# `tt()`

## Schedule

1. Review (boilerplate, framework)
2. SVG anatomy
3. D3 Introduction
4. D3 Concepts
5. All together!

## Schedule

1. **Review (boilerplate, framework)**
2. SVG anatomy
3. D3 Introduction
4. D3 Concepts
5. All together!

# Libraries, frameworks & bundlers

***Library:***
*Om visualisaties te maken.*

***Framework:***
*Om in componenten te werken.*

***Bundler:***
*Om alles samen te voegen.*

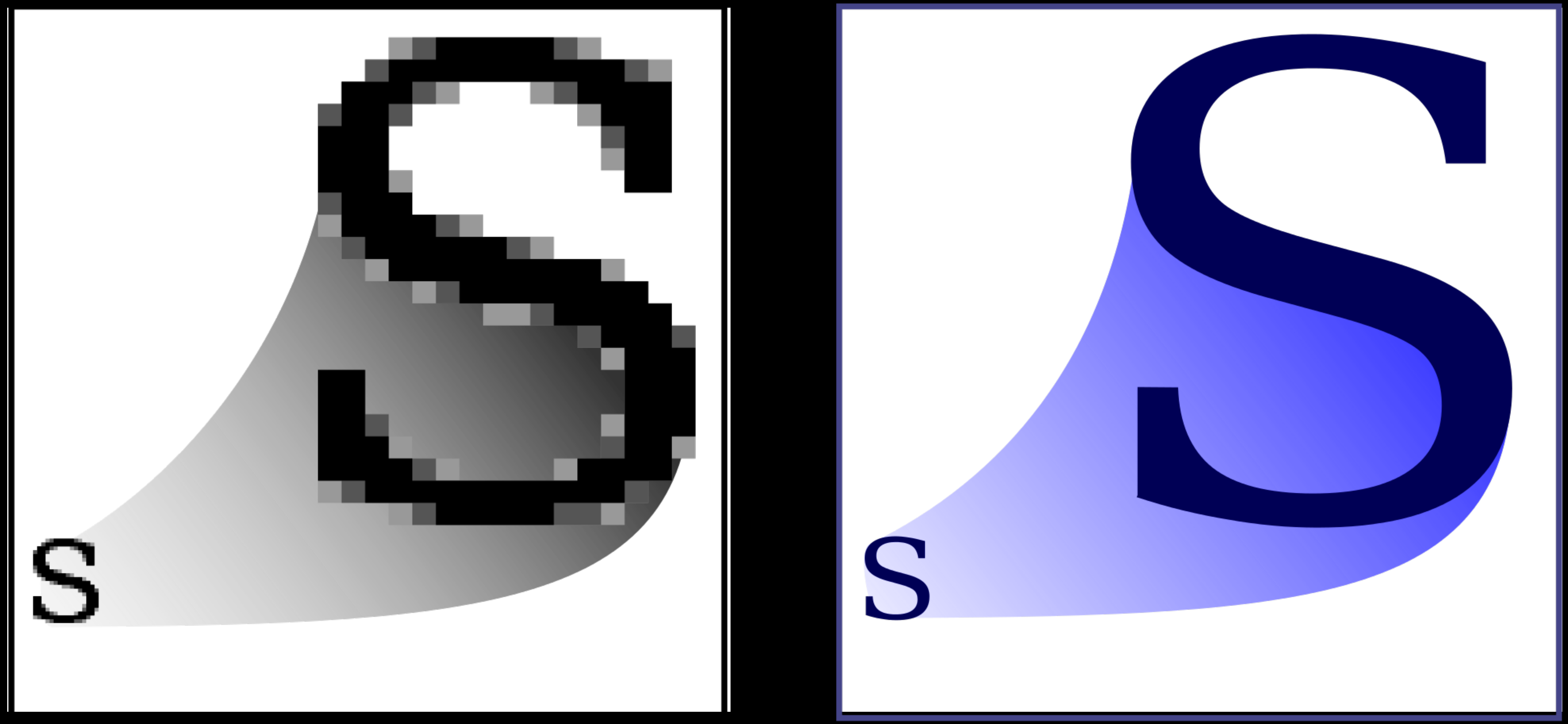
# Schedule

1. Review (boilerplate, framework)
2. **SVG anatomy**
3. D3 Introduction
4. D3 Concepts
5. All together!

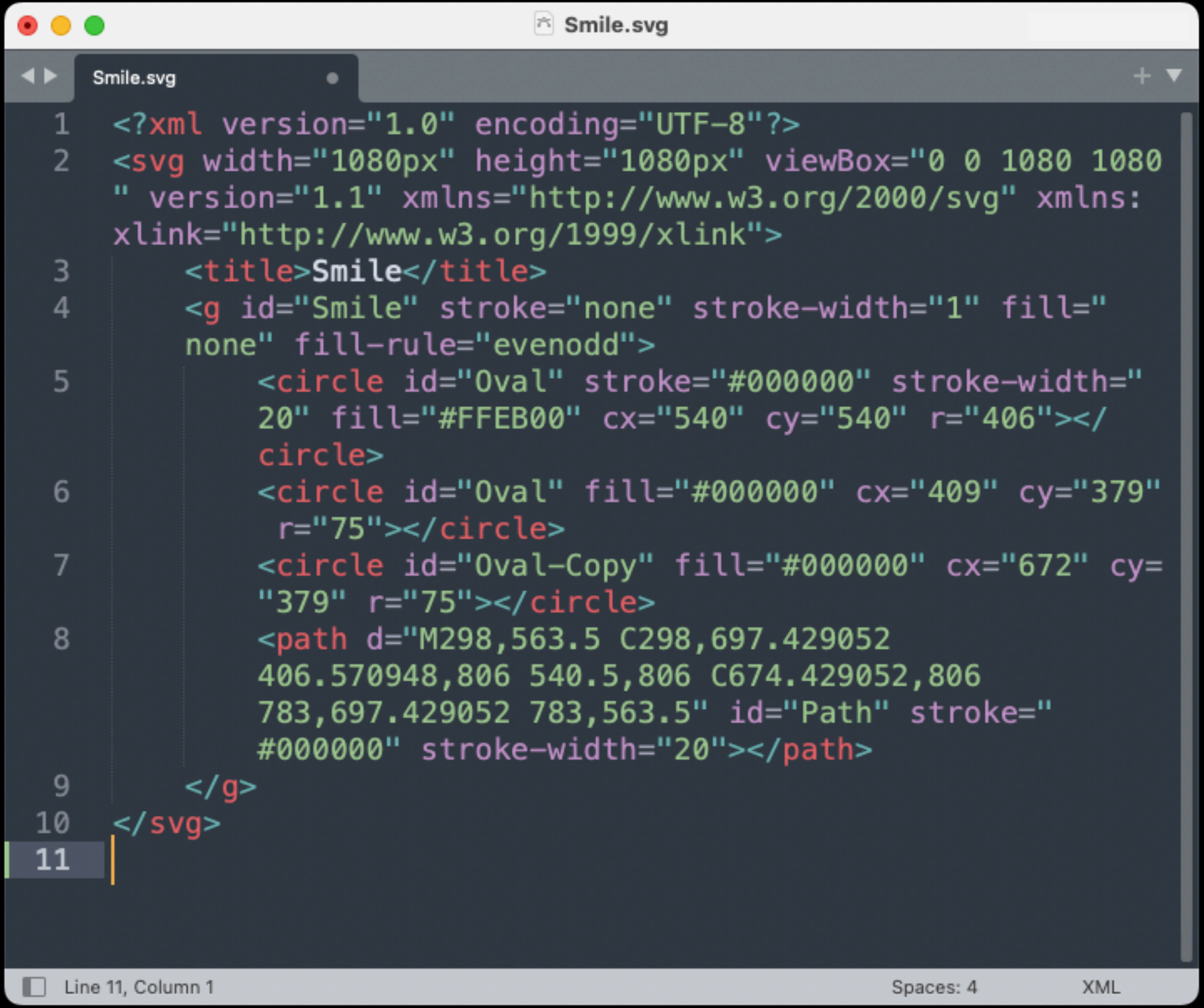
## SVG

**S**calable **V**ector **G**raphics

## SVG



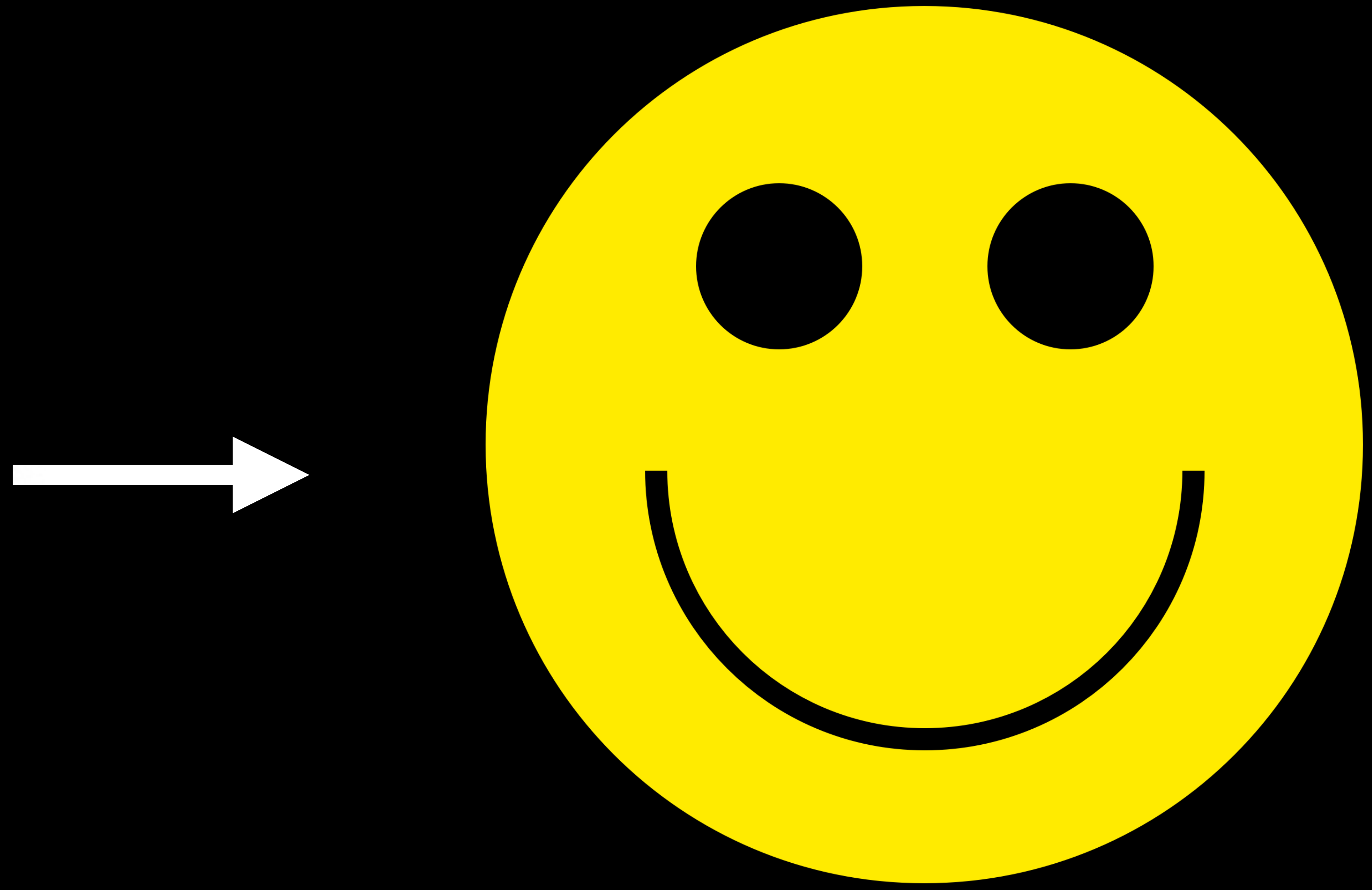
# SVG



Smile.svg

