Information Visualization
Coding Session 1

2024.03.13.

Seongouk Kim

HAiV

Web Development

HAiV

Frontend

Focuses on layout, animations, content organization, navigation, graphics.

Programming languages:
JavaScript, HTML, CSS

Backend

Focuses on building code, debugging, database management.

Programming languages:
Node.js, Python, Java

Web Development

Environment

HAiV

Editor: Any text editor (even notepad!)

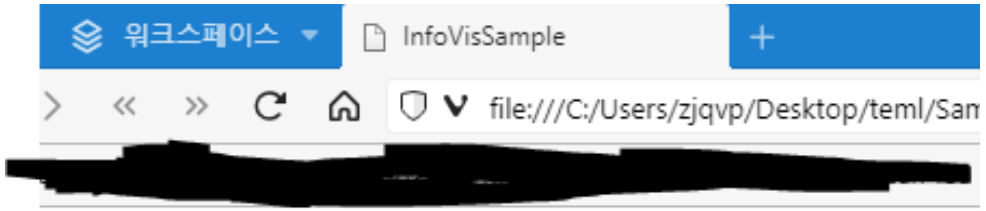
- **But recommend to use proper editor (vscode, webstorm, etc)**

Rendering: Any web browsers which supports HTML5 (newer than IE8)

HTML (Hyper Text Markup Language)

Sample.html

```
Sample.html > html > body > ul > li
1   <!DOCTYPE html>
2   <html>
3
4   <head>
5       <title> InfoVisSample </title>
6   </head>
7
8   <body>
9       <h1>Hello World!</h1>
10      <h2>This is Sample Page!</h2>
11
12      <p>And this is sample List! </p>
13      <ul>
14          <li>HII</li>
15          <li>HELLOO</li>
16          <li>BYEEE</li>
17      </ul>
18  </body>
19
20  </html>
```



Hello World!

This is Sample Page!

And this is sample List!

- HII
- HELLOO
- BYEEE

CSS (Cascading Style Sheet)

```html
<head>
    <title> InfoVisSample </title>
    <link rel="stylesheet" href="style.css">
</head>
```

Sample.html | # style.css

style.css > p

```css
h1 {
    font-size: 64px;
}

h2 {
    font-size: 56px;
}

p {
    font-size: 24px;
    color: blueviolet;
}

li {
    font-style: italic;
}
```

Hello World!

This is Sample Page!

And this is sample List!

- HII
- HELLOO
- BYEEE

JavaScript

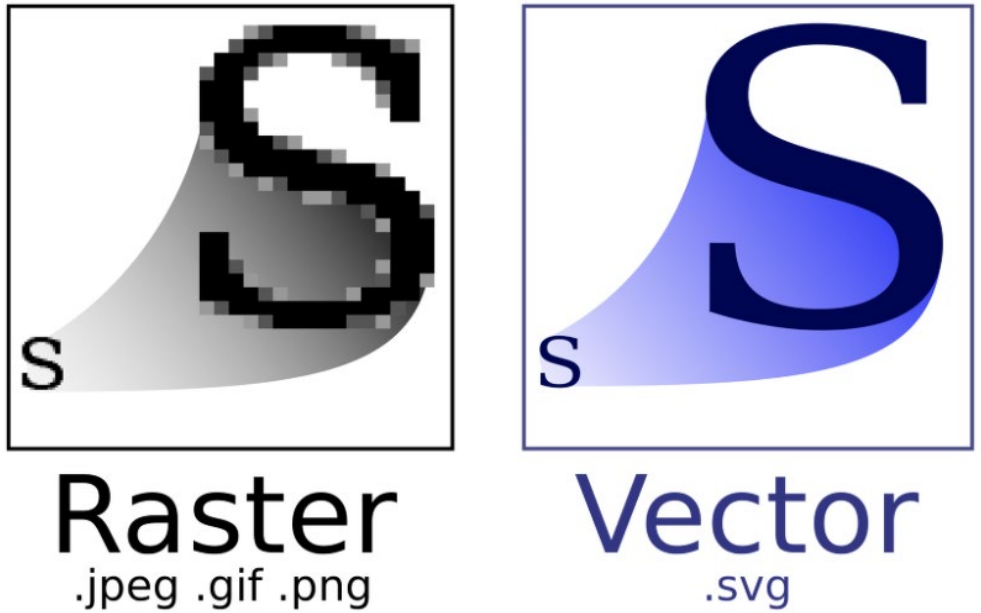
```
<p id="target">And this is sample List! </p>
<ul>
    <li>HII</li>
    <li>HELLOO</li>
    <li>BYEEE</li>
</ul>

<button type="button" onclick="SampleFunction()">Button</button>
<script src="index.js"></script>
```

```
function SampleFunction() {
    document.getElementById("target").innerHTML = "Text Changed!";
}
```

