



CMPT 384 – Information Visualization

D3.js

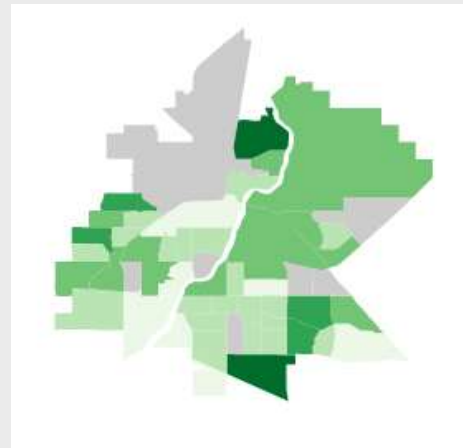
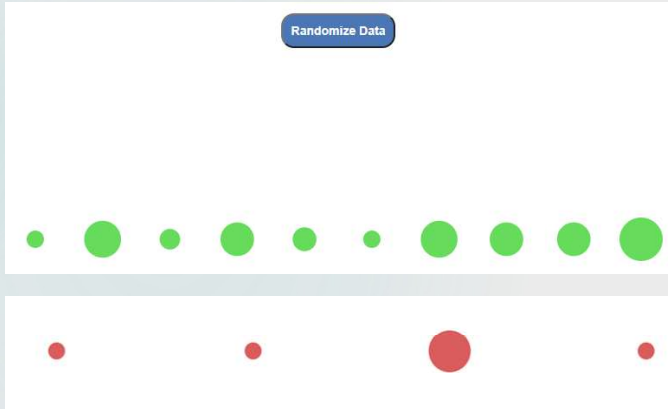
Lab 8

Course Instructor: Debajyoti Mondal

Lab Tutorial Instructor : **Arman Heydari**(arman.heydari@usask.ca)

Agenda

- D3 Geographic Map (Saskatoon)
- D3 Enter, Update and Exit
- D3 Zoom



Finding the Right GEOJSON

LIST OF MOST NORTH AMERICAN CITIES AND PROVINCES -

https://github.com/codeforamerica/click_that_hood/tree/master/public/data

CONVERT PUBLICLY AVAILABLE SHAPE FILES INTO GEOJSON -

<http://geojson.io/#map=2/20.0/0.0>

Finding the Right Data

OPEN-DATA Websites –

- <http://www.opendatask.ca/data/> - Curated List of all data on saskatoon available for free
- <http://opendata-saskatoon.cloudapp.net/> - Catalogue of Saskatoon open data
- <https://data.calgary.ca/> - Calgary open data

D3 Maps – Centering Projection

```
//Define map projection  
var projection = d3.geoMercator()  
  .center([-106.67, 52.1332])  
  .translate([w / 2, h / 2])  
  .scale([50000]);
```

Since we are looking at a city scale the projection to a higher value thus zooming into the map.

The default is 1000 for a world level overview.

Since saskatoon is to the west of the Prime (Greenwich) meridian , the value is negative.

Similarly a latitude south of the Equator would be negative

Saskatoon / Coordinates

52.1332° N, 106.6700° W



People also search for



Saskatchewan
52.9399° N,
106.4509°
W



Canada
56.1304° N,
106.3468°
W



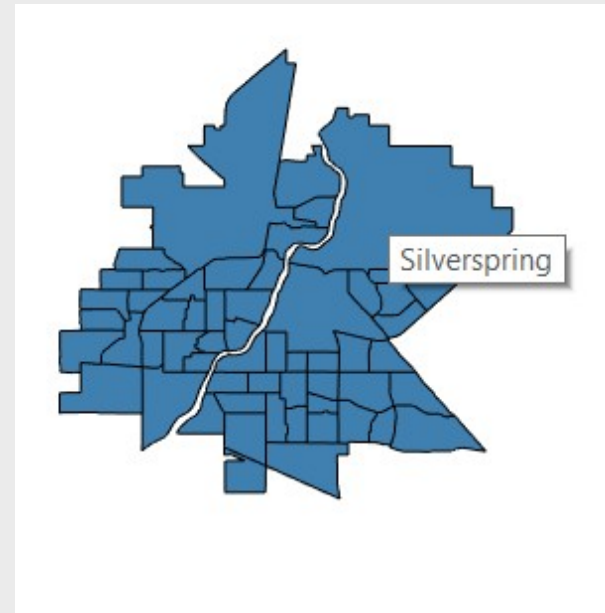
Regina
50.4452° N,
104.6189°
W

[Feedback](#)