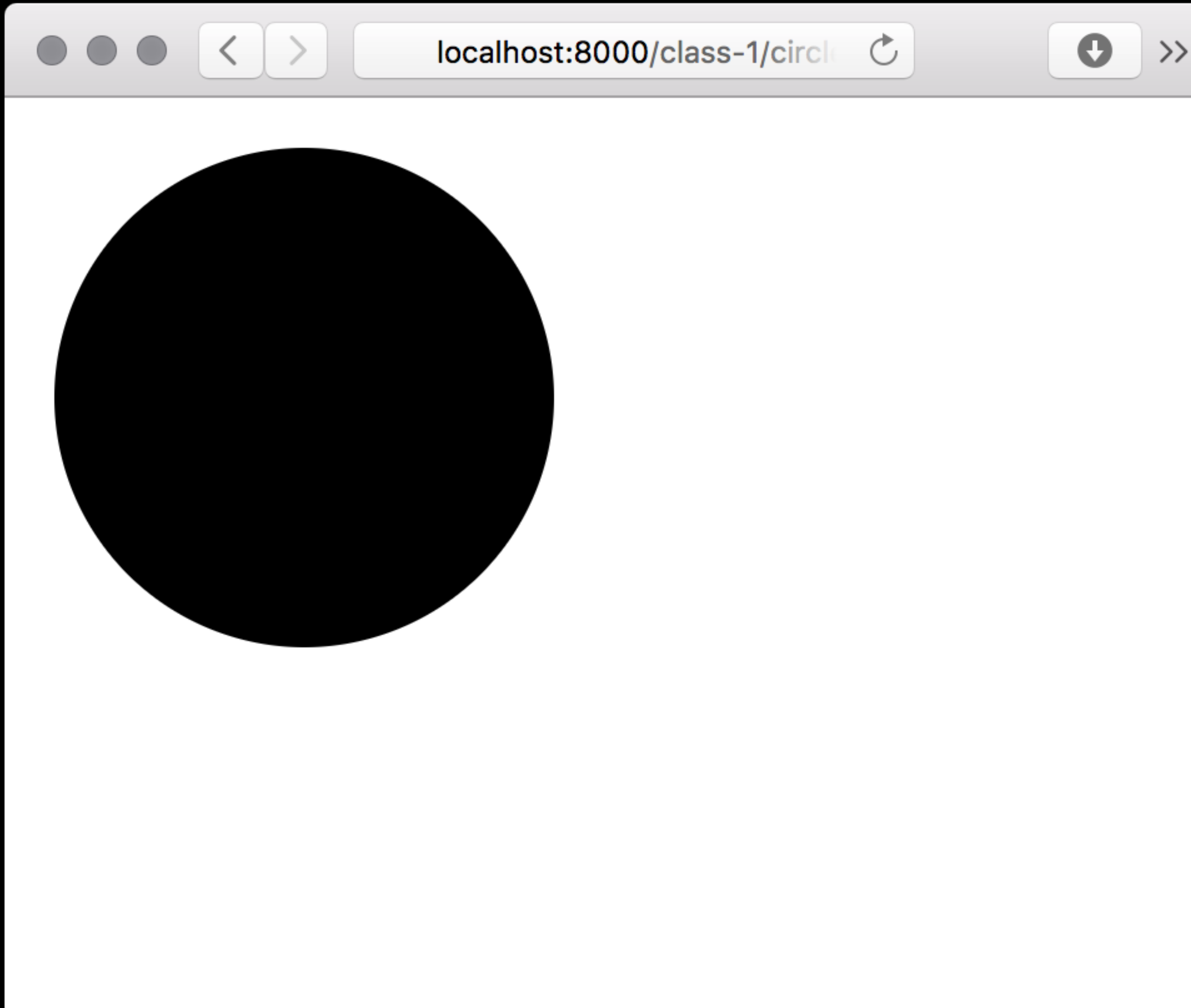
```xml
<?xml version="1.0" encoding="UTF-8"?>
<svg width="1080px" height="1080px" viewBox="0 0 1080 1080" version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
    <title>Smile</title>
    <g id="Smile" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd">
        <circle id="Oval" stroke="#000000" stroke-width="20" fill="#FFEB00" cx="540" cy="540" r="406"></circle>
        <circle id="Oval" fill="#000000" cx="409" cy="379" r="75"></circle>
        <circle id="Oval-Copy" fill="#000000" cx="672" cy="379" r="75"></circle>
        <path d="M298,563.5 C298,697.429052 406.570948,806 540.5,806 C674.429052,806 783,697.429052 783,563.5" id="Path" stroke="#000000" stroke-width="20"></path>
    </g>
</svg>

