Lecture 6
D3: Bar Chart

DTS204TC Data Visualisation

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Outline

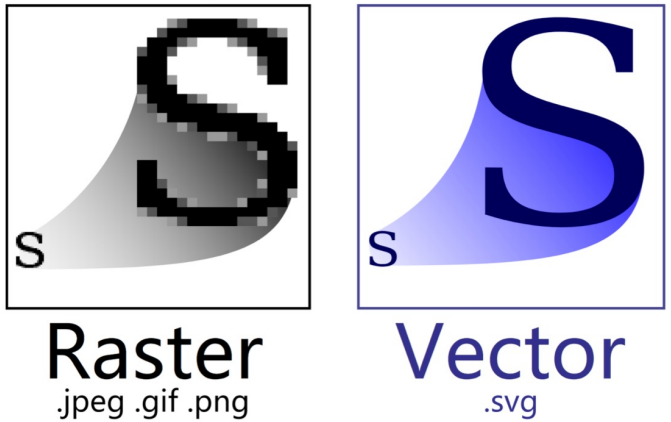
- Review
- Scales in D3
- Axes in D3
- Data Binding
- Bar Chart
- Data Loading

Review

- HTML-Tags
  - <html>: Main Tag. Necessary for every HTML file.
  - <head>: May contains title, links…
  - <body>:main part of html
  - <title>:
  - <script>: For JavaScript codes
  - <svg>: For SVG elements

Review

- SVG (Scalable Vector Graphics)
  - SVG ≈ the canvas for D3
  - SVG is the main object for D3 to operate on.
  - It contains different elements (lines, circles …)
  - Vector
  - <svg> → <g>



Review

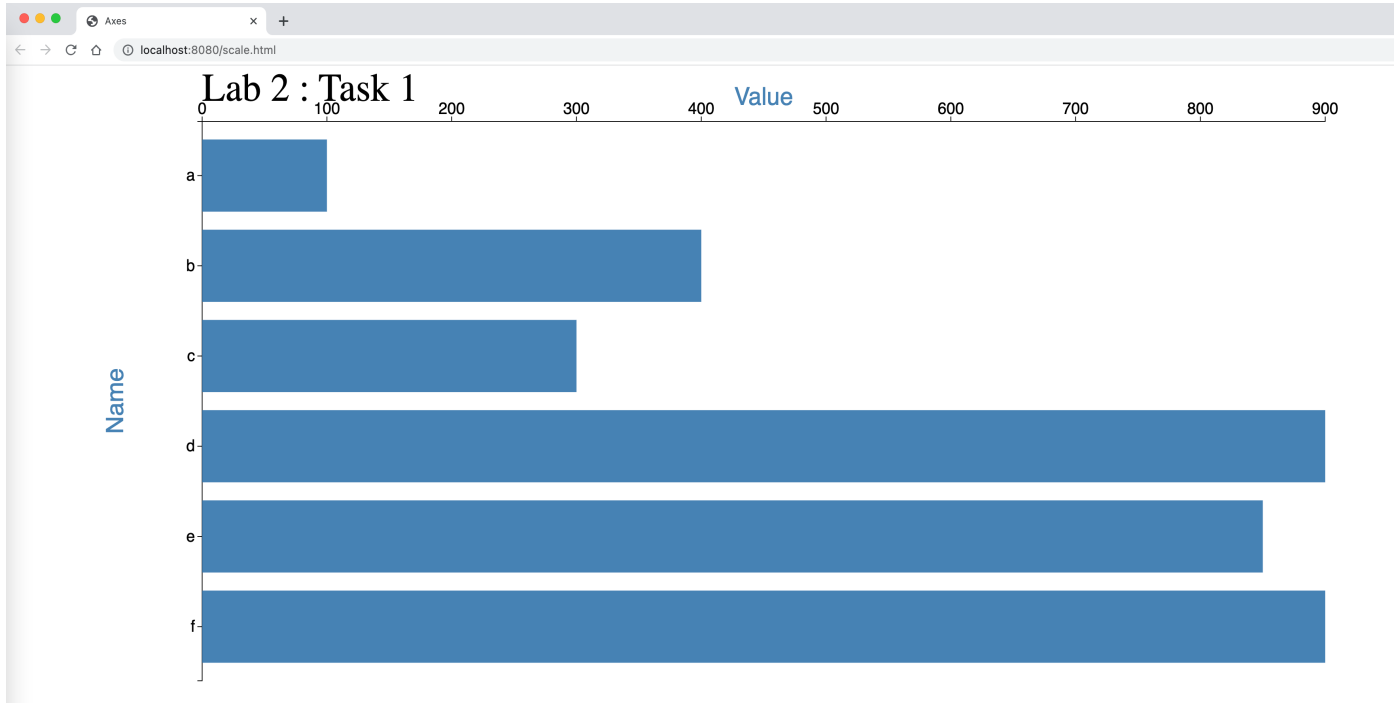
- JavaScript - Web Development Language
  - When you declare a variable, you do not need to specify its type (int, float…)
    - const (constant value), let (block scoped), var (globally scoped)
  - Functions
    - function abc(a){ return a + 5; }
    - let f = datum => datum.value;
    - const p = function(a, b) { return a + b; }
    - let myFunction = (a, b) => a + b
  - A function can be declared as a variable

Review

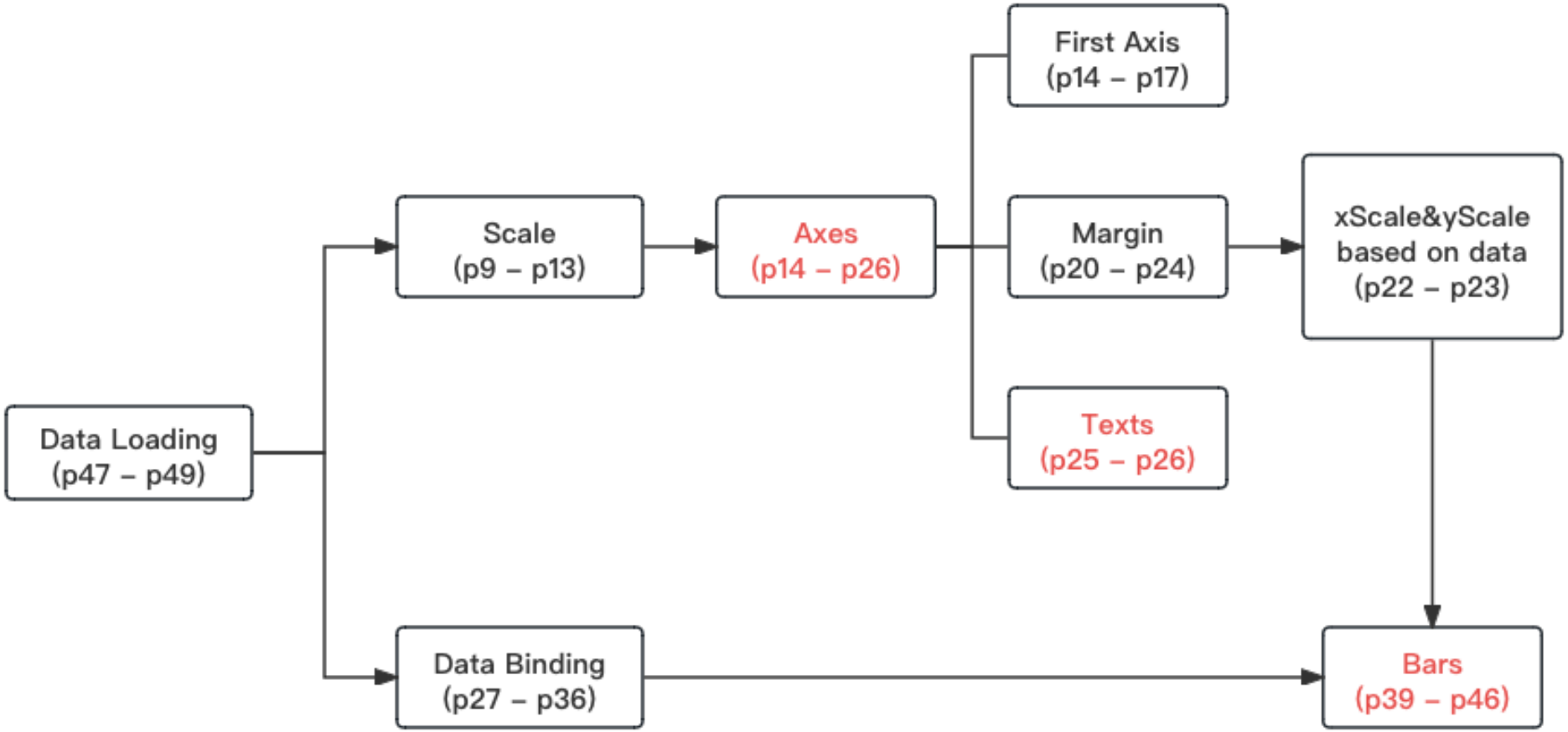
- D3.js
  - D3 stands for **Data-Driven Documents**. Console.log(“Hello World”)
  - id and class
    - id is a unique identifier for an element.
    - class is a identifier for a set of elements (designed by you).
  - D3 Query
    - d3.select(“#id”) * by id
    - d3.select(“svg”) *by tag
    - d3.selectAll(“.class1”) *by class
    - d3.select(“#secondgroup rect”)

