



D3 IMPLEMENTATION I

Contents

- Overview and Installation
- D3 Selections
- Data Join
- Connect to SVG
- Connect to SVG for SVG Transformation
- Transition
- Animation
- Drawing Charts
- Graphs

Overview and Installation

What is D3.js?

- D3.js is a **JavaScript** library used to create **interactive visualizations** in the browser.
- The D3.js library allows us to **manipulate** elements of a webpage in the context of a data set. These elements can be **HTML, SVG, or Canvas elements** and can be introduced, removed, or edited according to the contents of the data set. It is a library for manipulating the DOM objects.
- D3.js can be a valuable aid in data exploration, it gives you control over your **data's representation** and lets you add **interactivity**.

Why Do We Need D3.js?

- D3.js is one of the premier framework when compare to other libraries. This is because it works on the web and its data visualizations are par excellence. Another reason it has worked so well is owing to [its flexibility](#).
- Since it works seamlessly with the existing web technologies and can manipulate any part of the document object model, it is as flexible as the **Client Side Web Technology Stack** (HTML, CSS, and SVG). It has a great community support and is easier to learn.

D3.js Benefits

- D3.js is an [open source project](#) and works without any plugin. It requires very less code and comes up with the following benefits –
 - Great data visualization.
 - It is modular. You can download a small piece of D3.js, which you want to use. [No need to load the whole library every time.](#)
 - Easy to build a charting component.
 - DOM manipulation.

Requirements before we starts

- Before we start, we need the following components –
 - D3.js library
 - Editor
 - Web browser
 - Web server
- 3 ways to run D3 under different scenarios

Way 1. Download

- After the download is complete, **unzip** the file and look for **d3.min.js**. This is the **minified version** of the D3.js source code. **Copy** the d3.min.js file and **paste** it into your project's root folder or any other folder, where you want to keep all the library files. **Include** the d3.min.js file in your HTML page as shown below.

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <script src = "/path/to/d3.min.js"></script>
  </head>

  <body>
    <script>
      // write your d3 code here..
    </script>
  </body>
</html>
```

- D3.js is a JavaScript code, so we should write all our D3 code within “**script**” tag. We may need to manipulate the existing DOM elements, so it is advisable to write the D3 code just before the end of the “body” tag.

Way 2: link to the [d3js.org's](https://d3js.org) the newest version

- We can use the D3.js library by linking it directly into our HTML page from the Content Delivery Network (CDN).

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <script src = "https://d3js.org/d3.v6.min.js"></script>
  </head>

  <body>
    <script>
      // write your d3 code here..
    </script>
  </body>
</html>
```

Way 3. Web Server

- Most browsers serve **local HTML files** directly from the local file system. However, there are certain **restrictions** when it comes **to loading external data files such as** loading data from external files like **CSV** and **JSON**. Therefore, it will be easier for us, if we set up the web server right from the beginning.
- You can use any **web server**, which you are comfortable with – e.g. **IIS**, **Apache**, etc.
- In most cases, we can just open your HTML file in a web browser to view it. However, when loading external data sources, it is more reliable to run a local web server and view your page from the server (**`http://localhost:8080`**).

Download Another Edit

- **Sublime Text**:是一套跨平台的文字編輯器，支援基於Python的外掛程式。**Sublime Text**是專有軟體，可透過套件（**Package**）擴充功能。大多數的套件使用自由軟體授權釋出，並由社群建置維護



D3 Selections

Selectors(選擇集)

- D3.js helps to select elements from the HTML page using the following two methods –
 - **select()** – Selects only one DOM element by matching the given CSS selector. If there are more than one elements for the given CSS selector, it selects **the first one** only.
 - **selectAll()** – Selects all DOM elements by matching the given CSS selector. If you are familiar with selecting elements with jQuery, D3.js selectors are almost the same.



Select DOM Elements using D3

- D3 allows us to manipulate DOM elements in the HTML document and for that we first need to **select** a particular **element**, or a group of **elements** and then manipulate those elements **using various D3 methods**.

Method	Description
<code>d3.select(css-selector)</code>	Returns the first matching element in the HTML document based on specified css-selector
<code>d3.selectAll(css-selector)</code>	Returns all the matching elements in the HTML document based on specified css-selector

1. Selection by Tag (以標籤來選擇)

- You can select HTML elements using its TAG. The following syntax is used to select the “div” tag elements, `d3.select("div")`

```
ex06-01.html x
1 <!DOCTYPE html>
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <script src = "https://d3js.org/d3.v6.min.js"></script>
6   </head>
7
8   <body>
9     <div>
10      Hello World!
11    </div>
12
13    <script>
14      alert(d3.select("div").text());
15    </script>
16  </body>
17 </html>
```

這個網頁顯示

Hello World!