Text on Mouse Hover


```
//Load in GeoJSON data
d3.json("saskatoon.json").then(function(json) {

  //Bind data and create one path per GeoJSON feature
  svg.selectAll("path")
    .data(json.features)
    .enter()
    .append("path")
    .attr("d", path)
    .attr('fill', 'steelblue')
    .attr('stroke', 'black')
    .append('title')
    .text(function(d) {
      return d.properties.name;
    })
});
```



Title is SVG's way of creating a tooltip , the way this looks however can be dependent on the browser .

Merging Data with Maps

**City of Saskatoon**
Open Data Catalogue BETA

[Data](#) [Developers](#) [Terms of Use](#) [About](#)

Recently Published

Transit Agency	03/10/2018
Transit Routes	03/10/2018
Transit Calendar	03/10/2018
Transit Stops	03/10/2018
Transit Trips	03/10/2018

Most Rated
Traffic Detours + 1

Most Viewed

Parcel	15168
ParcelAddress	13200
Transit Stop Times	9177
Neighbourhood Age Groups	6916
SLSN	3067

Datasets

CATEGORY

- ☐ Municipal Poll Area
- ☐ MunicipalWardArea
- ☐ Parcel
- ☐ Park
- ☐ School
- ☐ Service Alerts
- ☐ Transit
- ☐ Transportation

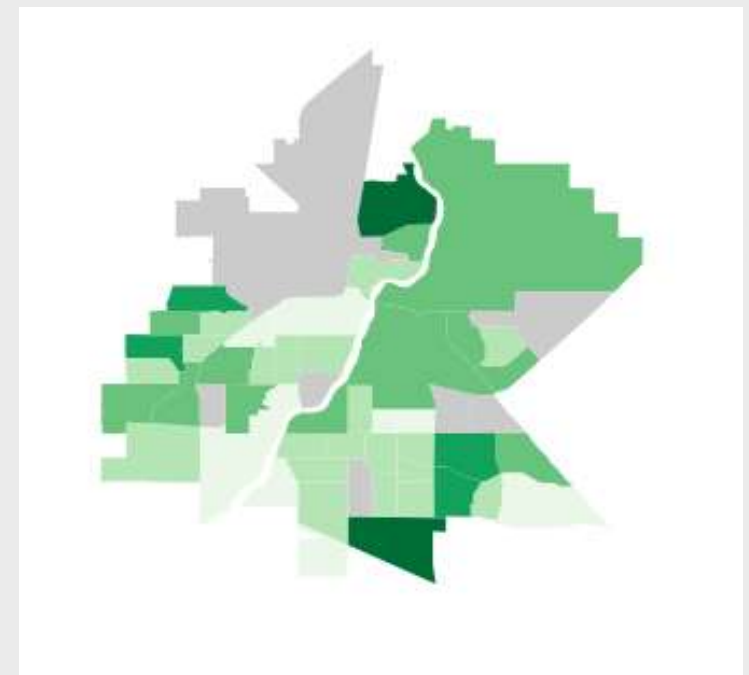
DATA SOURCE

- ☐ Saskatoon Open Data Catalogue Beta

DATES
from to
KEYWORDS

STATUS
☐ Published ☐ Planned
[Clear](#) [Filter](#)