Bar Charts

- Axis
- Bars
- Texts
- …

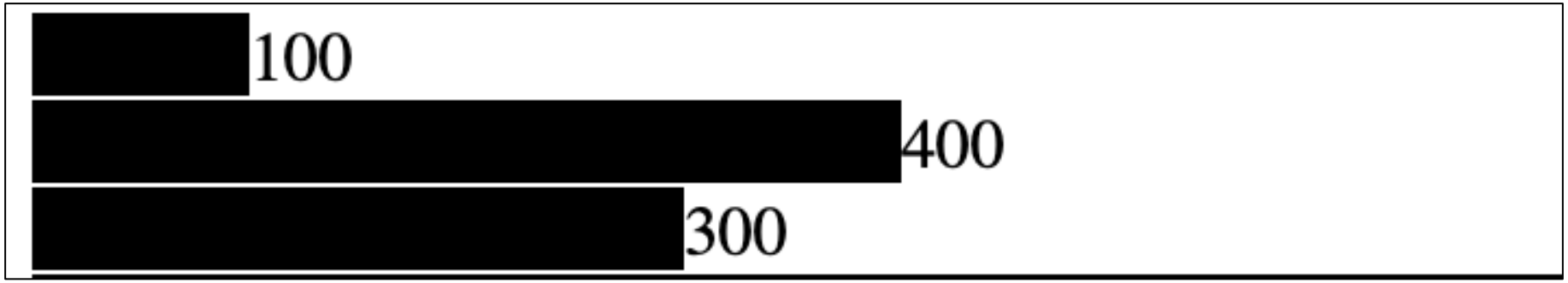


Bar Charts


First Axis
(p14 – p17)
Scale
(p9 – p13)
Axes
(p14 – p26)
Margin
(p20 – p24)
xScale&yScale
based on data
(p22 – p23)
Data Loading
(p47 – p49)
Texts
(p25 – p26)
Data Binding
(p27 – p36)
Bars
(p39 – p46)

Scales in D3

- D3 Scales provide a convenient solution to map our **data values** to values that would be better represented in **visualizations**.
- Linear Scale
- Band Scale
- Example: [100, 400, 300]



Scales in D3

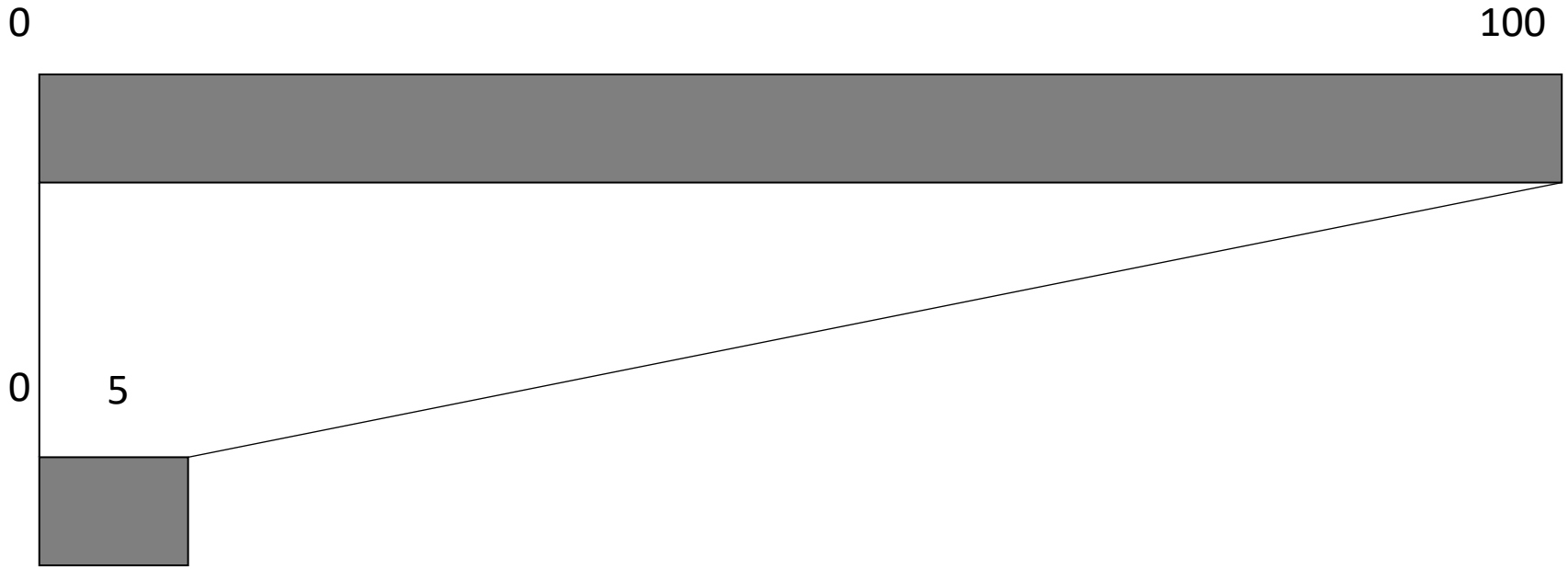
- Preparations
  - d3.max(array) ➔ max value
  - d3.min(array) ➔ min value
  - d3.extent(array) ➔ [min,max]
  - array.map() ➔ list

Scales in D3

- Linear Scale
  - d3.scaleLinear().domain([min_d, max_d]).range([min,max])
  - **Domain**: Continuous
  - **Range**: continuous

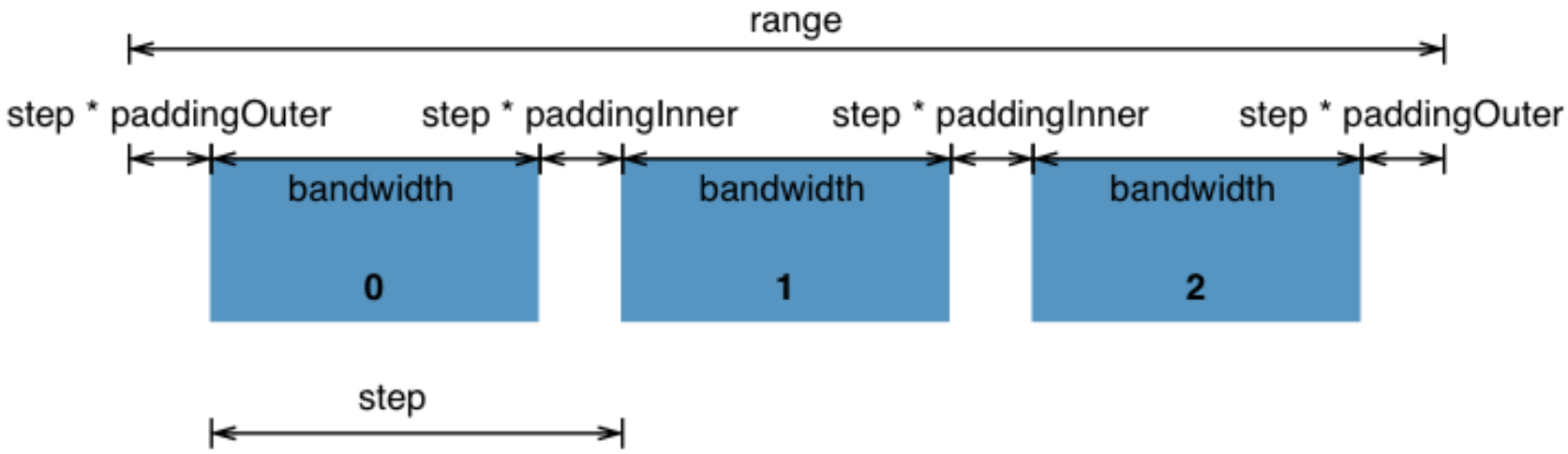
Scales in D3

- Linear Scale
  - d3.scaleLinear().domain([min_d, max_d]).range([min,max])
  - Eg： const LScale = d3.scaleLinear().domain([0,100]).range([0,5])



Scales in D3

- Band Scale
  - d3.scaleBand.domain(array).range([min, max])
  - Domain: discrete
  - Range: continuous
  - Eg： const Bscale = d3.scaleBand().domain(data.map(…)).range([0,5])



Axes in D3

- The axes renders human-readable reference marks for scales. Graphs have two axes: the horizontal axis (x-axis) and the vertical axis (y-axis).
- D3 provides four functions to draw axes
  - d3.axisTop() ➔ Top horizontal axis
  - d3.axisLeft() ➔ Left vertical axis
  - d3.axisBottom() ➔ Bottom horizontal axis
  - d3.axisRight() ➔ Right vertical axis

Axes in D3

- Define axis:
  - const yAxis = d3.axisLeft(yScale);
  - const xAxis = d3.axisBottom(xScale);
- Rendering axis (append a group element and insert axis)
  - svg.append(“g”).call(xAxis)

```
<script>
        const svg = d3.select("#mainsvg");
        const data1 = [100, 400, 300, 900, 850, 900];
        //scale
        const scale1 = d3.scaleLinear().domain([0,900]).range([0,500]);
        //axis
        const x_axis = d3.axisTop().scale(scale1);
        svg.append ("g").call(x_axis);
</script>
```

