Before D3
HTML, CSS, Javascript
Data Visualization

OUTLINE

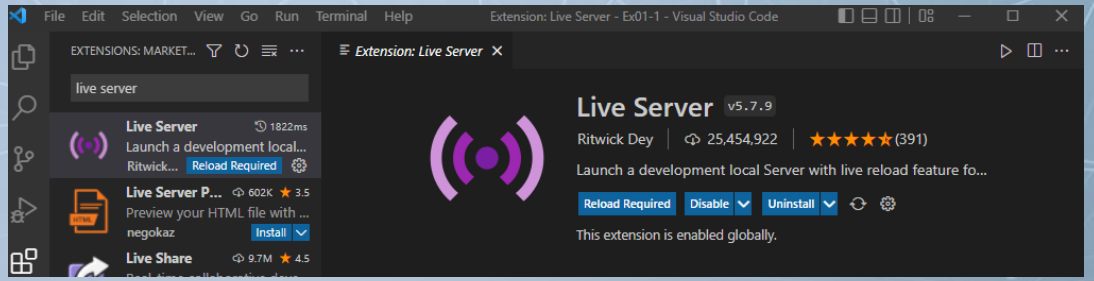
- What is D3
- HTML/CSS/SVG(path & g)
- JavaScript
  - Console , ;
  - Variables(Declaring /types(dic.))
  - function

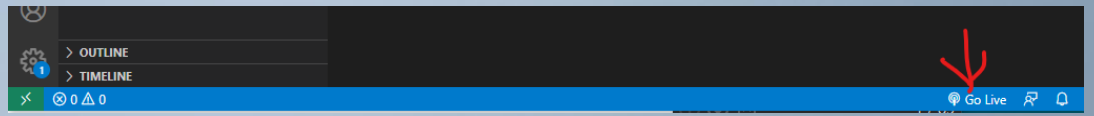
What is D3

- Data-Driven Documents
- Is a JavcaScript lib.
- Web-based.
- Need HTML SVG CSS to work together
- Latest version is v7. (V3~v4 big difference)
- Official website (https://d3js.org/)
- Find more example here
  (https://observablehq.com/@d3/gallery)

Environment and Tool to Write D3

- Web browser – Chrome
- Editor –“Visual Studio Code” (VSCode)
- Http-server
  - I recommend a VSCode plug-in “Live Server” for small scale test





Before D3

- We need to know some web-based related languages and knowledge
  - HTML
  - CSS
  - SVG
  - JavaScript

HTML – Hyper Text Markup Language

- HTML is the standard markup language for creating Web pages

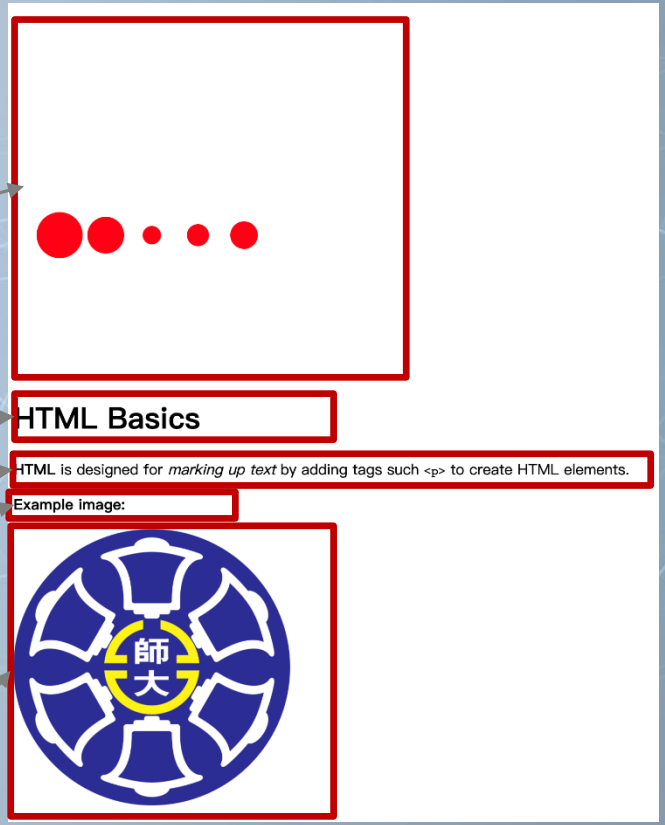
- HTML elements
  - HTML elements are the building blocks of HTML pages
  - Represented by tags

- Tag
  - HTML tags label pieces of content such as
    - <head>, <p>, <table>…..
  - Browsers do not display the HTML tags, but use them to render content of the page

Example (Ex01-2)

- index.html

```
<!doctype html>
<html>
<head>
    <title>Example</title>
</head>
<body>
    <div class="container">
        <div class="row">
            <div id="chart-area"></div>
        </div>
    </div>

    <h1>HTML Basics</h1>
    <p>
        <Strong>HTML</Strong>
        is designed for
        <em>marking up text</em>
        by adding tags such
        <code>&lt;p&gt;</code>
        to create HTML elements.
    </p>

    <p>
        <strong>Example image:</strong>
    </p>
    <img width="300" height="300" src="https://upload.wikimedia.org/wikipedia/en/thumb/c/c3/National_Taiwan_Norma

    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="main.js"></script>
</body>
</html>
```



