Home

Table of Contents

General Info

This project is a series of HTML webpages that allow users to see Covid-19 data, guidance, and info, in the city of Cincinnati, OH, USA. It contains 5 different webpages: index.html (home page), symptoms.html, protectingYourself.html, whatToDoIfYouAreSick.html, and publicTransport.html. It also contains a style.css file for all of the webpages, plus an index.js file which is linked to the index.html page only.

Technologies

The technologies used in this project are:

- Bootstrap: Version 5.0.0
- jQuery: Version 3.5.1

Interaction

Each HTML page contains a column on the left showing a menu/navigation, and references to images used. The menu initially contains 3 items: 'Interactive Map', 'Guidance', and 'Public transport info'. The 'Guidance' item is collapsible and when clicked, will show 3 more items: 'Symptoms', 'Protecting yourself', and 'What to do if you feel ill'. All of these menu items contain hyperlinks to the corresponding page.

index.html page

The index page contains an interactive map of Cincinnati, showing the 53 different neighborhoods in the city. Clicking on a neighborhood area of the map will display the basic Covid-19 data such as cases and deaths in a column to the left of the map. An initially hidden collapsible row of advanced data, showing statistics such as cases by age, can be shown by clicking the 'Show advanced data' button. Clicking the 'Total' button to the left of the 'Show advanced data' button will display the data for the whole of Cincinnati.

Also initially hidden, is a form containing checkboxes, and color-coded circles showing the locations of various services such as hospitals and Covid-19 testing sites on the map. These elements can be shown by clicking the 'Show services' button. All of the circles, which contain popover information that will be displayed when clicked, are initially visible after clicking the 'Show services' button, but their visibility can be changed by using the checkbox form on the left edge of the map.

Guidance pages

All of the guidance pages contain only textual information, giving information on Covid-19 such as symptoms, and instructions on what to do in various scenarios. All of the guidance pages, except for symptoms.html, contain collapsible elements, all initially hidden, that will display text when their header is clicked.

---

publicTransport.html page

This page shows textual information on how public transport in Cincinnati has been affected by Covid-19, and what measures certain companies are taking to try and reduce the spread of the virus on public transport vehicles.

Documentation generated by JSDoc 3.6.6 on Sat Jan 23 2021 15:14:41 GMT+0100 (Central European Standard Time)

Class: Hospital

Hospital(name, address, hours, phone, googleRating, website)

Class representing a hospital, whose information is displayed in a popover.

Constructor

`new Hospital(name, address, hours, phone, googleRating, website)`

Parameters:

Name	Type	Description
`name`	String	the name.
`address`	String	the address.
`hours`	String	the opening and closing times.
`phone`	String	the phone number.
`googleRating`	String	the rating, taken from Google.
`website`	String	the hyperlink to the website.

Source: index.js, line 357

Methods

`ReturnPopoverContentHtml() → {String}`

Source: index.js, line 383

Returns:

Returns the HTML code for the content of the popover by looping through the properties of the object and addign HTML code for each one.

Type
String

ReturnPopoverTitleHtml() → {String}

Source: index.js, line 377

Returns:

Returns the HTML code for the title of the popover by inserting the name property of the object into a HTML h5 tag.

Type
String

Documentation generated by JSDoc 3.6.6 on Sat Jan 23 2021 15:53:43 GMT+0100 (Central European Standard Time)

Class: Place

Place(name, cases, hosp, deaths, recovered, casesBySex, casesByAge, caseTrendData)

Class representing a place that is being affected by Covid-19.

Constructor

`new Place(name, cases, hosp, deaths, recovered, casesBySex, casesByAge, caseTrendData)`

Create a place

Parameters:

Name	Type	Description
`name`	String	the name.
`cases`	Number	the total number of cases.
`hosp`	Number	the total number of hospitalizations.
`deaths`	Number	the total number of deaths.
`recovered`	Number	the total number of recovered cases.
`casesBySex`	Array	the total number of male cases and female cases.
`casesByAge`	Array	the total number of cases in different age groups.
`caseTrendData`	Array	the total number of cases at the end of each month.

Source: index.js, line 258

Home

Classes

Hospital
Place
Store
TestingSite

Global

advancedDataBtn
barAddresses
barHealthAndSafety
barHours
barNames
barPhones
barRatings
bars
barWebsites
br
casesByAgeData
casesBySexData
casesData
caseTrendData
checkboxes
cincinnati
deathsData
DOMContentLoaded
FormatNumberForText
hospitalAddress
hospitalHours
hospitalNames
hospitalPhones
hospitalRatings
hospitals
hospitalWebsites
hospsData
imageMapAreas
MakeWordFromPropertyName

Methods

DisplayData()

Displays all the Covid-19 data for the place on the index page.

Source: index.js, line 283

Documentation generated by JSDoc 3.6.6 on Sat Jan 23 2021 15:53:43 GMT+0100 (Central European Standard Time)

Class: Store

Store(name, address, hours, healthAndSafety, phone, googleRating, website)

Class representing a store, whose information is displayed in a popover.

Constructor

`new Store(name, address, hours, healthAndSafety, phone, googleRating, website)`

Parameters:

Name	Type	Description
`name`	String	the name.
`address`	String	the address.
`hours`	String	the opening and closing times.
`healthAndSafety`	String	the Covid-19 health and safety requirements inside the store.
`phone`	String	the phone number.
`googleRating`	String	the rating, taken from Google.
`website`	String	the hyperlink to the website.

Source: index.js, line 419

Extends

- Hospital

Methods

ReturnPopoverContentHtml() → {String}

Overrides: Hospital#ReturnPopoverContentHtml
Source: index.js, line 383

Returns:

Returns the HTML code for the content of the popover by looping through the properties of the object and addign HTML code for each one.

