JS Tutorial

Agenda

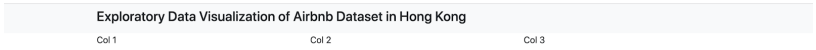
1st Stage: Application bootstrap

Topics to cover:

1. How to create a web page?
2. How to create a basic layout for the web page?

Final result:

Up and running web application in the browser with layout ready to display the chars.



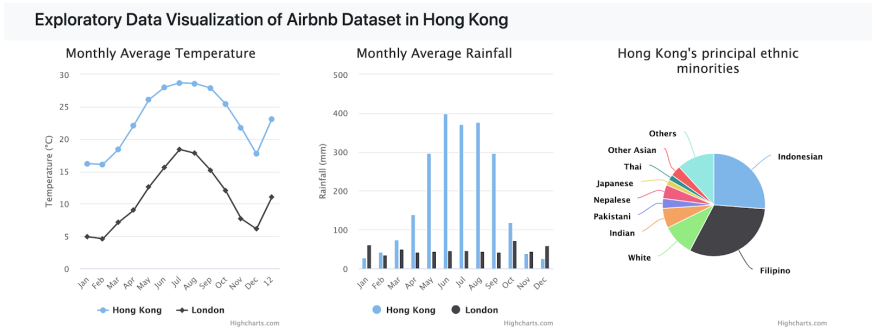
2nd Stage: Data Visualization libraries

Topics to cover:

1. How to add libraries for the Data Visualization project for the web page?
2. How to use JS Data Visualization libraries?

Final result:

We have implemented a different kind of charts using JS Data Visualization libraries.



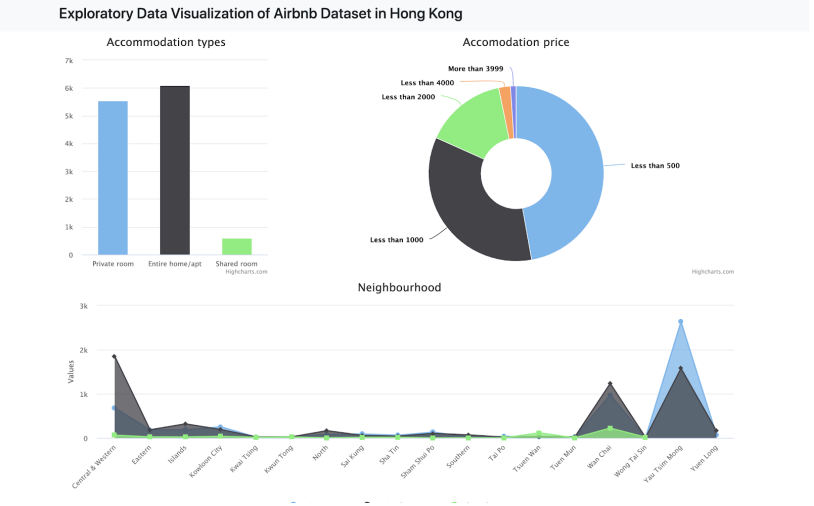
3rd Stage: Visualize data

Topics to cover:

1. How to display the data at the carts?
2. How to extract the data from the JSON collection

Final result:

Our application now can display the data.



4rh Stage: Network requests

Topics to cover:

1. How to request the data from the network?

Final result:

The application can request the data from the network.

1st Stage: Application bootstrap

In this stage, we have to:

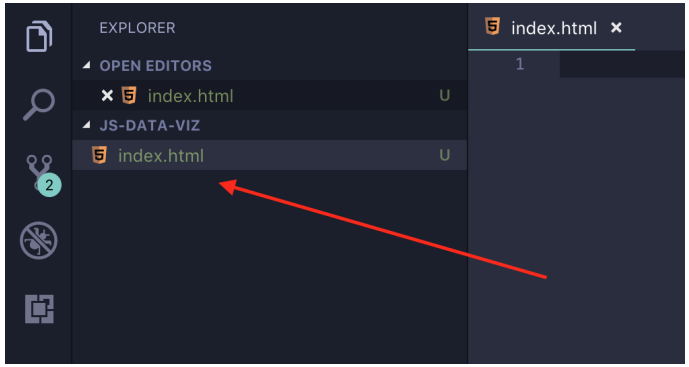
1. Bootstrap the application in the browser
2. Make it look good with some styles
3. Include the data visualization library.

By having this completed we will be able to continue to learn more about data visualization tools. So let's start.

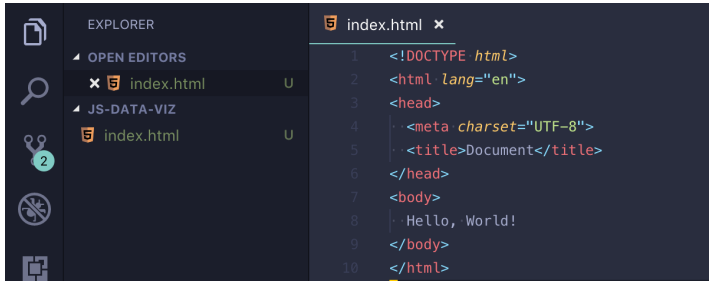
1st step

The 1st step will be to create a directory with an HTML file and run this in the browser.

1. Create a directory with a name data-viz-js
2. Create an index.html file inside of this directory



3. Now we have to add an HTML markup. Populate the index.html with the following code.



By having this done we should be able to open the HTML file in the browser and we should have the following result:

Please find the result for the 1st step by following this link:
https://github.com/Luodina/js-data-viz/tree/1st-stage/step-1
Now we are ready for the 2nd step.