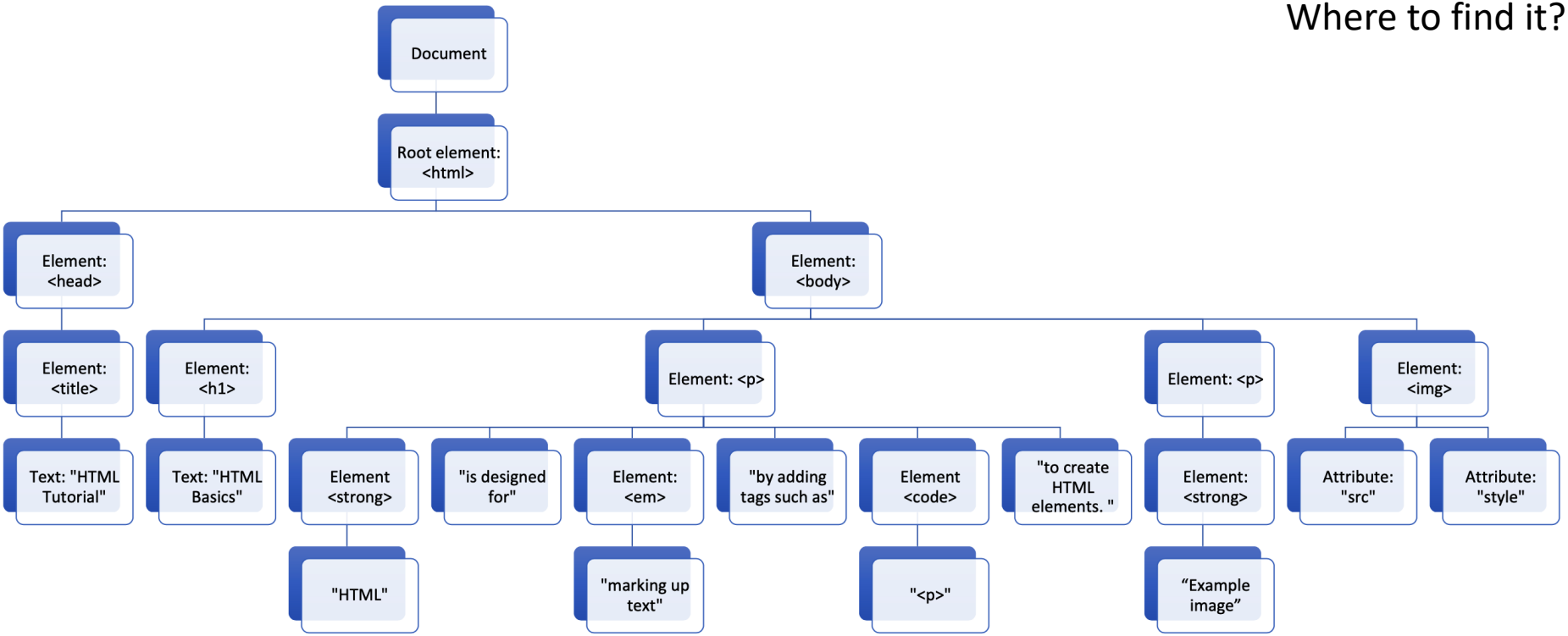
HTML - DOM

- When a web page is loaded, the browser create a Document Object Model (DOM) for the page

- The HTML DOM model is constructed as a **tree** of Objects

HTML - DOM

Where to find it?

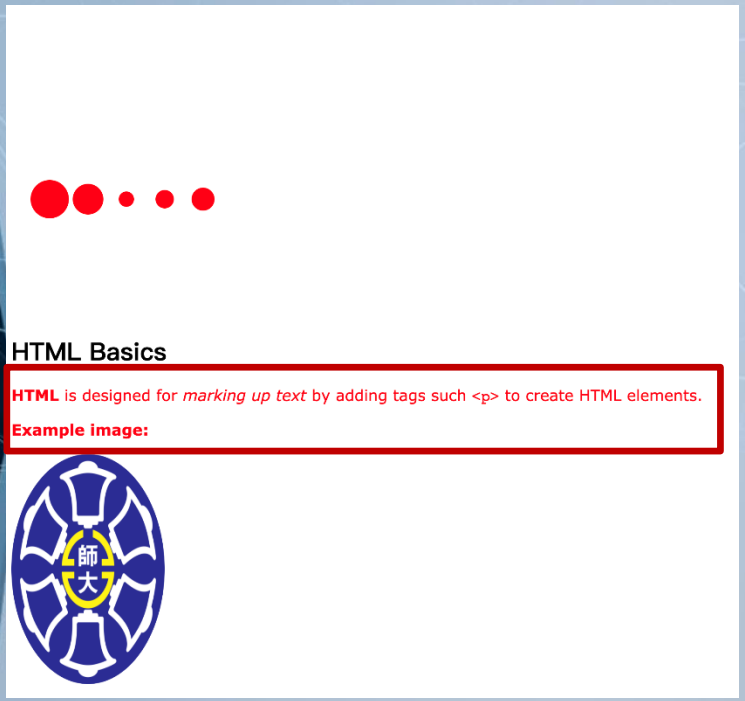


CSS – Cascading Style Sheets

- CSS describe how HTML elements are to be displayed on screen
- CSS saves a lot of work
  - It can control the appearance of multiple elements and web pages all at once

