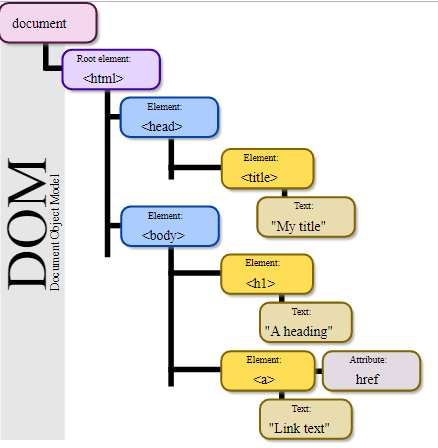
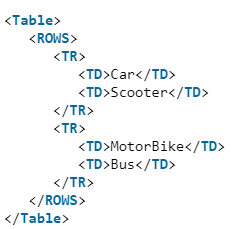
**D3.JS-DATA DRIVEN DOCUMENTS**

**D3.js –Data Driven documents**

D3 means Data-Driven Documents and is a widely popular platform for creating web-based data visualizations on the web. Developed by Mike Bostock, is a New York Times graphics editor. It is a powerful tool for creating data visualizations on the web browser and extraordinary tool for creating custom visualizations based on HTML, CSS, SVG, and JS. D3.js is an open source JavaScript library for manipulating documents based on data.D3.js is a dynamic, interactive, online data visualizations framework used in a large number of websites.





DOM: Document object model is a programming interface for HTML and XML (Extensible markup language) documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

<!DOCTYPE html> Document type declaration for html5.

<html lang="en"> language

<head>

<meta charset="UTF-8">-----**UTF** stands for Unicode Transformation Format.UTF-8 is the preferred

<title> StratApps </title> encoding for email and web pages.

</head>

<body>

<ol type= “Fruits”>

<li>Mango</li>

<li>Graps</li>

<li>Oranges</li>

</ol>

</body>

</html>

**Why do we need D3.js :**

**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 per excellence. 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.

**Features of D3.js :**

**D3.js Features:**

* **Uses Web Standards:** D3 is an extremely powerful visualization tool to create interactive data visualizations. It exploits the modern web standards: SVG, HTML and CSS to create data visualization.
* **Data Driven:** D3 is data driven. It can use static data or fetch it from the remote server in different formats such as Arrays, Objects, CSV, JSON, XML etc. to create different types of charts.
* **DOM Manipulation:** D3 allows you to manipulate the Document Object Model (DOM) based on your data.
* **Data Driven Elements:** It empowers your data to dynamically generate elements and apply styles to the elements, be it a table, a graph or any other HTML element and/or group of elements.
* **Dynamic Properties:**D3 gives the flexibility to provide dynamic properties to most of its functions. Properties can be specified as functions of data. That means your data can drive your styles and attributes.
* **Types of visualization:**With D3, there are no standard visualization formats. But it enables you to create anything from an HTML table to a Pie chart, from graphs and bar charts to geospatial maps.
* **Custom Visualizations:**Since D3 works with web standards, it gives you complete control over your visualization features.
* **Transitions:**D3 provides the transition() function. This is quite powerful because internally, D3 works out the logic to interpolate between your values and find the intermittent states.
* **Interaction and animation:**D3 provides great support for animation with functions like duration(), delay() and ease(). Animations from one state to another are fast and responsive to user interactions.
* **Extremely flexible.**
* Easy to use and fast.
* Support large datasets.
* Declarative programming.
* Code reusability.
* Associates data to an element or group of elements in the html page.

**Advantages of D3**

**Advantages of D3.js :**

* D3.js is a Javascript library. So, it can be used with any JS framework of your choice like Angular.js, React.js or Ember.js.
* D3 focuses on data, so it is the most appropriate and specialized tool for data visualizations.
* D3 is open-source. So you can work with the source code and add your own features.
* It works with web standards so you don't need any other technology or plugin other than a browser to make use of D3.
* D3 works with web standards like HTML, CSS and SVG, there is no new learning or debugging tool required to work on D3.
* D3 does not provide any specific feature, so it gives you complete control over your visualization to customize it the way you want. This gives it an edge over other popular tools like Tableau or QlikView.
* Since D3 is lightweight, and works directly with web standards, it is extremely fast and works well with large datasets.
* 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.

D3.js Few visualization Example:

🡪Bar Chart

🡪Bubble Chart

🡪Circle Packing

🡪Stream Graph…etc.

# Web Standards

**Web Standards :**

Before we can start using D3 to create visualizations, we need clear with these standards.

* HTML
* DOM
* CSS
* SVG
* Java script

**HTML:**

**HTML**

HTML = Hyper Text Markup Language

HTML is used to structure the content of the web page. The current version is HTML 5. It is stored in a text file with the extension ".html".

Ex:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title> StratApps</title>

</head>

<body>

<ol type= “Fruits”>

<li>Mango</li>

<li>Graps</li>

<li>Oranges</li>

</ol>

</body>

</html>

**DOM**

**DOM:**

DOM = Document Object Model

When you write html code for your page, it gets converted to a hierarchical structure on the browser. Every tag in html gets converted to an element in the DOM with a parent-child hierarchy. It makes your html more logically structured. Once the DOM is formed, it makes it easier to manipulate (add/modify/remove) the elements on the page.