```

Line 11, Column 1    Spaces: 4    XML



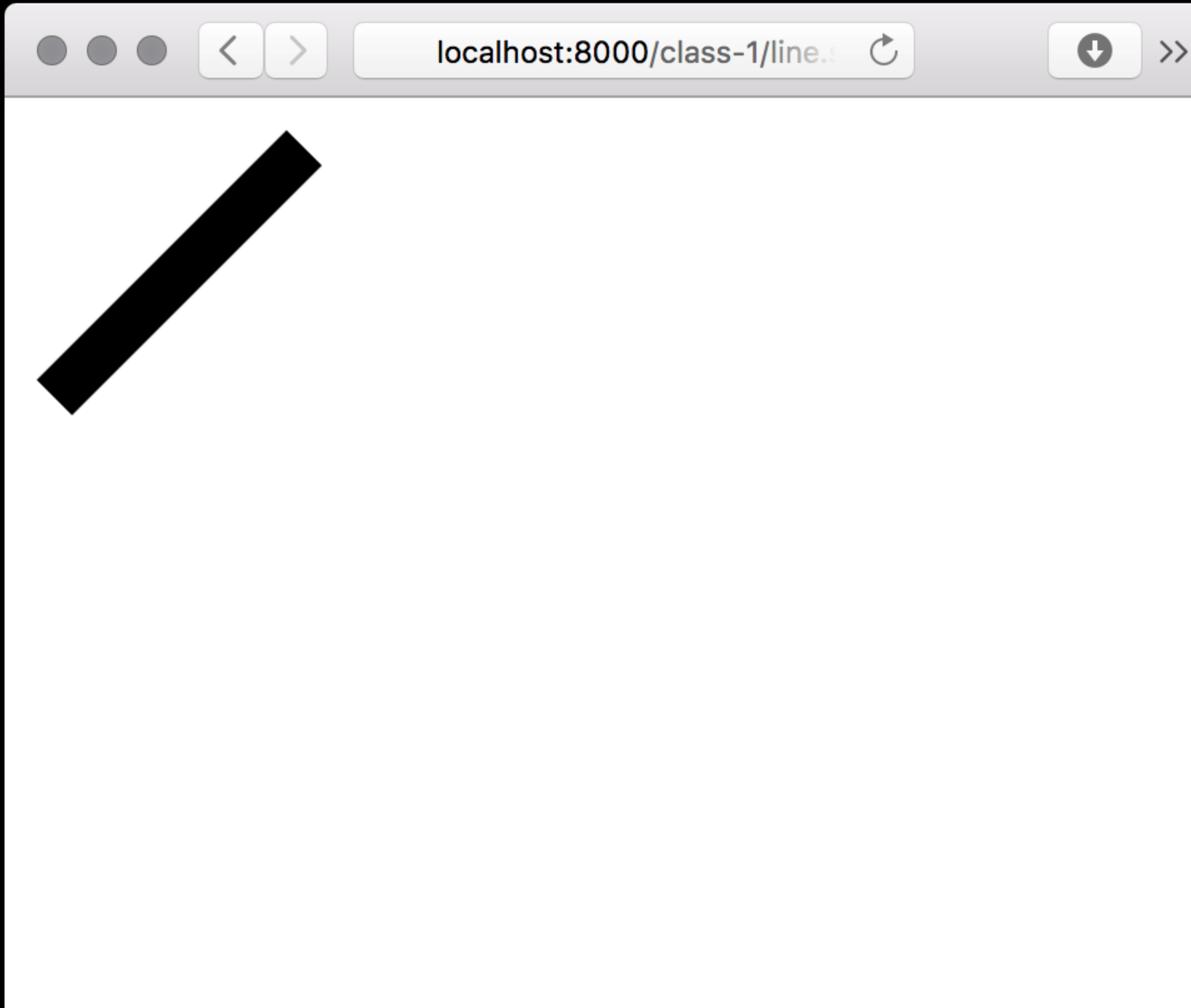
## SVG Elements

```
<circle
  cx="120"
  cy="120"
  r="100"
/>
```