Example (Ex01-3)

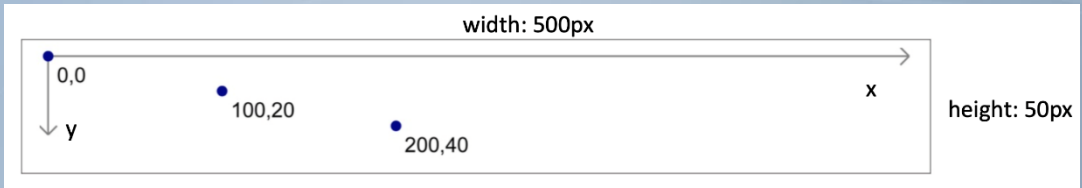
- index.html




```
<html>
<head>
    <title>Example</title>
</head>
<style>
    p{
        color: red;
        font-family: verdana;
        font-size: 20px;
    }
    img{
        width: 200px;
        border-radius:50%;
    }
</style>
<body>
    <div class="container">
        <div class="row">
            <div id="chart-area"></div>
        </div>
    </div>

    <h1>HTML Basics</h1>
    <p>
        <Strong>HTML</Strong>
        is designed for
        <em>marking up text</em>
        by adding tags such
        <code>&lt;p&gt</code>
        to create HTML elements.
    </p>

    <p>
        <strong>Example image:</strong>
    </p>
    <img width="300" height="300" src="https://upload.wikimedia.org/wikipedia/en/thumb/c/c3/Nati

    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="main.js"></script>
</body>
</html>
```

SVG – Scalable Vector Graphics

- SVG defines vector-based graphics for the web
- Svg HTML tag
  - <svg width="500" height="50"></svg>
  - Create a SVG canvas with 500x50pixels
- SVG coordinates system **(important!!!)**

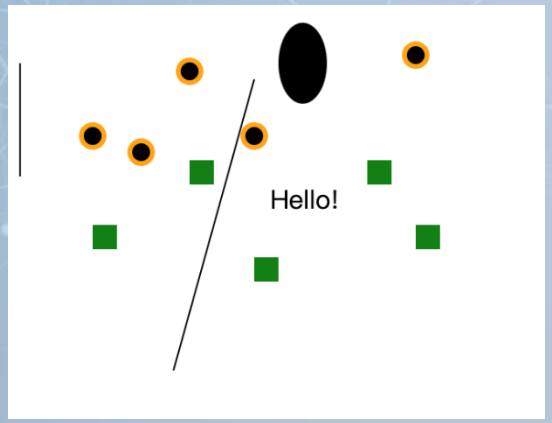


SVG - Shape

- On SVG, we can draw
  - Lines: <line>
  - Rectangles: <rect>
  - Circles: <circle>
  - Ellipse: <ellipse>
  - Text: <text>
  - Polygon: <polygon> 
  - Path: <path> (curve defined by points) 
  - …….

Example (Ex01-4)

- Draw lines, rectangles, circles, ellipses and text on SVG
- Use CSS
- index.html

