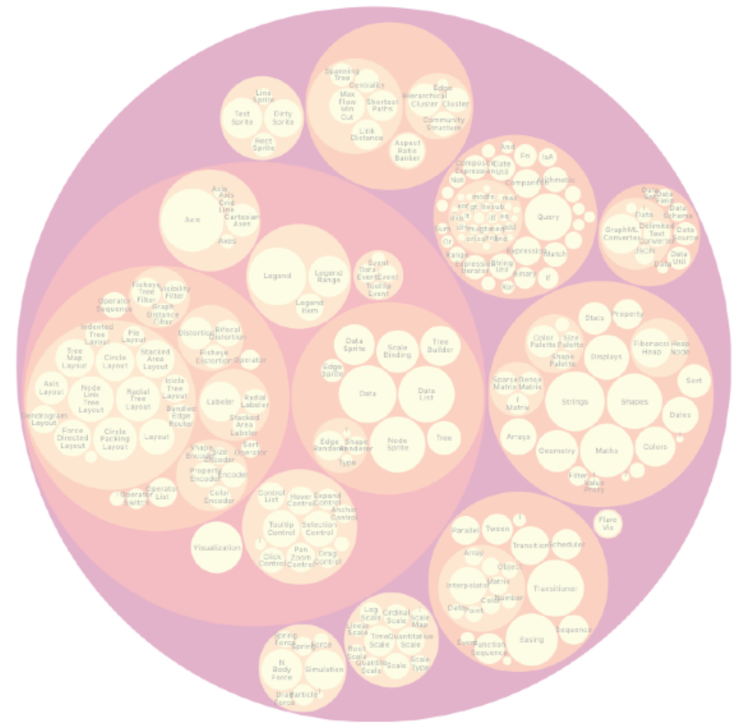
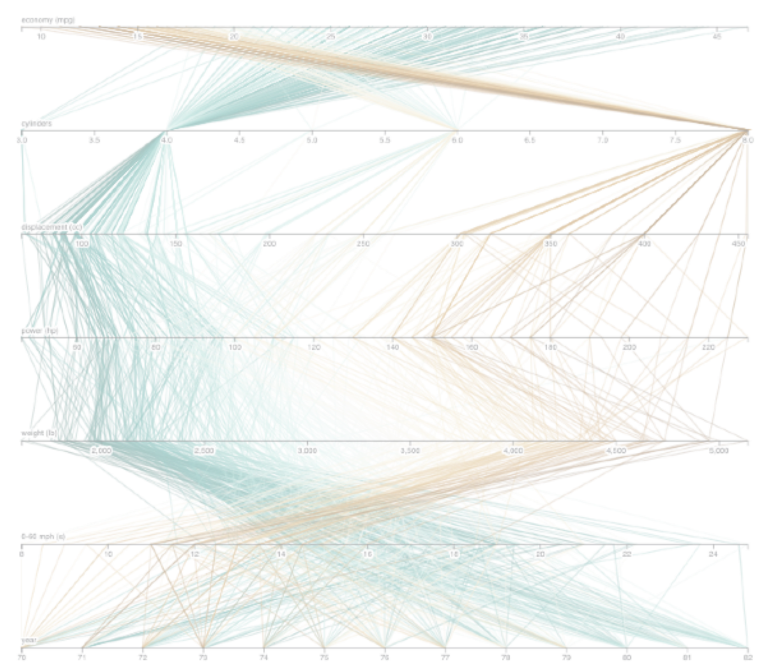
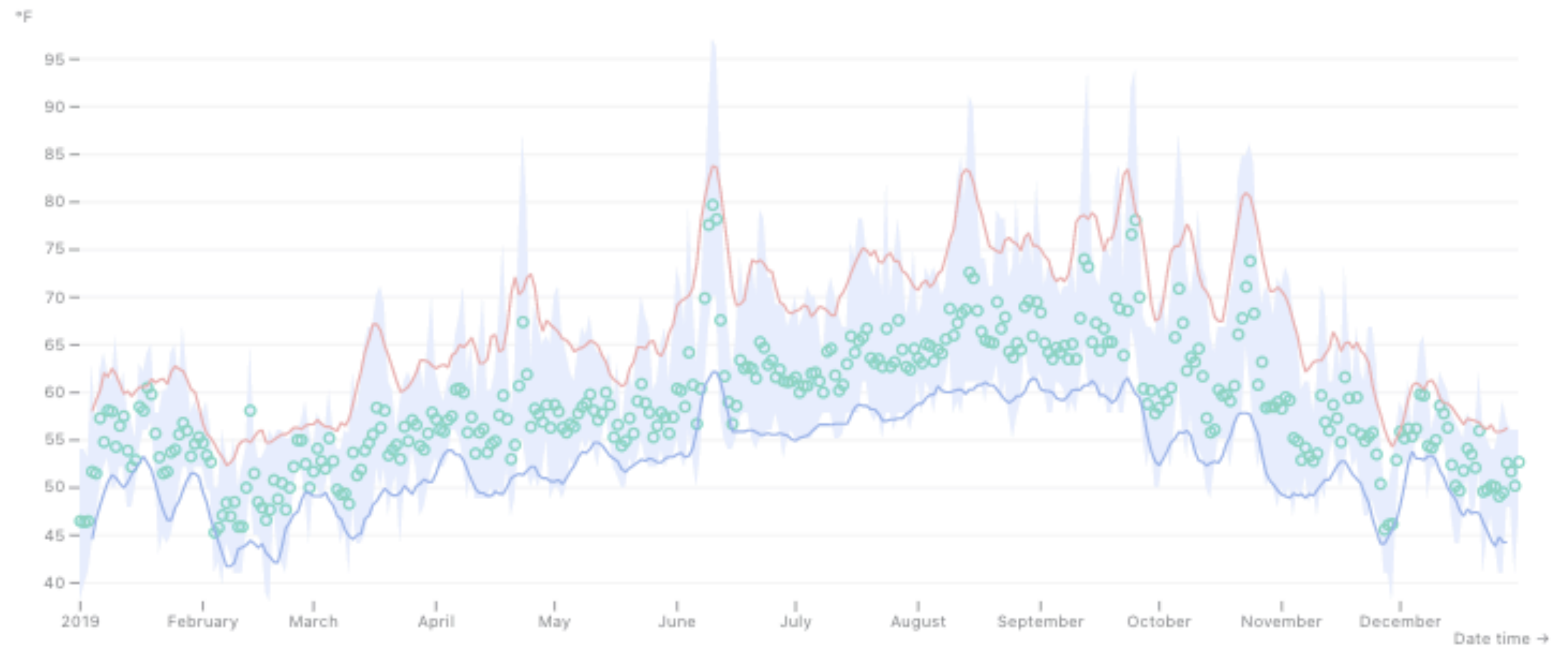
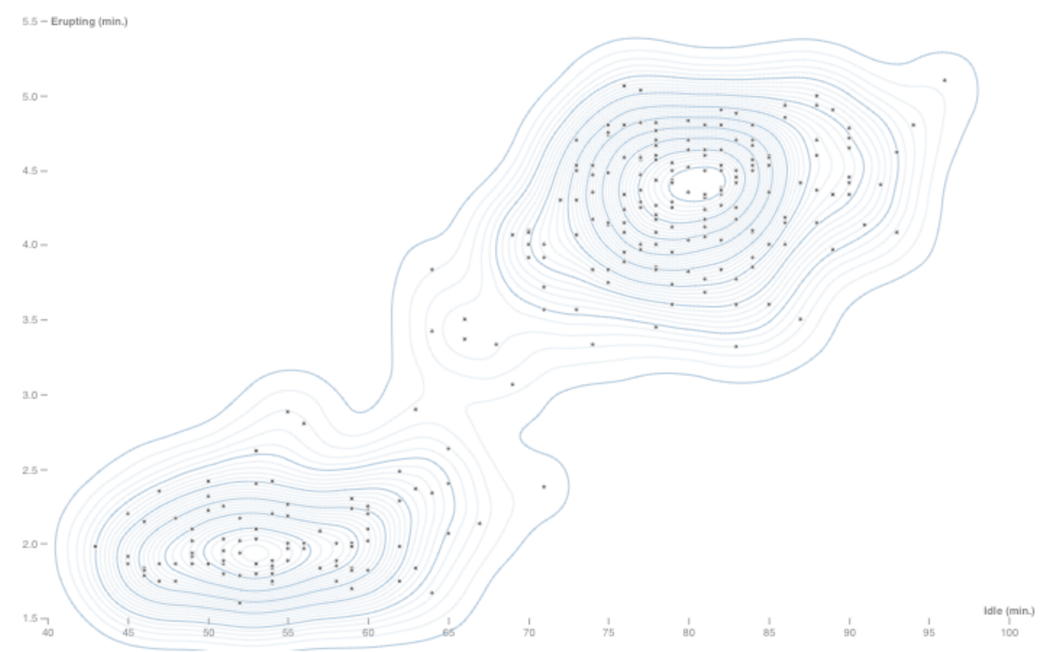
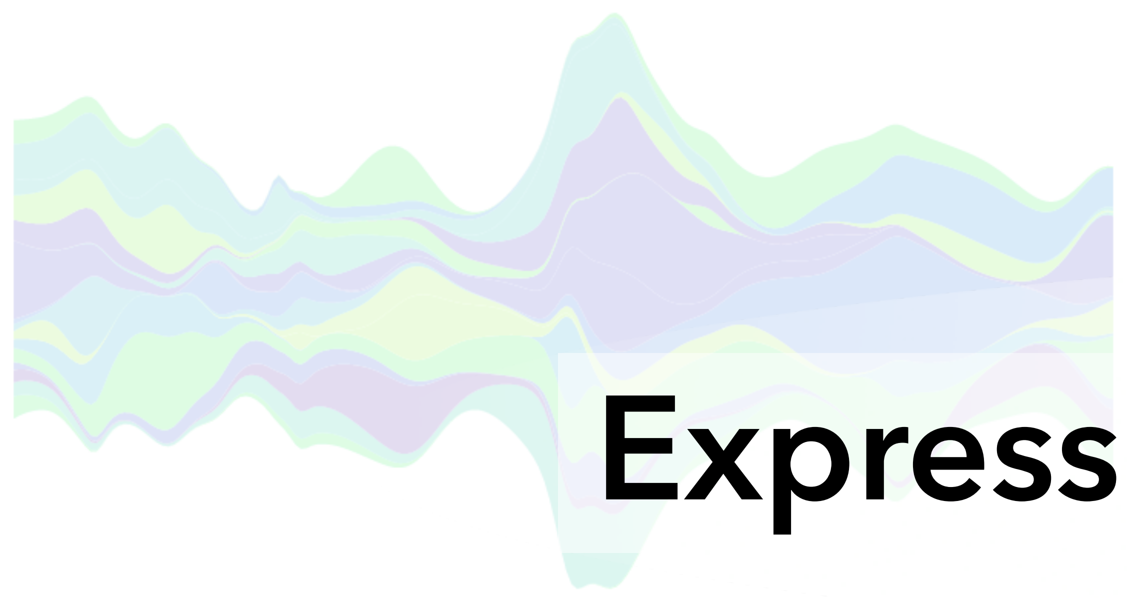
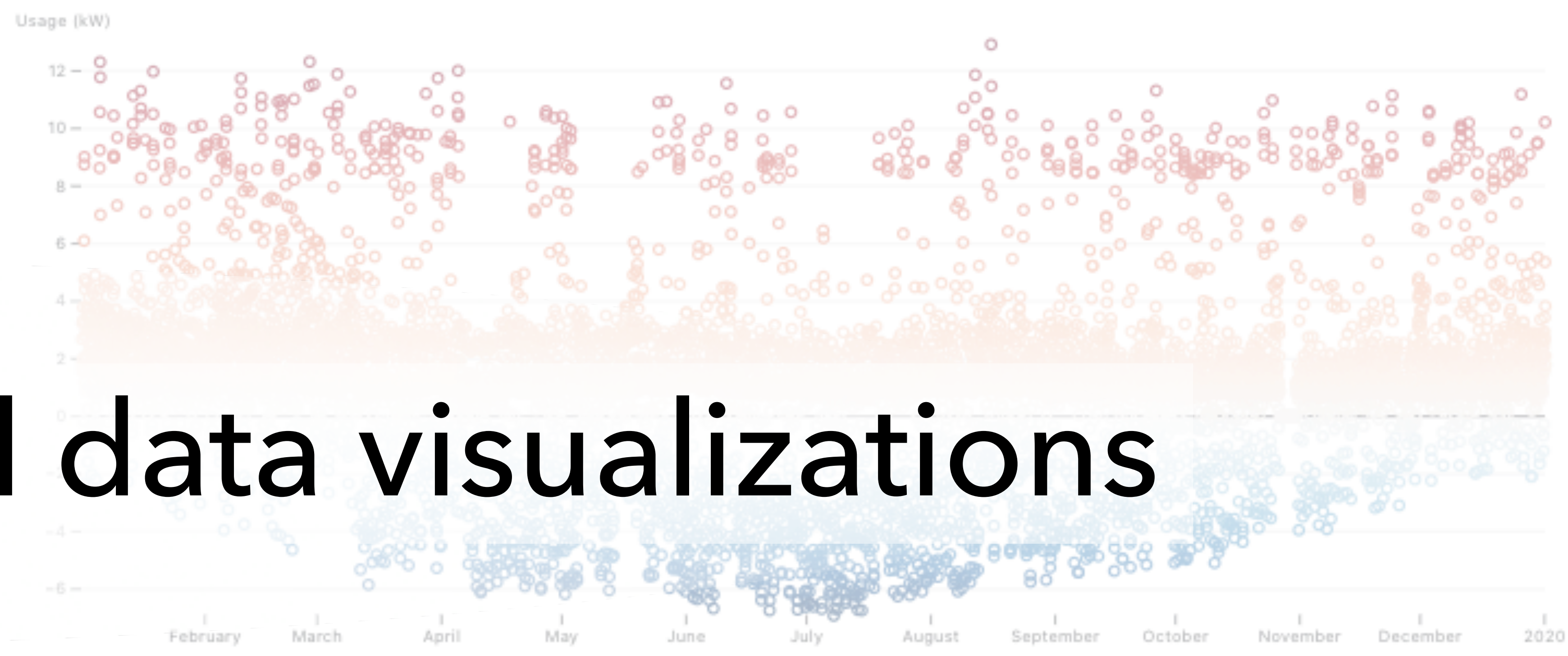
A Tour of D3

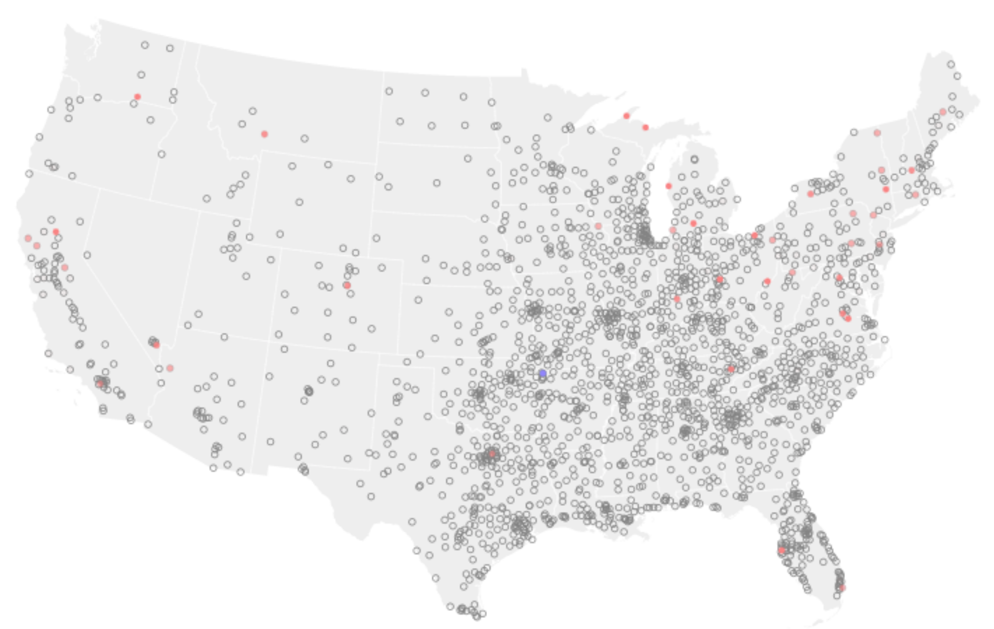
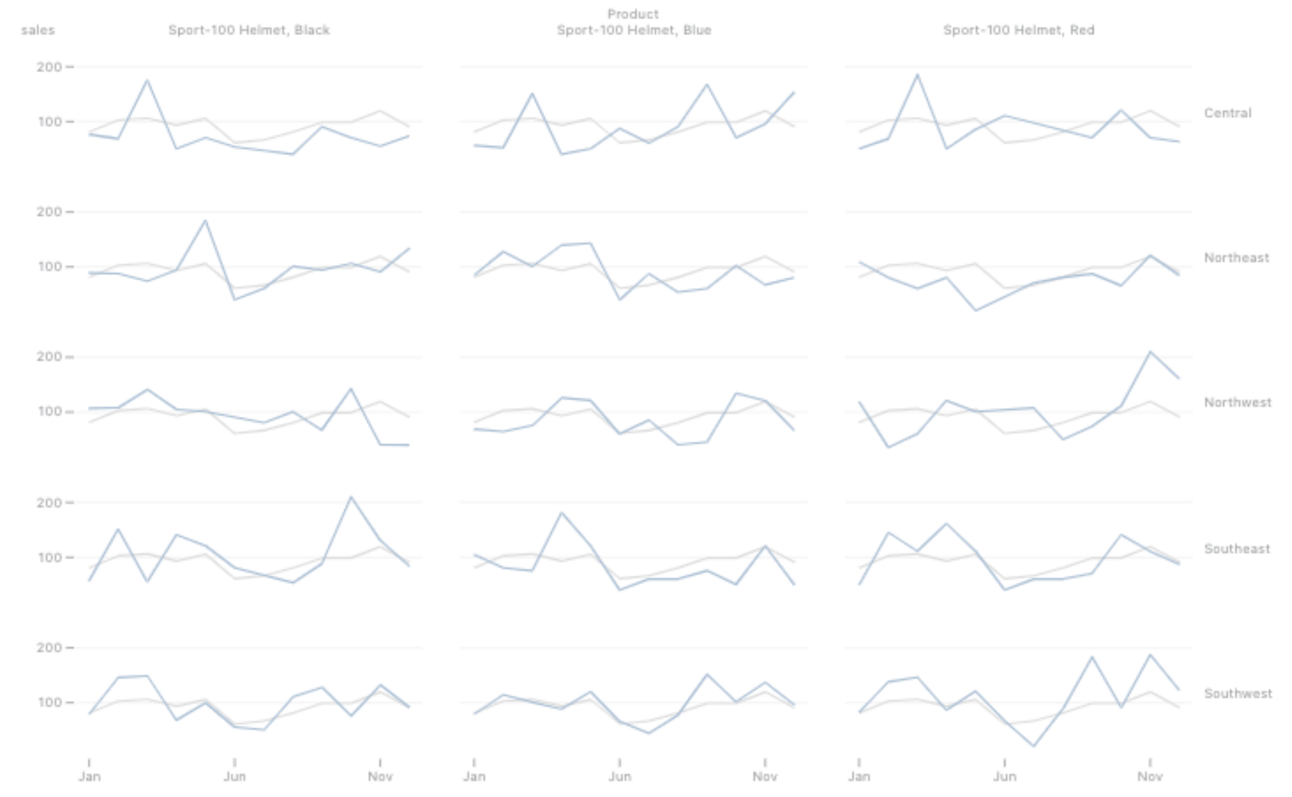
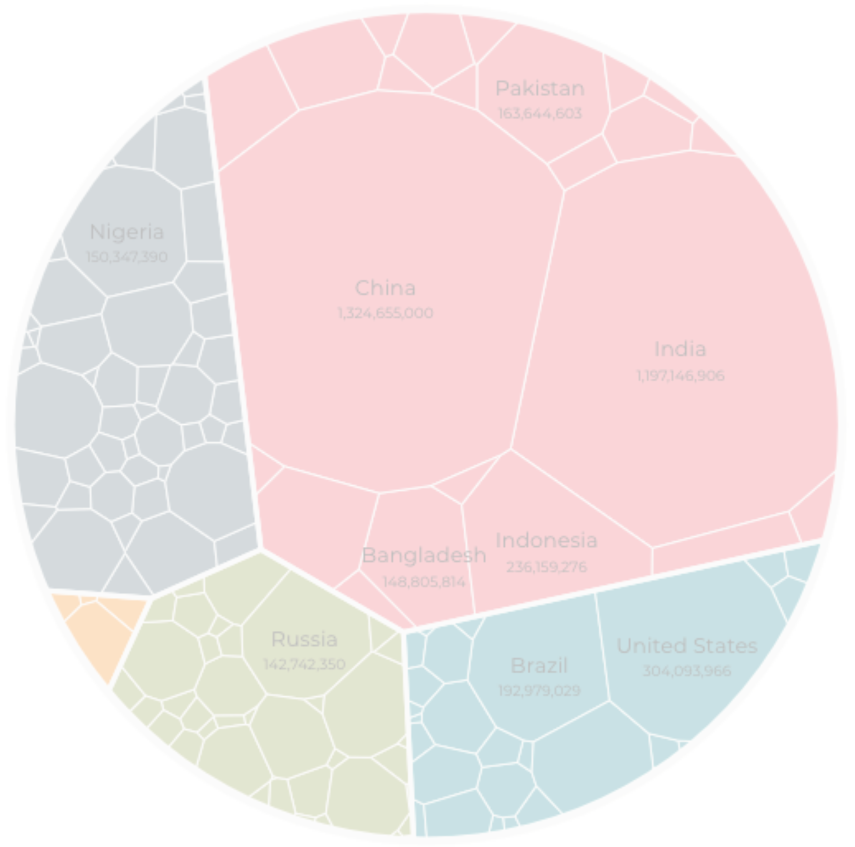
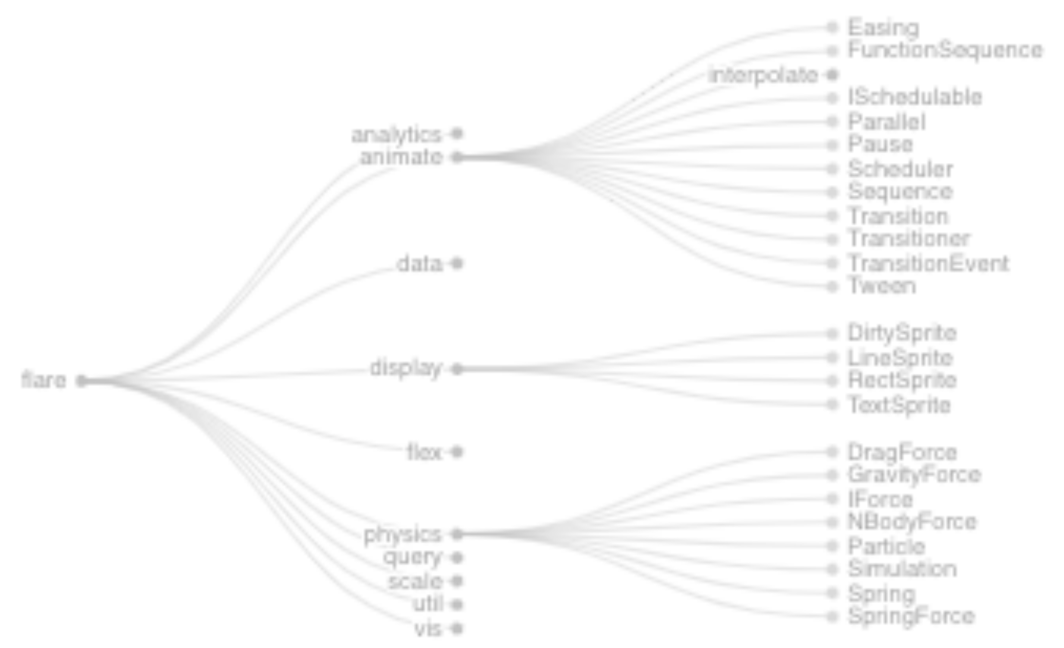
Michael Oppermann | Bio+Med+Vis 2021

michaeloppermann.com/d3



Expressive, interactive, web-based data visualizations

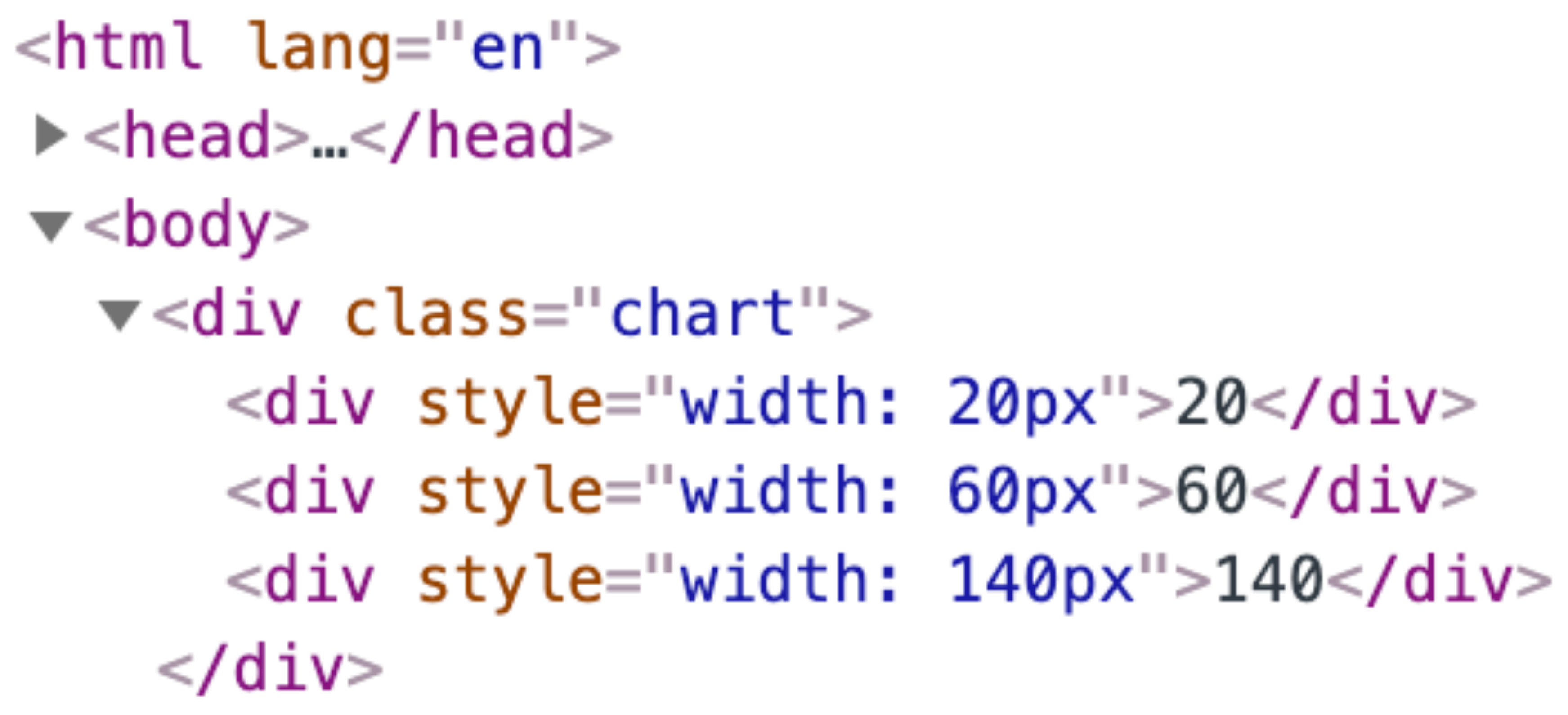





What is D3?

data driven documents

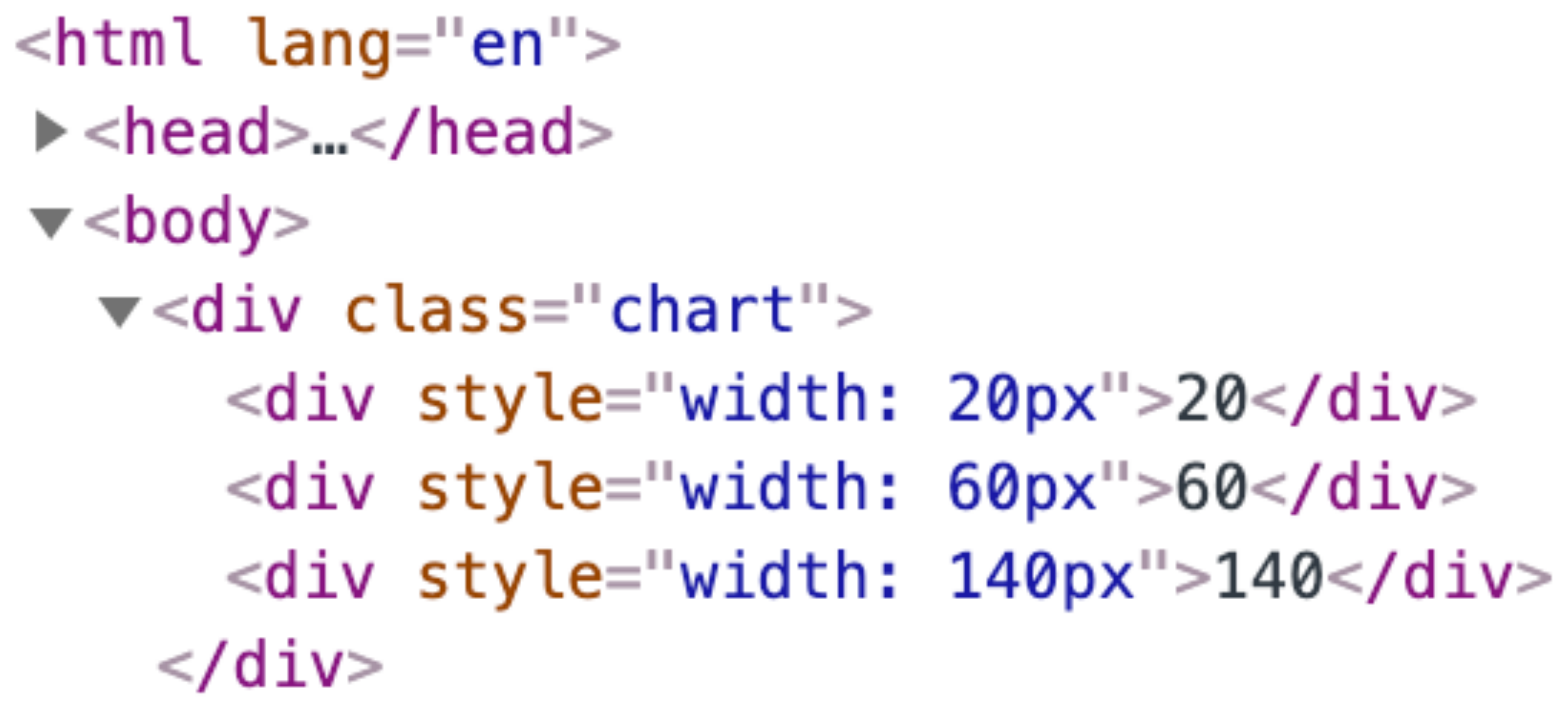
bind data to DOM elements

Document object model (DOM)



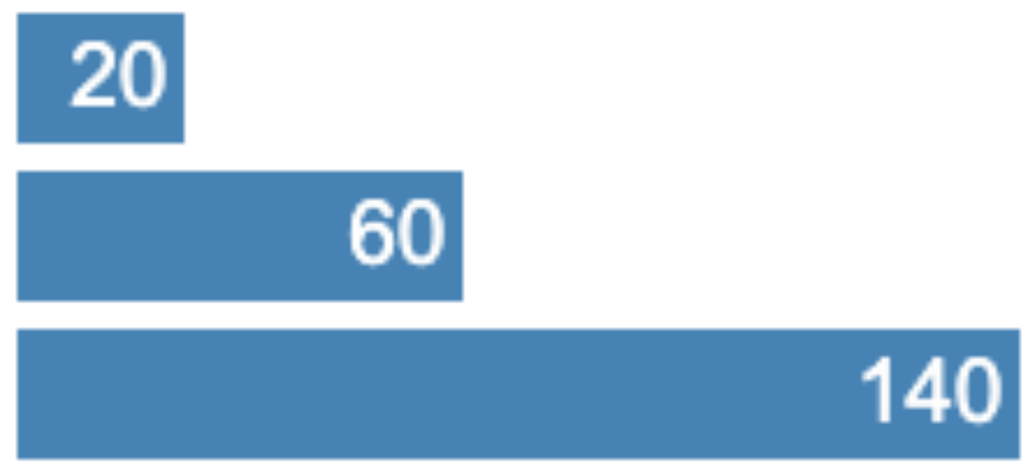
```
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▼<div class="chart">
      <div style="width: 20px">20</div>
      <div style="width: 60px">60</div>
      <div style="width: 140px">140</div>
    </div>
```

Document object model
(DOM)



```
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▼<div class="chart">
      <div style="width: 20px">20</div>
      <div style="width: 60px">60</div>
      <div style="width: 140px">140</div>
    </div>
```

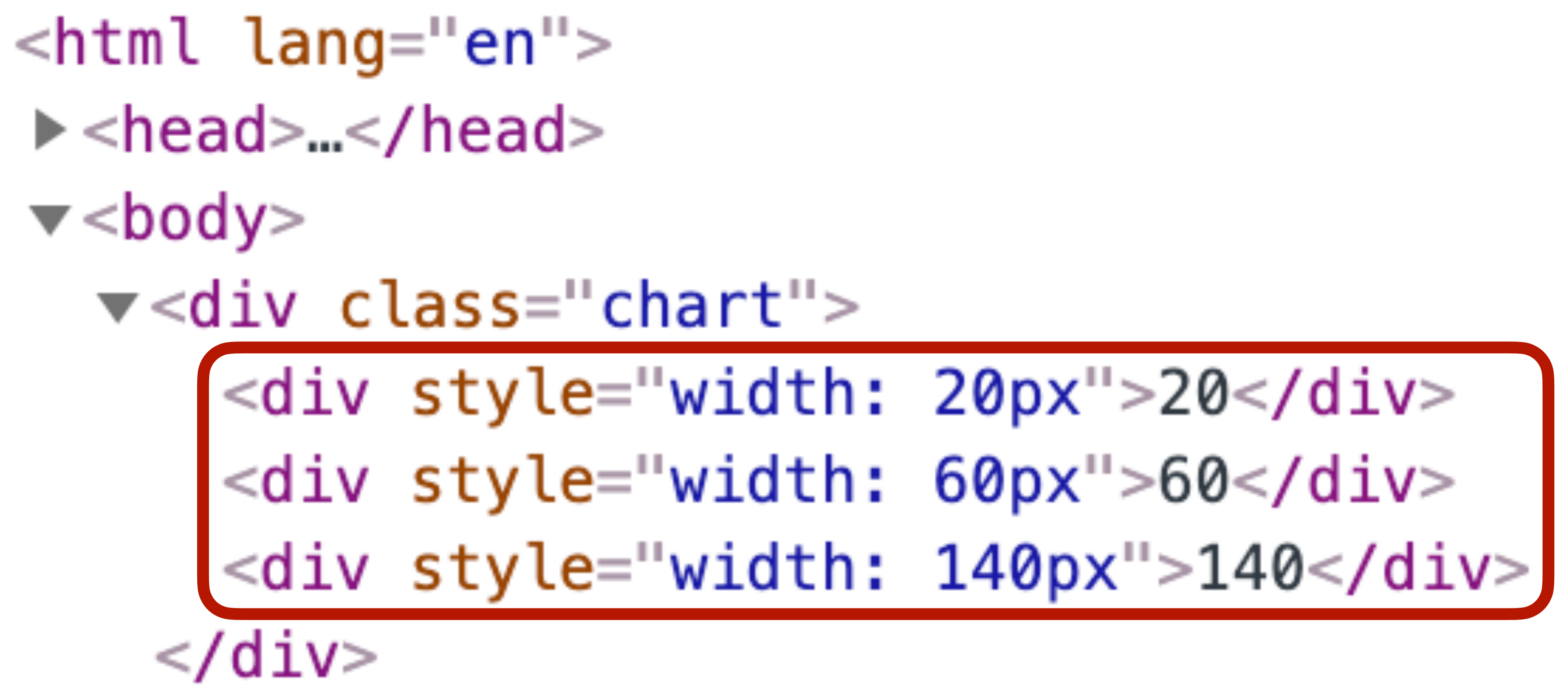
Web page



Data

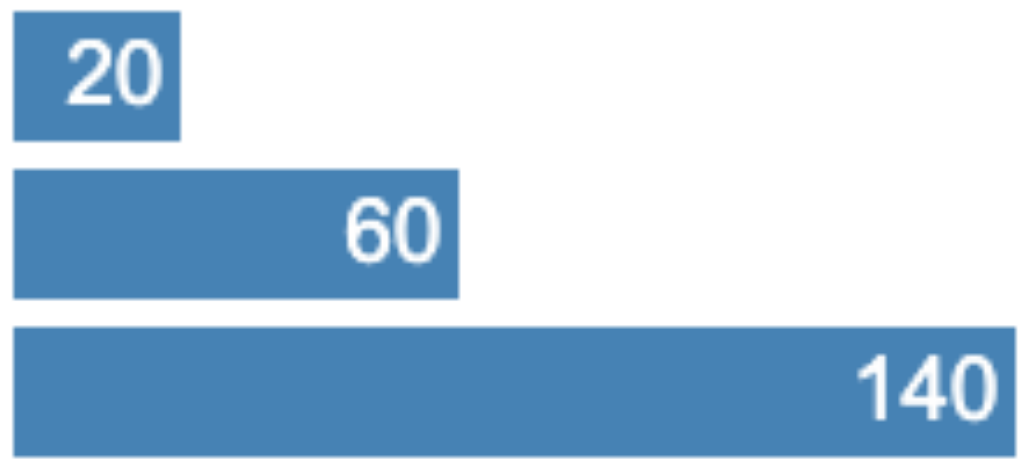
```
const data = [20,60,140];
```

Document object model (DOM)



```
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▼<div class="chart">
      <div style="width: 20px">20</div>
      <div style="width: 60px">60</div>
      <div style="width: 140px">140</div>
    </div>
```

Web page



What is D3?

data driven documents

bind data to DOM elements

What is D3?

data driven documents

bind data to
DOM elements

low-level building blocks

axes,
zooming & panning,
colour palettes, …

What is D3?

data driven documents

bind data to DOM elements

low-level building blocks

axes,
zooming & panning,
colour palettes, …

utility functions

load external data,
parse dates,
binning, …

When should I use D3?