NAME	CATEGORY	STATUS	DATE	RATING	VIEWS
Neighbourhood Parks The City of Saskatoon neighbourhood parks.	City Administration	Published	11/16/2016	not rated	63
Neighbourhood Personal Income The City of Saskatoon neighbourhood personal income.	City Administration	Published	11/22/2016	not rated	125
Neighbourhood Population The City of Saskatoon neighbourhood population in last four years.	City Administration	Published	06/24/2016	not rated	446
Neighbourhood Real Estate Sales The City of Saskatoon neighbourhood real estate sales in 2014.	City Administration	Published	11/16/2016	not rated	88
Neighbourhood Registered Vehicles The City of Saskatoon neighbourhood registered vehicles.	City Administration	Published	11/17/2016	not rated	41
Neighbourhood Voter Turn-out The City of Saskatoon neighbourhood voter turn-out.	City Administration	Published	11/16/2016	not rated	65
NeighbourhoodArea Neighbourhood Areas of the City of Saskatoon. A complete dataset in kml format can be downloaded from this open data site by selecting "Download as kml".	City Administration	Published	06/28/2017	- 3	2082



Example 2 – Map Saskatoon – Population for 2015

D3 : Enter - Update - Exit Pattern

0. Initial

```
<svg width="800" height="400"></svg> == $0
```

1. Enter

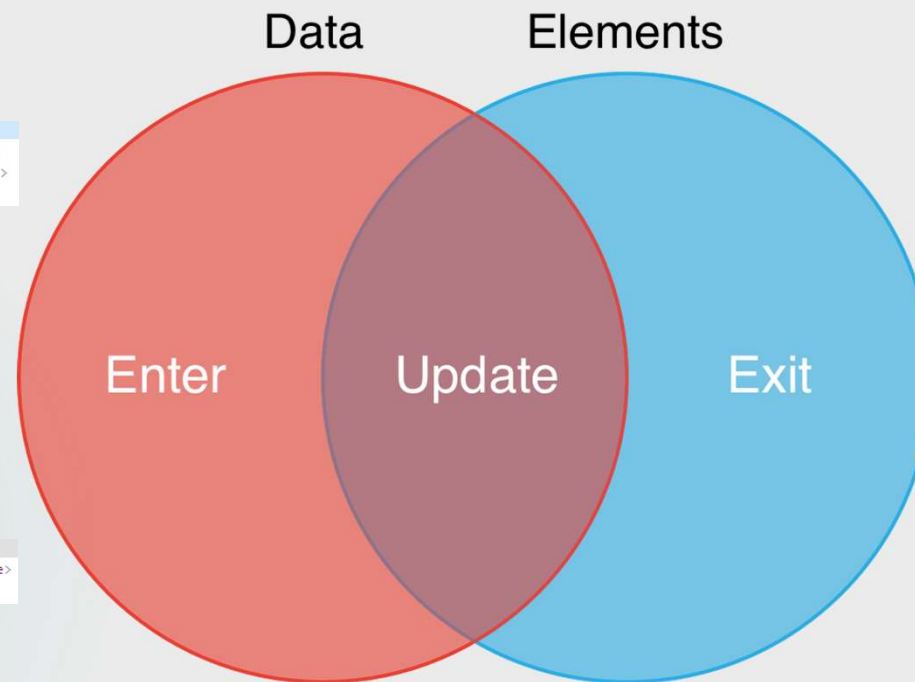
```
<svg width="800" height="400"> == $0
<circle cx="50" cy="200" fill="rgb(219, 92, 92)" r="25"></circle>
<circle cx="750" cy="200" fill="rgb(219, 92, 92)" r="10"></circle>
</svg>
```

2. Update + Enter

```
<svg width="800" height="400"> == $0
<circle cx="50" cy="200" fill="rgb(219, 92, 92)" r="25"></circle>
<circle cx="400" cy="200" fill="rgb(219, 92, 92)" r="25"></circle>
<circle cx="750" cy="200" fill="#66db5c" r="10"></circle>
</svg>
```

3. Update + Exit

```
<svg width="800" height="400"> == $0
<circle cx="50" cy="200" fill="rgb(219, 92, 92)" r="10"></circle>
</svg>
```



```
var svg = d3.select('body')
    .append('svg')
```

```
svg.selectAll('circle')
    .data(dataset)
```

0. dataset = []

1. dataset = [25, 10]

2. dataset = [25, 25, 10]