Type
String

ReturnPopoverTitleHtml() → {String}

Overrides: Hospital#ReturnPopoverTitleHtml
Source: index.js, line 377

Returns:

Returns the HTML code for the title of the popover by inserting the name property of the object into a HTML h5 tag.

Type
String

Documentation generated by JSDoc 3.6.6 on Sat Jan 23 2021 15:53:43 GMT+0100 (Central European Standard Time)

Class: TestingSite

TestingSite(name, address, hours, appointment, phone, googleRating, website)

Class representing a Covid-19 testing site, whose information is displayed in a popover.

Constructor

`new TestingSite(name, address, hours, appointment, phone, googleRating, website)`

Parameters:

Name	Type	Description
`name`	String	the name.
`address`	String	the address.
`hours`	String	the opening and closing times.
`appointment`	String	'yes' or 'no' depending on if the site requires an appointment to be made.
`phone`	String	the phone number.
`googleRating`	String	the rating, taken from Google.
`website`	String	the hyperlink to the website.

Source: index.js, line 400

Extends

- Hospital

Methods

ReturnPopoverContentHtml() → {String}

Overrides:	Hospital#ReturnPopoverContentHtml
Source:	index.js, line 383

Returns:

Returns the HTML code for the content of the popover by looping through the properties of the object and addign HTML code for each one.

Type
String

ReturnPopoverTitleHtml() → {String}

Overrides:	Hospital#ReturnPopoverTitleHtml
Source:	index.js, line 377

Returns:

Returns the HTML code for the title of the popover by inserting the name property of the object into a HTML h5 tag.

Type
String

Documentation generated by JSDoc 3.6.6 on Sat Jan 23 2021 15:53:43 GMT+0100 (Central European Standard Time)

Global

Members

(constant) advancedDataBtn :Element

represents the HTML button element whose id='advancedDataBtn'. Has a 'click' event listener so that when the button is clicked, it displays the collapsible SVG area containing the advanced data.

Type:

- Element

Source: index.js, line 560

(constant) barAddresses :Array.<String>

stores the addresses of bars in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 232

(constant) barHealthAndSafety :Array.<String>

stores the Covid-19 health and safety requirements of bars in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 240

(constant) barHours :Array.<String>

stores the opening and closing times of bars in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 236

(constant) barNames :Array.<String>

stores the names of bars in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 228

(constant) barPhones :Array.<String>

stores the phone numbers of bars in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 244

(constant) barRatings :Array.<String>

stores the Google ratings of bars in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 248

bars :Array.<Store>

- stores all of the bars and their information, corresponding to the bar data arrays.

Type:

- Array.<Store>

Source: index.js, line 485

(constant) barWebsites :Array.<String>

stores the website links of bars in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 252

(constant) br :Element

represents the HTML line break element whose id='bottomBr'. Necessary so that when the advanced data SVG area is hidden, there is no double line break at the bottom of the page.

Type:

- Element

Source: index.js, line 554

(constant) casesByAgeData :Array.<Array.<Number>>

stores the number of cases by age group in each neighborhood, in ascending order of age. The age groups are 0-9, 10-19, 20-29, 30-39 and so on, with the last age group being 90+. The data for each neighborhood is stored in an array of size 10.

Type:

- Array.<Array.<Number>>

Source: index.js, line 25

(constant) casesBySexData :Array.<Array.<Number>>

stores the male and female case numbers in each neighborhood as an array of size 2 (male first, then female).

Type:

- Array.<Array.<Number>>

Source: index.js, line 21

(constant) casesData :Array.<Number>

stores the number of cases in each neighborhood.

Type:

- Array.<Number>

Source: index.js, line 5

(constant) caseTrendData :Array.<Array.<Number>>

stores the total number of cases in each neighborhood at the end of each month for March-November 2020. The data for each neighborhood is stored in an array of size 9.

Type:

- Array.<Array.<Number>>

Source: index.js, line 84

(constant) checkboxes :Array.<Element>

stores all the CSS class .form-check-input elements, which represent the checkboxes on the index.html page.

Type:

- Array.<Element>

Source: index.js, line 519

(constant) cincinnati :Place

stores the name and Covid-19 data of the city of Cincinnati.

Type:

- Place

Source: index.js, line 434

(constant) deathsData :Array.<Number>

stores the number of deaths in each neighborhood.

Type:

- Array.<Number>

Source: index.js, line 13

DOMContentLoaded :function

- initalizes the popovers on the index.html page. Code taken from Bootstrap.

Type:

- function

Source: index.js, line 575

(constant) hospitalAddress :Array.<String>

stores the addresses of hospitals in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 148

(constant) hospitalHours :Array.<String>

stores the opening and closing times of hospitals in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 152

(constant) hospitalNames :Array.<String>

stores the names of hospitals in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 144

(constant) hospitalPhones :Array.<String>

stores the phone numbers of hospitals in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 156

(constant) hospitalRatings :Array.<String>

stores the Google ratings of hospitals in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 160

hospitals :Array.<Hospital>

- stores all of the hospitals and their information, corresponding to the hospital data arrays.

Type:

- Array.<Hospital>

Source: index.js, line 455

(constant) hospitalWebsites :Array.<String>

stores the website links of hospitals in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 164

(constant) hospsData :Array.<Number>

stores the number of hospitalizations in each neighborhood.

Type:

- Array.<Number>

Source: index.js, line 9

(constant) imageMapAreas :Array.<Element>

stores the HTML area elements in the index page, so that an event listener, which will display the data of the corresponding neighborhood, can be added to each one.

Type:

- Array.<Element>

Source: index.js, line 447

(constant) mapCircles :Array.<Element>

stores all the CSS class .map-circle elements, so that various event listeners can be added to each one.