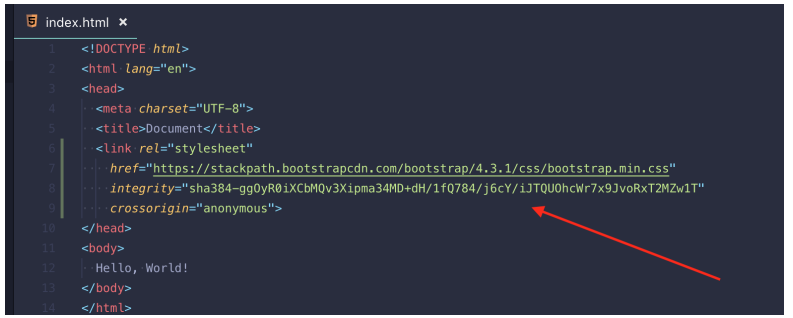
2nd Step

For the 2nd step, we should implement some layouting for the HTML page.

1. We have to include the style library. For this past the following line of code

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
```

   Like in the example below:



   You can also copy this line by following the link
   https://getbootstrap.com/docs/4.3/getting-started/introduction/
2. Now let's start with the layout. First of all, let's create a grid. Feel free to follow the documentation https://getbootstrap.com/docs/4.3/layout/grid/
   We will need 3 columns. Take a look at the following example:

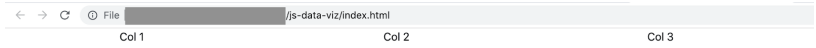
index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" ...
    crossorigin="anonymous">
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col">
                Col 1
            </div>

            <div class="col">
                Col 2
            </div>

            <div class="col">
                Col 3
            </div>
        </div>
    </div>
</body>
</html>
```

And the result will be like the following:

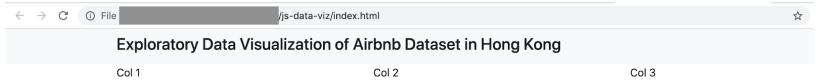


Please feel free to take a look at the source code by following this link
https://github.com/Luodina/js-data-viz/blob/1st-stage/step-1-grid/index.html

3. Now we are going to add a navbar.
   Let's take a look at the following example:

```
<body>
    <nav class="navbar navbar-light bg-light">
        <div class="container">
            <h4>Exploratory Data Visualization of Airbnb Dataset in Hong Kong</h4>
        </div>
    </nav>
```

And the result will look like:



Take a look at the documentation https://getbootstrap.com/docs/4.3/components/navbar/
Please feel free take a look at the source code
https://github.com/Luodina/js-data-viz/blob/1st-stage/step-1-navbar/index.html

And by having this done we are finished with the layout. The columns we just created we will be using to populate with charts. And now we are ready to move to the charts.

2nd Stage: Data Visualization libraries

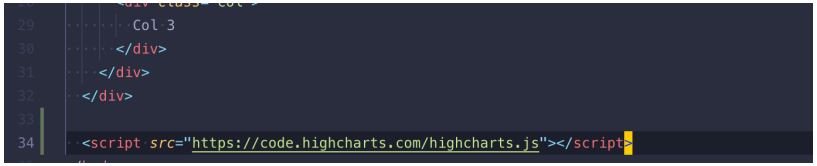
Now is the time to work with the chars. We will be using the Highcharts library, please take a look at the demos of the library https://www.highcharts.com/demo to be familiar with the possibilities of the library.

By the end of this stage, we should get familiar with how to create charts using this library. We will create the line, column, and pie chart with some abstract data, just in order to understand how to create the charts. Later on, we will be using some real data.

And now let's visualize this data.

1st step

First of all, we have to include the Highcharts library into our HTML page.
For this include the following before the closed <body> tag, like on the screenshot:



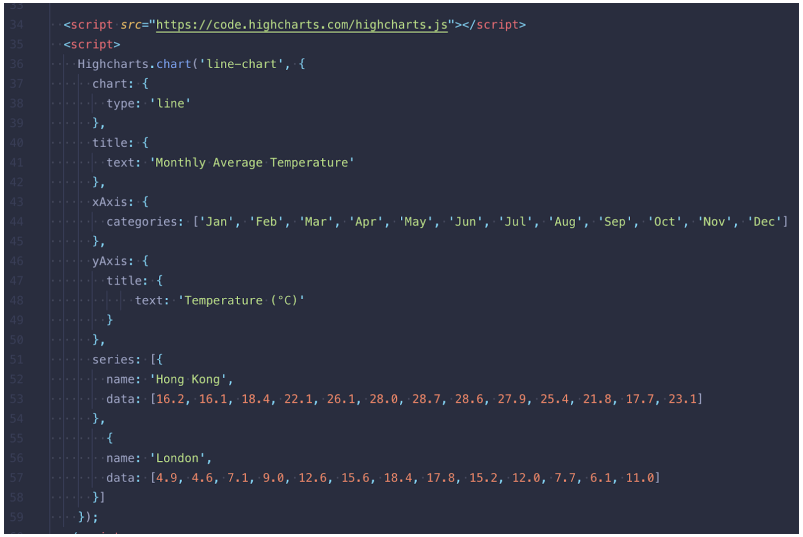
Please feel free to take a look at the source code
https://github.com/Luodina/js-data-viz/blob/2nd-stage/step-1-include-library/index.html#L34

