- Initially released in 2011
- BSD License (= open source)
- D3 = Data-Driven Documents
- Bind data to DOM and apply dynamic transformations
- Not only targeted at SVG, but also other HTML elements
- Well-suited for creating animated interactive diagrams

Developer Mike Bostock

Introduction to D3.js

Getting started

Installation:

a) Load directly from the Internet

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

b) Download latest “d3.zip“ from https://github.com/d3/d3/releases

```
<script src="d3.js"></script>
```

c) Run “npm install d3”

```
<script src="node_modules/d3/dist/d3.js"></script>
```

d) Use a module bundler (e.g. Webpack)

```
<script src="main.js"></script>
```

```
<script>
    d3.select("body")                  <body>
        .append("p")            →          <p>Hello World!</p>  →  Hello World!
        .text("Hello World!");         </body>
</script>
```

Method chaining

- Similar to jQuery

```
const svg = d3.select("body")
      .append("svg")
      .attr("width", 400)
      .attr("height", 225)
      .style("background-color", "green");
```

setters

```
→ <svg width="400" height="225" style="background-color: green;"></svg>
```

↓

```
svg.attr("width");
```

getter

```
→ 400
```

Selections

- Similar to document.querySelector or CSS selectors

```
<div id="app">
    <div class="container">
        <svg>
            <rect x="10" y="10" width="280" height="130" fill="red"></rect>
        </svg>
    </div>
    <div class="container">
        <svg>
            <rect x="10" y="10" width="280" height="130" fill="blue"></rect>
        </svg>
    </div>
</div>

const appElement = d3.select("#app");
appElement.node() → <div id="app">

const containerElements = appElement.selectAll(".container");
containerElements.nodes() → [<div class="container">, <div class="container">]

const image2Element = containerElements.select("rect[fill='blue']");
image2Element.node() → <rect x="10" y="10" width="280" height="130" fill="blue"></rect>
```

Data binding

- Distinct feature of D3.js

```
const bind = data => {
    const divs = d3.select("body")
        .selectAll("div")
        .data(data)
        .text((d, i) => "Updated Div " + i)
        .style("color", d => d);

    divs.enter()
        .append("div")
        .text((d, i) => "New Div " + i)
        .style("color", d => d);

    divs.exit()
        .text((d, i) => "Removed Div " + i)
        .style("color", "black");
    //or use .remove() to delete elements from the DOM
};
```

```
bind(["red", "green", "blue"]);
```

→ New Div 0
New Div 1
New Div 2

```
bind(["green", "blue", "red"]);
```

→ Updated Div 0
Updated Div 1
Updated Div 2

```
bind([]);
```

→ Removed Div 0
Removed Div 1
Removed Div 2

Data requesting and parsing

- Formats: JSON, CSV, XML, …
- More convenient than HTML5 Fetch API

```
const data = await d3.csv("src/cars.csv", d => ({
    year: new Date(Number(d.Year), 0, 1),
    make: d.Make,
    model: d.Model,
    length: Number(d.Length)
}));
```

cars.csv

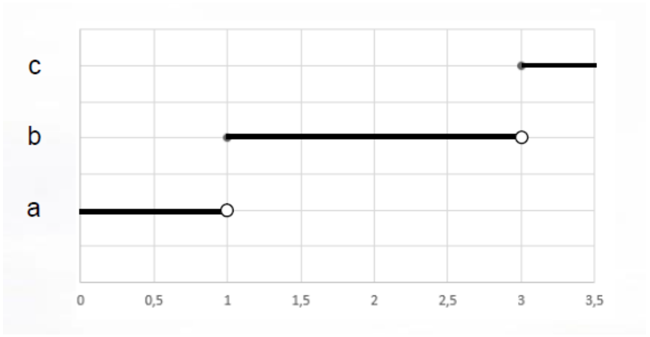
```
Year,Make,Model,Length
1997,Ford,E350,2.34
2000,Mercury,Cougar,2.38
```

```
data[0]
→{ year: Date Wed Jan 01 1997 00:00:00 GMT+0100 (Central European Standard Time),
    make: "Ford", model: "E350", length: 2.34 }

data[1]
→{ year: Date Sat Jan 01 2000 00:00:00 GMT+0100 (Central European Standard Time),
    make: "Mercury", model: "Cougar", length: 2.38 }
```

Scales

- Functions which map values from an input domain to an output range

```
const t = d3.scaleThreshold()
    .domain([1, 3])
    .range(["a", "b", "c"]);

t(0) → "a"
t(1) → "b"
t(2) → "b"
t(3) → "c"
t(1000) → "c"
```



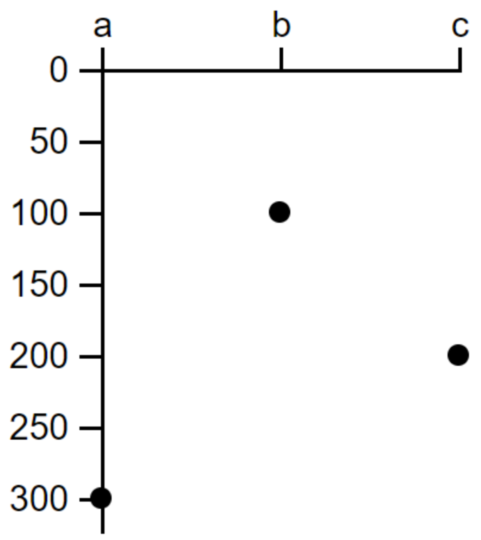
This diagram illustrates the scale function.
It is not created by D3.js.