SVG (Scalable Vector Graphics)

HAiV

- XML (eXtensible Markup Language) to generate Vector Image
- Can generate directly in HTML or dynamically in JavaScript
- Quality doesn’t change when resizing
- Depth order follows the generated order

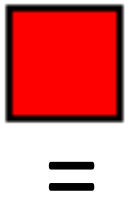


SVG with HTML

```
<!-- Modified Code from https://github.com/dryjins/d3-starter-template  -->

<svg width="450" height="100">

<!-- Rectangle (x and y specify the coordinates of the upper-left corner -->
<rect x="10" y="20" width="50" height="50" fill="red" stroke="black" stroke-width = "3px" />

<!-- Circle: cx and cy specify the coordinates of the center and r the radius -->
<circle cx="100" cy="45" r="25" fill="green" stroke="black" stroke-width = "3px"  />

<!-- Ellipse: rx and ry specify separate radius values -->
<ellipse cx="150" cy="45" rx="15" ry="25" fill="purple" stroke="black" stroke-width = "3px"  />

<!-- Line: x1,y1 and x2,y2 specify the coordinates of the ends of the line -->
<line x1="185" y1="20" x2="230" y2="70" stroke="black" stroke-width="12px"  />

<!-- Text: x specifies the position of the left edge and y specifies the vertical position of the baseline -->
<text x="260" y="60" fill="Yellow" font-size = "42px" stroke="black" stroke-width = "1px" >SVG Text</text>

</svg>
```

SVG with D3.js

```svg
<svg width="450" height="100">
<!-- Rectangle (x and y specify the coordinates of the upper-left corner -->
<rect x="10" y="20" width="50" height="50" fill="red" stroke="black" stroke-width = "3px" />
</svg>
```

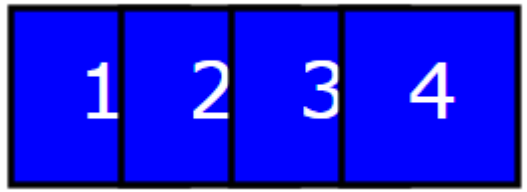


=

```d3
const svgWidth = 450;
const svgHeight = 100;
const svg = d3.select("body").append("svg")
    .attr("width", svgWidth)
    .attr("height", svgHeight);

// Draw Rectangle
svg.append("rect")
    .attr("x", 10)
    .attr("y", 20)
    .attr("width", 50)
    .attr("height", 50)
    .attr("fill", "red")
    .attr("stroke", "black")
    .attr("stroke-width", "3px");
```

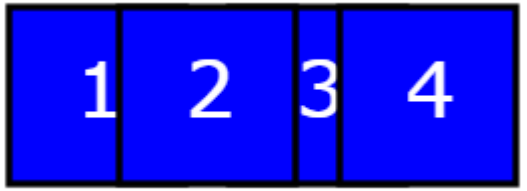
Depth order of SVG objects

```
<g>
  <rect x="10" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="40" y="60" font-family="Verdana" font-size="35" fill="white">1</text>
</g>

  <g>
  <rect x="60" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="90" y="60" font-family="Verdana" font-size="35" fill="white">2</text>
</g>

  <g>
  <rect x="110" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="140" y="60" font-family="Verdana" font-size="35" fill="white">3</text>
</g>

  <g>
  <rect x="160" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="190" y="60" font-family="Verdana" font-size="35" fill="white">4</text>
</g>
```