Type:

- Array.<Element>

Source: index.js, line 501

(constant) neighborhoodNames :Array.<String>

stores the string values of the names of the neighborhoods in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 1

neighborhoods :Array.<Place>

- stores all of the neighborhoods in Cincinnati, and their Covid-19 information, corresponding to the Covid-19 data arrays.

Type:

- Array.<Place>

Source: index.js, line 438

(constant) recoveredData :Array.<Number>

stores the number of recovered cases in each neighborhood.

Type:

- Array.<Number>

Source: index.js, line 17

(constant) showServicesBtn1 :Element

represents the HTML button element whose id='showServicesBtn1', shown on the index.html page as a button whose inner text is 'Show'. When clicked, it will read the 'checked' property of all the checkboxes, and subsequently show the HTML elements that are linked to the 'checked' checkboxes.

Type:

- Element

Source: index.js, line 514

(constant) showServicesBtn2 :Element

represents the HTML button element whose id='showServicesBtn2', shown on the index.html page as a button whose inner text is 'Show services'. An event listener is added so that, when clicked, it will toggle the visibility of the form and SVG area over the image map, and subsequently display the popover buttons corresponding to each hospital, testing site, store, and bar.

Type:

- Element

Source: index.js, line 529

(constant) storeAddresses :Array.<String>

stores the addresses of stores in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 203

(constant) storeHealthAndSafety :Array.<String>

stores the Covid-19 health and safety requirements of stores in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 211

(constant) storeHours :Array.<String>

stores the opening and closing times of stores in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 207

(constant) storeNames :Array.<String>

stores the names of stores in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 199

(constant) storePhones :Array.<String>

stores the phone numbers of stores in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 215

(constant) storeRatings :Array.<String>

stores the Google ratings of stores in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 219

stores :Array.<Store>

- stores all of the stores and their information, corresponding to the store data arrays.

Type:

- Array.<Store>

Source: index.js, line 475

(constant) storeWebsites :Array.<String>

stores the website links of stores in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 223

(constant) testingSiteAddresses :Array.<String>

stores the addresses of Covid-19 testing sites in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 173

(constant) testingSiteAppointments :Array.<String>

stores 'Yes' or 'No' for each Covid-19 testing site in Cincinnati, depending on whether or not a Covid-19 test must be booked at given testing site.

Type:

- Array.<String>

Source: index.js, line 181

(constant) testingSiteHours :Array.<String>

stores the opening and closing times of Covid-19 testing sites in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 177

(constant) testingSiteNames :Array.<String>

stores the names of Covid-19 testing sites in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 169

(constant) testingSitePhones :Array.<String>

stores the phone numbers of Covid-19 testing sites in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 186

(constant) testingSiteRatings :Array.<String>

stores the Google ratings of Covid-19 testing sites in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 190

testingSites :Array.<TestingSite>

- stores all of the testing sites and their information, corresponding to the testing site data arrays.

Type:

- Array.<TestingSite>

Source: index.js, line 465

(constant) testingSiteWebsites :Array.<String>

stores the website links of Covid-19 testing sites in Cincinnati.

Type:

- Array.<String>

Source: index.js, line 194

(constant) totalBtn :Element

represents the HTML button element whose id='totalBtn', shown on the index.html page as a button whose inner text is 'Total'. When clicked, it displays the data for the whole of Cincinnati.

Type:

- Element

Source: index.js, line 495

Methods

FormatNumberForText(num)

Returns a number in text format. E.g., if 13290 was the input to this function, the return value would be 13,290.

Parameters:

Name	Type	Description
num	Number	the number to be formatted.

Source: index.js, line 684

MakeWordFromPropertyName(property)

Function which turns the name of an object property into an actual word. Needed for the ReturnPopoverContentHtml function in the Hospital class.

Parameters:

Name	Type	Description
property	Property	the property that needs to be turned into an actual word.

Source: index.js, line 591

Max(arr)

Returns the maximum number of a called array.

Parameters:

Name	Type	Description
arr	Array. <Number>	the array whose maximum value is to be returned.

Source: index.js, line 670

SetCheckboxElementsVisibility(checkbox, elements)

Either hides or shows the corresponding HTML elements of a HTML checkbox, depending on whether or not that HTML checkbox is checked. Does so by calling the ToggleVisibility function for each element.

Parameters:

Name	Type	Description
checkbox	Element	the HTML checkbox whose corresponding HTML elements are called in the other parameter.
elements	Array. <Element>	an array of the corresponding HTML elements to the HTML checkbox.

Source: index.js, line 642

SetObjectPopovers(className, objectArray)

Adds content to the HTML popovers specified by the parameters of the function. Does so by calling the ReturnPopoverContentHtml and ReturnPopoverTitleHtml functions on each object in the called array.

Parameters:

Name	Type	Description
className	String	the CSS class name of the required popovers.
objectArray	Array. <Object>	the array storing the objects whose content is to be displayed on the required HTML popovers.

Source: index.js, line 627

ToggleVisibility(element)

Toggles the visibility of a called HTML element.

Parameters:

Name	Type	Description

Name	Type	Description
`element`	Element	the element whose visibility will be toggled.