2nd step - create the line chart

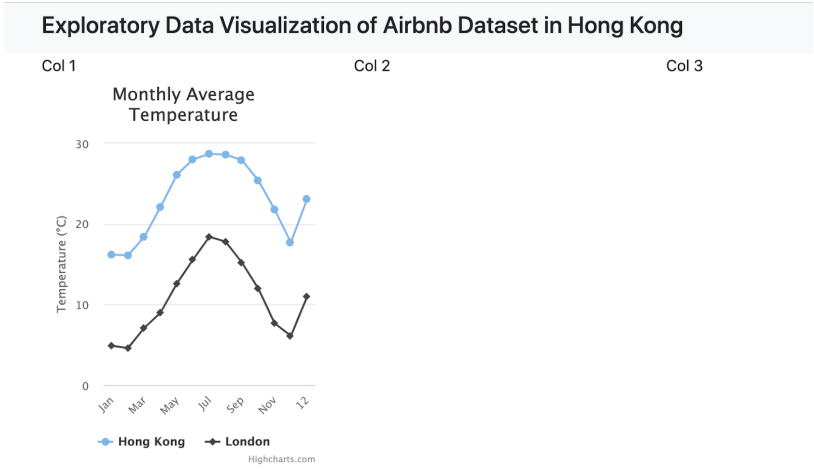
Let's start with the line chart. The task here will be to represent the monthly average temperature for Hong Kong and London. The data are gathered from the http://www.worldclimate.com site.
Please take a look at the monthly average temperature for Hong Kong
http://www.worldclimate.com/cgi-bin/data.pl?ref=N22E114+1102+45005W
And take a look at the monthly average temperature for London
http://www.worldclimate.com/cgi-bin/data.pl?ref=N51W000+1102+03772W

1. First of all, we have to create a container for the chart. For this, we will be using the first column from the grid. Take a look at the example:

2. Having this we are ready to create the chart itself. Take a look at the following code example:



And the result will look like:



Please take a look at the source code
https://github.com/Luodina/js-data-viz/blob/2nd-stage/step-2-add-line-chart/index.html

3rd step - create the column chart

Now let's take a look at the column chart. Now let's again compare Hong Kong with London in terms of monthly rainfall.
We will take the data from the http://www.worldclimate.com
For the Hong Kong monthly average rainfall, we will be using
http://www.worldclimate.com/cgi-bin/data.pl?ref=N22E114+2100+45005W
For the London monthly average rainfall, we will be using
http://www.worldclimate.com/cgi-bin/data.pl?ref=N51W000+2100+03772W

So, the steps are the same:

1. Create the container
2. Create a chart with and attach to the container

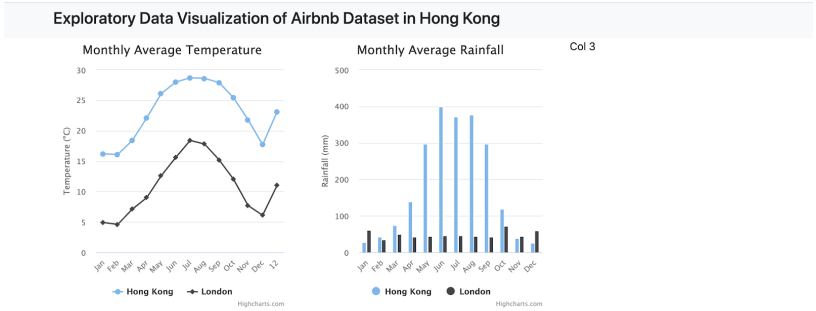
Create the container:

```
<div class="col">
  <div id="column-chart"></div>
</div>
```

And the chart itself:

```
// Column chart
Highcharts.chart('column-chart', {
  chart: {
    type: 'column'
  },
  title: {
      text: 'Monthly Average Rainfall'
  },
  xAxis: {
    categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
  },
  yAxis: {
    min: 0,
    title: {
      text: 'Rainfall (mm)'
    }
  },
  series: [{
    name: 'Hong Kong',
    data: [27.2, 43.6, 74.7, 139.8, 298.3, 399.4, 371.4, 377.0, 297.3, 119.0, 38.4, 25.4]
  },
    {
    name: 'London',
    data: [61.5, 36.2, 49.8, 42.5, 45.0, 45.8, 45.7, 44.2, 42.7, 72.6, 45.1, 59.3]
  }]
})
```

And the final result will look like:



Please take a look at the source code
https://github.com/Luodina/js-data-viz/blob/2nd-stage/step-3-add-column-chart/index.html

4th step - create the pie chart

Now let's create the pie chart. We will visualize the Hong Kong's principal ethnic minorities according to the following resource https://www.had.gov.hk/rru/english/info/info_dem.html

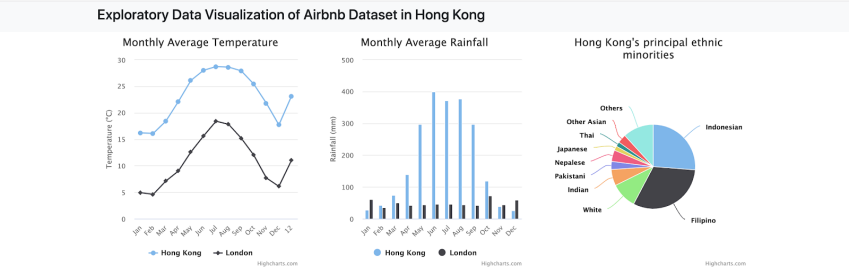
Create the container:

```
<div class="col">
  <div id="pie-chart"></div>
</div>
```