Depth order of SVG objects

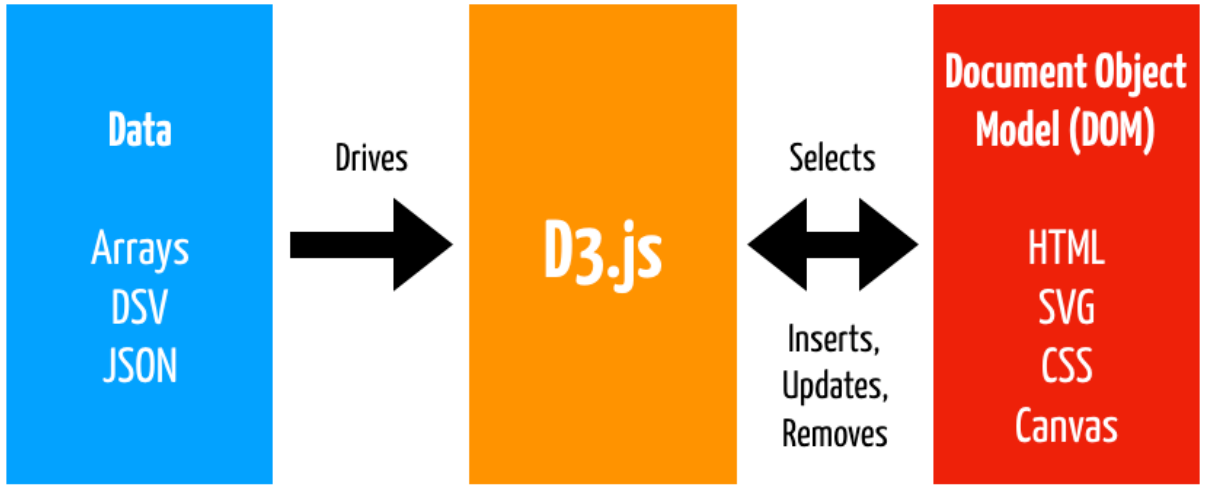
```
<g>
  <rect x="10" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="40" y="60" font-family="Verdana" font-size="35" fill="white">1</text>
</g>

  <g>
  <rect x="110" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="140" y="60" font-family="Verdana" font-size="35" fill="white">3</text>
</g>

  <g>
  <rect x="160" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="190" y="60" font-family="Verdana" font-size="35" fill="white">4</text>
</g>

  <g>
  <rect x="60" y="10" width="80" height="80" fill="blue" stroke="black", stroke-width="3px"></rect>
  <text x="90" y="60" font-family="Verdana" font-size="35" fill="white">2</text>
</g>
```



D3 (Data-Driven Document)

HAiV

- Open-source JS library to create dynamic and interactive visualization
- Manipulate documents in data-driven manner



D3 Environment Set Up

HAiV

- **Just add this code in HTML!**
- `<script src="https://d3js.org/d3.v7.min.js"></script>`

```
<head>
    <title> InfoVisSample </title>
    <link rel="stylesheet" href="style.css">
    <script src="https://d3js.org/d3.v7.min.js"></script>
</head>
```

D3 Basic - Selection

- **select(“*selector*”) – Return first element matching with selector**

```
d3.select("li").style("color", "blue");
```

- *HII*
- *HELLOO*
- *BYEEE*

D3 Basic - Selection

HAiV

- **selectAll(“*selector*”) – Return All elements matching with selector**

```
d3.selectAll("li").style("color", "blue");
```