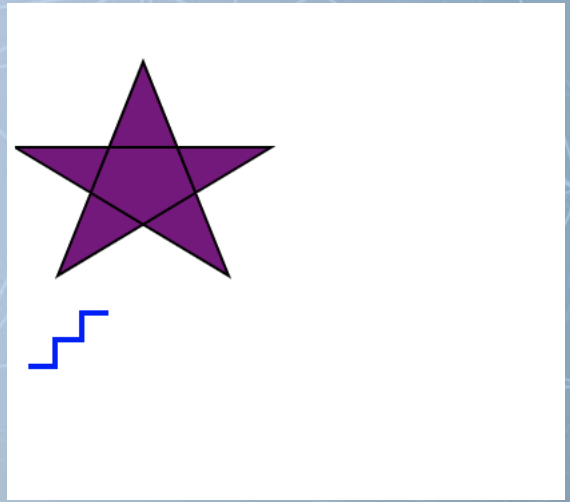



```html
<!doctype html>
<html>
<head>
    <title>Example</title>
    <style>
        rect{
            fill: green;
        }

        circle{
            stroke: orange;
            stroke-width: 3;
        }
    </style>
</head>

<body>
    <svg width="500" height="300">
        <line x1="5" x2="5" y1="100" y2="30" stroke="black"/>
          <line x1="100" x2="150" y1="220" y2="40" stroke="black"/>

          <rect x="150" y="150" width="15" height="15"/>
          <rect x="220" y="90" width="15" height="15"/>
          <rect x="110" y="90" width="15" height="15"/>
          <rect x="220" y="90" width="15" height="15"/>
          <rect x="50" y="130" width="15" height="15"/>
          <rect x="250" y="130" width="15" height="15"/>

          <circle cx="250" cy="25" r="7"/>
          <circle cx="150" cy="75" r="7"/>
          <circle cx="80" cy="85" r="7"/>
          <circle cx="110" cy="35" r="7"/>
          <circle cx="50" cy="75" r="7"/>

          <ellipse cx="180" cy="30" rx="15" ry="25"/>

          <text x="160" y="120">Hello!</text>
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```

Example (Ex01-5)

- <polygon>
- <polyline>

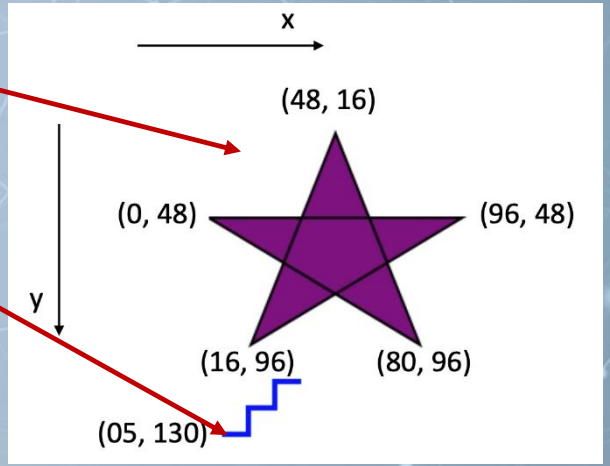
- File
  - Index.html



Example (Ex01-5)

- Check the coordinates

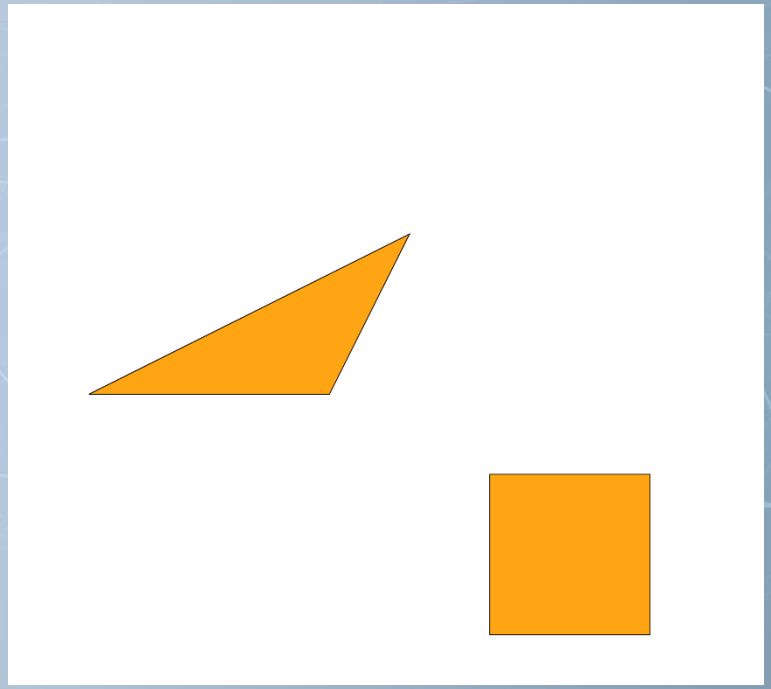
```
<body>
    <svg>
    <polygon style="fill: purple; stroke: black;"
        points="48,16  16,96  96,48  0,48  80,96" />

    <polyline fill="none" stroke="blue" stroke-width="2"
        points="05,130
                15,130
                15,120
                25,120
                25,110
                35,110" />
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```



Example (Ex01-6)

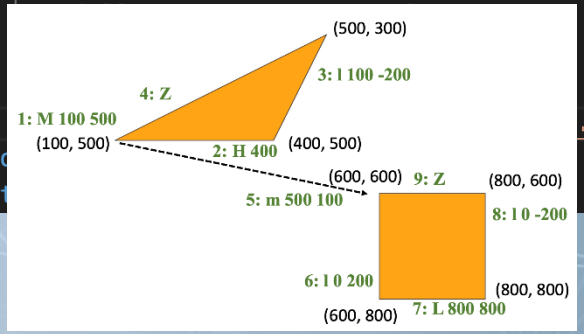
- <path>

- File:
  - Index.html



Example (Ex01-6)

```
<body>
    <svg width="1000" height="1000">
        <path d="
                M 100 500
                H 400
                l 100 -200
                Z

                m 500 100
                l 0 200
                L 800 800
                l 0 -200
                Z"
```