**CSS**

**CSS:**

CSS = Cascading Style Sheets

HTML gives a structure to the web page, while CSS styles your web page making it more pleasant to look at. It is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects like SVG or XHTML). CSS describes how elements should be rendered on a web page.

SVG = Scalable Vector Graphics

**SVG**

SVG is a way to render images on the web page. SVG is not a direct image but is just a way to create images using text. As its name suggests, it is scalable vector. It scales itself according to the size of the browser, so resizing your browser will not distort the image. All browsers support SVG except IE 8 and below.

Since data visualizations are visual representations, it is convenient to use SVG to render visualizations using D3.

Think of SVG as a canvas on which you can paint different shapes.

So to start off, create an SVG tag: <svg width="500" height="500"></<svg>

The default measurement for SVG is pixels, so you don't need to specify if your unit is pixel.

Now if you would like to draw a rectangle inside this SVG, draw it using <rect> :

**Example: Square in SVG**

<svg width="500" height="500">

<rect x="0" y="0" width="300" height="200"></rect>

</svg>

You can draw the square using <rect> by applying same width and height attribute. Some of the other shapes that can be drawn in SVG include line, circle, ellipse, text and path.

Just like styling html elements, styling SVG elements is simple. Let's color the above rectangle in yellow. All you need to add is an attribute "fill" and specify the color.

Example: **Colored Rectangle**

<svg width="500" height="500">

<rect x="0" y="0" width="300" height="200" fill="yellow"></rect>

</svg>

<svg width="300" height="180">

<circle cx="30" cy="50" r="25" />

<circle cx="90" cy="50" r="25" class="red" />

<circle cx="150" cy="50" r="25" class="fancy" />

<rect x="10" y="80" width="40" height="40"

fill="steelBlue" />

<rect x="70" y="80" width="40" height="40"

style="fill: steelBlue" />

<rect x="130" y="80" width="40" height="40"

class="fancy" />

</svg>

**.red** {

fill: red; */\* not background-color! \*/*

}

**.fancy** {

fill: none;

stroke: black; */\* similar to border-color \*/*

stroke-width: 3pt; */\* similar to border-width \*/*

stroke-dasharray: 3,5,10;

}

**Colored Circle**:

**--------🡪 <svg width="100" height="100">**

**<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />**

**</svg>**

**Java Script:**

**Java Script**

JavaScript is a loosely-typed client side scripting language that executes in the user's browser. JavaScript interact with html elements (DOM elements) in order to make the web user interface interactive. JavaScript is an open source & most popular client side scripting language supported by all browsers. JavaScript is used mainly for enhancing the interaction of a user with the webpage.

**Creating One Button:**

<button type="button"

onclick="document.getElementById('demo').innerHTML = Date()">

Click me to display Date and Time. </button>

**D3.js Development Environment:**

: D3.js Library

: Editor

: Web browser

: Web Server

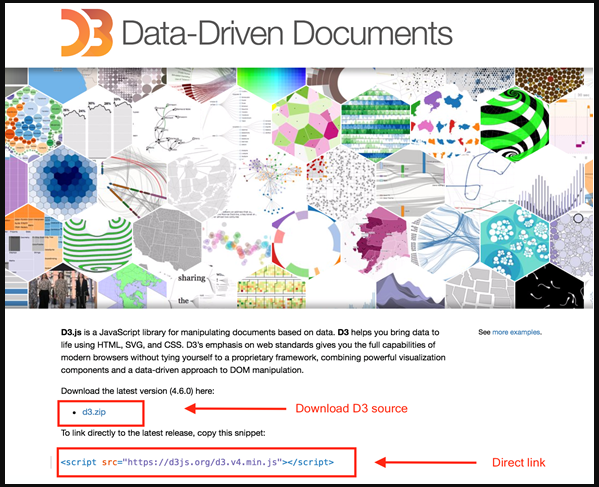
**D3.js Library:**

**Development Environment:**

We need to include the D3.js library into your HTML webpage in order to use D3.js to create data visualization. We can do it in the Following two ways: Latest version of d3.js is 5.9.7

* 1. Include the D3.js library from your projects folder.
  2. Include D3,js library from CDN(Content Delivery Network).

**Download D3.js Library Development Environment:**

****

**D3.js Library:**

**Development Environment:**

D3.js is an open-source library and the source code of the library is freely available on the web at <http://d3.js.org> (web site) latest version 5.0

After the download is complete unzip the file and look for d3.min.js. This is minified version of the D3.js source code. Copy the D3.min.js files and paste it into your projects 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.

**<!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.

### Include D3 Library from CDN

We can use the D3.js library by linking it directly into our HTML page from the Content Delivery Network (CDN). CDN is a network of servers where files are hosted and are delivered to a user based on their geographic location. If we use the CDN, we do not need to download the source code.