- *HII*
- *HELLOO*
- *BYEEE*

D3 Basic - Manipulation

- append(“*element*”) – Add *element* at the end of the selected element.
- insert(“*element*”, “location”) - Add *element* in the selected element, but right before the selected location.

```
d3.select("body").append("p").text("D3 appended me!");

d3.select("body").insert("p", "ul").text("And D3 insterted me before list!");
```

D3 Basic - Manipulation

- attr(“attribute”, value) - Set attribute of selected element into value

```
const svgWidth = 450;
const svgHeight = 100;
const svg = d3.select("body").append("svg")
    .attr("width", svgWidth)
    .attr("height", svgHeight);

// Draw Rectangle
svg.append("rect")
    .attr("x", 10)
    .attr("y", 20)
    .attr("width", 50)
    .attr("height", 50)
    .attr("fill", "red")
    .attr("stroke", "black")
    .attr("stroke-width", "3px");
```

D3 Basic – Data Binding

- **Can load external data with d3.csv(), d3.json(), d3.tsv(), etc**

*** To load external data, you should run the web on the server and follow CORS policy. If you don't, browsers will prohibit importing the data.**

- **data(*data*) – Bind *data* to selected elements**

```
const sampleData = [1, 2, 3];

d3.select("ul").selectAll("li").data(sampleData).text(function (d) { return d; });
```

D3 Basic – Data Binding

- **enter() – Dynamically create placeholder references corresponding to the number of data values**

```
const sampleData = [1, 2, 3, 4, 5, 6];
```

```
const li = d3.select("ul").selectAll("li").data(sampleData).text(function (d) { return d; });

li.enter().append("li").text(function (d) { return d; });
```

