|  |
| --- |
| **D3.js** |
| I2 Decisions |
|  |
|  |
|  |
|  |
|  |
|  |

Table of Contents

[2 Data Driven Documents for beginners 3](#_Toc463537816)

[2.1 Why d3.js? 3](#_Toc463537817)

[2.2 Pre-Requirements 3](#_Toc463537818)

[2.2.1 What do you need to get started? 3](#_Toc463537819)

[2.2.2 Powerful visualization elements of D3 3](#_Toc463537820)

[2.2.3 Where to work? 3](#_Toc463537821)

[2.2.4 Getting to d3 4](#_Toc463537822)

[2.3 Where to start the d3.js code? 5](#_Toc463537823)

[2.3.1 Text editor 5](#_Toc463537824)

[2.4 D3.js source file setup 6](#_Toc463537825)

[2.4.1 D3.v2.min.js source file 6](#_Toc463537826)

[2.5 How to start code? 7](#_Toc463537827)

[2.5.1 Code structure 7](#_Toc463537828)

[2.6 Basics of d3.js 7](#_Toc463537829)

[2.6.1 HTML – HyperText Markup Language 7](#_Toc463537830)

[2.6.2 CSS – Cascading Style Sheet 7](#_Toc463537831)

[2.6.3 Java Script 8](#_Toc463537832)

[2.6.4 SVG – Scalar Vector Graphics 8](#_Toc463537833)

[2.6.5 DOM – Document Object Model 9](#_Toc463537834)

[2.6.6 Setting up the margin and graph area 9](#_Toc463537835)

[2.6.7 Adding SVG elements 10](#_Toc463537836)

[2.6.8 D3.append function 10](#_Toc463537837)

[2.6.9 Attribute function 11](#_Toc463537838)

[2.6.10 D3 chaining function 11](#_Toc463537839)

[2.6.11 Setting up the axis 11](#_Toc463537840)

# Data Driven Documents for beginners

## Why d3.js?

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3’s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

You should use D3.js because it lets you build the data visualization framework that you want. Graphic / Data Visualization frameworks make a great deal of decisions to make the framework easy to use. D3.js focuses on binding data to DOM elements.

D3.js is written in JavaScript and uses a functional style which means you can reuse code and add specific functions to your heart's content. Which means it is as powerful as you want to make it. How you chose to style, manipulate, and make interactive the data is up to you.

## Pre-Requirements

### What do you need to get started?

Essential to learn d3.js are,

* HyperText Markup Language (HTML)
* JavaScript
* Cascading Style Sheets (CSS)
* Web Servers
* PHP

Below are the links useful to learn the pre-requirements,

* <http://www.w3schools.com/html/>
* <http://www.w3schools.com/js/>

### Powerful visualization elements of D3

* SVG
* DOM

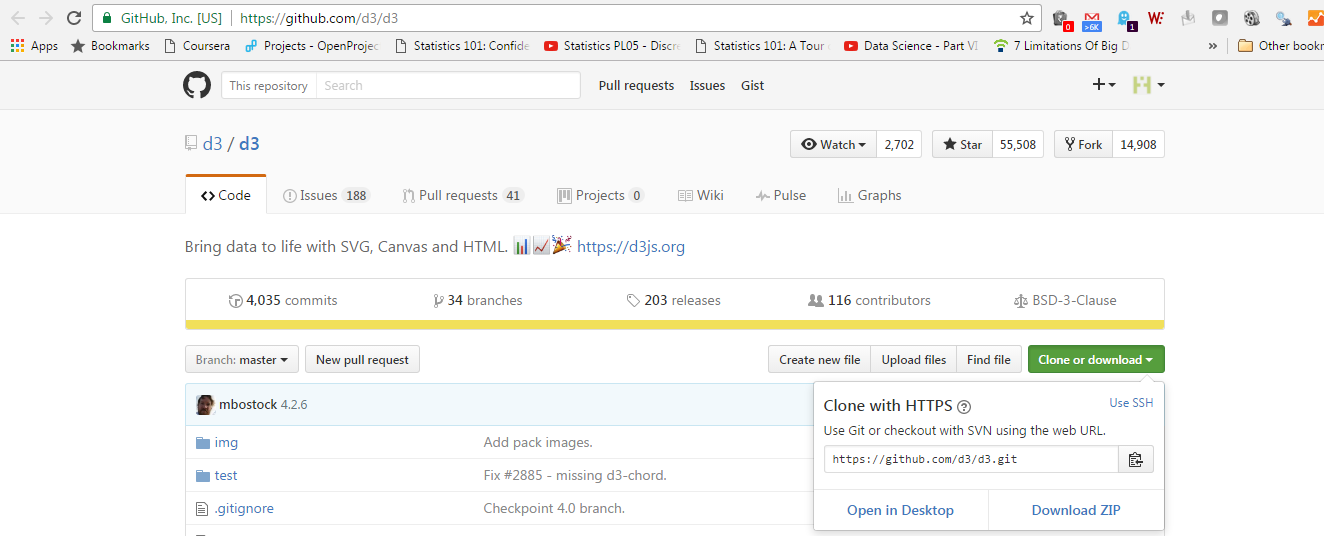
### Where to work?

