

Source: index.js

Home

Classes

- DanceabilityXAxis
- energyXAxis
- findOcc
- GenresXAxis
- InstrumentalnessXAxis
- LoudnessXAxis
- mapFilterToTrack
- TracksXAxis
- yAxis

Global

- Resources

```
1.  /**
2.   * Dataset Used:
3.   *
4.   * https://www.kaggle.com/datasets/saurabhshahane/mu
5.   * dataset-1950-to-2019?resource=download
6.   * This dataset uses the Attribution 4.0
7.   * International (CC BY 4.0) license
8.   *
9.  */
10. function Resources() {}
11.
12. /**
13.  * Function that updates the HTMLElements given
14.  * the id of the element
15.  * @typedef {{artistName: string, choice:
16.  * string}}
17.  */
18. function updateHTMLElement(artistName, choice){
19.     //
20.     document.getElementById("display").innerHTML =
21.     artistName;
22.     //
23.     document.getElementById("choice").innerHTML =
24.     choice;
25.     console.log(choice)
26.     var artistArr = display("artists.csv",
27.     artistName, choice);
28.     // console.log(artistArr.length);
29. }
30.
31. /**
32.  * Returns the title description of the graph
33.  * given a string ("trackName", "danceability"...)
```

```

23.     * @typedef {{choice: string}}
24.     */
25.     function filterDescription(choice){
26.         console.log(choice);
27.         if (choice == "nameOfTracks"){
28.             return "A list of names of the track
this artist created!";
29.         } else if (choice == "genres"){
30.             return "The number of occurrences of
the genre/topic for each track";
31.         } else if (choice == "danceability"){
32.             return "How danceable each track is!";
33.         } else if (choice == "instrumentalness"){
34.             return "How instrumental each track
is!";
35.         } else if (choice == "loudness"){
36.             return "How loud each track is!";
37.         } else if (choice == "energy"){
38.             return "The energy level each track is!
";
39.         }
40.     }
41.     /**
42.     * Does all the parsing of the dataset, and
filters the object specified. Displays the d3
and svg graphs towards the end
43.     * @typedef {{csv: object, artistName: string,
choice: string}}
44.     */
45.     function display(csv, artistName, choice){
46.         artistArr = [];
47.
48.         /**
49.         * Handles the parsing of the data. Has
functions inside of it because data can only be
accessed inside the function.
50.         * @typedef {}

```

```

51.      */
52.      async function arr(){
53.          await d3.csv(csv, function(data) {
54.              if (data.artist_name == artistName)
55.          {
56.              artistArr.push(data);
57.          }
58.          });
59.
60.          if (artistArr.length != 0){
61.              console.log(choice);
62.              if (choice == "nameOfTracks"){
63.                  document.getElementById("choiceTitle").innerHTML
64.                  = "Name of Tracks";
65.              } else {
66.                  document.getElementById("choiceTitle").innerHTML
67.                  = choice.charAt(0).toUpperCase() +
68.                  choice.slice(1);
69.              }
70.
71.              document.getElementById("choiceDescription").inne
72.              = filterDescription(choice);
73.
74.              myData = [];
75.              filter = "";
76.              console.log(choice)
77.              if (choice == "nameOfTracks"){
78.                  filter = "track_name";
79.              } else if (choice == "genres"){
80.                  filter = "topic";
81.              } else if (choice ==
82.                  "danceability"){
83.                  filter = "danceability";
84.              } else if (choice ==
85.                  "instrumentalness"){

```

```

78.         filter = "instrumentalness";
79.     } else if (choice == "loudness"){
80.         filter = "loudness";
81.     } else if (choice == "energy"){
82.         filter = "energy";
83.     }
84.
85.     myData = findOcc(artistArr,
filter);
86.
87.     if ((filter == "loudness") ||
(filter == "instrumentalness") || (filter ==
"danceability") || (filter == "energy")){
88.         myData =
mapFilterToTrack(artistArr, filter);
89.     }
90.
91.     console.log(myData);
92.     max = Math.max(...myData.map(o =>
o.occurrence));
93.
94.     const width = 900;
95.     const height = 450;
96.     const margin = { top: 50, bottom:
50, left: 50, right: 50 };
97.     d3.select("svg").remove();
98.     const svg =
d3.select("#chart").append('svg')
99.         .attr('width', width -
margin.left - margin.right)
100.        .attr('height', height -
margin.top - margin.bottom)
101.        .attr("viewBox", [0, 0, width,
height]);
102.
103.     const x = d3.scaleBand()
104.        .domain(d3.range(myData.length))