Level of abstraction

Level of abstraction

from scratch
Low
High
three.js
(graphic libraries)

Level of abstraction

from scratch

ready-to-use
chart templates

Low

High

three.js
(graphic libraries)

Google Charts
Chart.js

Level of abstraction

from scratch

composable
building blocks

ready-to-use
chart templates

Low

High

three.js
(graphic libraries)

D3

Vega-lite

Google Charts
Chart.js

Level of abstraction

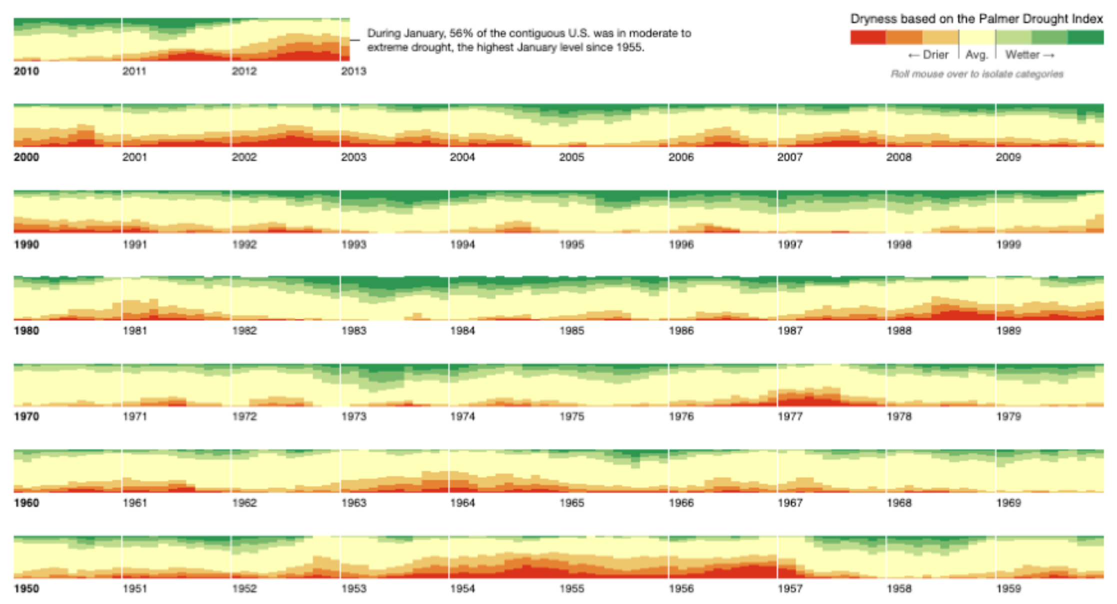
from scratch

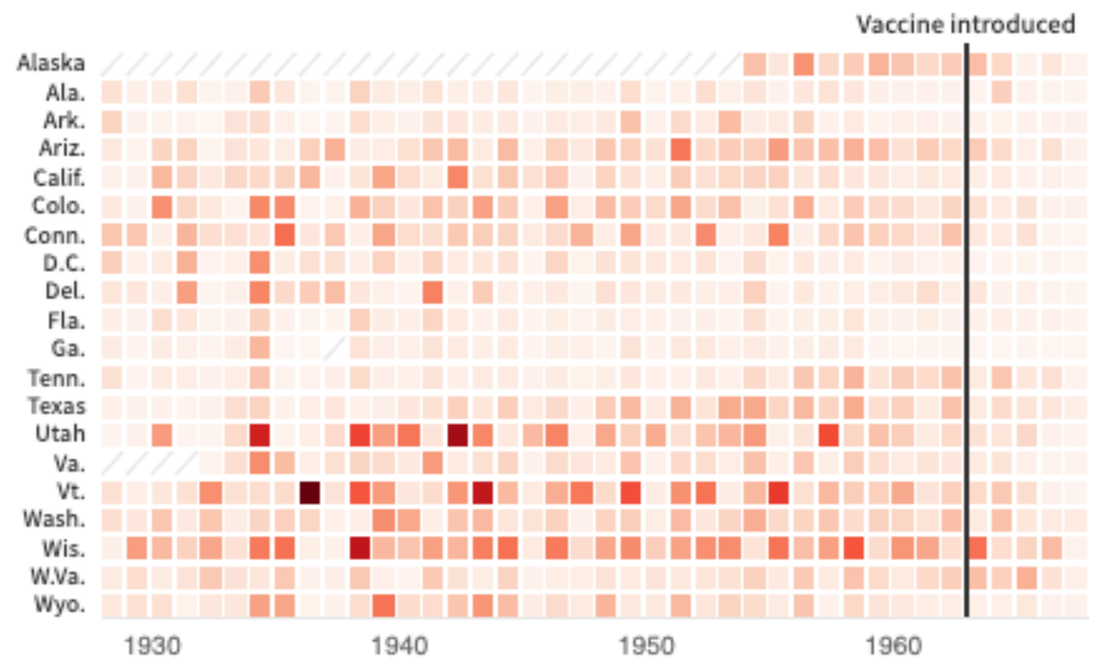
composable
building blocks

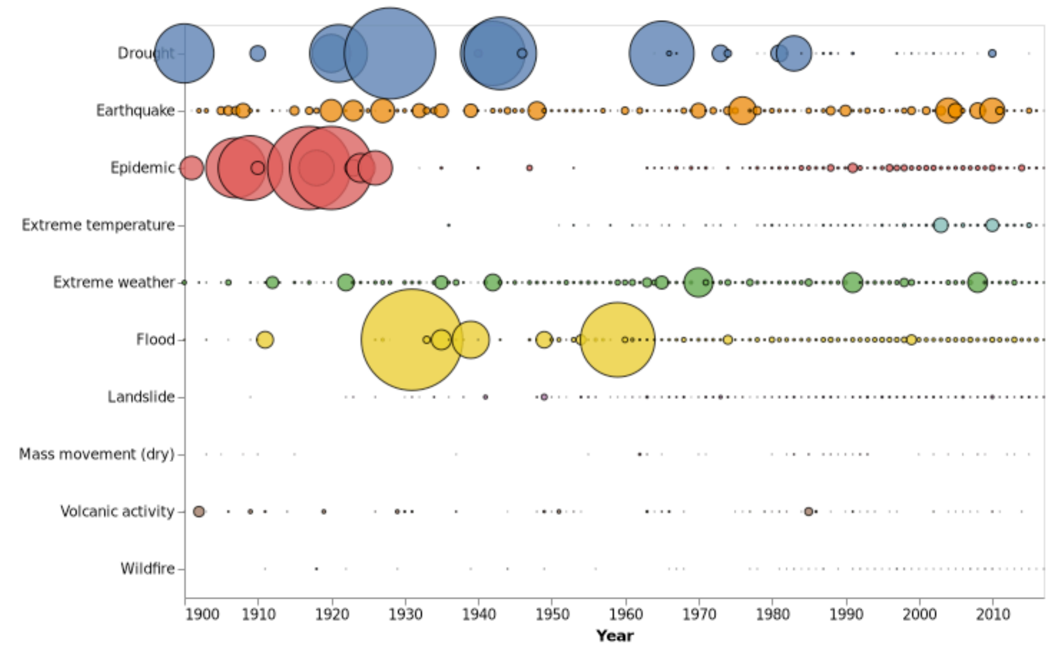
ready-to-use
chart templates

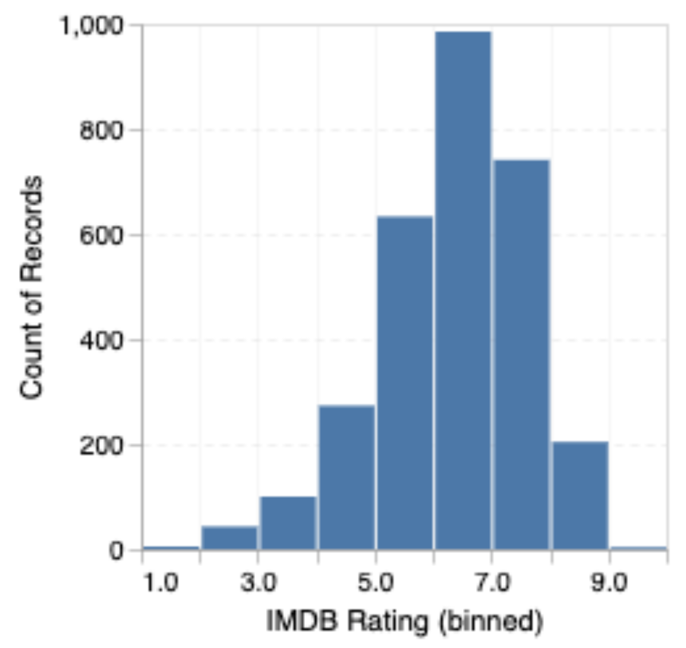
Low

High



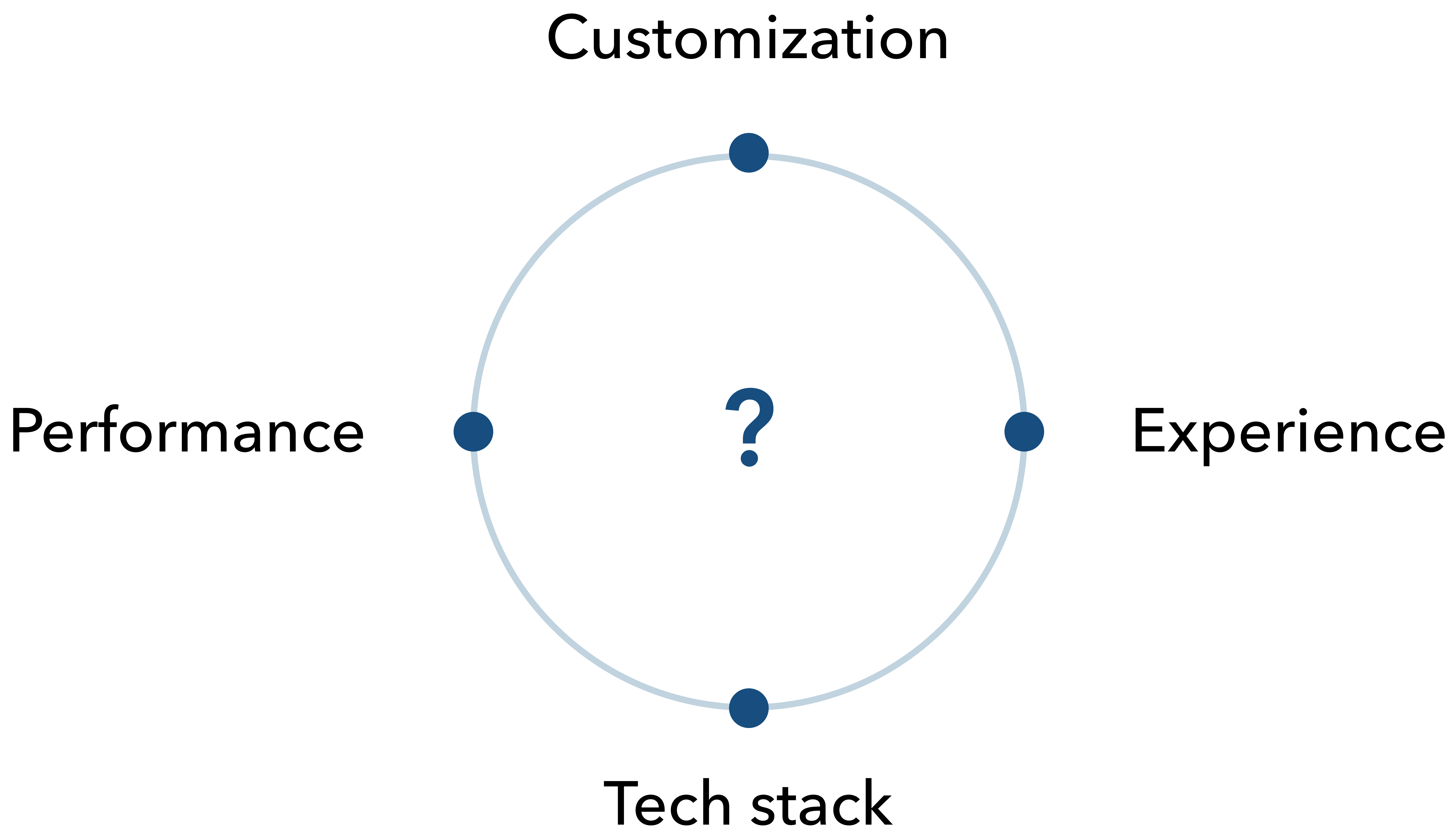






Expressivity

When should I use D3?



1. D3 project setup
2. Bar chart
3. Other D3 tools
4. UpSet plot

D3 & web development

- **D3, version 6**
- **Collection of small modules**
  - Use individual modules or D3 bundle

D3 & web development

- **D3, version 6**
- **Collection of small modules**
  - Use individual modules or D3 bundle
- **Front-end web technologies**
  - HTML, CSS, JavaScript
  - SVG (scalable vector graphics)

D3 & web development

- **D3, version 6**
- **Collection of small modules**
  - Use individual modules or D3 bundle
- **Front-end web technologies**
  - HTML, CSS, JavaScript
  - SVG (scalable vector graphics)
- **Environment**
  - Run a local web server
    - Command line (if *Python* is installed): `python -m http.server`
    - IDE (e.g., WebStorm)
  - **Observable notebooks** (observablehq.com)

D3 project structure

```
project-folder/
  index.html
  js/
    d3.v6.min.js   (download from d3js.org)
    main.js
  css/
    styles.css
  data/
    ...
```

HTML boilerplate

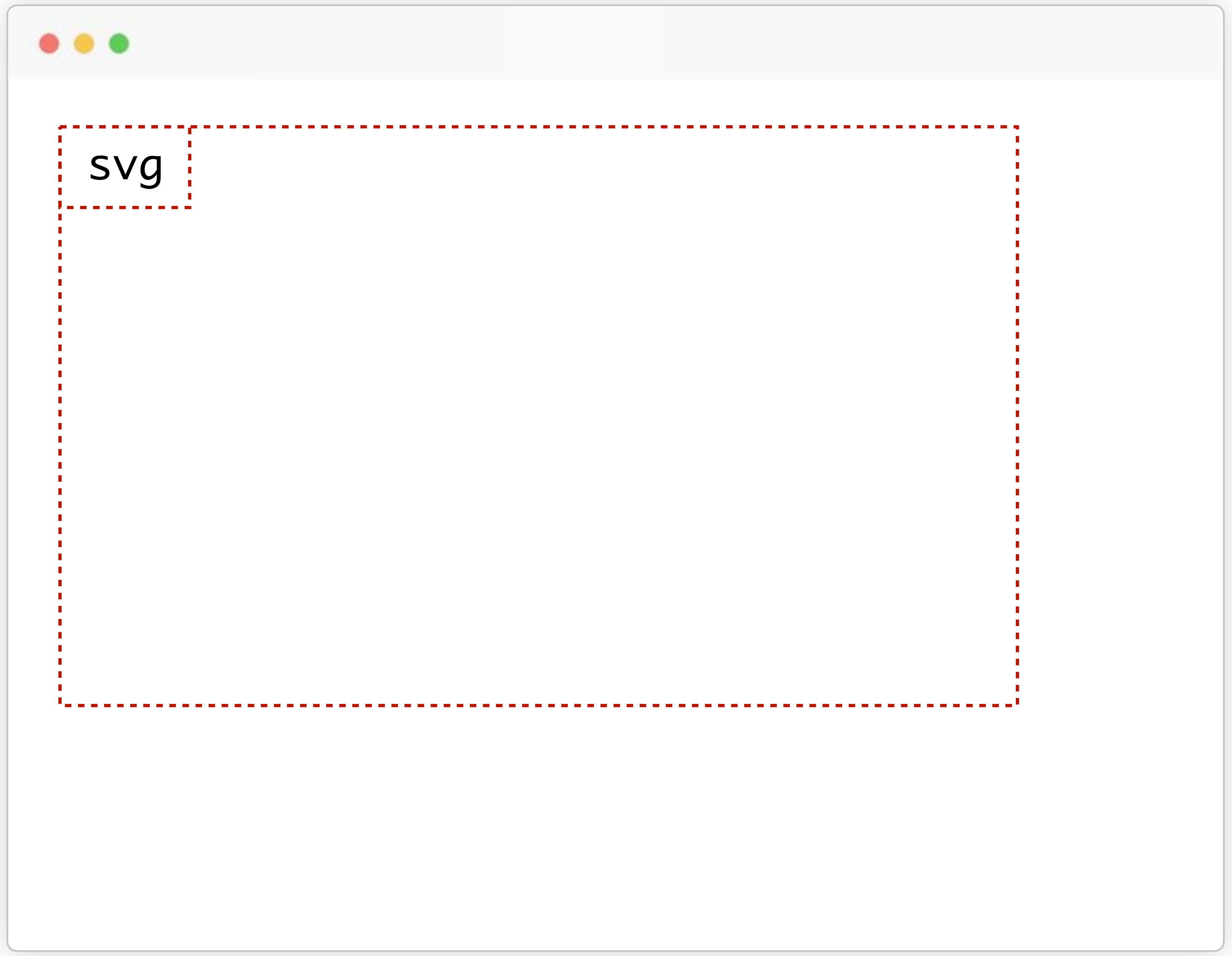
HTML **index.html**

```html
<!DOCTYPE HTML>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Learning D3</title>

    <!-- Load external CSS file -->
    <link href="css/styles.css" rel="stylesheet">
</head>
<body>
    <svg id="chart" width="500" height="300"></svg>

    <!-- Load external JS files -->
  <script src="js/d3.v6.min.js"></script>
  <script src="js/main.js"></script>
</body>
</html>
```

HTML boilerplate

HTML index.html

```html
<!DOCTYPE HTML>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Learning D3</title>

    <!-- Load external CSS file -->
    <link href="css/styles.css" rel="stylesheet">
</head>
<body>
    <svg id="chart" width="500" height="300"></svg>

    <!-- Load external JS files -->
  <script src="js/d3.v6.min.js"></script>
  <script src="js/main.js"></script>
</body>
</html>
```

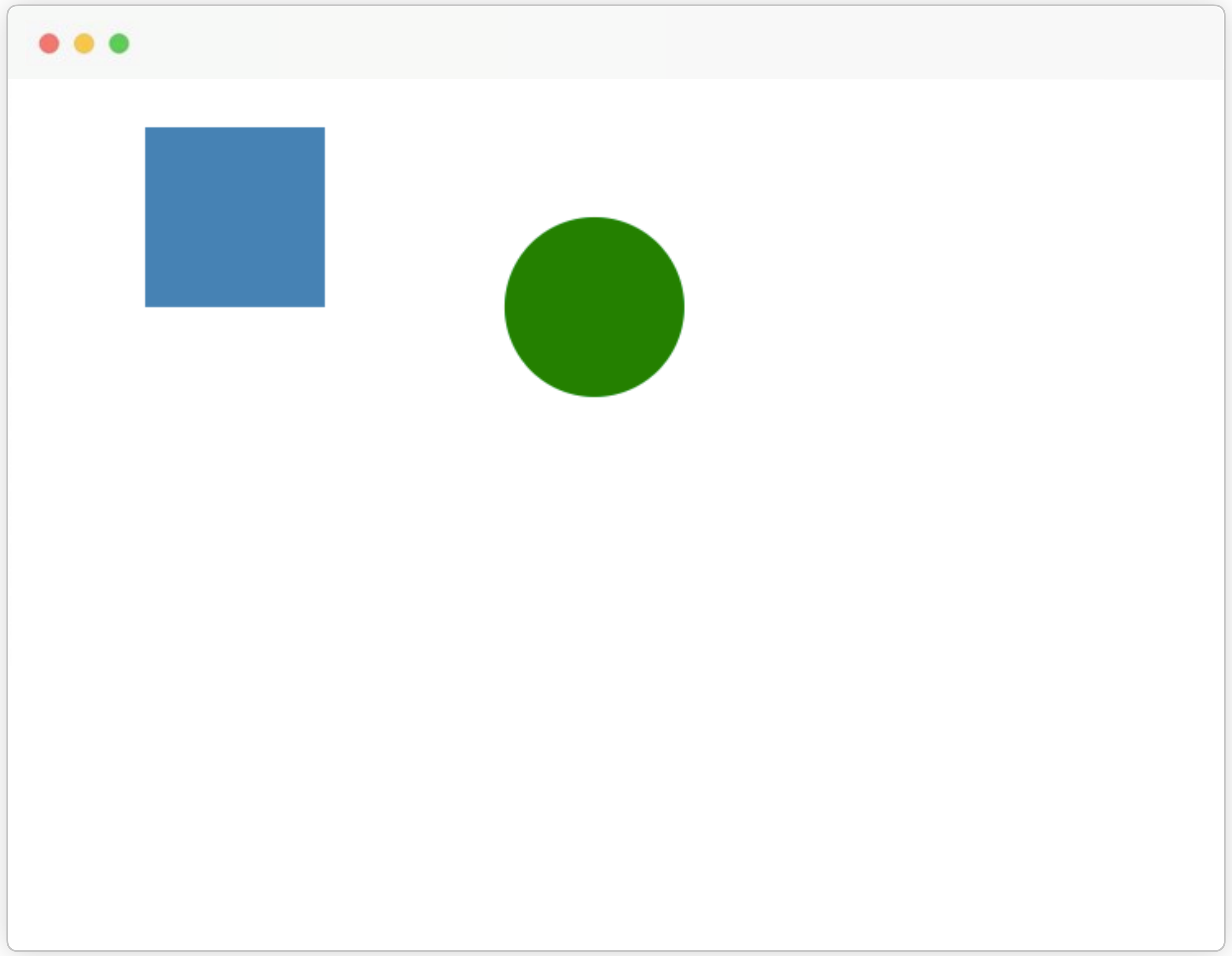


Add SVG elements

Manually add elements

HTML index.html

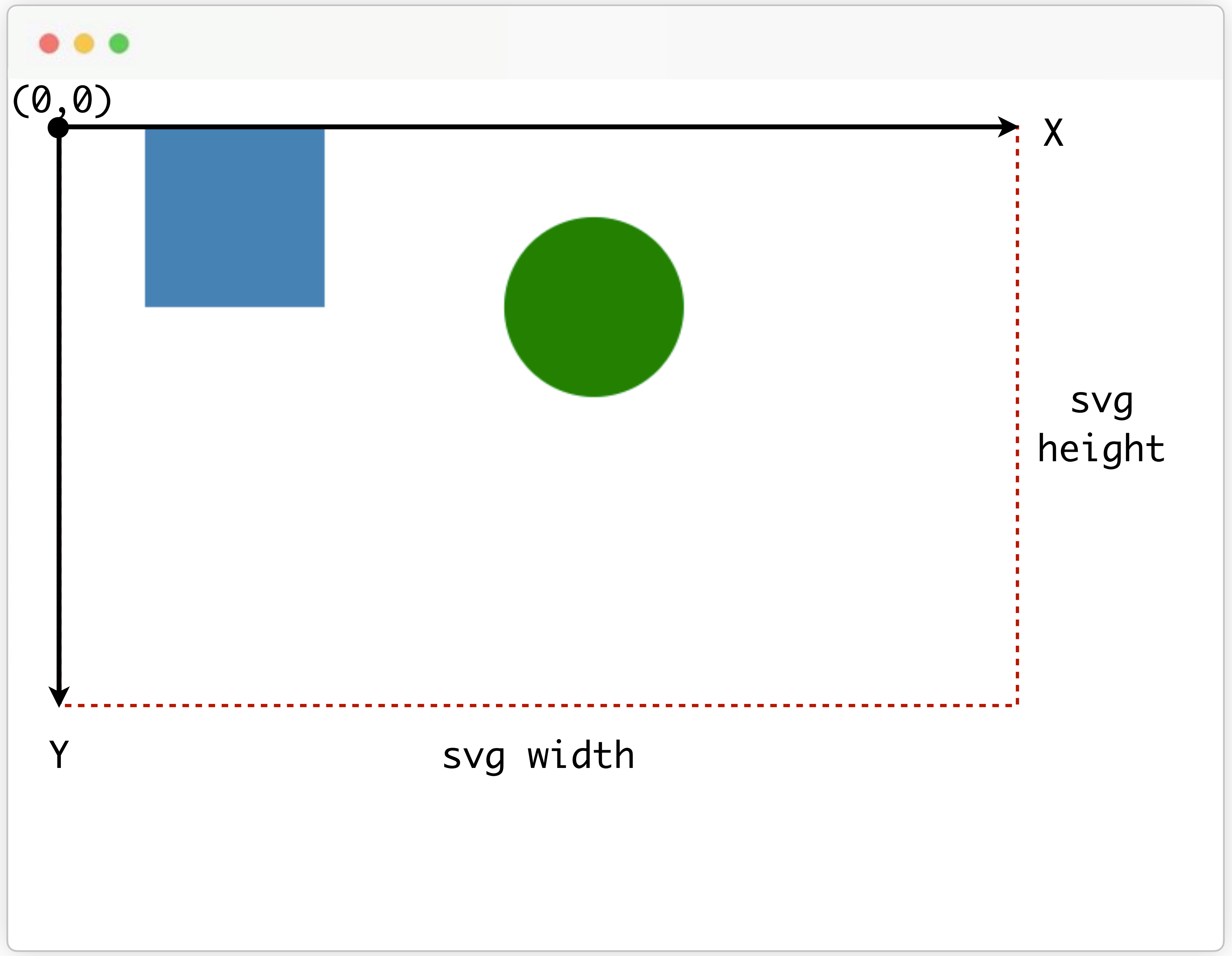
```
<body>

  <svg width="500" height="300">

    <rect
      width="100"
      height="100"
      x="50"
      y="0"
      fill="steelblue"
    />

    <circle
      r="50"
      cy="100"
      cx="300"
      fill="green"
    />

  </svg>
```



Manually add elements

HTML index.html

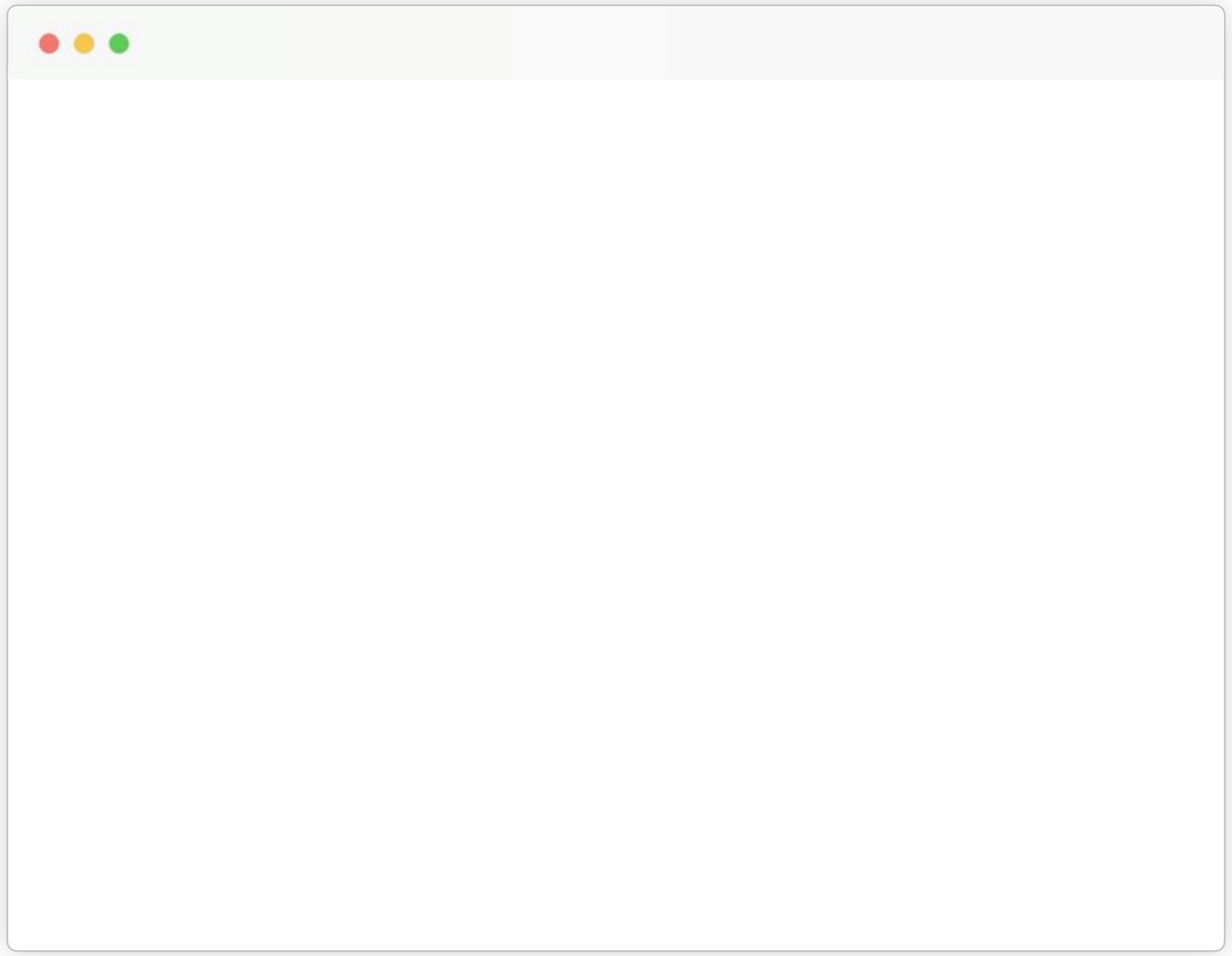
```
<body>

  <svg width="500" height="300">

    <rect
      width="100"
      height="100"
      x="50"
      y="0"
      fill="steelblue"
    />

    <circle
      r="50"
      cy="100"
      cx="300"
      fill="green"
    />

  </svg>
```