* Text editor
* Web server (xampp, wamp)
* Browser to run the file

### Getting to d3

D3 has made huge advances in providing an extensible and practical framework for manipulating data as web objects. At the same time there has been significant increase in information available for people to use it. The following is a far from exhaustive list of sources, but from my own experience it represents a useful subset of knowledge.

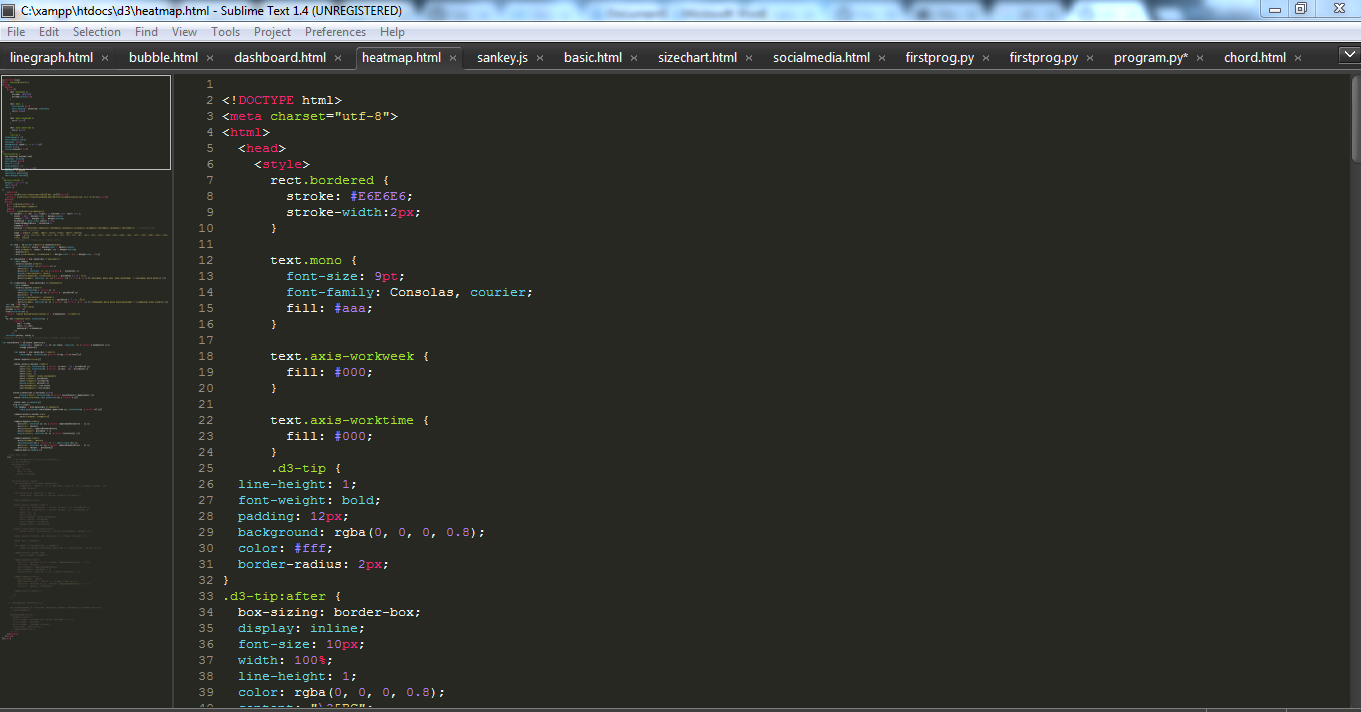
<https://github.com/d3/d3>



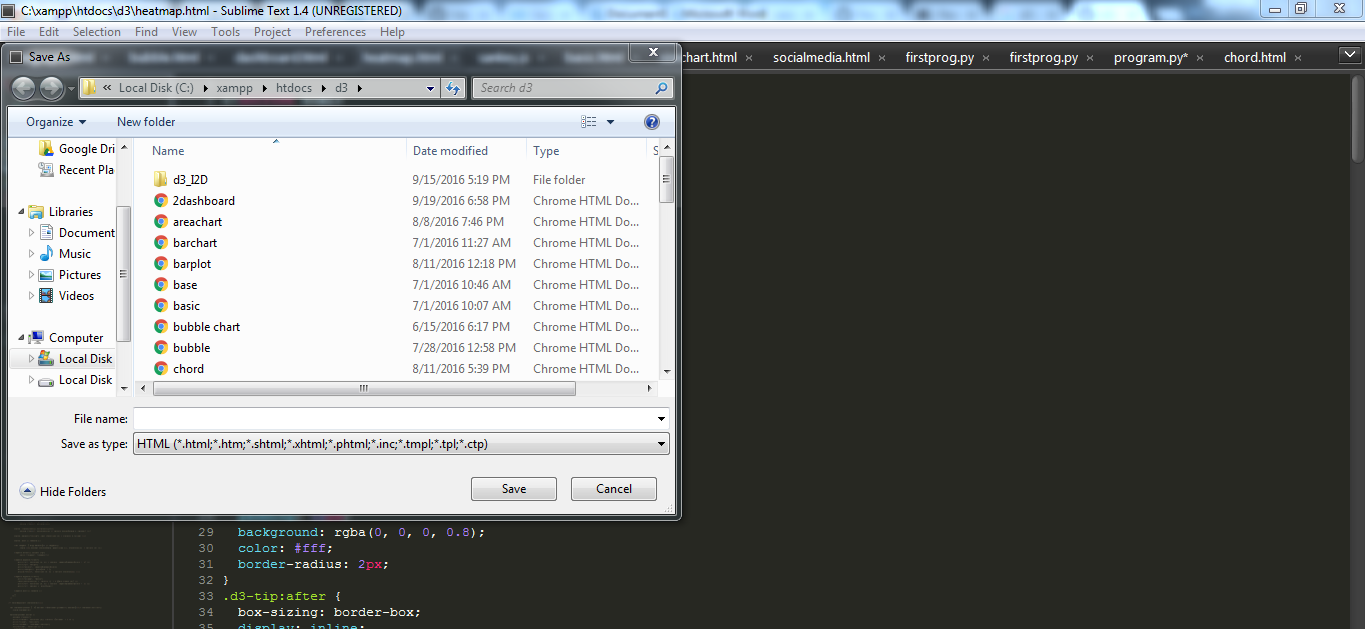
Download the zip file

## Where to start the d3.js code?

### Text editor



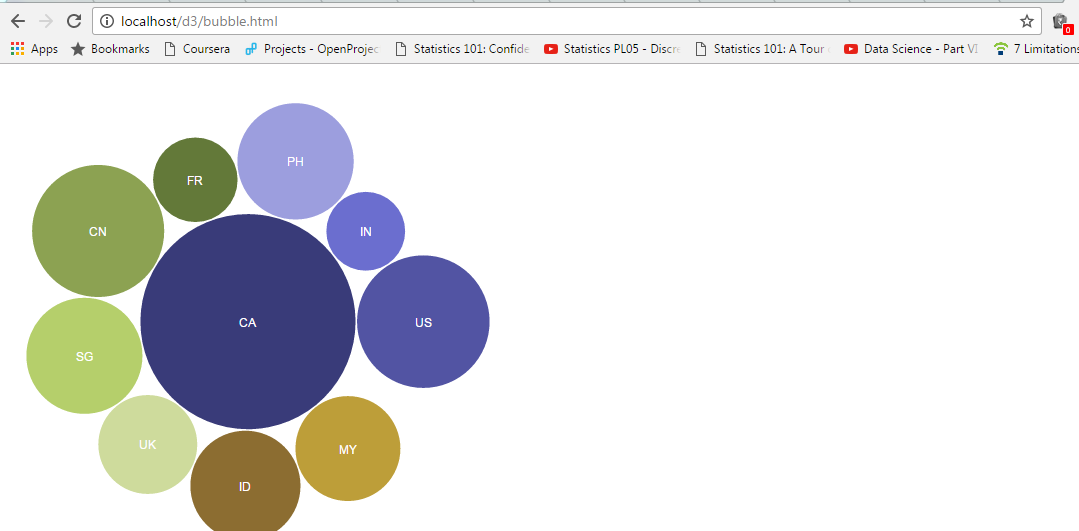
#### Save the edited file



Save as html file

