**Data Visualization**

**URL:** [**https://in.udacity.com/course/data-visualization-and-d3js--ud507**](https://in.udacity.com/course/data-visualization-and-d3js--ud507)

**Concepts:**

[Introduction](#introduction)

[D3 Building Blocks](#bulding_blocks)

[Design Principles](#design_principles)

[Dimple.js](#dimple)

[Narrative Structures](#narrative)

[Animation and Interaction](#animation)

[Project](#project)

**Introduction:**

**What is Data Visualization?**

It is about conveying a story or an idea as effectively as possible. It’s often said that a picture is worth 1,000 words and I think data visualization works in a same way. Data Scientists used data visualization everyday to explore patterns in their data and to ultimately convey the results.

Simply, how best to represent visually some underlying data using things like color, size, shape to convey some information or some insight to their audience or a reader.

**5-Steps for Data Visualization:**

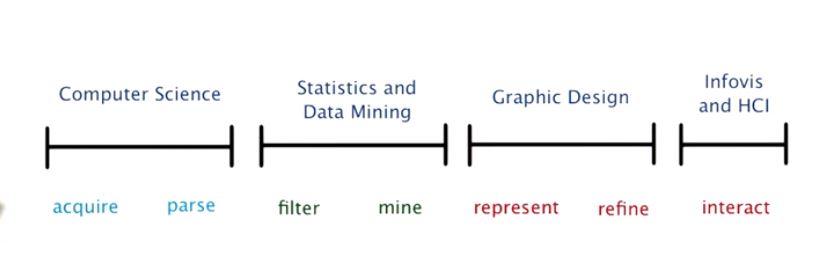
* Robust understanding of the context
* Choosing an appropriate type of visual
* Clutter
* Drawing the audience attention
* Story

In this you are going to practice using open source tools like D3.

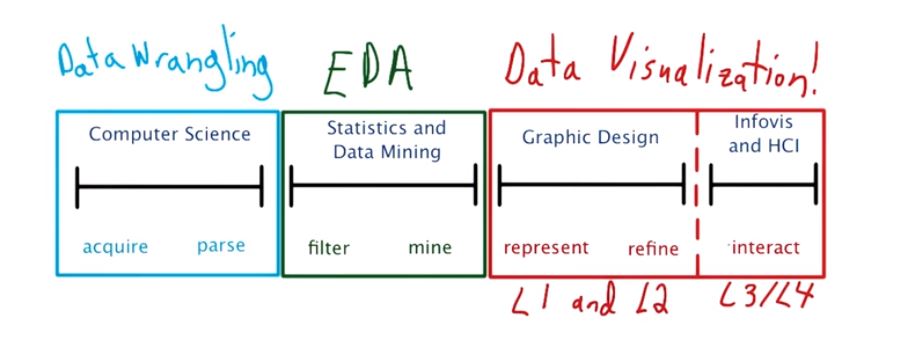
**The Data Science Process:**

* Acquiring and munging data
* Filtering and Mining
* Visual representation and how we are presenting the data
* How a reader is interact with your visualization?

The diagram of the process is as below:

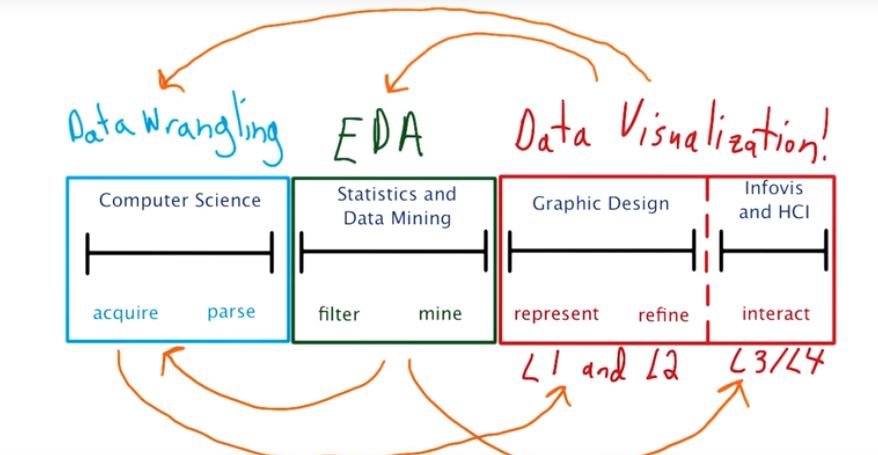


Each of these stages happens to map very nicely to the Dat C data science track.



**Interacting on Visualizations:**

The interaction will happen many times because we can’t visualize correctly for the first time.

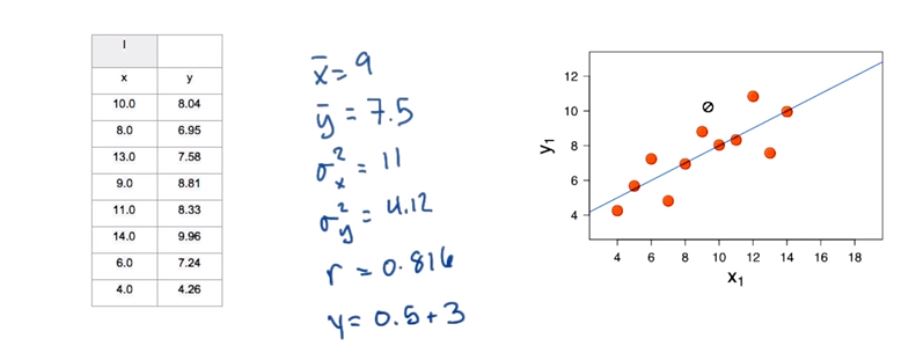


**Explanatory vs Exploratory:**

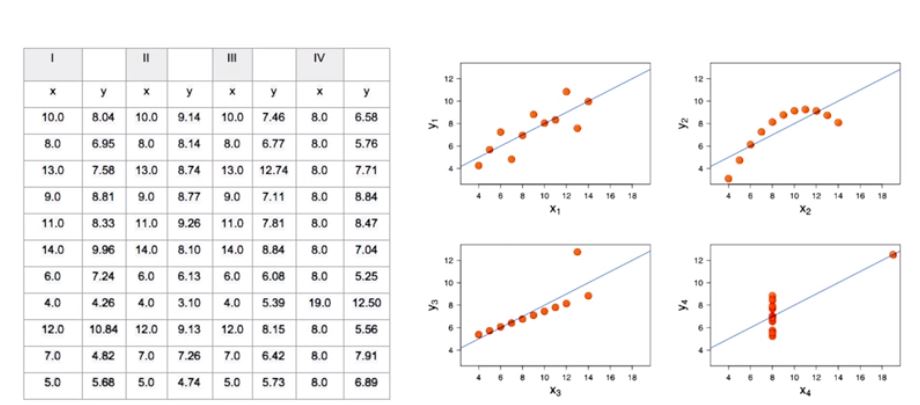
Eda means conversation between yourself and data. Data Visualization is a conversation between data and your audience.

**Anscombe’s Quartet:**

It is always important to plot a data. Let’s take an example.



Right side is the graph of the given data points. Let’s take few more graphs and find the relation between the points and the curvature.



Now we can see the there is difference in curvature. Tools like matplotlib in python or ggplot in R are great ways to create the plots.

**Data Types:**

We need to understand about data and data types first.

**Quantitative Data**: any variable that have exact number.

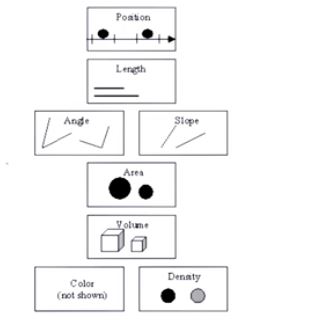
Ex: life expectancy, year, income per person (continuous), total population (discrete).

Categorical Data: categorizing into groups.

Ex: grouped into regions, ordered population bins, class difficulty.

**Visual Encodings:**

Visual encodings are mapping from data to display elements. There are some rankings for visual encodings.



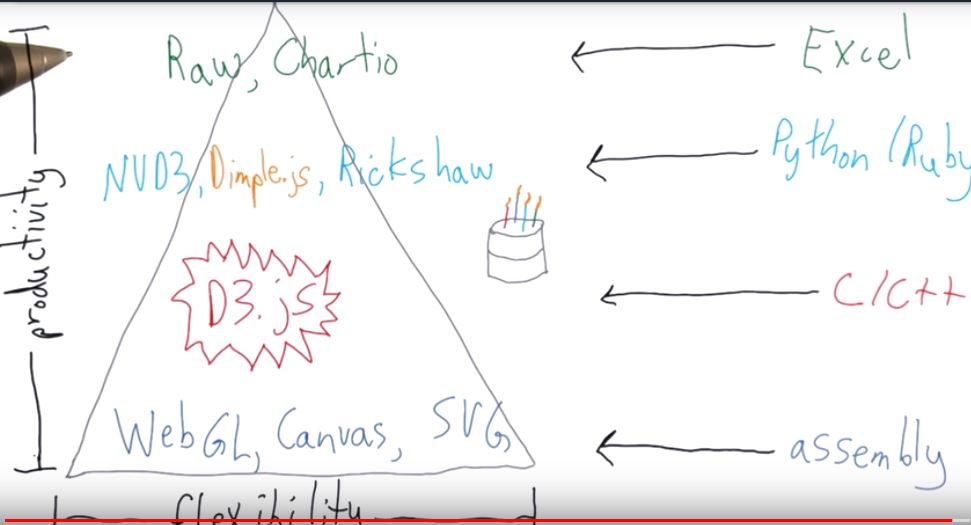
The top is most accurate and bottom is less accurate.

**Transition to Tech Portion:**

The what, why and how of D3?

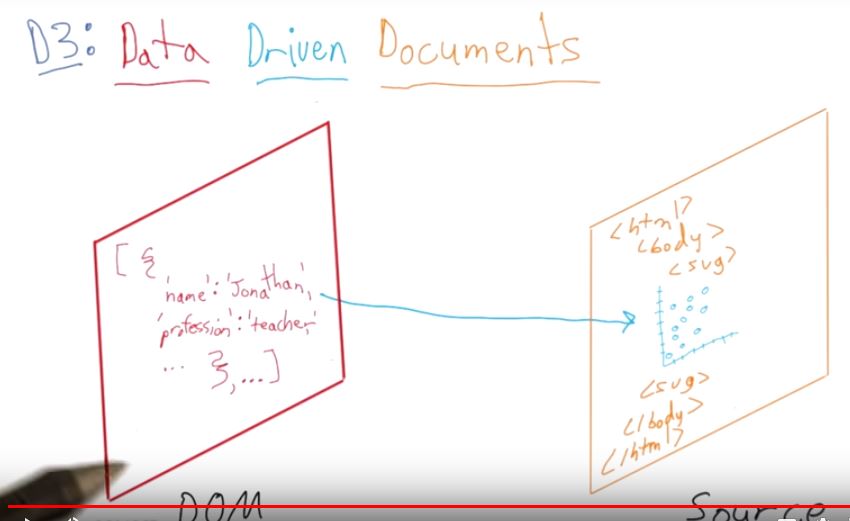
It is done by color, shape, length and scales.

There is a spectrum of visualization technologies.



**Unpacking the Name:**

D3: Data Driven Documents.



DOM is created during the page load.

DOM can be accessed by javascript API. It is a specification and hierarchical object.

**Why D3?**

* Leverages existing technologies. (DOM, CSS, SVG etc.)
* Separation of concerns.
* Benefits from advances in browser related technologies.(java script, HTML5 etc;)

**D3 Building Blocks:**

In this we are using D3 for interactive data visualizations. D3 stands on HTML, SVG, CSS and Java script. We will need to be comfortable before we start using it.

References:

D3 tutorial: <http://alignedleft.com/tutorials/d3>

Intro to HTML and CSS: <https://in.udacity.com/course/intro-to-html-and-css--ud001>

New version D3: <https://github.com/d3/d3/blob/master/CHANGES.md>

**Developer Tools and Console:**

By taking a web page we can learn how we can use it and became familiar with syntax, methods that comes along with D3. Because D3 JS is client-side java script library.

* Open a web browser.
* We can see both java script code and html code. To see the java script code we have to use.
* In Windows or Linux: ctrl+shift+j
* In Mac : ctrl+opt+j

Now we are going to see how to load D3.

**Loading D3:**

We can copy the source code of D3.js and paste it directly on the console. After pasting the code press enter or return then we can see all the functions are loaded into our web browser.

Another way to load D3 is to add D3 lib into script tag.



One important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications.

**Working in the JavaScript console:**

We can execute any arbitrary calculation it performs the operation like 10+2 returns 12. We can also even create functions. Here’s the start of the function that has no parameters and is called say\_hello. Start defining the function by using the keyword function. I give it a name without any parameter. I will hit shift+enter at the end of this line to go down into the console and then again press shift+enter to end the function. For body of the I simply return the hello world.



**Document Selectors and Queries:**

If the document word is in the console, it also get highlighted in the HTML view.



The other top level object that is available is window. The corresponds to the browser window while the document corresponds to the HTML page that has already been loaded.



In the above image if we observe, when we want to know the syntax. It can be easily find out by using the getElementById () method.

**D3 Selections:**

You can inspect the methods available by typing a dot after a variable name. In order to use D3 chainable API we need to create D3 selection.

**D3 Chain Syntax and Changing the Navbar:**

Using D3.js you can add, remove and change DOM nodes programmatically. The style property operates or acts on selection. This is important that when we first do the selection and for the second time there is no need to write the code. Just apply directly on it.

**Caution with Class Selections:**

We have an assumption about the class name navbar being unique to the page. Class names are not guaranteed to be unique on the webpage and many times a class name is used on multiple elements on a webpage, so that way we can repeat particular styles.

**Manipulating DOM Nodes:**

There are few more methods in order to change or transform the DOM nodes. You have already see the style function there more other like text, HTML and append. Also, some of these methods are both getters and setters.

**Nested Selections:**

Every method in D3 returns the element or selection of elements that method was called upon, you can naturally chain method from one to the next. The changes what we have done is on client- side, they are not permanent.

**Removing Elements with D3:**

If I want to remove a method. Actually I am not removing the method but I am removing inner HTML content.

**How Scales Work?**

The first thing you need to know is that when working with an SVG canvas the coordinates, in particular the Y coordinate starts at 0 and then increase going down. Actually the graph starts at 0 and it moves up for Y coordinate but the reverse is true for SVG. As you move right on an SVG element, the X value increases. And as move down on SVG the Y value increases.

**D3 Recap:**

 You learned to...

* append a scalable vector graphic (svg) element to the course viewer window
* set the appropriate scale for x and y coordinates
* append a red circle to the svg

The placement of the circle on the x and y axis was determined by the type of scale used to describe the data (linear or log), the domain values (max and min values of data) and range (max and min pixel values available on each axis). We inverted the range values for the y axis to account for the fact that the top of the y axis has a pixel value of 0. The max value extends down the y-axis rather than up the y-axis.

Here are the steps you took to accomplish this in the JavaScript console while viewing a course viewer page in Chrome. Note that the quotes used in expressions are straight, plain quotes (text-only) not curly, smart quotes (rich text format).

