# D3 in depth Tutorial

## Follow the steps below and answer the questions

Create a new project directory and copy in the html file where we used dimple to build the line chart of world cup attendances

Start a python server from your project directory

1. Navigate to the directory in the command prompt and issue the following command
   1. For python2 - **python -m SimpleHTTPServer**
   2. For python3 - **python -m http.server**
2. Navigate to localhost:8000 to view the file

**Note: If a function returns an array your answer should include what type of object and how many**

We will now replace the dimple code with D3 code, line by line, in the browser debugger

1. Add

## 1 Parsing/Formatting Data while importing

We need to ensure the data we import is in a format we can work with. The **json(), tsv()** and **csv()** functions all include the ability to parse the data using anonymous functions.

The tsv stores all data in string forma. We must transform the date column into a date object to ensure orderings and comparisons are correctly performed.

Change the import code as follows – note the anonymous function and understand that the transformation is performed before the data is passed to the draw function

**d3.tsv(“world\_cup\_geo.tsv”, function(d){**

**d3[‘date’] = format.parse(d[‘date’]);**

**d3[‘attendance’] = +d[‘attendance’];**

**return d;**

**}, draw);**

## 2 Adding data points to scatter plot

1. Add a debugger line before the dimple code is executed in the html file
2. Comment out the dimple code
3. Reload the page
4. What object exist before the dimple code gets executed
   1. What is the data object?
   2. What objects are stored in the data object?
   3. How would you display the data in a tabular form in the javascript console?
   4. What are the first three objects in the data when you view the data in tabular form
5. Run **d3.select(“svg”);** in the console
   1. What object gets returned?
6. Run **d3.select(“svg”).selectAll(‘circle’);**
   1. What object gets returned?
7. Run **d3.select(“svg”).selectAll(‘circle’).data(data);**
   1. Note: we are passing in the data object we examined earlier
   2. What object does this function return?
8. Run **d3.select(“svg”).selectAll(‘circle’);**
   1. How many circle elements are on the page?
   2. Did the command in line 6 create any? Why?
9. Run **d3.select(“svg”).selectAll(‘circle’).data(data).enter();**
   1. What object does this command return?
10. Run **d3.select(“svg”).selectAll(‘circle’)data(data).enter().append(‘circle’);**
    1. What object does this command return? Why?
11. Run **d3.select(“svg”).selectAll(‘circle’);**
    1. How is the return value different from the call in step 7?
    2. Check the \_\_data\_\_ field in the first circle object – where is it from?
    3. What are the attributes of the returned circles?
12. Add the code to create visual elements that are mapped to data points to the html code before the debugger line

## Adding Axis

We need an “X” for time and a “Y” axis for attendance.

Documentation here: <https://github.com/d3/d3-axis>

Good tutorial on axis here:

<http://www.d3noob.org/2012/12/setting-scales-domains-and-ranges-in.html>

We will use the extent function to build the axis

<https://github.com/d3/d3-array/blob/master/README.md#extent>

Returns the [minimum](https://github.com/d3/d3-array/blob/master/README.md#min) and [maximum](https://github.com/d3/d3-array/blob/master/README.md#max) value in the given iterable using natural order.

1. Create “date\_extent” and “attendance\_extent” variables for date and attendance using the documentation above as a guide
   1. Do so in the console first, and when you are happy your code works add it to the html file *before* the debugger
2. Create a time scale for the x-axis using our extent variables and the static variables for width height and margin
   1. **d3.time.scale().range(<pixel values, margin, width>).domain(<date\_extent>);**
3. . Create a linear scale for the y-axis
   1. **d3.scale().linear().range(<pixel values, margin, height>).domain(<count\_extent>);**
4. Create the D3 axis objects for our “X” and “Y” axis
   1. To create an axis - **d3.svg.axis()**
   2. To set the scale call **scale()** from the axis object and pass in the relevant scale object we created
   3. Check out **ticks()** and **orient()** functions as well
   4. When you understand these functions look at chaining the functions