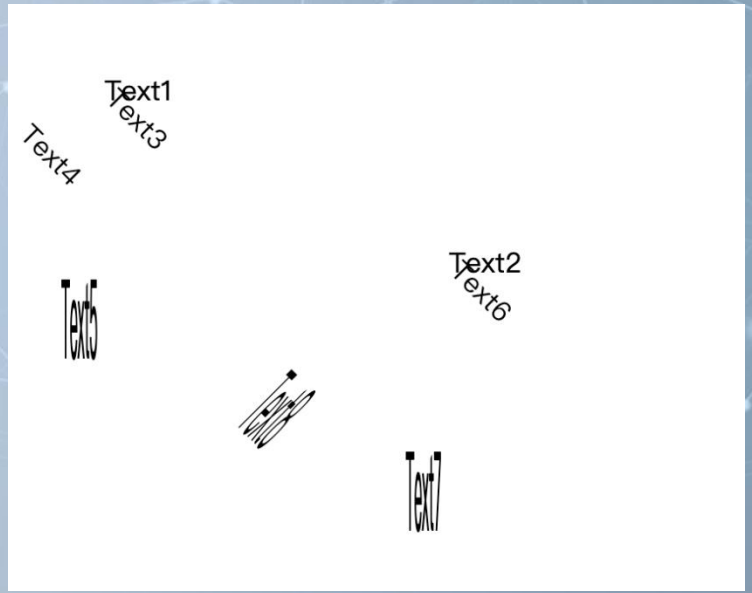


- M = move (pen) to
  - m = move by
- L = line to
  - l = line by
- H = horizontal line to
- V = vertical line to
- C = curve to
- S = smooth curve to
- Q = quadratic Bézier curve
- T = smooth quadratic Bézier curve to
- A = elliptical Arc
- Z = close path

- More details: check https://www.w3schools.com/graphics/svg_path.asp
- play to see different effect : https://yqnn.github.io/svg-path-editor/

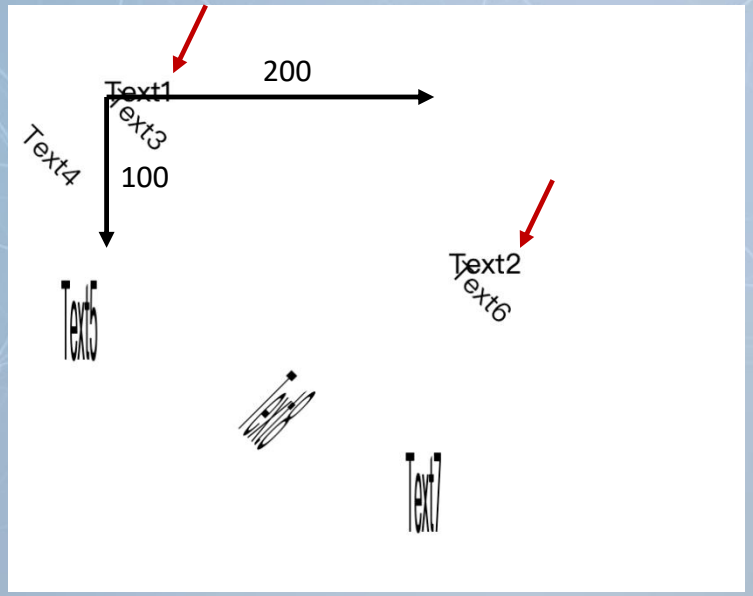
Example (Ex01-7)

- SVG – Transformation
  - translate
  - scale
  - rotate
- File
  - index.html



Example (Ex01-7)

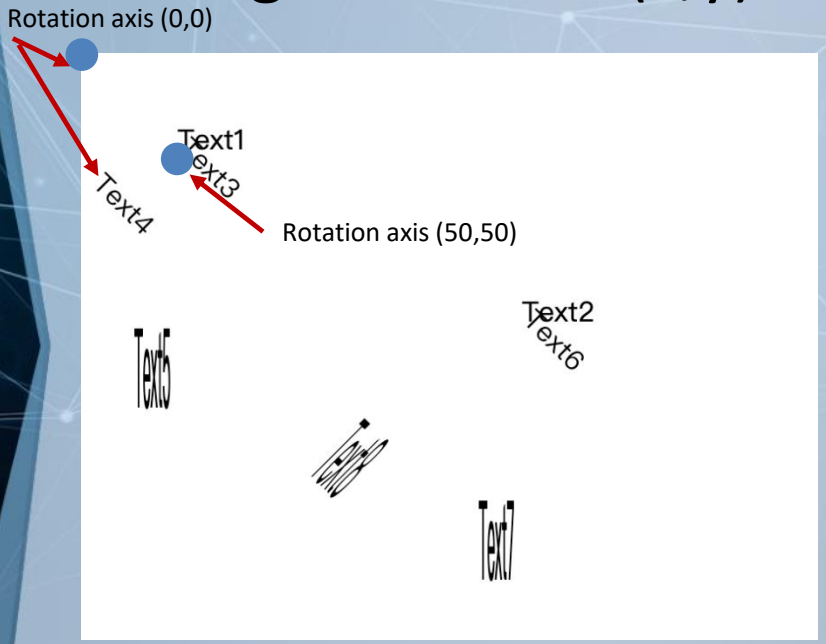
- translate – move by (dx, dy)




```
<!doctype html>
<html>
<head>
    <title>Example</title>
</head>

<body>
    <svg width="1000" height="1000">
        <text x="50" y="50">
            Text1
        </text>
        <text x="50" y="50" transform="translate(200,100)">
            Text2
        </text>
        <text x="50" y="50" transform="rotate(45, 50, 50)">
            Text3
        </text>
        <text x="50" y="50" transform="rotate(45, 0, 0)">
            Text4
        </text>
        <text x="50" y="50" transform="scale(0.5,4.0)">
            Text5
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50)">
            Text6
        </text>
        <text x="50" y="50" transform="translate(200,100) scale(0.5,4.0)">
            Text7
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50) scale(0.5,4.0) ">
            Text8
        </text>
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```

Example (Ex01-7)

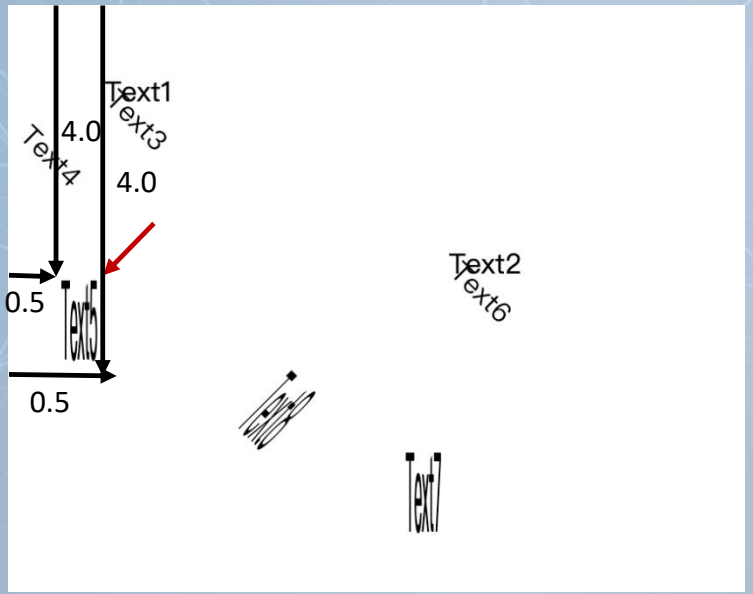
- rotate(a, x, y) – rotate a shape by ‘a’ degrees about (x,y)