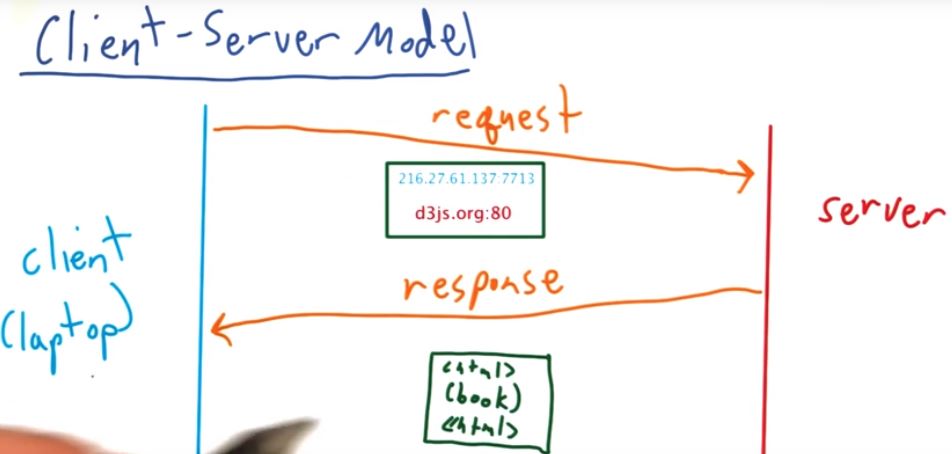
1. Paste contents of d3.min.js file into console.
2. File can be found <https://d3js.org/d3.v3.min.js>.
3. Expected Output: True
4. Clear content from div in course viewer: Input: d3.select('.main').html('');
5. Expected Output: >[Array[1]]
6. Define svg variable: Input: var svg = d3.select('.main').append('svg')
7. Expected Output: undefined
8. Assign y axis linear scale to y variable. This axis describes life expectancy. Note that while the height of the svg is 300 pixels, only 250 are used for the y axis to leave a buffer. Also, for the y axis the max value comes first because of a quirk with how objects are drawn in the browser: highest value at bottom of axis. Input: var y = d3.scale.linear().domain([15,90]).range([250,0]);
9. Expected Output: undefined
10. Assign X axis scale to x variable. This axis describes annual income. Input: var x = d3.scale.log().domain([250,100000]).range([0,600]);
11. Expected Output: undefined
12. Assign radius scale to r variable. The radius corresponds to the square root of the population. Input: var r = d3.scale.sqrt().domain([52070, 1380000000]).range([10, 50]);
13. Check scaling with console.log by plugging in life expectancy for China in y variable, and annual income per person for China in x variable, and population for China in r variable. Input: console.log(y(77), x(13330), r(1380000000));
14. Expected Output appox: 43.33333333333314 398.1976156961321 50
15. Append circle with attribute values for radius, fill color, center x and center y of circle: Input: svg.append('circle').attr('r', r(1380000000)).attr('fill','red').attr('cx', x(13330)).attr('cy', y(77));
16. Expected Output: >[Array[1]]
17. Look at placement of the circle in the svg, and compare with the placement in the original Gapminder World graph: [https://www.gapminder.org/world/#$majorMode=chart$is;shi=t;ly=2003;lb=f;il=t;fs=11;al=30;stl=t;st=t;nsl=t;se=t$wst;tts=C$ts;sp=5.59290322580644;ti=2013$zpv;v=0$inc\_x;mmid=XCOORDS;iid=phAwcNAVuyj1jiMAkmq1iMg;by=ind$inc\_y;mmid=YCOORDS;iid=phAwcNAVuyj2tPLxKvvnNPA;by=ind$inc\_s;uniValue=8.21;iid=phAwcNAVuyj0XOoBL\_n5tAQ;by=ind$inc\_c;uniValue=255;gid=CATID0;by=grp$map\_x;scale=log;dataMin=194;dataMax=96846$map\_y;scale=lin;dataMin=23;dataMax=86$map\_s;sma=49;smi=2.65$cd;bd=0$inds=;modified=60](https://www.gapminder.org/world/%23$majorMode=chart$is;shi=t;ly=2003;lb=f;il=t;fs=11;al=30;stl=t;st=t;nsl=t;se=t$wst;tts=C$ts;sp=5.59290322580644;ti=2013$zpv;v=0$inc_x;mmid=XCOORDS;iid=phAwcNAVuyj1jiMAkmq1iMg;by=ind$inc_y;mmid=YCOORDS;iid=phAwcNAVuyj2tPLxKvvnNPA;by=ind$inc_s;uniValue=8.21;iid=phAwcNAVuyj0XOoBL_n5tAQ;by=ind$inc_c;uniValue=255;gid=CATID0;by=grp$map_x;scale=log;dataMin=194;dataMax=96846$map_y;scale=lin;dataMin=23;dataMax=86$map_s;sma=49;smi=2.65$cd;bd=0$inds=;modified=60)

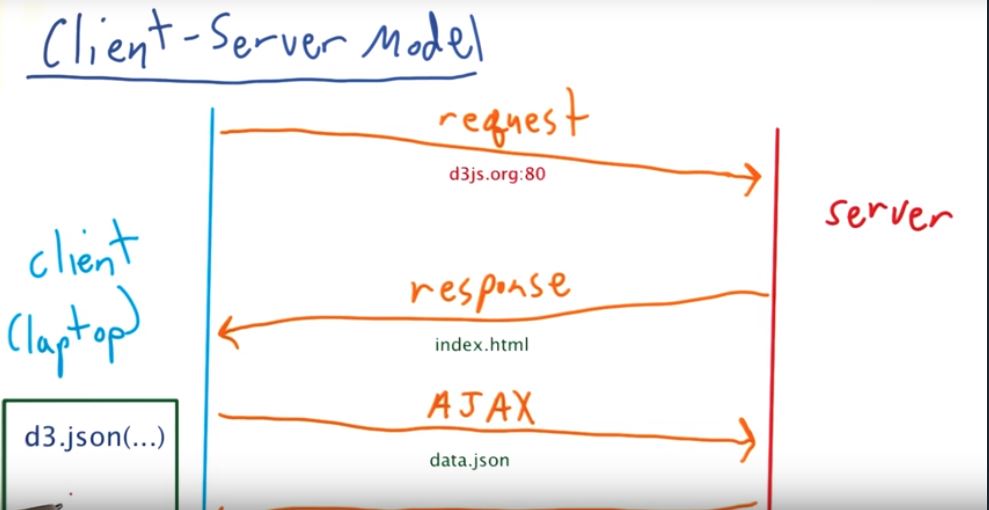
**Server Requests and D3:**

D3 special is the ability to easily load and manipulate data, often from external files or sources. And in addition there are d3 loading functions. The client- server model is mostly our laptop, and some external server.

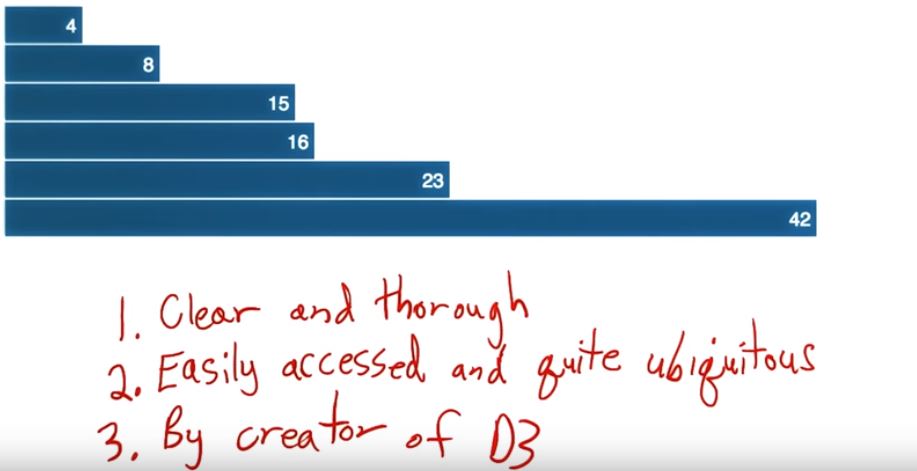
For example, if we want to go to D3 homepage, located at d3js.org and load index.html page.

* When you type the address in the web browser, the web browser is local on our laptop.
* It needs to send the request out to the internet, asking for the HTML of the d3.js homepage.
* Typically, it found can be found on one, if not many servers on the World Wide Web.

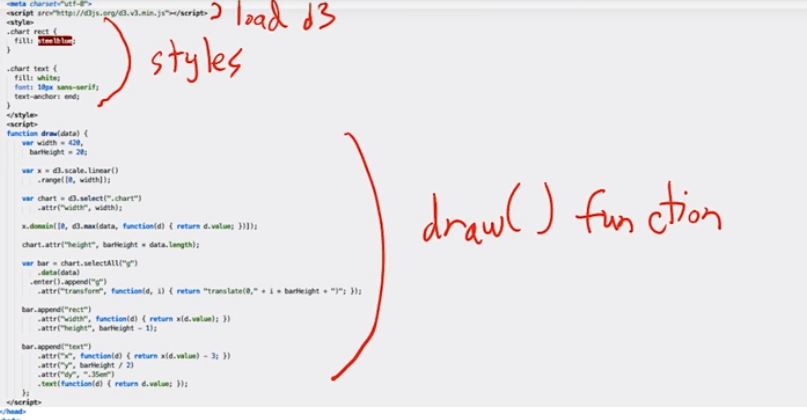




**Let’s Make a Bar Chart:**

****

**Code Structure and Java Script:**

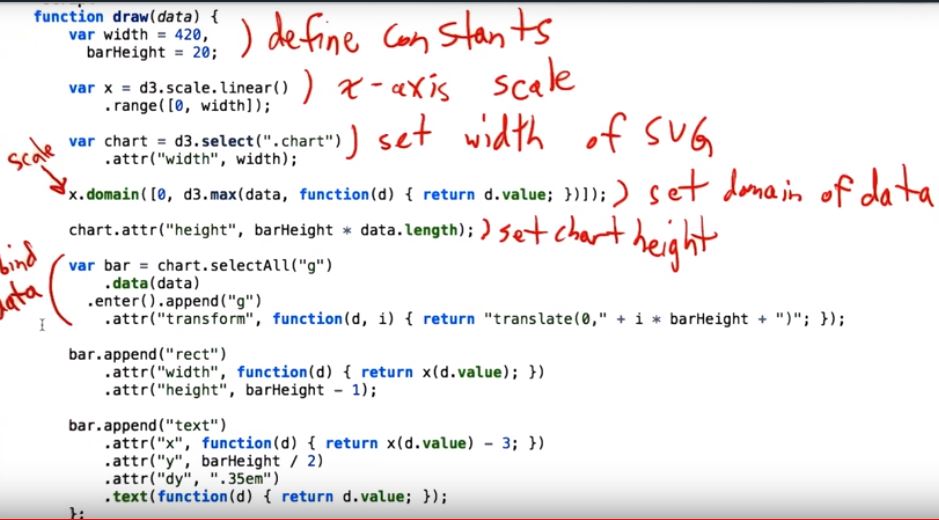


The loading, style and the draw () function are all in the head tag of our HTML page. As soon as we open the body tag we put an SVG with class chart to hold the visualization. And finally last bit of JavaScript is loads our external data file and then calls our draw function. Java script is asynchronous.

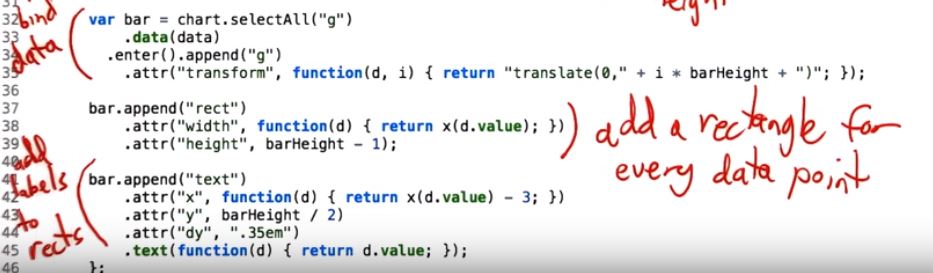
**Layout and Scales:**



**Binding Data:**



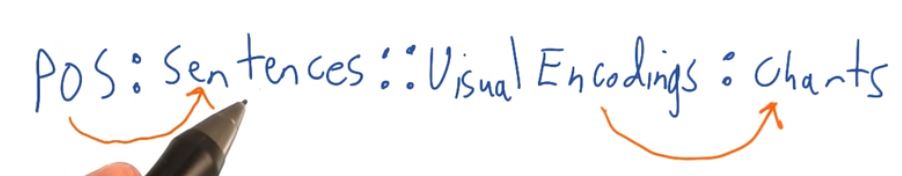
**Adding bars and text:**



**Design Principles:**

In this we are going to learn how to refine a data and also few metrics that will help you measure their effectiveness.

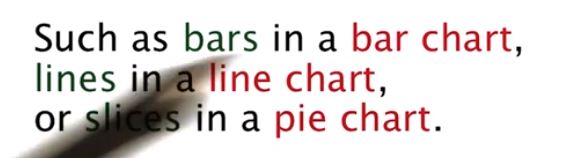
**Parts of Speech Analogy for Data Vis:**



Just as sentences are composed of parts of speech, charts are composed of visual encodings, each of which most effectively convey a specific aspect of underlying data.

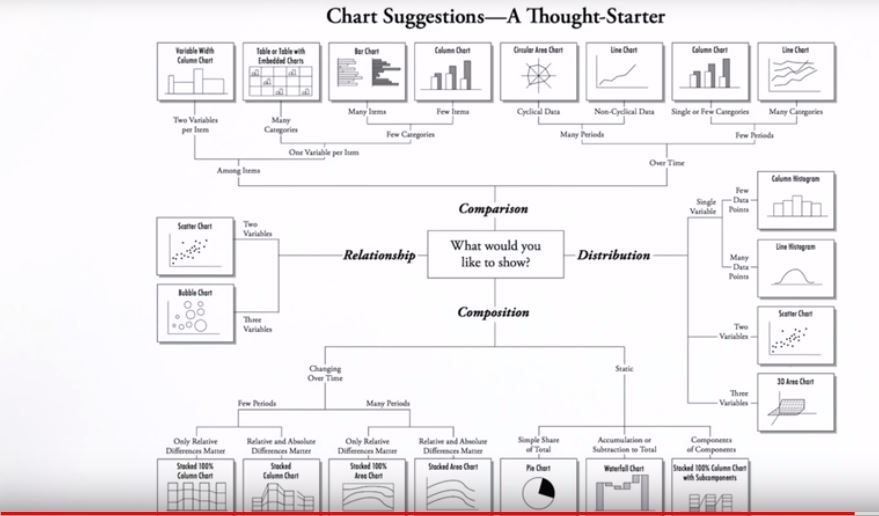
A chart is a graphical representation of data, in which data is represented by symbols.

Ex:



**Chart Types:**

There is a flow chart created by Andrew Abela to help navigate the somewhat overwhelming amount of different chart types.