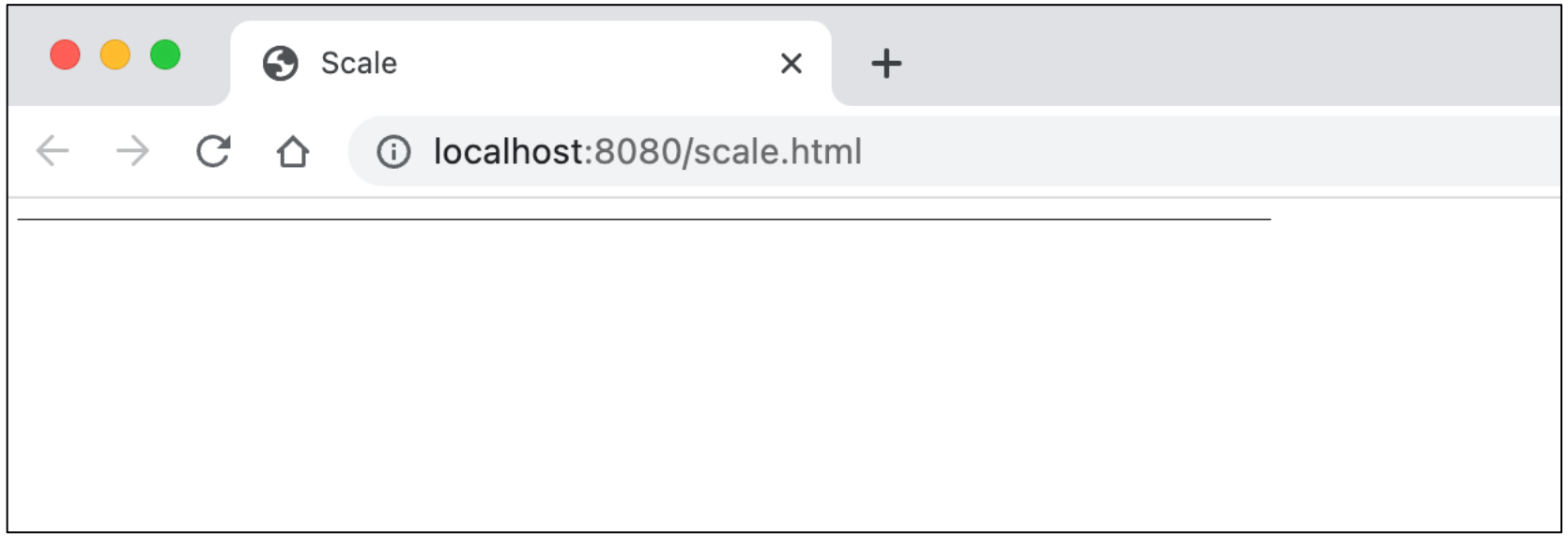
Axes in D3

- If we directly insert the axis to the svg, what would be happened?

Axes in D3

- If we directly insert the axis to the svg, what would be happened?

The axis cannot be rendered correctly !



Axes in D3

- If we directly insert the axis to the svg, what would be happened?

The axis cannot be rendered correctly !

We need to translate the axis !

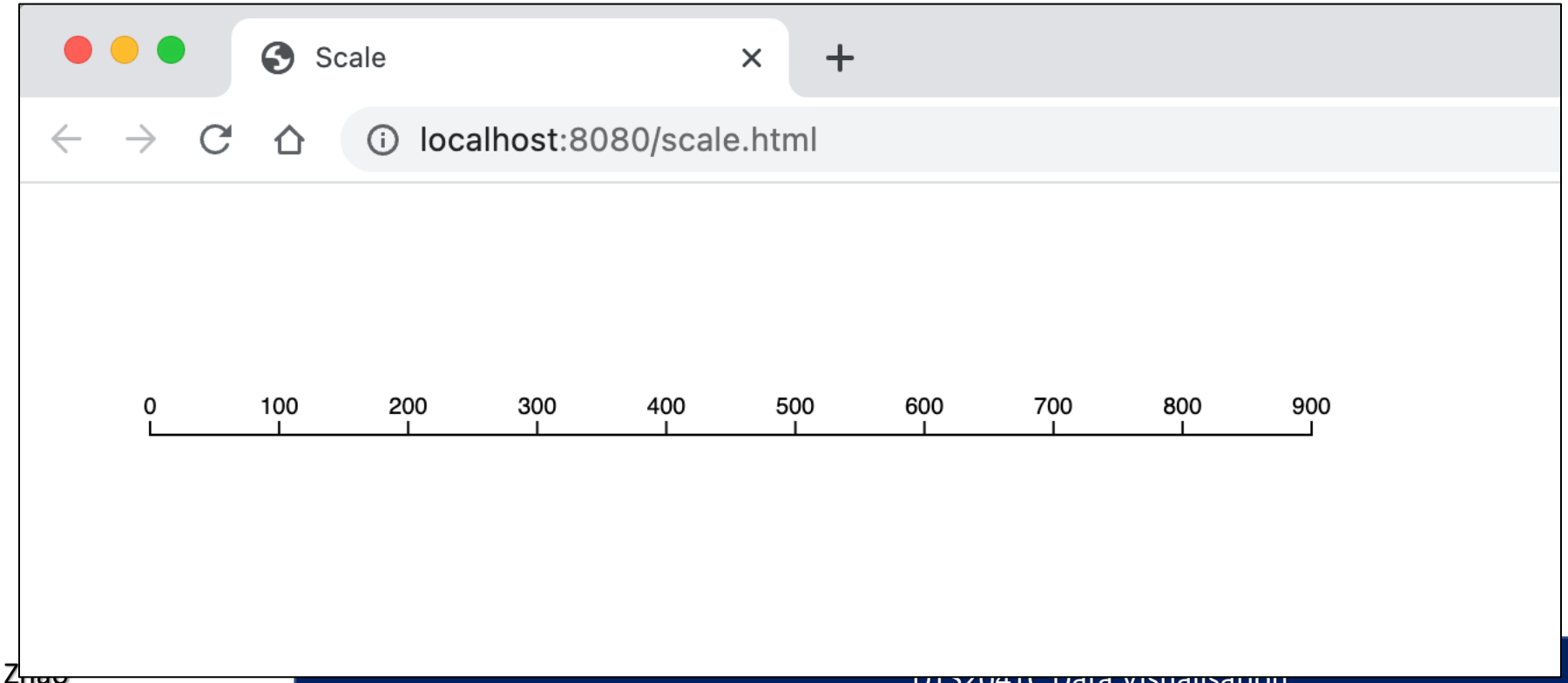
```
<script>
        const svg = d3.select("#mainsvg");
        const data1 = [100, 400, 300, 900, 850, 900];
        //scale
        const scale1 = d3.scaleLinear().domain([0,900]).range([0,500]);
        //axis
        const x_axis = d3.axisTop().scale(scale1);
        svg.append (“g”).call(x_axis).attr(“transform”, “translate(50,100)”);
</script>
```

Axes in D3

- If we directly insert the axis to the svg, what would be happened?

The axis cannot be rendered correctly !

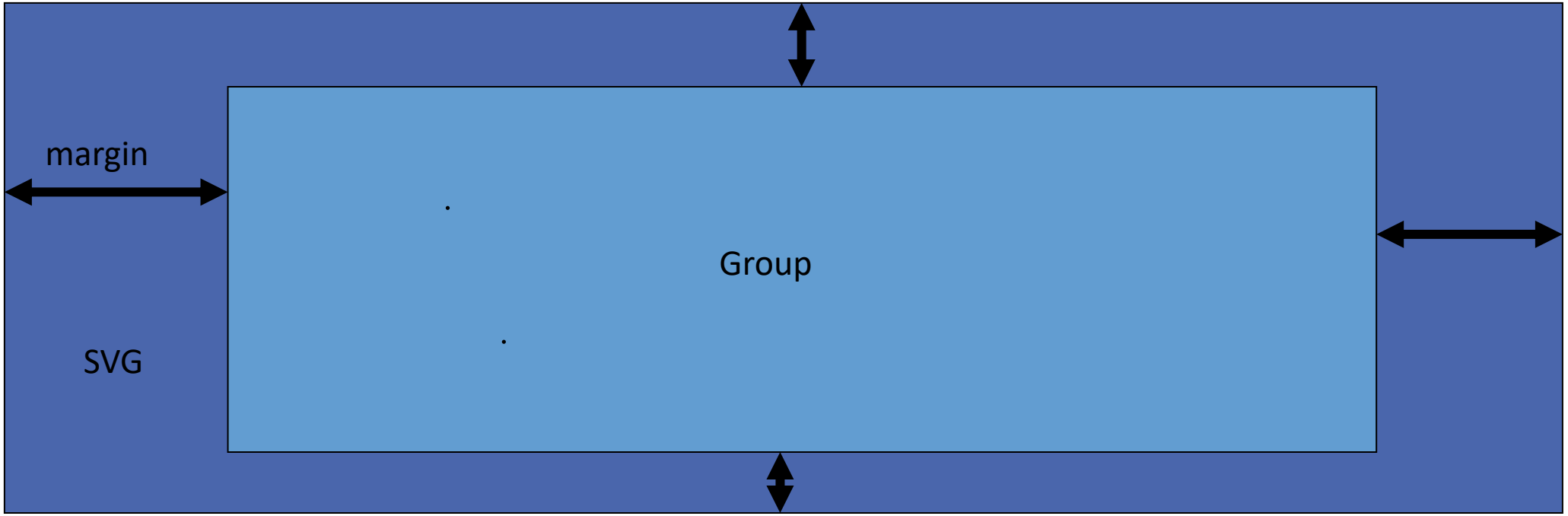
We need to translate the axis



Axes in D3

- Margin
  - Why margin?