Add elements with D3

JS main.js

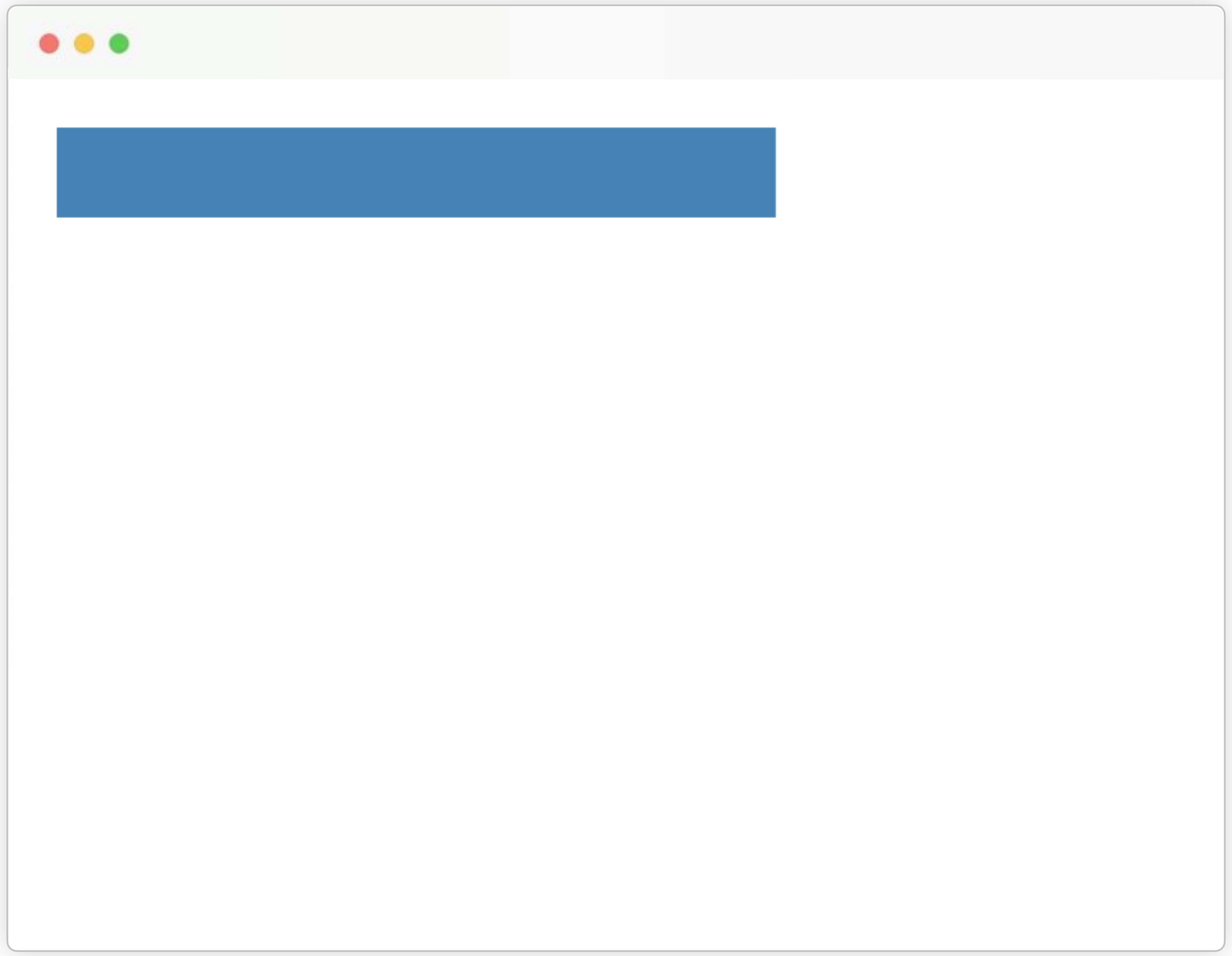
```
d3.select('svg')
```



Add elements with D3

JS main.js

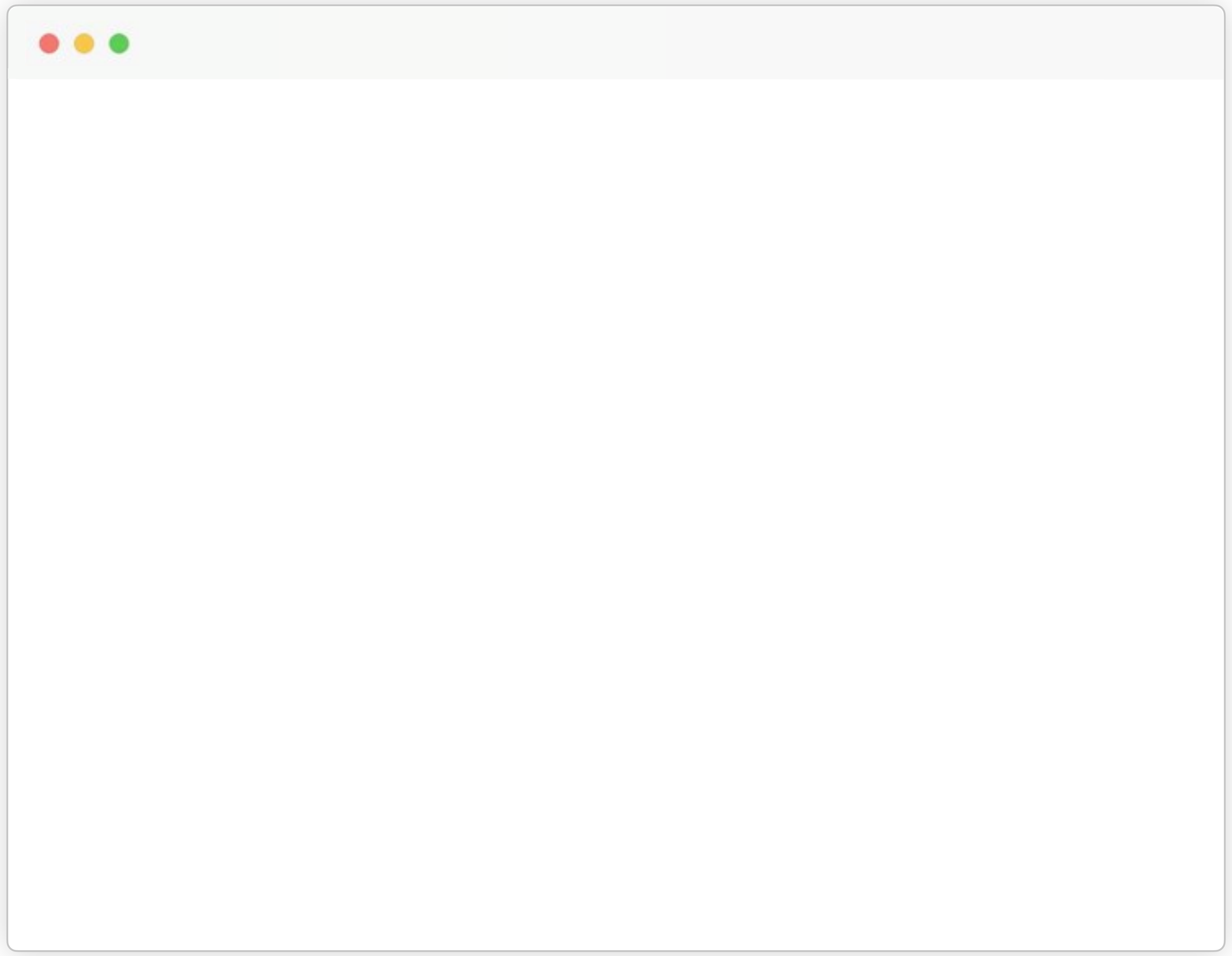
```
d3.select('svg').append('rect')
    .attr('fill', 'steelblue')
    .attr('width', 400)
    .attr('height', 50)
```



Bind data to visual elements (D3 selections)

JS main.js

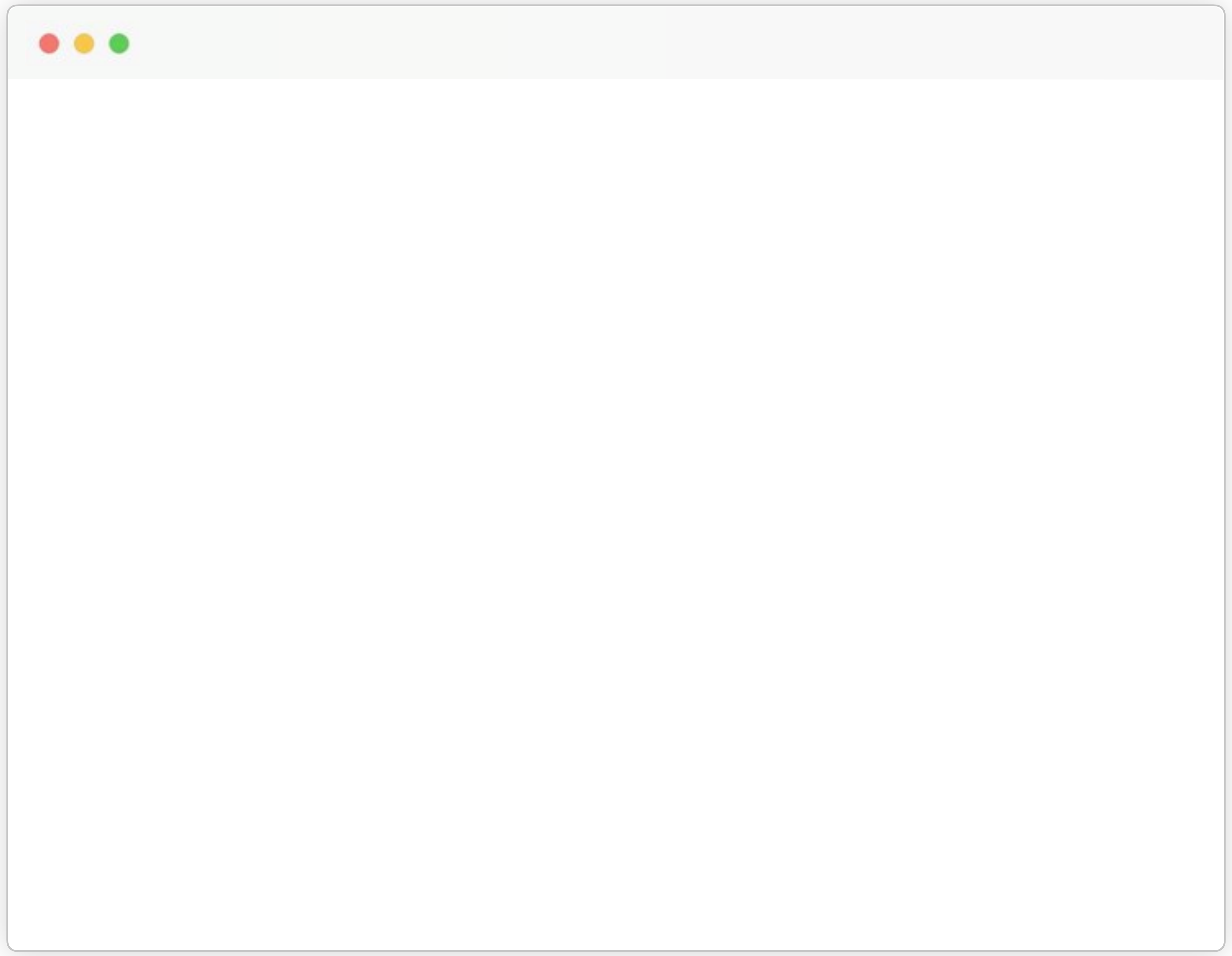
```
const summits = ['Everest', 'Aconcagua', 'Denali'];
```



Bind data to visual elements (D3 selections)

JS **main.js**

```
const summits = ['Everest', 'Aconcagua', 'Denali'];

const svg = d3.select('svg');

svg.selectAll('rect')
  .data(summits)
```



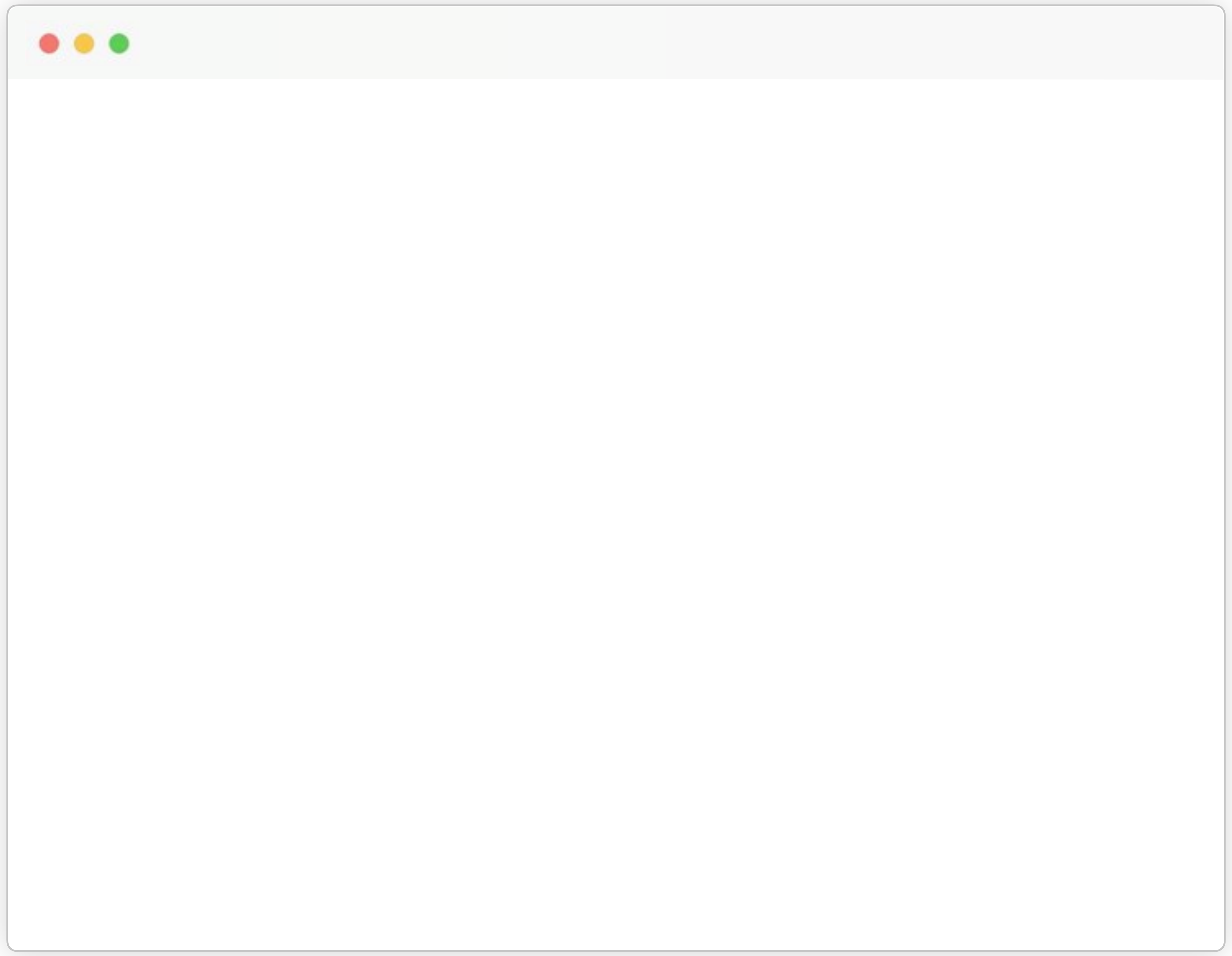
Bind data to visual elements (D3 selections)

JS main.js

```
const summits = ['Everest', 'Aconcagua', 'Denali'];

const svg = d3.select('svg');

svg.selectAll('rect')
  .data(summits)
```

$\longrightarrow$ **0 elements**

$\longrightarrow$ **3 elements**

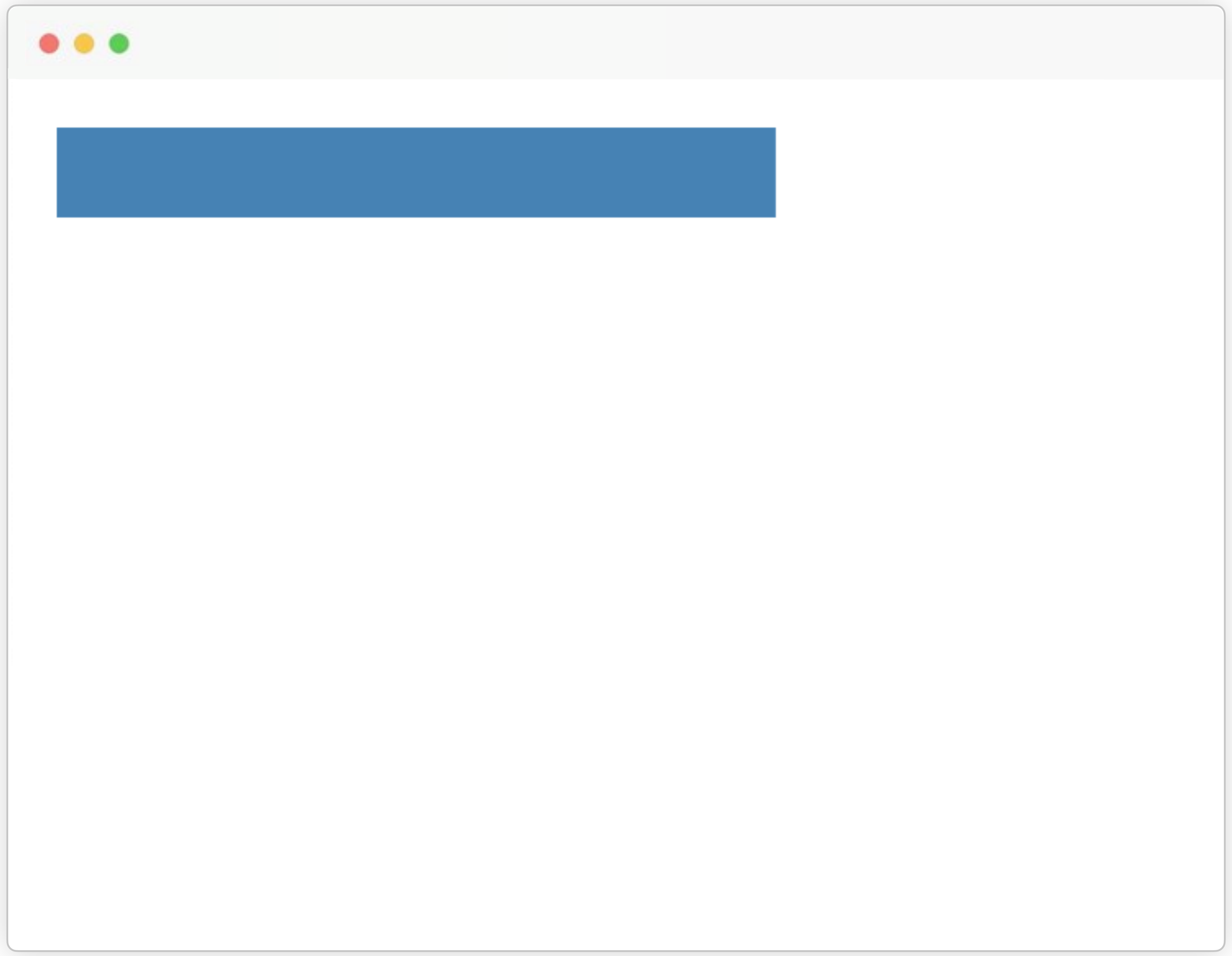


Bind data to visual elements (D3 selections)

JS main.js

```
const summits = ['Everest', 'Aconcagua', 'Denali'];

const svg = d3.select('svg');

svg.selectAll('rect')
  .data(summits)
  .join('rect')
    .attr('fill', 'steelblue')
    .attr('width', 400)
    .attr('height', 50);
```

$\longrightarrow$ 0 elements
$\longrightarrow$ 3 elements
$\longrightarrow$ **append 3 elements**

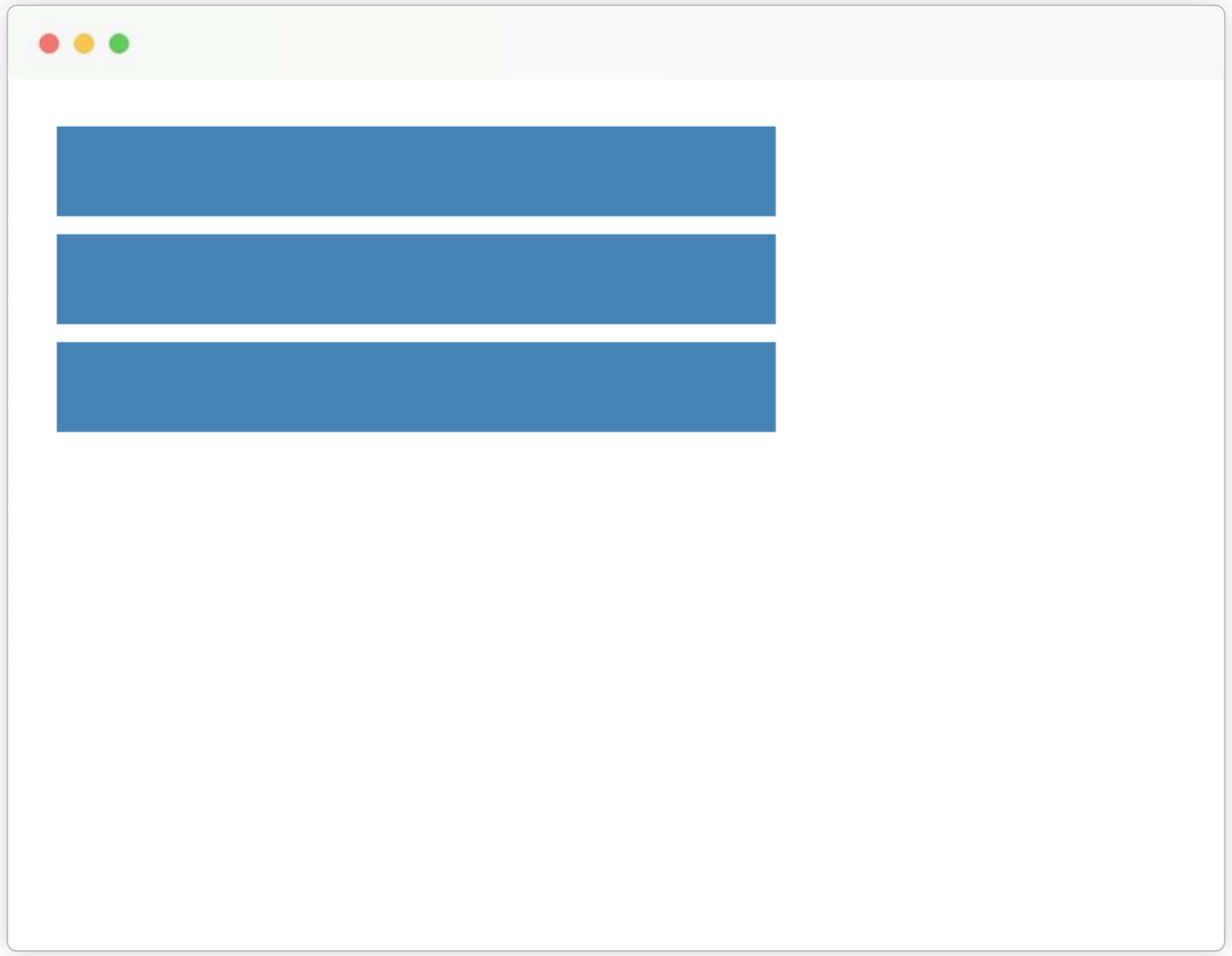


Bind data to visual elements (D3 selections)

JS main.js

```
const summits = ['Everest', 'Aconcagua', 'Denali'];

const svg = d3.select('svg');

svg.selectAll('rect')
  .data(summits)
  .join('rect')
    .attr('fill', 'steelblue')
    .attr('width', 400)
    .attr('height', 50)
    .attr('y', (d, index) => index * 60);
```

Anonymous function



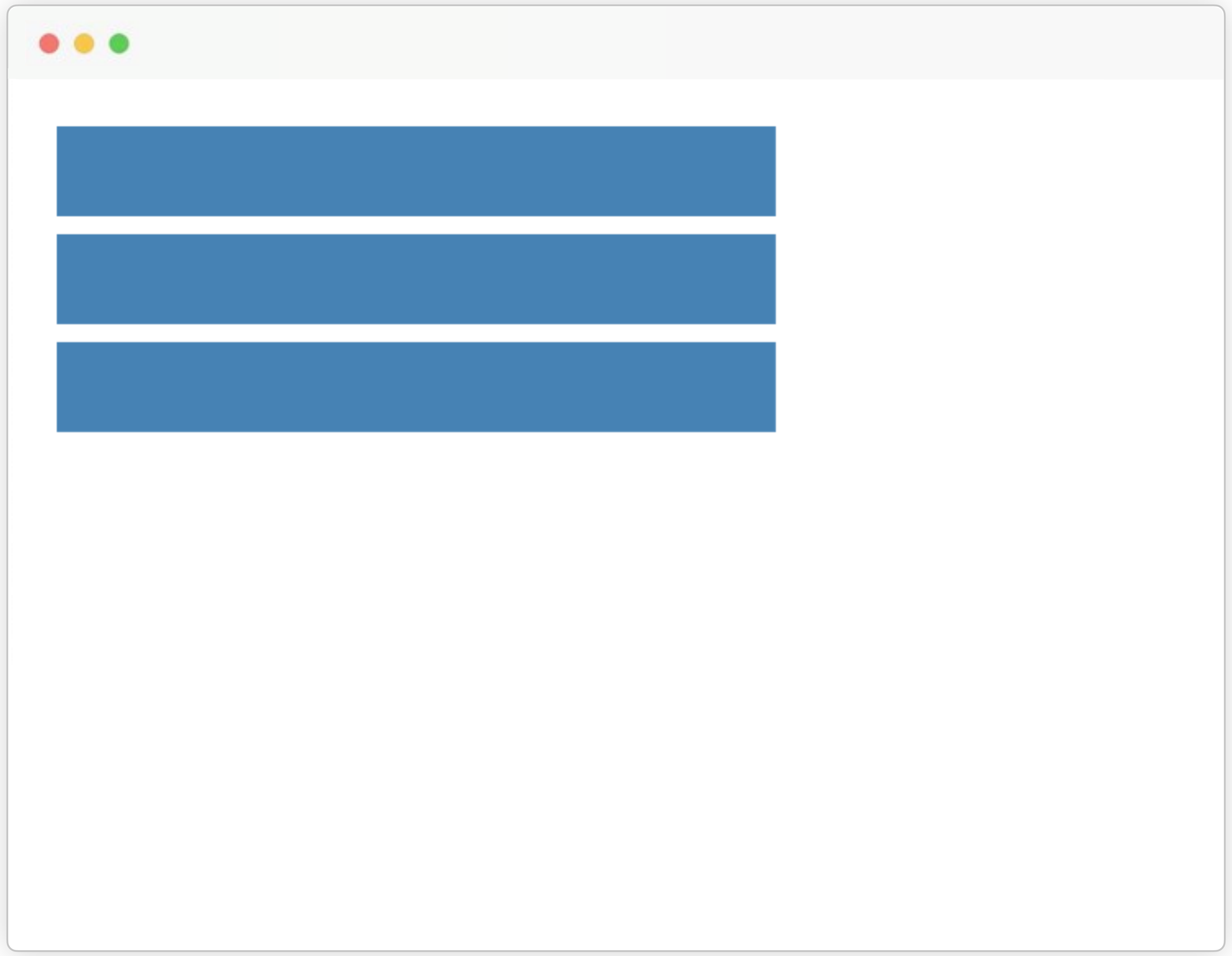
Style elements using CSS

JS main.js

```
const summits = ['Everest', 'Aconcagua', 'Denali'];

const svg = d3.select('svg');

svg.selectAll('rect')
  .data(summits)
  .join('rect')
    .attr('class', 'bar')
    .attr('width', 400)
    .attr('height', 50)
    .attr('y', (d, index) => index * 60);
```

CSS styles.css

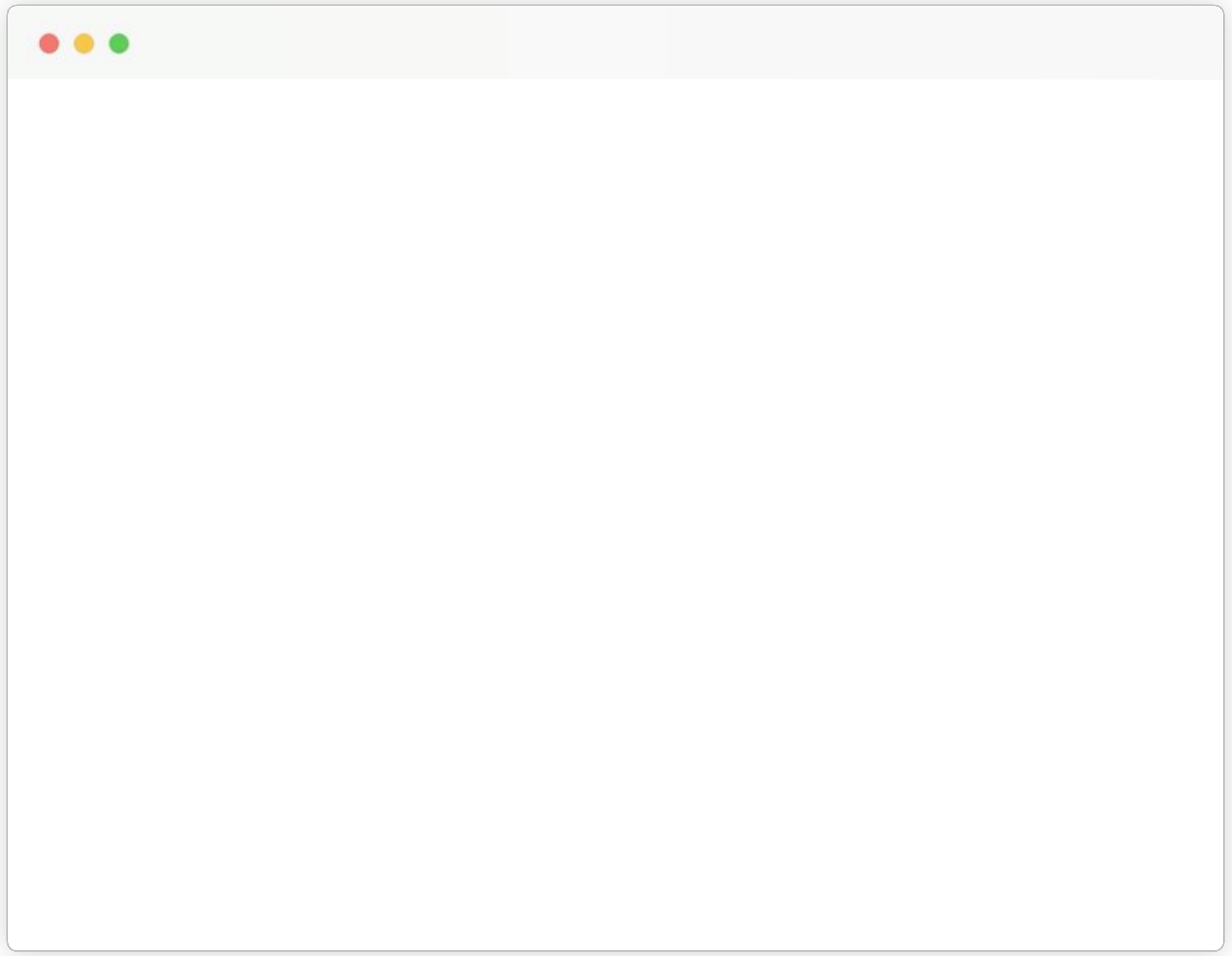
```
.bar {
  fill: steelblue;
}
```



Load external data

CSV summits.csv

title	elevation
Everest	8849
Kilimanjaro	5895
Vinson	4892
Aconcagua	6961
Denali	6194
Elbrus	5642
Puncak Jaya	4884



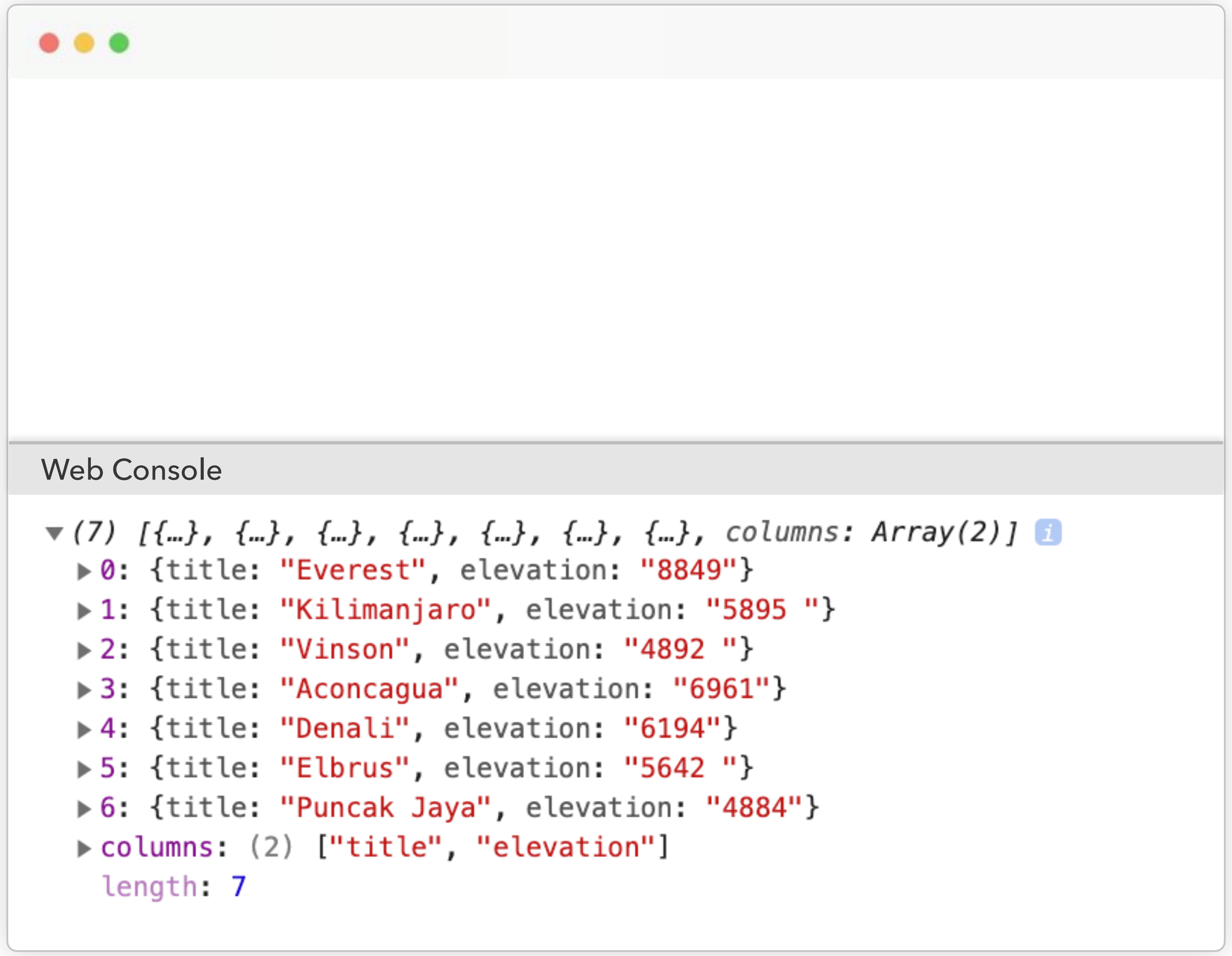
Load external data

CSV summits.csv

title	elevation
Everest	8849
Kilimanjaro	5895
Vinson	4892
Aconcagua	6961
Denali	6194
Elbrus	5642
Puncak Jaya	4884

JS main.js

```
d3.csv('data/summits.csv')
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.error('Error loading the data');
  });
```



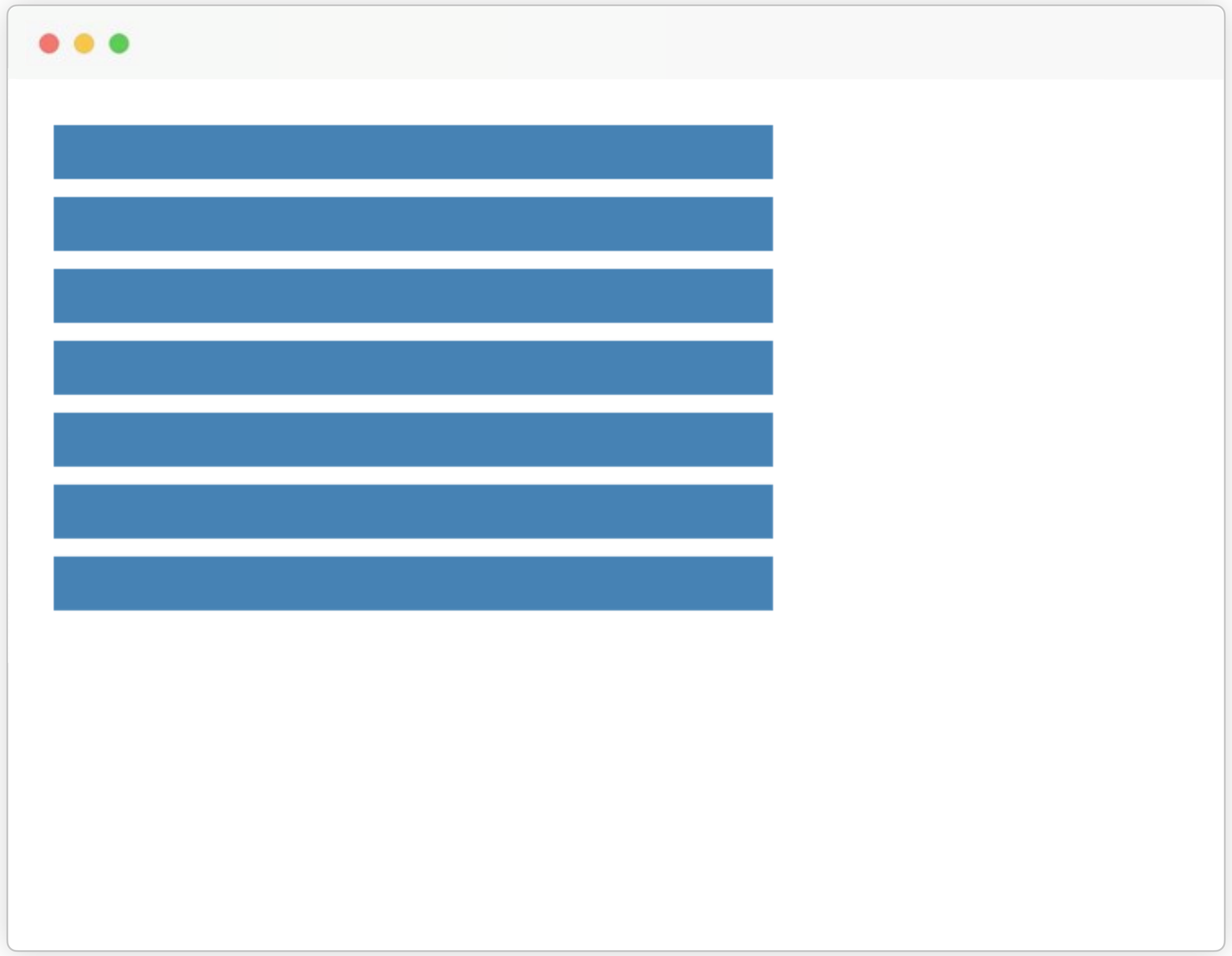
Web Console

```
▼(7) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, columns: Array(2)]
  ▶0: {title: "Everest", elevation: "8849"}
  ▶1: {title: "Kilimanjaro", elevation: "5895 "}
  ▶2: {title: "Vinson", elevation: "4892 "}
  ▶3: {title: "Aconcagua", elevation: "6961"}
  ▶4: {title: "Denali", elevation: "6194"}
  ▶5: {title: "Elbrus", elevation: "5642 "}
  ▶6: {title: "Puncak Jaya", elevation: "4884"}
  ▶columns: (2) ["title", "elevation"]
   length: 7
```