# SVG Elements

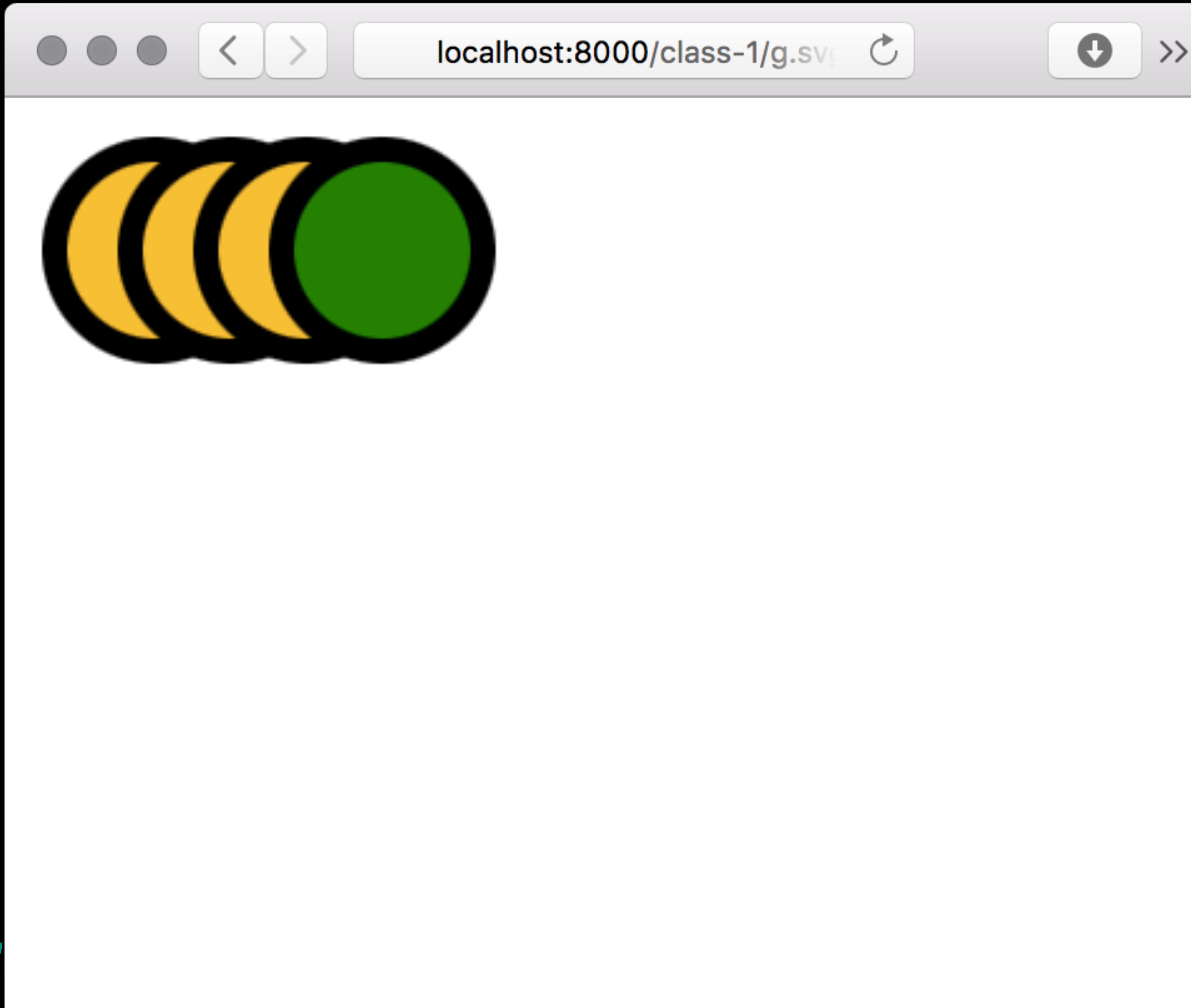
```
<line
  x1="20"
  y1="120"
  x2="120"
  y2="20"
  stroke-width="20"
  stroke="black"
/>
```



# SVG Elements

https://codepen.io/vijnv/pen/mdKdNwJ

```
<style>
  circle {
    fill: #f7bf33;
    stroke: black;
    stroke-width: 10;
  }
  circle.highlight {
    fill: green;
  }
</style>
<circle cx="60" cy="60" r="40" />
<circle cx="90" cy="60" r="40" />
<circle cx="120" cy="60" r="40" />
<circle class="highlight" cx="150" cy="60'
```



## Schedule

1. Review (boilerplate, framework)
2. SVG anatomy
3. **D3 Introduction**
4. D3 Concepts
5. All together!

## D3

**D**ata **D**riven **D**ocuments

## Library (D3)

A **JavaScript library** is a collection of pre-written JavaScript code that provides specific, reusable functions and utilities to help developers accomplish common tasks more easily. Instead of writing complex code from scratch, you can leverage a library to perform repeated tasks.

## Library (D3)

**D3** (or **D3.js**) is a free, open-source JavaScript library for visualizing data. Its low-level approach built on web standards offers unparalleled flexibility in authoring dynamic, data-driven graphics.