確定

2. Selection by Class name

- HTML elements styled using CSS classes can be selected by using the following syntax.

```
d3.select(".<class name>")
```

```
ex06-02.html  ex06-01.html  x
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <script src = "https://d3js.org/d3.v6.min.js"></script>
5    </head>
6    <body>
7      <div class = "myclass">
8        Kun Shan University!
9      </div>
10
11     <script>
12       alert(d3.select(".myclass").text());
13     </script>
14   </body>
15 </html>
```

這個網頁顯示

Kun Shan University!

確定

3. Selection by ID

- Every element in a HTML page should have a unique ID. We can use this unique ID of an element to access it using the select() method as specified below.

```
<body>
  <div id = "hello">
    Hello World!
  </div>

  <script>
    alert(d3.select("#hello").text());
  </script>
</body>
```

4. Adding DOM Elements

- The D3.js selection provides the **append()** and the **text()** methods to append new elements into the existing HTML documents. This section explains about adding DOM elements in detail.
- The **append()** method **appends a new element** as the last child of the element in the current selection. This method can also **modify** the **style of the elements, their attributes, properties, HTML and text content**.

```
<body>
  <div class = "myclass">
    Hello World!
  </div>

  <script>
    d3.select("div.myclass").append("span");
  </script>
</body>
```

Append on the memory

```
<div class = "myclass">
  Hello World!<span></span>
</div>
```



Chain syntax: append().text()

```
ex06-03.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src = "https://d3js.org/d3.v6.min.js"></script>
5   </head>
6
7   <body>
8     <div class = "myclass">
9       Kun Shan University
10    </div>
11
12    <script>
13      d3.select("div.myclass").append("span").text("from D3.js");
14    </script>
15  </body>
16 </html>
```

Kun Shan University from D3.js

Method Chaining in D3

- In the previous sections, we wrote **D3 functions** 'connected' to each other with **dots**. Does that make you curious? This is called "**chain syntax**". If you are familiar with JQuery, you might be familiar with the following.
- The output of the first method is passed as an input to the next method in the chain.

```
d3.select("body").append("p").text("Hello World!");
```

- We also could have written our D3 code without using chaining as below.

```
var bodyElement = d3.select("body");  
  
var paragraph = bodyElement.append("p");  
  
paragraph.text("Hello World!");
```

text()

- The output of the first method is passed as an input to the next method in the chain.
- The text() method receives the paragraph element from the previous method and adds the text provided to it.

```
d3.select("body").append("p").text("Hello World!");
```

```
<body>
  <div class = "myclass">
    Kun Shan University
  </div>

  <script>
    d3.select("div.myclass").append("span").text("from D3.js");
  </script>
</body>
```

The other method instead of Chain Syntax

- By chaining methods together with periods, you can perform several actions in a single line of code. It is fast and easy. The same script can also access **without chain syntax** as shown below.

```
var body = d3.select("div.myclass");  
var span = body.append("span");  
span.text("from D3.js");
```

Modifying Elements

- D3.js provides various methods, **html()**, **attr()** and **style()** to modify the content and style of the selected elements. Let us see how to use modify methods



5.A html() Method

- The `html()` method is used to **reset** the html content of the selected / appended elements.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src = "https://d3js.org/d3.v6.min.js"></script>
5   </head>
6
7   <body>
8     <div class = "myclass">
9       Hello World!
10    </div>
11
12    <script>
13      d3.select(".myclass").html("New Hello World! <span>from D3.js</span>");
14    </script>
15  </body>
16 </html>
```

New Hello World! from D3.js



6.B attr() Method

- The **attr()** method is used to add or update the attribute of the selected elements.

```
ex06-05.html x
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <script src = "https://d3js.org/d3.v6.min.js"></script>
5    </head>
6
7    <body>
8      <div class = "myclass">
9        Spider Man!
10     </div>
11
12     <script>
13       d3.select(".myclass").attr("style", "color: blue");
14     </script>
15   </body>
16 </html>
```

Spider Man!



7.C style() Method

- The style() method is used to set the style property of the selected elements.