SVG ➔ group ➔axis



Axes in D3

- Margin
  - Set margin
    - Define margin
      - const margin = {top: 60, right: 30, bottom: 60, left: 200};
    - Compute the width and height for the group
      - const innerWidth = width - margin.left - margin.right;
      - const innerHeight = height - margin.top - margin.bottom;
    - Append group
      - const g = svg.append('g')
      - .attr('id', 'maingroup')
      - .attr('transform', `translate(${margin.left}, ${margin.top})`);

Axes in D3

- Example

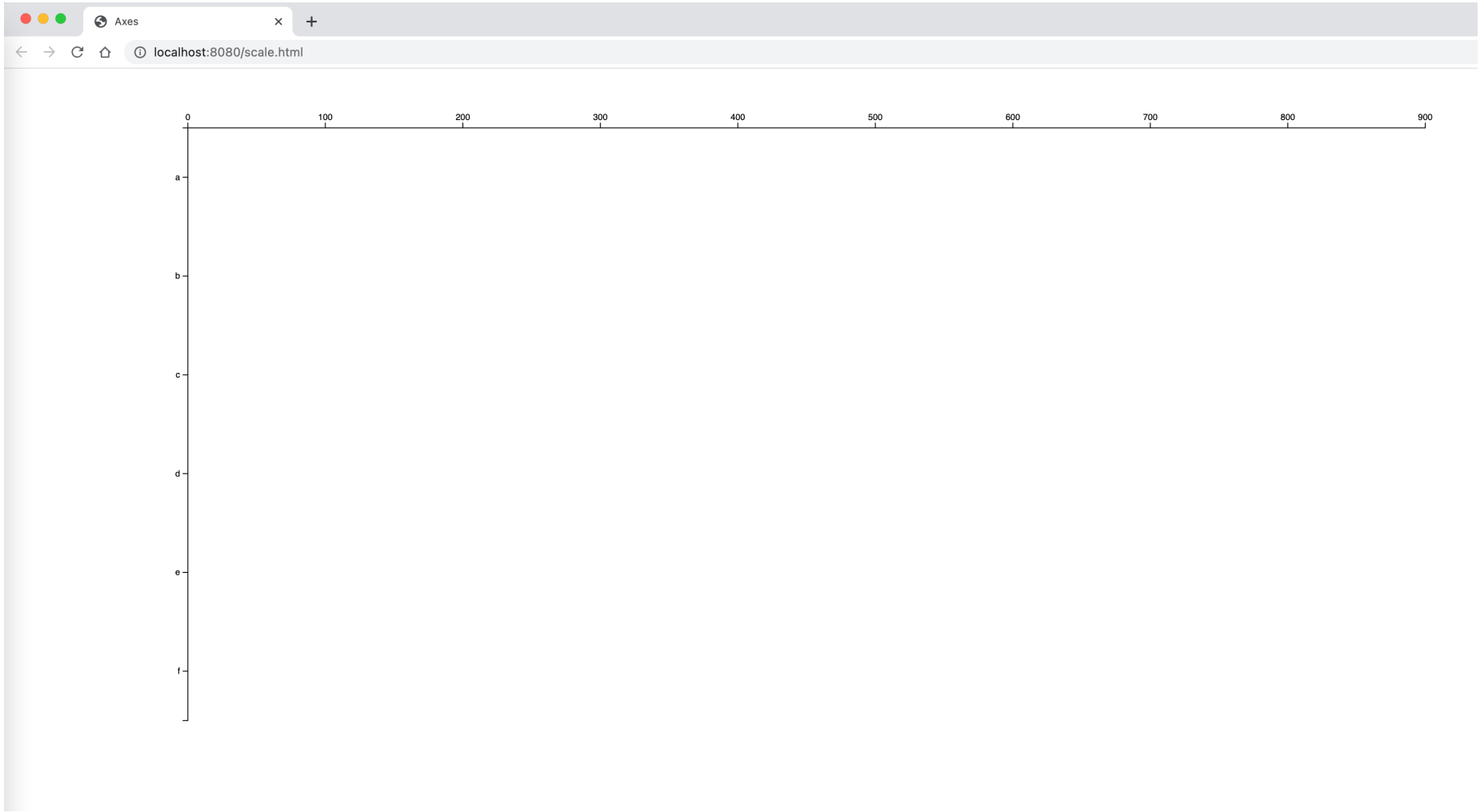
```
<script>
        //set svg and margin
        const svg = d3.select("#mainsvg");

        const width = +svg.attr("width");
        const height = +svg.attr("height");
        const margin = {top: 60, right: 30, bottom: 60, left: 150};
        const innerWidth = width - margin.left - margin.right;
        const innerHeight = height - margin.top - margin.bottom;

        //data
        const data = [
        {name:"a", value:100},{name:"b", value: 400},{name:"c", value: 300},
        {name:"d", value:900},{name:"e", value: 850},{name:"f", value: 900},
        ];
…
```

Axes in D3

```
…
        //set 2 scales
        const xScale = d3.scaleLinear()
        .domain([0, d3.max(data, d => d.value)])
        .range([0,innerWidth]);
        const yScale = d3.scaleBand()
        .domain(data.map(d => d.name))
        .range([0,innerHeight]);

        //set the group and insert axis
        const g = svg.append("g")
                .attr("id", "maingroup")
                .attr("transform", `translate(${margin.left}, ${margin.top})`);

        const x_axis = d3.axisTop().scale(xScale);
        const y_axis = d3.axisLeft().scale(yScale);
        g.append ("g").call(x_axis);
        g.append ("g").call(y_axis);
</script>
```

Axes in D3

- Example



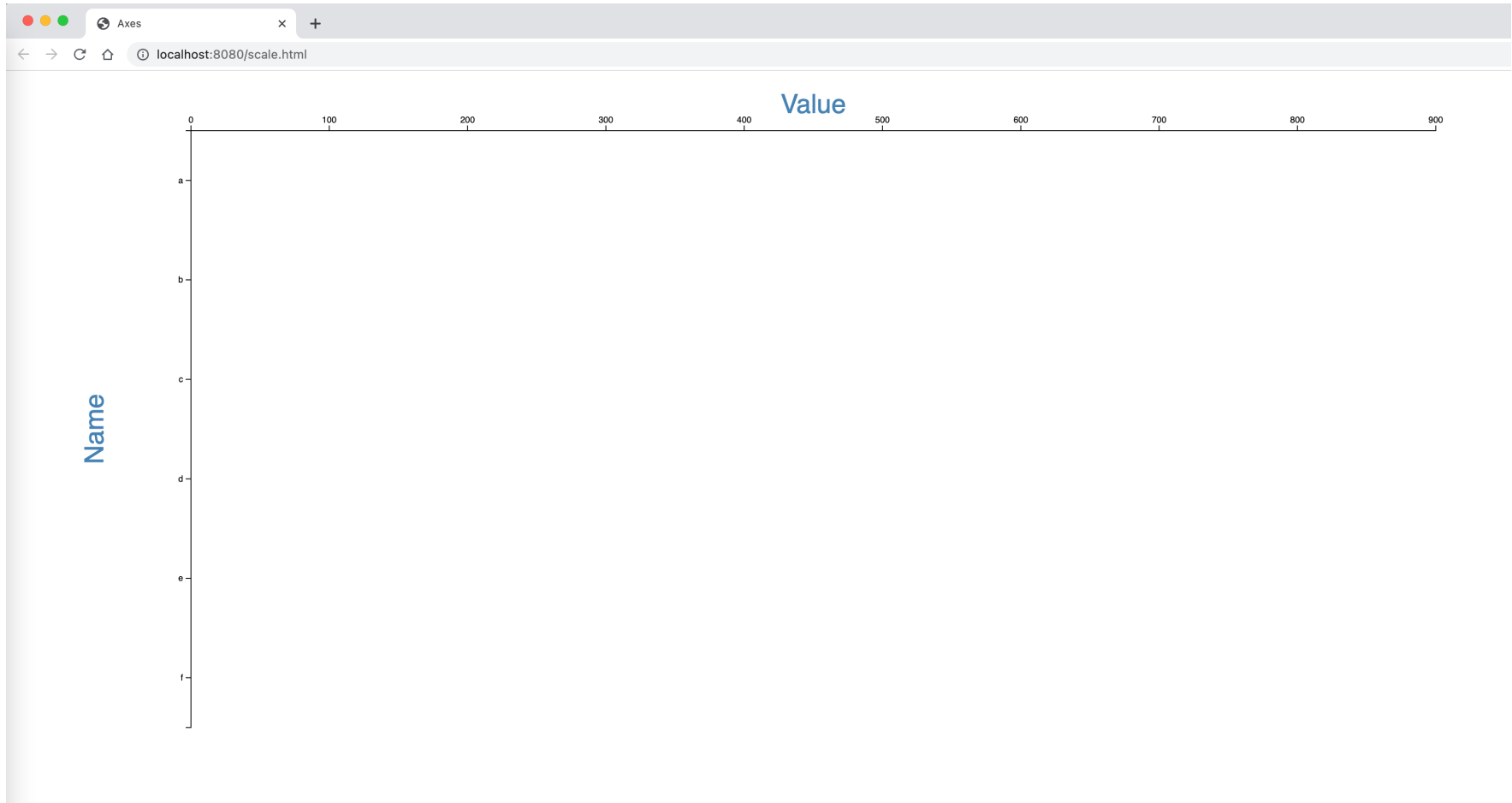
Axes in D3

- Title

```
g.append ("g").call(x_axis)
.append("text")
.text("Value")
.attr("font-size", "3em")
.attr("x", innerWidth / 2)
.attr("y", -20)
.attr("text-anchor", "middle")
.attr("fill", "steelblue");

g.append ("g").call(y_axis)
.append("text")
.text("Name")
.attr("font-size", "3em")
.attr("x", -innerHeight / 2)
.attr("y", -100)
.attr("transform", "rotate(-90)")
.attr("text-anchor", "middle")
.attr("fill", "steelblue");
```

Axes in D3

- Title



Data Binding

- Data ➔element
  - how to bind data to DOM elements and create new elements based on your data
  - .data(dataArray)
- Data-join (enter)
  - selection.data(dataArray).enter()…
- Data-join (update)
  - selection.data(dataArray)…
- Data-join (remove)
  - selection.data(dataArray).exit().remove()