Load external data

JS main.js

```
const svg = d3.select('svg');

d3.csv('data/summits.csv')
  .then(data => {
    svg.selectAll('rect')
      .data(data)
      .join('rect')
        .attr('class', 'bar')
        .attr('width', 400)
        .attr('height', 30)
        .attr('y', (d, index) => index * 40);
  });
```



Load external data

JS main.js

```
const svg = d3.select('svg');

d3.csv('data/summits.csv')
  .then(data => {
    svg.selectAll('rect')
      .data(data)
      .join('rect')
        .attr('class', 'bar')
        .attr('width', 400)
        .attr('height', 30)
        .attr('y', (d, index) => index * 40);
  });
```

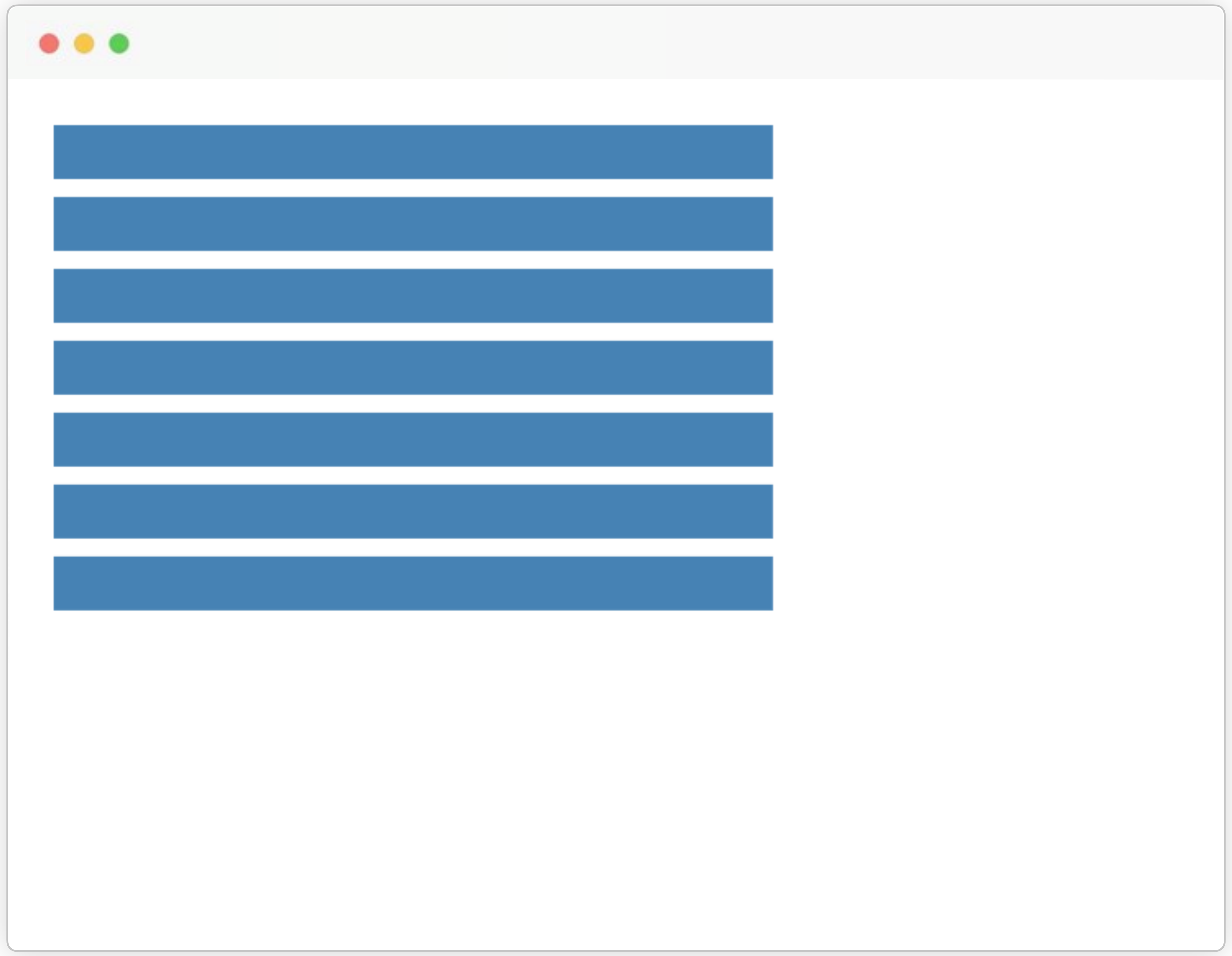
Load data asynchronously

Use data

Load external data

JS main.js

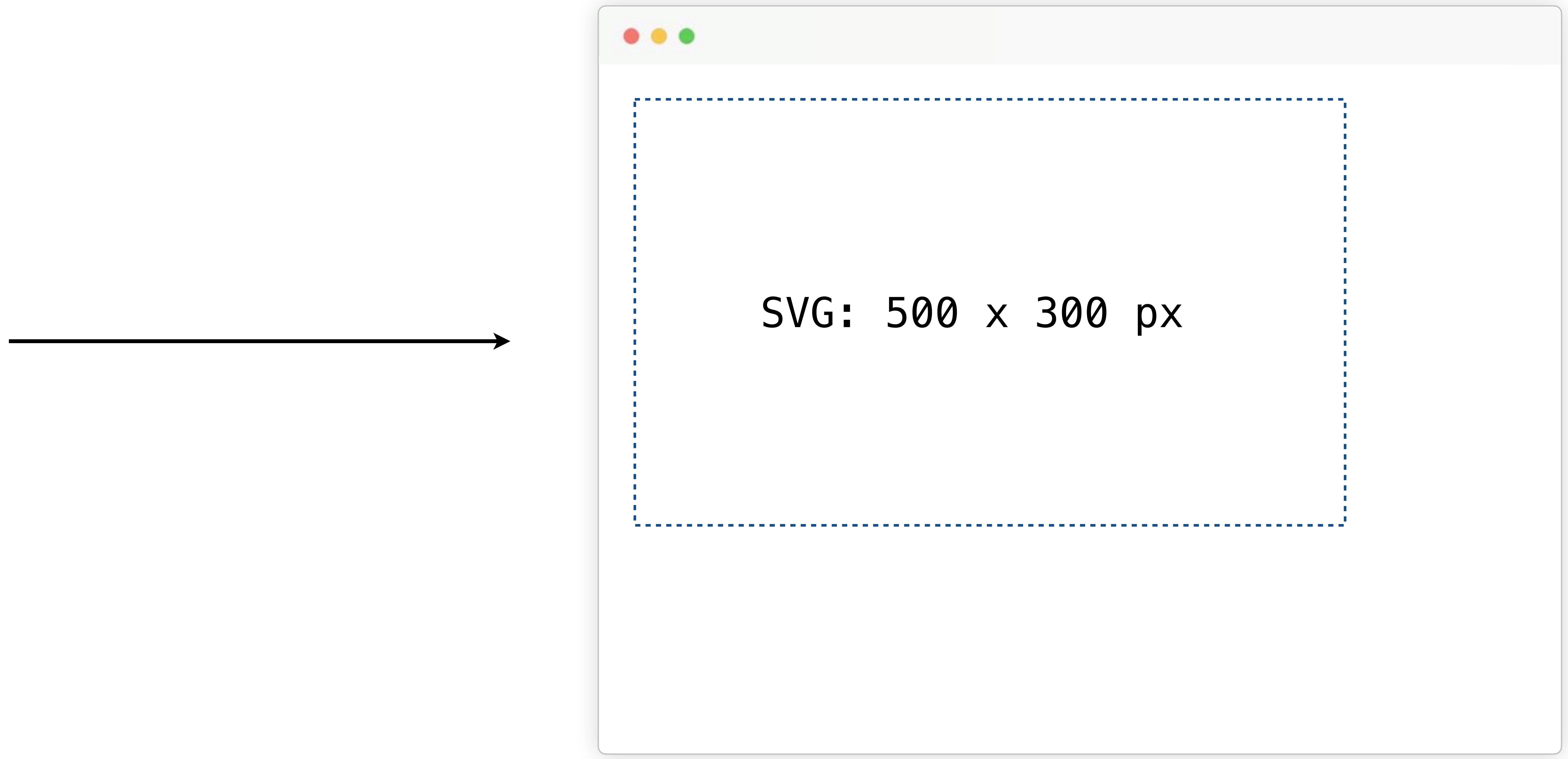
```
const svg = d3.select('svg');

d3.csv('data/summits.csv')
  .then(data => {
    drawChart();
  });


function drawChart(data) {
  svg.selectAll('rect')
    .data(data)
    .join('rect')
      .attr('class', 'bar')
      .attr('width', 400)
      .attr('height', 30)
      .attr('y', (d, index) => index * 40);
}
```



Scales

CSV summits.csv

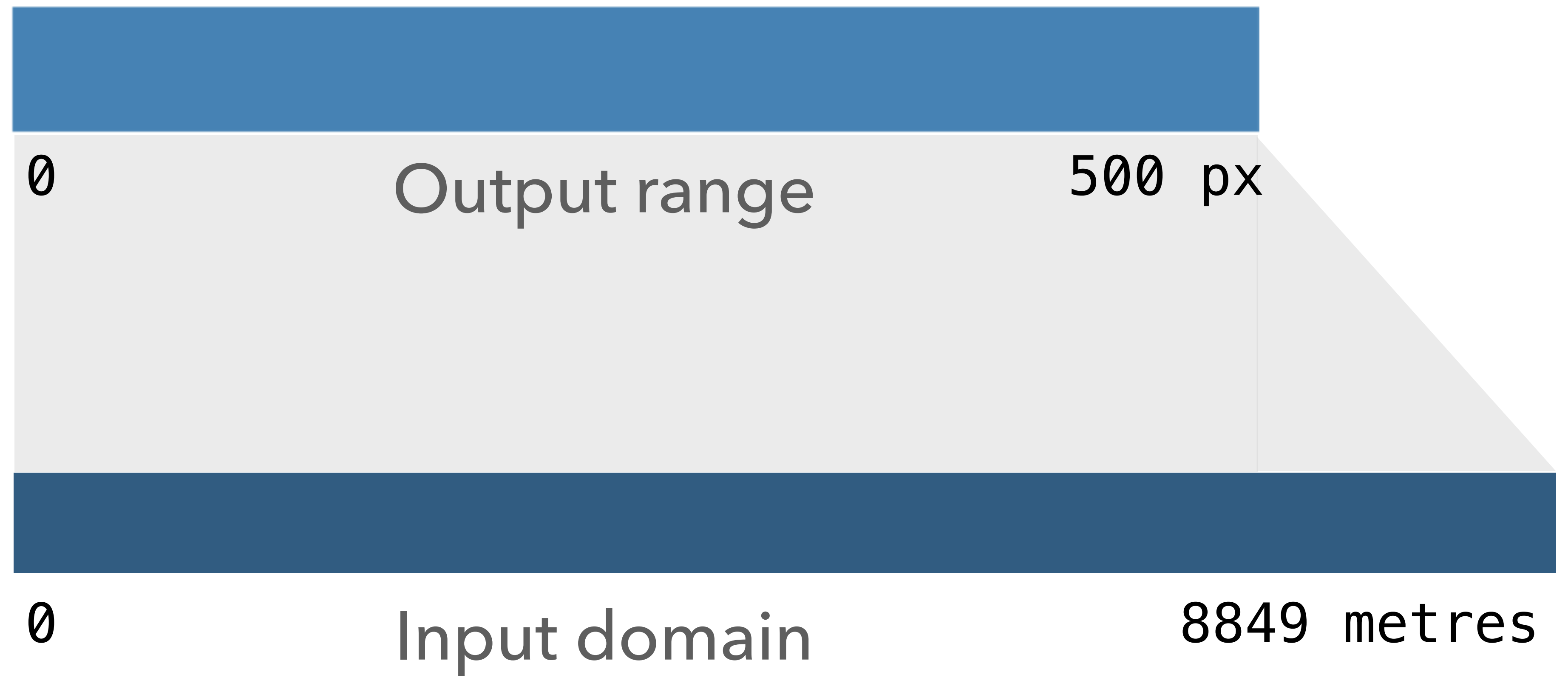
title	elevation
Everest	8849
Kilimanjaro	5895
Vinson	4892
Aconcagua	6961
Denali	6194
Elbrus	5642
Puncak Jaya	4884



Linear scales

JS main.js

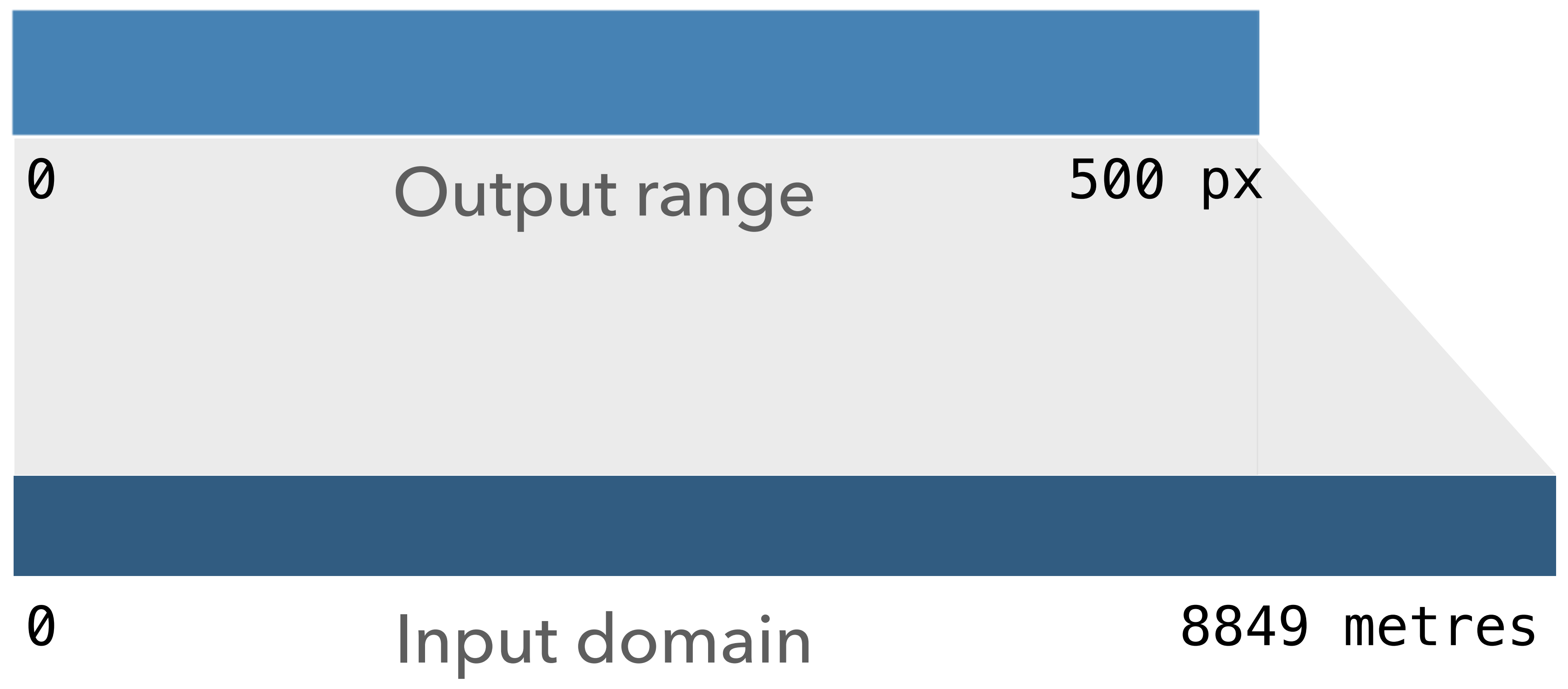
```
// Create a linear scale function
const xScale = d3.scaleLinear()
    .domain([0, 8849])
    .range([0, 500]);
```



Linear scales

JS main.js

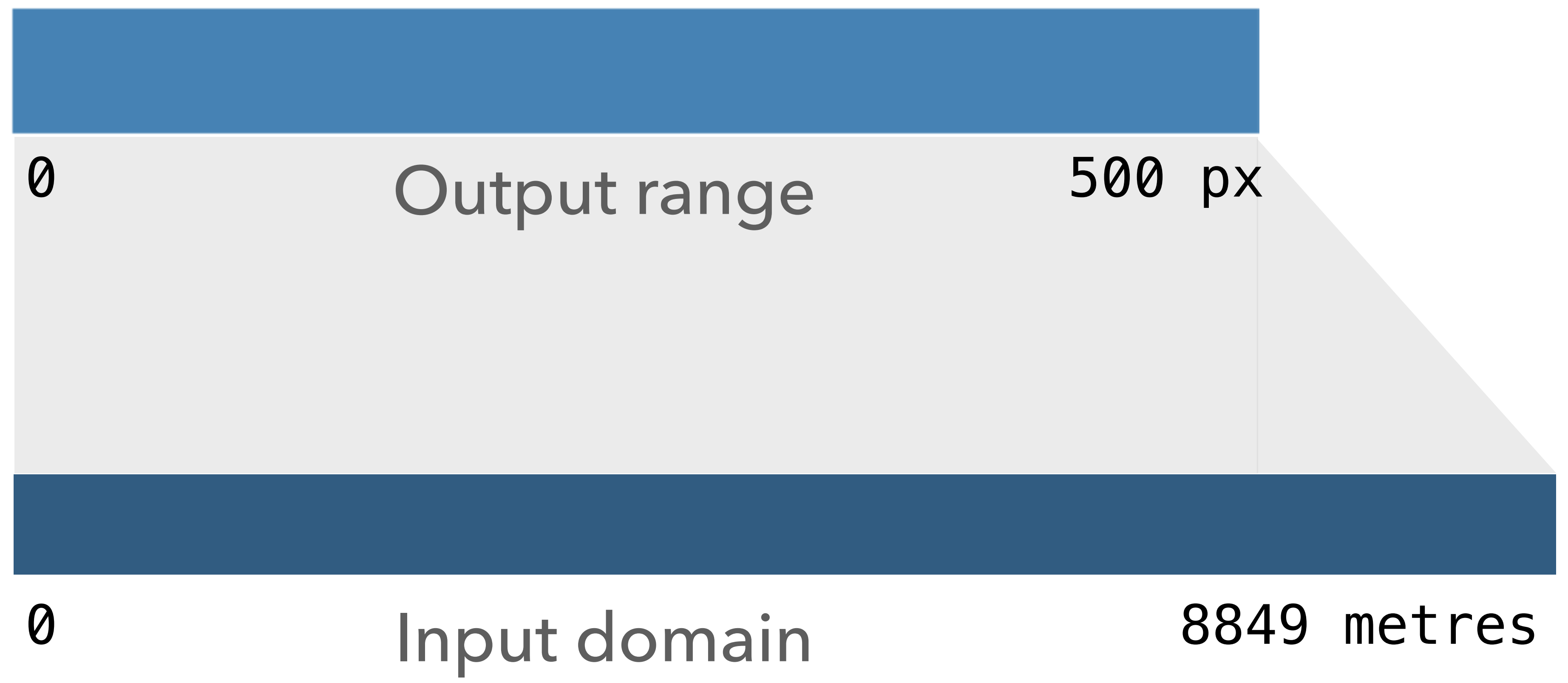
```
// Create a linear scale function
const xScale = d3.scaleLinear()
    .domain([0, 8849])
    .range([0, 500]);

// Call the function and pass an input value
console.log( xScale(8849) );    // Returns: 500 px
console.log( xScale(6000) );    // Returns: 339 px
```



Linear scales

JS main.js

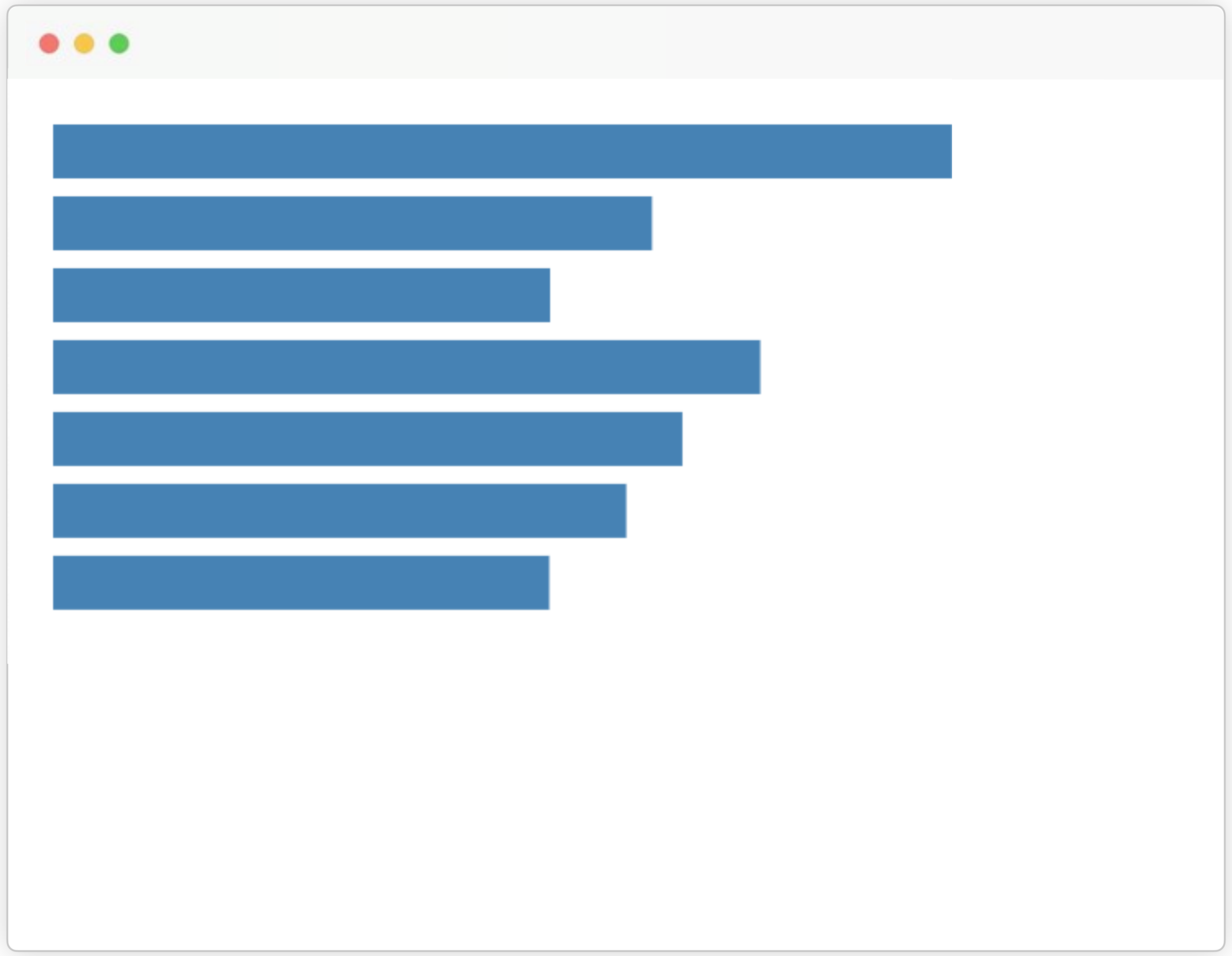
```
// Find maximum value
const max = d3.max(data, d => d.elevation);

// Create a linear scale function
const xScale = d3.scaleLinear()
    .range([0, 500])
    .domain([0, max]);

// Call the function and pass an input value
console.log( xScale(8849) );    // Returns: 500 px
console.log( xScale(6000) );    // Returns: 339 px
```



Linear scales

JS main.js

```
const max = d3.max(data, d => d.elevation);

const xScale = d3.scaleLinear()
    .range([0, 500])
    .domain([0, max]);

svg.selectAll('rect')
    .data(data)
    .join('rect')
      .attr('class', 'bar')
      .attr('width', (d) => xScale(d.elevation))
      .attr('height', 30)
      .attr('y', (d, index) => index * 40);
```



Categorical scales

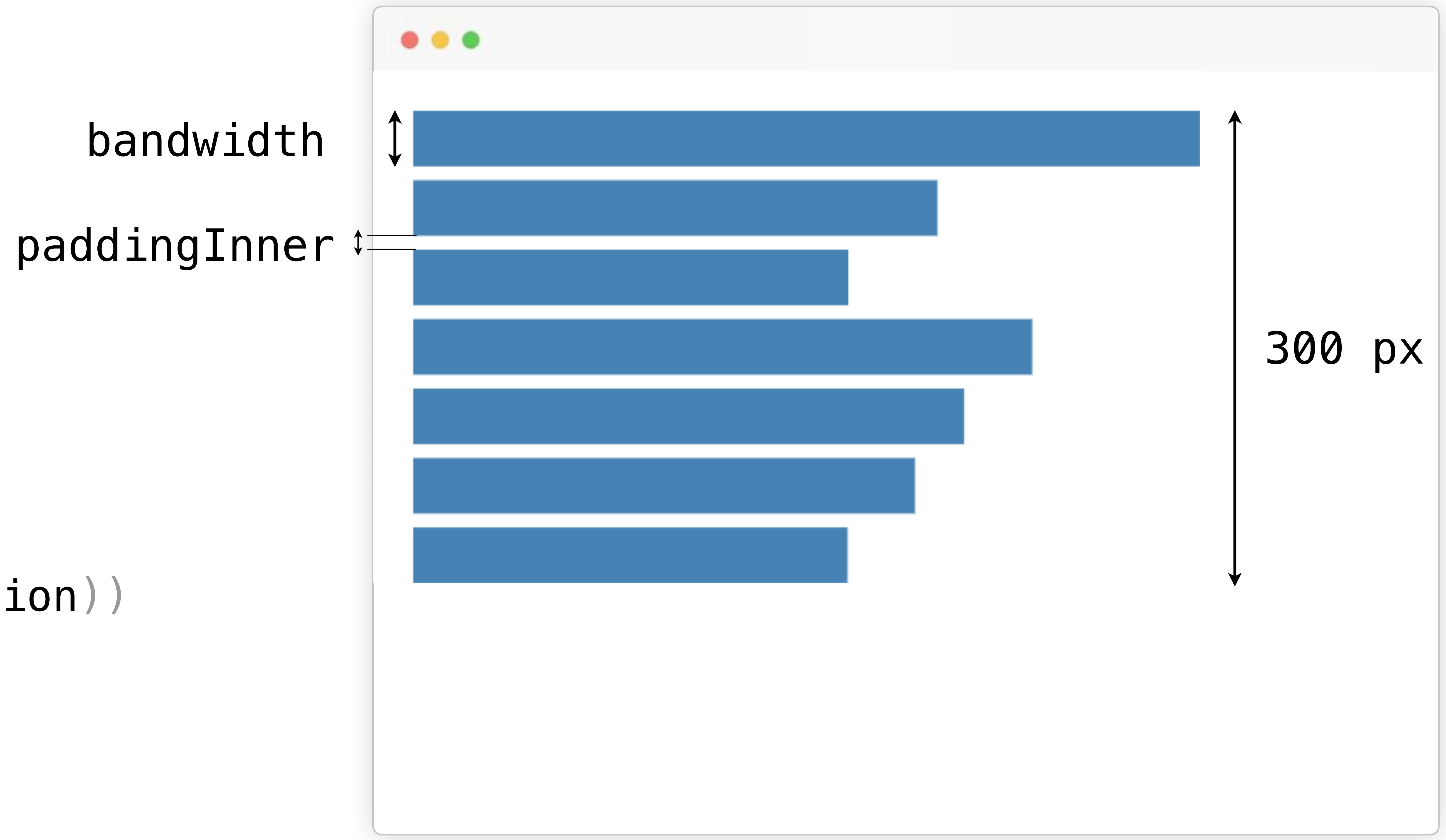
JS main.js

```
const yScale = d3.scaleBand()
    .domain(data.map((d) => d.title ))
    .range([0, 300])
    .paddingInner(0.2);
```

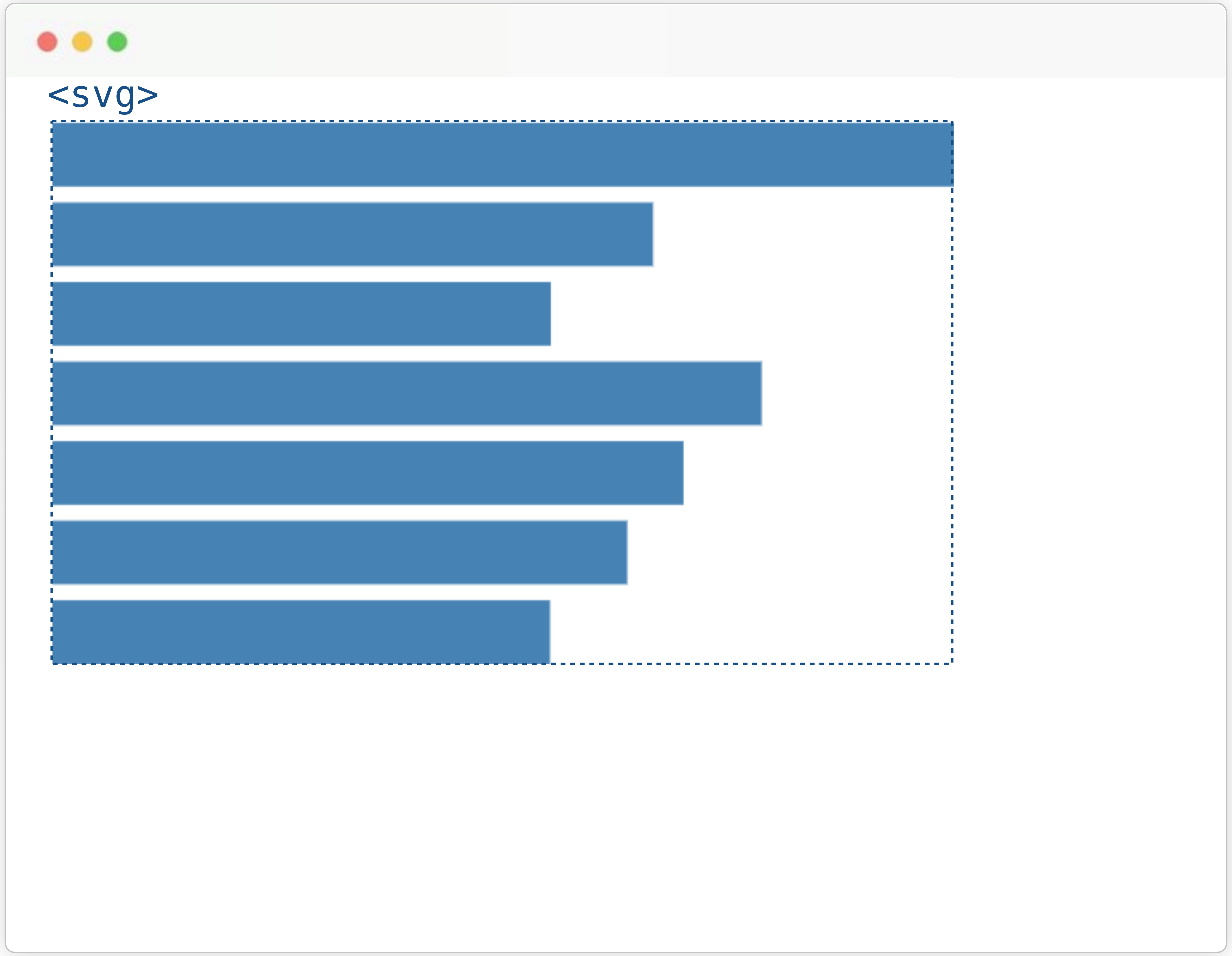
Categorical scales

JS main.js

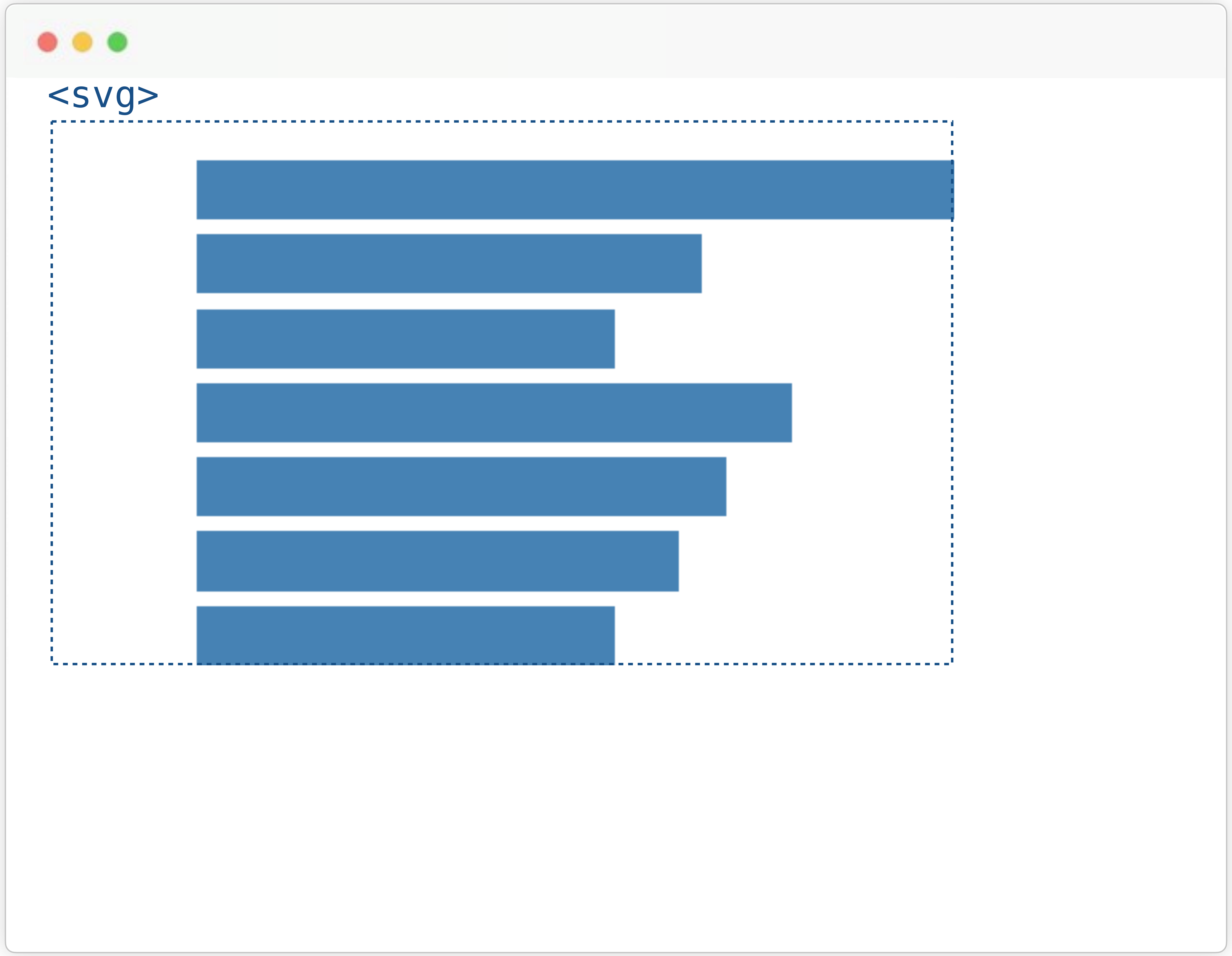
```
const yScale = d3.scaleBand()
    .domain(data.map((d) => d.title ))
    .range([0, 300])
    .paddingInner(0.2);

svg.selectAll('rect')
    .data(data)
    .join('rect')
      .attr('class', 'bar')
      .attr('width', (d) => xScale(d.elevation))
      .attr('height', yScale.bandwidth())
      .attr('y', (d) => yScale(d.title));
```