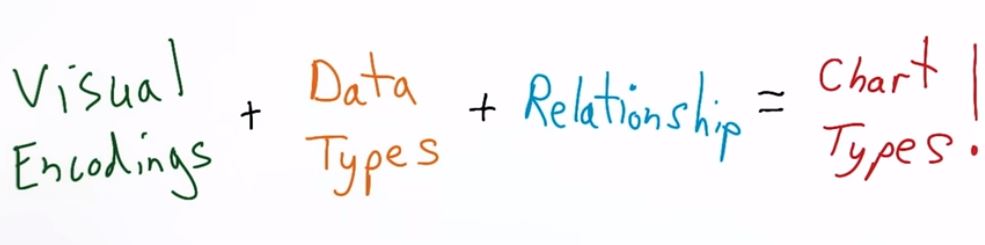


**Visualizations in Data Science:**

When creating visualizations, you should almost focus on communicating in the most effective manner to solve the problems at hand.

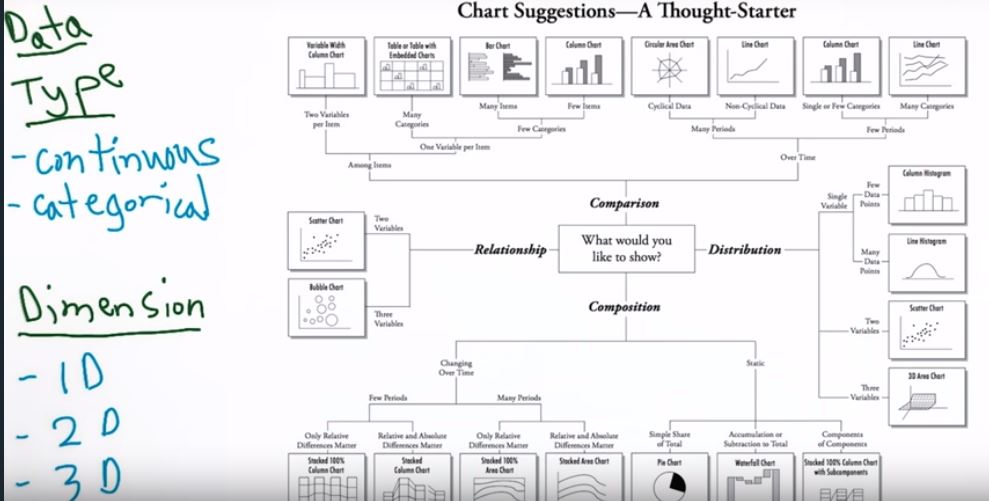
* Data Scientists first priority is solving problems with simple solutions.
* Choose the right tool to visualize the data like chart etc.

**Common Chart Types:**



For Example:

A scatterplot is an X coordinate and Y coordinate with a shape, in this case a circle as applied to two continuous variables, continuous being their datatype, in which the y value is dependent on the x value, which corresponds to relation between them.



**Small Multiples:**

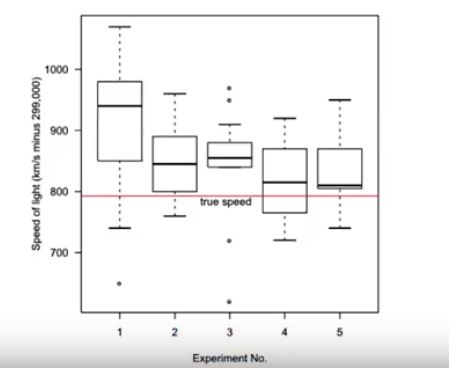
A small multiple sometimes known as trellis or a lattice plot is a series of chart laid out in a grid. They often represent common data.

Ex:



On x axis we have different months of year and on y axis we have different salary expenses for each department.

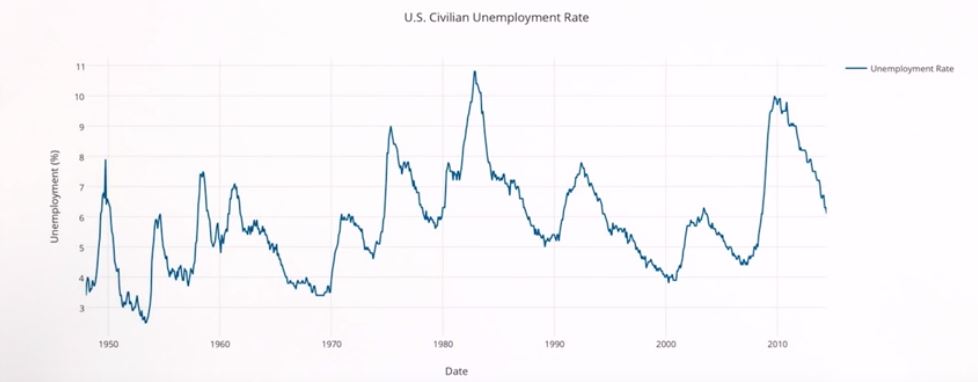
Box plot is another technique to visualize multiple distributions in one graphic.



Explanatory data analysis is to learn how your data’s distributed.

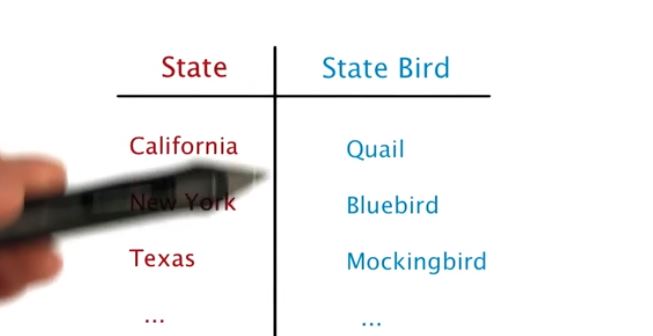
**Unemployment Line Plot:**

Here we are seeing the time series plot of U.S civilian unemployment rate over time. The data goes all the way back to about 1950, all the way up until the present time. On Y axis we have unemployment percentage, on X axis we have years.



**Additional Chart Types:**

The two important are text and maps.

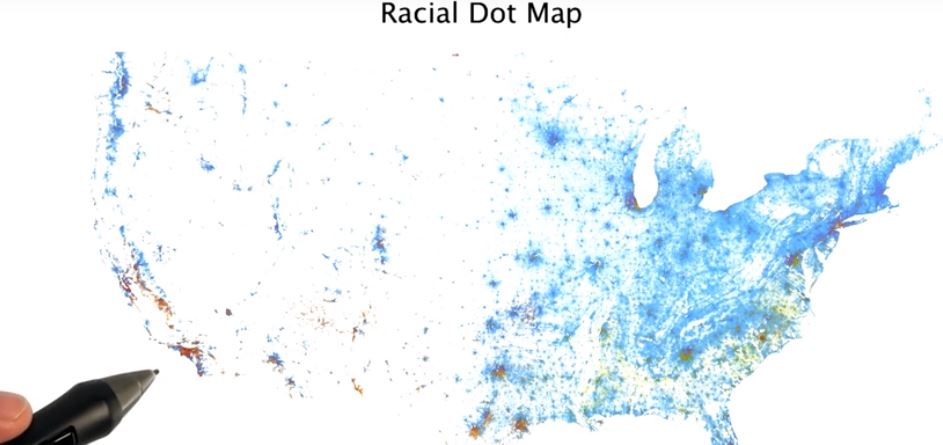


Example of table.

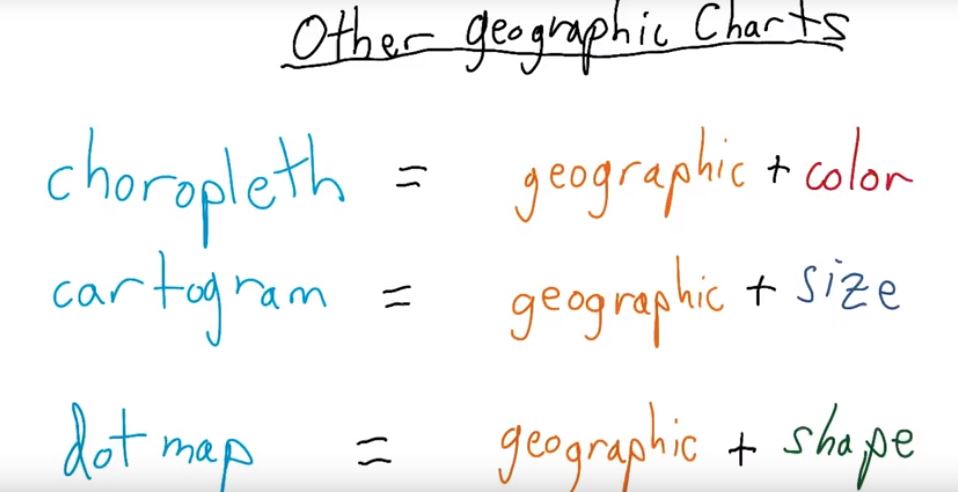
The definition of map is as defined below.



Map example:



Other Geographic Charts:



**More Amazing Chart Types:**

There are many other chart types for designing effective data visualizations. There's also more waiting to be created!

Read about these chart types by following the links in the titles below. Bullet graphs and sparklines are often good choices for graphics on dashboards. We'll leave it you to determine why.

Bullet Graph: <http://www.perceptualedge.com/articles/misc/Bullet_Graph_Design_Spec.pdf>

Stephen Few developed the bullet graph to replace meters and gauges that often fill too much valuable space on dashboards. You can read more about bullet graphs on <https://en.wikipedia.org/wiki/Bullet_graph>.

Sparklines:

<https://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0001OR>

Edward Tufte invented these bit-sized graphics to pack a punch of information in a small chart area. A reader can quickly see historical trends, anomalies, and the current status of a metric by viewing a sparkline. You can read more about sparklines on <https://en.wikipedia.org/wiki/Sparkline>.

Cycle Plots: <http://www.perceptualedge.com/articles/guests/intro_to_cycle_plots.pdf>

Originally created by Cleveland, Dunn, and Terpenning in 1978, cycle plots offer a way to investigate time series data in a different way than conventional line charts.

Connected Scatter Plots:

<http://www.thefunctionalart.com/2012/09/in-praise-of-connected-scatter-plots.html>

Think back to the Gapminder data visualization. Could you reveal the same patterns in the data over the years without animation?

Alberot Cairo says "Yes!.” Alberto praises connected scatter plots and shares examples of them on his blog, The Functional Art.

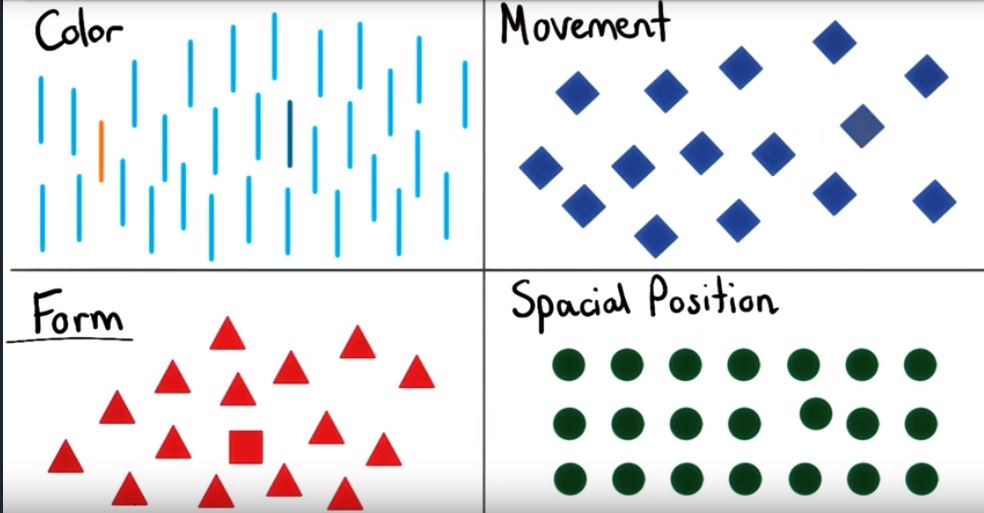
Violin Plot:

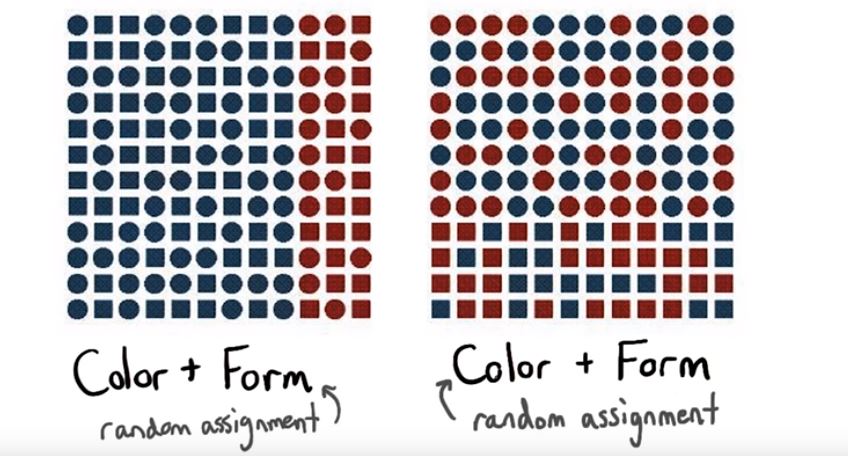
<https://en.wikipedia.org/wiki/Violin_plot>

Violin plots are similar to box plots, except that they show the probability density of the data at different values. Nathan Yau describes violin plots and other ways <http://flowingdata.com/2012/05/15/how-to-visualize-and-compare-distributions/> on his blog Flowing Data.

**Pre- Attentive Attributes:**

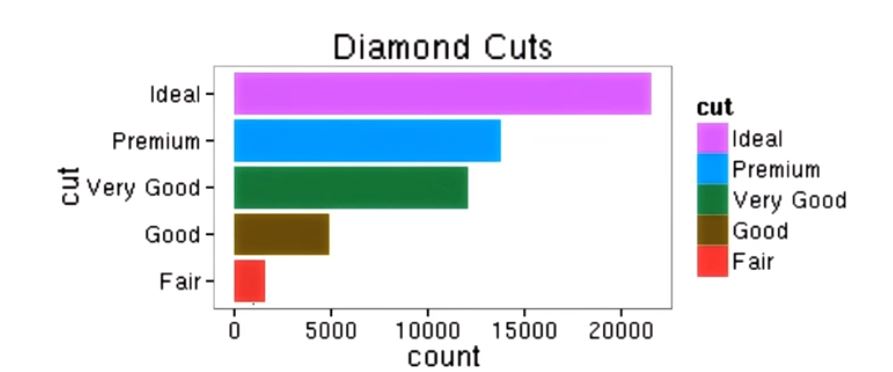
Preattentive processing taps into our automatic processes of vision and perception.





**Redundant Encodings:**

I would say that these colors distract from the data.