```
<body>
  <h2 class = "myclass">KSU </h2>
  <div class = "myclass">IE Dept.</div>
  <h2 id = "myclass"> Wang </h2>

  <script>
    d3.selectAll(".myclass").attr("style", "color: blue");
    d3.selectAll("#myclass").style("color", "red");
  </script>
</body>
```



KSU

IE Dept.

Wang

8. `classed()` Method

- The `classed()` method is exclusively used to set the “class” attribute of an HTML element. Since, a single HTML element can have multiple classes; we need to be careful while assigning a class to an HTML element. This method knows how to handle one or many classes on an element, and it will be performed.

- **Add class** – To add a class, the second parameter of the `classed` method must be set to `true`. It is defined below –

```
d3.select(".myclass").classed("myanotherclass", true);
```

- **Remove class** – To remove a class, the second parameter of the `classed` method must be set to `false`. It is defined below –

```
d3.select(".myclass").classed("myanotherclass", false);
```

- **Check class** – To check for the existence of a class, just leave off the second parameter and pass the class name you are querying. This will return `true`, if it exists, `false`, if it does not.

```
d3.select(".myclass").classed("myanotherclass");
```

This will return `true`, if any element in the selection has the class. Use `d3.select` for single element selection.

- **Toggle class** – To flip a class to the opposite state – remove it if it exists already, add it if it does not yet exist – you can do one of the following.

For a single element, the code might look as shown below –

```
var element = d3.select(".myclass")
element.classed("myanotherclass", !oneBar.classed("myanotherclass"));
```



Example

- Use a **DOM** method to check if a class has been added in the **DOM tree** that is implemented by a **D3** method. The results are shown in the html console.

```
<body>
  <div>
    <a>KSU Depts.</a>
  </div>

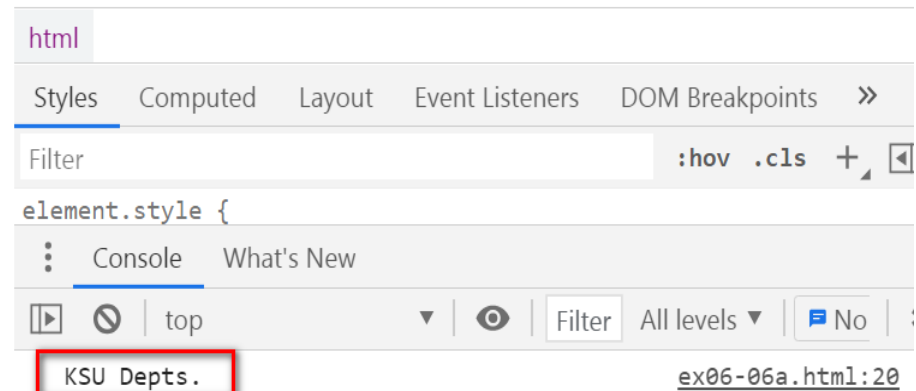
  <script>
    // Sets the class to the a tag
    var a = d3.select("a")
      .classed("className", true);
    // This will select the anchor tag
    var divselect = document.querySelector(".className");
    console.log(divselect.innerHTML);
  </script>
</body>
```



```
▼ <body>
  ▼ <div>
    <a class="className">KSU Depts.</a>
  </div>
  ▼ <script>

    // Sets the class to the a tag
    var a = d3.select("a")
      .classed("className", true);

    // This will select the anchor tag
    var divselect =
    document.querySelector(".className");
    console.log(divselect.innerHTML);
```



Example

ex06-06b.html

```
<body>
  <div>
    <p class="classGiven classGiven2">
      GeeksforGeeks
    </p>
  </div>

  <script>
    // Unsets the class named classGiven
    // to the "p" tag
    var a = d3.select("p")
      .classed("classGiven2", false);

    // This will select the "p" tag
    var divselect = document
      .querySelector(".classGiven");

    console.log(divselect.innerHTML);