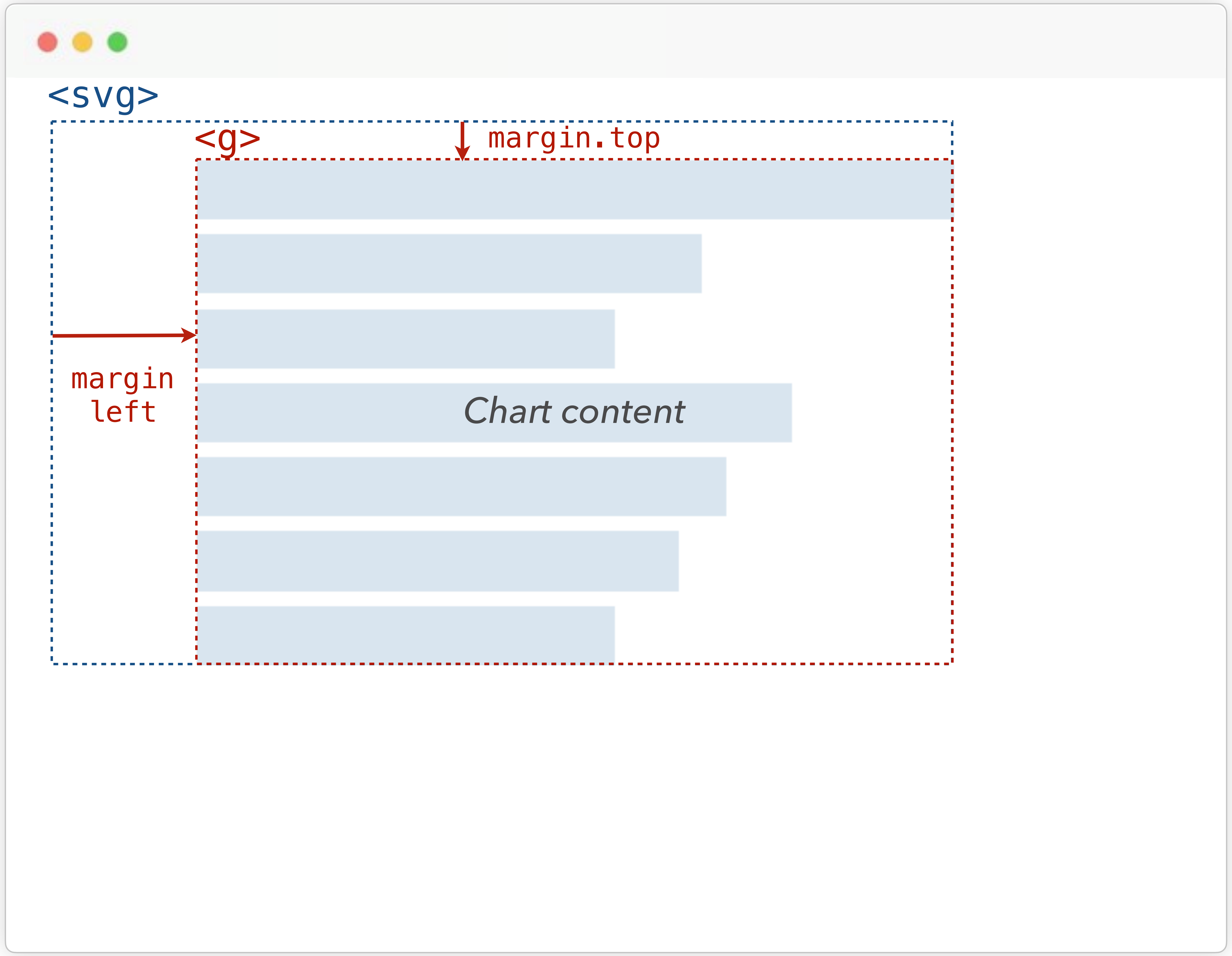
Axes



Axes



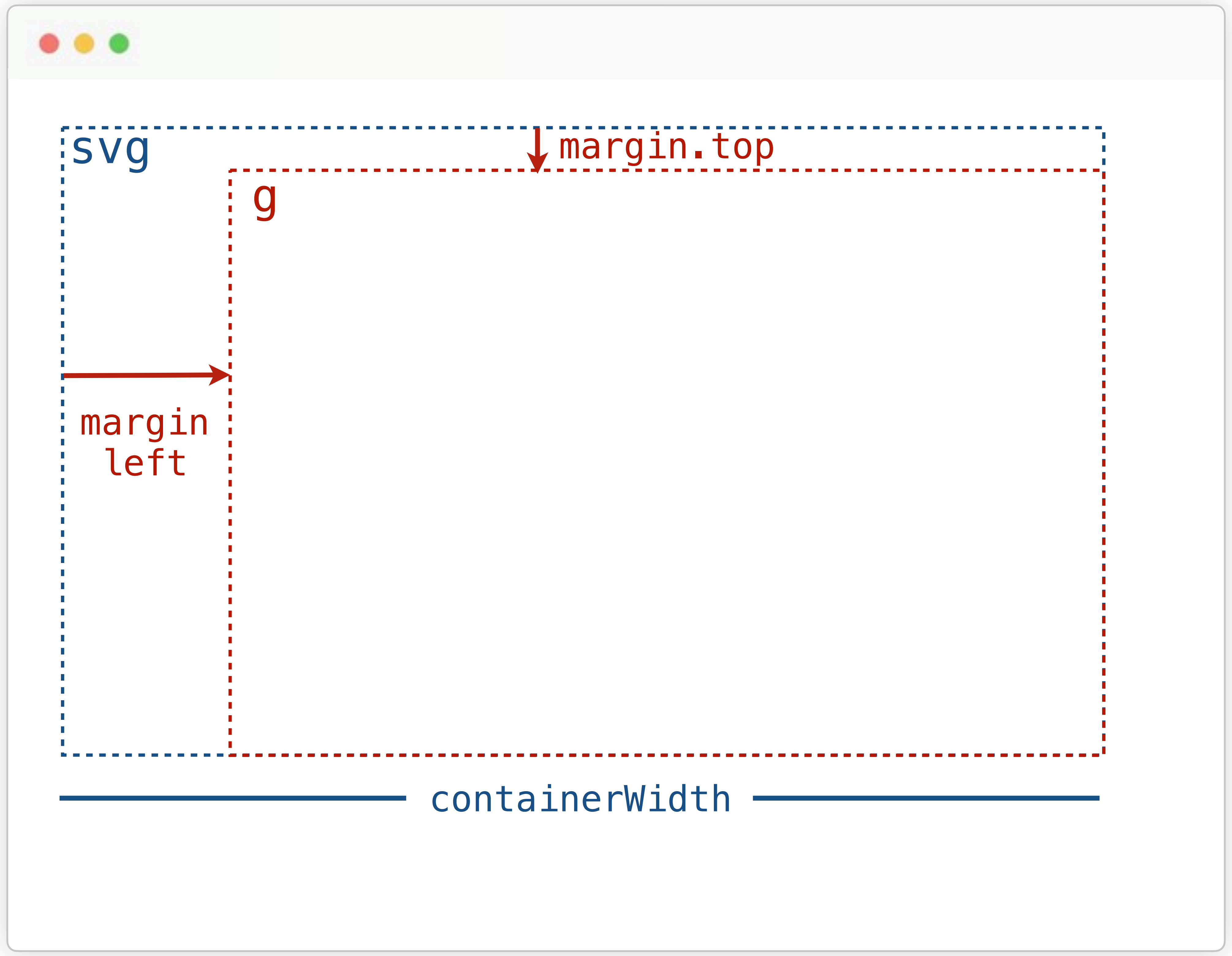
Axes



Axes

JS main.js

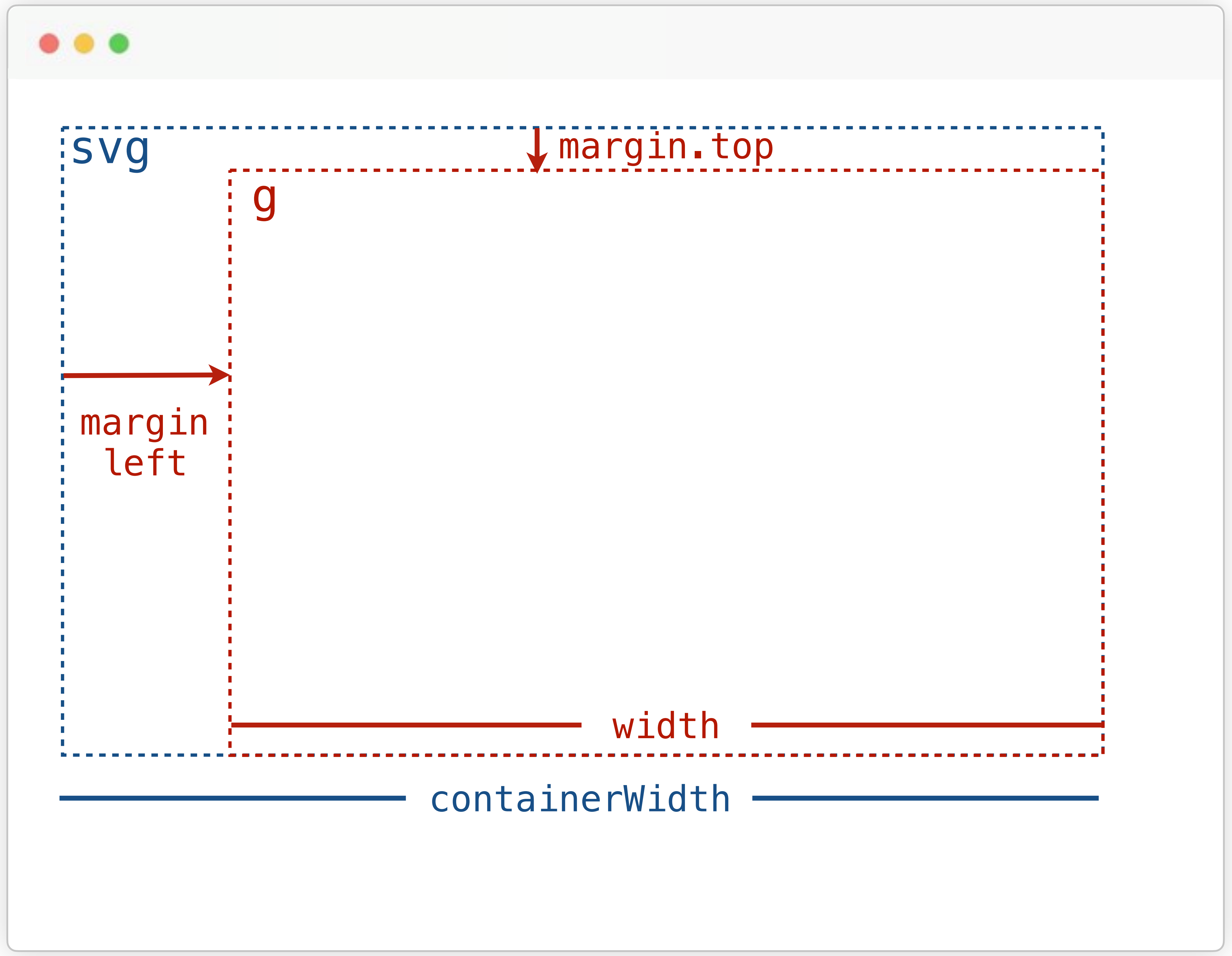
```
const containerWidth = 500;
const containerHeight = 300;

const margin = { top: 20, right: 0, bottom: 0, left: 80 };
```



Axes

JS main.js

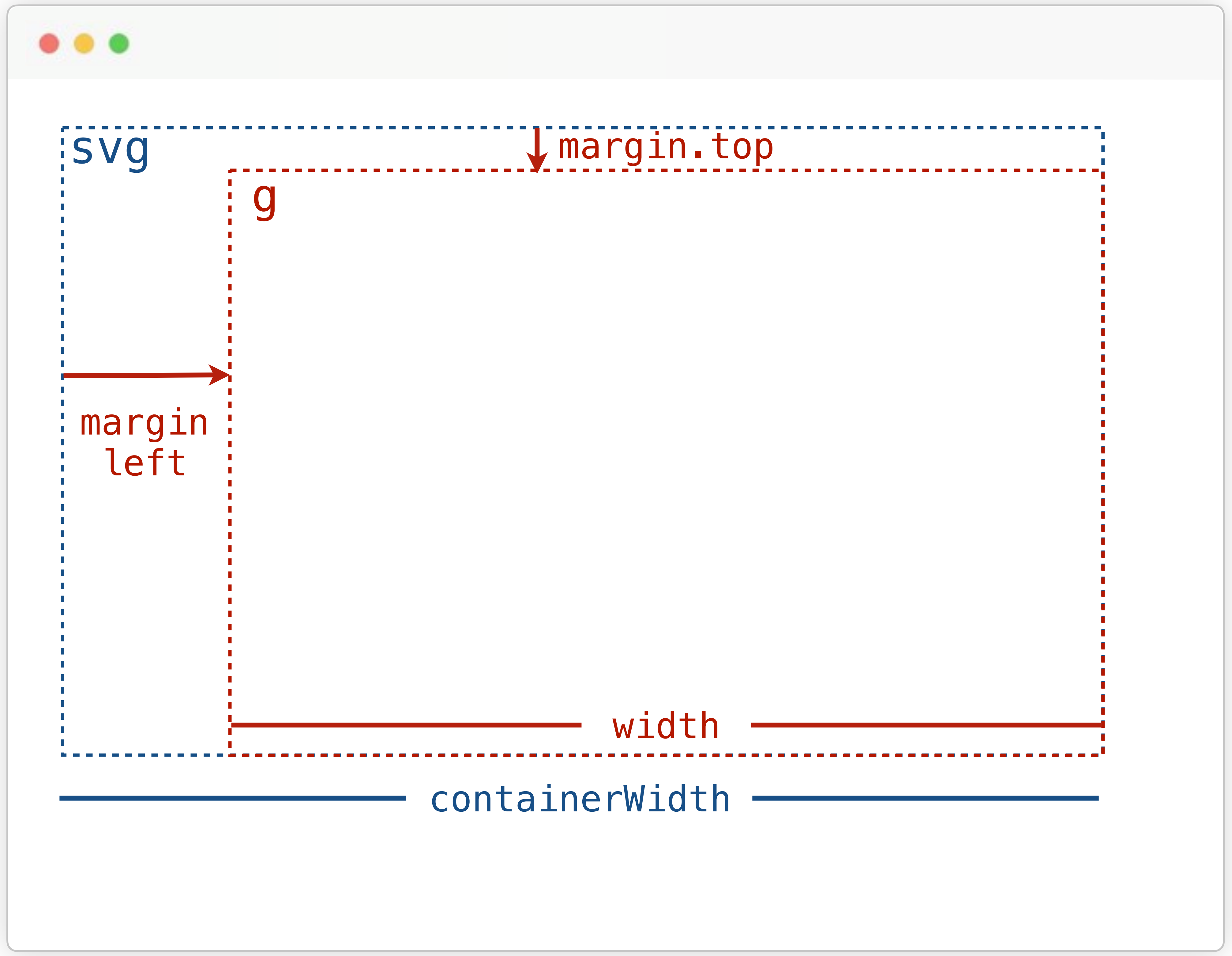
```
const containerWidth = 500;
const containerHeight = 300;

const margin = { top: 20, right: 0, bottom: 0, left: 80 };

const width = containerWidth - margin.left - margin.right;
const height = containerHeight - margin.top - margin.bottom;
```



Axes

JS main.js

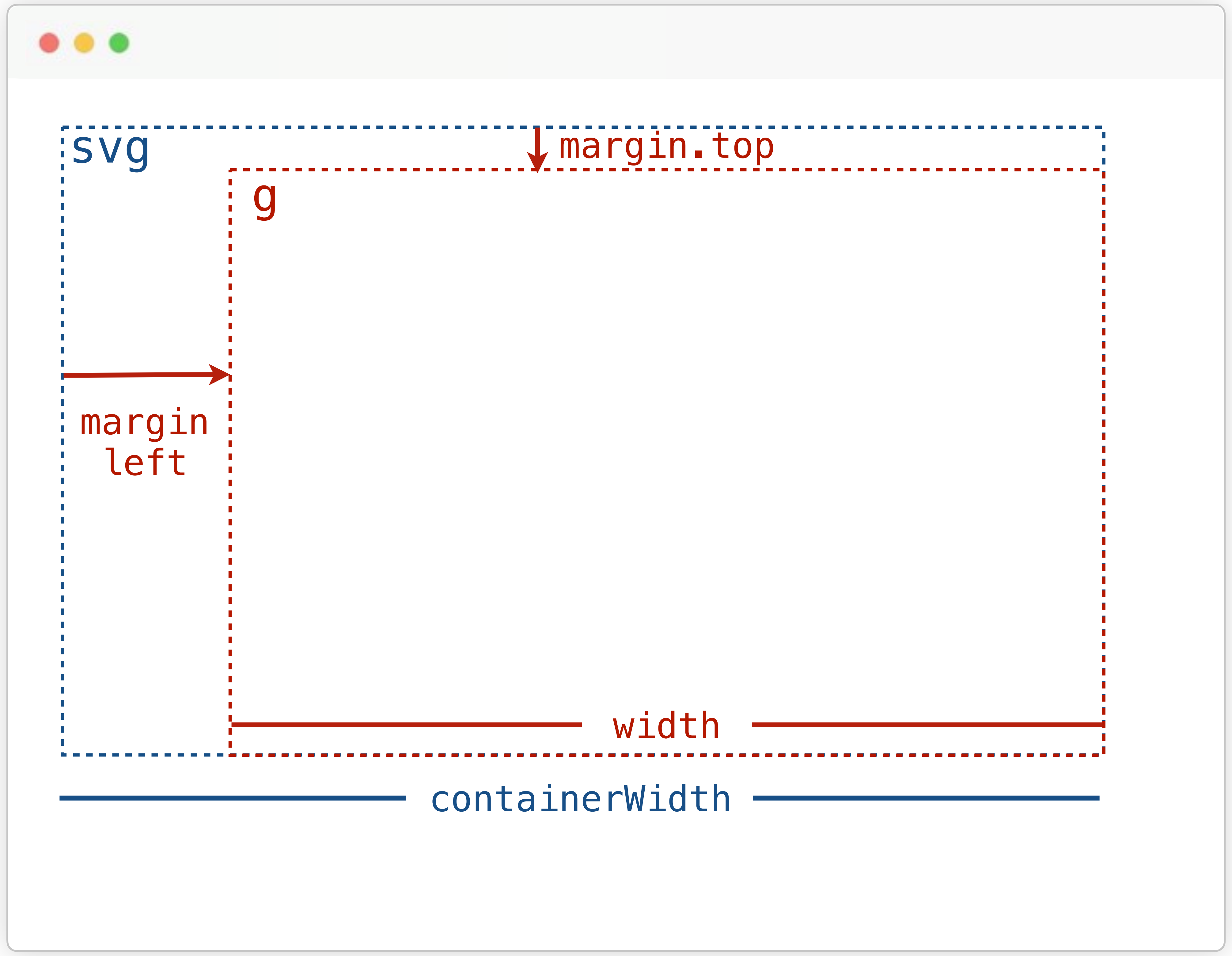
```
const containerWidth = 500;
const containerHeight = 300;

const margin = { top: 20, right: 0, bottom: 0, left: 80 };

const width = containerWidth - margin.left - margin.right;
const height = containerHeight - margin.top - margin.bottom;

const svg = d3.select('svg');

const chart = svg.append('g')
    .attr('transform', `translate(${margin.left},
                                  ${margin.top})`);
```



Axes

JS main.js

```
const containerWidth = 500;
const containerHeight = 300;

const margin = { top: 20, right: 0, bottom: 0, left: 80 };

const width = containerWidth - margin.left - margin.right;
const height = containerHeight - margin.top - margin.bottom;

const svg = d3.select('svg');

const chart = svg.append('g')
    .attr('transform', `translate(${margin.left},
                                  ${margin.top})`);
```



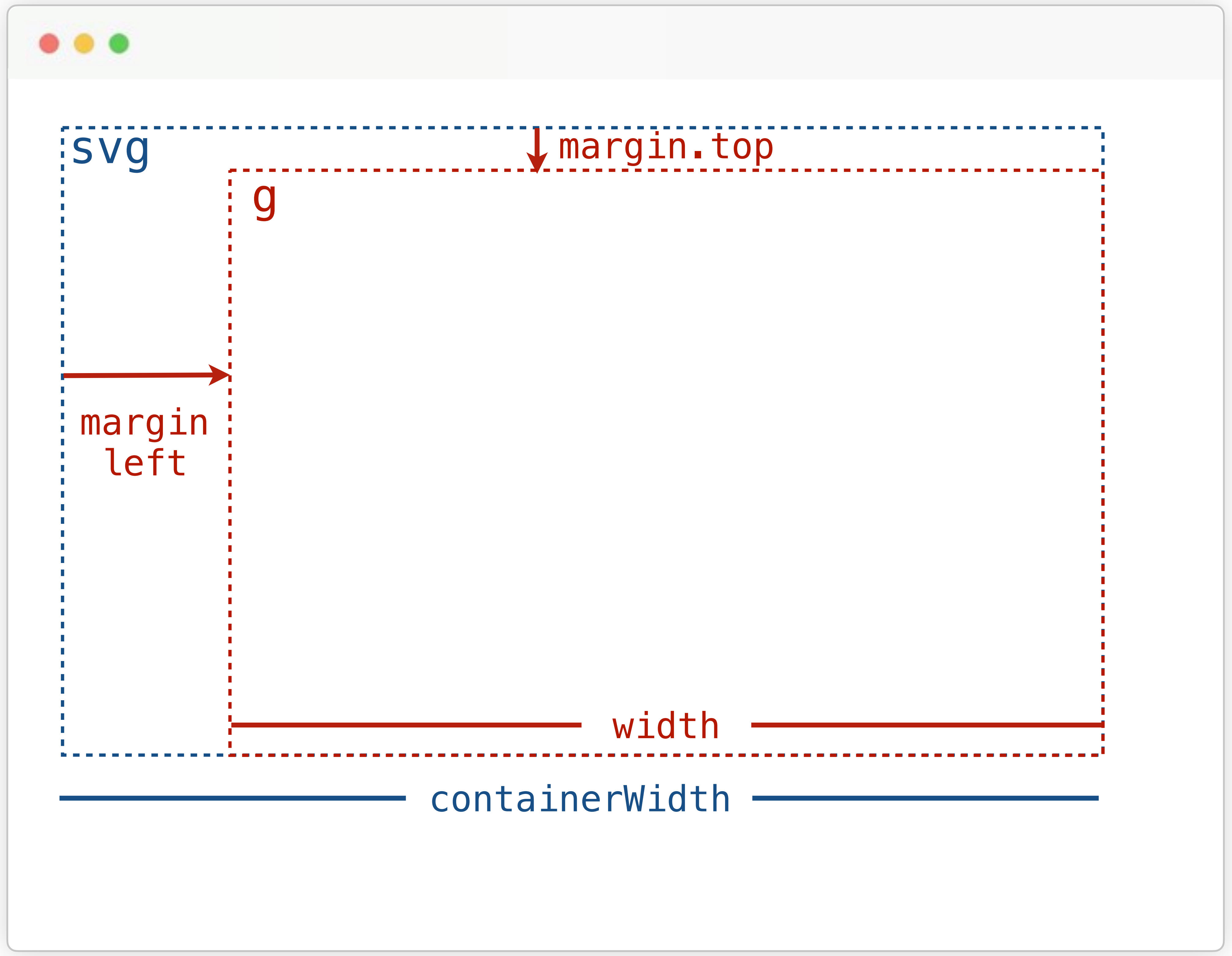
Template literals: `` `translate(${margin.left}, ${margin.top})` ``

Traditional version: `'translate(' + margin.left + ',' + margin.top + ')'`

Axes

JS main.js

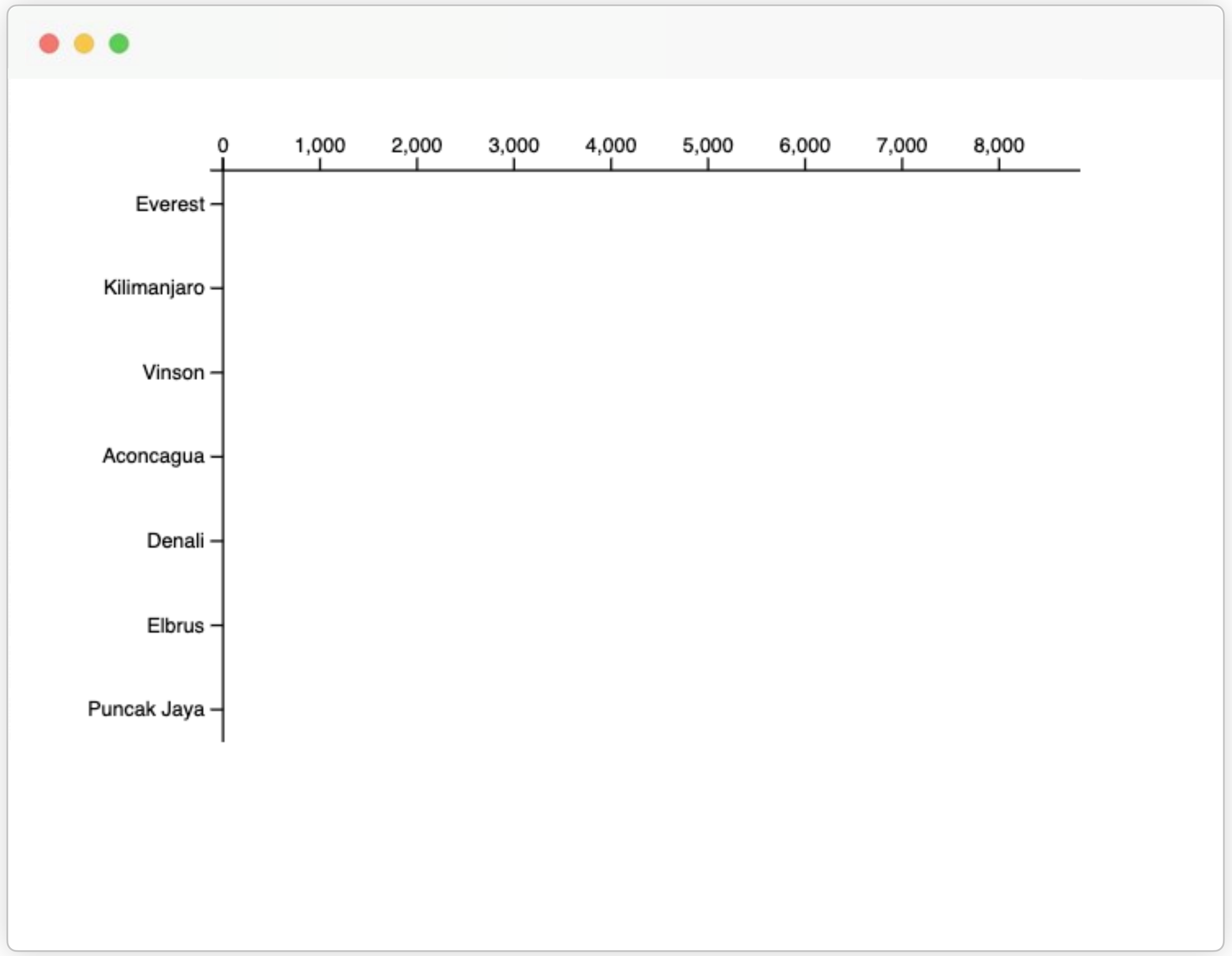
```
const containerWidth = 500;
const containerHeight = 300;

const margin = { top: 20, right: 0, bottom: 0, left: 80 };

const width = containerWidth - margin.left - margin.right;
const height = containerHeight - margin.top - margin.bottom;

const svg = d3.select('svg');

const chart = svg.append('g')
    .attr('transform', `translate(${margin.left},
                                  ${margin.top})`);
```



Axes

JS main.js

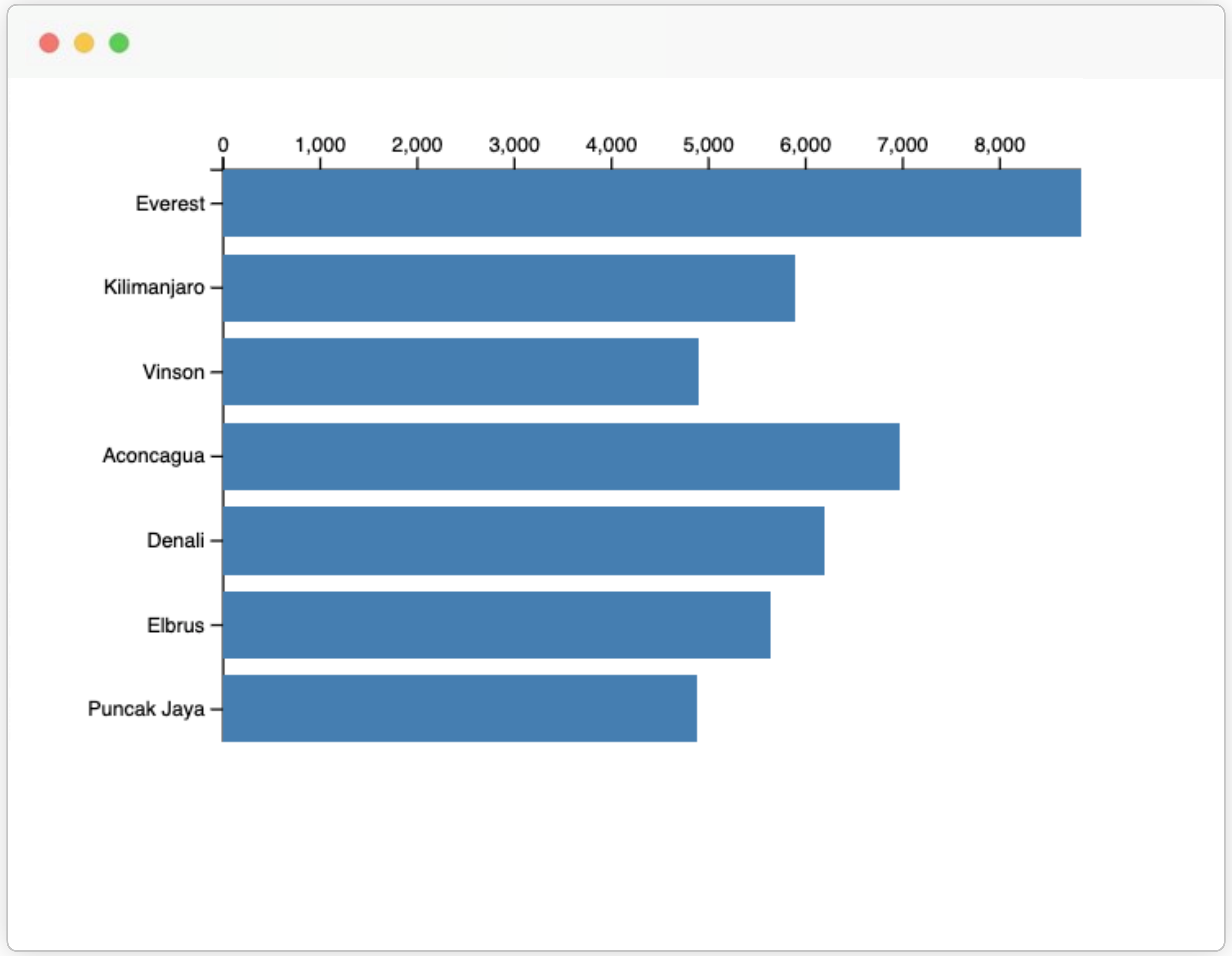
```
const chart = svg.append('g')
    .attr('transform', `translate(${margin.left},
                                  ${margin.top})`);

// Initialize axes
const xAxis = d3.axisTop(xScale);
const yAxis = d3.axisLeft(yScale);

// Draw the axis
const xAxisGroup = chart.append('g')
    .call(xAxis);

const yAxisGroup = chart.append('g')
    .call(yAxis);
```



Axes

JS main.js

```
const chart = svg.append('g')
    .attr('transform', `translate(${margin.left},
                                  ${margin.top})`);

// Initialize axes
const xAxis = d3.axisTop(xScale);
const yAxis = d3.axisLeft(yScale);

// Draw the axis
const xAxisGroup = chart.append('g')
    .call(xAxis);

const yAxisGroup = chart.append('g')
    .call(yAxis);

chart.selectAll('rect')
    .data(data)
    .join('rect')
      .attr('class', 'bar')
      .attr('width', (d) => xScale(d.elevation))
      .attr('height', yScale.bandwidth())
      .attr('y', (d) => yScale(d.title));
```