```
<!doctype html>
<html>
<head>
    <title>Example</title>
</head>

<body>
    <svg width="1000" height="1000">
        <text x="50" y="50">
            Text1
        </text>
        <text x="50" y="50" transform="translate(200,100)">
            Text2
        </text>
        <text x="50" y="50" transform="rotate(45, 50, 50)">
            Text3
        </text>
        <text x="50" y="50" transform="rotate(45, 0, 0)">
            Text4
        </text>
        <text x="50" y="50" transform="scale(0.5,4.0)">
            Text5
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50)">
            Text6
        </text>
        <text x="50" y="50" transform="translate(200,100) scale(0.5,4.0)">
            Text7
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50) scale(0.5,4.0) ">
            Text8
        </text>
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```

Example (Ex01-7)

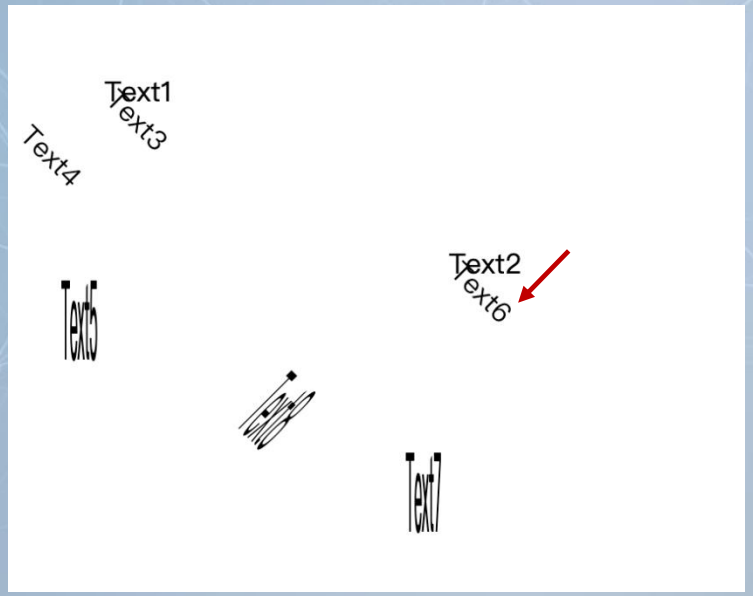
- scale(sx, sy)




```
<!doctype html>
<html>
<head>
    <title>Example</title>
</head>

<body>
    <svg width="1000" height="1000">
        <text x="50" y="50">
            Text1
        </text>
        <text x="50" y="50" transform="translate(200,100)">
            Text2
        </text>
        <text x="50" y="50" transform="rotate(45, 50, 50)">
            Text3
        </text>
        <text x="50" y="50" transform="rotate(45, 0, 0)">
            Text4
        </text>
        <text x="50" y="50" transform="scale(0.5,4.0)">
            Text5
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50)">
            Text6
        </text>
        <text x="50" y="50" transform="translate(200,100) scale(0.5,4.0)">
            Text7
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50) scale(0.5,4.0) ">
            Text8
        </text>
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```

Example (Ex01-7)

- Multiple transformations - from left to right

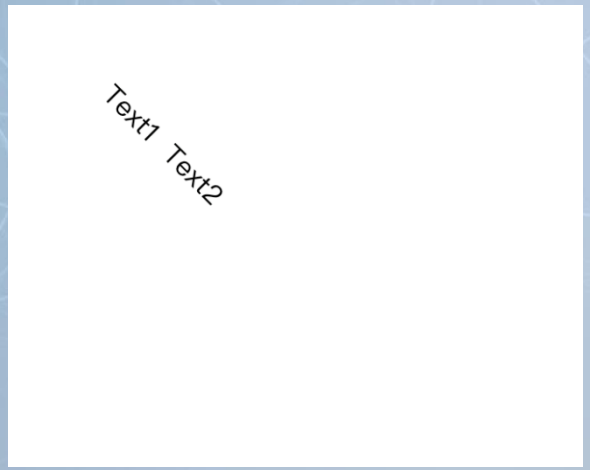