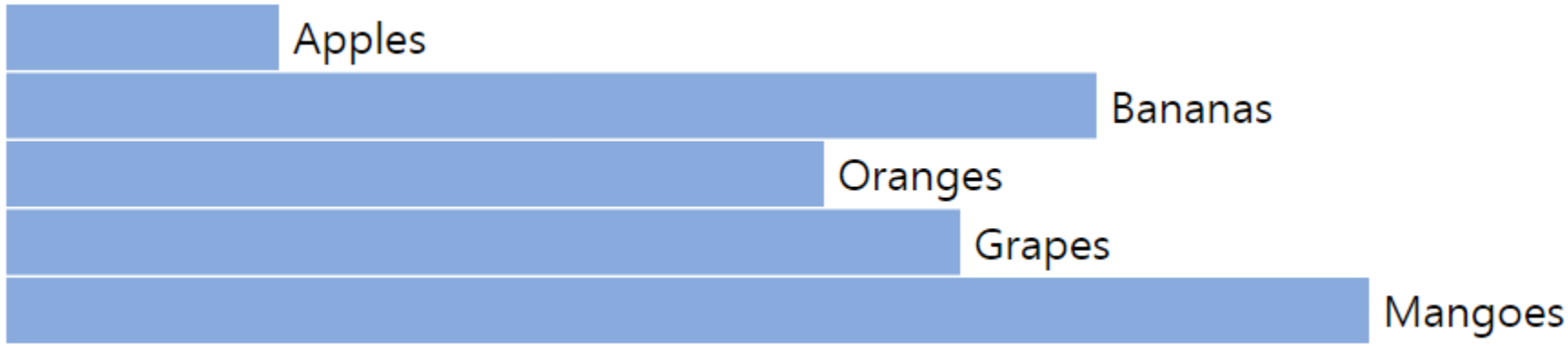
- 1
- 2
- 3
- 4
- 5
- 6

D3 Basic

```
const data = [
    { "Category": "Apples", "Value": 10 },
    { "Category": "Bananas", "Value": 40 },
    { "Category": "Oranges", "Value": 30 },
    { "Category": "Grapes", "Value": 35 },
    { "Category": "Mangoes", "Value": 50 }
];
```

```
// modified code from https://www.tutorialsteacher.com/d3js/create-svg-chart-in-d3js
const width = 700,
    scaleFactor = 10,
    barHeight = 25;



const graph = d3.select("body")
    .append("svg")
    .attr("width", width)
    .attr("height", barHeight * data.length);

const bar = graph.selectAll("g")
    .data(data)
    .enter()
    .append("g")
    .attr("transform", function (d, i) {
        return "translate(0," + i * barHeight + ")";
    });

bar.append("rect")
    .attr("width", function (d) {
        return d.Value * scaleFactor;
    })
    .attr("height", barHeight - 1)
    .attr("fill", "#88AADD");

bar.append("text")
    .attr("x", function (d) { return (d.Value * scaleFactor + 5); })
    .attr("y", barHeight / 2)
    .attr("dy", ".35em")
    .text(function (d) { return d.Category; });
```

D3 Basic



D3 Basic - Scaling

- **Map input ranges(Domain) into output ranges(Range)**

Scale Type	Method	Description
Continuous	d3.scaleLinear()	Construct continuous linear scale where input data (domain) maps to specified output range.
	d3.scaleIdentity()	Construct linear scale where input data is the same as output.
	d3.scaleTime()	Construct linear scale where input data is in dates and output in numbers.
	d3.scaleLog()	Construct logarithmic scale.
	d3.scaleSqrt()	Construct square root scale.
	d3.scalePow()	Construct exponential scale.
Sequential	d3.scaleSequential()	Construct sequential scale where output range is fixed by interpolator function.
Quantize	d3.scaleQuantize()	Construct quantize scale with discrete output range.
Quantile	d3.scaleQuantile()	Construct quantile scale where input sample data maps to discrete output range.
Threshold	d3.scaleThreshold()	Construct scale where arbitrary input data maps to discrete output range.
Band	d3.scaleBand()	Band scales are like ordinal scales except the output range is continuous and numeric.
Point	d3.scalePoint()	Construct point scale.
Ordinal	d3.scaleOrdinal()	Construct ordinal scale where input data includes alphabets and are mapped to discrete numeric output range.

D3 Basic - Axes

HAiV

- **Renders human-readable reference marks based on the scales**

Axis Method	Description
d3.axisTop()	Creates top horizontal axis.
d3.axisRight()	Creates vertical right-oriented axis.
d3.axisBottom()	Creates bottom horizontal axis.
d3.axisLeft()	Creates left vertical axis.

D3 Basic

```
const secondData = [
    { "Date": "2022-01-01", "Value": 105 },
    { "Date": "2022-01-02", "Value": 12 },
    { "Date": "2022-01-03", "Value": 167 },
    { "Date": "2022-01-04", "Value": 45 },
    { "Date": "2022-01-05", "Value": 82 },
    { "Date": "2022-01-06", "Value": 32 },
    { "Date": "2022-01-07", "Value": 190 },
    { "Date": "2022-01-08", "Value": 55 },
    { "Date": "2022-01-09", "Value": 88 },
    { "Date": "2022-01-10", "Value": 130 },
    { "Date": "2022-01-11", "Value": 175 },
    { "Date": "2022-01-12", "Value": 60 },
    { "Date": "2022-01-13", "Value": 100 },
    { "Date": "2022-01-14", "Value": 25 },
    { "Date": "2022-01-15", "Value": 140 },
    { "Date": "2022-01-16", "Value": 185 },
    { "Date": "2022-01-17", "Value": 72 },
    { "Date": "2022-01-18", "Value": 95 },
    { "Date": "2022-01-19", "Value": 150 },
    { "Date": "2022-01-20", "Value": 40 }
];
```