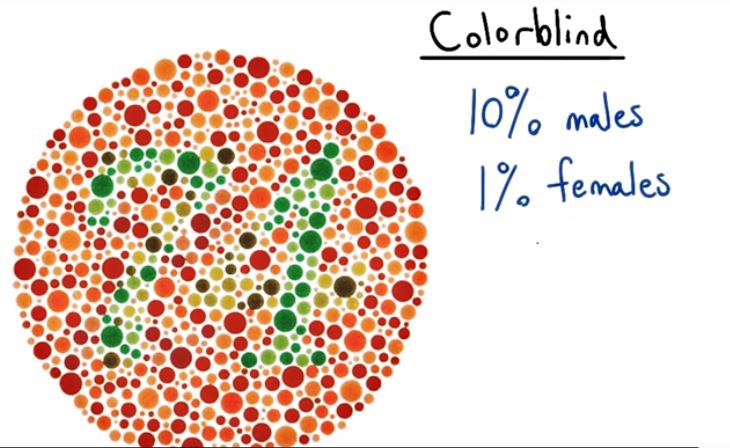
The default rainbow colors for the ggplot2 package for the R programming language. Use medium hues or pastels. Avoid using different colors.

**How Visualization can save lives?:**

The results of this research were staggering. Diagnostic accuracy increased from 31% to 91 %.

**Color Blind:**

Use color scales that everyone can see.



Suffering from color blindness.

**Gestalt Principles of Perception:**

We organize what we see in particular ways to make sense of visual information. There are six principles that influence the ways human see and understand visuals.

**PROXIMITY**

**SIMILARITY**

**FIGURE AND GROUND**

**CONTINUITY**

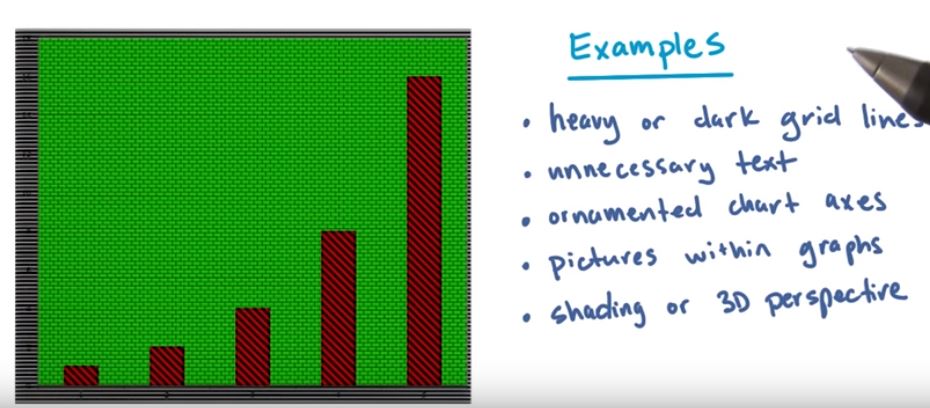
**CLOSURE**

**SIMPLICITY**

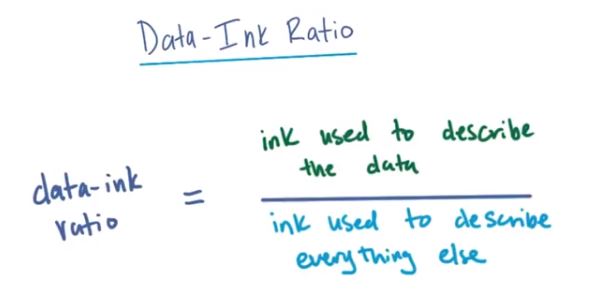
Take 15-20 minutes to review the following resources, and think about how you might use these principles when designing data visualizations. Many of these principles play an important role in choosing visual encodings and creating a hierarchy of information in a graphic.

**Chartjunk:**

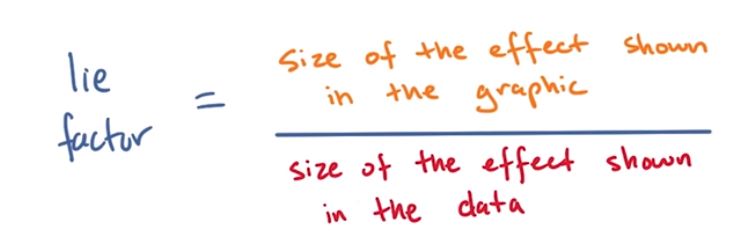
It is a term which refers to all visual elements in charts and graphs that are not necessary to comprehend the information represented on the graph.

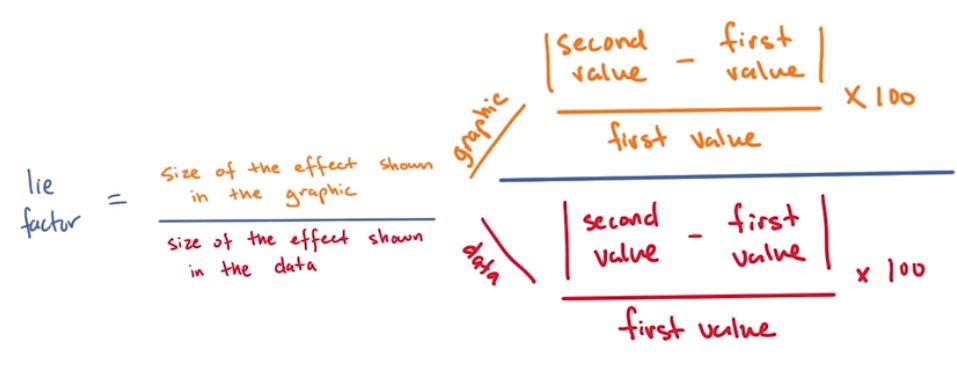


**Bad Visual Quizzes:**



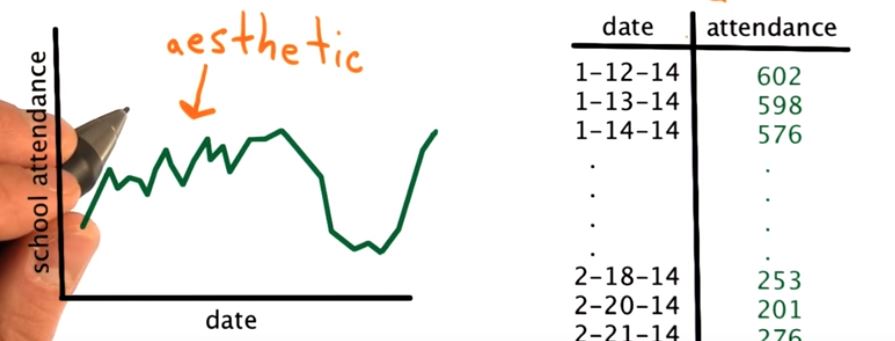
**Lie Factor:**





**The Grammar of Graphics:**

It is useful for creating visualizations.



This is the primary model.

**Grammar of Graphics:**

The Grammar of Graphics is a visualization theory developed by <https://en.wikipedia.org/wiki/Leland_Wilkinson> in 1999 with the publication of the eponymous <https://www.springer.com/in/book/9780387245447> .

It is quite an extensive theory which has influenced the development of graphics and visualization libraries alike (including D3 and its precursors), but in this class you will focus on 3 of its key principles:

* Separation of data from aesthetics
* Definition of common plot/chart elements
* Composition of these common elements

### Separation of Concerns:

You just saw some of the benefits of separating the data from the visual presentation of that data in the previous videos. The main take-aways are:

* Independently transform data and present data
* Delegate work and responsibilities
  + Engineer focuses on data manipulation
  + Designer focuses on visual encoding of data
* Present multiple visual representations of a dataset
* Ex: Bubble chart and line chart show [http://dc-js.github.io/dc.js/ facets](http://dc-js.github.io/dc.js/ facets%20) of a dataset.

### Common Elements

When thinking about creating a chart or graphic, it is often helpful to visually decompose what you want to achieve. In previous videos you saw how to abstract a chart into more basic visual encodings. In the map example, you saw that a choropleth is a combination of geography and color while a cartogram is a combination of geography and size. When talking about **composable** elements, a few of the most common are:

* Coordinate System (cartesian vs. radial/polar)
* Scales (linear, logarithmic, etc.)
* Text annotations
* Shape (lines, circles, etc.)
* Data Types (Categorical, Continuous, etc.)

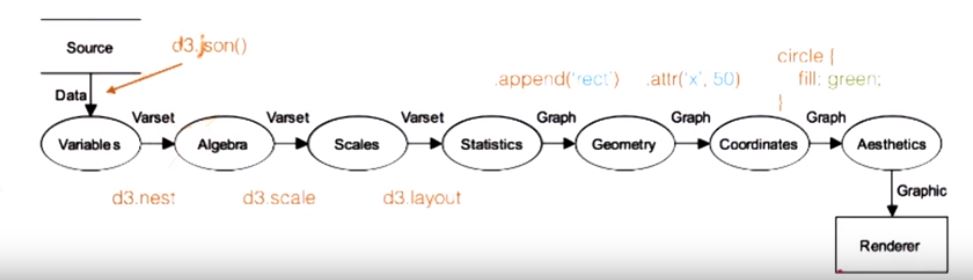
### Composition

The beauty of the Grammar of Graphics surfaces when you combine these common components. For example, you can create a bar chart by mapping a value in the data to the height of the bar in cartesian space, but you can also can also map these values in polar coordinates, in which the data value corresponds to the radial degree of a slice, to get a pie chart.

* Categorical + Continuous x Cartesian = Bar Chart
* Categorical + Continuous x Polar = Pie Chart
* Continuous + Continuous x Cartesian = Scatter Chart

And you can create a plethora of other charts by combining these common components in different ways. How might you achieve a line plot with a logarithmic scale from these common components?

**Grammar of Graphics and D3 Functions:**

****

**The Spectrum of Tech in Data Vis:**

****

**Dimple.js:**

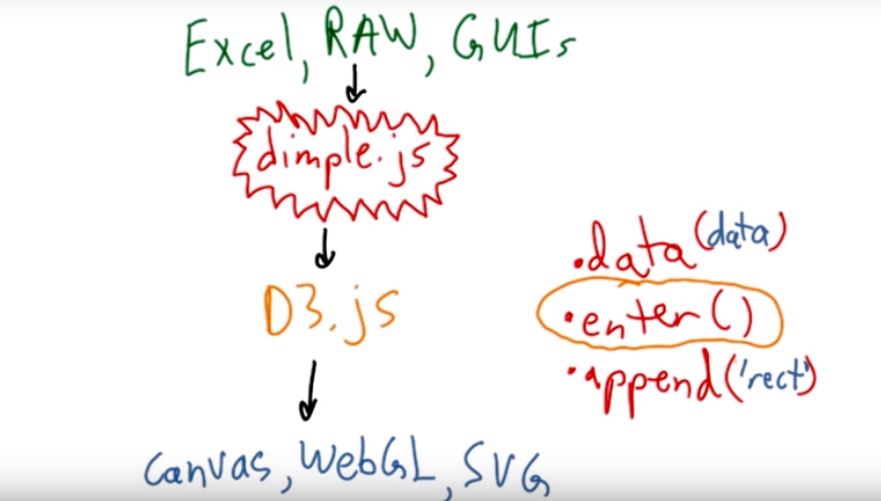
**Dimple Interlude:**

This is the homepage of Dimple.js. The link is given below:

<http://dimplejs.org/>

Dimple is a library built a top of D3 that allow us to work at abstraction level lower chart. It is a gentle learning curve designed for analysts. It exposes the native d3 object.

It make business analysts more predictive with chart. .enter and .exit commands can use in your data.



Interactive process of designing a visualization is SKETCHING.

**Dimple.js Code Files:**

In the following videos, Jonathan will walk through creating a bar chart using the Dimple.js plotting library. **Please download the file from the Downloadables section.**

Try opening the basic\_charts.html in your browser. What happens?

You should see the two axes and their labels. However, the data visualization will not show all of the data in the main display so you will need to do something else to get the visualization to render correctly in the browser.

To load the file, you will need to start a local server on your computer. Jonathan will explain how to do this in the next video. **To start a local web server, you will need to have** [**https://www.python.org/downloads/**](https://www.python.org/downloads/) **installed on your machine.**

Once you have Python installed, Jonathan will explain how to start a local web server and load the file in the next video.

If you have trouble installing Python, please post in the discussions for support. Be detailed about the version of your operating system and any error messages you receive.

There are other ways to start a local web server. To learn more about why you need to start a local web server and other ways of setting up a local web server, please read [https://www.oreilly.com/#\_setting\_up\_a\_web\_server](https://www.oreilly.com/%23_setting_up_a_web_server) from Scott Murray's book, Interactive Data Visualization for the Web.

All of the code files for the course can be downloaded as a zip file from the resources tab.

As a reminder, you will need to run local server to render the visualizations. You can start a local server using Python. Navigate to the directory that contains all of the files and then type python -m SimpleHTTPServer in the command line. If you type localhost:8000 into the address bar of your browser, then you should see the files that you can display in the web page.

If you're in an environment running Python 3 instead of Python 2, you can use python -m http.server to start up the local server instead.

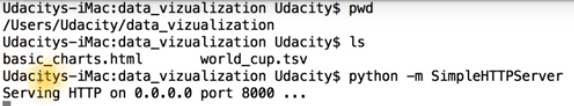
**A Simple HTTP Server:**

First we have to check the present working directory.

We have to see the list of variables.

We have to start the server.

D3 uses AJAX to load external files and our HTML file.

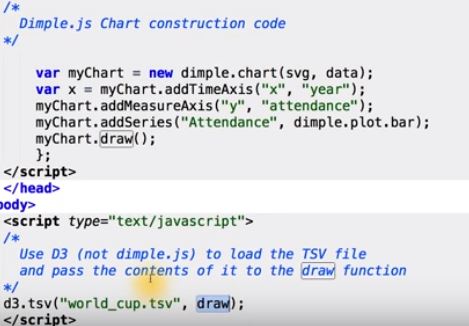


**Dimple Bar Chart Code Overview:**

We are loading dimple.js and d3. We know the draw function.



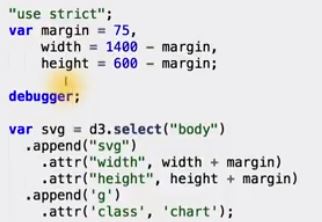
To load tab separated value.



**Java Script Debugger:**

Actually working from Chrome’s Debugger JavaScript console, once the AJAX request has finished, and my data has been loaded.

To do this, I can put a debugger statement inside the draw a callback function.



The debugger is used inside the draw function.

* When chrome loads the page, it first call our d3.tsv function to load world\_cup.tsv.
* Once the AJAX request has returned, it will pass data file in a JavaScript object to our draw a callback function.
* Once our callback function is called, code will execute until it hits the debugger statement, in which execution would stop, and it will put us into the chrome debugger at this exact line.

**Inspecting Variables and Data:**

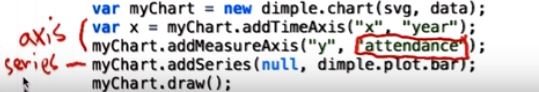
You have to first open chrome java script console. On mac It is done by cmd+option+I or by simply navigating to the toolbar, looking under the tools menu, and going to JavaScript console.

If we want in table format:



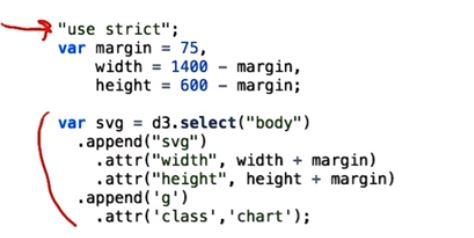
**Dimple Axes, Series and Fields:**

I will try is a bar chart. To do with the dimple there are few steps to go through, but many less than doing this with d3 itself.