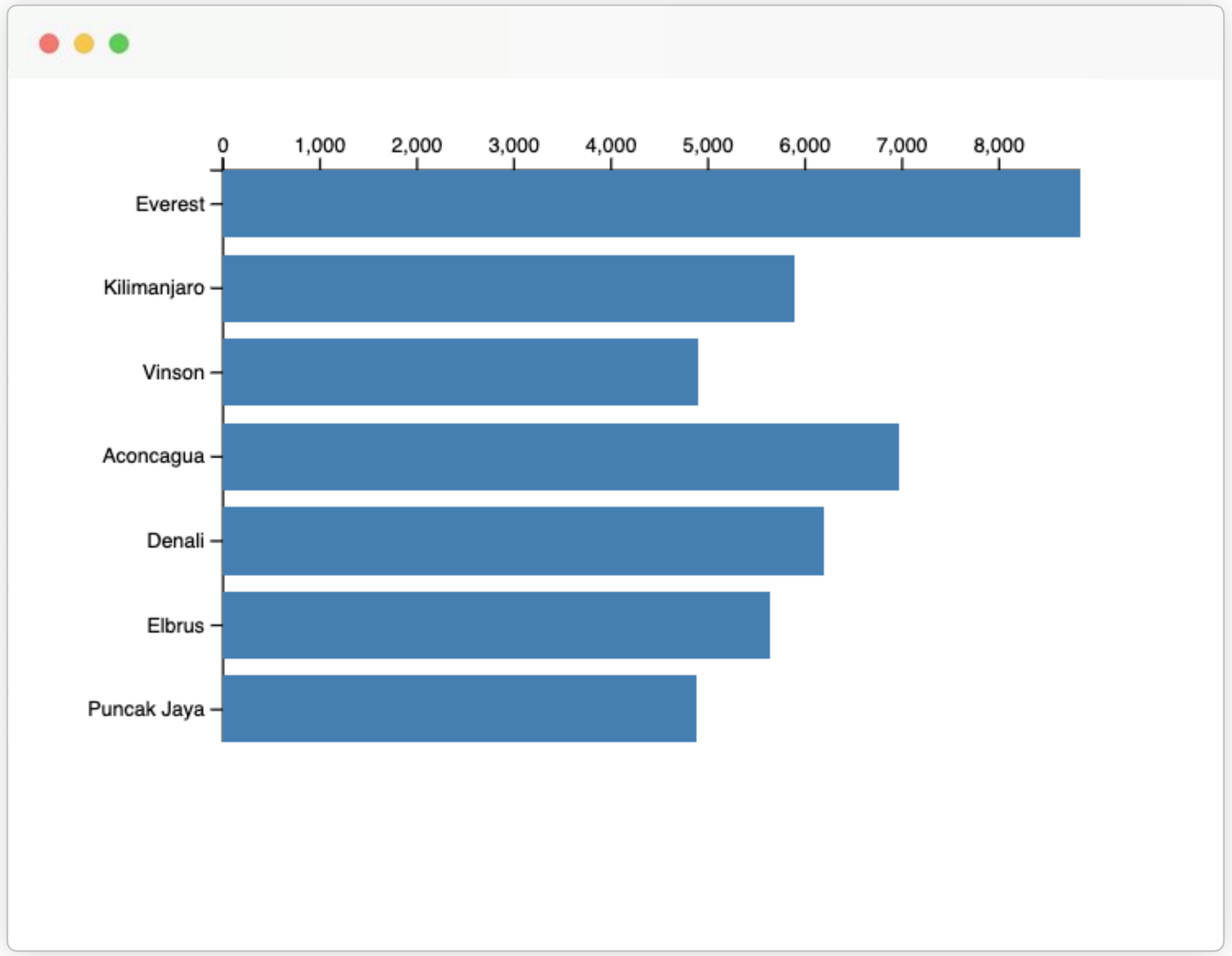
Dimensions/ layout

Scales

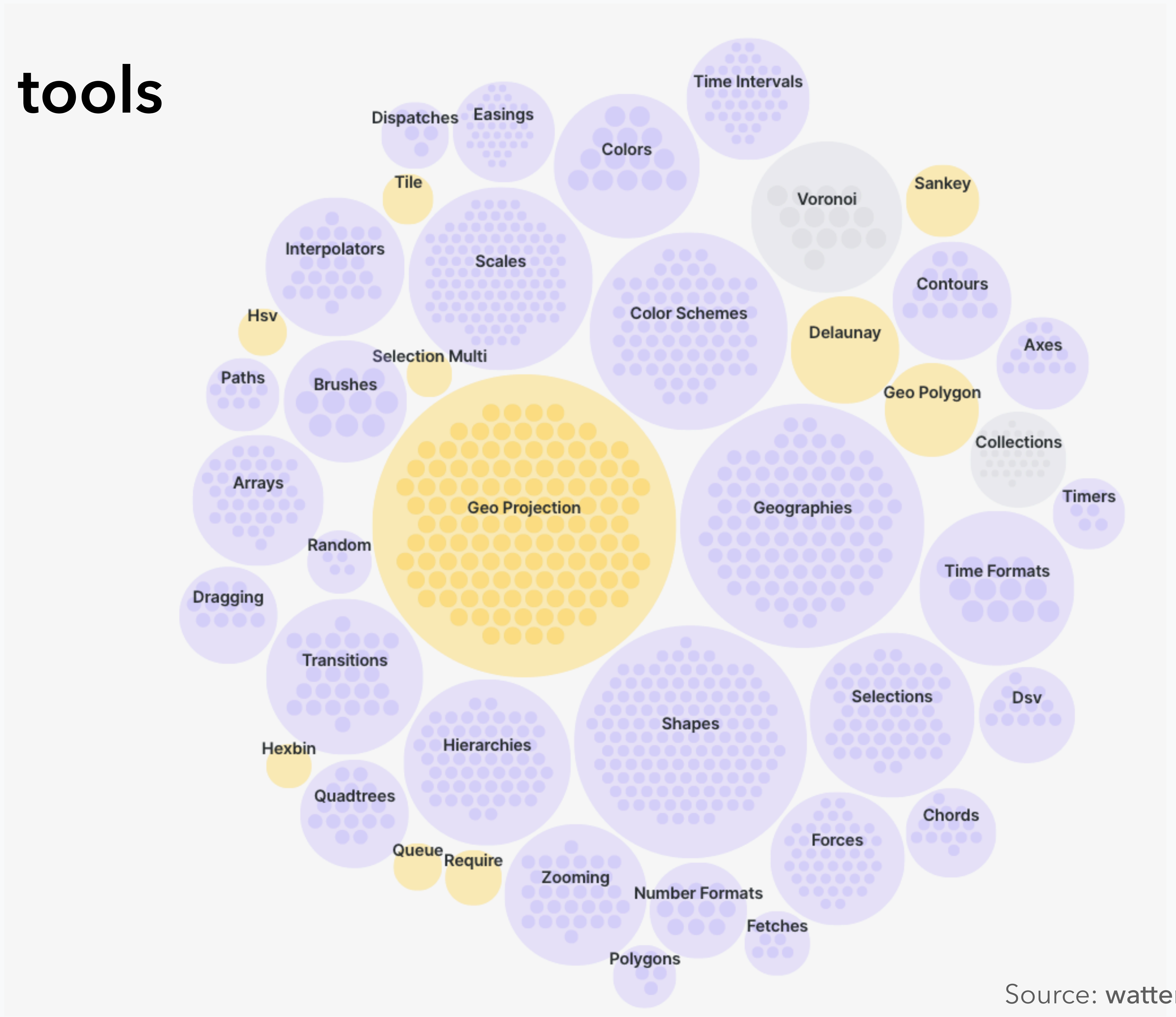
Axes

Draw marks

```
function drawChart(data) {
  const containerWidth = 500;
  const containerHeight = 300;
  const margin = { top: 20, right: 0, bottom: 0, left: 80 };

  const width = containerWidth - margin.left - margin.right;
  const height = containerHeight - margin.top - margin.bottom;

  const chart = d3.select('svg').append('g')
      .attr('transform', `translate(${margin.left}, ${margin.top})`);

  const max = d3.max(data, d => d.elevation);

  const xScale = d3.scaleLinear()
      .range([0, width])
      .domain([0, max]);

  const yScale = d3.scaleBand()
      .range([0, height])
      .domain(data.map((d) => d.title ))
      .paddingInner(0.2);

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

  const xAxisGroup = chart.append('g')
      .call(xAxis);

  const yAxisGroup = chart.append('g')
      .call(yAxis);

  chart.selectAll('rect')
      .data(data)
      .join('rect')
        .attr('class', 'bar')
        .attr('width', (d) => xScale(d.elevation))
        .attr('height', yScale.bandwidth())
        .attr('y', (d) => yScale(d.title));
}
```



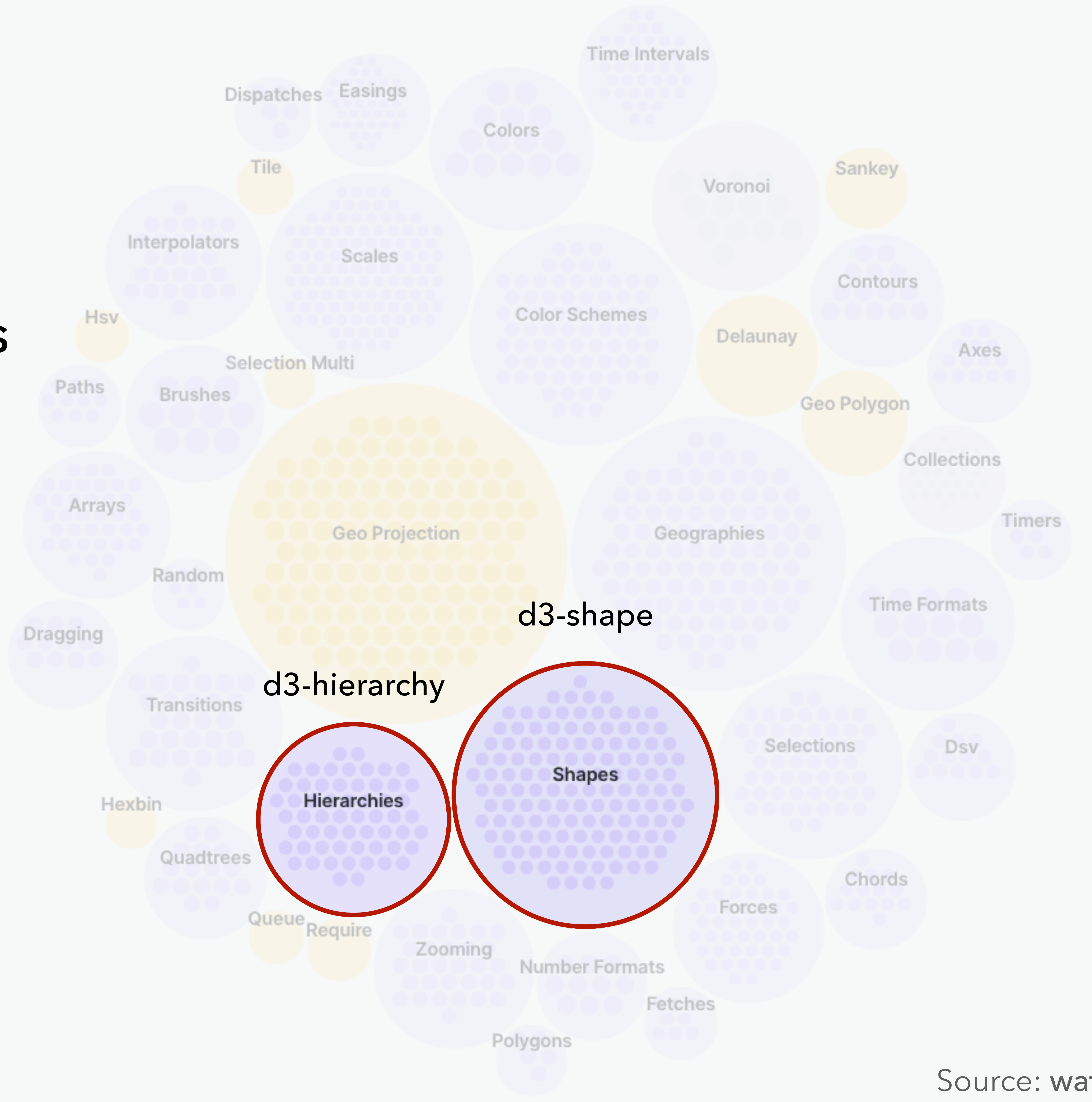
Other D3 tools



Source: wattenberger.com/blog/d3

Other D3 tools

- Shape generators
- Layout generators

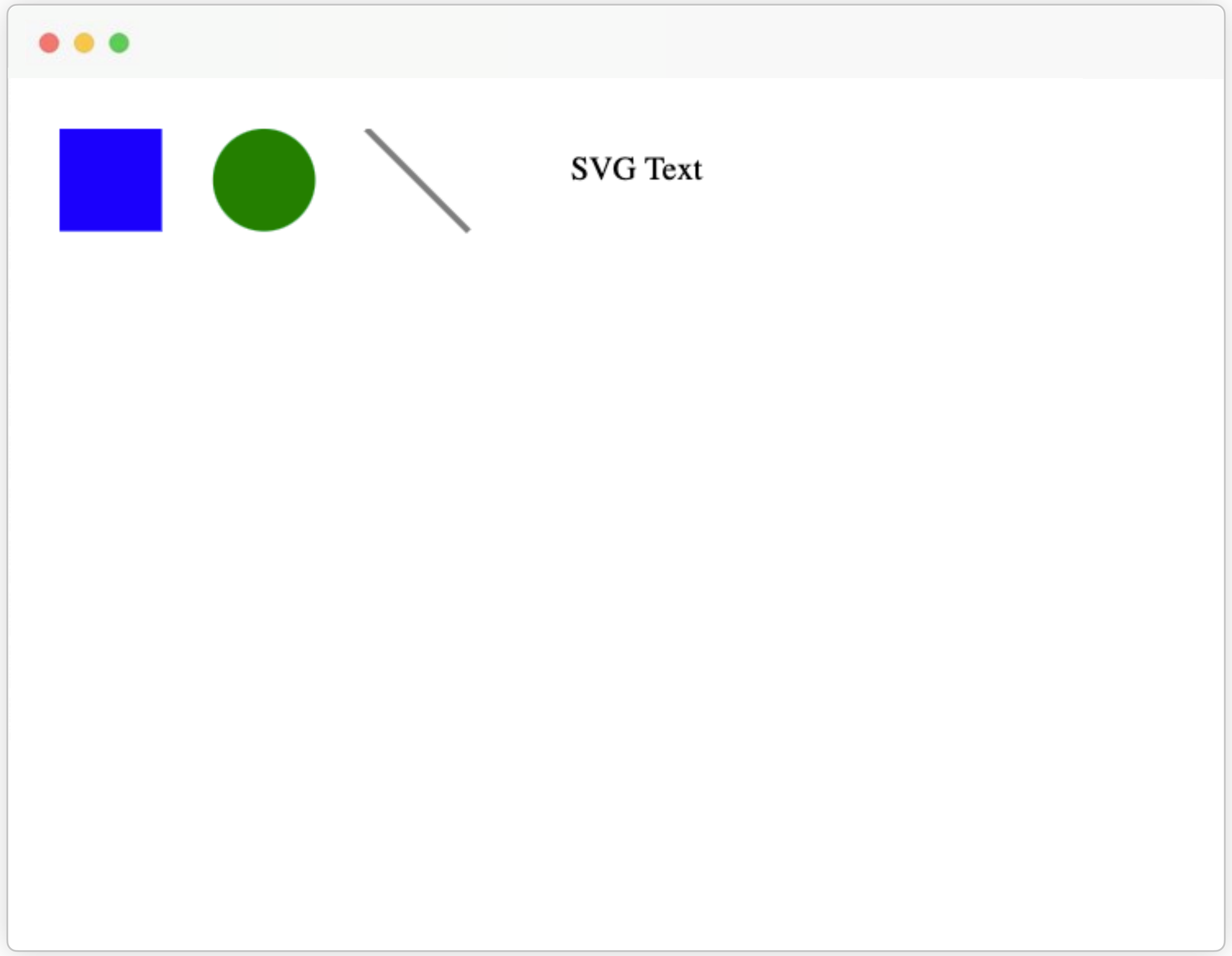


Source: wattenberger.com/blog/d3

SVG shapes

HTML

```
<rect width="50" height="50" fill="blue" />

<circle cx="100" cy="25" r="25" fill="green" />

<line x1="150" y1="0" x2="200" y2="50" stroke="gray"
  stroke-width="3" />

<text x="250" y="25">SVG Text</text>
```

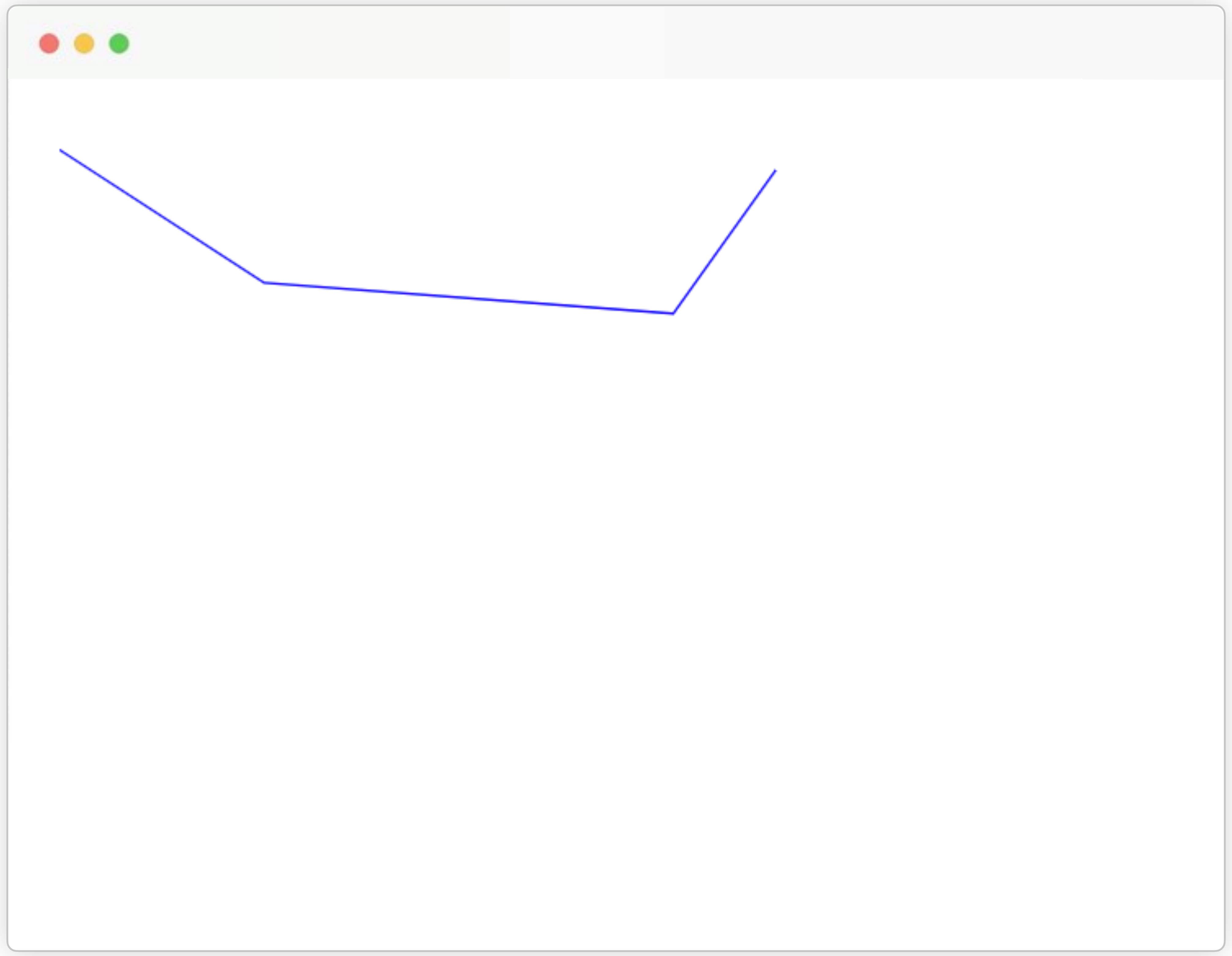


SVG shapes

HTML

```
<path
    style="fill: none; stroke: blue"
    d="M0 10 L100 75 L300 90 L350 20"
/>
```

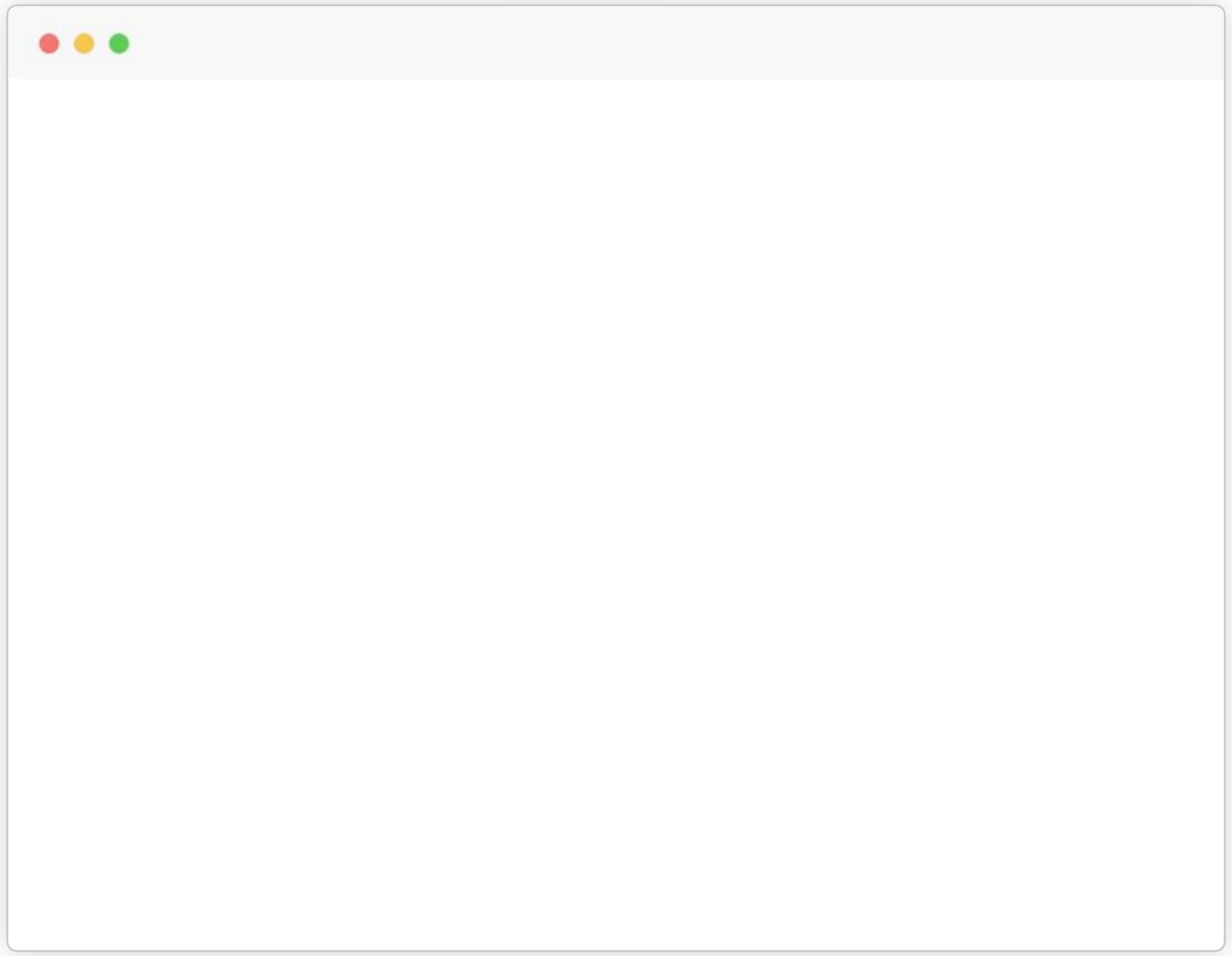
Complex path instructions



D3 shape generators

JS main.js

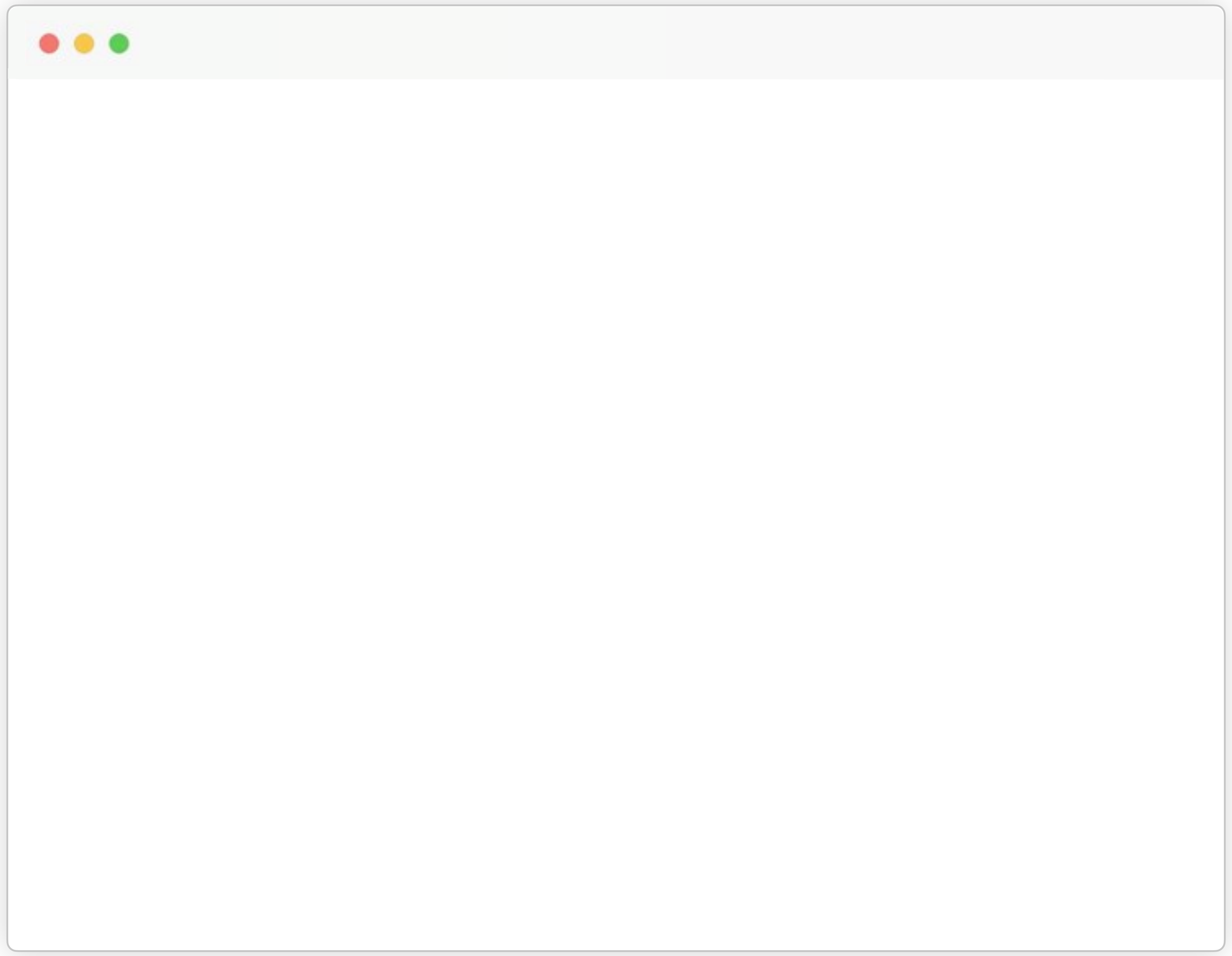
```
const data = [
  {x: 0, y: 10},
  {x: 100, y: 75},
  {x: 300, y: 90},
  {x: 350, y: 20}
];
```



D3 shape generators

JS main.js

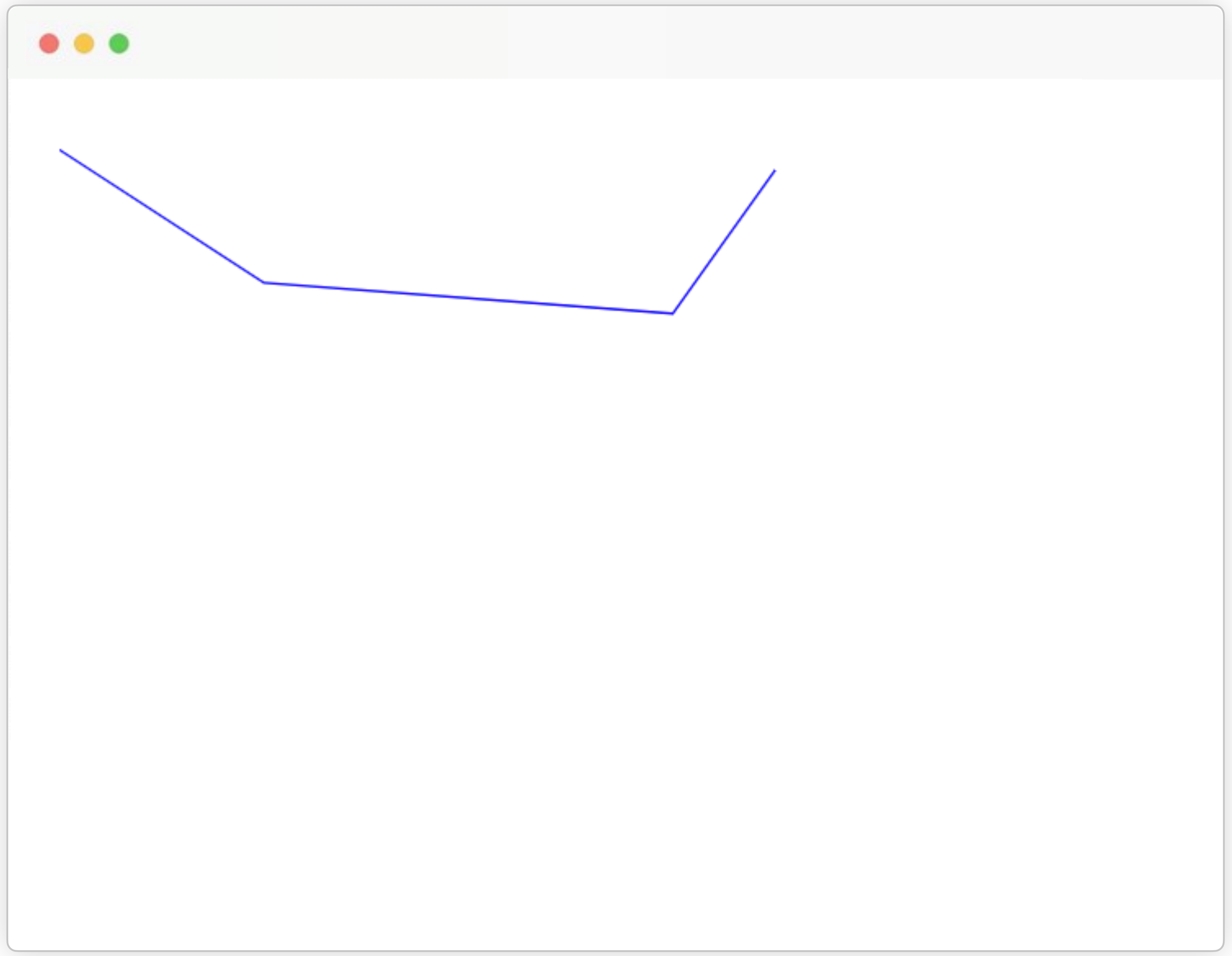
```
const data = [
  {x: 0, y: 10},
  {x: 100, y: 75},
  {x: 300, y: 90},
  {x: 350, y: 20}
];

// Initialize the shape generator
const line = d3.line()
    .x(d => d.x)
    .y(d => d.y);
```



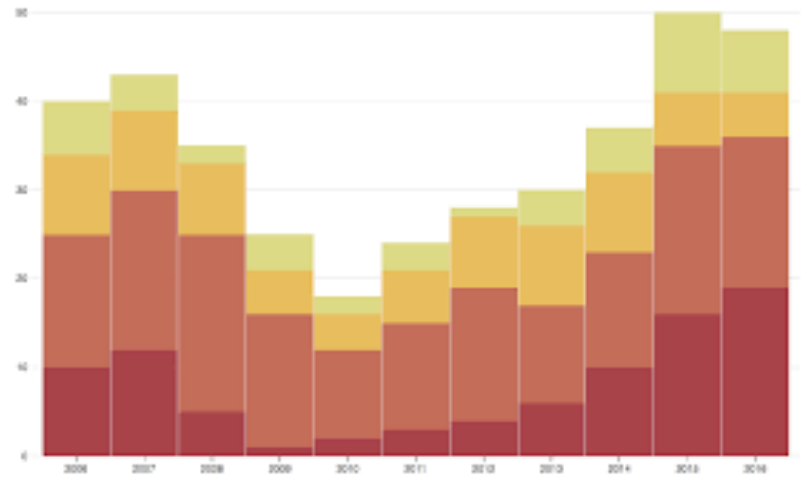
D3 shape generators

JS main.js

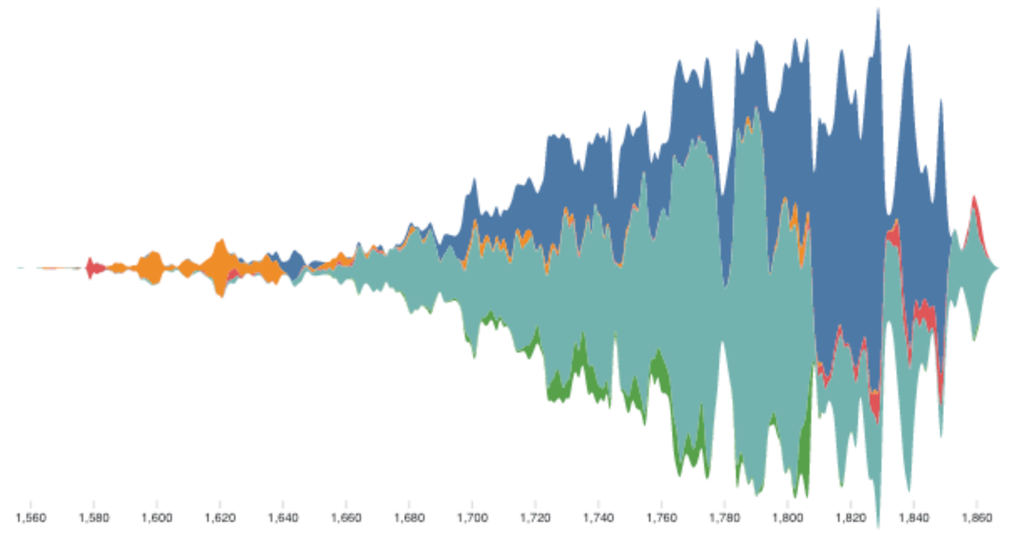
```
const data = [
  {x: 0, y: 10},
  {x: 100, y: 75},
  {x: 300, y: 90},
  {x: 350, y: 20}
];

// Initialize the shape generator
const line = d3.line()
    .x(d => d.x)
    .y(d => d.y);

// Add the <path> to the <svg> container
d3.select('svg').append('path')
    .attr('d', line(data))
    .attr('stroke', 'red')
    .attr('fill', 'none');
```