    // This will not select the "p" tag
    // As the classGiven 2 is unset
    var divselect2 = document
      .querySelector(".classGiven2.innerHTML");

    console.log(divselect2);
  </script>
</body>
```

▼ <div>

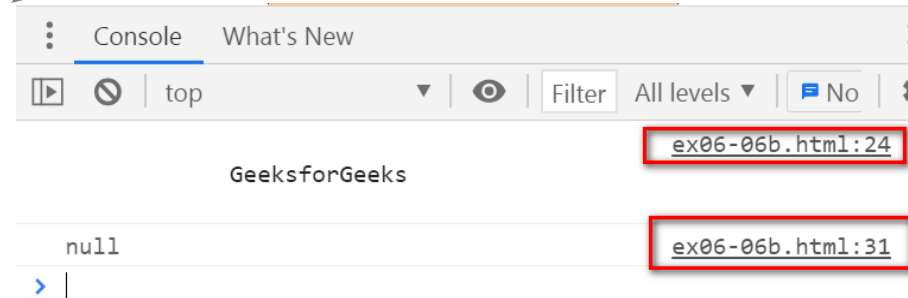
```
<p class="classGiven">
  GeeksforGeeks
</p>
</div>
▼ <script>
  // Unsets the class named classGiven
  // to the "p" tag
  var a = d3.select("p")
    .classed("classGiven2", false);

  // This will select the "p" tag
  var divselect = document
    .querySelector(".classGiven");

  console.log(divselect.innerHTML);

  // This will not select the "p" tag
  // As the classGiven 2 is unset
  var divselect2 = document
    .querySelector(".classGiven2");

  console.log(divselect2);
</script>
```



9. selectAll() Method

- The selectAll() method is used to select multiple elements in the HTML document. The select method selects the first element, but the selectAll method selects all the elements that match the specific selector string.
- In case the selection matches none, then it returns an empty selection. We can chain all the appending modifying methods, **append()**, **html()**, **text()**, **attr()**, **style()**, **classed()**, etc., in the selectAll() method as well

```
ex06-06.html x ex06-06a.html ex06-06b.html x ex06-07.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <script src = "https://d3js.org/d3.v6.min.js"></script>
5 </head>
6 <body>
7   <h2 class = "myclass">KSU </h2>
8   <div class = "myclass">IE Dept.</div>
9   <h2 id = "myclass"> Wang </h2>
10
11   <script>
12     d3.selectAll(".myclass").attr("style", "color: blue");
13     d3.selectAll("#myclass").style("color", "red");
14   </script>
15 </body>
16 </html>
```

KSU
IE Dept.
Wang

Data Join

Data Join

- Data join is another important concept in D3.js. It works along with **selections** and enables us to manipulate the HTML document with respect to **our data set** (*a series of numerical values*).
- By default, D3.js gives data set the highest priority in its methods and each item in **the data set** corresponds to **a HTML element**.

How Data Join Works?

- The primary purpose of the **Data join** is to map the elements of the existing document with the given data set. It creates a **virtual representation** of the document with respect to the given data set and provides methods to work with the **virtual representation**. Let us consider a simple data set as shown below.

[10, 20, 30, 25, 15]

- The data set has five items and so, it can be mapped to five elements of the document. Let us map it to the **li** element of the following document using the selector's **selectAll()** method and data join's **data()** method.

Basic Idea

```
[10, 20, 30, 25, 15]
```



HTML

```
<ul id = "list">  
  <li>  
  <li></li>  
</ul>
```

D3.js code

```
d3.select("#list").selectAll("li").data([10, 20, 30, 25, 15]);
```



1. li - 10
2. li - 20

Advanced Idea

- We can use all the selector's element modifying methods like **attr()**, **style()**, **text()**, etc., for the first two **li** as shown below. The function in the **text()** method is used to get the **li** elements mapped data. Here, **d** represent 10 for first **li** element and 20 for second **li** element.

```
d3.select("#list").selectAll("li")
  .data([10, 20, 30, 25, 15])
  .text(function(d) { return d; });
```

- The next three elements can be mapped to any elements and it can be done using the data join's **enter()** and selector's **append()** method. The **enter()** method gives access to **the remaining data** (which is not mapped to the existing elements) **and the **append()** method is used to create a new element from the corresponding data.** Let us create **li** for the remaining data items as well. The data map is as follows and the code to create new a **li** element is as follows --