Create the chart:

```
// Pie chart
Highcharts.chart('pie-chart', {
  chart: {
    type: 'pie'
  },
  title: {
    text: 'Hong Kong\'s principal ethnic minorities'
  },
  series: [
    {
      name: 'Ethnic minorities',
      data: [{
        name: 'Indonesian',
        y: 153299
      }, {
        name: 'Filipino',
        y: 184081
      }, {
        name: 'White',
        y: 58209
      }, {
        name: 'Indian',
        y: 36462
      }, {
        name: 'Pakistani',
        y: 18094
      }, {
        name: 'Nepalese',
        y: 25472
      }, {
        name: 'Japanese',
        y: 9976
      }, {
        name: 'Thai',
        y: 10215
      }, {
        name: 'Other Asian',
        y: 19589
      }, {
        name: 'Others',
        y: 68986
      }]
```

And the final result should be like:



Please feel free to take a look at the source code
https://github.com/Luodina/js-data-viz/blob/2nd-stage/step-4-add-pie-chart/index.html

And that's all for stage 2. So far we know the basics of how to create charts. So far we have been working with hardcoded data, even thou we know the source of the data, but we had to hardcode it. Now is the time to learn how to request data from the network.

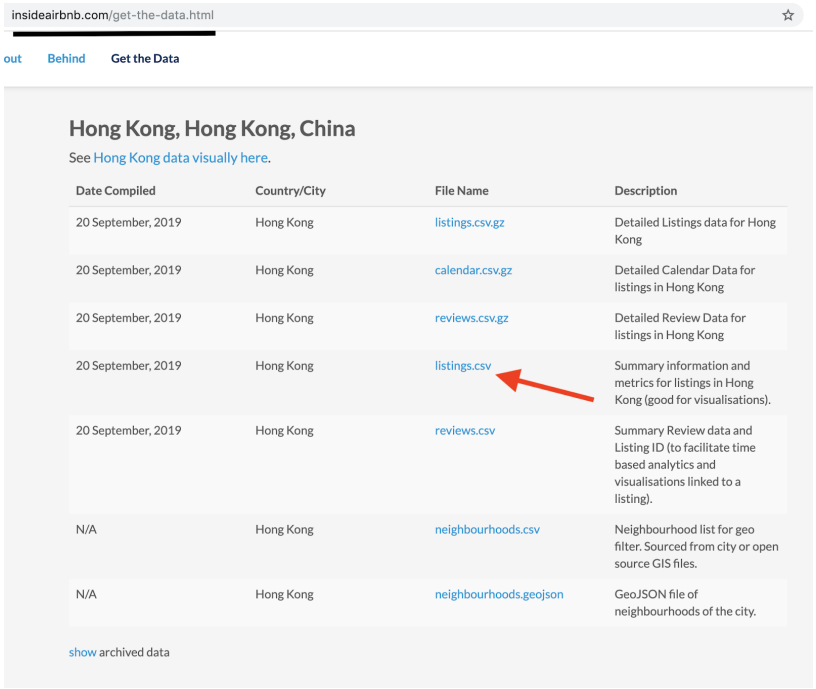
3rd Stage: Visualize data

So far we know how to bootstrap a web application, how to include Charts library and how to create charts. So far we have a solid base of knowledge to get started with real-life data visualization tasks.

Step 1: set up the data

Let's say we have a task to visualize the data from Airbnb. Airbnb stores the data in public, you can take a look at the data by following this link http://insideairbnb.com/get-the-data.html. By following this link we can search for Hong Kong and download the listings.csv file. Below the screenshot attached.

Please take a look at the following screenshot:



We will be using this file as a source of data for our visualizations. Let's now take a look at what kind of data this file contains.
Please take a look at the following screenshot:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	id	name	host_id	host_nam	neighbou	neighbourhood	latitude	longitude	room_type	price	minimu	number_	last_review	reviews_pe	calculated_h	availability_365	
2	69074	Beautiful oasis of	160139	Amy		Central & Western	22.28352	114.15018	Entire home/apt	1409	3	132	2019-09-05	1.26	1	173	
3	101765	Lamma Island fla	532909	Michael		Islands	22.20022	114.13461	Entire home/apt	431	2	11	2019-06-09	0.62	1	19	
4	103760	Central Centre 5	304876	Brend		Central & Western	22.28407	114.1557	Entire home/apt	853	2	262	2019-07-22	2.61	12	329	
5	132773	Fabulous 2 Bdrm	304876	Brend		Central & Western	22.28868	114.14494	Entire home/apt	1057	2	263	2019-08-15	2.64	12	323	
6	133390	Soho, Hong Kon	654642	Robin		Central & Western	22.28343	114.15539	Entire home/apt	939	2	27	2015-06-30	0.27	1	0	
7	163214	50m2 nest in Cer	767910	Pierre		Central & Western	22.28494	114.15251	Entire home/apt	1049	5	0			1	84	
8	163664	Soho off Hollywo	304876	Brend		Central & Western	22.28651	114.14874	Entire home/apt	697	2	213	2019-08-27	2.17	12	298	
9	163742	Soho off Hollywo	304876	Brend		Central & Western	22.28694	114.14855	Entire home/apt	775	2	219	2019-09-05	2.22	12	334	
10	228510	Creative Residen	1192493	Michael		Yau Tsim Mong	22.30983	114.16911	Private room	697	1	17	2017-02-08	0.18	1	365	
11	239788	Room nearby 4-s	1256401	Eric		Yuen Long	22.45758	114.0059	Private room	352	3	4	2017-03-23	0.05	1	0	
12	248140	Bright Studio - Sc	1300549	Darren		Central & Western	22.28291	114.15137	Entire home/apt	994	1	155	2019-09-08	1.61	1	204	
13	262212	Wan Chai MTR C	1375696	C S		Wan Chai	22.27626	114.17293	Entire home/apt	1315	1	73	2019-03-31	0.77	18	0	
14	263081	3BR, 2 Helpers, I	1370155	Chuster		Central & Western	22.27757	114.15205	Private room	4306	10	0			1	0	
15	274589	8 mins HKCEC, S	1435069	Shanshan		Wan Chai	22.2794	114.17118	Entire home/apt	1080	2	234	2019-08-26	2.48	1	345	
16	277437	Flat near HKU/IF	1443229	Alice		Central & Western	22.28359	114.13138	Entire home/apt	900	2	41	2019-04-23	0.43	5	333	
17	278277	FANTASTIC (HK	1452499	Sally		Wan Chai	22.27938	114.17368	Entire home/apt	861	3	244	2019-08-26	2.58	6	300	
18	284141	Big, quiet apt abc	1469210	Abheek		Central & Western	22.28317	114.15028	Entire home/apt	626	2	22	2016-06-20	0.23	1	0	
19	290161	Amazing Locatio	1505903	Sara		Central & Western	22.28114	114.15306	Entire home/apt	783	1	34	2018-09-29	0.49	1	0	
20	306515	HongKong,Centr	1576511	Frederic		Central & Western	22.28222	114.15448	Entire home/apt	1041	30	20	2015-08-14	0.21	1	263	
21	320448	SUITE 2: Next2 S	1375696	C S		Wan Chai	22.27599	114.17976	Private room	1002	1	15	2015-11-14	0.16	18	67	
22	320447	3BedR + 3BathR	1375696	C S		Wan Chai	22.27711	114.17772	Entire home/apt	2255	1	103	2019-07-23	1.13	18	176	
23	354442	5500 sq ft family	1794577	Simon		Southern	22.26791	114.12554	Entire home/apt	2497	5	2	2018-06-26	0.07	1	0	
24	405722	Soho/ Mid-levels	2022316	Tereza		Central & Western	22.2816	114.15042	Entire home/apt	1018	1	0			1	0	
25	410921	SUITE 1: Next to	1375696	C S		Wan Chai	22.27736	114.17949	Private room	783	1	18	2017-10-06	0.2	18	67	
26	412404	Budget Apt 1-5 p	1991696	Gigi		Yau Tsim Mong	22.29701	114.17138	Entire home/apt	548	1	221	2019-08-10	3.54	3	347	
27	415520	Nice, Harmony &	1564983	Patrick		Yau Tsim Mong	22.30598	114.1659	Entire home/apt	1088	3	446	2019-08-31	5.24	2	132	
28	442781	Spacious Lofty ar	2200572	Edouard		Central & Western	22.28756	114.13367	Entire home/apt	1245	5	10	2019-09-09	4.76	1	0	

Note: the listings.csv was imported to the Google docs, in order to display.

So basically here we can view detail information for the accommodations, the prices for each, the neighborhood and other useful things.

But you see the problem? It is hard to understand at least something from this file. And here is where the data visualization comes into play. And our task will be to visualize the data.
So, without useless speaking let's get into work!

First of all, for us, it would be easier to work with JSON representation of the data. Please take a look at the following screenshot:


```json
{
  "id": 69074,
  "name": "Beautiful oasis of plants & art @ best location",
  "host_id": 160139,
  "host_name": "Amy",
  "neighbourhood_group": "",
  "neighbourhood": "Central & Western",
  "latitude": 22.28352,
  "longitude": 114.15018,
  "room_type": "Entire home/apt",
  "price": 1409,
  "minimum_nights": 3,
  "number_of_reviews": 132,
  "last_review": "2019-09-05",
  "reviews_per_month": 1.26,
  "calculated_host_listings_count": 1,
  "availability_365": 173
},
{
  "id": 101765,
  "name": "Lamma Island flat 2min walk from the beach!",
  "host_id": 532909,
  "host_name": "Michael",
  "neighbourhood_group": "",
  "neighbourhood": "Islands",
  "latitude": 22.20022,
  "longitude": 114.13461,
  "room_type": "Entire home/apt",
  "price": 431,
  "minimum_nights": 2,
  "number_of_reviews": 11,
  "last_review": "2019-06-09",
  "reviews_per_month": 0.62,
  "calculated_host_listings_count": 1,
  "availability_365": 19
}
```


Basically, we have an array of objects. Each object represents a row from the CSV file. You don't have to do it yourself, the files are already prepared, by following this link https://github.com/Luodina/js-data-viz/tree/3rd-stage/step-1-local-data/data.

But just to let you know how it is done, please take a look at the following resource https://www.csvjson.com/csv2json where you can convert the CSV file to the JSON file.