```
<!doctype html>
<html>
<head>
    <title>Example</title>
</head>

<body>
    <svg width="1000" height="1000">
        <text x="50" y="50">
            Text1
        </text>
        <text x="50" y="50" transform="translate(200,100)">
            Text2
        </text>
        <text x="50" y="50" transform="rotate(45, 50, 50)">
            Text3
        </text>
        <text x="50" y="50" transform="rotate(45, 0, 0)">
            Text4
        </text>
        <text x="50" y="50" transform="scale(0.5,4.0)">
            Text5
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50)">
            Text6
        </text>
        <text x="50" y="50" transform="translate(200,100) scale(0.5,4.0)">
            Text7
        </text>
        <text x="50" y="50" transform="translate(200,100) rotate(45, 50, 50) scale(0.5,4.0) ">
            Text8
        </text>
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```

Example (Ex01-8)

<g>

group multiple shapes




```
<!doctype html>
<html>
<head>
    <title>Example</title>
</head>

<body>
    <svg width="1000" height="1000">
        <g transform="rotate(45,50,50)">
            <text x="50" y="50">
                Text1
            </text>
            <text x="100" y="50">
                Text2
            </text>
        </g>
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```

Example (Ex01-9)

- SVG has no layer
  - Any pixel-paint applied later obscure any earlier paint and therefore appear to be “in front”

```
<!doctype html>
<html>
<head>
    <title>Example</title>
</head>

<body>
    <svg>
        <rect x="0" y="0" width="30" height="30" fill="purple"/>
        <rect x="20" y="5" width="30" height="30" fill="blue"/>
        <rect x="40" y="10" width="30" height="30" fill="green"/>
        <rect x="60" y="15" width="30" height="30" fill="yellow"/>
        <rect x="80" y="20" width="30" height="30" fill="red"/>
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
</body>
</html>
```

SVG Elements Reference

- https://developer.mozilla.org/en-US/docs/Web/SVG/Element

SVG elements A to Z

A

- <a>
- <animate>
- <animateMotion>
- <animateTransform>

C

- <circle>
- <clipPath>
- <color-profile>

D

- <defs>
- <desc>
- <discard>

E

- <ellipse>

F

- <feBlend>
- <feColorMatrix>
- <feComponentTransfer>
- <feComposite>
- <feConvolveMatrix>
- <feDiffuseLighting>
- <feDisplacementMap>
- <feDistantLight>
- <feDropShadow>
- <feFlood>
- <feFuncA>
- <feFuncB>
- <feFuncG>
- <feFuncR>
- <feGaussianBlur>
- <feImage>
- <feMerge>
- <feMergeNode>
- <feMorphology>
- <feOffset>
- <fePointLight>
- <feSpecularLighting>
- <feSpotLight>
- <feTile>
- <feTurbulence>
- <filter>
- <foreignObject>

G

- <g>

H

- <hatch>
- <hatchpath>

I

- <image>

L

- <line>
- <linearGradient>

M

- <marker>
- <mask>
- <mesh>
- <meshgradient>
- <meshpatch>
- <meshrow>
- <metadata>
- <mpath>

P

- <path>
- <pattern>
- <polygon>
- <polyline>

R

- <radialGradient>
- <rect>

S

- <script>
- <set>
- <solidcolor>
- <stop>
- <style>
- <svg>
- <switch>
- <symbol>

T

- <text>
- <textPath>
- <title>
- <tspan>

U

- <unknown>
- <use>

V

- <view>

Brief Introduction to JavaScript

High-level definition

HTML

- Define paragraphs, headings, data tables, or embed pictures and videos on the page.

CSS

- Set the background color, font, or make the content presented in multiple columns 。

JavaScript

- Allow you to dynamically update content, control multimedia, animation --- almost everything.

Console

- The console is a panel that displays important messages, like errors, for developers. We can print, or log, to our console directly.

semi-colon(;)

- In JavaScript, all code instructions should end with a semi-colon (;)
- Your code may work correctly for single lines, but probably won't when you are writing multiple lines of code together. Try to get into the habit of including it.

Comments

- Single line
- Multiple line

```
> // Single line comment
< undefined
> /* multiple line comment
  multiple line comment
  multiple line comment */
< undefined
```

Declaring variables

```
> let x;
< undefined
> x = 2021;
< 2021
> console.log(x);
  2021
```

```
> const a;
✖ Uncaught SyntaxError: Missing initializer in const declaration
> const a = 50;
< undefined
> a = 80;
✖ ▶Uncaught TypeError: Assignment to constant variable.
      at <anonymous>:1:3
```

const

- Declare read-only variables.

The difference between var and let

- why do we need two keywords for defining variables? Why have **var** and **let**?