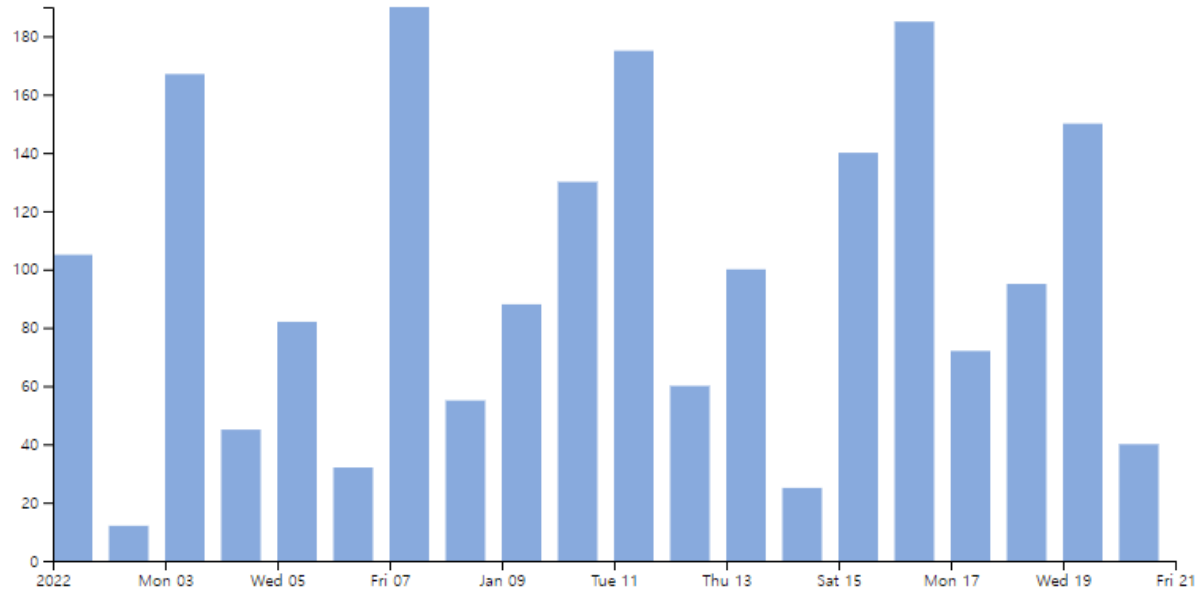
```
parsedData = secondData.map(d => {
    return { Date: d3.timeParse("%Y-%m-%d")(d.Date), Value: d.Value }
})

// Set time offset at the end one day
var extent = d3.extent(parsedData, function (d) { return d.Date });
extent[1] = d3.timeDay.offset(extent[1], 1); // Add one day to the end date

// Set x,y scale
var x = d3.scaleTime().domain(extent).range([0, barChartWidth]);
var y = d3.scaleLinear().domain([0, d3.max(parsedData, function (d) { return d.Value })]).range([barChartHeight, 0]);



// Add the x Axis
svg.append("g")
    .attr("transform", "translate(0," + barChartHeight + ")")
    .call(d3.axisBottom(x))
    .append('text')
    .attr('text-anchor', 'middle')
    .text('Date');

// Add the y Axis
svg.append("g")
    .call(d3.axisLeft(y));
```

D3 Basic


180
160
140
120
100
80
60
40
20
0
2022
Mon 03
Wed 05
Fri 07
Jan 09
Tue 11
Thu 13
Sat 15
Mon 17
Wed 19
Fri 21

D3 Basic

```
//Draw Bars
var g = svg.append("g")

g.selectAll(".bar")
    .data(parsedData)
    .enter()
    .append("rect")
    .attr("class", "bar")
    .attr("x", d => { return x(d.Date) })
    .attr("y", d => { return y(d.Value) })
    .attr("width", barWidth)
    .attr("height", function (d) {
        return barChartHeight - y(d.Value);
    })
    .attr("fill", "#88AADD")
```



* Added this code above axis code

Sources

- https://www.w3schools.com - Basics of HTML/JS/CSS & SVG
- https://www.tutorialsteacher.com/d3js - Basics of D3
- https://observablehq.com/@d3/gallery - Various examples of D3 examples

Thank you!