By following the link https://github.com/Luodina/js-data-viz/tree/3rd-stage/step-1-local-data/data with the source code you will find the following files:

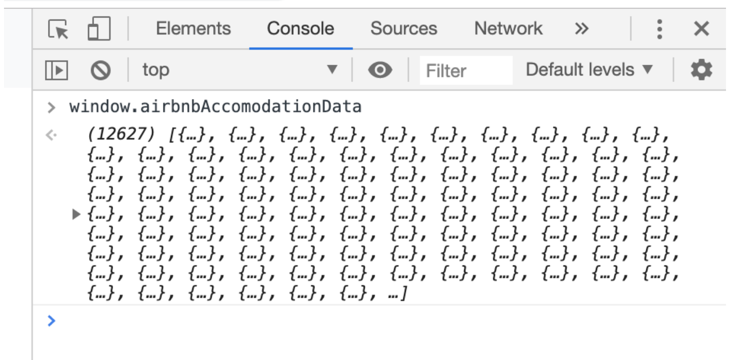
1. airbnb-data.csv - it is the listings CSV file downloaded from the Airbnb site.
2. airbnb-data.json - it is the JSON representation of the data.
3. airbnb-data.js - and also we have the JS file, which actually contains the JSON data, but it will export all of the data to the global *window* object in order to read this data.

Currently, we will be using the JS file airbnb-data.js, we will store this file locally. Later on, we will learn how to request the JSON files through the network requests.

Let's take a look at the airbnb-data.js file:

```
window.airbnbAccomodationData = [
  {
    "id": 69074,
```

Here we export the data to the window object. And now if we will open the console in the Developer Tools we will see something like:



Which means that the data are accessible through the *window* object.

And now that we have set up the data we are ready to move forward to visualize the data.

Step 2: Visualize the data

According to the data we have we can visualize the following information:

1. Types of accommodation
2. Accommodation prices
3. Neighborhood chart

Types of accommodation chart

Let's start with the "Types of accommodation" chart. You can find the source code by following the link
https://github.com/Luodina/js-data-viz/tree/3rd-stage/step-2-render-types-of-accomodation-chart
And now the step by step:

1. Create a charts directory
2. Create a types-of-accommodation.js file inside the charts directory
3. Load the data like in the screenshot below:

```
<!-- Load data -->
<script src="data/airbnb-data.js"></script>

<!-- Load types of accommodation chart -->
<script src="charts/types-of-accommodation.js"></script>
```

4. Load the types of accommodation chart:

```
<!-- Load data -->
<script src="data/airbnb-data.js"></script>

<!-- Load types of accommodation chart -->
<script src="charts/types-of-accommodation.js"></script>
```

5. And now let's take a look at the source code of the types-of-accommodation.js https://github.com/Luodina/js-data-viz/blob/3rd-stage/step-2-render-types-of-accomodation-chart/charts/types-of-accommodation.js
We have to understand how much private rooms, entire homes, and shared places do we have in our data. We will store this data in the object typeOfAccommodation

```
const typeOfAccommodation = {
  privateRoom: 0,
  entireHome: 0,
  sharedRoom: 0
};
```

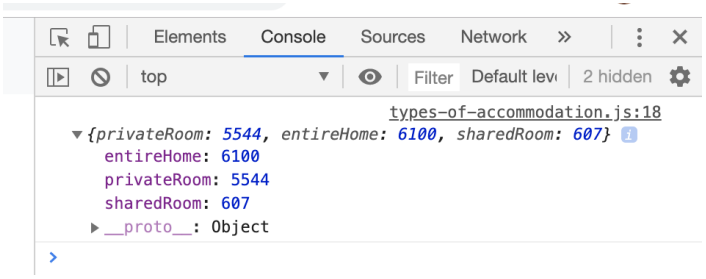
We should populate this data, for this, we will be running through the collection of accommodations. If we will find the "Private room" we will increase the privateRoom field by 1, the same for the other fields:

```
accommodations.forEach(accommodation => {
  if (accommodation.room_type === 'Private room') {
    typeOfAccommodation.privateRoom += 1;
  } else if (accommodation.room_type === 'Entire home/apt') {
    typeOfAccommodation.entireHome += 1;
  } else if (accommodation.room_type === 'Shared room') {
    typeOfAccommodation.sharedRoom += 1;
  }
});
```

Now if we will console.log the typeOfAccommodation

```
console.log(typeOfAccommodation);
```

We will see the following result



You don't have to do the console.log, it is here just for the demonstration. And so, what we see here is that we have 6100 of Entire Home accommodations, 5544 of Private Room and 607 of Shared Room.

And now by having this data, we are ready to display them in the chart:

```
Highcharts.chart('types-of-accommodation-chart', {
  chart: {
    type: 'column'
  },
  title: {
      text: 'Accommodation types'
  },
  xAxis: {
    type: 'category'
  },
  yAxis: {
    title: {
      text: null
    }
  },
  legend: {
    enabled: false
  },
  series: [
    {
      name: 'Accommodation types',
      colorByPoint: true,
      data: [{
        name: 'Private room',
        y: typeOfAccommodation.privateRoom
      }, {
        name: 'Entire home/apt',
        y: typeOfAccommodation.entireHome
      }, {
        name: 'Shared room',
        y: typeOfAccommodation.sharedRoom
      }]
    }]
});
```

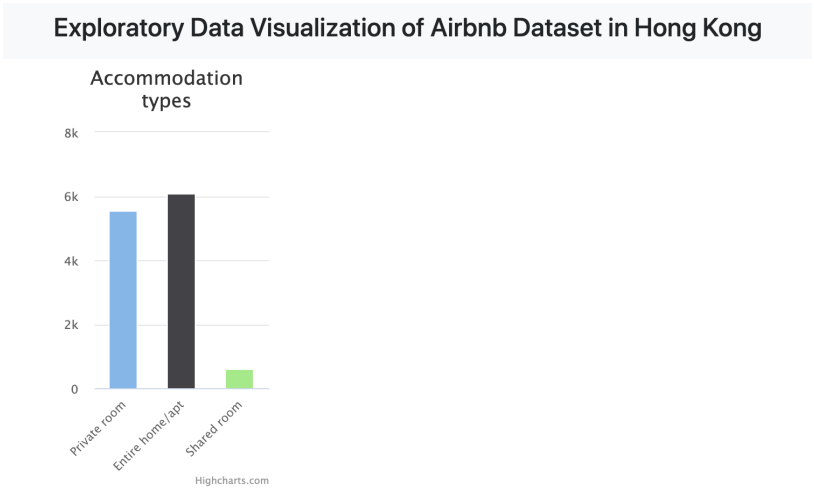
6. And the last step is to call the function. For this in the index.html we have to do the following:

```
<!-- Load types of accommodation chart -->
<script src="charts/types-of-accommodation.js"></script>

<script>
  // Create types of accommodation chart
  makeChartForAccommodationTypes(window.airbnbAccomodationData);
</script>
```