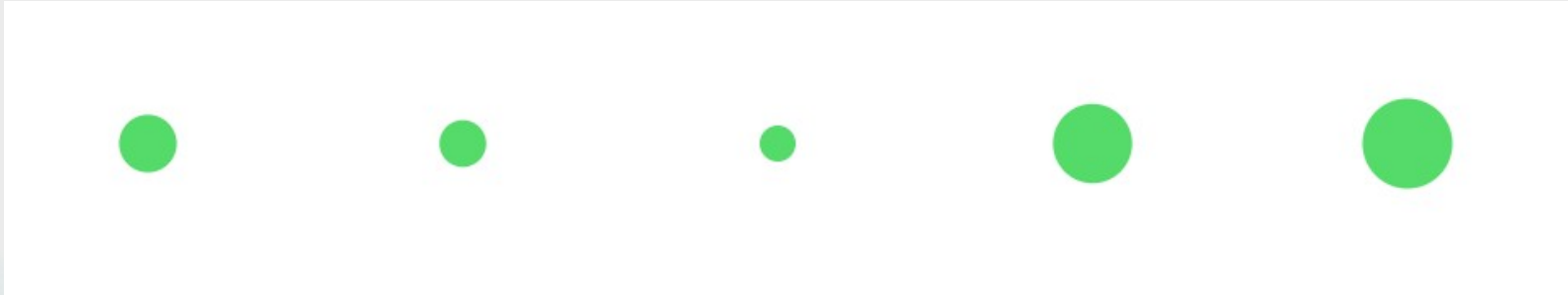
3. dataset = [10]

3 Scenarios on Data Update

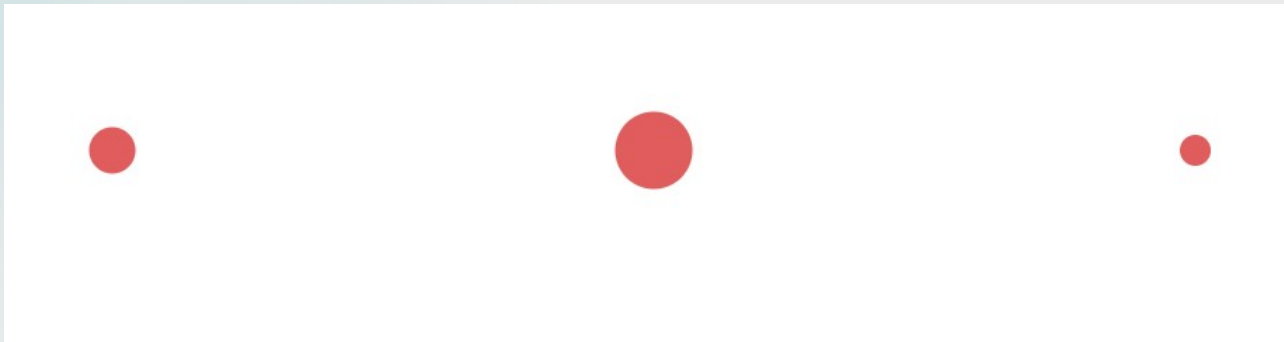
- ▶ The initial dataset and the next dataset have the same number of datapoints – Update the elements with new attributes
- ▶ The next dataset has more elements – Enter + Update , create new elements and update existing ones with new attributes
- ▶ The next dataset has fewer elements – Exit + Update , Remove elements that no longer exist in the new dataset and update the remaining ones

EXIT + UPDATE PATTERN

INITIAL DATASET - [3, 2, 1, 5, 6]



NEXT DATASET - [4,3,1]



ENTER + UPDATE PATTERN

INITIAL DATASET - [3, 2, 1, 5, 6]



NEXT DATASET - [5, 1, 1, 2, 7, 9]



Random Array Generator

```
// Returns a random number between 1 to 10
function getRandomNumber() {
    // random returns a value in decimal between 0 and 1 , so we multiply it by 10
    // to get one digit and then ceil the value to the nearest number
    // so if random returns 0.5688 , it becomes 5.688 after multiplying with 10
    // and then is ceiled to 6
    return Math.ceil(Math.random() * 10);
}

// Returns a array having random numbers between 1 to 10
function getRandomArray(length) {
    // create empty array
    var dataArray = new Array(length);
    // Fill the content with random numbers
    for (var i = 0; i < length; i++) {
        dataArray[i] = getRandomNumber();
    }
    return dataArray;
}
```

```
> getRandomNumber()
< 6

> getRandomArray(5)
< ► (5) [9, 3, 8, 1, 7]

> getRandomArray(4)
< ► (4) [8, 1, 9, 5]

> getRandomArray(2)
< ► (2) [6, 1]

> |
```