```
3. li - 30
4. li - 25
5. li - 15
```

```
d3.select("#list").selectAll("li")
  .data([10, 20, 30, 25, 15])
  .text(function(d) { return "This is pre-existing element and the value is " + d; })
  .enter()
  .append("li")
  .text(function(d)
    { return "This is dynamically created element and the value is " + d; });
```

Advanced Idea

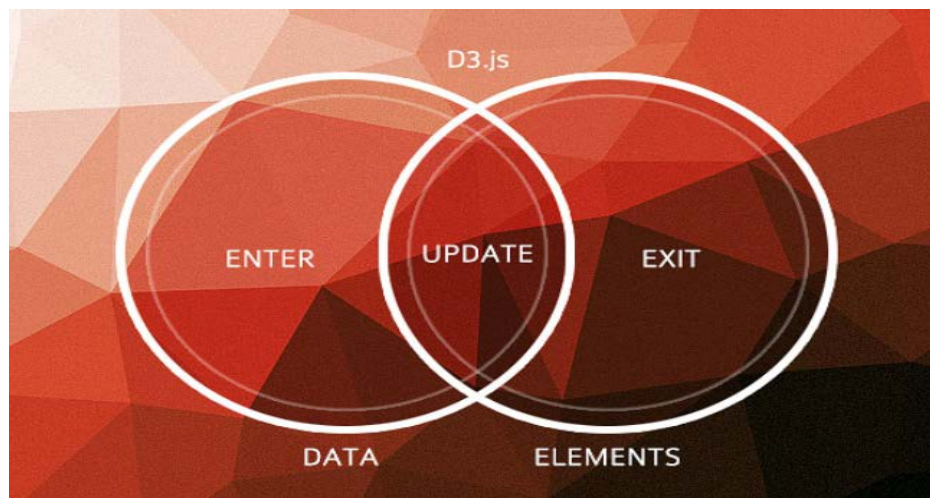
- Data join provides another method called as the **exit() method** to process the data items removed dynamically from the data set as shown below.

```
d3.selectAll("li")  
  .data([10, 20, 30, 15])  
  .exit()  
  .remove()
```

- **.remove()** doesn't take any arguments, it is just a method you can use on any d3 selection.

enter() 、 update() and exit()

- 這三個元素可以處理當畫面元素 (Elements) 和資料數量 (data) 不相等的情形
- 以下面這張圖來說，我們利用 **enter** 來放入資料，如果資料有多少筆，就會在背景「預先」產生多少個元素，當 **enter** 指令發生的當下，就會把這些預先產生的元素放到畫面裏，而資料和元素數量相等的部分，我們就稱之為「**update**」，如果資料比元素多，「**enter**」就會自動生成對應數量的元素來因應，如果資料比元素少，就可以用 **exit** 指令來列出多的元素，再利用 **remove** 來移除，這就是 **enter** 到 **update** 到 **exit** 的標準流程。(簡單來說，**enter** 就是多出來的，**update** 就是相等，**exit** 就是少掉的)。
- 我們也可以用 **exit.remove()** 將多出來的資料移除，就可以呈現資料相對應的數量元素在畫面上



```
<ul id = "list">
  <li></li>
  <li></li>
</ul>
<input type = "button" name = "remove" value = "Remove value"
  onclick = "javascript:remove()" />

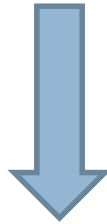
<script>
  d3.select("#list").selectAll("li")
    .data([10, 20, 30, 25, 15])
    .text(function(d)
      { return "This is pre-existing element and the value is " + d; })
    .enter()
    .append("li")
    .text(function(d)
      { return "This is dynamically created element and the value is " + d; });

  function remove() {
    d3.selectAll("li")
      .data([10, 20, 30, 25])
      .remove()
  }
</script>
```

ex06-07a.html

- This is pre-existing element and the value is 10
- This is pre-existing element and the value is 20
- This is dynamically created element and the value is 30
- This is dynamically created element and the value is 25
- This is dynamically created element and the value is 15

Remove value



- This is dynamically created element and the value is 15

Remove value

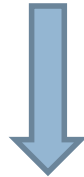
ps Delete the first 4 elements from its front
by using remove. Nothing related to the elements in data()