Include the D3.js library using the CDN URL <https://d3js.org/d3.v4.min.js>

**Example** − Let us consider the following example.

**<!DOCTYPE html>**

**<html lang = "en">**

**<head>**

**<script src = "https://d3js.org/d3.v4.min.js"></script>**

**</head>**

**<body>**

**<script>**

**// write your d3 code here..**

**</script>**

**</body>**

**</html>**

**D3.js Editor :**

**Development Environment:**

There are some great IDEs (Integrated Development Environment) with support for JavaScript like −

* Visual Studio Code
* WebStorm
* Eclipse
* Sublime Text

These IDEs provide intelligent code completion as well as support some of the modern JavaScript frameworks. If you do not have fancy IDE, you can always use a basic editor like Notepad, VI, etc.

## Web Browser

**D3.js Web Browser :**

**Development Environment:**

D3.js works on all the browsers except IE8 and lower.

**D3.js Web Server :**

**Development Environment:**

### 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.

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)**.

**D3.Selections:**

Selections are one of the core concepts in D3.js. It is based on CSS selectors. It allows us to select one or more elements in a webpage. In addition, it allows us to modify, append, or remove elements in a relation to the pre-defined dataset.

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 is more than one element 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.

## The select() method

The select() method selects the HTML element based on CSS Selectors. In CSS Selectors, you can define and access HTML-elements in the following three ways −

* Tag of a HTML element (e.g. div, h1, p, span, etc.,)
* Class name of a HTML element
* ID of a HTML element

Let us see it in action with examples.

### Selection by Tag:

You can select HTML elements using its TAG. The following syntax is used to select the “div” tag elements,

d3.select(“div”)

**Example** − Create a page “select\_by\_tag.html” and add the following changes,

**<!DOCTYPE html>**

**<html>**

**<head>**

**<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>**

**</head>**

**<body>**

**<div>**

**Hello World!**

**</div>**

**<script>**

**alert(d3.select("div").text());**

**</script>**

**</body>**

**</html>**

By requesting the webpage through the browser, you will see the following output on the screen −

### Selection by Class name

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

d3.select(“.<class name>”)

Create a webpage “select\_by\_class.html” and add the following changes −

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div class = "myclass">

Hello World!

</div>

<script>

alert(d3.select(".myclass").text());

</script>

</body>

</html>

By requesting the webpage through the browser, you will see the following output on the screen −

### 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.

d3.select(“#<id of an element>”)

Create a webpage “select\_by\_id.html” and add the following changes.

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div id = "hello">

Hello World!

</div>

<script>

alert(d3.select("#hello").text());

</script>

</body>

</html>

By requesting the webpage through the browser, you will see the following output on the screen.

## 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

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.

Create a webpage “select\_and\_append.html” and add the following changes −

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div class = "myclass">

Hello World!

</div>

<script>

d3.select("div.myclass").append("span");

</script>

</body>

</html>

Requesting the webpage through browser, you could see the following output on the screen,

Here, the append() method adds a new tag span inside the div tag as shown below −

<div class = "myclass">

Hello World!<span></span>

</div>

### The text() Method

The text() method is used to set the content of the selected / appended elements. Let us change the above example and add the text() method as shown below.

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div class = "myclass">

Hello World!

</div>

<script>

d3.select("div.myclass").append("span").text("from D3.js");

</script>

</body>

</html>

Now refresh the webpage and you will see the following response.

Here, the above script performs a chaining operation. D3.js smartly employs a technique called the **chain syntax**, which you may recognize from **jQuery**. 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 in this chapter.

### The html() Method

The html() method is used to set the html content of the selected / appended elements.

Create a webpage “select\_and\_add\_html.html” and add the following code.

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div class = "myclass">

Hello World!

</div>

<script>

d3.select(".myclass").html("Hello World! <span>from D3.js</span>");

</script>

</body>

</html>

By requesting the webpage through the browser, you will see the following output on the screen.

### The attr() Method

The attr() method is used to add or update the attribute of the selected elements. Create a webpage “select\_and\_modify.html” and add the following code.

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div class = "myclass">

Hello World!

</div>

<script>

d3.select(".myclass").attr("style", "color: red");

</script>

</body>

</html>

By requesting the webpage through the browser, you will see the following output on the screen.

### The style() Method

The style() method is used to set the style property of the selected elements. Create a webpage “select\_and\_style.html” and add the following code.

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div class = "myclass">

Hello World!

</div>

<script>

d3.select(".myclass").style("color", "red");

</script>

</body>

</html>

By requesting the webpage through the browser, you will see the following output on the screen.

### The 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 performant.

* **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"));

## The 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. In this case, the methods will affect all the matching elements. Let us understand by creating a new webpage “select\_multiple.html” and add the following script −

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<h2 class = "myclass">Message</h2>

<div class = "myclass">

Hello World!

</div>

<script>

d3.selectAll(".myclass").attr("style", "color: red");

</script>

</body>

</html>

By requesting the webpage through the browser, you will see the following output on the screen.

Here, the attr() method applies to both **div** and **h2 tag** and the color of the text in both tags changes to Red.