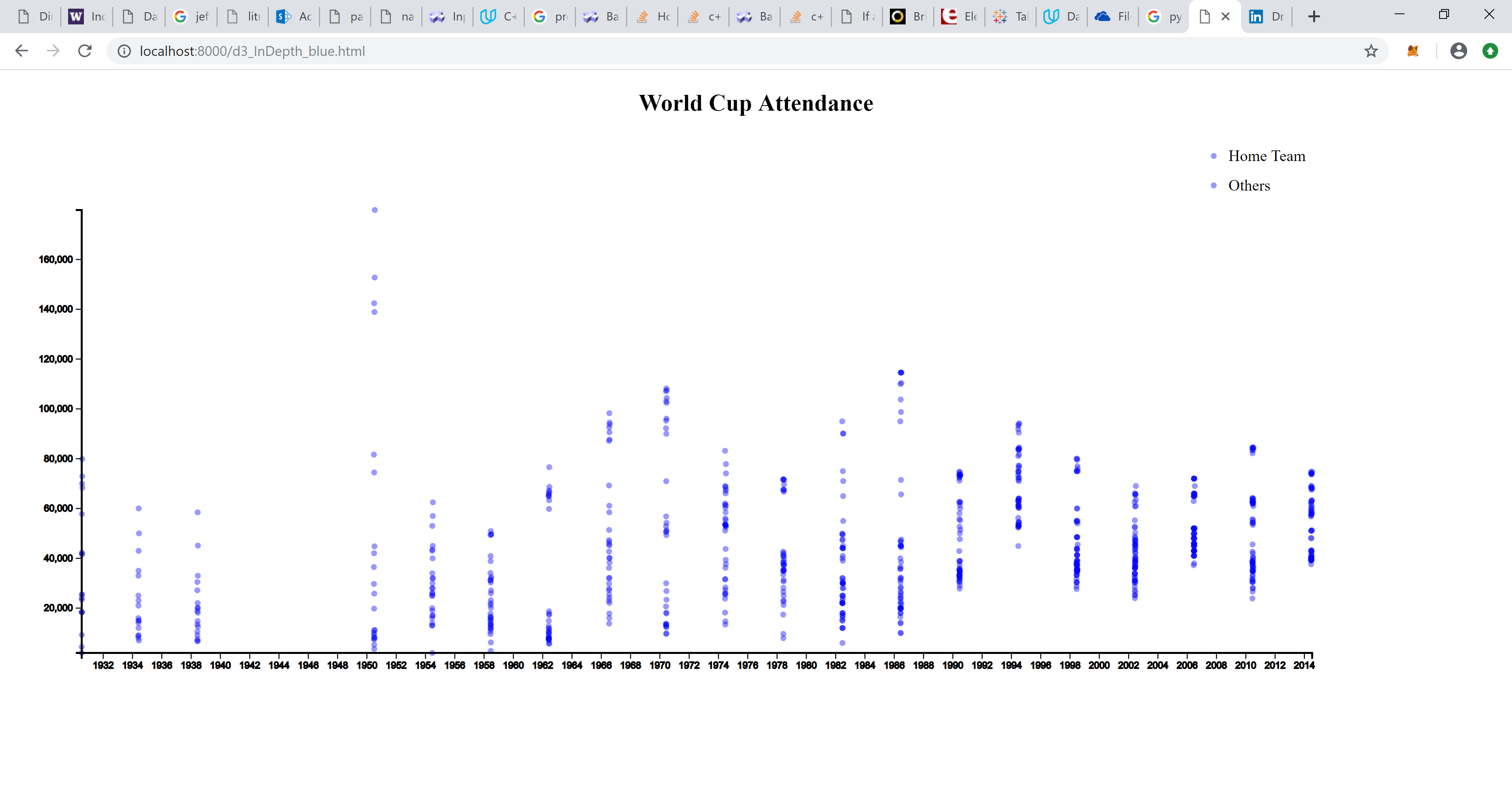
## Setting Attributes for the Visual Elements

Ensure there is a **debugger** statement after the axis are added.

Reload and inspect the page in your browser and see the axes, and the ticks.

Where are the circles you added to the SVG in section 2 step 9?

1. Run **d3.select(“svg”).selectAll(‘circle’);**
   1. What does this return? Why?
2. Run **d3.select(“svg”).selectAll(‘circle’)[0][0];**
   1. What does this return? Why?
3. Run **d3.select(“svg”).selectAll(‘circle’)[0][0].\_\_data\_\_;**
   1. What does this return? Why?
4. How can we select all the circles to edit their attributes?
5. What attributes do we want to visually encode?
   1. What values should we use to set the attributes?
   2. Where will we get access to these values?
6. Set the relevant attributes of each circle using anonymous/accessor functions.
   1. You will need to use the scale objects we created earlier to map data values to pixel values
   2. Use fixed variables for ‘r’ and ‘fill’ attributes
7. Check the graph is generated as follows in the browser



1. Now make encode any game involving the home country in a different colour and with a larger radius
   1. Use if statements in the anonymous functions we previously used to set attributes

## Legend

We know what the red points represent. A user who stumbles across the chart will not know what we are charting and why. In order to explain our chart, we should add a legend.

var legend = svg.append("g")

.attr("class", "legend")

.attr("transform", "translate(" + (width - 100) + "," + 20 + ")")

.selectAll("g")

.data(["Home Team", "Others"])

.enter().append("g");

legend.append("circle")

.attr("cy", function(d, i) {

return i \* 30;

})

.attr("r", function(d) {

if (d == "Home Team") {

return radius \* multiplier;

} else {

return radius;

}

})

.attr("fill", function(d) {

if (d == "Home Team") {

return 'red';

} else {

return 'blue';

}

});

legend.append("text")

.attr("y", function(d, i) {

return i \* 30 + 5;

})

.attr("x", radius \* 5)

.text(function(d) {

return d;

});