Source: index.js, line 657

Documentation generated by JSDoc 3.6.6 on Sat Jan 23 2021 15:53:43 GMT+0100 (Central European Standard Time)

Source: index.js

```
1.  /**
2.   * @const {Array<String>} neighborhoodNames -
3.   */
4.  const neighborhoodNames = ["Avondale", "Bond H
5.  /**
6.   * @const {Array<Number>} casesData - stores t
7.   */
8.  const casesData = [364, 288, 7, 42, 74, 307, 1
9.  /**
10.  * @const {Array<Number>} hospsData - stores t
11.  */
12. const hospsData = [50, 25, 0, 2, 6, 9, 18, 41,
13. /**
14.  * @const {Array<Number>} deathsData - stores
15.  */
16. const deathsData = [6, 1, 0, 0, 0, 2, 8, 8, 0,
17. /**
18.  * @const {Array<Number>} recoveredData - stor
19.  */
20. const recoveredData = [203, 136, 1, 10, 41, 90
21. /**
22.  * @const {Array<Array<Number>>} casesBySexDat
23.  */
24. const casesBySexData = [[167, 197], [125, 163]
25. /**
26.  * @const {Array<Array<Number>>} casesByAgeDat
27.  * and so on, with the last age group being 90
28.  */
29. const casesByAgeData = [
```

```
30.     [12, 25, 80, 56, 44, 62, 49, 27, 6, 3],
31.     [9, 23, 53, 47, 38, 36, 57, 20, 4, 1],
32.     [0, 2, 0, 0, 0, 0, 4, 1, 0, 0],
33.     [1, 0, 8, 16, 8, 5, 3, 1, 0, 0],
34.     [6, 5, 10, 15, 17, 8, 9, 2, 2, 0],
35.     [4, 8, 146, 76, 23, 27, 16, 5, 1, 1],
36.     [2, 19, 68, 23, 11, 16, 18, 16, 8, 2],
37.     [11, 34, 85, 88, 60, 89, 52, 18, 20, 20],
38.     [1, 2, 18, 9, 3, 3, 4, 0, 0, 0],
39.     [0, 54, 132, 7, 8, 2, 5, 3, 0, 0],
40.     [2, 164, 317, 12, 2, 5, 3, 5, 2, 0],
41.     [0, 2, 14, 6, 1, 4, 3, 2, 2, 0],
42.     [17, 38, 93, 113, 80, 60, 35, 8, 4, 0],
43.     [1, 7, 32, 20, 17, 15, 7, 8, 13, 5],
44.     [10, 4, 12, 8, 8, 11, 13, 7, 2, 0],
45.     [3, 1, 0, 3, 0, 0, 2, 0, 0, 0],
46.     [6, 55, 70, 45, 24, 30, 17, 5, 2, 2],
47.     [3, 4, 22, 10, 3, 2, 3, 2, 0, 0],
48.     [3, 12, 38, 39, 33, 13, 16, 13, 13, 11],
49.     [0, 105, 188, 15, 6, 5, 1, 2, 3, 0],
50.     [9, 50, 155, 67, 29, 56, 28, 14, 12, 1],
51.     [2, 13, 31, 37, 22, 25, 11, 9, 0, 3],
52.     [0, 0, 2, 3, 4, 1, 2, 0, 0, 0],
53.     [1, 1, 4, 10, 4, 3, 1, 0, 0, 0],
54.     [7, 20, 89, 71, 37, 34, 28, 12, 15, 17],
55.     [3, 1, 12, 12, 2, 3, 4, 0, 1, 0],
56.     [0, 0, 28, 5, 5, 5, 4, 2, 2, 1],
57.     [19, 37, 78, 58, 48, 49, 34, 6, 1, 0],
58.     [5, 29, 79, 44, 20, 6, 9, 1, 2, 1],
59.     [4, 24, 41, 24, 18, 27, 15, 2, 0, 0],
60.     [6, 25, 70, 65, 49, 32, 31, 12, 2, 1],
61.     [3, 7, 19, 5, 10, 6, 5, 6, 1, 0],
62.     [0, 8, 4, 11, 11, 4, 1, 2, 0, 0],
63.     [5, 6, 43, 51, 20, 22, 17, 8, 4, 0],
```