- Others: scaleOrdinal, scaleLinear, scalePow, scaleExp, scaleLog, scaleTime, scaleBand, scalePoint, scaleIdentity

Axes

```
const data = [
    {"key": "a", "value": 300},
    {"key": "b", "value": 100},
    {"key": "c", "value": 200}
];

const xScale = d3.scaleOrdinal()
    .domain(["a", "b", "c"])
    .range([0, 50, 100]);

const yScale = d3.scaleLinear()
    .domain([0, 500])
    .range([0, 200]);

const svgG = d3.select("body")
    .append("svg")
    .append("g")
    .attr("transform", "translate(30, 20)");

svgG.selectAll("circle")
    .data(data)
    .enter()
    .append("circle")
    .attr("cx", d => xScale(d.key))
    .attr("cy", d => yScale(d.value))
    .attr("r", 3);
```

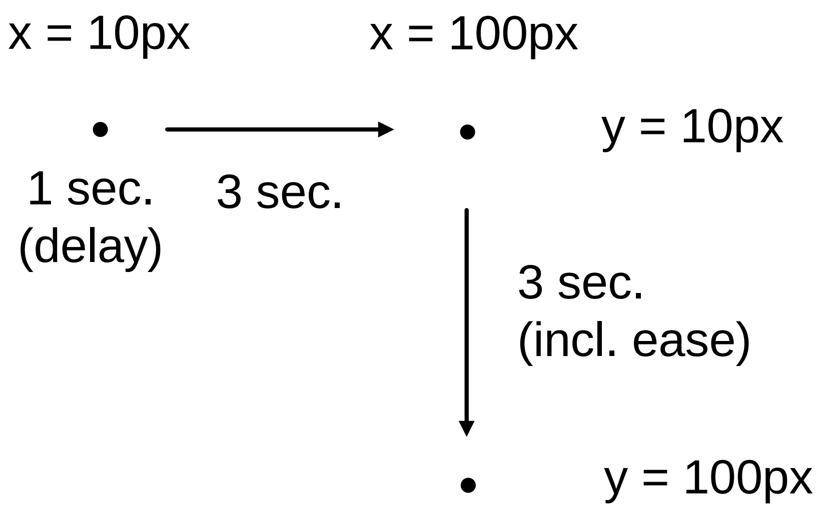
```
const xAxis = d3.axisTop(xScale);
const yAxis = d3.axisLeft(yScale);

svgG.append("g")
    .call(xAxis);

svgG.append("g")
    .call(yAxis);
```



Animation

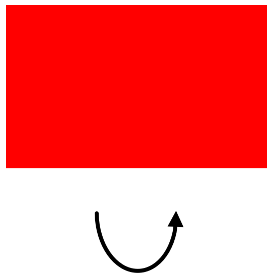
```
const svg = d3.select("body")
    .append("svg");

const circle = svg.append("circle")
    .attr("cx", 10)
    .attr("cy", 10)
    .attr("r", 3);

const xTransition = circle
    .transition()
    .delay(1000)
    .duration(3000)
    .attr("cx", 100);

xTransition
    .transition()
    .ease(d3.easeElastic)
    .duration(3000)
    .attr("cy", 100);
```

chained
transition



Interactivity

```
let backgroundColor = "";

const svg = d3.select("body")
    .append("svg");

svg.on("click", () => {
    if (backgroundColor === "green")
        backgroundColor = "red";
    else
        backgroundColor = "green";

    svg.style("background-color", backgroundColor)
});

svg.dispatch("click");
```



Generator functions

- Symbols
- Lines, curves
- Arcs, pies
- Areas
- Stacks
- Chords
- Hierarchy
- Voronoi
- Geography

```
const pieGenerator = d3.pie();
const pieData = pieGenerator([1, 2, 3, 4]);

pieData →

[{
    "data": 1,
    "index": 3,
    "value": 1,
    "startAngle": 5.654866776461628,
    "endAngle": 6.283185307179586,
    "padAngle": 0
}, ... {
    "data": 4,
    "index": 0,
    "value": 4,
    "startAngle": 0,
    "endAngle": 2.5132741228718345,
    "padAngle": 0
}]
```

Helper functions

- Arrays
- Brushes
- Numbers
- Dates and times
- Colors
- Map projections
- Zooming and panning, dragging
- Forces

```
d3.extent([3, -1, 2, 4]);
→ [-1, 4]

const numberFormatter = d3.format(",");
numberFormatter(1000000);
→ 1,000,000

const color = d3.rgb("limegreen");
→ {"r": 50, "g": 205, "b": 50, "opacity": 1}
color.hex()
→ "#32cd32"
```

Learning resources

Homepage: https://d3js.org/

API: https://github.com/d3/d3/blob/master/API.md

Overview: https://observablehq.com/@d3/learn-d3
Examples: https://observablehq.com/@d3/gallery

Book: Philipp K. Janert (2019) - D3 for the Impatient
https://learning.oreilly.com/library/view/d3-for-the/9781492046783/