: 50, left: 50, right: 50 };
            d3.select("svg").remove();
            const svg = d3.select("#chart").append('svg')
                .attr('width', width - margin.left - margin.right)
                .attr('height', height - margin.top - margin.bottom)
                .attr("viewBox", [0, 0, width, height]);

            const x = d3.scaleBand()
            .domain(d3.range(myData.length))
            .range([margin.left, width - margin.right])
            .padding(0.1)

            const y = d3.scaleLinear()
            .domain([0, max])
            .range([height - margin.bottom, margin.top])

            /**
             * @property {object}  svg      - The SVG object which handles the display of the graph at hand. It includes attributes
             */
            svg
            .append("g")
            .attr("fill", 'orange')
            .selectAll("rect")
            .data(myData.sort((a, b) => d3.descending(a.occurrence, b.occurrence)))
            .join("rect")
                .attr("x", (d, i) => x(i))
                .attr("y", d => y(d.occurrence))
                .attr('title', (d) => d.occurrence)
                .attr("class", "rect")
                .attr("height", d => y(0) - y(d.occurrence))
                .attr("width", x.bandwidth());

            if (choice == "nameOfTracks"){
                svg.append("g").call(TracksXAxis);
            } else if (choice == "genres"){
                svg.append("g").call(GenresXAxis);
            } else if (choice == "danceability"){
                svg.append("g").call(DanceabilityXAxis);
            } else if (choice == "instrumentalness"){
                svg.append("g").call(InstrumentalnessXAxis);
            } else if (choice == "loudness"){
                svg.append("g").call(LoudnessXAxis);
            } else if (choice == "energy"){
                svg.append("g").call(energyXAxis);
            }
            svg.append("g").call(yAxis);
            svg.node();

            document.getElementById("notFound").innerHTML = "";
            window.scrollBy(0, 550);

            /**
             * Formats the yAxis and the scaling of it.
             * @constructor
             * @param {object} g - The object of the xAxis.
            */
            function yAxis(g) {
                g.attr("transform", `translate(${margin.left}, 0)`)
                    .call(d3.axisLeft(y).ticks(null, myData.format))
                    .attr("font-size", '20px')
            }
            /**
             * Formats the xAxis and the scaling of it for Name of Tracks bar graph. xAxis = nameOfTrack
             * @constructor
             * @param {object} g - The object of the xAxis.
            */
            function TracksXAxis(g) {
                g.attr("transform", `translate(0,${height - margin.bottom})`)
                    .call(d3.axisBottom(x).tickFormat(i => myData[i].track_name))
                    .attr("font-size", '20px')
            }
            /**
             * Formats the xAxis and the scaling of it for Genres bar graph. xAxis = nameOfTrack
             * @constructor
             * @param {object} g - The object of the xAxis.
            */
            function GenresXAxis(g) {
                g.attr("transform", `translate(0,${height - margin.bottom})`)
                    .call(d3.axisBottom(x).tickFormat(i => myData[i].topic))
                    .attr("font-size", '20px')
            }
            /**
             * Formats the xAxis and the scaling of it for Danceability bar graph. xAxis = nameOfTrack
             * @constructor
             * @param {object} g - The object of the xAxis.
            */
            function DanceabilityXAxis(g) {
                g.attr("transform",
`translate(0,${height - margin.bottom})`)

.call(d3.axisBottom(x).tickFormat(i =>
myData[i].track_name))
                    .attr("font-size", '20px')
            }
            /**
             * Formats the xAxis and the
scaling of it for Instrumentalness bar graph.
xAxis = nameOfTrack
             * @constructor
             * @param {object} g - The object
of the xAxis.
            */
            function InstrumentalnessXAxis(g) {
                g.attr("transform",
`translate(0,${height - margin.bottom})`)

.call(d3.axisBottom(x).tickFormat(i =>
myData[i].track_name))
                    .attr("font-size", '20px')
            }
            /**
             * Formats the xAxis and the
scaling of it for Loudness bar graph. xAxis =
nameOfTrack
             * @constructor
             * @param {object} g - The object
of the xAxis.
            */
            function LoudnessXAxis(g) {
                g.attr("transform",
`translate(0,${height - margin.bottom})`)

.call(d3.axisBottom(x).tickFormat(i =>
myData[i].track_name))
```

```
                    .attr("font-size", '20px')
            }
            /**
             * Formats the xAxis and the scaling of it for energyn bar graph. xAxis = nameOfTrack
             * @constructor
             * @param {object} g - The object of the xAxis.
            */
            function energyXAxis(g) {
                g.attr("transform", `translate(0,${height - margin.bottom})`)
                    .call(d3.axisBottom(x).tickFormat(i => myData[i].track_name))
                    .attr("font-size", '20px')
            }



            /**
             * Formats the xAxis and the scaling of it for Instrumentalness bar graph. xAxis = nameOfTrack
             * @constructor
             * @param {array} arr - Array of objects of filter [{object1: ....}, {object2: ....}...]
             * @param {string} key - String of the filter. For ex: "nameOfTracks", "genres", "instrumentalness"...
            */
            function findOcc(arr, key){
                let arr2 = [];

                arr.forEach((x)=>{

                    if(arr2.some((val)=>{
```

```
return val[key] == x[key] })){
                        arr2.forEach((k)=>{
                        if(k[key] === x[key]){
                            k["occurrence"]++
                        }
                        })

                    }else{
                        let a = {}
                        a[key] = x[key]
                        a["occurrence"] = 1
                        arr2.push(a);
                    }
                })

                return arr2
            }



            /**
             * Filters the hashmap into a format like [ {track_name: value}, {track_name: value2}...]
             * @constructor
             * @param {array} arr - Array of objects of filter [{object1: ....}, {object2: ....}...]
             * @param {string} key - String of the filter. For ex: "nameOfTracks", "genres", "instrumentalness"...
            */
            function mapFilterToTrack(arr, key) {
                let arr2 = [];
                arr.forEach((x)=>{
                    let a = {};
```

```
          console.log(key)
          a["track_name"] = x.track_name;
          if (key == "loudness"){
            a["occurrence"] = parseFloat(x.loudness);
          } else if (key == "instrumentalness"){
            a["occurrence"] = parseFloat(x.instrumentalness);
          } else if (key == "danceability"){
            a[x.track_name] = x.danceability;
            a["occurrence"] = parseFloat(x.danceability);
          } else if (key == "energy"){
            a[x.track_name] = x.danceability;
            a["occurrence"] = parseFloat(x.danceability);
          }
          arr2.push(a)
        })
        console.log(arr2)

        return arr2
      }
    } else {
        document.getElementById("choiceTitle").innerHTML = "";
        document.getElementById("choiceDescription").inne = "";
        document.getElementById("notFound").innerHTML = "No Artist was found";
      d3.select("svg").remove();
```

```
        }


    }

    arr();

    console.log(artistArr.length);
    // // console.log(artistArr.length);
    // // artistArr = [{1: "test1"}, {2:
"test2"}, {3: "test3"}]
    // return artistArr;
}


```

Documentation generated by JSDoc 3.6.11 on Sun Jul 31 2022 15:22:48 GMT-0700 (Pacific Daylight Time)