#### Run the file

Run the file in browser

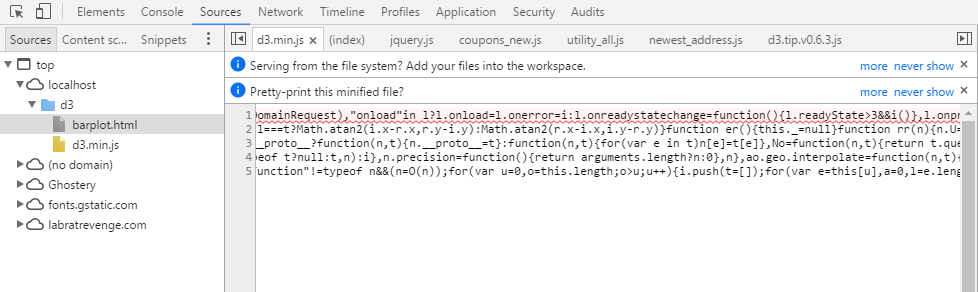


## D3.js source file setup

D:\d3\d3 source.PNG

The file d3.v2.min.js is the source file for the d3 code and it have to saved in the same folder as the HTML file so that it can be referenced it easily. We reference the JavaScript file from the head of the HTML file.

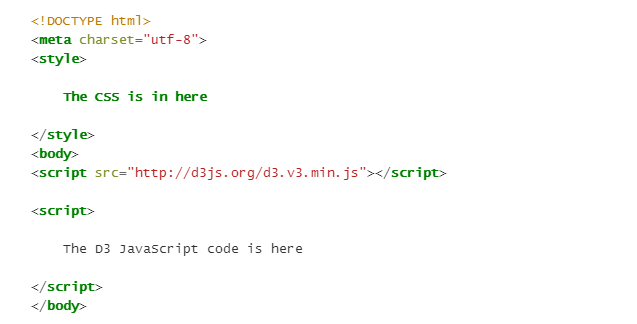
### D3.v2.min.js source file



## How to start code?

### Code structure

HTML portion of the code



D3.JS script portion

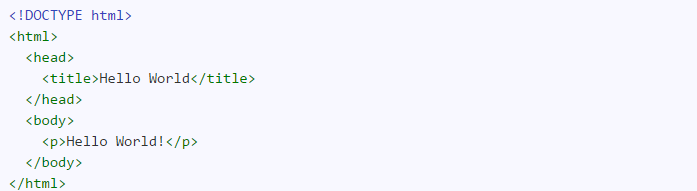
CSS portion

## Basics of d3.js

### HTML – HyperText Markup Language

The Hypertext Markup Language (HTML) resource is the main markup language for displaying web pages. HTML elements are the building blocks of the HTML web page. The elements consist of a pair of tags (starting and ending tags) and the textual or graphical content inside of the tags.

#### Example HTML code



### CSS – Cascading Style Sheet

The Cascading Style Sheet (CSS) resource is the style sheet language used for describing the presentation of the document. The presentation includes both the look as well as the formatting. CSS can be applied to HTML, XML, and most importantly for D3.js Scalable Vector Graphics (SVG). CSS formatting can be applied to html elements as an attribute.

#### Example CSS code

Style tag used to write CSS code.