```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>
5     <style>
6       body { font-family: Arial; }
7     </style>
8   </head>
9   • ex06-06.html
10  <body>
11    <ul id = "list">
12      <li></li>
13      | <li></li>
14    </ul>
15
16    <input type = "button" name = "remove" value = "Remove fourth value"
17      onclick = "javascript:remove()" />
18
19    <script>
20      d3.select("#list").selectAll("li")
21        .data([10, 20, 30, 25, 15])
22        .text(function(d)
23          { return "This is pre-existing element and the value is " + d; })
24        .enter()
25        .append("li")
26        .text(function(d)
27          { return "This is dynamically created element and the value is " + d; });
28
29      function remove() {
30        d3.selectAll("li")
31          .data([10, 20, 30, 15])
32          .exit()
33          .remove()
34      }
35    </script>
36  </body>
37 </html>
```

ex06-07.html

- This is pre-existing element and the value is 10
- This is pre-existing element and the value is 20
- This is dynamically created element and the value is 30
- This is dynamically created element and the value is 25
- This is dynamically created element and the value is 15

Remove fourth value



- This is pre-existing element and the value is 10
- This is pre-existing element and the value is 20
- This is dynamically created element and the value is 30
- This is dynamically created element and the value is 25


Remove fourth value

ps 因data()有4個元素，藉由exit().remove()每次從後面移除1個
非對應相對的資料 Delete the last 1 element from its back
by using exit().remove(). Nothing related to the elements in data()

Connect to SVG

Review: Drawing a line and shapes

```
ex06-08.html x
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <style>
5        body { font-family: Arial; }
6      </style>
7    </head>
8    <body>
9      <div id = "svgcontainer">
10        <svg width = "300" height = "300">
11          <line x1 = "100" y1 = "100"
12            x2 = "200" y2 = "200" style = "stroke:rgb(255,0,0);
13            stroke-width:2"/>
14        </svg>
15      </div>
16      <p></p>
17      <p></p>
18    </body>
19  </html>
20
```



- Let us **create a minimal SVG image** and include it in the HTML document as above.

- Step 1** – Create a SVG image and set width as 300 pixel and height as 300 pixel.

```
<svg width = "300" height = "300">  
  
</svg>
```

- Step 2** – Create a line starting at (100, 100) and ending at (200, 100) and set red color for the line.

```
<line x1 = "100" y1 = "100" x2 = "200" y2 = "200"  
      style = "stroke:rgb(255,0,0);stroke-width:2"/>
```

- Here, the **line** tag draws a line and its attributes **x1**, **y1** refers to the starting point and **x2**, **y2** refers to the ending point. The style attribute sets color and thickness of the line using the **stroke** and the **stroke-width** styles.

- **x1** – This is the x-coordinate of the first point.
- **y1** – This is the y-coordinate of the first point.
- **x2** – This is the x-coordinate of the second point.
- **y2** – This is the y-coordinate of the second point.
- **stroke** – Color of the line.
- **stroke-width** – Thickness of the line.

- Step 3** – Create a HTML document, “ex06-08.html ” and integrate the above SVG as shown above.

SVG Using D3.js to do the same thing

- To create SVG using D3.js, let us follow the steps given below.

- **Step 1** – Create a container to hold the SVG image as given below.

```
<div id = "svgcontainer"></div>
```

- **Step 2.** Select the SVG container using the select() method and inject the SVG element using the append() method. Add the attributes and styles using the attr() and the style() methods.

```
var width = 300;  
var height = 300;  
var svg = d3.select("#svgcontainer")  
    .append("svg").attr("width", width).attr("height", height);
```

- **Step 3** – Similarly, add the **line** element inside the **svg** element as shown below.

```
svg.append("line")  
    .attr("x1", 100)  
    .attr("y1", 100)  
    .attr("x2", 200)  
    .attr("y2", 200)  
    .style("stroke", "rgb(255,0,0)")  
    .style("stroke-width", 2);
```

```
ex06-09.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src = "https://d3js.org/d3.v6.min.js"></script>
5     <style>
6       body { font-family: Arial; }
7     </style>
8   </head>
9   <body>
10    <div id = "svgcontainer">
11    </div>
12    <script language = "javascript">
13      var width = 300;
14      var height = 300;
15      var svg = d3.select("#svgcontainer")
16        .append("svg")
17        .attr("width", width)
18        .attr("height", height);
19      svg.append("line")
20        .attr("x1", 100)
21        .attr("y1", 100)
22        .attr("x2", 200)
23        .attr("y2", 200)
24        .style("stroke", "rgb(255,0,0)")
25        .style("stroke-width", 2);
26    </script>
27  </body>
28 </html>
```



Rectangle Using D3.js

- A rectangle is represented by the **<rect>** tag as shown below.