mapCircles
Max
neighborhoodNames
neighborhoods
recoveredData
SetCheckboxElementsVisibilit
SetObjectPopovers
showServicesBtn1
showServicesBtn2
storeAddresses
storeHealthAndSafety
storeHours
storeNames
storePhones
storeRatings
stores
storeWebsites
testingSiteAddresses
testingSiteAppointments
testingSiteHours
testingSiteNames
testingSitePhones
testingSiteRatings
testingSites
testingSiteWebsites
ToggleVisibility
totalBtn

```
64.     [8, 11, 232, 98, 27, 39, 24, 12, 10, 1],
65.     [3, 21, 10, 7, 3, 2, 2, 2, 0, 0],
66.     [5, 12, 81, 70, 13, 22, 10, 3, 0, 0],
67.     [2, 6, 12, 7, 4, 7, 5, 1, 0, 0],
68.     [1, 1, 27, 9, 5, 4, 0, 0, 0, 0],
69.     [5, 17, 61, 31, 32, 40, 27, 16, 13, 5],
70.     [0, 0, 0, 1, 1, 4, 3, 0, 0, 0],
71.     [2, 1, 9, 9, 8, 5, 11, 1, 0, 0],
72.     [7, 24, 43, 38, 28, 30, 48, 18, 9, 3],
73.     [5, 7, 18, 18, 10, 22, 16, 3, 1, 0],
74.     [0, 2, 3, 5, 0, 0, 5, 0, 0, 0],
75.     [0, 4, 3, 3, 4, 5, 3, 4, 2, 0],
76.     [0, 0, 7, 10, 14, 10, 9, 7, 1, 0],
77.     [2, 10, 47, 52, 25, 18, 27, 12, 7, 0],
78.     [5, 14, 50, 38, 13, 19, 18, 8, 2, 1],
79.     [28, 69, 121, 140, 92, 106, 58, 48, 13, 1]
80.     [39, 110, 249, 221, 157, 155, 109, 59, 49,
81.     [5, 10, 30, 23, 5, 14, 7, 3, 1, 0],
82.     [1, 8, 14, 11, 10, 11, 2, 1, 1, 0]
83. ];
84. /**
85.  * @const {Array<Array<Number>>} caseTrendData
86.  * is stored in an array of size 9.
87.  */
88. const caseTrendData = [
89.     [2, 23, 83, 136, 192, 227, 284, 304, 343],
90.     [2, 17, 64, 92, 140, 199, 223, 257, 288],
91.     [0, 1, 3, 3, 3, 5, 5, 7, 7],
92.     [0, 0, 12, 20, 25, 31, 37, 40, 42],
93.     [3, 3, 14, 23, 37, 51, 61, 70, 74],
94.     [2, 20, 72, 133, 181, 217, 254, 280, 307],
95.     [2, 15, 29, 47, 76, 100, 112, 149, 183],
96.     [2, 12, 77, 141, 203, 269, 312, 374, 477],
97.     [0, 7, 10, 16, 20, 22, 28, 33, 40],
```

```
98.     [1, 11, 23, 37, 78, 102, 125, 150, 211],
99.     [3, 31, 69, 103, 167, 230, 292, 364, 512],
100.    [1, 5, 12, 16, 22, 25, 28, 30, 34],
101.    [2, 48, 83, 110, 156, 219, 277, 333, 448],
102.    [0, 12, 21, 33, 54, 78, 101, 112, 127],
103.    [0, 3, 10, 22, 39, 51, 60, 69, 75],
104.    [0, 0, 2, 2, 5, 6, 8, 8, 9],
105.    [2, 19, 44, 81, 103, 123, 158, 197, 256],
106.    [0, 5, 10, 13, 20, 26, 31, 38, 49],
107.    [1, 11, 27, 44, 62, 100, 123, 148, 191],
108.    [3, 24, 68, 112, 173, 222, 257, 299, 325],
109.    [3, 50, 113, 189, 264, 300, 331, 369, 421]
110.    [2, 9, 43, 60, 85, 104, 119, 130, 153],
111.    [0, 1, 3, 4, 4, 7, 7, 9, 11],
112.    [1, 4, 8, 10, 13, 16, 18, 20, 24],
113.    [2, 37, 81, 105, 144, 192, 230, 270, 332],
114.    [0, 6, 10, 15, 22, 28, 33, 35, 38],
115.    [0, 9, 14, 19, 25, 30, 34, 40, 52],
116.    [2, 20, 51, 83, 131, 174, 221, 268, 331],
117.    [1, 11, 25, 54, 82, 111, 130, 150, 196],
118.    [1, 9, 17, 31, 48, 69, 93, 110, 156],
119.    [3, 21, 40, 63, 92, 134, 172, 220, 295],
120.    [0, 2, 8, 17, 28, 33, 40, 49, 62],
121.    [0, 0, 6, 9, 14, 20, 25, 33, 41],
122.    [2, 11, 27, 40, 67, 98, 120, 147, 176],
123.    [7, 31, 66, 100, 171, 236, 297, 380, 463],
124.    [0, 1, 3, 8, 14, 20, 26, 38, 50],
125.    [4, 16, 23, 46, 74, 112, 137, 180, 216],
126.    [1, 2, 5, 9, 14, 20, 24, 31, 44],
127.    [0, 0, 3, 7, 12, 18, 22, 32, 47],
128.    [3, 9, 20, 47, 78, 123, 157, 200, 247],
129.    [0, 0, 0, 1, 1, 1, 3, 3, 9],
130.    [1, 1, 4, 9, 15, 20, 26, 35, 46],
131.    [3, 7, 21, 53, 92, 133, 180, 212, 248],
```

```
132.     [0, 2, 8, 16, 25, 37, 49, 74, 100],
133.     [1, 1, 1, 3, 3, 3, 6, 9, 15],
134.     [0, 1, 4, 6, 10, 13, 18, 23, 28],
135.     [1, 3, 5, 11, 18, 26, 32, 40, 58],
136.     [1, 10, 23, 37, 62, 90, 119, 158, 200],
137.     [0, 3, 17, 40, 52, 69, 93, 137, 171],
138.     [10, 33, 64, 100, 159, 241, 332, 490, 677]
139.     [30, 91, 167, 252, 363, 499, 651, 842, 117
140.     [2, 4, 9, 20, 32, 45, 59, 78, 98],
141.     [0, 3, 5, 10, 18, 25, 30, 42, 59]
142. ];
143.
144. /**
145.  * @const {Array<String>} hospitalNames - stor
146.  */
147. const hospitalNames = ["University of Cincinna
148. /**
149.  * @const {Array<String>} hospitalAddress - st
150.  */
151. const hospitalAddresses = ["3200 Burnet Ave, C
152. /**
153.  * @const {Array<String>} hospitalHours - stor
154.  */
155. const hospitalHours = ["24 hours", "24 hours",
156. /**
157.  * @const {Array<String>} hospitalPhones - sto
158.  */
159. const hospitalPhones = ["+1 513-475-8000", "+1
160. /**
161.  * @const {Array<String>} hospitalRatings - st
162.  */
163. const hospitalRatings = ["3.9/5", "3.3/5", "3.
164. /**
165.  * @const {Array<String>} hospitalWebsites - s
```

```
166.  */
167. const hospitalWebsites = ["https://www.uchealt
168.
169. /**
170.  * @const {Array<String>} testingSiteNames - s
171.  */
172. const testingSiteNames = ["TriHealth Priority
173. /**
174.  * @const {Array<String>} testingSiteAddresses
175.  */
176. const testingSiteAddresses = ["6139 Glenway Av
177. /**
178.  * @const {Array<String>} testingSiteHours - s
179.  */
180. const testingSiteHours = ["8am-7:30pm", "8:30a
181. /**
182.  * @const {Array<String>} testingSiteAppointme
183.  * at given testing site.
184.  */
185. const testingSiteAppointments = ["Yes", "Yes",
186. /**
187.  * @const {Array<String>} testingSitePhones -
188.  */
189. const testingSitePhones = ["+1 513-346-3399",
190. /**
191.  * @const {Array<String>} testingSiteRatings -
192.  */
193. const testingSiteRatings = ["4.0/5", "3.6/5",
194. /**
195.  * @const {Array<String>} testingSiteWebsites
196.  */
197. const testingSiteWebsites = ["http://www.trihe
198.
199. /**
```

```
200.  * @const {Array<String>} storeNames - stores
201.  */
202. const storeNames = ["Target", "Target", "Targe