Pass Random Data on Click

```
//Create SVG element
var svg = d3.select("body")
    .append("svg")
    .attr("width", w)
    .attr("height", h);

// Call the function the first time for an initial set of 10 circles
// Since all the circles are new , they will be all green
updateCircles(10);

// Attach a click function to the button
// Each time the button is clicked we call the updateCircles function
// with a random number generated on the fly
d3.select('button').on("click", function() {
    var randomArrayLength = getRandomNumber();
    updateCircles(randomArrayLength);
});
```

Create Scales for X position and Radius

```
// xScale to get the x position of the circles
// evenly distributed in the available width
var xScale = d3.scaleLinear()
  .domain([0, dataset.length - 1])
  // 50 pixels padding on both sides
  .range([50, w - 50]);

// Size scale so the numbers in range of 1 to 10 are converted
// into range of 10 to 25 pixels for setting radius of the circles
var sizeScale = d3.scaleLinear()
  .domain([d3.min(dataset), d3.max(dataset)])
  .range([10, 25]);
```

JOIN Data

Get all the circles on the webpage and merge them with the new dataset

```
// JOIN - Join the data to the circles  
var circles = svg.selectAll('circle')  
    .data(dataset);
```

Exit Data

Remove circles that are no longer needed smoothly by taking them to the bottom of the screen and also reducing their size via transition

```
// EXIT - Circles that are no longer in the dataset  
circles  
    .exit()  
    .transition()  
    .duration(500)  
    .attr('r', 0)  
    .attr('cy', h)  
    .remove();
```

Update Elements

```
// UPDATE - Circles that are in the dataset but have different values
circles
  .transition()
  .duration(500)
  .attr('r', function(d) {
    return sizeScale(d);
  })
  .attr("cx", function(d, i) {
    return xScale(i);
  })
  .attr('fill', '#db5c5c')
```

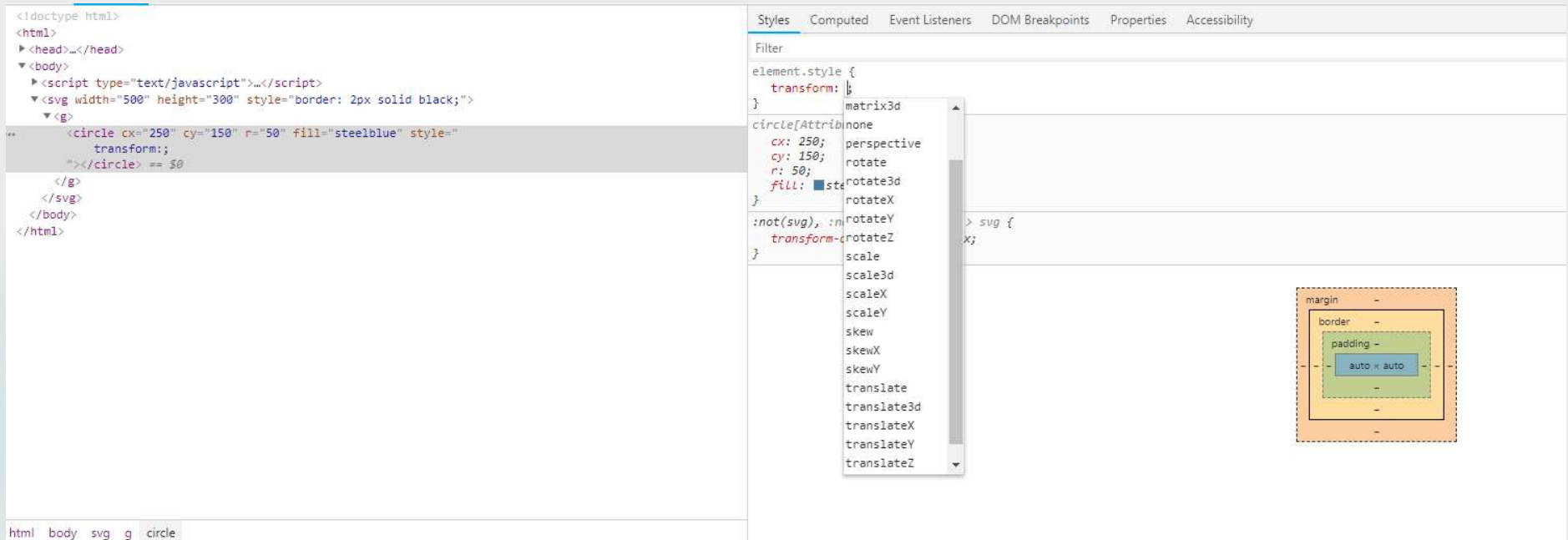
ENTER – New Elements

```
// ENTER - New circles that need to be added
circles
  .enter()
  .append('circle')
  .attr("cx", function(d, i) {
    return xScale(i);
  })
  .attr("cy", 0)
  .attr('fill', '#66db5c')
  .attr("r", "0")
  .transition()
  .duration(500)
  .attr('r', function(d) {
    return sizeScale(d);
  })
  .attr("cy", h / 2)
```