```

```

105.         .range([margin.left, width -
margin.right])
106.         .padding(0.1)
107.
108.         const y = d3.scaleLinear()
109.         .domain([0, max])
110.         .range([height - margin.bottom,
margin.top])
111.
112.         /**
113.          * @property {object} svg -
The SVG object which handles the display of the
graph at hand. It includes attributes
114.          */
115.         svg
116.         .append("g")
117.         .attr("fill", 'orange')
118.         .selectAll("rect")
119.         .data(myData.sort((a, b) =>
d3.descending(a.occurrence, b.occurrence)))
120.         .join("rect")
121.         .attr("x", (d, i) => x(i))
122.         .attr("y", d =>
y(d.occurrence))
123.         .attr('title', (d) =>
d.occurrence)
124.         .attr("class", "rect")
125.         .attr("height", d => y(0) -
y(d.occurrence))
126.         .attr("width", x.bandwidth());
127.
128.         if (choice == "nameOfTracks"){
129.
svg.append("g").call(TracksXAxis);
130.         } else if (choice == "genres"){
131.
svg.append("g").call(GenresXAxis);

```

```

132.         } else if (choice ==
    "danceability"){
133.
    svg.append("g").call(DanceabilityXAxis);
134.         } else if (choice ==
    "instrumentalness"){
135.
    svg.append("g").call(InstrumentalnessXAxis);
136.         } else if (choice == "loudness"){
137.
    svg.append("g").call(LoudnessXAxis);
138.         } else if (choice == "energy"){
139.
    svg.append("g").call(energyXAxis);
140.         }
141.         svg.append("g").call(yAxis);
142.         svg.node();
143.
144.
    document.getElementById("notFound").innerHTML =
    "";
145.         window.scrollTo(0, 550);
146.
147.         /**
148.          * Formats the yAxis and the
    scaling of it.
149.          * @constructor
150.          * @param {object} g - The object
    of the xAxis.
151.          */
152.         function yAxis(g) {
153.             g.attr("transform",
    `translate(${margin.left}, 0)`)
154.
    .call(d3.axisLeft(y).ticks(null,
    myData.format))
155.
    .attr("font-size", '20px')
156.         }

```

```

157.         /**
158.          * Formats the xAxis and the
scaling of it for Name of Tracks bar graph.
xAxis = nameOfTrack
159.          * @constructor
160.          * @param {object} g - The object
of the xAxis.
161.          */
162.          function TracksXAxis(g) {
163.              g.attr("transform",
`translate(0,${height - margin.bottom})`)
164.
.call(d3.axisBottom(x).tickFormat(i =>
myData[i].track_name))
165.              .attr("font-size", '20px')
166.          }
167.          /**
168.          * Formats the xAxis and the
scaling of it for Genres bar graph. xAxis =
nameOfTrack
169.          * @constructor
170.          * @param {object} g - The object
of the xAxis.
171.          */
172.          function GenresXAxis(g) {
173.              g.attr("transform",
`translate(0,${height - margin.bottom})`)
174.
.call(d3.axisBottom(x).tickFormat(i =>
myData[i].topic))
175.              .attr("font-size", '20px')
176.          }
177.          /**
178.          * Formats the xAxis and the
scaling of it for Danceability bar graph. xAxis
= nameOfTrack
179.          * @constructor
180.          * @param {object} g - The object
of the xAxis.