203. /**
204.  * @const {Array<String>} storeAddresses - sto
205.  */
206. const storeAddresses = ["6150 Glenway Ave, Cin
207. /**
208.  * @const {Array<String>} storeHours - stores
209.  */
210. const storeHours = ["8am-10pm", "8am-10pm", "8
211. /**
212.  * @const {Array<String>} storeHealthAndSafety
213.  */
214. const storeHealthAndSafety = ["Mask required,
215. /**
216.  * @const {Array<String>} storePhones - stores
217.  */
218. const storePhones = ["+1 513-719-1076", "+1 51
219. /**
220.  * @const {Array<String>} storeRatings - store
221.  */
222. const storeRatings = ["4.0/5", "4.1/5", "4.2/5
223. /**
224.  * @const {Array<String>} storeWebsites - stor
225.  */
226. const storeWebsites = ["https://www.target.com
227.
228. /**
229.  * @const {Array<String>} barNames - stores th
230.  */
231. const barNames = ["JerZees Pub and Grub", "Sun
232. /**
233.  * @const {Array<String>} barAddresses - store
```

```
234.  */
235. const barAddresses = ["708 Monmouth St, Newpor
236. /**
237.  * @const {Array<String>} barHours - stores th
238.  */
239. const barHours = ["11am-2am", "4pm-11pm", "3pm
240. /**
241.  * @const {Array<String>} barHealthAndSafety -
242.  */
243. const barHealthAndSafety = ["Mask required, st
244. /**
245.  * @const {Array<String>} barPhones - stores t
246.  */
247. const barPhones = ["+1 859-491-3500", "+1 513-
248. /**
249.  * @const {Array<String>} barRatings - stores
250.  */
251. const barRatings = ["4.4/5", "4.7/5", "4.5/5",
252. /**
253.  * @const {Array<String>} barWebsites - stores
254.  */
255. const barWebsites = ["https://www.jerzeespub.c
256.
257. /** Class representing a place that is being a
258. class Place{
259.     /**
260.      * Create a place
261.      * @param {String} name - the name.
262.      * @param {Number} cases - the total numbe
263.      * @param {Number} hosp - the total number
264.      * @param {Number} deaths - the total numb
265.      * @param {Number} recovered - the total n
266.      * @param {Array} casesBySex - the total n
267.      * @param {Array} casesByAge - the total n
```

```
268.          * @param {Array} caseTrendData - the tota
269.          */
270.         constructor(name, cases, hosp, deaths, rec
271.             this.name = name;
272.             this.cases = cases;
273.             this.hosp = hosp;
274.             this.deaths = deaths;
275.             this.recovered = recovered;
276.             this.casesBySex = casesBySex;
277.             this.casesByAge = casesByAge;
278.             this.caseTrendData = caseTrendData;
279.         }
280.         /**
281.          * Displays all the Covid-19 data for the
282.          */
283.         DisplayData(){
284.             const levelsDataDisplayer = document.g
285.             if (this.name != "Cincinnati"){
286.                 document.getElementById("dataTitle
287.                 if (this.cases >= 500){
288.                     levelsDataDisplayer.style.colo
289.                     levelsDataDisplayer.innerHTML
290.                 }
291.                 else if (this.cases < 500 && this.
292.                     levelsDataDisplayer.style.colo
293.                     levelsDataDisplayer.innerHTML
294.                 }
295.                 else{
296.                     levelsDataDisplayer.style.colo
297.                     levelsDataDisplayer.innerHTML
298.                 }
299.             }
300.             else{
301.                 document.getElementById("dataTitle
```

```
302.             levelsDataDisplayer.innerHTML = ""
303.         }
304.         d3.select("body")
305.         .selectAll(".data")
306.         .data([this.cases, this.hosp, this.dea
307.         .text(function (data) {
308.             return FormatNumberForText(data);
309.         });
310.         const percentage = ((100 * this.casesB
311.         const circumference = (2 * 25 * Math.P
312.         document.getElementById("overlapCircle
313.         document.getElementById("maleCasesText
314.         document.getElementById("femaleCasesTe
315.         let highestNum = Max(this.casesByAge);
316.         var barHeight = 0;
317.         for (let i = 0; i < 10; i++) {
318.             var currentBar = document.getEleme
319.             var currentBarNum = document.getEl
320.             if (this.casesByAge[i] != highestN
321.                 barHeight = this.casesByAge[i]
322.             }
323.             else {
324.                 barHeight = 80;
325.             }
326.             currentBar.setAttribute("height",
327.             currentBarNum.innerHTML = FormatNu
328.             var barYCoord = 100 - barHeight;
329.             currentBar.setAttribute("y", (barY
330.             currentBarNum.setAttribute("y", (b
331.             if (this.casesByAge[i] >= 1000){
332.                 currentBarNum.setAttribute("x"
333.             }
334.             else if (this.casesByAge[i] >= 100
335.                 currentBarNum.setAttribute("x"
```

```
336.             }
337.             else if (this.casesByAge[i] >= 10
338.                 currentBarNum.setAttribute("x"
339.             }
340.             else{
341.                 currentBarNum.setAttribute("x"
342.             }
343.         }
344.         for (let i = 1; i <= 5; i++) {
345.             document.getElementById("lineNum"
346.         }
347.         var lineYCoords = new Array(8);
348.         const multiplier = 90 / this.caseTrend
349.         for (let i = 0; i < lineYCoords.length
350.             lineYCoords[i] = 100 - this.caseTr
351.         }
352.         document.getElementById("lineGraph").s
353.         "40, " + lineYCoords[0].toString() + "
354.     }
355. }