## Library (D3)

**D3** is not a charting library in the traditional sense. **It has no concept of “charts”.**

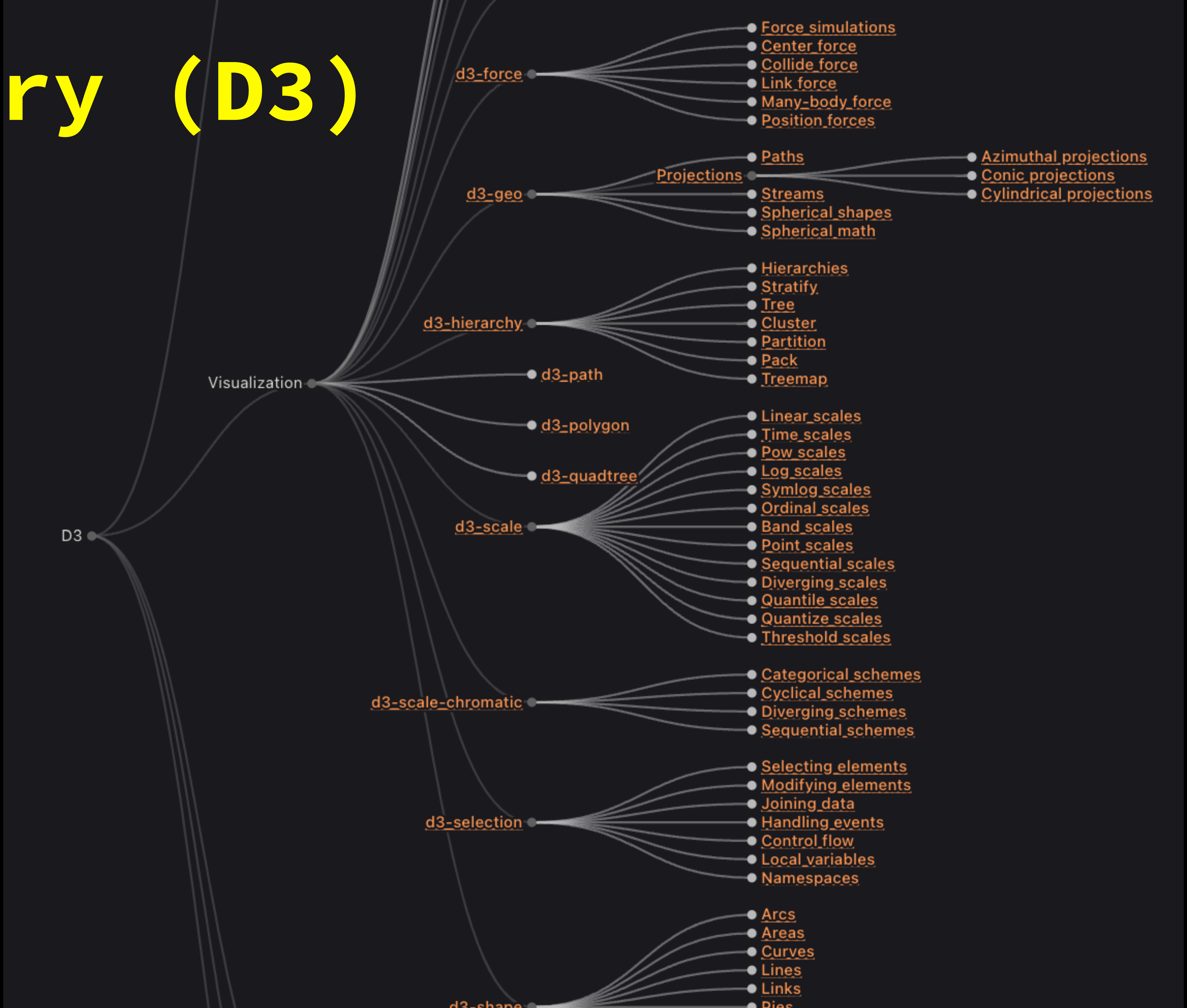
# D3 vs Instant Mix

- RawGraphs.io
- LocalFocus.nl
- Flourish.studio

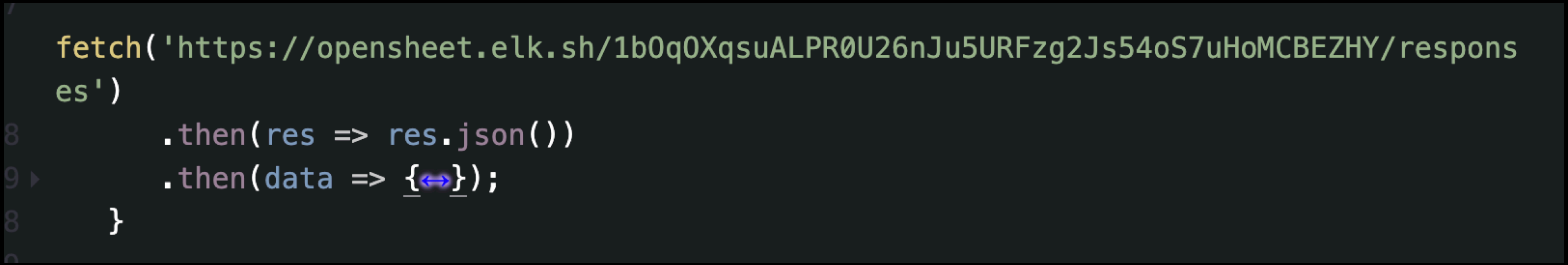
👈 both use D3

Just add water!

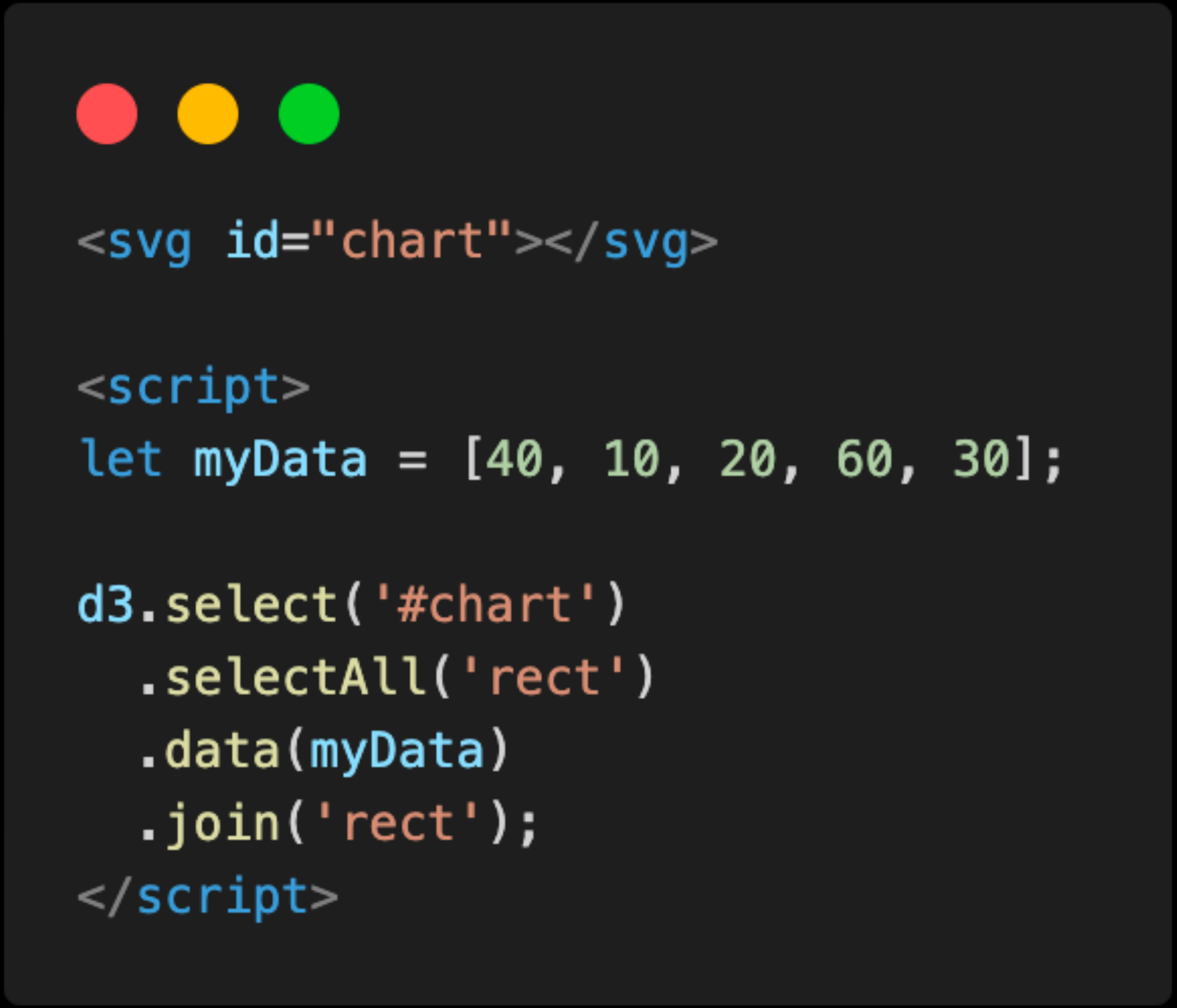
# Library (D3)


D3
Visualization
d3-force
Force simulations
Center force
Collide force
Link force
Many-body force
Position forces
d3-geo
Paths
Projections
Azimuthal projections
Conic projections
Cylindrical projections
Streams
Spherical shapes
Spherical math
d3-hierarchy
Hierarchies
Stratify
Tree
Cluster
Partition
Pack
Treemap
d3-path
d3-polygon
d3-quadtree
d3-scale
Linear scales
Time scales
Pow scales
Log scales
Symlog scales
Ordinal scales
Band scales
Point scales
Sequential scales
Diverging scales
Quantile scales
Quantize scales
Threshold scales
d3-scale-chromatic
Categorical schemes
Cyclical schemes
Diverging schemes
Sequential schemes
d3-selection
Selecting elements
Modifying elements
Joining data
Handling events
Control flow
Local variables
Namespaces
d3-shape
Arcs
Areas
Curves
Lines
Links
Pies

# Method chaining



```
fetch('https://opensheet.elk.sh/1bOqOXqsuALPR0U26nJu5URFzg2Js54oS7uHoMCBEZHY/respons
es')
        .then(res => res.json())
        .then(data => {↔});
    }
```