```
<rect x = "20" y = "20" width = "300" height = "300"></rect>
```

The attributes of a rectangle are as follows –

- **x** – This is the x-coordinate of the top-left corner of the rectangle.
- **y** – This is the y-coordinate of the top-left corner of the rectangle.
- **width** – This denotes the width of the rectangle.
- **height** – This denotes the height of the rectangle.

- A simple rectangle in SVG is defined as explained below.

```
<svg width = "300" height = "300">  
  <rect x = "20" y = "20" width = "300" height = "300" fill = "green"></rect>  
</svg>
```



```
ex06-10.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src = "https://d3js.org/d3.v6.min.js"></script>
5   </head>
6
7   <body>
8     <div id = "svgcontainer"></div>
9     <script>
10       var width = 300;
11       var height = 300;
12       //Create SVG element
13       var svg = d3.select("#svgcontainer")
14         .append("svg")
15         .attr("width", width)
16         .attr("height", height);
17       //Create and append rectangle element
18       svg.append("rect")
19         .attr("x", 20)
20         .attr("y", 20)
21         .attr("width", 200)
22         .attr("height", 100)
23         .attr("fill", "green");
24     </script>
25   </body>
26 </html>
```



Circle Using D3.js

- A circle is represented by the **<circle>** tag as explained below.

```
<circle cx = "200" cy = "50" r = "20"/>
```

The attributes of circle are as follows –

- **cx** – This is the x-coordinate of the center of the circle.
 - **cy** – This is the y-coordinate of the center of the circle.
 - **r** – This denotes the radius of the circle.
- A simple circle in SVG is described below.

```
<svg width = "300" height = "300">  
  <circle cx = "200" cy = "50" r = "20" fill = "green"/>  
</svg>
```

```
ex06-11.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src = "https://d3js.org/d3.v6.min.js"></script>
5   </head>
6
7   <body>
8     <div id = "svgcontainer"></div>
9     <script>
10       var width = 300;
11       var height = 300;
12       //Create SVG element
13       var svg = d3.select("#svgcontainer")
14         .append("svg")
15         .attr("width", width)
16         .attr("height", height);
17       //Append circle
18       svg.append("circle")
19         .attr("cx", 200)
20         .attr("cy", 50)
21         .attr("r", 20)
22         .attr("fill", "green");
23     </script>
24   </body>
25 </html>
```

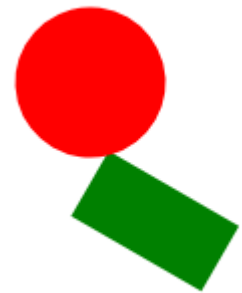


Connect to SVG for SVG Transformation

Review: SVG Transformation

- Transformation can be applied to the SVG group element as well. This enables to transform complex graphics defined in the SVG as described below.

```
ex06-12.html x
1  <!DOCTYPE html>
2  <body>
3      <svg width = "300" height = "300">
4          <g transform = "translate(60,60) rotate(30)">
5              <rect x = "20"
6                  y = "20"
7                  width = "60"
8                  height = "30"
9                  fill = "green">
10             </rect>
11             <circle cx = "0"
12                 cy = "0"
13                 r = "30"
14                 fill = "red"/>
15         </g>
16     </svg>
17 </body>
18 </html>
```



SVG Using D3.js to do the same thing

- To create an SVG image, try to scale, and rotate it using transformation, let us follow the steps given below.
 - **Step 1** – Create an SVG image and set width as 300 pixels and height as 300 pixels.

```
<svg width = "300" height = "300">  
  
</svg>
```

- **Step 2** – Create an SVG group.

```
<svg width = "300" height = "300">  
  <g>  
  </g>  
</svg>
```

- **Step 3** – Create a rectangle of length 60 and height 30 and fill it with green color.

```
<svg width = "300" height = "300">  
  <g>  
    <rect x = "20"  
      y = "20"  
      width = "60"  
      height = "30"  
      fill = "green">  
    </rect>  
  </g>  
</svg>
```

SVG Using D3.js to do the same thing

- To create an SVG image, try to scale, and rotate it using transformation, let us follow the steps given below.
 - **Step 4** – Create a circle of radius 30 and fill it with red color.