Here we are importing the types of accommodation file and calling the function and we are passing as a parameter the data that we have stored on the window.

7. The result will look like:



Please find the source code, by following the link
https://github.com/Luodina/js-data-viz/tree/3rd-stage/step-2-render-types-of-accomodation-chart
For the index.html you can take a look at the
https://github.com/Luodina/js-data-viz/blob/3rd-stage/step-2-render-types-of-accomodation-chart/index.html
For the type-of-accommodation.js you can take a look at the
https://github.com/Luodina/js-data-viz/blob/3rd-stage/step-2-render-types-of-accomodation-chart/charts/types-of-accommodation.js
We have done the “Types of accommodation chart”. It was the first item in the list of the chart to do. The next in the list is the “Accommodation prices” chart and the “Neighbourhood chart” chart.

Accommodation prices chart

Now is the time for the accommodation prices chart. The steps will be exactly the same as with the previous chart. Here we will take a look at:

1. How to extract the data for the charts?
2. How to create the chart.

We will follow up with the source code from
https://github.com/Luodina/js-data-viz/tree/3rd-stage/step-3-render-accomodation-price-chart

So, let’s assume that we already have created the script for the chart and already have included it to the index.html. And now we have to take a look at how to extract the data from the JSON. Please feel free to take a look at the source code
https://github.com/Luodina/js-data-viz/blob/3rd-stage/step-3-render-accomodation-price-chart/charts/accomodation-price-chart.js

1. We need to store somewhere the data for the chart.

```
const accomodationPrice = {
    lessThan500: 0,
    lessThan1000: 0,
    lessThan2000: 0,
    lessThan4000: 0,
    moreThan3999: 0
};
```

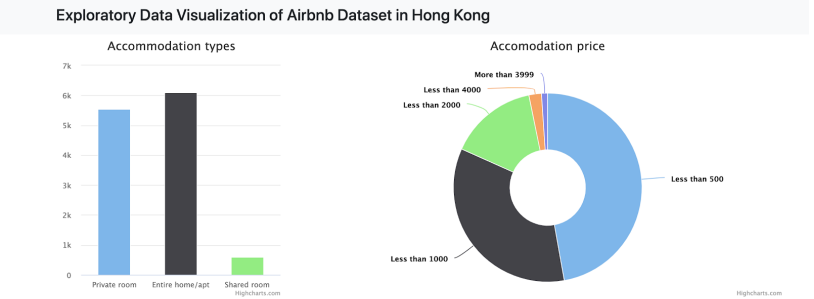
2. We need to populate this data

```
accomodations.forEach(accomodation => {
    if (accomodation.price < 500) {
        accomodationPrice.lessThan500 += 1;
    } else if (accomodation.price < 1000) {
        accomodationPrice.lessThan1000 += 1;
    } else if (accomodation.price < 2000) {
        accomodationPrice.lessThan2000 += 1;
    } else if (accomodation.price < 4000) {
        accomodationPrice.lessThan4000 += 1;
    } else if (accomodation.price > 3999) {
        accomodationPrice.moreThan3999 += 1;
    }
});
```

3. And now once we have the data ready to be displayed we can draw the chart

```
Highcharts.chart('accomodation-price-chart', {
    chart: {
        type: 'pie'
    },
    title: {
        text: 'Accomodation price'
    },
    series: [{
        name: 'Brands',
        innerSize: '40%',
        colorByPoint: true,
        data: [{
            name: 'Less than 500',
            y: accomodationPrice.lessThan500,
        }, {
            name: 'Less than 1000',
            y: accomodationPrice.lessThan1000
        }, {
            name: 'Less than 2000',
            y: accomodationPrice.lessThan2000
        }, {
            name: 'Less than 4000',
            y: accomodationPrice.lessThan4000
        }, {
            name: 'More than 3999',
            y: accomodationPrice.moreThan3999
        }]
    }]
});
}
```

And the final result will look like:



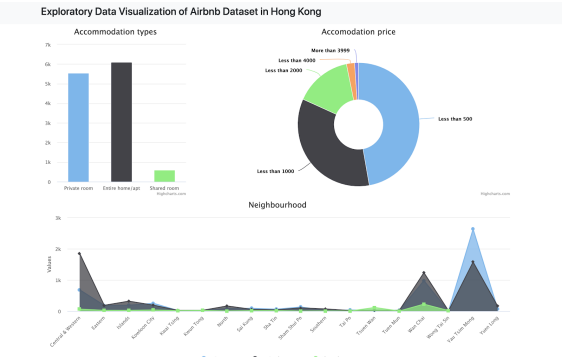
Neighborhood chart

For the neighborhood chart please follow up with the source code from https://github.com/Luodina/js-data-viz/tree/3rd-stage/step-4-render-neighbourhood-chart

The steps are exactly the same, only the algorithm to extract the data is different, please take a look at the source code:

1. The index.html https://github.com/Luodina/js-data-viz/blob/3rd-stage/step-4-render-neighbourhood-chart/index.html
2. The JS script https://github.com/Luodina/js-data-viz/blob/3rd-stage/step-4-render-neighbourhood-chart/charts/neighbourhood-chart.js

The final result should look like:



4rh Stage: Network requests

So far we made huge progress.