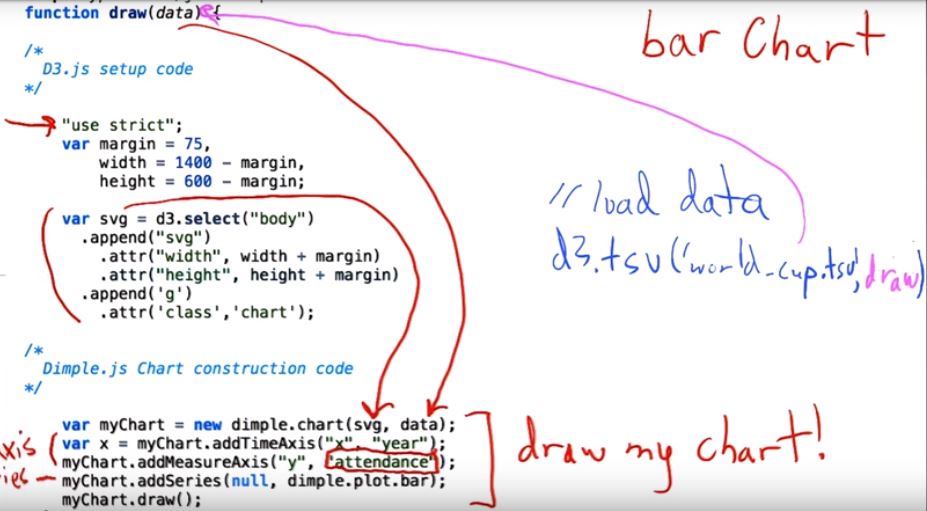
**Chart Layout and Group Tags:**

Dimple.js’s API is much closer to what the grammar of graphics is specifies than d3 itself. This is because dimple.js is at higher level abstraction than d3.



**Creating and Drawing the Chart:**

The Dimple code to draw bar chart is only 5 lines.

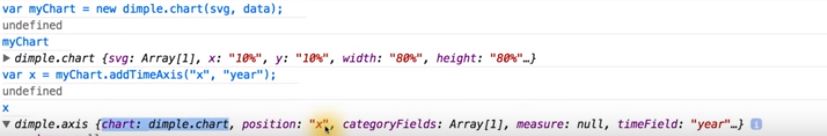


**The Benefits of Dimple:**

Using dimple will not only allow us to iterate much quicker on our visual design process.

**Inspecting the X-axis:**

The x axis is time.



**Adding the Series and Drawing the Chart:**



**Object Oriented Design and Charts:**



**Changing the Chart Type:**



The bar graph is changed to scatter plot.

**EDA vs Sketching Data Visualizations:**

Eda is performed when you are exploring the structure and values of the data to find

* Insights
* Erroneous values
* Structure
* yourself

Sketching in contrast is performed to experiment with

* Visual layout
* Visual encodings
* Others

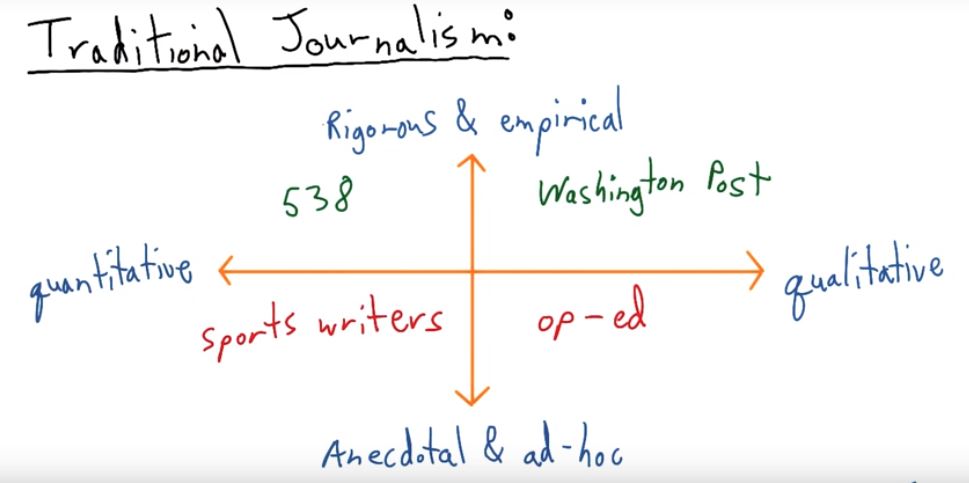
**Narrative Structures:**

**Finding Stories:**

In this we are going to learn narrative structures in addition to it we are going to learn about the data collection process and data processing.

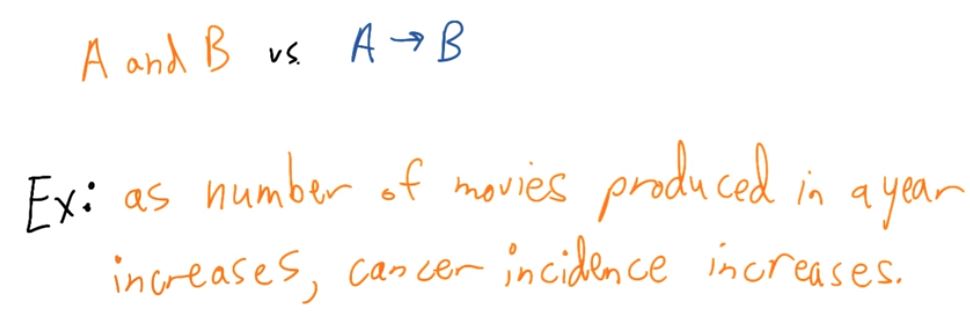
**Approaches to Journalism:**

Before we dive into techniques to add context to our visualization, and see how to create a narrative around our data. First we have to know about traditional journalism takes when treating its stories.



**Correlation vs Causation:**

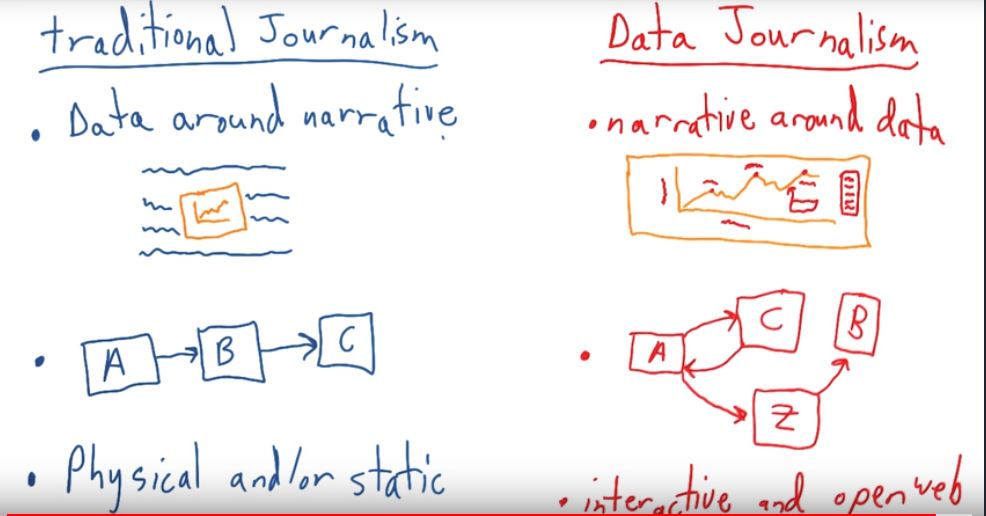
One of the most common reasons for incorrectly interpreting data stems from the difference between correlation and causation.



This confusion is mainly by

* Serious consequences
* Misleading conclusions

**The New Journalism:**



**Getting Data:**

<http://datajournalismhandbook.org/1.0/en/getting_data.html>

Scott explained how about 90% of the time spent creating a data visualization is actually spent "finding the data, verifying the data, parsing the data, filtering the data, and exploring the data." This is certainly true in the context of data science and when trying to answer questions with data. Fortunately, there are many resources at your disposal to tackle those important tasks.

The link in the title text, "Getting Data," has excellent tips for tracking down data. We encourage you to review two of the chapters: <http://datajournalismhandbook.org/1.0/en/getting_data_0.html> and <http://datajournalismhandbook.org/1.0/en/getting_data_3.html> . You might also want to bookmark the "Getting Data" resource and come back to it at a later time.

When it comes to verifying data, it is up to you to know the data inside and out, including the data's source, collection methods or other data generating processes. Scott Murray will discuss an example of this in the next series of videos covering objective and subjective data visualizations.

If you've ever tried cleaning up a messy data set, you know that data processing and manipulation can be cumbersome. Tools such as

<http://openrefine.org/>

<https://www.google.com/sheets/about/>

<http://ipython.org/notebook.html>

<http://pandas.pydata.org/>

<http://nbviewer.jupyter.org/github/fonnesbeck/pytenn2014_tutorial/blob/master/Part%201.%20Data%20Wrangling%20with%20Pandas.ipynb>

<https://www.r-project.org/> can help you parse data and get the data you need to create visualizations.

Once you have the data you need, you're ready to create a visualization. In the next series of videos, Chris will share common pitfalls and how subjective data visualizations can mislead readers.

## Types of Bias

The previous data visualizations that Cole and Matt shared contain bias. There are three types of bias of which you should be aware when constructing graphics: **author bias**, **data bias**, and **reader bias**.

### Author Bias

Cole and Matt's data visualizations contain author bias. That is, the designers and presenters of the visualizations (knowing or unknowingly) misrepresented data through visual encodings or other design choices such as the chart type. If you'd like to use a 3D Pie Chart in the future, please remember Andy Kriebel's statement "Friends Don't Let Friends Use Pie Charts".

As the designer or presenter of data visualizations, your design choices should establish trust between the reader and the graphic. Your design choices should facilitate communication. Otherwise as Cole mentioned, you risk the overall credibility of your message among readers.

### Data Bias

Before you learn about data bias, we'd like you to watch a video in which Scott answers the same question as Cole and Matt. Scott takes a different approach to what he considers to be a "subjective" data visualization.

### Reader Bias

This type of bias will be covered after a few videos.

**Personal Data Visualization:**

So there is a new project, the website is aprilzero.com. And it’s really in depth sort of personal data reporting platform. And the interface is really beautiful.

**Data Bias:**

Data bias arises from the process of collecting data. Systematic measurement errors or faulty devices can bias raw data values, and selection bias can lead to subgroups that are not representative of the population of interest for a given question. Data bias and sampling methods are beyond the scope of this class; however, we encourage you to learn more about these topics. Try reading articles about <https://en.wikipedia.org/wiki/Data_collection> , [https://en.wikipedia.org/wiki/Sampling\_(statistics)#Sampling\_methods](https://en.wikipedia.org/wiki/Sampling_(statistics)%23Sampling_methods) or <https://en.wikipedia.org/wiki/Bias_(statistics)> .

**Personal Data Visualization:**

Technology has changed how we track and share our personal data. Watches, phones, customer cards, and other gadgets allow us to share our movements, calories, and spending habits on a daily basis. In the next video, Scott shares a different example of personal data visualization. The example is extremely personal and captures the life of a single human being.

**Reader Bias**

The final type of bias that we'll cover is reader bias. Reader bias encompasses any preconceived notions or assumptions that a reader brings to interpreting a visualization.

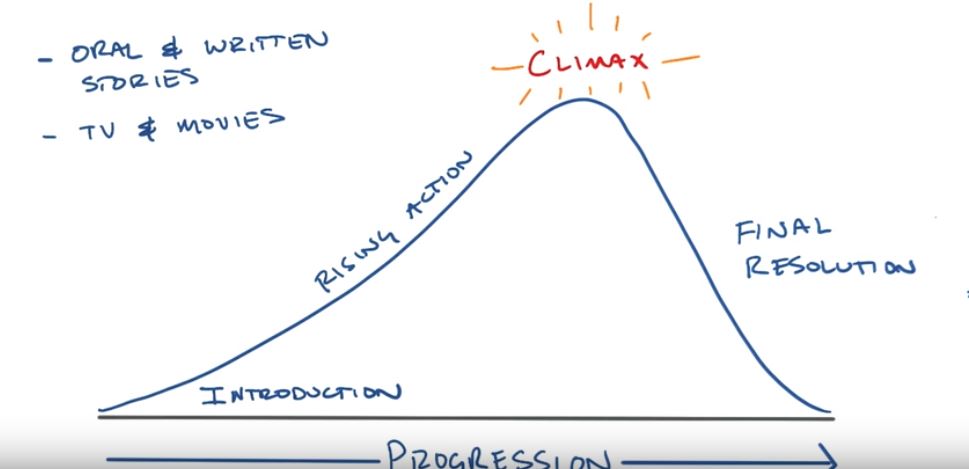
The assumptions may pertain to a reader's domain knowledge or the topic of the data visualization. For example if you know a little bit about the World Cup, you are more likely to be aware of the tournament's occurrence and the structures of the World Cup's competition and stages.

Other assumptions may involve political, religious, or cultural beliefs or a reader's familiarity with a specific chart type. You should always consider your audience's background and familiarity with graphics when designing a data visualization.

Both the designer (encoder) and the reader (decoder) ought to be aware of bias. Communication can go awry due to the designer's choices or the reader's interpretation of a graphic. This blurs the line of misleading and lying in data visualization, and it is with analytical thinking and due diligence that both designer and reader can partake in a trusting exchange of information.

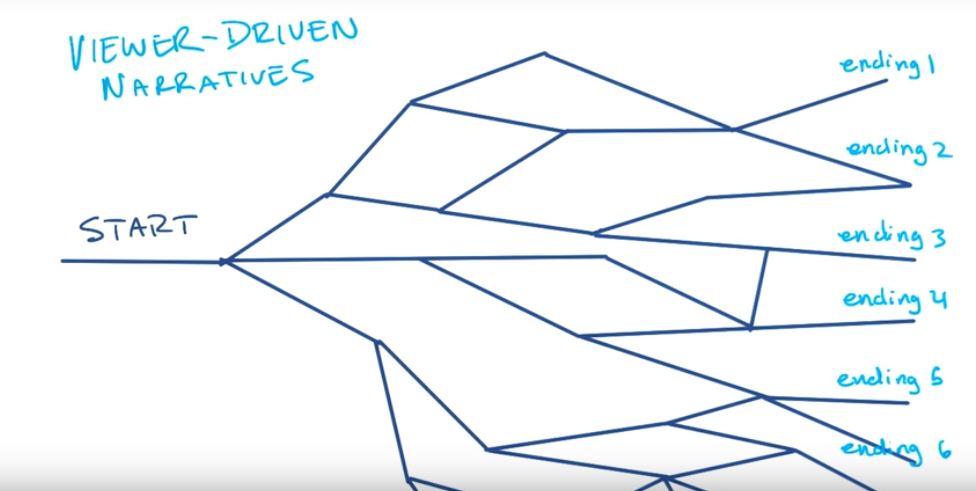
For more examples of author and reader bias, please read <https://visualisingadvocacy.org/blog/disinformation-visualization-how-lie-datavis> by Mushon Zer-Aviv.