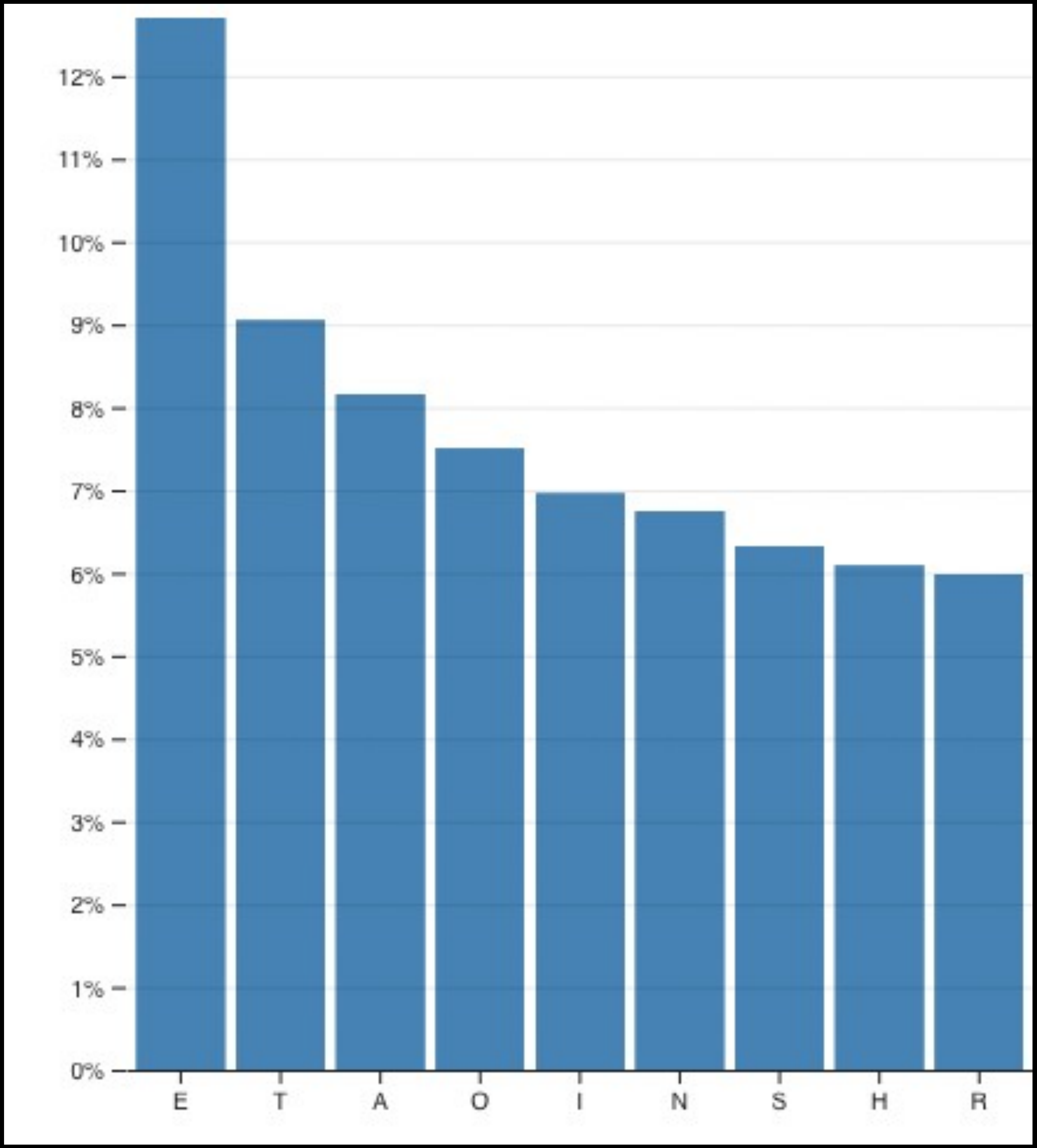
```
<svg id="chart"></svg>

<script>
let myData = [40, 10, 20, 60, 30];

d3.select('#chart')
  .selectAll('rect')
  .data(myData)
  .join('rect');
</script>
```

# Library (D3)

- Flexibel: er zijn geen ‘chart’ types dus veel customizability
- Standaarden: wat er al op het web is DOM, SVG
- Dynamisch: goed voor interactieve viz want data joining

## Schedule

1. Review (boilerplate, framework)
2. SVG anatomy
3. D3 Introduction
4. **D3 Concepts**
5. All together!



# D3 Concepts

1. Selections
2. Joins
3. Scales
4. Axes
5. Ticks
6. Accessor functions

## Selections

d3.select() is kinda like document.querySelector()

d3.selectAll is kinda like document.querySelectorAll()

```
d3.selectAll('circle')
```

```
d3.selectAll('circle').style('fill', 'red')
```

## Selections

`d3.select() is kinda like document.querySelector()`

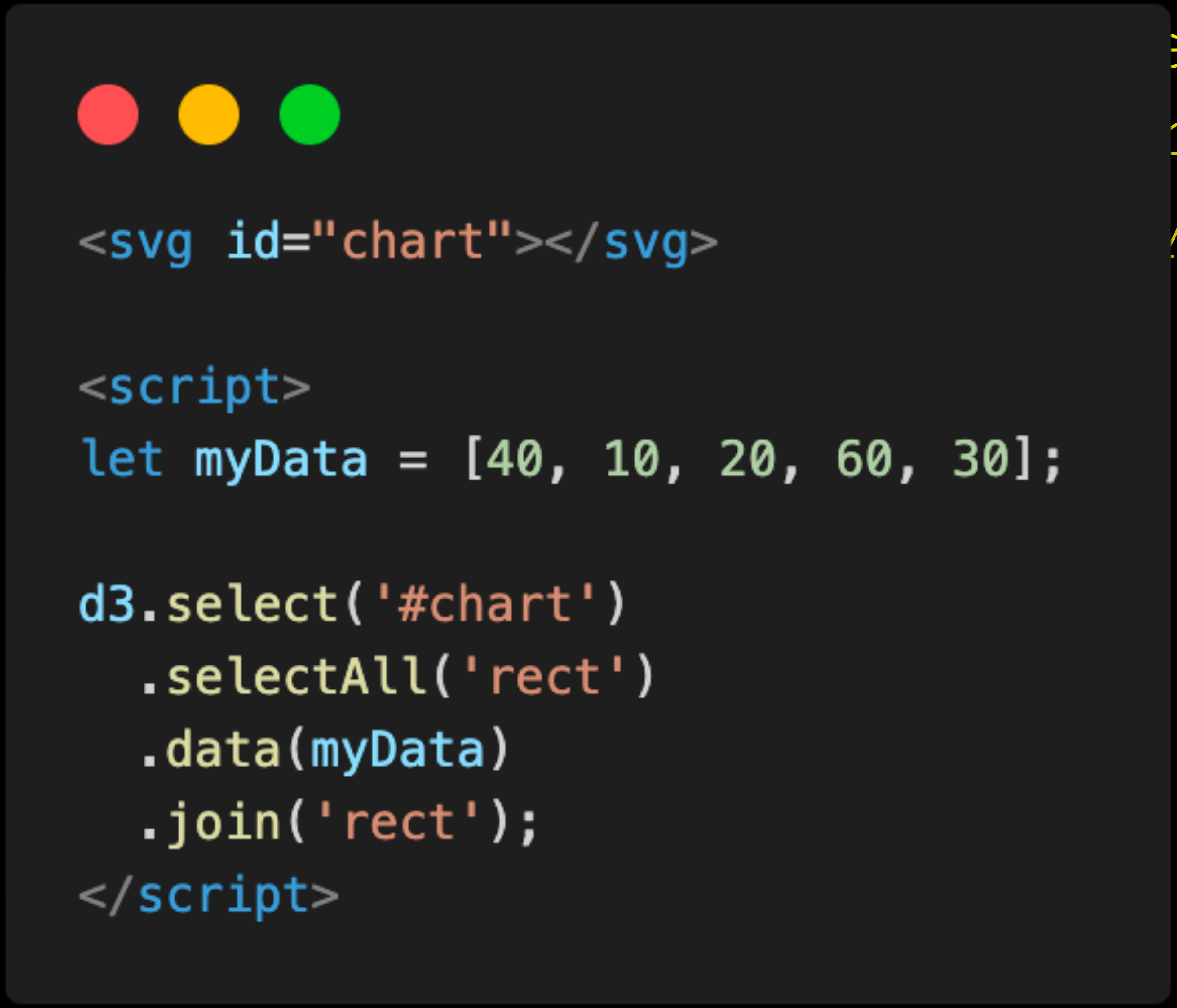
`d3.selectAll is kinda like document.querySelectorAll()`