356. /**Class representing a hospital, whose inform
357. class Hospital{
358.     /**
359.      * @param {String} name - the name.
360.      * @param {String} address - the address.
361.      * @param {String} hours - the opening and
362.      * @param {String} phone - the phone numbe
363.      * @param {String} googleRating - the rati
364.      * @param {String} website - the hyperlink
365.      */
366.     constructor(name, address, hours, phone, g
367.         this.name = name;
368.         this.address = address;
369.         this.hours = hours;
```

```
370.         this.phone = phone;
371.         this.googleRating = googleRating;
372.         this.website = website;
373.     }
374.     /**
375.      * @return {String} Returns the HTML code
376.      */
377.     ReturnPopoverTitleHtml(){
378.         return "<h5>" + this.name + "</h5>"
379.     }
380.     /**
381.      * @return {String} Returns the HTML code
382.      */
383.     ReturnPopoverContentHtml(){
384.         var str = "";
385.         for (var property in this){
386.             if (property.toString() != "name")
387.                 if (property.toString() != "we
388.                     str += "<p><strong>" + Mak
389.                 }
390.             }
391.         }
392.         str += "<a href=\"" + this.website + "
393.         return str;
394.     }
395. }
396. /**
397.  * Class representing a Covid-19 testing site,
398.  * @extends Hospital
399.  */
400. class TestingSite extends Hospital{
401.     /**
402.      * @param {String} name - the name.
403.      * @param {String} address - the address.
```

```
404.      * @param {String} hours - the opening and
405.      * @param {String} appointment - 'yes' or
406.      * @param {String} phone - the phone numbe
407.      * @param {String} googleRating - the rati
408.      * @param {String} website - the hyperlink
409.      */
410.     constructor(name, address, hours, appointm
411.         super(name, address, hours, phone, goo
412.         this.appointment = appointment;
413.     }
414. }
415. /**
416.  * Class representing a store, whose informati
417.  * @extends Hospital
418.  */
419. class Store extends Hospital{
420.     /**
421.      * @param {String} name - the name.
422.      * @param {String} address - the address.
423.      * @param {String} hours - the opening and
424.      * @param {String} healthAndSafety - the C
425.      * @param {String} phone - the phone numbe
426.      * @param {String} googleRating - the rati
427.      * @param {String} website - the hyperlink
428.      */
429.     constructor(name, address, hours, healthAn
430.         super(name, address, hours, phone, goo
431.         this.healthAndSafety = healthAndSafety
432.     }
433. }
434. /**
435.  * @const {Place} cincinnati - stores the name
436.  */
437. const cincinnati = new Place("Cincinnati", 139
```

```
438.  /**
439.   * @name neighborhoods
440.   * @type {Array<Place>}
441.   * @description - stores all of the neighborho
442.   */
443.  var neighborhoods = new Array(neighborhoodName
444.  for (let i = 0; i < neighborhoods.length; i++)
445.      neighborhoods[i] = new Place(neighborhoodN
446.  }
447.  /**
448.   * @const {Array<Element>} imageMapAreas - sto
449.   * neighborhood, can be added to each one.
450.   */
451.  const imageMapAreas = document.querySelectorAl
452.  for (let i = 0; i < imageMapAreas.length; i++)
453.      imageMapAreas[i].addEventListener("click",
454.  }
455.  /**
456.   * @name hospitals
457.   * @type {Array<Hospital>}
458.   * @description - stores all of the hospitals
459.   */
460.  var hospitals = new Array(hospitalNames.length
461.  for (let i = 0; i < hospitals.length; i++) {
462.      hospitals[i] = new Hospital(hospitalNames[
463.  }
464.  SetObjectPopovers(".hospital-circle", hospital
465.  /**
466.   * @name testingSites
467.   * @type {Array<TestingSite>}
468.   * @description - stores all of the testing si
469.   */
470.  var testingSites = new Array(testingSiteNames.
471.  for (let i = 0; i < testingSites.length; i++)
```

```
472.     testingSites[i] = new TestingSite(testingS
473. }
474. SetObjectPopovers(".testing-site-circle", test
475. /**
476.  * @name stores
477.  * @type {Array<Store>}
478.  * @description - stores all of the stores and
479.  */
480. var stores = new Array(storeNames.length);
481. for (let i = 0; i < stores.length; i++) {
482.     stores[i] = new Store(storeNames[i], store
483. }
484. SetObjectPopovers(".store-circle", stores);
485. /**
486.  * @name bars
487.  * @type {Array<Store>}
488.  * @description - stores all of the bars and t
489.  */
490. var bars = new Array(barNames.length);
491. for (let i = 0; i < stores.length; i++) {
492.     bars[i] = new Store(barNames[i], barAddres
493. }
494. SetObjectPopovers(".bar-circle", bars);
495. /**
496.  * @const {Element} totalBtn - represents the
497.  * When clicked, it displays the data for the
498.  */
499. const totalBtn = document.getElementById("tota
500. totalBtn.addEventListener("click", function(){
501. /**
502.  * @const {Array<Element>} mapCircles - stores
503.  */
504. const mapCircles = document.querySelectorAll("
505. mapCircles.forEach(circle => {
```

```
506.     circle.setAttribute("r", "5");
507.     circle.addEventListener("mouseover", funct
508.         circle.setAttribute("r", "7.5");
509.     });
510.     circle.addEventListener("mouseout", functi
511.         circle.setAttribute("r", "5");
512.     });
513. });
514. /**
515.  * @const {Element} showServicesBtn1 - represe
516.  * 'Show'. When clicked, it will read the 'che
517.  */
518. const showServicesBtn1 = document.getElementBy
519. /**