**Different Types of Narrative Structures:**



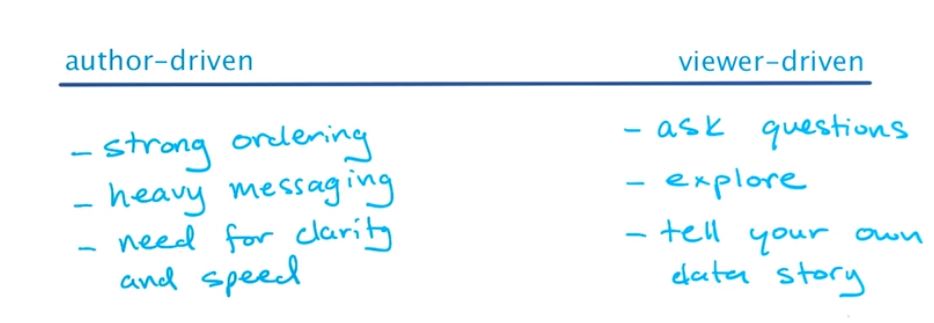


It is a linear flow.



The viewer can expect many types of ending.

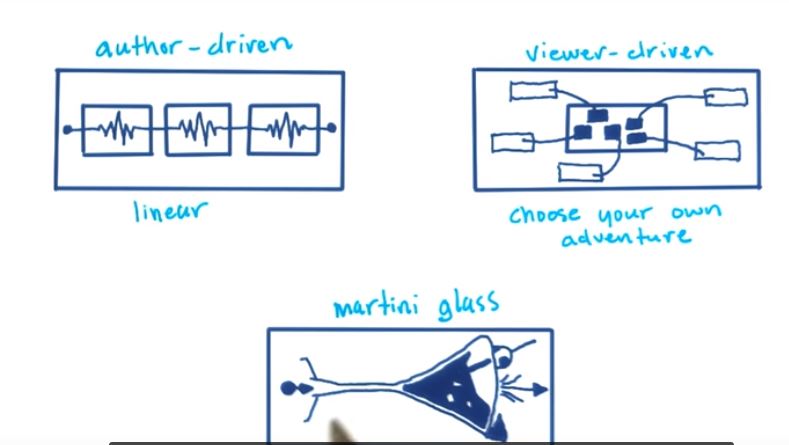
Visual narratives often called narrative visualizations, combining conventions of communication and explanatory information visualization to convey an intended story.



**The Martini Glass:**

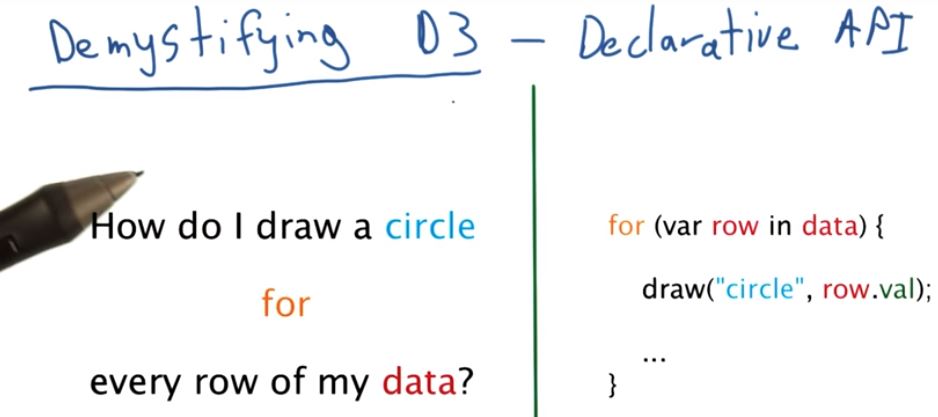
There’s a third narrative structure, common in more complex data visualizations called martini glass. It is called martini glass because it is unique in shape.



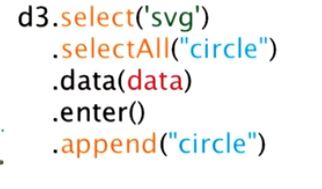


**Annotations and World Cup Attendance Chart:**

****

**Demystifying D3: **

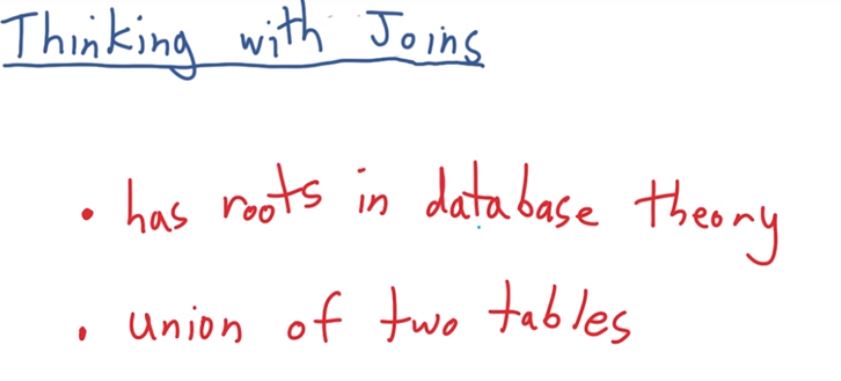
It is in java script.



In d3 the code for drawing a circle is in the above image.

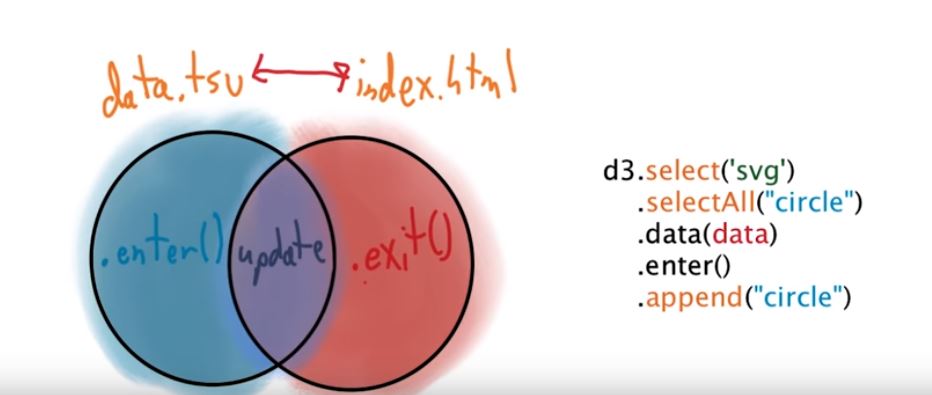
**Thinking with Joins:**

The key of d3 magic is in the selection before data binding, the data binding itself and selection after the data has been bound, in this case .enter is the special type of select which we then eventually add shapes or circle, SVG or HTML to our page for every element in the enter selection.

**Defining Join:** 

The idea of join has its roots in SQL.

**Thinking with Venn Diagrams:**



**Communicating Your Message:**



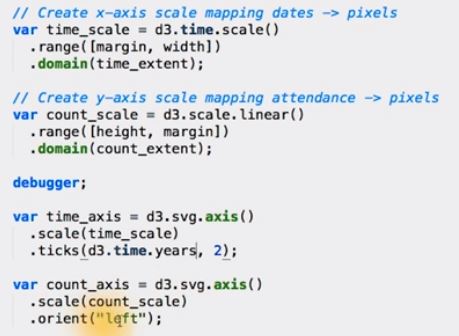
**Using extent () for Scales:**



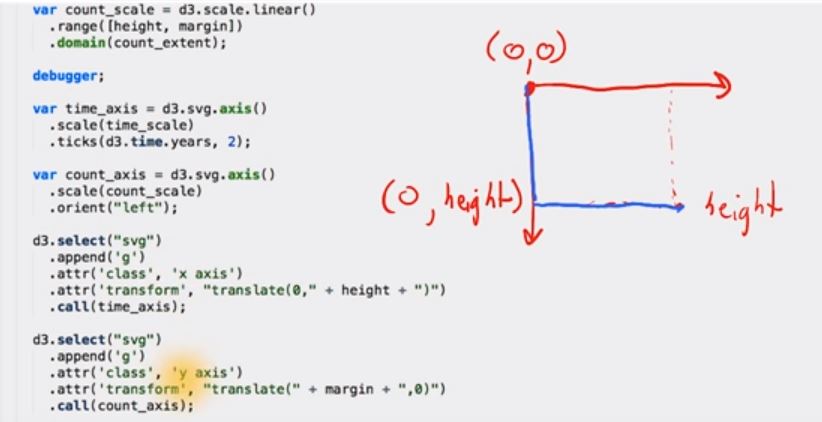
**Parsing Overview and Parsing Attendance:**

****

**Creating Axes:**

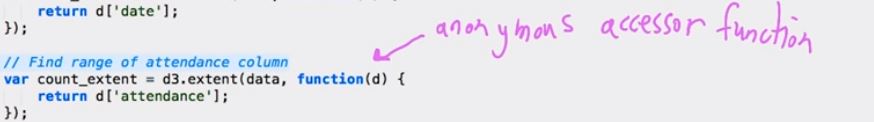
****

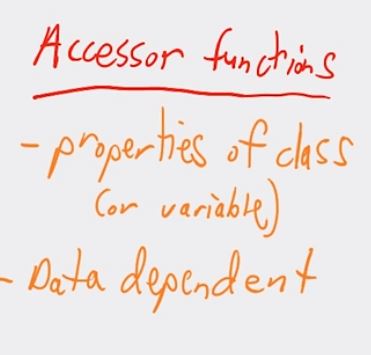
**Adding the axis to the page:**

****

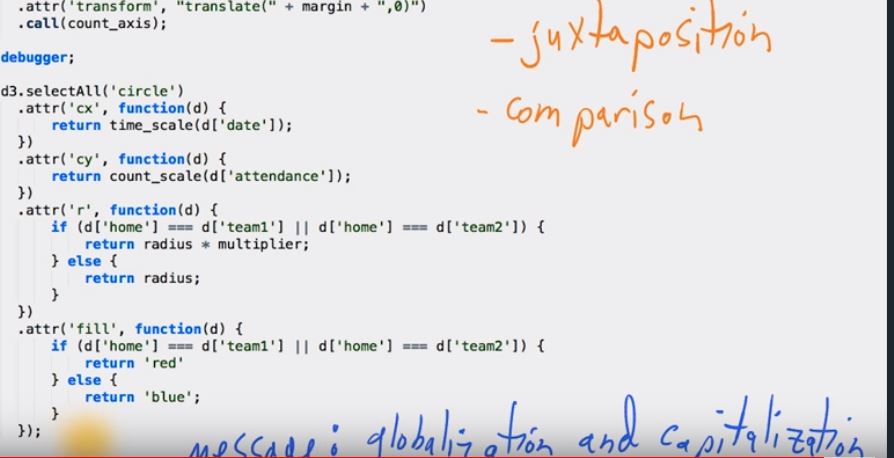
**The anonymous Accessor Function:**

Changes the behaviour of a generic D3 function.

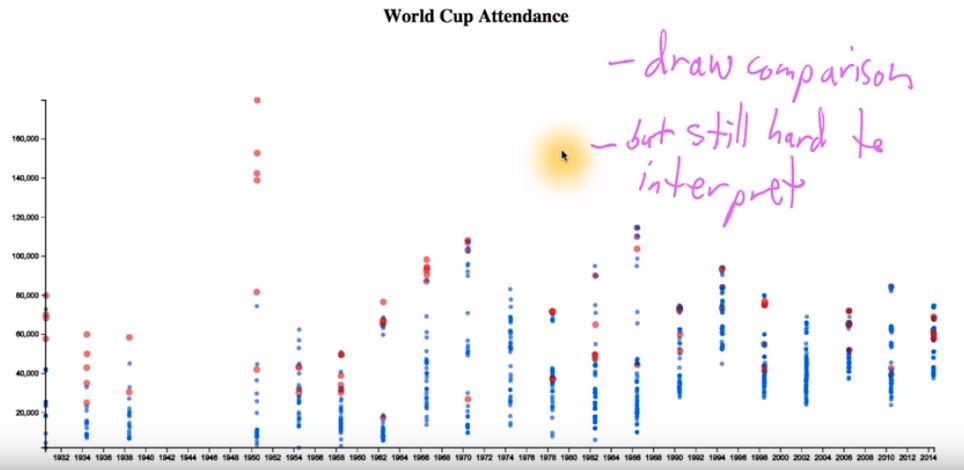


****

**Juxta position and Comparison:**



**Context for the Reader:**

****

**Animation and Interaction:**

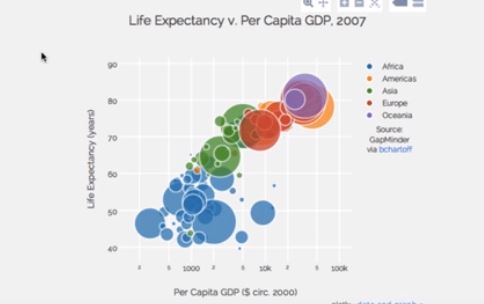
**Animation and Interaction:**

In this how d3 enables you to both animation and interaction to your visualization. What is possible by using geographic features to create a map with d3?

**The Benefits of Interactivity:**

It let you understand more about the actual data in the graph.

Ex:

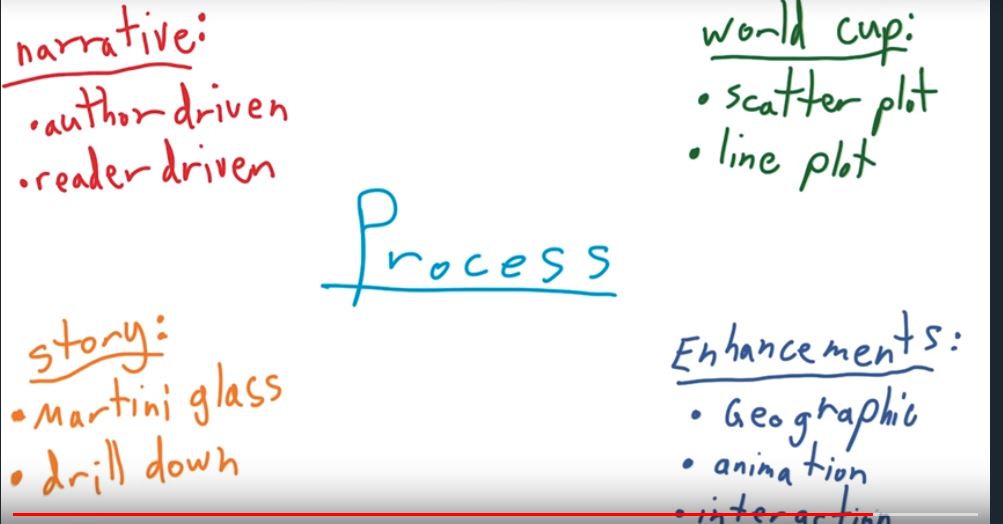


The actual data behind the graph is as follows:



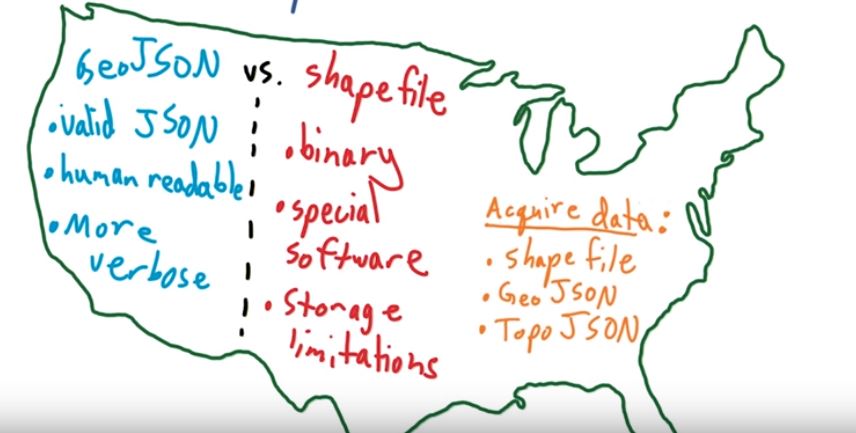
We can make use of any programming language.