| Name | Behaviour | Example |
|---|---|---|
| `.style` | Update the style | `d3.selectAll('circle').style('fill', 'red')` |
| `.attr` | Update an attribute | `d3.selectAll('rect').attr('width', 10)` |
| `.classed` | Add/remove a class attribute | `d3.select('.item').classed('selected', true)` |
| `.property` | Update an element's property | `d3.selectAll(‘input[type=checkbox]’).property('checked', true)` |
| `.text` | Update the text content | `d3.select(‘h1').text('Hello world’)` |
| `.html` | Change the html content | `d3.select(‘form').html('<button>Turn off</button>’)` |

https://codepen.io/vijnv/pen/ExrNBYL

# Joins

Data joins are kinda like doing a mail merge in Office to create address labels based on a list in Excel



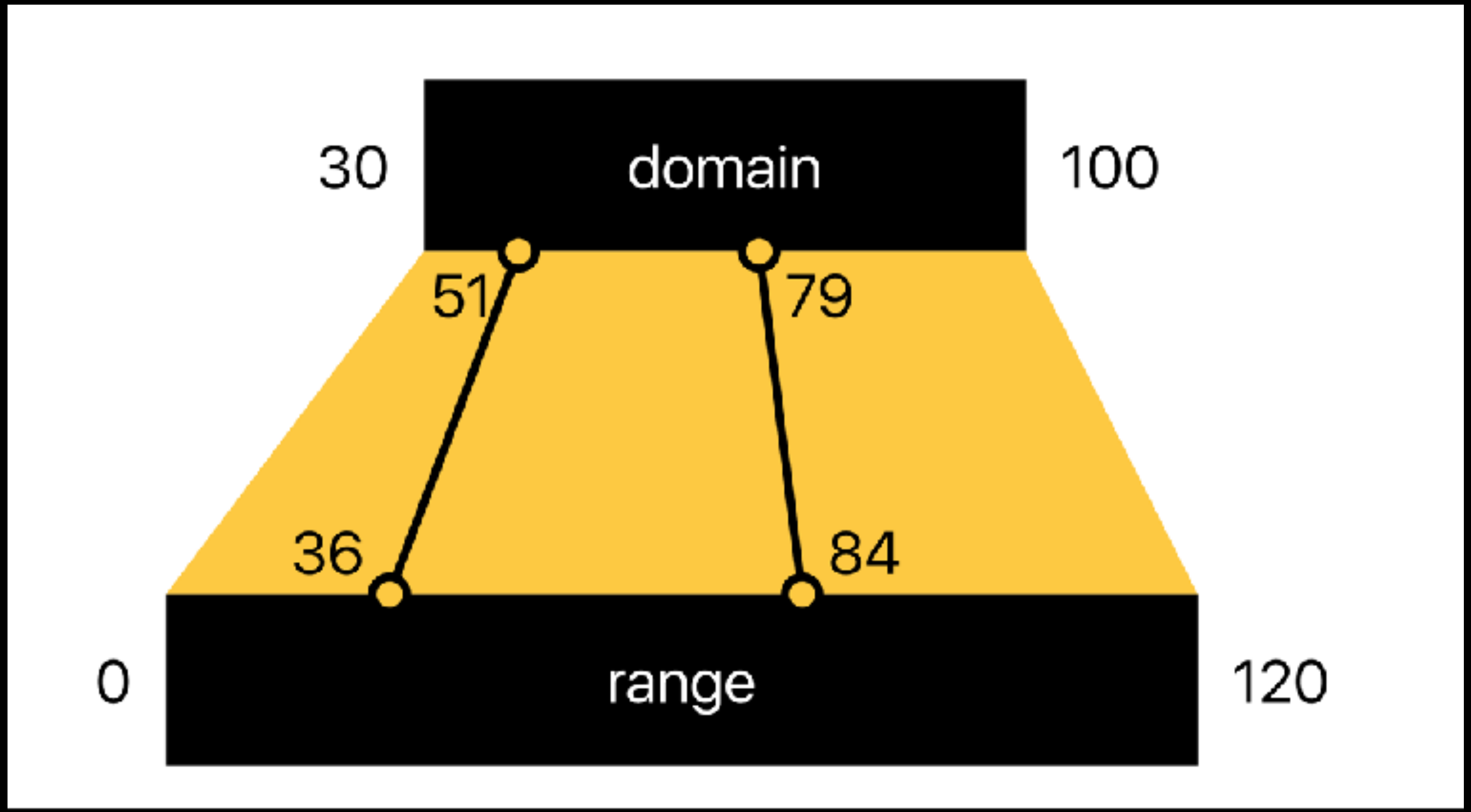
```
<svg id="chart"></svg>

<script>
let myData = [40, 10, 20, 60, 30];

d3.select('#chart')
  .selectAll('rect')
  .data(myData)
  .join('rect');
</script>
```