Data Binding

- Data-join (update)
  - selection.data()…

```
<p>a</p>
<p>b</p>
<p>c</p>
<script>
        var myData = ["hello","world","d3"];
        var p = d3.select("body")
        .selectAll("p")
        .data(myData)
        .text((d,i) => d);
</script>
```

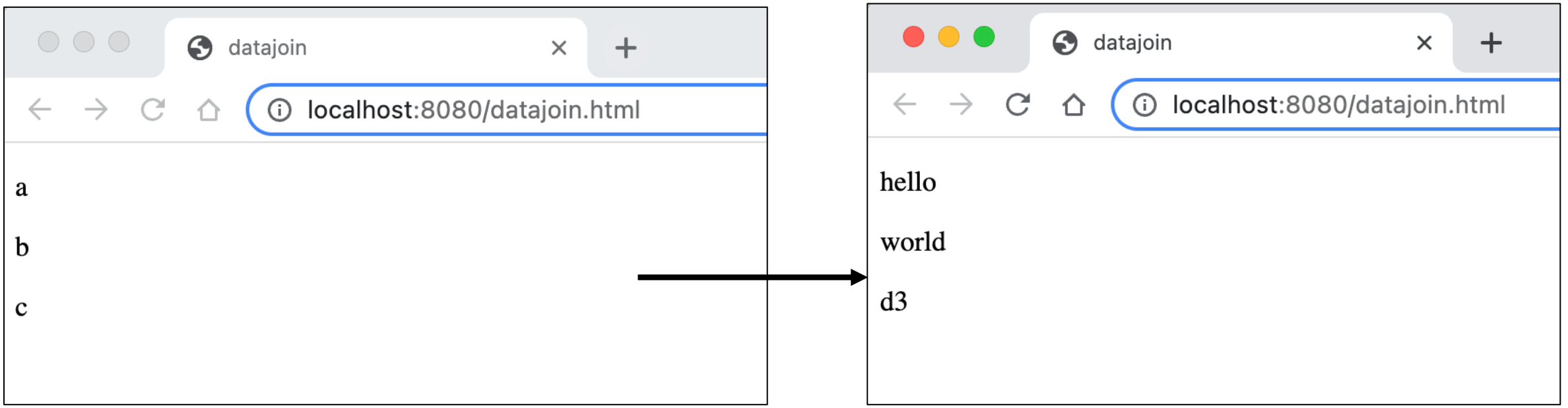
Data Binding

- The code selects the `<body>` element of the HTML document using `d3.select("body")`. This means that the subsequent operations will be applied to the body of the HTML document.
- Then, it selects all `<p>` elements within the `<body>` using `.selectAll("p")`. This selects all paragraph elements on the page.
- The `.data(myData)` method binds the data from the `myData` array to the selected `<p>` elements. This means that each string in the array will be associated with a corresponding `<p>` element.
- Next, the `.text((d, i) => d)` method is used to set the text content of each `<p>` element. In this method, `(d, i) => d` is an arrow function that takes two parameters:
  - `d` represents the current data point (each string in the array),
  - `i` represents the index of the current data point in the array.

```
<p>a</p>
<p>b</p>
<p>c</p>
<script>
        var myData = ["hello","world","d3"];
        var p = d3.select("body")
        .selectAll("p")
        .data(myData)
        .text((d,i) => d);
</script>
```

Data Binding

- Data-join (update)
  - selection.data()…



Data Binding

- Data-join (enter)
  - selection.data().enter()

```
<body>
        <script>
                var myData = ["hello","world","d3"];
                var p = d3.selectAll("p")
                .data(myData)
                .enter()
                .append("p")
                .text(d => d);
        </script>
</body>
```

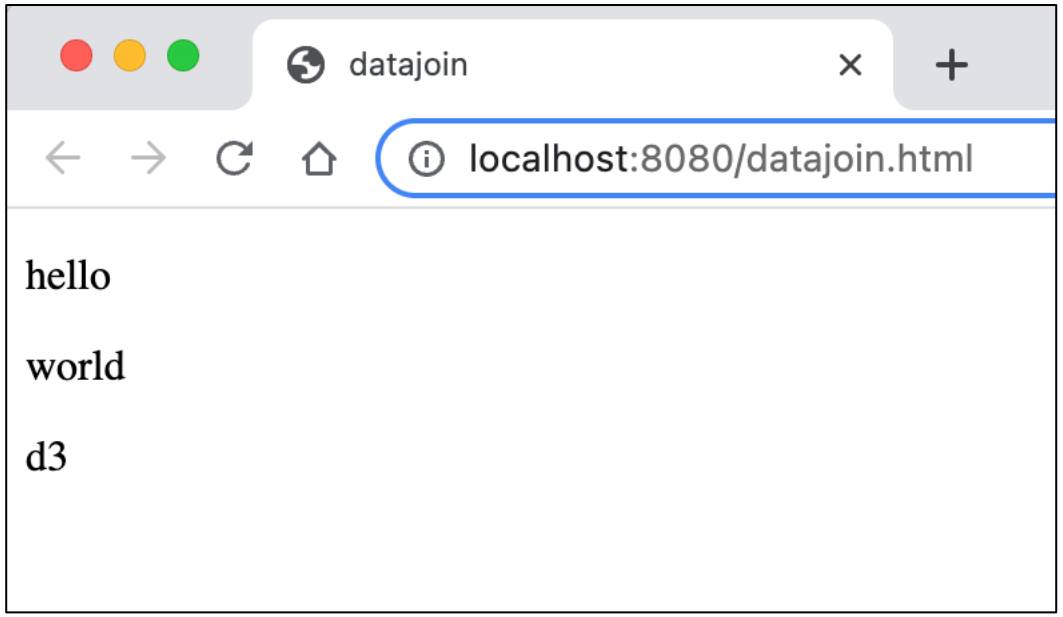
Data Binding

- `var p = d3.selectAll(“p”)`: Selects all existing `<p>` elements on the page. (empty selection)
- `.data(myData)`: Binds the data from `myData` array to the selected `<p>` elements. (nothing happened)
- `.enter()`: Creates a new selection containing the data points that are not yet bound to any element.
- `.append("p")`: Appends a new `<p>` element for each unbound data point.
- `.text(d => d)`: Sets the text content of each new `<p>` element to the corresponding data point.
- This effectively adds `<p>` elements to the page with text content taken from the `myData` array.

```
<body>
        <script>
                var myData = ["hello","world","d3"];
                var p = d3.selectAll("p")
                .data(myData)
                .enter()
                .append("p")
                .text(d => d);
        </script>
</body>
```

Data Binding

- Data-join (enter)
  - selection.data().enter()



Data Binding

- Data-join (remove)
  - selection.data().exit().remove()

```
<p>D3</p>
<p>aa</p>
<p>aa</p>
<script>
        var myData = ["Hello World!"];

        var p = d3.select("body")
        .selectAll("p")
        .data(myData)
        .text(d =>d)
        .exit()
        .remove();
</script>
```

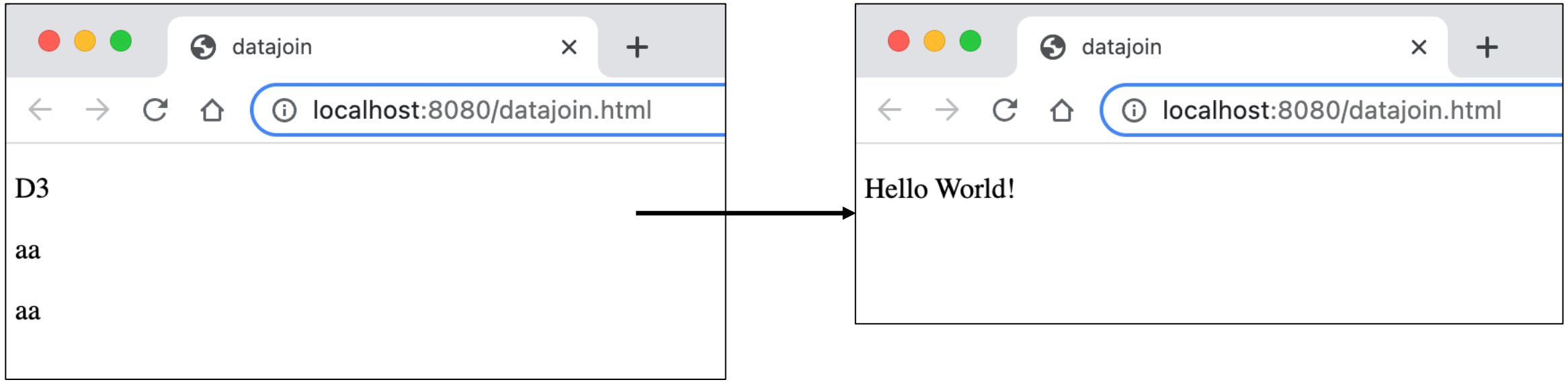
Data Binding

- `var p = d3.select("body")`: Selects the `<body>` element using `d3.select("body")`, indicating that operations will be performed on the body of the document.
- `.selectAll("p")`: Selects all `<p>` elements on the page.
- `.data(myData)`: Binds the data from `myData` array to the selected `<p>` elements. As there are fewer data points than selected elements, this creates a selection containing unbound data points.
- `.text(d => d)`: Sets the data point ("Hello World!") as the text content of each selected element.
- `.exit().remove()`: Finally, `.exit().remove()` removes any extra elements. Since there are fewer data points than selected elements, `.exit()` selects elements that are not bound to any data, and `.remove()` removes them from the document.

```
<p>D3</p>
<p>aa</p>
<p>aa</p>
<script>
        var myData = ["Hello World!"];

        var p = d3.select("body")
        .selectAll("p")
        .data(myData)
        .text(d =>d)
        .exit()
        .remove();
</script>
```