1. We know how to visualize the data.
2. We know how to extract the data for the visualization.

That's really great!
But there is one thing that we need to cover also. I am talking about network requests.

So far we are reading the data from the local machine. This approach is okay, it works and it seems that there are no problems with it. But in most cases, we don't have the data on our local machine and we will have to send the HTTP network request in order to fetch the data.

In this part of the tutorial, we will take a look at how we can send the HTTP network request and based on the response visualize the data.

The data we will be working

First of all please open the following link and take a look at the data that we will fetch using the HTTP request
https://raw.githubusercontent.com/Luodina/js-data-viz/3rd-stage/step-4-render-neighbourhood-chart/data/airbnb-data.json
This data are from the GitHub repository
https://github.com/Luodina/js-data-viz/tree/4rh-stage/step-1-http-requests/data but in a raw format. They are the same data with what we have been working so far, but the only difference is that we will fetch them through the network.

Run the local server

In order to be able to send HTTP requests, we have to run our application on a local server. We will be using the npm package http-server https://www.npmjs.com/package/http-server, please take a look at the documentation.

1. Install the HTTP server. For this run in the command line
   npm install http-server -g

   ```
   $ npm install http-server -g
   ```

2. Go to the root directory of the project and run in the console:
   http-server

   ```
   /js-data-viz$ http-server
   ```

   The result should be like:

   ```
   Starting up http-server, serving ./
   Available on:
     http://127.0.0.1:8080
     http://192.168.0.38:8080
   Hit CTRL-C to stop the server
   ```

3. Now if you will open in the browser the http://localhost:8080/ you will see that the application up and running.

And that's it. Now we are ready to send HTTP requests.

Send HTTP requests.

And now is the time to send the HTTP requests. Please follow up with the source code from https://github.com/Luodina/js-data-viz/blob/4rh-stage/step-1-http-requests/index.html#L49

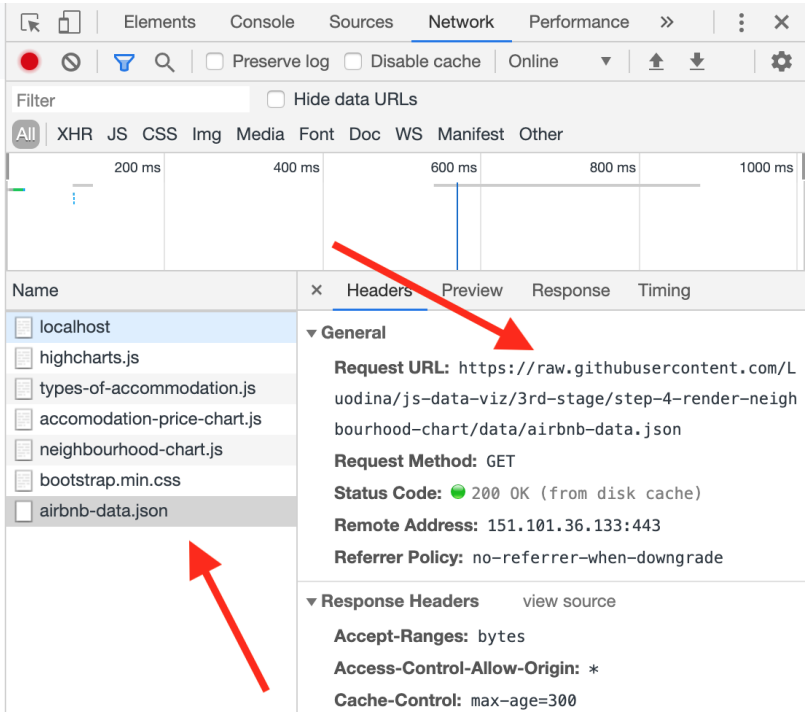
There is no so much to change in our existing code:

```
<script>
    const airbnbDataUrl = 'https://raw.githubusercontent.com/Luodina/js-data-viz/' +
                          '3rd-stage/step-4-render-neighbourhood-chart/data/airbnb-data.json';
    fetch(airbnbDataUrl)
        .then(response => response.json())
        .then(accommodations => {
            //Create types of accommodation chart
            makeChartForAccommodationTypes(accommodations);
            //Create accomodation prices chart
            accomodationPriceChart(accommodations);
            //Create neighbourhood chart
            neighbourhoodChart(accommodations);
        });
</script>
```

So what do we have here?

1. We have the constant airbnbDataUrl, which is the URL https://raw.githubusercontent.com/Luodina/js-data-viz/3rd-stage/step-4-render-neighbourhood-chart/data/airbnb-data.json from the GitHub repository.
2. We are using the fetch function in order to send the HTTP request.
3. By receiving the response we will extract from the response the JSON data
4. And once we have the JSON data we are ready to pass the data to our functions with the charts.

Please take a look at the following screenshot:



By opening the Network panel in Chrome developer tools, you will be able to see the request for the airbnb-data.json and the requested URL. That means that we actually sent the HTTP request.

And that basically it is about how to deal with HTTP requests.