520.  * @const {Array<Element>} checkboxes - stores
521.  */
522. const checkboxes = document.querySelectorAll("
523. showServicesBtn1.addEventListener("click", fun
524.     SetCheckboxElementsVisibility(checkboxes[0
525.     SetCheckboxElementsVisibility(checkboxes[1
526.     SetCheckboxElementsVisibility(checkboxes[2
527.     SetCheckboxElementsVisibility(checkboxes[3
528. });
529. /**
530.  * @const {Element} showServicesBtn2 - represe
531.  * 'Show services'. An event listener is added
532.  * display the popover buttons corresponding t
533.  */
534. const showServicesBtn2 = document.getElementBy
535. showServicesBtn2.addEventListener("click", fun
536.     if (this.innerHTML == "Show services"){
537.         this.innerHTML = "Hide services";
538.         mapCircles.forEach(circle =>{
539.             circle.style.visibility = "visible
```

```
540.         });
541.         checkboxes.forEach(checkbox =>{
542.             checkbox.checked = true;
543.         });
544.     }
545.     else{
546.         this.innerHTML = "Show services";
547.         mapCircles.forEach(circle =>{
548.             circle.style.visibility = "hidden"
549.         });
550.     }
551.     ToggleVisibility(document.getElementById("
552.     ToggleVisibility(document.getElementById("
553. });
554. /**
555.  * @const {Element} br - represents the HTML l
556.  * line break at the bottom of the page.
557.  */
558. const br = document.getElementById("bottomBr")
559. br.style.display = "none";
560. /**
561.  * @const {Element} advancedDataBtn - represen
562.  * it displays the collapsible SVG area contai
563.  */
564. const advancedDataBtn = document.getElementByI
565. advancedDataBtn.addEventListener("click", func
566.     if (this.innerHTML == "Show advanced data"
567.         this.innerHTML = "Hide advanced data";
568.         br.style.display = "initial";
569.     }
570.     else{
571.         this.innerHTML = "Show advanced data";
572.         br.style.display = "none";
573.     }
```

```
574. });
575. /**
576.  * @name DOMContentLoaded
577.  * @type {Function}
578.  * @description - initalizes the popovers on t
579.  */
580. document.addEventListener("DOMContentLoaded",
581.     const popoverTriggerList = [].slice.call(d
582.     const popoverList = popoverTriggerList.map
583.         return new bootstrap.Popover(popoverTr
584.     });
585. });
586. /**
587.  *
588.  * Function which turns the name of an object
589.  * @param {Property} property - the property t
590.  */
591. function MakeWordFromPropertyName(property){
592.     var propertyStr = property.toString();
593.     var isMoreThanOneWord = false;
594.     for (let i = 0; i < propertyStr.length; i+
595.         if (propertyStr.charCodeAt(i) >= 65 &&
596.             isMoreThanOneWord = true;
597.             break;
598.         }
599.     }
600.     if (isMoreThanOneWord){
601.         var str = "";
602.         for (let i = 0; i < propertyStr.length
603.             if (propertyStr.charCodeAt(i) >= 6
604.                 str += " " + propertyStr[i].to
605.             }
606.             else if (i == 0){
607.                 str += propertyStr[i].toUpperC
```

```
608.             }
609.             else{
610.                 str += propertyStr[i];
611.             }
612.         }
613.         return str;
614.     }
615.     else{
616.         return propertyStr[0].toUpperCase() +
617.     }
618. }
619.
620. /**
621.  *
622.  * Adds content to the HTML popovers specified
623.  * functions on each object in the called arra
624.  * @param {String} className - the CSS class n
625.  * @param {Array<Object>} objectArray - the ar
626.  */
627. function SetObjectPopovers(className, objectAr
628.     const popovers = document.querySelectorAll
629.     for (let i = 0; i < popovers.length; i++)
630.         popovers[i].setAttribute("data-bs-cont
631.         popovers[i].setAttribute("title", obje
632.     }
633. }
634.
635. /**
636.  *
637.  * Either hides or shows the corresponding HTM
638.  * for each element.
639.  * @param {Element} checkbox - the HTML checkb
640.  * @param {Array<Element>} elements - an array
641.  */
```

```
642. function SetCheckboxElementsVisibility(checkbo
643.     elements.forEach(element => {
644.         element.style.visibility = "hidden";
645.     });
646.     if (checkbox.checked == true){
647.         elements.forEach(element => {
648.             ToggleVisibility(element);
649.         });
650.     }
651. }
652. 
653. /**
654.  * Toggles the visibility of a called HTML ele
655.  * @param {Element} element - the element whos
656.  */
657. function ToggleVisibility(element){
658.     if (element.style.visibility == "visible")
659.         element.style.visibility = "hidden";
660.     }
661.     else{
662.         element.style.visibility = "visible";
663.     }
664. }
665. 
666. /**
667.  * Returns the maximum number of a called arra
668.  * @param {Array<Number>} arr - the array whos
669.  */
670. function Max(arr){
671.     var max = 0;
672.     for (let i = 0; i < arr.length; i++){
673.             if (arr[i] > max){
674.                 max = arr[i];
675.             }
```

```
676.     }
677.     return max;
678. }
679.
680. /**
681.  * Returns a number in text format. E.g., if 1
682.  * @param {Number} num - the number to be form
683.  */
684. function FormatNumberForText(num){
685.     if (num < 1000){
686.         return num.toString();
687.     }
688.     else{
689.         const indexForComma = num.toString().l
690.         return num.toString().substr(0, indexF
691.     }
692. }
```

Documentation generated by JSDoc 3.6.6 on Sat Jan 23 2021 15:53:43 GMT+0100 (Central European Standard Time)