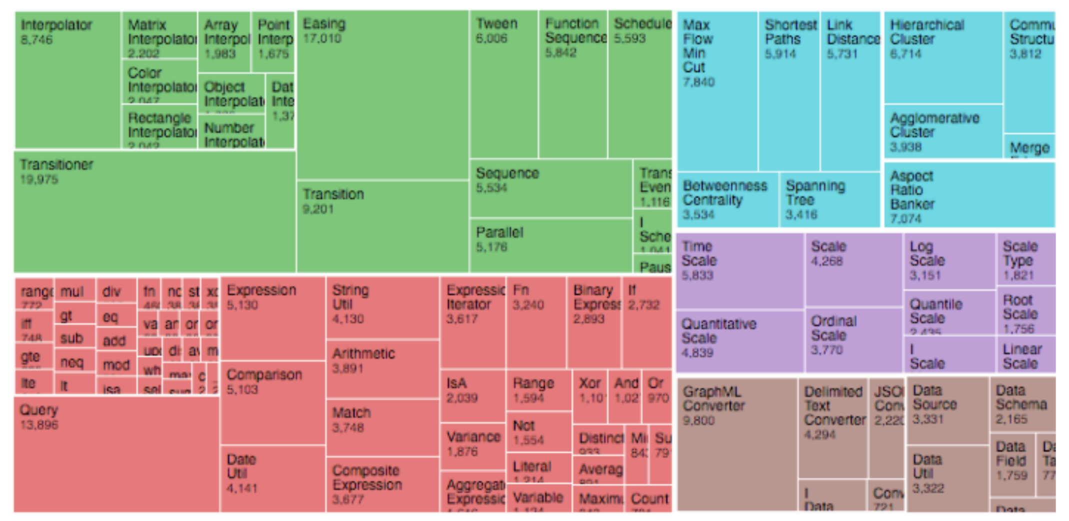
Stacked bar chart



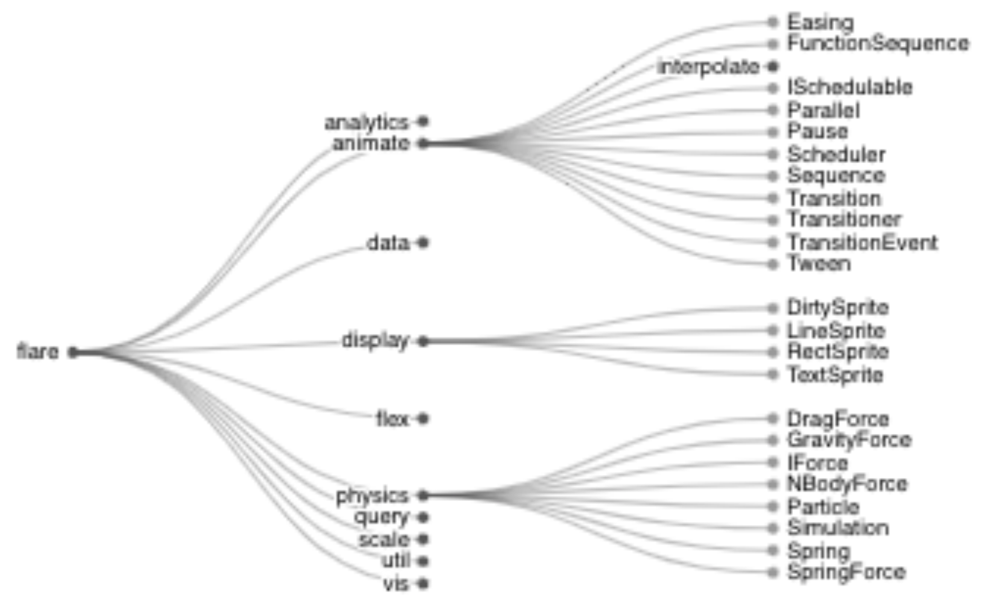
Streamgraph



Treemap

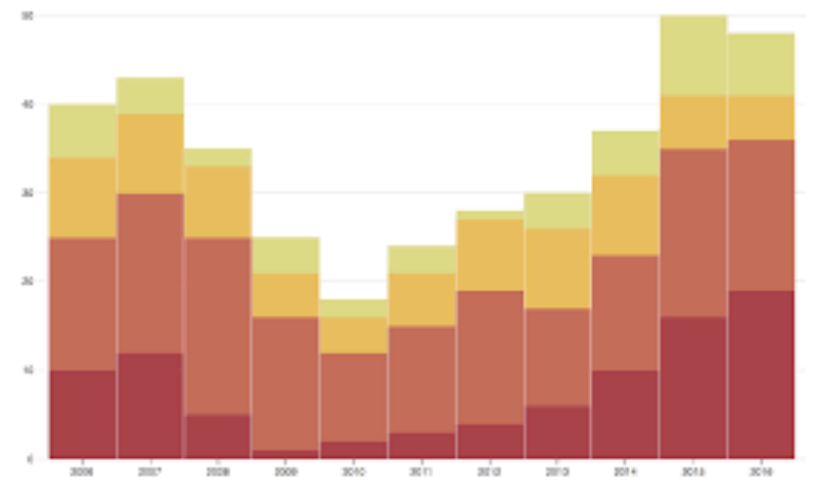


Tidy tree



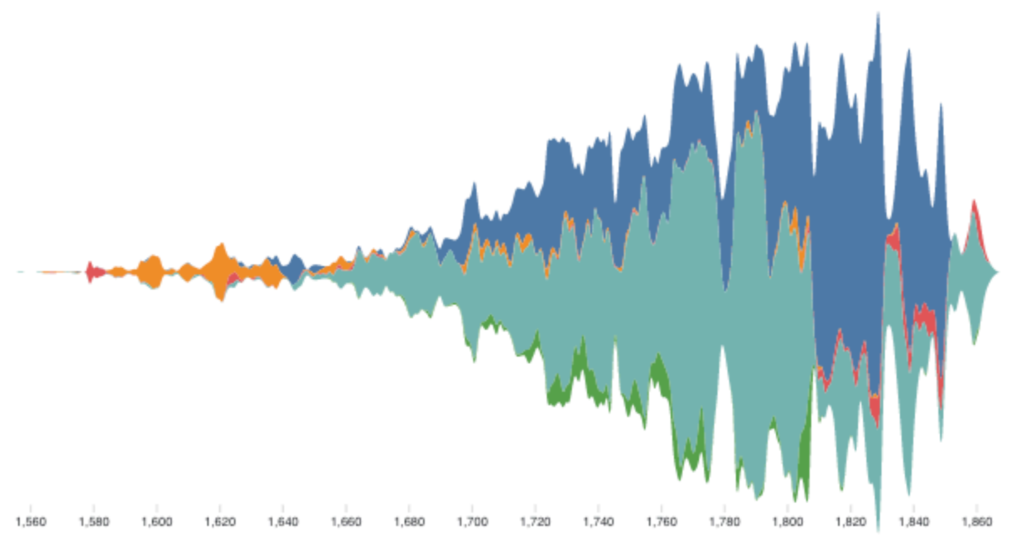
D3 layout generators

Stacked bar chart

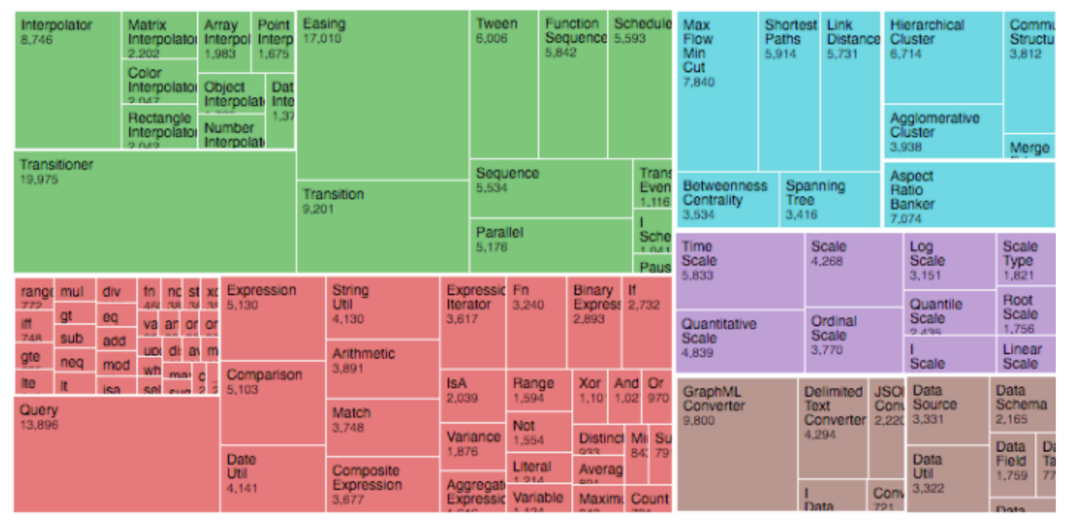


`d3.stack()`

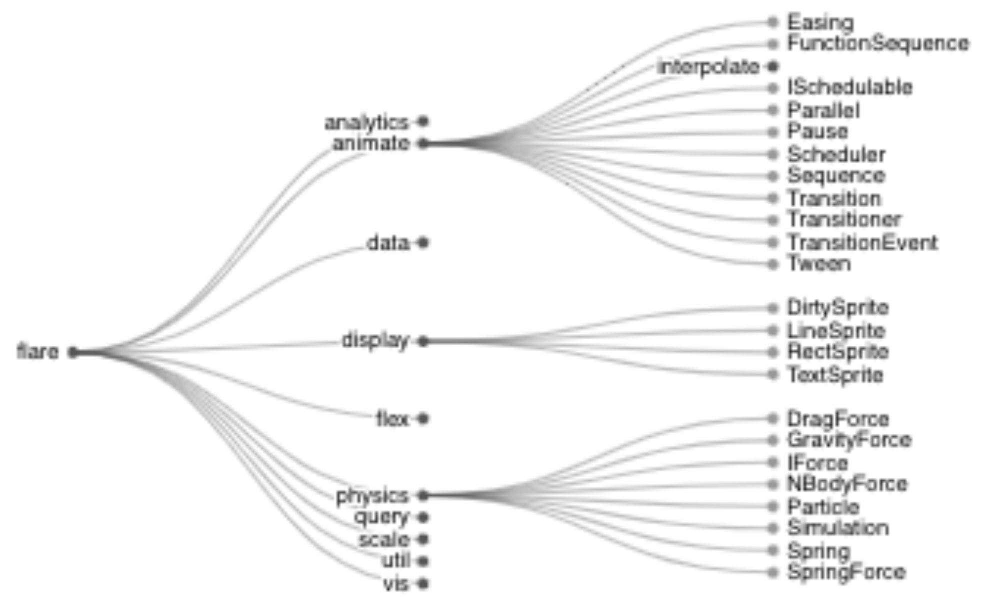
Streamgraph



Treemap

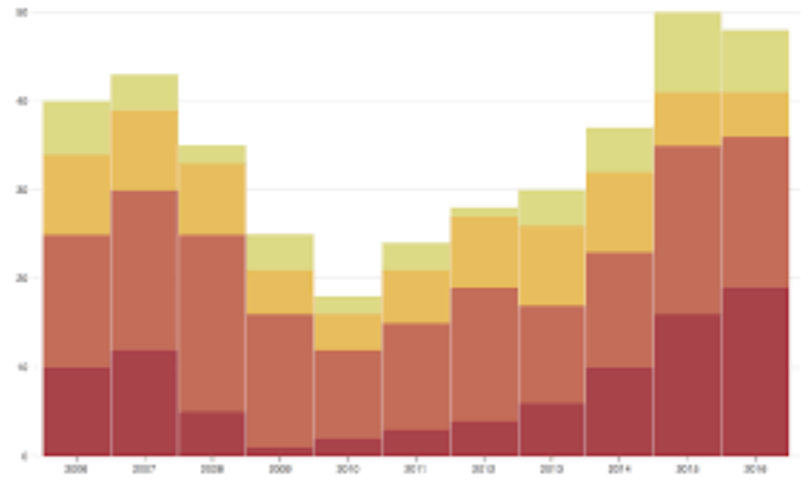


Tidy tree



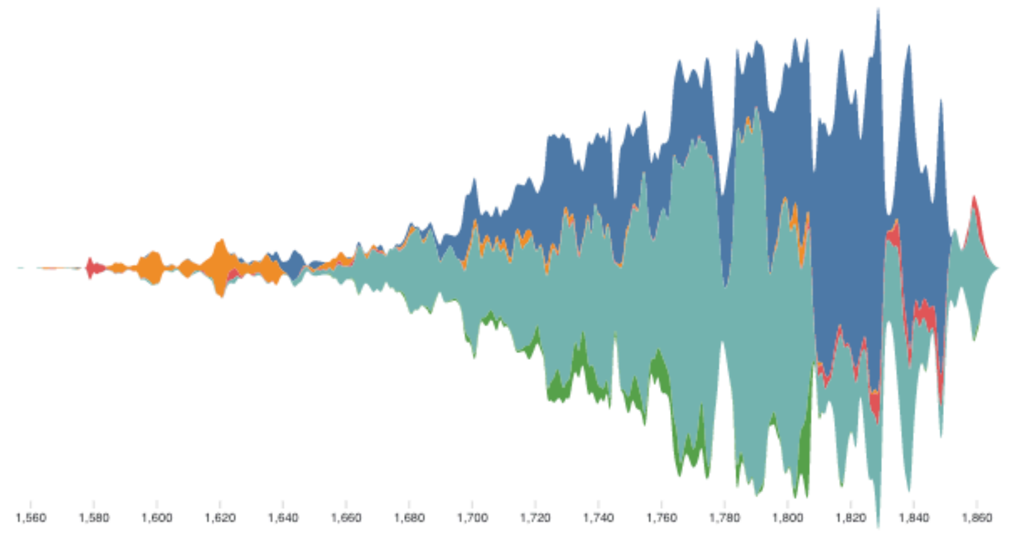
D3 layout generators

Stacked bar chart



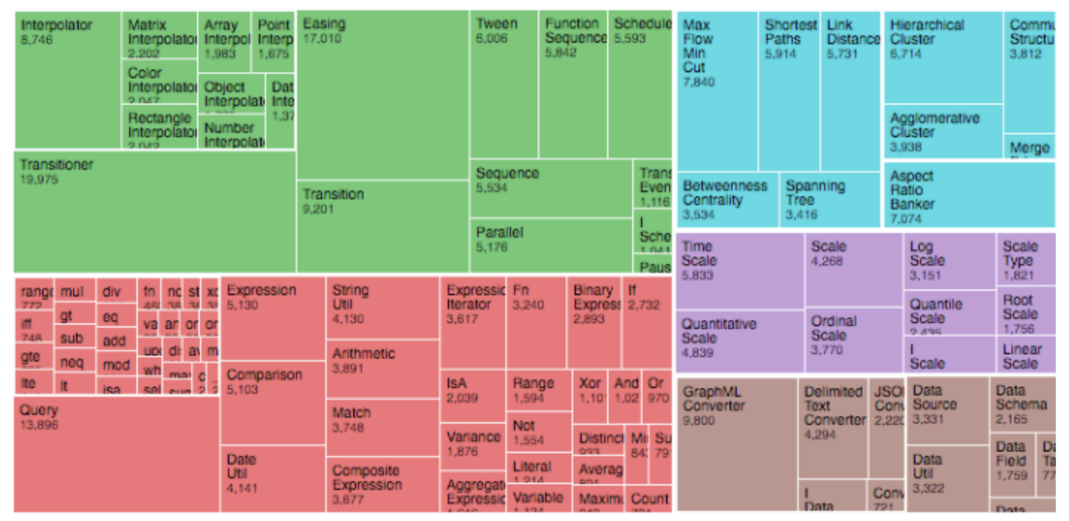
`d3.stack()`

Streamgraph



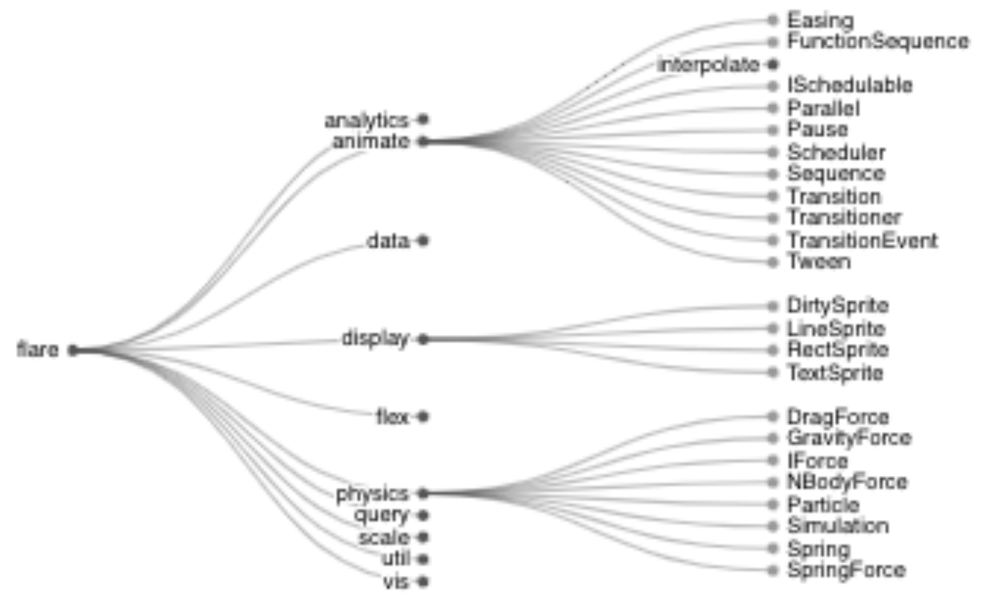
`d3.stack()` **and** `d3.area()`

Treemap



`d3.hierarchy()` **and** `d3.treemap()`

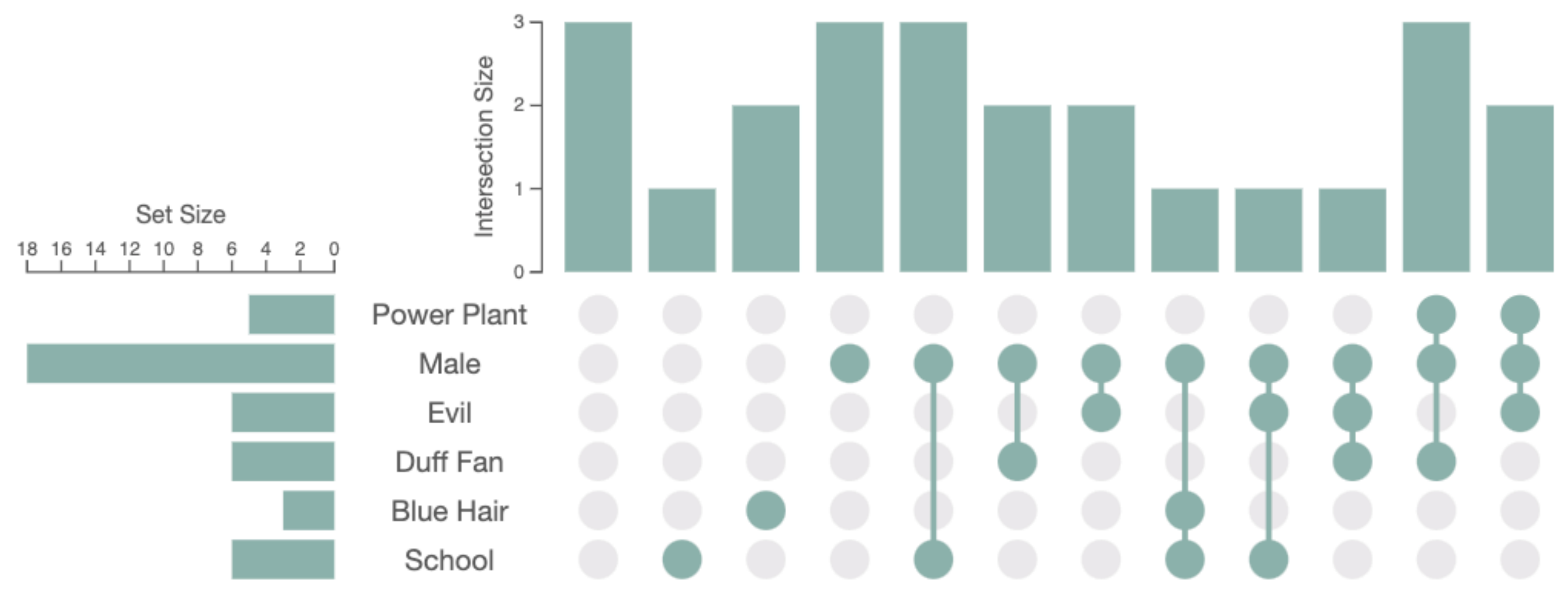
Tidy tree



`d3.hierarchy()` **and** `d3.tree()`

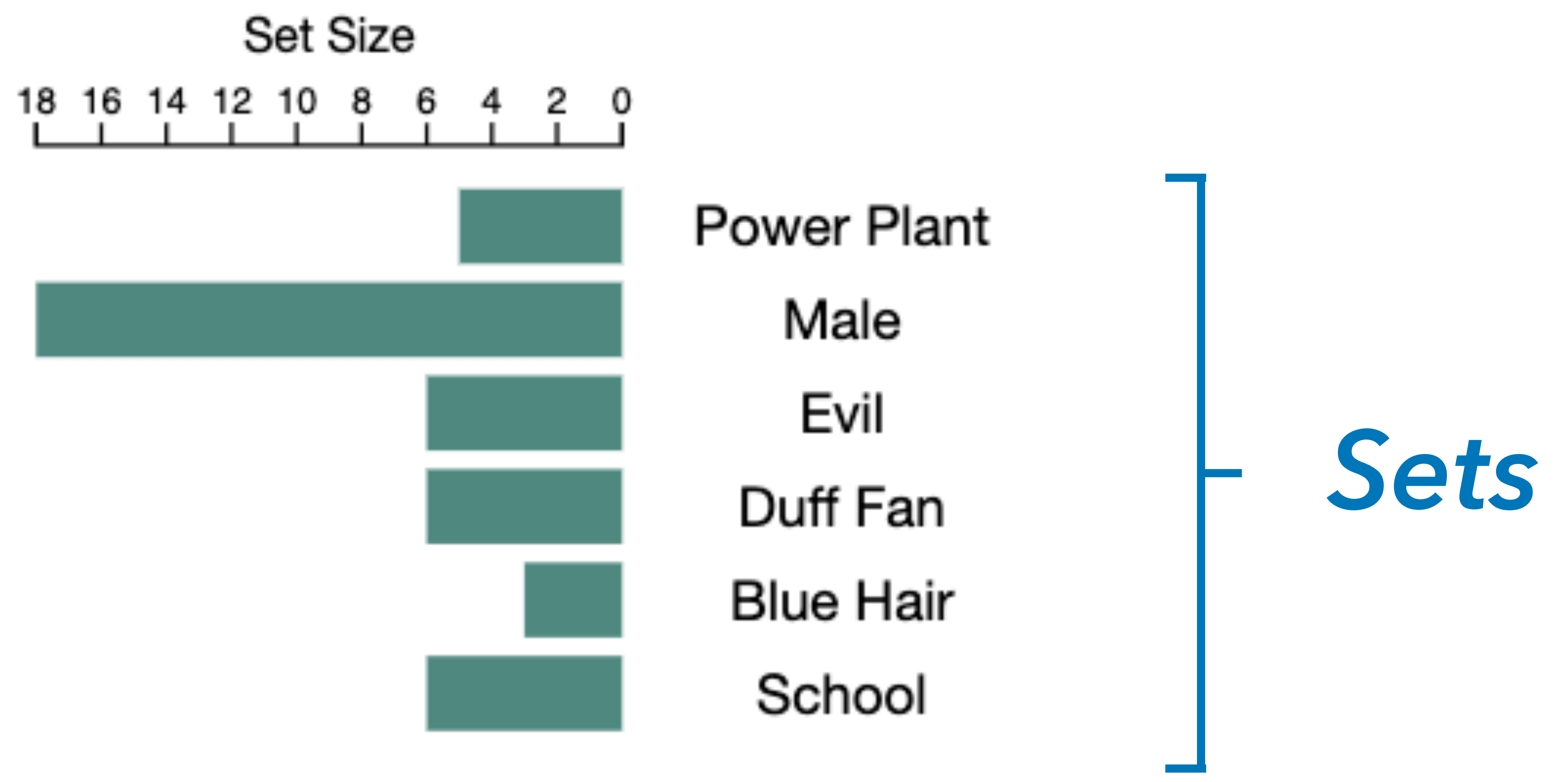
UpSet plot

visualizing intersections of multiple sets

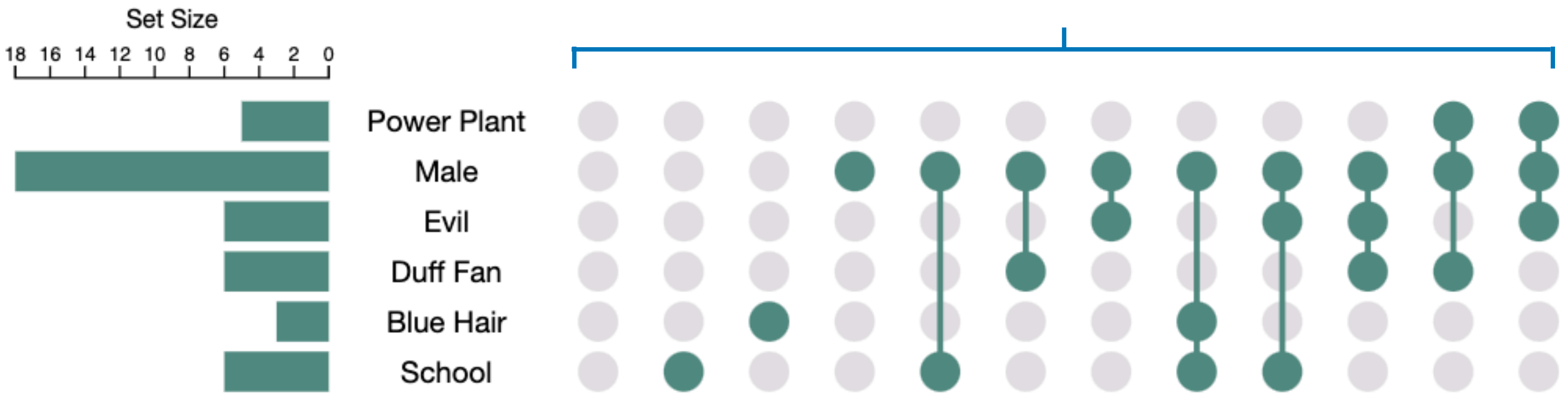


See vcg.github.io/upset/

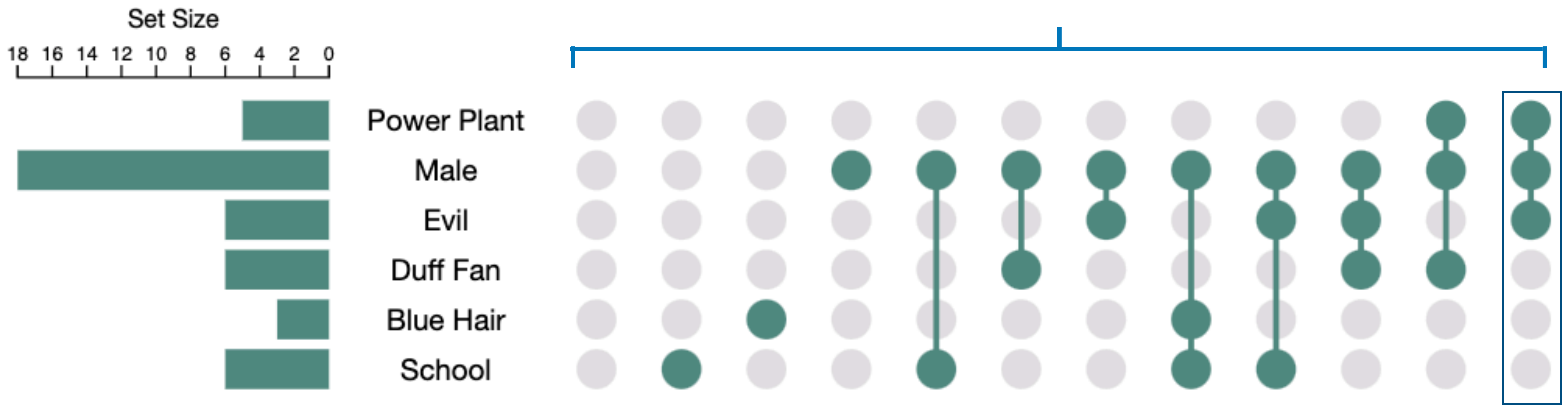
Sets of Simpsons characters



Set intersections

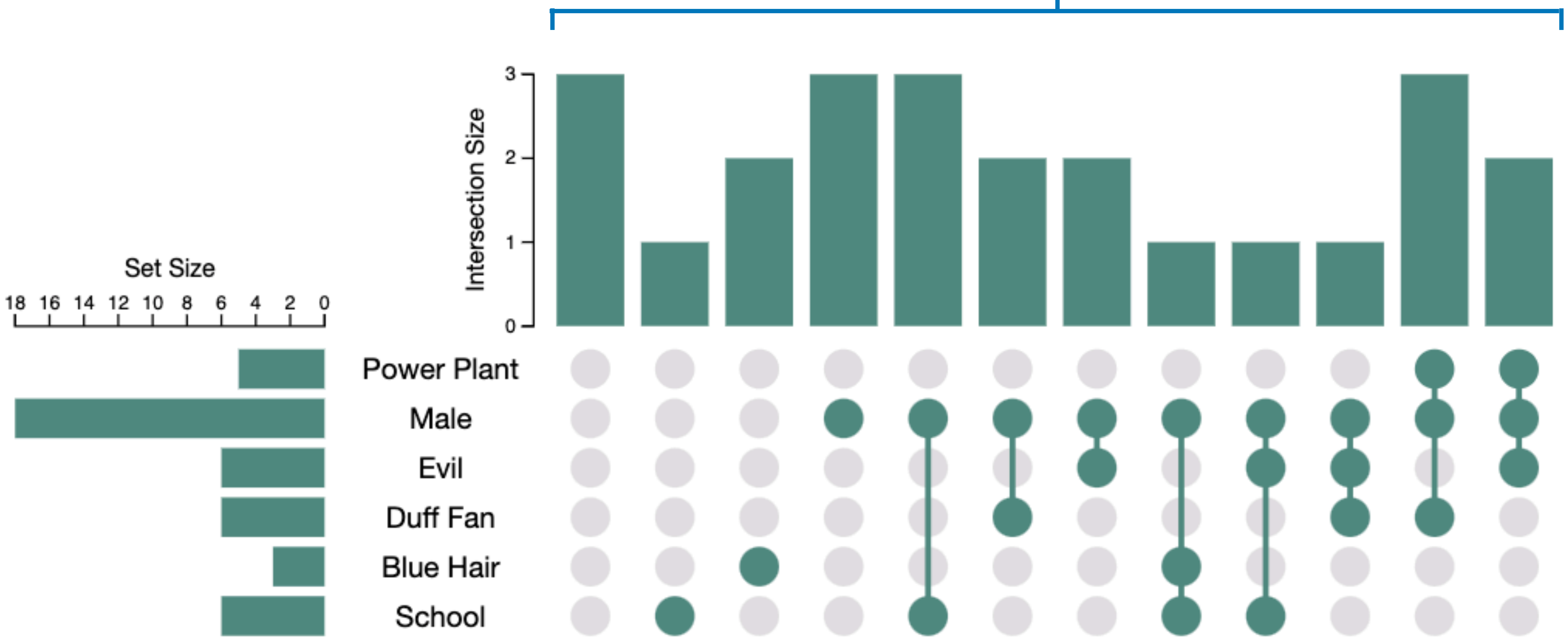


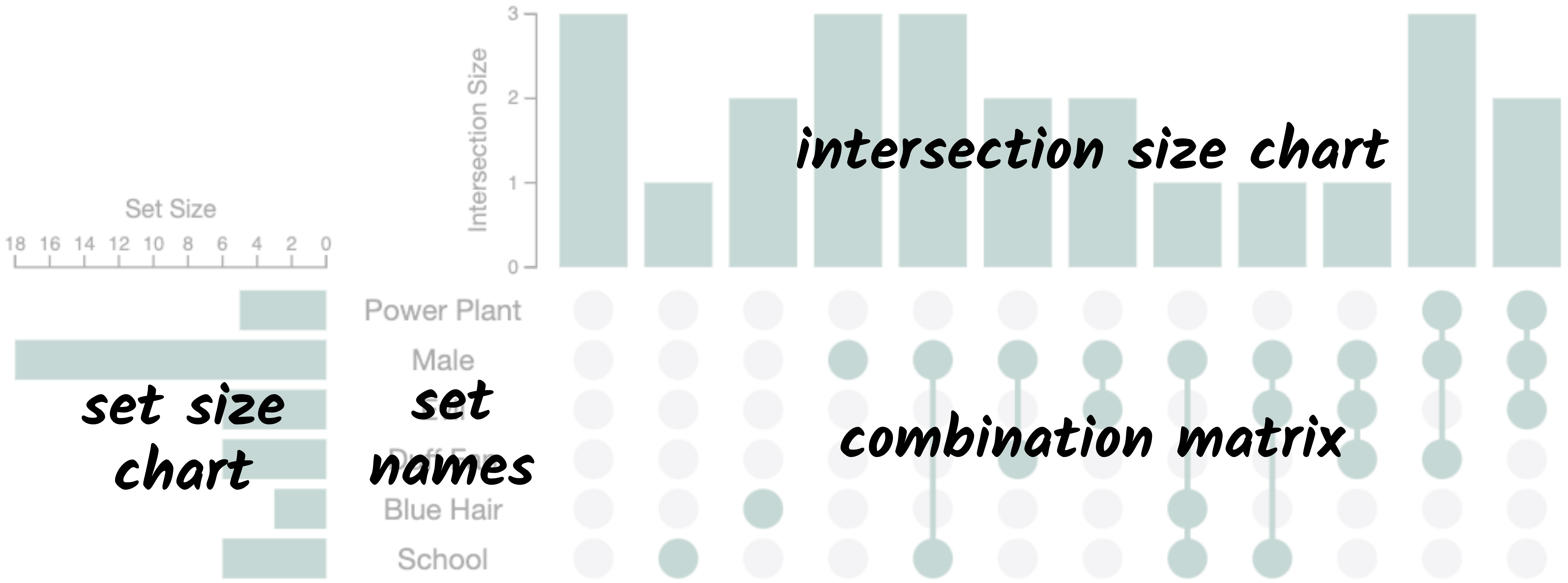
Set intersections



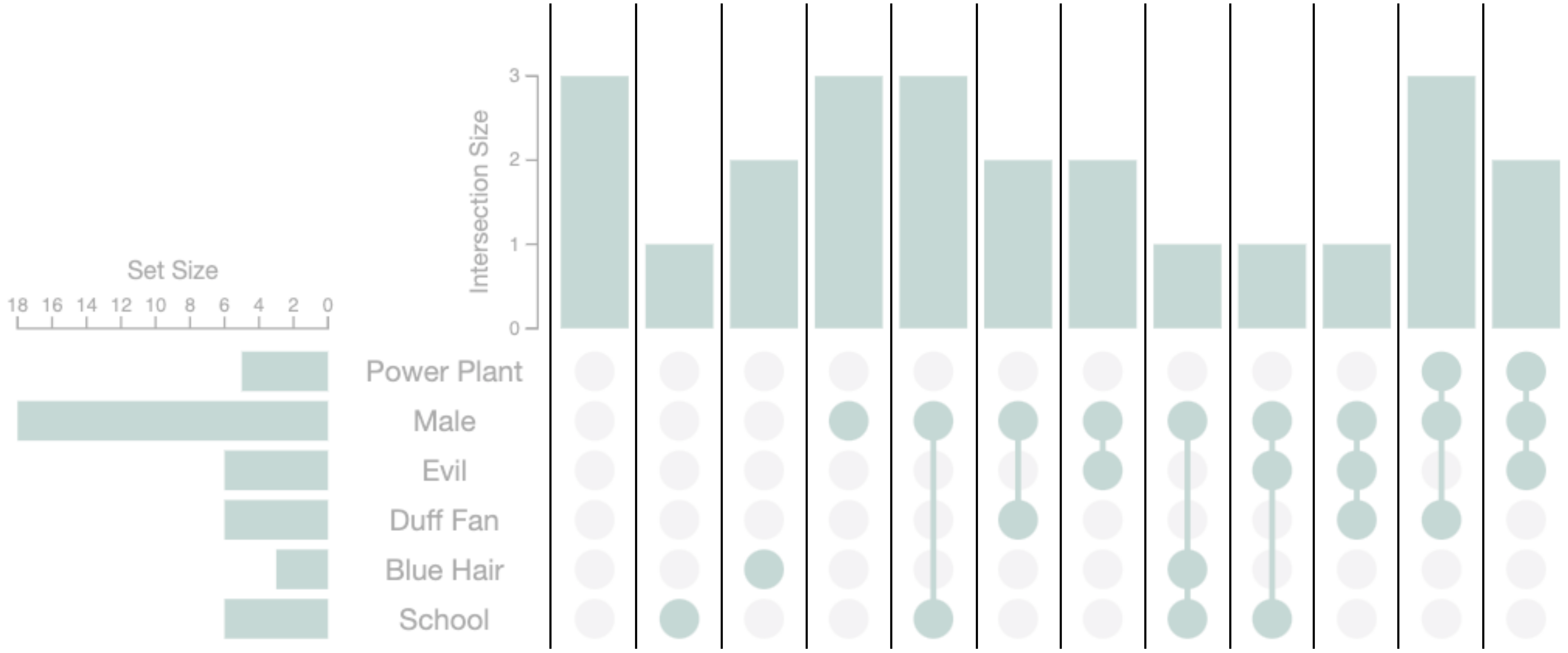
power plant + male + evil

Intersection size

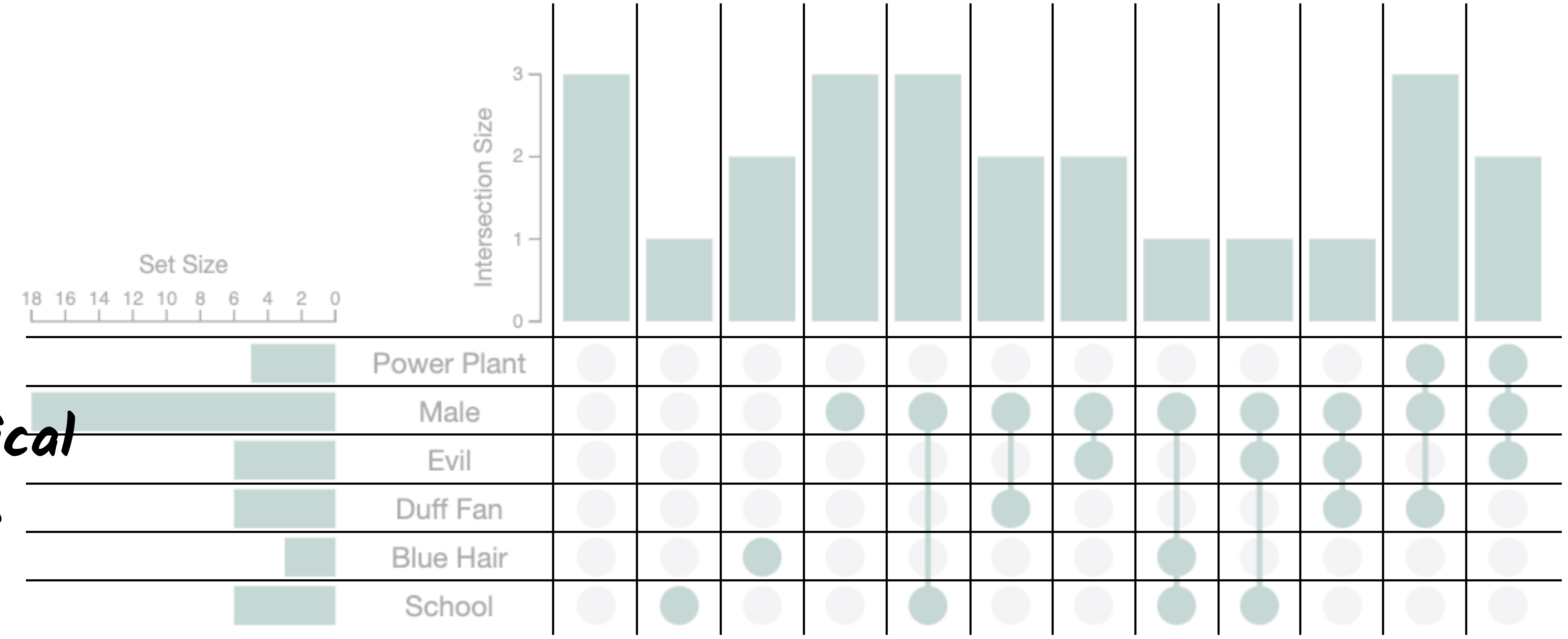



intersection size chart
Intersection Size
Set Size
Power Plant
Male
Blue Hair
School
set size chart
set names
combination matrix