**Interactive process:**



**Let’s make a Map:**

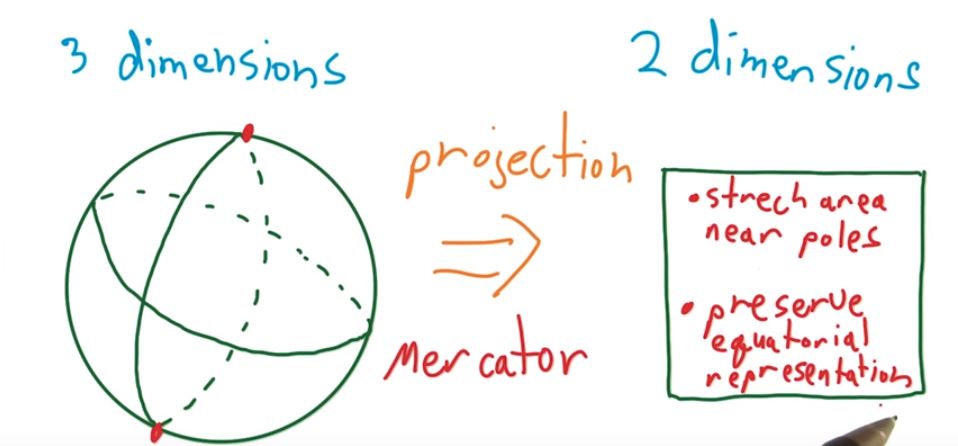
* The first step in the process of adding context is to create a map.
* You can create charts using other libraries.
* We would be hard pressed to find another tool that makes creating data – driven maps, as powerful and fun.



**What is a projection?**

In the scatterplot we use d3’s scale to go from our data domain to our pixel range. To convert our geographic coordinates to a pixel range we have to use a projection, or more specifically in our case, we will be using the Mercator projection.

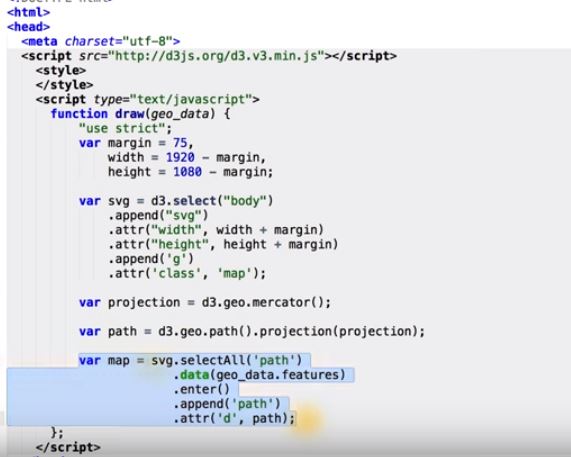




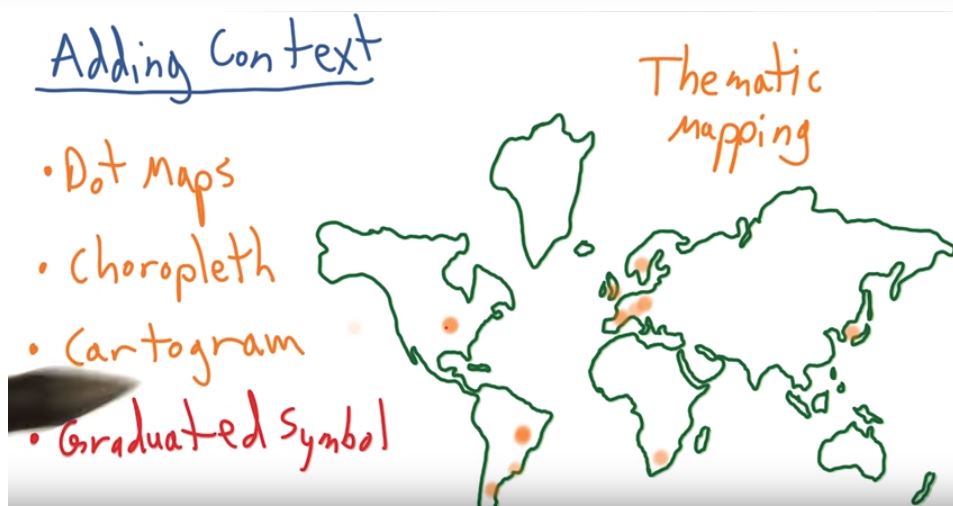
**Maps in D3:**



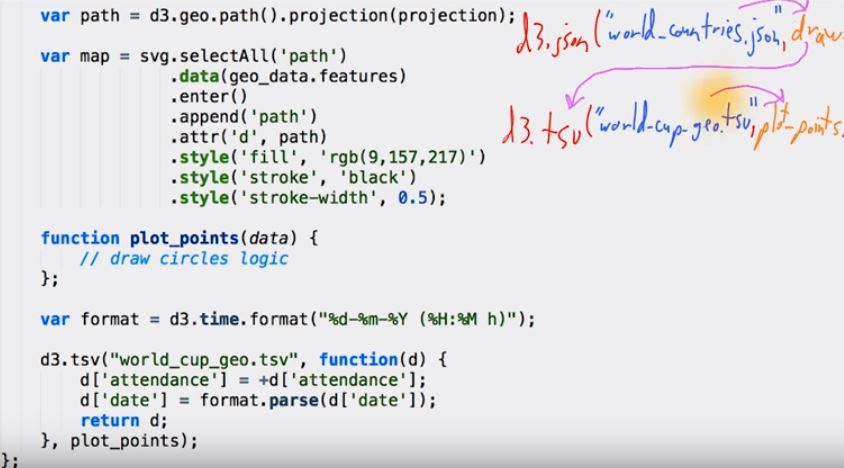
**Drawing a Map from SVG Path:**



**Thematic Maps:**



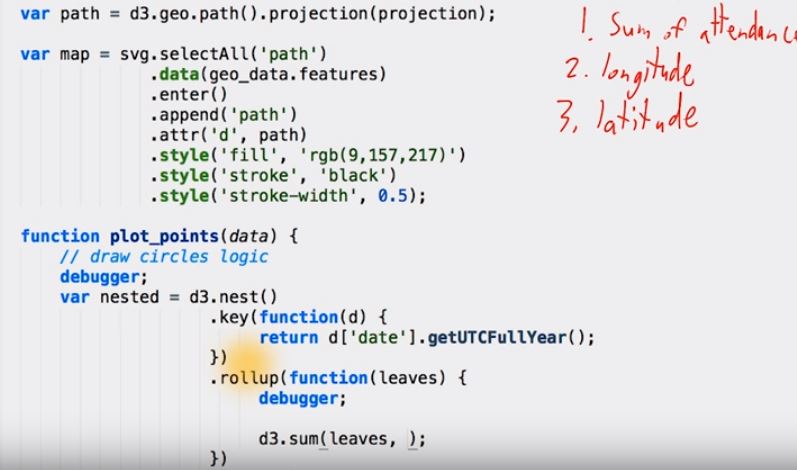
**Loading a Data with Nested Functions:**



**Nest Function:**

****

**Aggregating Data:**

****

## How to Lie with Circles

When working with circles or any other shapes for visual encoding, you should pay careful attention to data represented by **area** or **volume**. If you are not careful with the data and its visual encoding, then you can misrepresent the data, report exaggerated findings and, perhaps worst of all, lose trust with readers.

For example, take these two <http://images.huffingtonpost.com/2014-08-29-Donating.vs.DeathGraph.REVISED.jpg> . The original design was published in the article, <https://www.vox.com/2014/8/20/6040435/als-ice-bucket-challenge-and-why-we-give-to-charity-donate> by Vox media, and a corrected version of the graphic was later released. The article now shows the corrected version, the original graphic is <http://cdn3.vox-cdn.com/uploads/chorus_asset/file/663618/Donating.vs.Death-Graph.0.jpg> one.

Note: The following text appears at the bottom of the article.

**Correction**: In an earlier version of this article, the size of the graph's circles did not accurately reflect the data.

If you are not familiar with the Ice Bucket Challenge, please refer to the article. You may have seen YouTube videos from celebrities or even your own family members participating in the donation campaign.

In the <http://images.huffingtonpost.com/2014-08-29-Donating.vs.DeathGraph.REVISED.jpg>, you should notice that the circles in the original graphic are much larger than the circles on the corrected graphic on Vox's website.

Let's take this as a learning opportunity to understand how to accurately represent data values with a circle's area.

The problem in the original <http://cdn3.vox-cdn.com/uploads/chorus_asset/file/663618/Donating.vs.Death-Graph.0.jpg> is that the data values are used for the radius of the circles. If you use a data value for the radius of a circle, then the area of the circle will be about three times the **square** of the radius.

Area of Circle = π\*r2

or

Area of Circle ≈ 3.14\*r2

For example, a data value of 4 will create a circle with an area 16π.  
A data value of 5 will create a circle with an area of 25π.

We are effectively squaring the data values to create the new visual representation. This causes the circles to appear much larger than they should appear in the graphic.

### To avoid the problem, the data values should match the areas of the circles. You can take the square root of the data values to determine the radius of each circle.

Jonathan will explain how to do this using code in the next few videos. He will make use of an anonymous accessor function and d3.scale.sqrt ().

As an exercise, you should think about how you would redesign and improve the graphic. Feel free to share your ideas, sketches, or visualizations in the discussions.

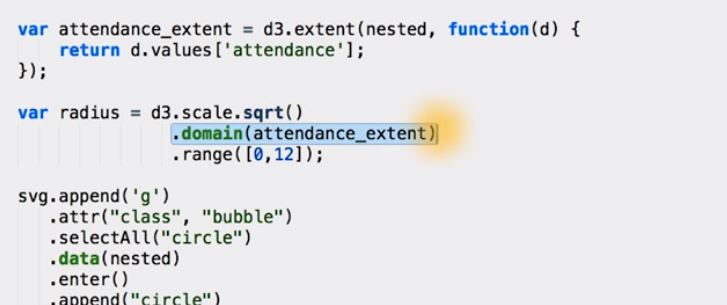
To read about additional improvements to the graphic and redesigns, see the related reading links below.

**Related readings:**

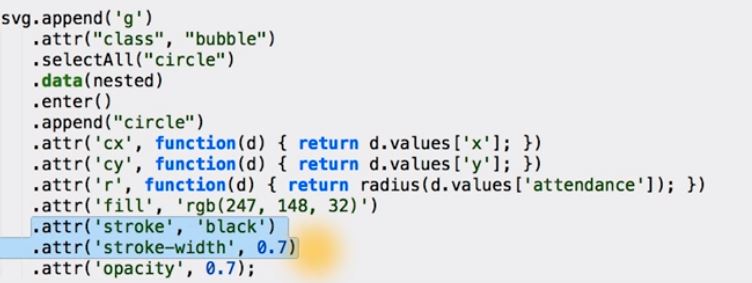
<https://www.huffingtonpost.com/randy-krum/false-visualizations-when_b_5736106.html>

<https://themendozaline.org/post/95757674381/this-bubble-chart-is-killing-me>

**The Radius Scale:**

****

**Adjusting the Map Aesthetics:**





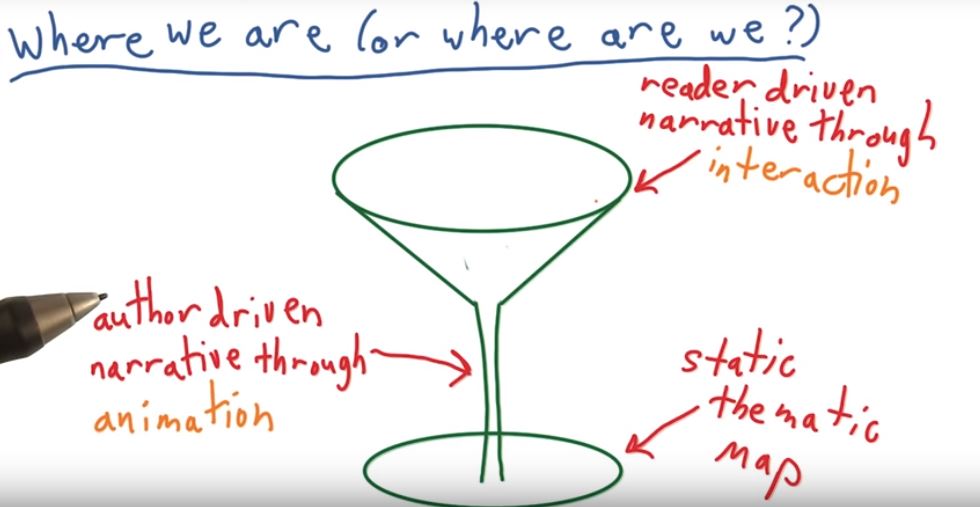
The circles are on the map which are created by the svg.

**Sorting the order for Drawing Order:**

****

The sort is done by using JavaScript.sort () function.

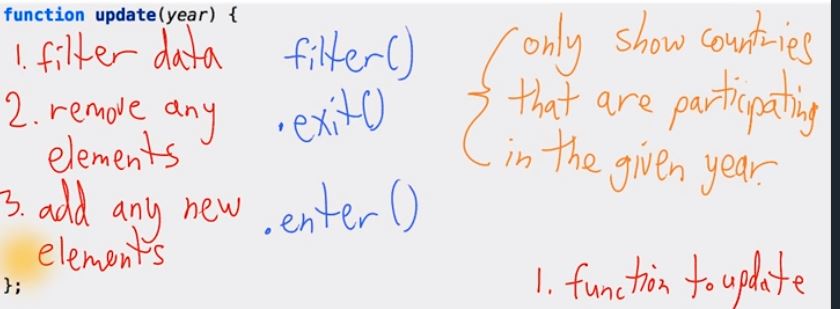
**Where are we?**

****

**The Update Function:**

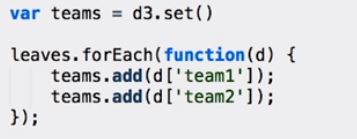
To animate our map over the years of the World Cup, we need two things. The first of which, is a function to update our map, since we will be repeatedly calling the updates we need to for every year, we want to encapsulate into function, that we can call easily whenever we want. And the second thing we need is, a way to cycle through all the years that the world cup was held, and pass those years to our update function.