D:\d3\css style.PNG

Calling CSS in script

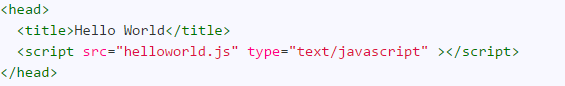
Calling CSS inside the script

D:\d3\css.PNG

### Java Script

[JavaScript](http://en.wikipedia.org/wiki/JavaScript) is what’s called a ‘scripting language’. It is the code that will be contained inside the HTML file that will make D3 do all its fanciness. In fact, D3 is a JavaScript Library, it’s the native language for using D3.

JavaScript is a programming language used to give sophisticated functionality to webpages before, during and after they load. It can also be used for the server. For D3.js purposes, we will use it to write functions to interface with the data and D3.js.



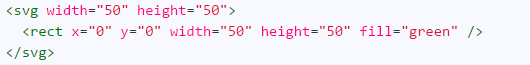
### SVG – Scalar Vector Graphics

Scalable Vector Graphics (SVG) is a family of specifications for creating two-dimensional vector graphics. Vector graphics are not created out of pixels. Vector graphics are created with paths having a start and end point, as well as points, curves and angles in between. Since Vector Graphics are not created out of pixels, they can be scaled up to larger or smaller sizes without losing image quality.

SVG images and their behaviors are defined in XML text files. The XML text can be included in an HTML document. This text means that images can be created and edited in a text editor. Also, since the DOM includes XML as part of the DOM specification, we can use the DOM Tree to access and update the structure, content and style of SVG Images. SVG comes up with the basic shape elements,

* Rectangle
* Circle
* Ellipse
* Straight Line
* Polyline
* Polygon

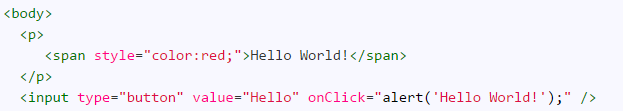
#### Example SVG element for rectangle



### DOM – Document Object Model

Document Object Model (DOM) is a convention for representing and interacting with objects in HTML, XML, and XHTML documents. The DOM is separated into three parts: Core, HTML, and XML. The DOM allows programs and scripts to dynamically access and update the structure, content, and style of a document. A DOM structured document has a DOM Tree which can be used to address, manipulate, and navigate the structure of the Document.

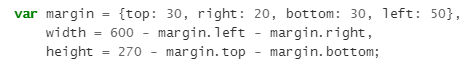
The structure of the DOM Tree is can be navigated in reference to relationships of its branches. This gives us descendant, ancestor, as well as parent, child and sibling. Thus we can specify a particular element in the DOM, look at the DOM tree, and select all of the elements above, below or along it easily.



The p element has a parent => body

p element has a sibling => input  
 p element has a child => span

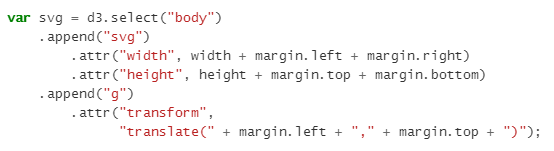
### Setting up the margin and graph area



The part of the code responsible for defining the canvas (or the area where the graph and associated bits and pieces is placed) is this part.

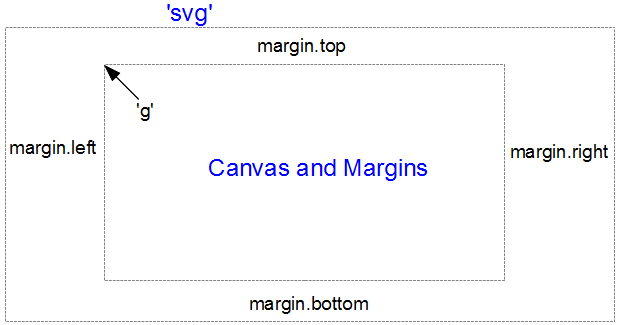
So there will be a border of 30 pixels at the top, 20 at the right and 30 and 50 at the bottom and left respectively. Now the cool thing about how these are set up is that they use an array to define everything.