categorical scale: `d3.scaleBand()`



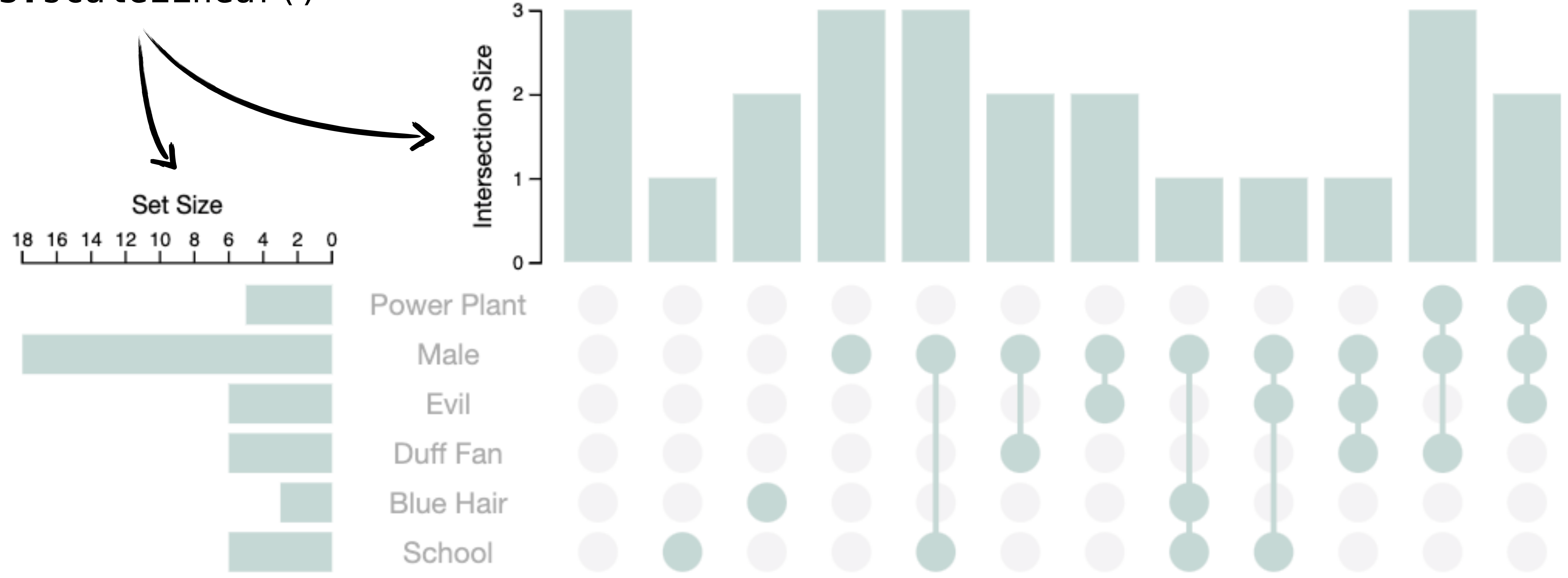
categorical scale: `d3.scaleBand()`



categorical scale

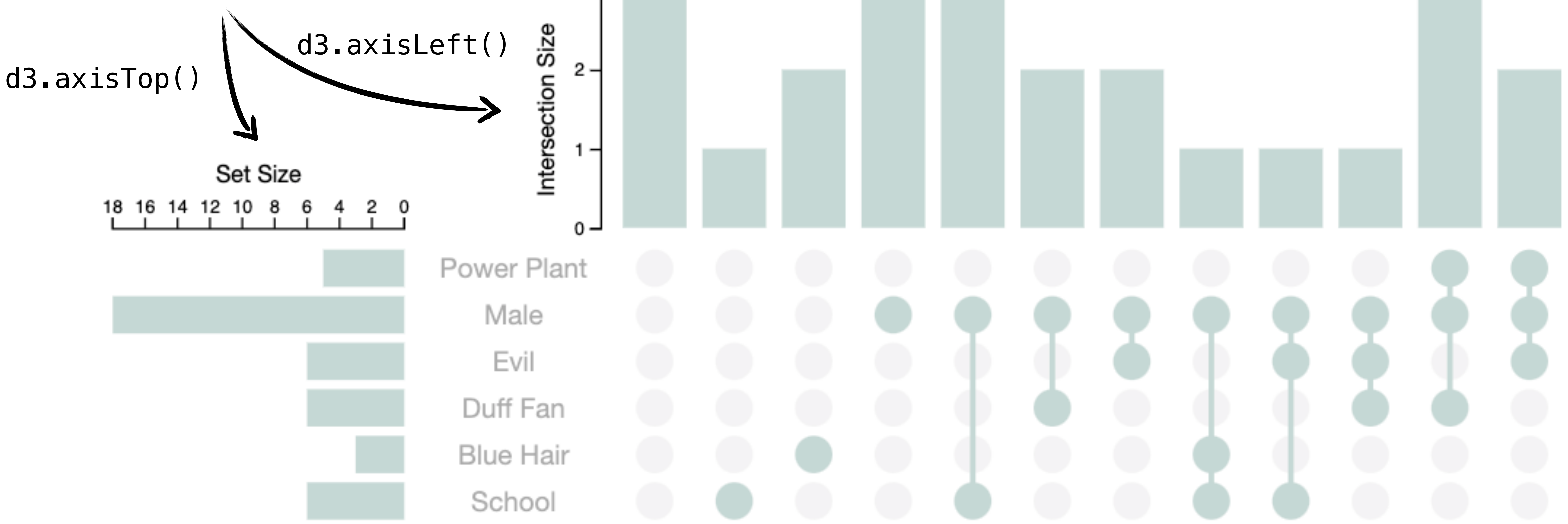
2 linear scales:

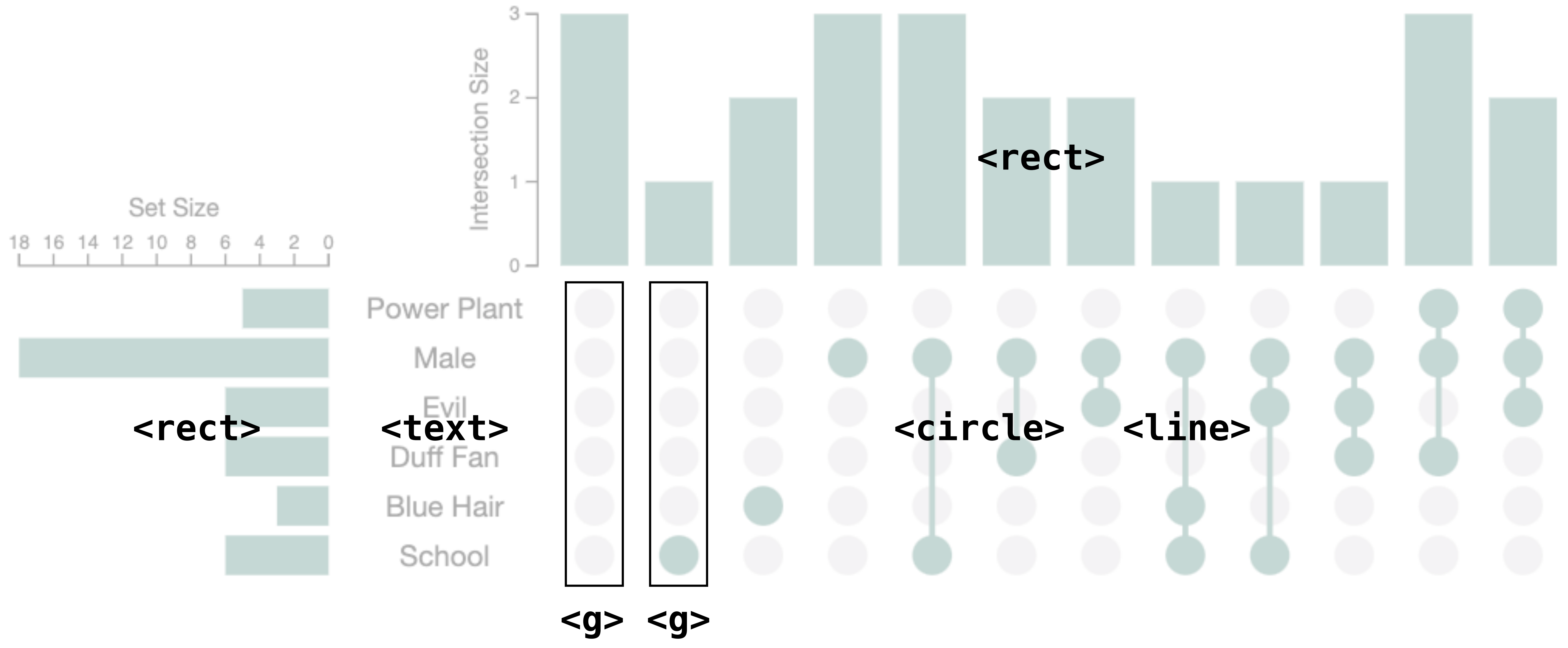
`d3.scaleLinear()`



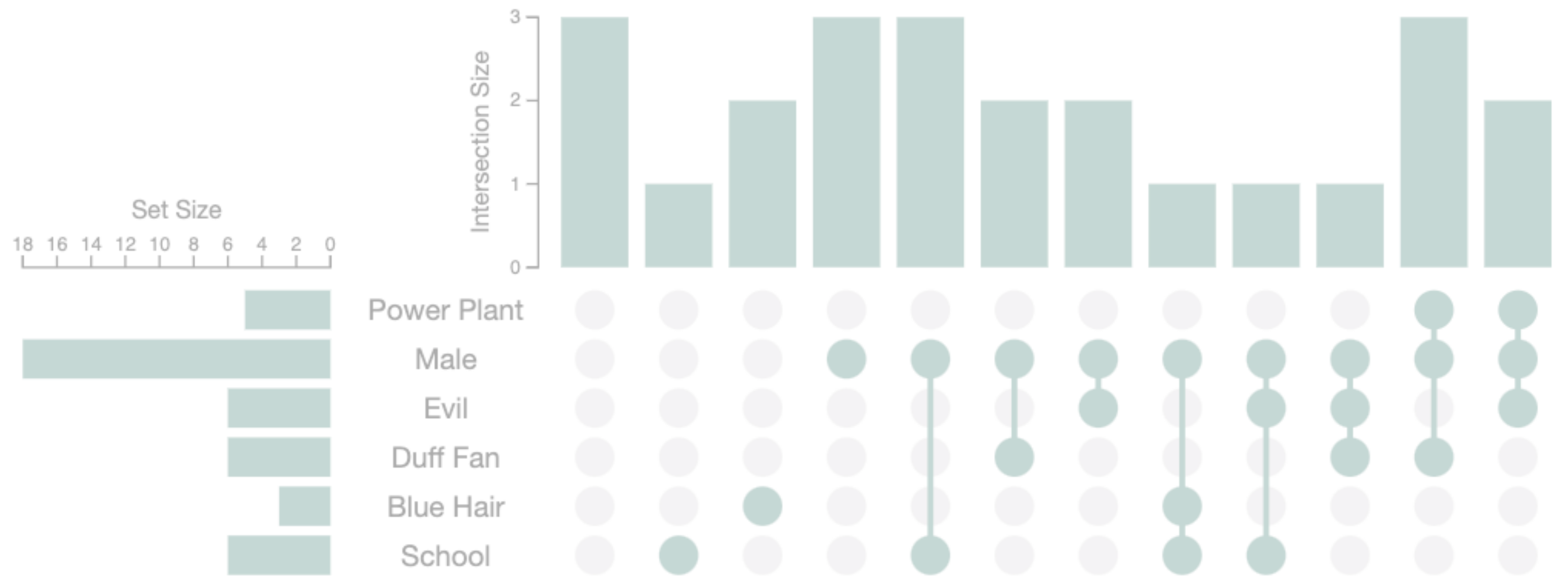
2 linear scales:

`d3.scaleLinear()`

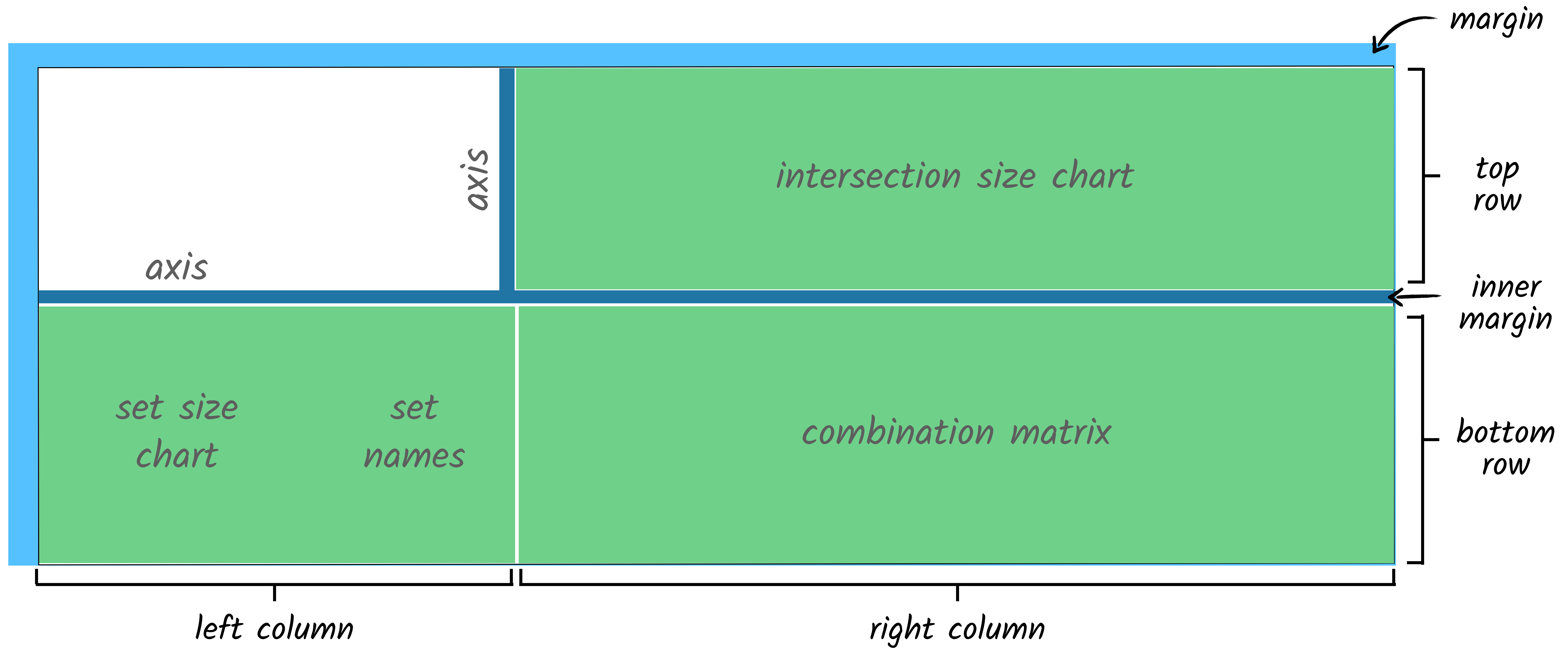




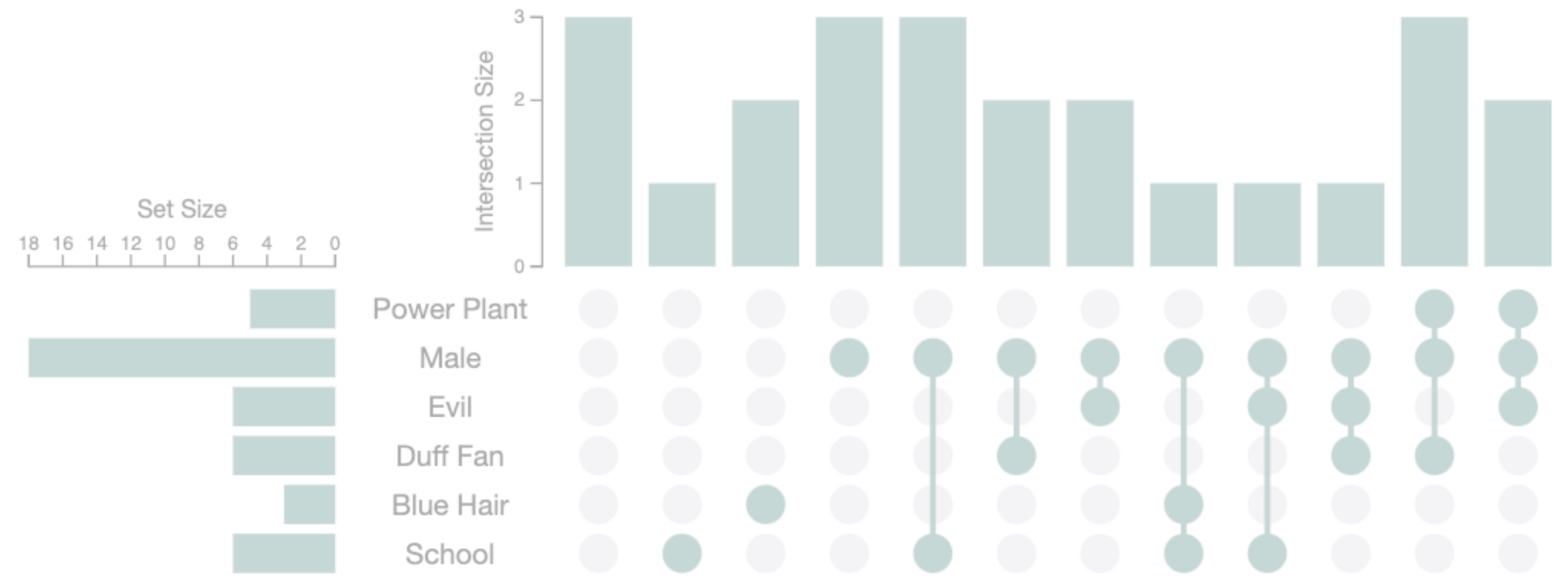
Set Size
18
16
14
12
10
8
6
4
2
0
Intersection Size
3
2
1
0
Power Plant
Male
Evil
Duff Fan
Blue Hair
School
<rect>
<rect>
<text>
<circle>
<line>
<g>
<g>

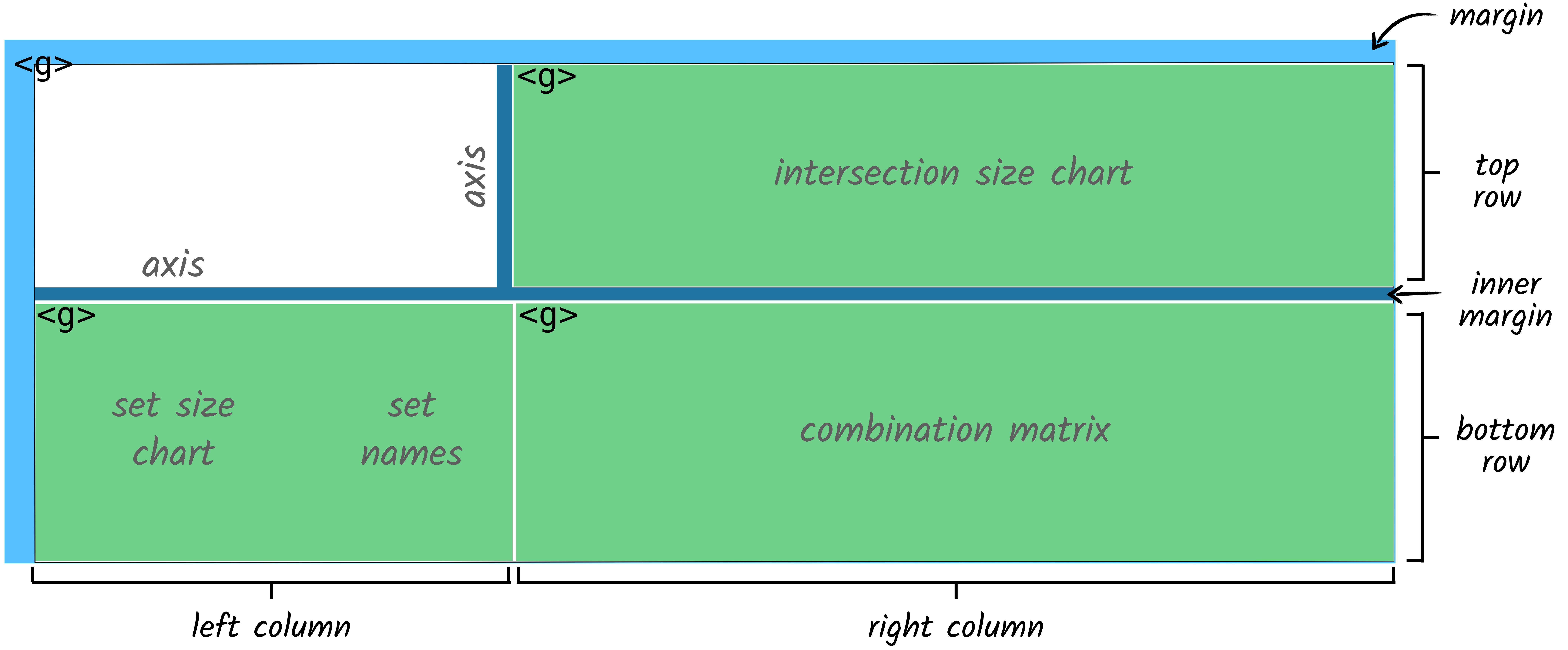


Set Size
18 16 14 12 10 8 6 4 2 0
Intersection Size
0
1
2
3
Power Plant
Male
Evil
Duff Fan
Blue Hair
School

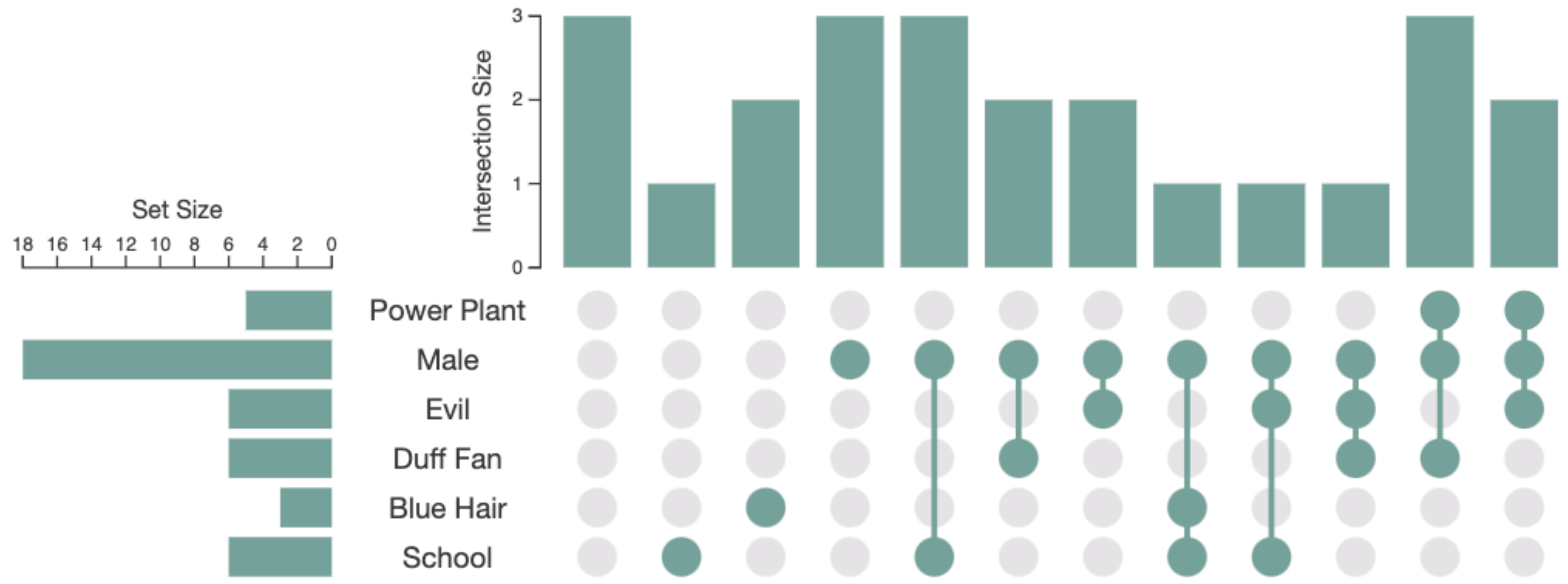


margin
axis
axis
intersection size chart
top row
inner margin
set size chart
set names
combination matrix
bottom row
left column
right column


Set Size
18 16 14 12 10 8 6 4 2 0
Intersection Size
0 1 2 3
Power Plant
Male
Evil
Duff Fan
Blue Hair
School


margin
<g>
axis
axis
<g>
intersection size chart
top row
inner margin
<g>
set size chart
set names
<g>
combination matrix
bottom row
left column
right column

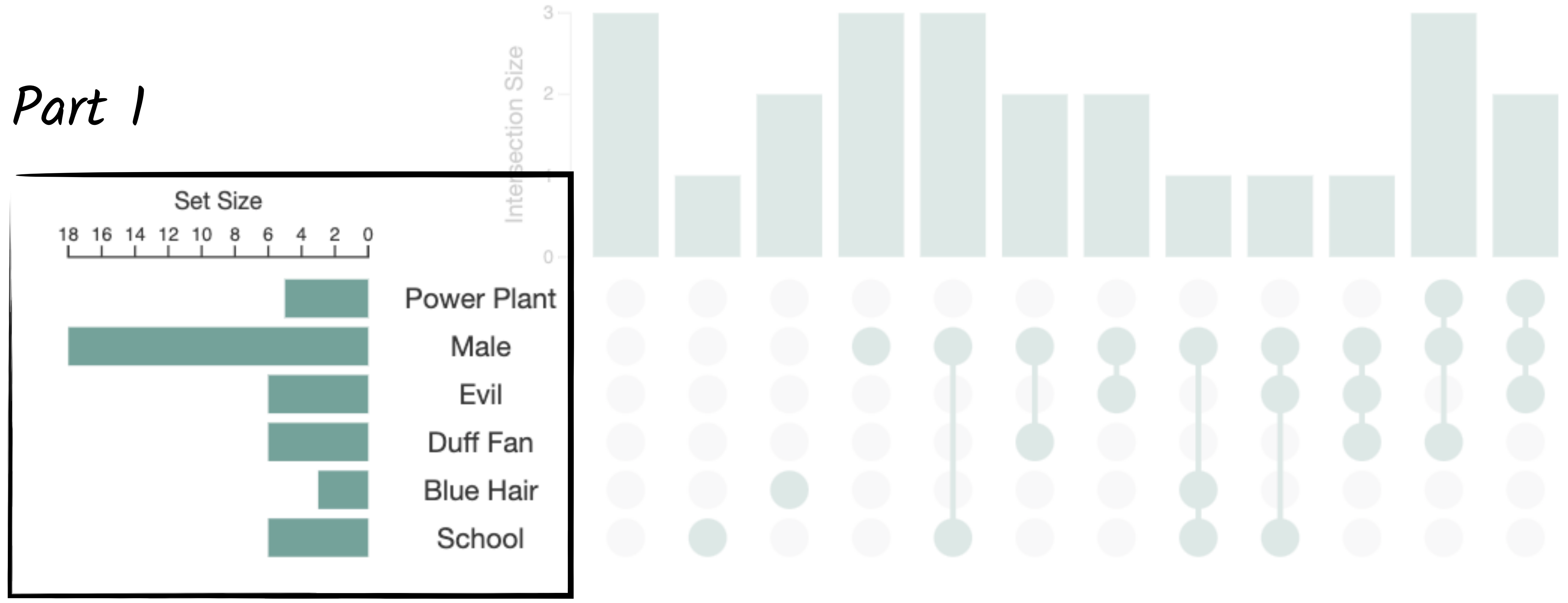
Data preparation



2 parts: ***sets*** *and* ***intersections***

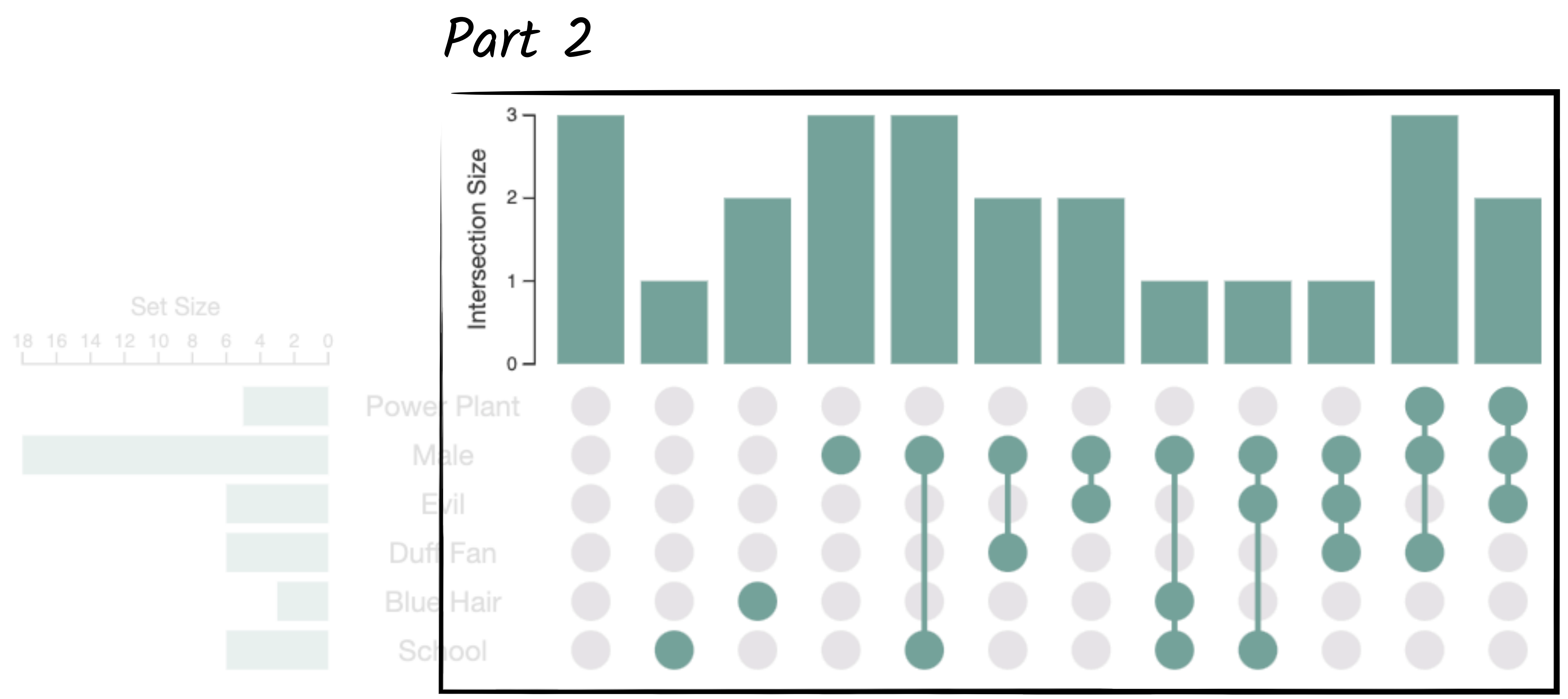
Data preparation

Part 1



```
"sets": [
  {
    "setId": "Power Plant",
    "size": 5
  },
  {
    "setId": "Male",
    "size": 18
  },
  {
    "setId": "Evil",
    "size": 6
  },

  ...
]
```

Data preparation



```
"combinations": [
  {
    "combinationId": "a",
    "setMembership": [],
    "values": ["Maggie", "Patty Bouvier", "Selma Bouvier"]
  },
  {
    "combinationId": "b",
    "setMembership": ["School"],
    "values": ["Lisa"]
  },
  ...
  {
    "combinationId": "e",
    "setMembership": ["School", "Male"],
    "values": ["Bart", "Ralph", "Martin Prince"]
  },
  ...
]
```

michaeloppermann.com/d3/upset

A Tour of D3

Michael Oppermann | michaeloppermann.com/d3

data driven documents	bind data to DOM elements
low-level building blocks	axes, brush, zooming, colour palettes, …
utility functions	load external data, shape and layout generators, …

More resources

- d3js.org
- wattenberger.com/blog/d3
- d3-graph-gallery.com
- observablehq.com/@d3
- christopheviau.com/d3list
- d3indepth.com
- *Navigating the Wide World of Data Visualization Libraries (K. Wongsuphasawat)*