New Circles are created as dots on the top of the page which is why the cy and radius are initially set to 0

Then they transition to their correct places also while increasing in size to their actual radius.

SVG Transform Property



The screenshot displays a web browser's developer tools interface. On the left, the HTML element inspector shows a `circle` element within an `svg` container. The `circle` element has attributes `cx="250"`, `cy="150"`, `r="50"`, `fill="steelblue"`, and `style="transform: rotate(30deg);"`. The `svg` element has attributes `width="500"` and `height="300"`. On the right, the 'Styles' panel shows the `transform` property being edited. A dropdown menu is open, listing various transform functions: `matrix3d`, `none`, `perspective`, `rotate`, `rotate3d`, `rotateX`, `rotateY`, `rotateZ`, `scale`, `scale3d`, `scaleX`, `scaleY`, `skew`, `skewX`, `skewY`, `translate`, `translate3d`, `translateX`, `translateY`, and `translateZ`. A diagram on the right side of the Styles panel illustrates the box model with labels for `margin`, `border`, `padding`, and `auto * auto`.

For zoom effect we will use a mixture of Scale and Translate .

Scale creates the actual zooming while Translate is for panning the zoomed image so the zooming is happening with respect to the mouse cursor position

D3 – Zoom

```
//Create SVG element
var svg = d3.select("body")
    .append("svg")
    .attr("width", w)
    .attr("height", h)
    // create a border around the svg so the user has a reference perspective
    // when zooming or panning the chart
    .style('border', '2px solid black')
```

Initial SVG given black border so user has a reference container while zooming and panning the chart

```
// We are attaching the zoom to the svg but the actual zooming happens on the inner container
// this way we dont zoom the actual object we are observing for zoom events

// Create an inner g element inside the svg
// and create a circle inside that
var innerGraphicContainer = svg.append('g');

innerGraphicContainer.append('circle')
    .attr('cx', '250')
    .attr('cy', '150')
    .attr('r', '50')
    .attr('fill', 'steelblue');
```

Attaching the Zoom Callback

```
var attachZoom = d3.zoom().on('zoom', function(d) {
  d3.select('g').attr('transform', d3.event.transform);
});

//Create SVG element
var svg = d3.select("body")
  .append("svg")
  .attr("width", w)
  .attr("height", h)
  // create a border around the svg so the user has a reference perspective
  // when zooming or panning the chart
  .style('border', '2px solid black')
  // To the svg attach the zoom function
  // When a call function is called on an object , the object is passed as an argument
  // to the function being called , so in this case the object is the svg and the function
  // being called in the zoom function
  // so zoom function is attached to the svg
  .call(attachZoom);
```