### Adding SVG elements



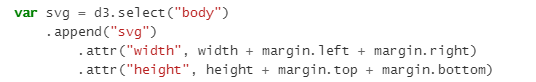
D3 needs to be able to have a space defined for it to draw things. When you define the space it’s going to use, you can also give the space you’re going to use an identifying name and attributes.

We also add an element ‘g’ that is referenced to the top left corner of the actual graph area on the canvas. ‘g’ is actually a grouping element in the sense that it is normally used for grouping together several related elements. So in this case those grouped elements will have a common reference.



#### D3.select function

* The first part of the JavaScript code that we wrote is **.select ("body")**.
* The D3.js Select method uses CSS3 selectors to grab DOM elements. D3 looks at the document and selects the first descendant DOM element that contains the tag **body**.
* Once an element is selected, D3.js allows you to apply **operators** to the element you have selected. These operators can get or set things like "attributes", "properties", "styles", "HTML", and "text content".



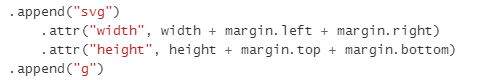
The above piece of code explains that, d3.select function select the body element. It will select the tags and elements in the body tag.

### D3.append function

The operator that we used is .append ("p").

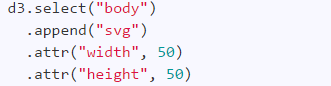
The Append Operator appends a new element as the last child of the element in the current selection.

Since our current selection was d3.select ("body"), the append operator added the "p" element last.



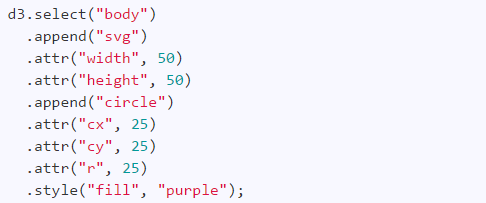
The above piece of code explains that it appends the SVG element with the currently selected body tag and group is appended with the SVG element.

### Attribute function



In d3 we can give different elements to a selected object using attribute function. The above piece of code explains that it gives the height and width to the selected SVG element

### D3 chaining function



D3.js chaining function is continuously we can add the functionality of d3.js in a single line. Order of the functions should be careful. It will execute from left to right.

### Setting up the axis

* [d3.axisTop](https://github.com/d3/d3-axis/blob/master/README.md#axisTop) - create a new top-oriented axis generator.
* [d3.axisRight](https://github.com/d3/d3-axis/blob/master/README.md#axisTight) - create a new right-oriented axis generator.
* [d3.axisBottom](https://github.com/d3/d3-axis/blob/master/README.md#axisBottom) - create a new bottom-oriented axis generator.
* [d3.axisLeft](https://github.com/d3/d3-axis/blob/master/README.md#axisTeft) - create a new left-oriented axis generator.
* [axis](https://github.com/d3/d3-axis/blob/master/README.md#_axis) - generate an axis for the given selection.
* [axis.scale](https://github.com/d3/d3-axis/blob/master/README.md#axis_scale) - set the scale.
* [axis.ticks](https://github.com/d3/d3-axis/blob/master/README.md#axis_ticks) - customize how ticks are generated and formatted.
* [axis.tickArguments](https://github.com/d3/d3-axis/blob/master/README.md#axis_tickArguments) - customize how ticks are generated and formatted.
* [axis.tickValues](https://github.com/d3/d3-axis/blob/master/README.md#axis_tickValues) - set the tick values explicitly.
* [axis.tickFormat](https://github.com/d3/d3-axis/blob/master/README.md#axis_tickFormat) - set the tick format explicitly.
* [axis.tickSize](https://github.com/d3/d3-axis/blob/master/README.md#axis_tickSize) - set the size of the ticks.
* [axis.tickSizeInner](https://github.com/d3/d3-axis/blob/master/README.md#axis_tickSizeInner) - set the size of inner ticks.
* [axis.tickSizeOuter](https://github.com/d3/d3-axis/blob/master/README.md#axis_tickSizeOuter) - set the size of outer (extent) ticks.
* [axis.tickPadding](https://github.com/d3/d3-axis/blob/master/README.md#axis_tickPadding) - set the padding between ticks and labels.

These are the d3 functionalities used to setup the axis.