Data Binding

- Data-join



Bar Chart

- Review

```
<script>
        //set svg and margin
        const svg = d3.select("#mainsvg");

        const width = +svg.attr("width");
        const height = +svg.attr("height");
        const margin = {top: 60, right: 30, bottom: 60, left: 150};
        const innerWidth = width - margin.left - margin.right;
        const innerHeight = height - margin.top - margin.bottom;

        //data
        const data = [
        {name:"a", value:100},{name:"b", value: 400},{name:"c", value: 300},
        {name:"d", value:900},{name:"e", value: 850},{name:"f", value: 900},
        ];
…
```

Bar Chart

```
…
        //set 2 scales
        const xScale = d3.scaleLinear()
        .domain([0, d3.max(data, d => d.value)])
        .range([0,innerWidth]);

        const yScale = d3.scaleBand()
        .domain(data.map(d => d.name))
        .range([0,innerHeight]);

        //set the group and insert axis
        const g = svg.append("g")
                .attr("id", "maingroup")
                .attr("transform", `translate(${margin.left}, ${margin.top})`);

        const x_axis = d3.axisTop().scale(xScale);
        const y_axis = d3.axisLeft().scale(yScale);
        g.append ("g").call(x_axis);
        g.append ("g").call(y_axis);
</script>
```

Bar Chart

- We have scales and axes now, to complete a bar chart, we need bars
- **bar = rectangle**. Therefore, we need to append rectangle according to the data.

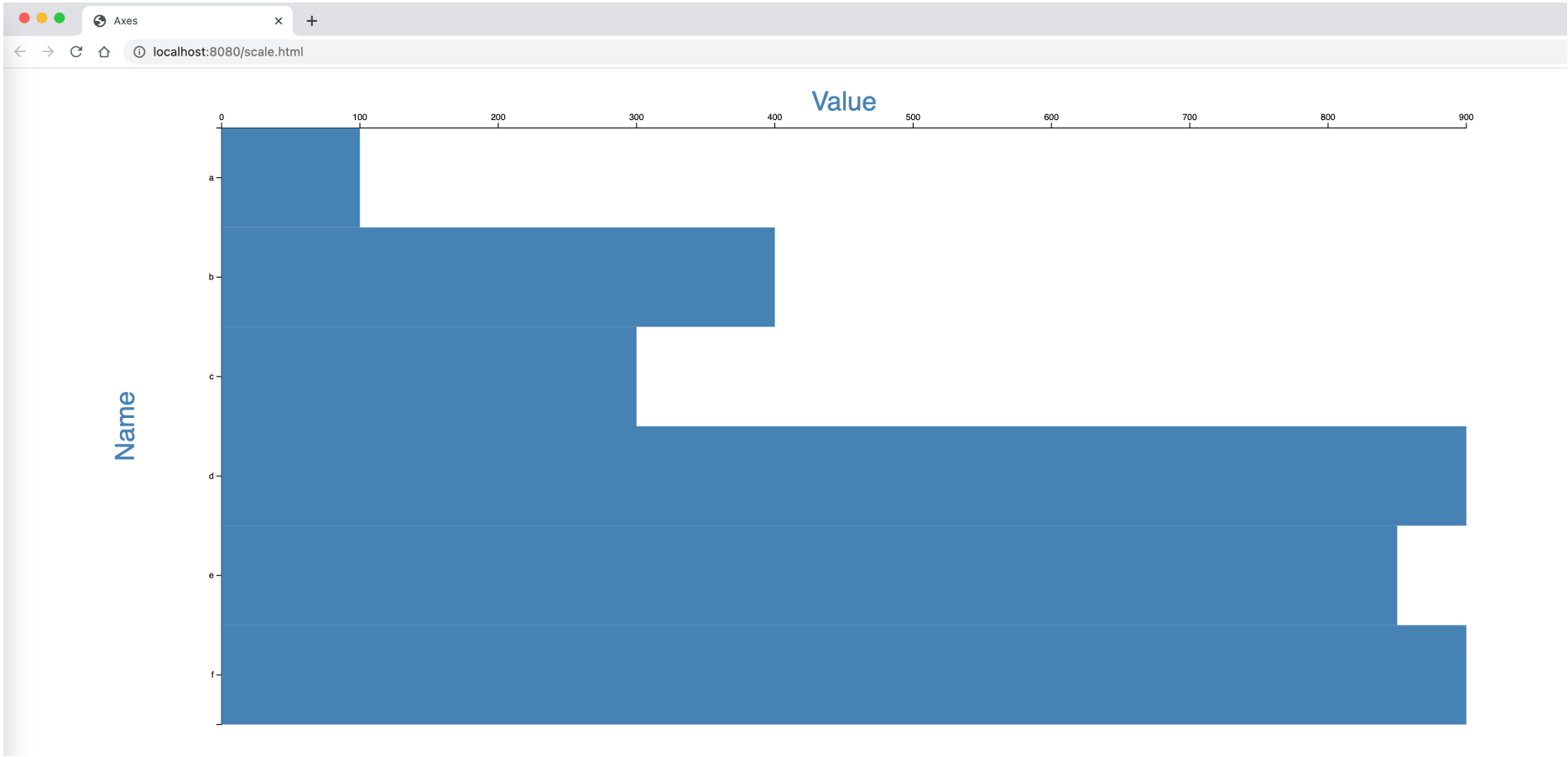
```
g.selectAll(".bar").data(data).enter()
.append("rect")
.attr("class","bar")
.attr("x",0)
.attr("y", d => yScale(d.name))
.attr("width",d => xScale(d.value))
.attr("height",d => yScale.bandwidth())
.attr("fill","steelblue");
```

Bar Chart

```
//there are bar class, so it will return an empty array
g.selectAll(".bar")

//We provide our data array to the data() function.
.data(data)

//Create a selection with placeholder
.enter()

//append rectangles to each group element.
.append("rect")

//make rectangles in the bar class for further updating or exiting
.attr("class","bar")
```