```
<svg width = "300" height = "300">
  <g>
    <rect x = "20"
      y = "20"
      width = "60"
      height = "30"
      fill = "green">
    </rect>
    <circle cx = "0"
      cy = "0"
      r = "30"
      fill = "red"/>
  </g>
</svg>
```

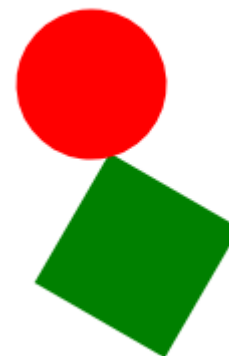
SVG Using D3.js to do the same thing

- To create an SVG image, try to scale, and rotate it using transformation, let us follow the steps given below.
 - **Step 5** – Add a transform attribute and add translate and rotate as shown below.

```
<svg width = "300" height = "300">
  <g transform = "translate(60,60) rotate(30)">
    <rect x = "20"
      y = "20"
      width = "60"
      height = "60"
      fill = "green">
    </rect>
    <circle cx = "0"
      cy = "0"
      r = "30"
      fill = "red"/>
  </g>
</svg>
```



```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <script src = "https://d3js.org/d3.v6.min.js"></script>
5      <style>
6        body { font-family: Arial; }
7      </style>
8    </head>
9    <body>
10     <div id = "svgcontainer">
11       <svg width = "300" height = "300">
12         <g transform = "translate(60,60) rotate(30)">
13           <rect x = "20"
14             y = "20"
15             width = "60"
16             height = "60"
17             fill = "green">
18         </rect>
19         <circle cx = "0"
20           cy = "0"
21           r = "30"
22           fill = "red"/>
23       </g>
24     </svg>
25   </div>
26 </body>
27 </html>
```



Transform Library

- D3.js provides a separate library to manage transform without manually creating the transform attributes. It provides methods to handle all type of transformation. Some of the methods are **transform()**, **translate()**, **scale()**, **rotate()**, etc. You can include **d3-transform** in your webpage using the following script.
 - `<script src = "http://d3js.org/d3.v6.min.js">`
`</script> <script src = "d3-transform.js"></script>`
- In the above example, the transform code can be written as shown below

```
var my_transform = d3Transform()  
    .translate([60, 60])  
    .rotate(30);  
  
var group = svg  
    .append("g")  
    .attr("transform", my_transform);
```

Transition

Transition

- Transition is the process of changing from one state to another of an item. D3.js provides a **transition()** method to perform transition in the HTML page

transition()

- The transition() method - It is available for all selectors and it starts the transition process. This method supports most of the selection methods such as – attr(), style(), etc. But, It does not support the append() and the data() methods, which need to be called before the transition() method. Also, it provides methods specific to transition like duration(), ease(), etc. A simple transition can be defined as follows –

```
d3.select("body")
  .transition()
  .style("background-color", "lightblue");
```

- A transition can be directly created using the d3.transition() method and then used along with selectors as follows.

```
var t = d3.transition()
  .duration(2000);
d3.select("body")
  .transition(t)
  .style("background-color", "lightblue");
```

```
ex06-14.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src = "https://d3js.org/d3.v6.min.js"></script>
5   </head>
6
7   <body>
8     <h3>Simple transitions</h3>
9     <script>
10       d3.select("body").transition().style("background-color", "lightblue");
11     </script>
12   </body>
13 </html>
```

Simple transitions

Anamation

Anamation

- D3.js supports animation through transition. We can do animation with proper use of transition. Transitions are a limited form of **Key Frame Animation** with only two key frames – start and end.
- The starting key frame is typically the current state of the DOM, and the ending key frame is a set of attributes, styles and other properties you specify. Transitions are well suited for transitioning to a new view without a complicated code that depends on the starting view.

duration()

- The duration() method allows property changes to occur smoothly over a specified duration rather than instantaneously. Let us make the transition which takes 5 seconds using the following code.

```
<!DOCTYPE html>
<html>
  <head>
    <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>
  </head>

  <body>
    <h3>Simple transitions</h3>
    <script>
      d3.selectAll("h3").transition().style("color","green").duration(5000);
    </script>
  </body>
</html>
```

- Here, the transitions occurred smoothly and evenly. We can also assign RGB color code value directly using the following method.

```
d3.selectAll("h3").transition().style("color","rgb(0,150,120)").duration(5000);
```

```
ex06-15.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src = "https://d3js.org/d3.v6.min.js"></script>
5   </head>
6
7   <body>
8     <h3> Simple transitions </h3>
9     <script>
10       d3.selectAll("h3").transition()
11         .style("font-size", "28px").delay(2000).duration(2000);
12     </script>
13   </body>
14 </html>
```

Simple transitions

Simple transitions

Simple transitions

Drawing Charts

Graphs