**Outlining the Update Function:**



**Collecting participating Countries:**

The functions are adding below the aggregate function. The code is as follows:



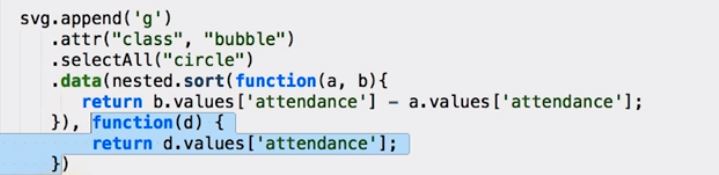
**Filtering:**

Our update function takes a single year as an argument, and with this year we have to filter our data. In this we are actually filtering nested object. The nested object has its key property set to the year.



**Data join with a Key Function:**

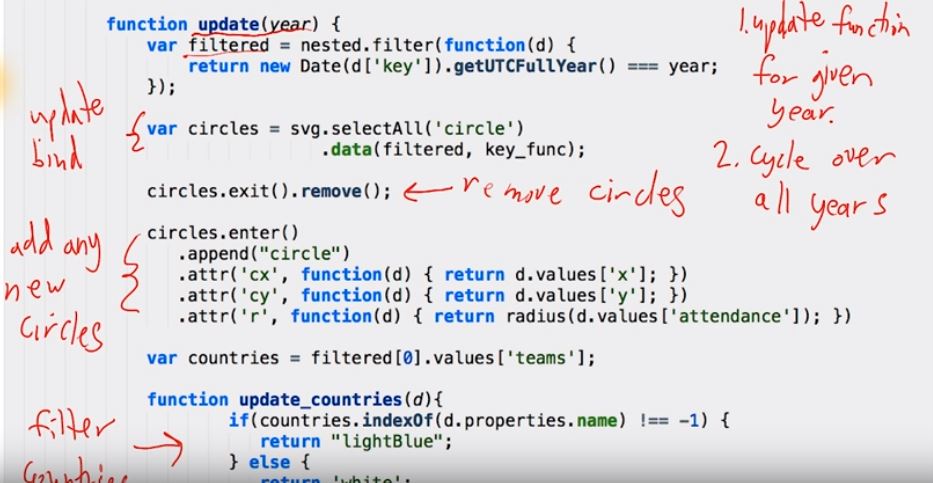
The d value in the function is any arbitrary value.



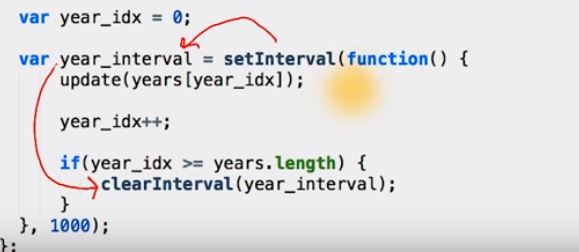
**Completing the Update Function:**



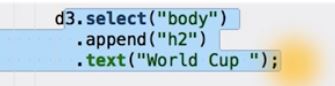
**Summary of the Update Function:**

****

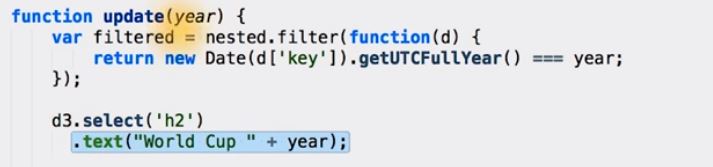
**Setinterval and clearinterval:**



**Update the Title:**

****

**Now the update function changes with the year.**

****

**Delaying the Displaying of buttons:**

The CSS code that Jonathan used to style the buttons is below. You can add it between the style tags at the top of the HTML file.

div.years\_buttons {

position: fixed;

top: 5px;

left: 50px;

}

div.years\_buttons div {

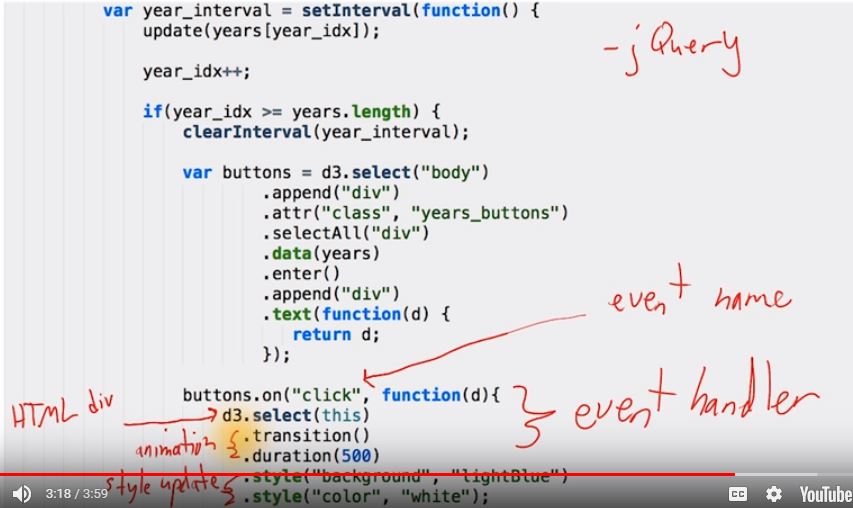
background-color: rgb(251, 201, 127);

padding: 3px;

margin: 7px;

}

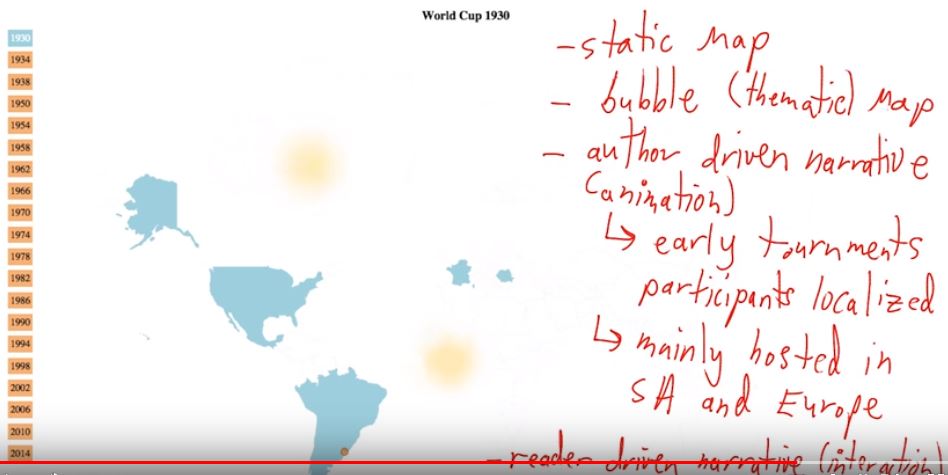
**Adding events to buttons:**

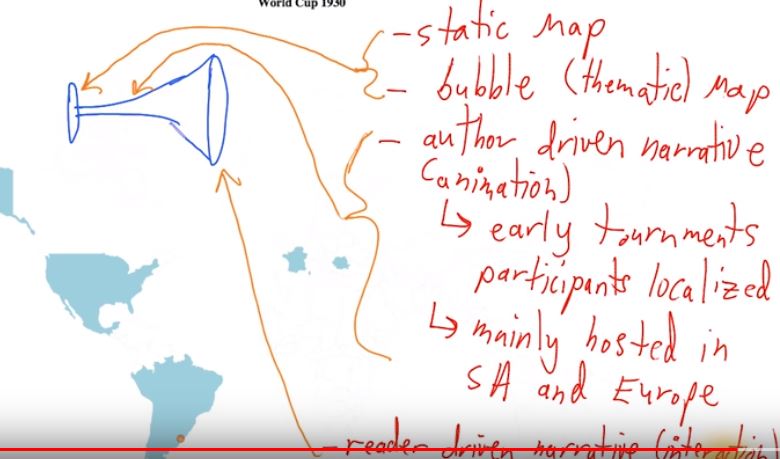
****

**Fixing the Buttons:**

****

**World cup code and narrative Recap:**

****

****

**The future of Data Visualization looks very bright.**

**Project:**

## Project Overview

For this project, you will create a data visualization from a data set that tells a story or allows a reader to explore trends or patterns. You will need to use either <http://dimplejs.org/> or <https://d3js.org/> to create the visualization. Your work should be a reflection of the theory and practice of data visualization, such as visual encodings, design principles, and effective communication.

## What do I need to install?

To work on your data visualization, you will need to start a local server on your computer. To start a local web server, you will need to have <https://www.python.org/downloads/> installed on your machine.

If you do not have Python installed on your machine, these instructions come from the <https://in.udacity.com/course/python-foundation-nanodegree--nd002-inpy> course.

Once you have Python installed, you can start a local web server and view your data visualization. Refer to the following <https://classroom.udacity.com/courses/ud507> to see how to do so.

Remember, you must start your web server in the top level directory to serve all code and data files. If you do not use this folder as the root directory for the web server, be aware that you will need to change the file paths.

There are other ways to start a local web server. To learn more about why you need to start a local web server and other ways of setting up a local web server, please read [https://www.oreilly.com/#\_setting\_up\_a\_web\_server](https://www.oreilly.com/%23_setting_up_a_web_server) from Scott Murray's book, Interactive Data Visualization for the Web.

**Why this Project?**

This project will touch on the overarching attitudes and beliefs important to effective data visualization, such as:

* visualization is a dialog
* showcasing and sharing visualization with others
* visualization is a fluid process that typically requires multiple iterations of improvements

You will have an opportunity to experience the end-to-end process of creating effective data visualizations and highlighting important information from data that may otherwise be hidden or hard to uncover.

**What will I learn?**

After completing the project, you will be able to:

* Demonstrate the ability to choose optimal visual elements to encode data and critically assess the effectiveness of the visualization
* Communicate a story or finding to the appropriate audience using interactive visualizations
* Undergo the iterative process of creating a visualization, and build interactive visualizations with dimple.js or d3.js.

## Why is this Important to my Career?

Data analyst are storytellers that can translate data findings that other people can easily understand. They view data visualization as an important form of communication.

If you, as a data analyst, can create visualizations to explore data, articulate clear findings to drive business decisions, or use data to elicit consensus from diverse perspectives, then you will be a deeply invaluable member on your team.

### Introduction

For the final project, you will create a data visualization from a data set that tells a story or allows a reader to explore trends or patterns. Your work should be a reflection of the theory and practice of data visualization, and you must use either dimple.js or d3.js.

We will provide some options of data sets to explore; however, you may choose to explore an entirely different data set. You should be aware that finding your own data set and cleaning it using Python, R, or some other language can take considerable time and effort. This can add as much as a day, a week, or even months to your project so embark on the adventure to find and clean a data set if you are truly prepared with programming and data wrangling skills.

There are **three difficulty** levels to this project, and you should choose an appropriate level depending on your experience with data munging and exploratory data analysis. The difficulty level you choose will not affect the evaluation of the project.

* **Beginner** - Choose this option if you have no experience with cleaning data or exploratory data analysis.

Select one of the beginner data sets, which already has a summary of findings, from the <https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub> (dataset options) document. Then, create a visualization that communicates the findings.

* **Intermediate** - Choose this option if you have some experience cleaning and analysing data.  
    
  Select one of the intermediate data sets from the [https://docs.google.com/document/d/1w7KhqotVi5eoKE3I\_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub](https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub%20) document. These data sets are not necessarily clean or tidy data sets. You will investigate the data set to share a story or message about the data and then create a suitable visualization.
* **Advanced** - Choose this option if you are comfortable finding, cleaning, and analyzing a data set.

Find a data set, investigate it, and share your findings in a visualization. Your final graphic should primarily be explanatory, but it may also contain exploratory components. You can find a list of recommended websites to find data sets in the **above** document. You should be aware that finding your own data set, cleaning the data set, and analyzing it (using R, iPython Notebook, or another tool) can take considerable time and effort. This can lengthen the time you spend on your project by days, weeks, or even months. Choose the option only if you feel prepared for a challenge!

Now, on to the details!

### Step One - Choose a Data Set

First, you will choose a data set from the [**Data Set Options**](https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub) document or find a data set to explore and visualize. You should choose a data set based on your prior experiences in programming and working with data. The data set you choose will not increase or decrease your chances of passing this project.

### Step Two - Get Organized

Eventually you’ll want to submit your project and share it. If you are familiar with [**https://github.com/**](https://github.com/) we encourage you to create a public repository or a public [**https://gist.github.com/discover**](https://gist.github.com/discover) (GIST) for your project to track changes. Otherwise, you need to create the following files.

1. an **index.html** file containing the code to create your visualization (you may include the JavaScript and CSS in this file or separate them in other files)
2. a **README.md** file that includes four sections...

* **Summary** - in no more than 4 sentences, briefly introduce your data visualization and add any context that can help readers understand it
* **Design** - explain any design choices you made including changes to the visualization after collecting feedback
* **Feedback** - include all feedback you received from others on your visualization from the first sketch to the final visualization
* **Resources** - list any sources you consulted to create your visualization

1. **data files**

* the final data set used to create the visualization (usually .csv, .tsv, or .json file)
* a codebook or other files related to the data set (description, readme, license)

1. **OPTIONAL FOLDERS IF YOU USE** **GitHub**

* **data** folder to include all the data related files
* **js** folder to include .js files (not needed if javascript is in the index.html file)
* **css** folder to include .css files (not needed if CSS is in the index.html file)

### Step Three - Find a Data Story

Explore your data set and craft a message or story around your data! Think about the overall message you want to convey and think about the comparison(s) or relationship(s) you want your readers to see.

### Step Four - Create Your Visualization

First, sketch ideas for your visualization. Once you settle on a sketch, explain any design choices in that sketch, such as chart type, visual encodings, and layout, in the **Design** section of the **README.md** file. Then, write code to create your visualization using either dimple.js or d3.js. The visualization must include animation, interaction, or both.

### Step Five - Get Feedback

Share your visualization with **at least 3 other people** and document their feedback. There are many ways to get feedback, and more feedback is generally better! Here are some options.

* Share your visualization with others in person and have them think aloud as they read and explore the graphic so you can document what stands out to them and how they interpret the graphic.
* Share a link to your repository in the discussions and ask others to share constructive criticisms. Be sure to offer advice to others who are seeking feedback too!
* Create and share a [**Gist**](https://gist.github.com/), which contains an **index.html** file, data file, and any .js or .css files). Directions for creating and sharing a Gist can be found at <https://bl.ocks.org/> .

**Box Plots Gist EXAMPLE**:

* <https://gist.github.com/mbostock/4061502>
* <https://bl.ocks.org/mbostock/4061502>

You might want to ask specific questions when you collect feedback. Here are some questions to help you. You can, of course, ask others.

* What do you notice in the visualization?
* What questions do you have about the data?
* What relationships do you notice?
* What do you think is the main takeaway from this visualization?
* Is there something you don’t understand in the graphic?

### Step Six - Document Feedback and Improve the Visualization

For each person that gives you feedback, add the person’s feedback to your **README.md** file in the **Feedback** section. As you improve and iterate on your visualization, update your code **AND** describe any changes in the **Design** section of the **README.md** file. You should be building evidence that you have shared your visualization, received feedback, and responded to that feedback. You will need to submit the different versions of your visualization. At the least you need to submit the initial version of the index.html file and the final index.html file.

**Another Tool:**

If you want to use the tool for visualization, you can go through below link.

[https://classroom.udacity.com/courses/ud1006](https://classroom.udacity.com/courses/ud1006%20)