```

```

181.         */
182.         function DanceabilityXAxis(g) {
183.             g.attr("transform",
184. `translate(0,${height - margin.bottom})`)
185.
186.             .call(d3.axisBottom(x).tickFormat(i =>
187. myData[i].track_name))
188.
189.             .attr("font-size", '20px')
190.         }
191.         /**
192.          * Formats the xAxis and the
193.          scaling of it for Instrumentalness bar graph.
194.          xAxis = nameOfTrack
195.
196.          * @constructor
197.          * @param {object} g - The object
198.          of the xAxis.
199.          */
200.         function InstrumentalnessXAxis(g) {
201.             g.attr("transform",
202. `translate(0,${height - margin.bottom})`)
203.
204.             .call(d3.axisBottom(x).tickFormat(i =>
205. myData[i].track_name))
206.
207.             .attr("font-size", '20px')
208.         }
209.         /**
210.          * Formats the xAxis and the
211.          scaling of it for Loudness bar graph. xAxis =
212.          nameOfTrack
213.
214.          * @constructor
215.          * @param {object} g - The object
216.          of the xAxis.
217.          */
218.         function LoudnessXAxis(g) {
219.             g.attr("transform",
220. `translate(0,${height - margin.bottom})`)
221.
222.             .call(d3.axisBottom(x).tickFormat(i =>
223. myData[i].track_name))
224.
225.             .attr("font-size", '20px')
226.         }

```



```

205.             .attr("font-size", '20px')
206.         }
207.     /**
208.         * Formats the xAxis and the
scaling of it for energyn bar graph. xAxis =
nameOfTrack
209.         * @constructor
210.         * @param {object} g - The object
of the xAxis.
211.         */
212.         function energyXAxis(g) {
213.             g.attr("transform",
`translate(0,${height - margin.bottom})`)
214.
.call(d3.axisBottom(x).tickFormat(i =>
myData[i].track_name))
215.             .attr("font-size", '20px')
216.         }
217.
218.
219.     /**
220.         * Formats the xAxis and the
scaling of it for Instrumentalness bar graph.
xAxis = nameOfTrack
221.         * @constructor
222.         * @param {array} arr - Array of
objects of filter [{object1: ....}, {object2:
....}...]
223.         * @param {string} key - String of
the filter. For ex: "nameOfTracks", "genres",
"instrumentalness"...
224.         */
225.         function findOcc(arr, key){
226.             let arr2 = [];
227.
228.             arr.forEach((x)=>{
229.
230.                 if(arr2.some((val)=>{

```

```

return val[key] == x[key] })){
231.         arr2.forEach((k)=>{
232.             if(k[key] === x[key]){
233.                 k["occurrence"]++
234.             }
235.         })
236.
237.     }else{
238.         let a = {}
239.         a[key] = x[key]
240.         a["occurrence"] = 1
241.         arr2.push(a);
242.     }
243. })
244.
245.     return arr2
246. }
247.
248.
249.     /**
250.      * Filters the hashmap into a
format like [ {track_name: value}, {track_name:
value2}...]
251.      * @constructor
252.      * @param {array} arr - Array of
objects of filter [{object1: ....}, {object2:
....}]...
253.      * @param {string} key - String of
the filter. For ex: "nameOfTracks", "genres",
"instrumentalness"...
254.      */
255.     function mapFilterToTrack(arr, key)
{
256.         let arr2 = [];
257.         arr.forEach((x)=>{
258.             let a = {};

```

```

259.             console.log(key)
260.             a["track_name"] =
x.track_name;
261.             if (key == "loudness"){
262.                 a["occurrence"] =
parseFloat(x.loudness);
263.             } else if (key ==
"instrumentalness"){
264.                 a["occurrence"] =
parseFloat(x.instrumentalness);
265.             } else if (key ==
"danceability"){
266.                 a[x.track_name] =
x.danceability;
267.                 a["occurrence"] =
parseFloat(x.danceability);
268.             } else if (key == "energy")
{
269.                 a[x.track_name] =
x.danceability;
270.                 a["occurrence"] =
parseFloat(x.danceability);
271.             }
272.             arr2.push(a)
273.         })
274.         console.log(arr2)
275.
276.         return arr2
277.     }
278.     } else {
279.
document.getElementById("choiceTitle").innerHTML
= "";
280.
document.getElementById("choiceDescription").inne
= "";
281.
document.getElementById("notFound").innerHTML =
"No Artist was found";
282.         d3.select("svg").remove();

```

```

283.         }
284.
285.
286.     }
287.
288.     arr();
289.
290.     console.log(artistArr.length);
291.     // // console.log(artistArr.length);
292.     // // artistArr = [{1: "test1"}, {2:
"test2"}, {3: "test3"}]
293.     // return artistArr;
294. }
295.
296.

```