Bar Chart

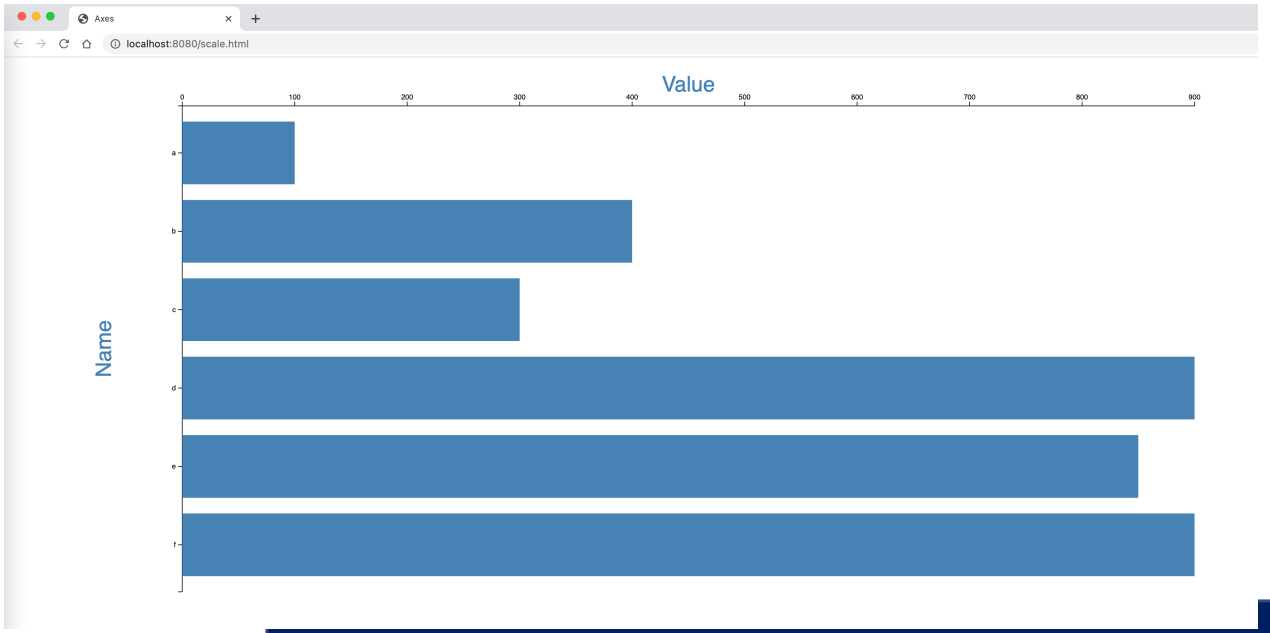
Some Problems



Bar Chart

- Add space between bars

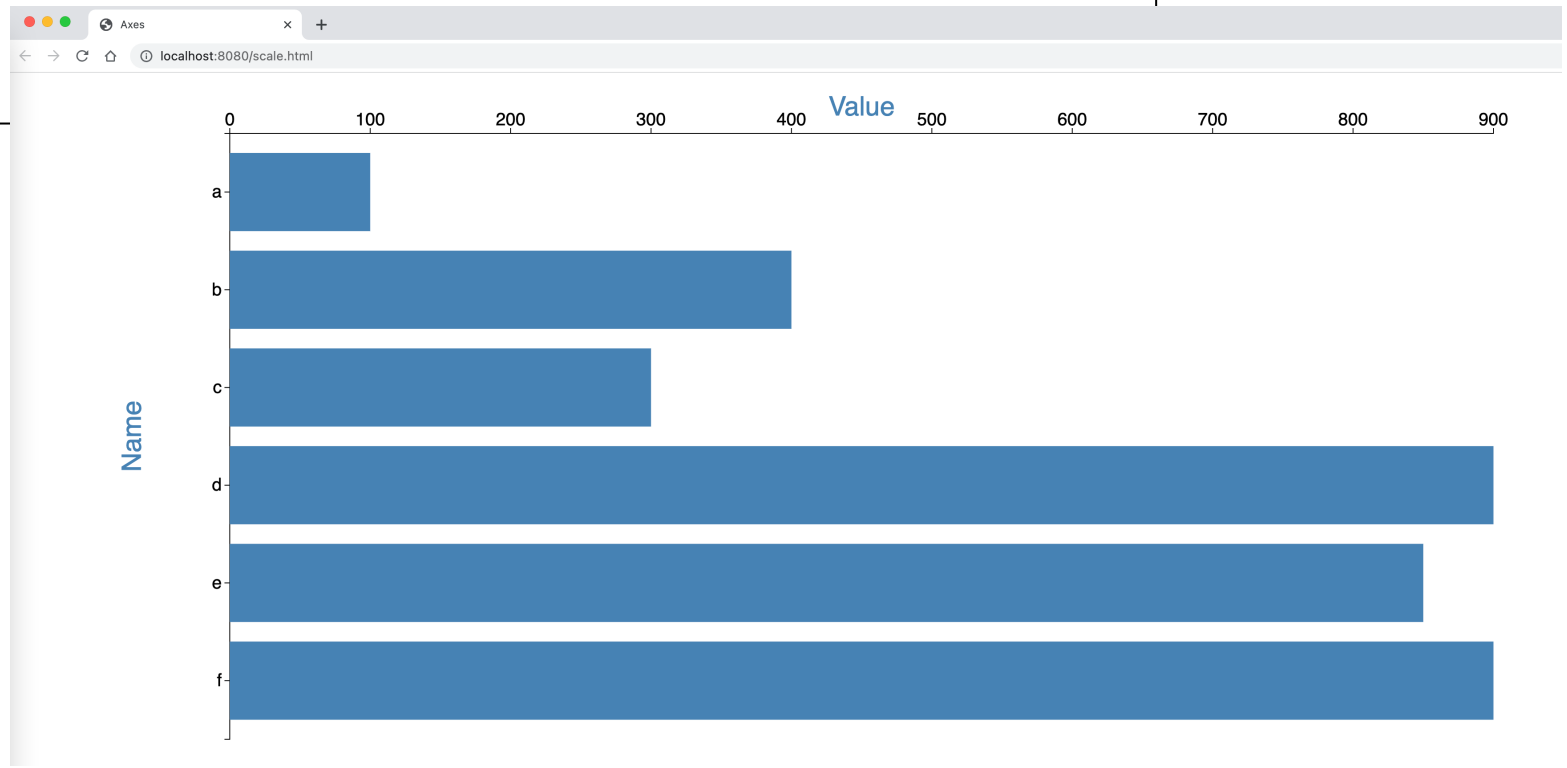
```
const yScale = d3.scaleBand()
.domain(data.map(d => d.name))
.range([0,innerHeight])
.padding(0.2);
```



Bar Chart

- make the size of text bigger

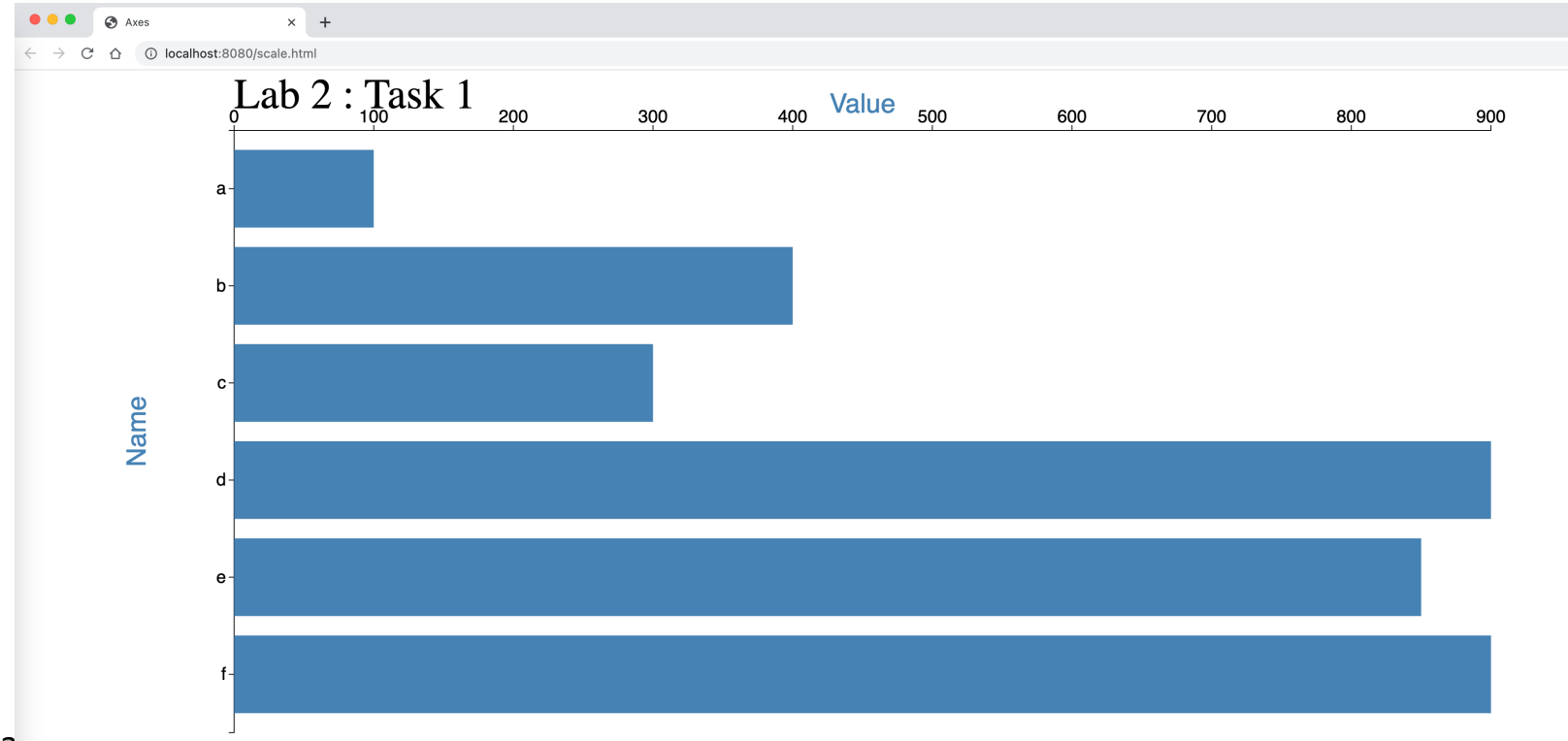
```
d3.selectAll(".tick text")
.attr("font-size","2em");
```



Bar Chart

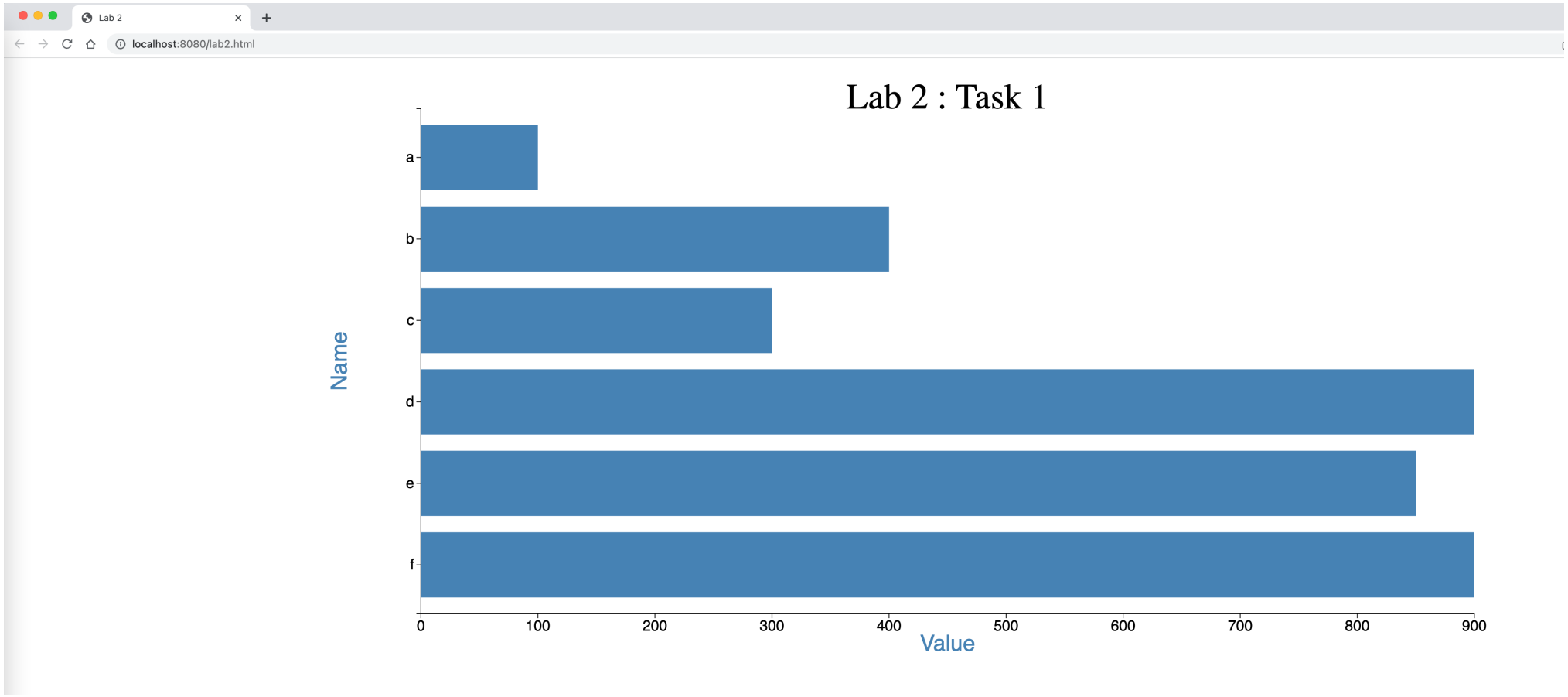
- add title

```
g.append("text").text("Lab 2 : Task 1")
.attr("font-size","3em")
.attr("transform","translate(0,-25)");
```