ere we use d3.join() to create a
ect> element for each item in our
yData array

https://codepen.io/vijnv/pen/poGNmme

# Scales

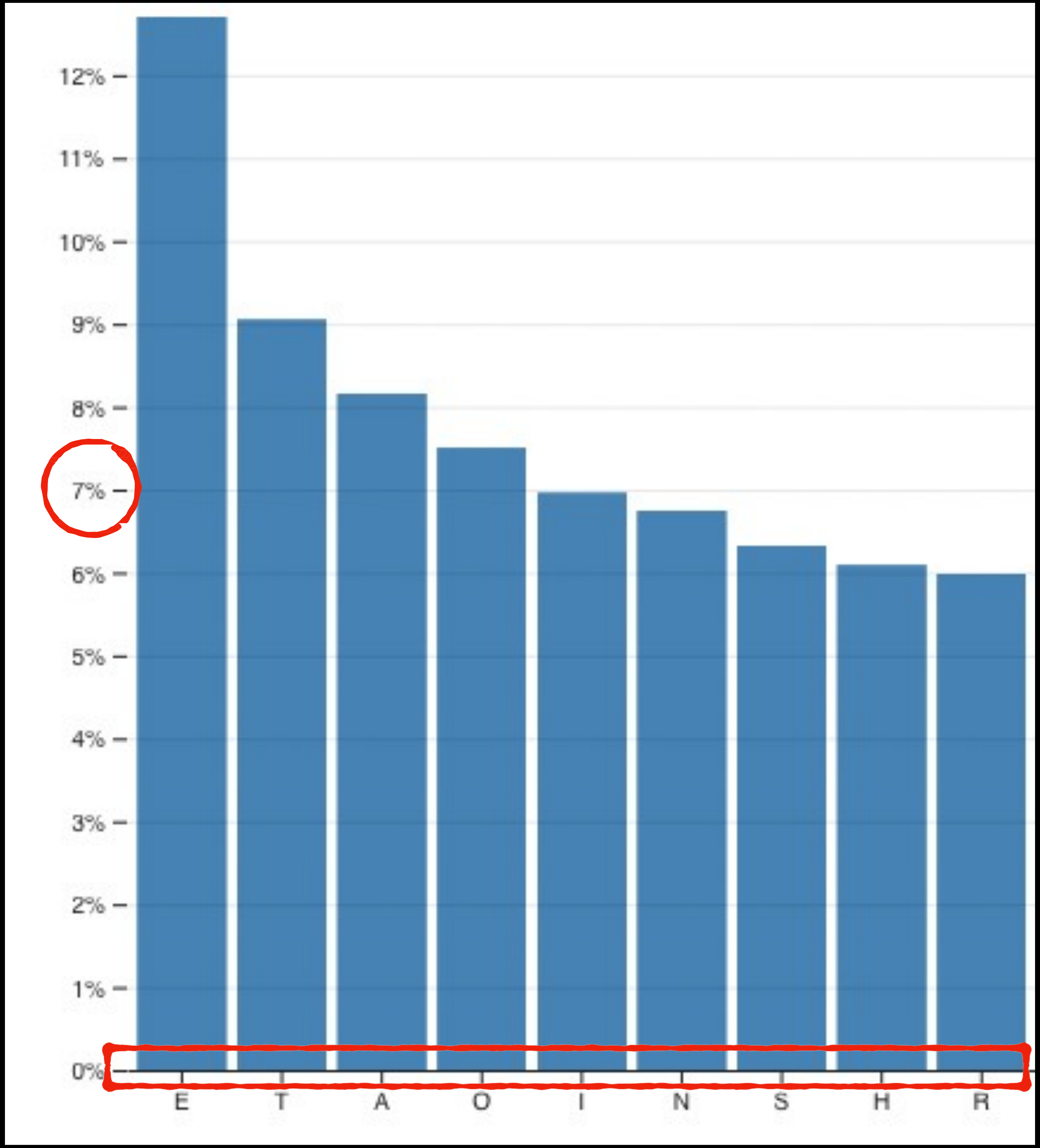
Scales help you calculate how big elements of your graph are going to be and where are they positioned. We'll cover these on Wednesday.



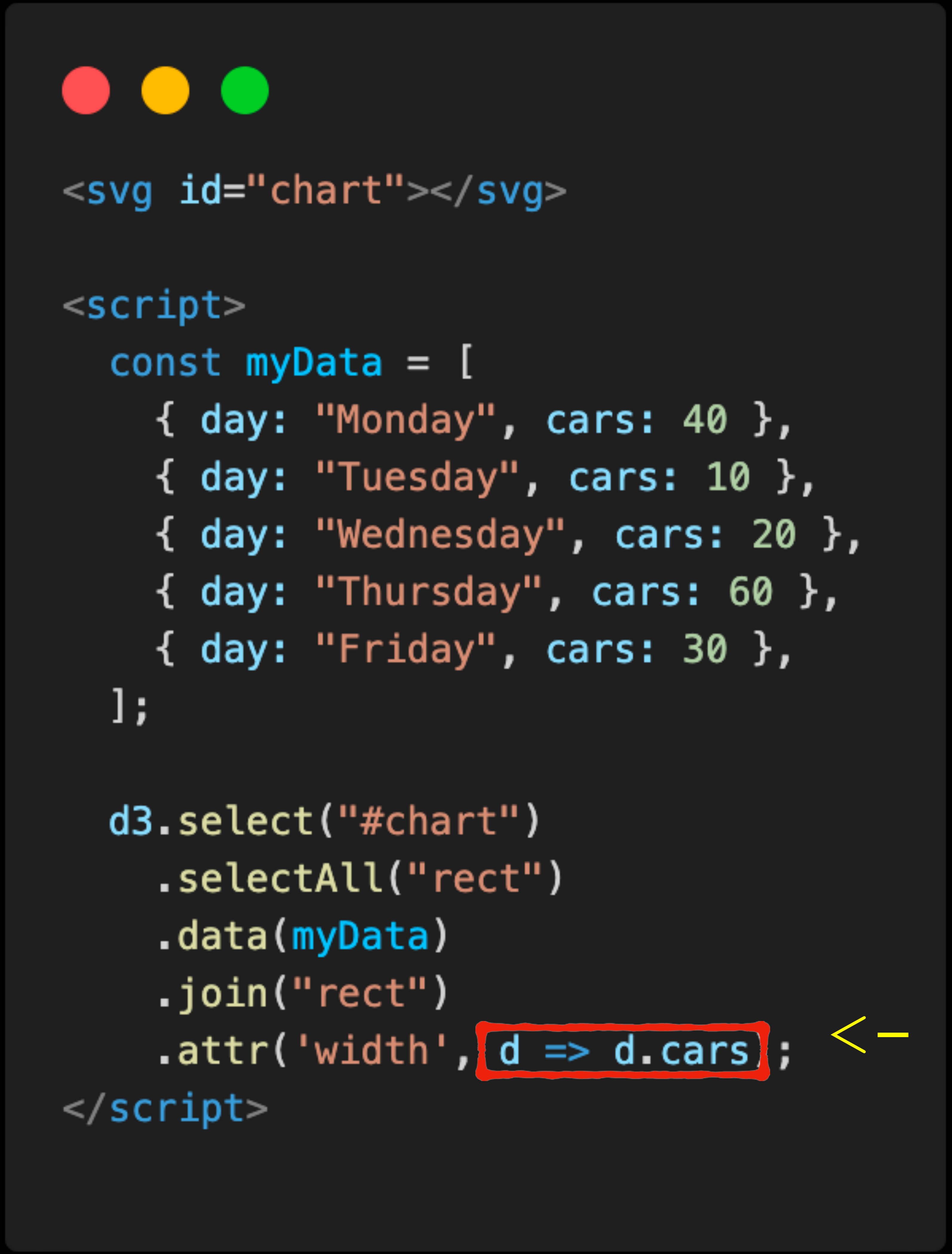
# Axes & Ticks

Ticks ->

Axes ->



# Accessor functions



```
<svg id="chart"></svg>

<script>
  const myData = [
    { day: "Monday", cars: 40 },
    { day: "Tuesday", cars: 10 },
    { day: "Wednesday", cars: 20 },
    { day: "Thursday", cars: 60 },
    { day: "Friday", cars: 30 },
  ];

  d3.select("#chart")
    .selectAll("rect")
    .data(myData)
    .join("rect")
    .attr('width', d => d.cars);
</script>
```

If you're using JSON (an array of objects) you'll need to tell D3 which property you want to use

<- Accessor function

# Schedule

1. Review (boilerplate, framework)
2. SVG anatomy
3. D3 Introduction
4. D3 Concepts
5. **All Together!**

# D3 inspiratie opdoen

- Kijk door de D3 examples.
- Welke charting types passen bij je concept?
- Welke filters, interactiviteit ga je toevoegen?

Lees:

1. https://observablehq.com/@d3/lets-make-a-bar-chart
2. https://www.d3indepth.com/selections/

```
Uncaught SyntaxError
Unexpected end of input
```