```
var myName = 'Chris';
var myName = 'Bob';
```

```
let myName = 'Chris';
let myName = 'Bob';
```

```
let myName = 'Chris';
myName = 'Bob';
```

Variable types

- Numbers
- Strings
- Booleans
- Dictionary
- Arrays
- Objects

Array

- Arrays consist of square brackets and elements that are separated by commas.

```
> let x = [1,2,3,'string',[5,6,7]];
< undefined
```

- You can then access individual items in the array using bracket notation.

```
> console.log(x[0]);
  1
```

- You can find out the length of an array (how many items are in it) by using the **length** property.

```
> console.log(x.length);
  5
```

Dictionary

```
> let dic = {'name': 'Taipei', 'population': 2000000, 'Region': 'north'};
< undefined
> console.log(dic);
  ▶ {name: "Taipei", population: 2000000, Region: "north"}
< undefined
> console.log(dic.name);
  Taipei
< undefined
> console.log(dic['name']);
  Taipei
< undefined
> console.log(dic.population);
  2000000
< undefined
> console.log(dic['population']);
  2000000
< undefined
```

Array of Dictionary

```
> let dic = [{'name': 'Taipei', 'population': 2000000, 'Region': 'north'},
             {'name': 'Kaohsiung', 'population': 1700000, 'Region': 'south'},
             {'name': 'Tainan', 'population': 1500000, 'Region': 'south'}

            ];
< undefined
> console.log(dic[1].population);
  1700000
< undefined
> console.log(dic[2].name);
  Tainan
< undefined
> console.log(dic[0]['Region']);
  north
< undefined
```

Conditions

- Logical Operators
- if...else statements
- else if clause
- switch statements

Logical operators

- Or ||

```
> (3 < 5) || (3 > 5)
<- true
```

- And &&

```
> (3 < 5) && (3 > 5)
<- false
```

- Not !

```
> !(3 < 5)
<- false
```

if...else

```
> let x = 2021;
< undefined
> if(x===2021){
      console.log(x+100);
  }
  else{
      console.log(x);
  }
  2121
```

else if

```
> if(x===2021){
      console.log(x+100);
  }
  else if(x===2022){
      console.log(x+200);
  }
  else if(x===2023){
      console.log(x+300);
  }
  else{
      console.log(x);
  }
```

function

```
> function foo(){
      console.log("hello world!");
  }
< undefined
> foo();
  hello world!
```

```
> let foo2 = foo;
< undefined
> foo2();
  hello world!
```

Arrow function

```
> function foo(){
      console.log("hello world!");
  }
< undefined
> foo();
  hello world!
```

```
> let foo = () => {
      console.log("hello world!");
  }
< undefined
> foo();
  hello world!
```

Arrow function

```
> let foo = function(d){ return d*2; }
< undefined
> foo(3)
< 6
```

```
> let foo = (d) => (d*2);
< undefined
> foo(3)
< 6
```

++ and --

- ++

```
> let x = 2021;
< undefined
> console.log(x++);
  2021
< undefined
> console.log(x);
  2022
```

- --

```
> let x = 2021;
< undefined
> console.log(--x);
  2020
< undefined
> console.log(x);
  2020
```

Loops

- while loop
- do...while loop
- for loop

while

```
> while(x>0){
      console.log(x);
      x = x - 1;
  }
  5
  4
  3
  2
  1
```

do...while

```
> let x = 3;
< undefined
> do{
      console.log(x);
      x = x - 1;
  }while(x>3);
```

for

```
> let x = [2020,2021,2023,2024];
< undefined
> for(let i = 0 ; i < x.length ; i++){
      console.log(x[i]);
  }
  2020
  2021
  2023
  2024
```

Array.prototype.forEach()

```
> let x = [2020,2021,2023,2024];
< undefined
> for(let i = 0 ; i < x.length ; i++){
      console.log(x[i]);
  }
  2020
  2021
  2023
  2024
```

```
>>> x = [2020,2021,2023,2024]
>>> for i in range(len(x)):
...     print(x[i])
...
2020
2021
2023
2024
```

```
> function foo(a){
      console.log(a);
  }
< undefined
> x.forEach(foo);
  2020
  2021
  2023
  2024
```

```
> x.forEach(function(a){
      console.log(a);
  });
  2020
  2021
  2023
  2024
```

```
> x.forEach((a) => {
      console.log(a);
  });
  2020
  2021
  2023
  2024
```

```
> x.forEach(a => console.log(a));
  2020
  2021
  2023
  2024
```

Array.prototype.forEach()

```
> let x = [2021, 2022, 2023, 2024]
<- undefined
> x.forEach(function(d, i)
            {
                console.log(d, i);
            });
  2021 0
  2022 1
  2023 2
  2024 3
```

```
> let x = [2021, 2022, 2023, 2024];
<- undefined
> x.forEach((d,i)=>{
                console.log(d,i);
            });
  2021 0
  2022 1
  2023 2
  2024 3
```

Focal Point

- HTML DOM Struct.
- CSS saves a lot of work
- SVG
  - coordinates system
  - Path
  - g
  - SVG has no layer
- Javascript
  - Declaring variables
  - Array of Dictionary
  - Arrow Function
  - forEach

Reference

- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://www.codecademy.com/catalog/language/javascript