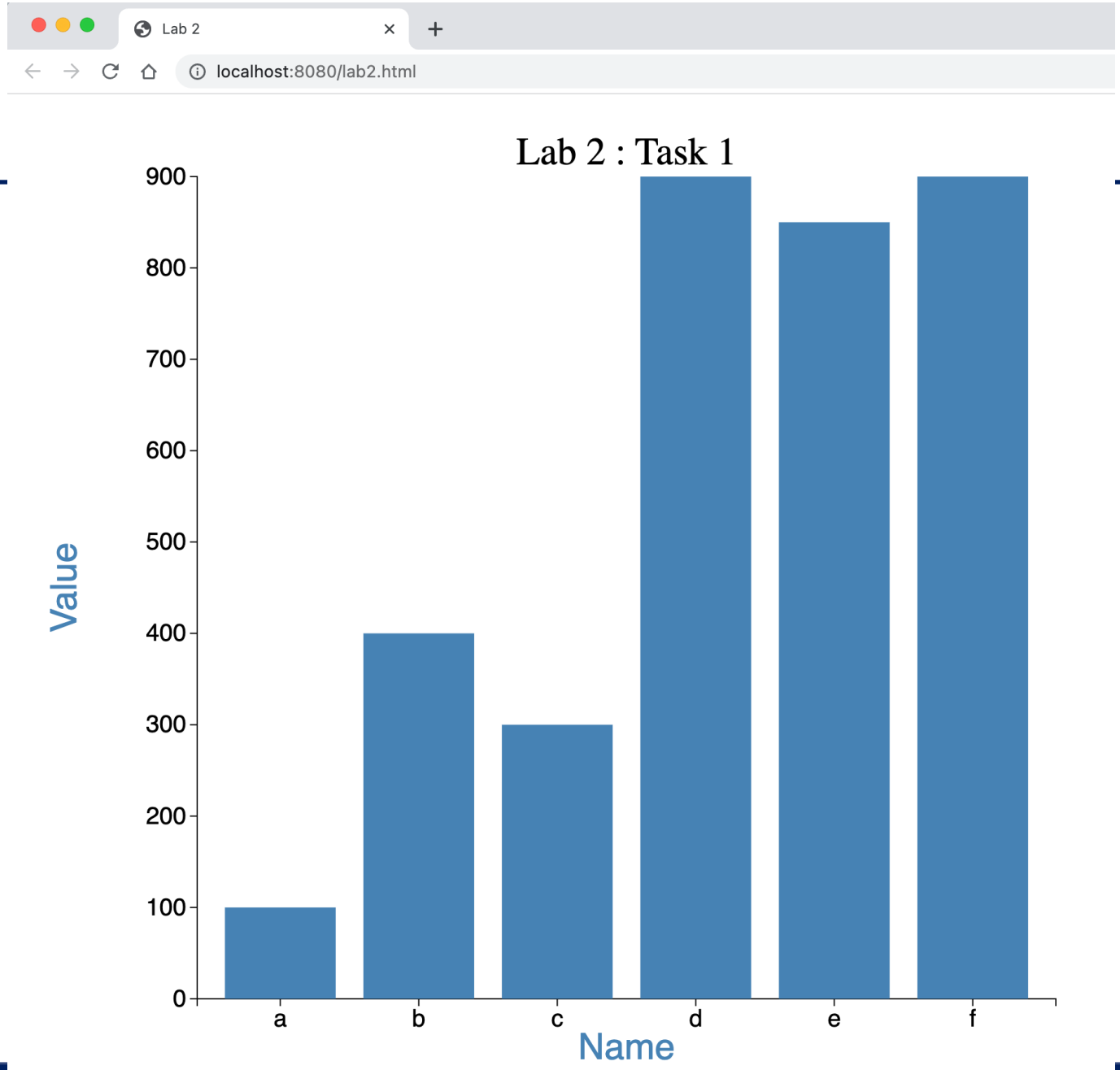
Bar Chart

- Lab tasks



Bar Chart

- Lab tasks



Data Loading

- We have worked with data stored in local variables. How to load data from different types of files and bind it to DOM elements ?
- D3 provides the following methods to load different types of data from external files.

Method	Description
d3.csv()	Sends http request to the specified url to load .csv file or data and executes callback function with parsed csv data objects.
d3.json()	Sends http request to the specified url to load .json file or data and executes callback function with parsed json data objects.
d3.tsv()	Sends http request to the specified url to load a .tsv file or data and executes callback function with parsed tsv data objects.
d3.xml()	Sends http request to the specified url to load an .xml file or data and executes callback function with parsed xml data objects.

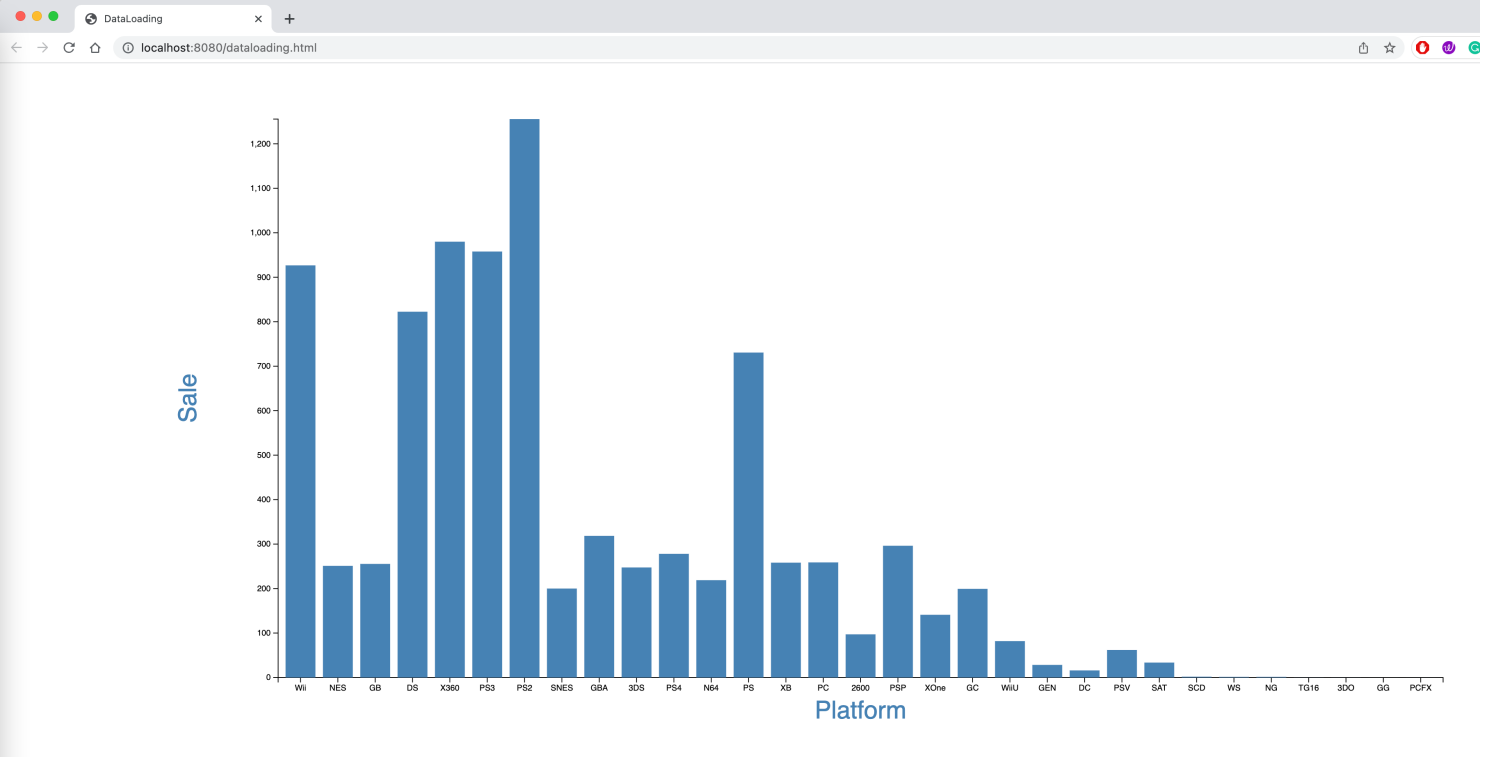
Data Loading

- d3.csv()
  - d3.csv(”path/to/data.csv”).then(data => {…})

```
d3.csv("platform_globalsale.csv").then(data =>{
        data.forEach(d => {d["globalsale"] = +(d["globalsale"]) })

        //Scale
        …
        //Axes
        …
        //Bar
        …
}
```

Data Loading

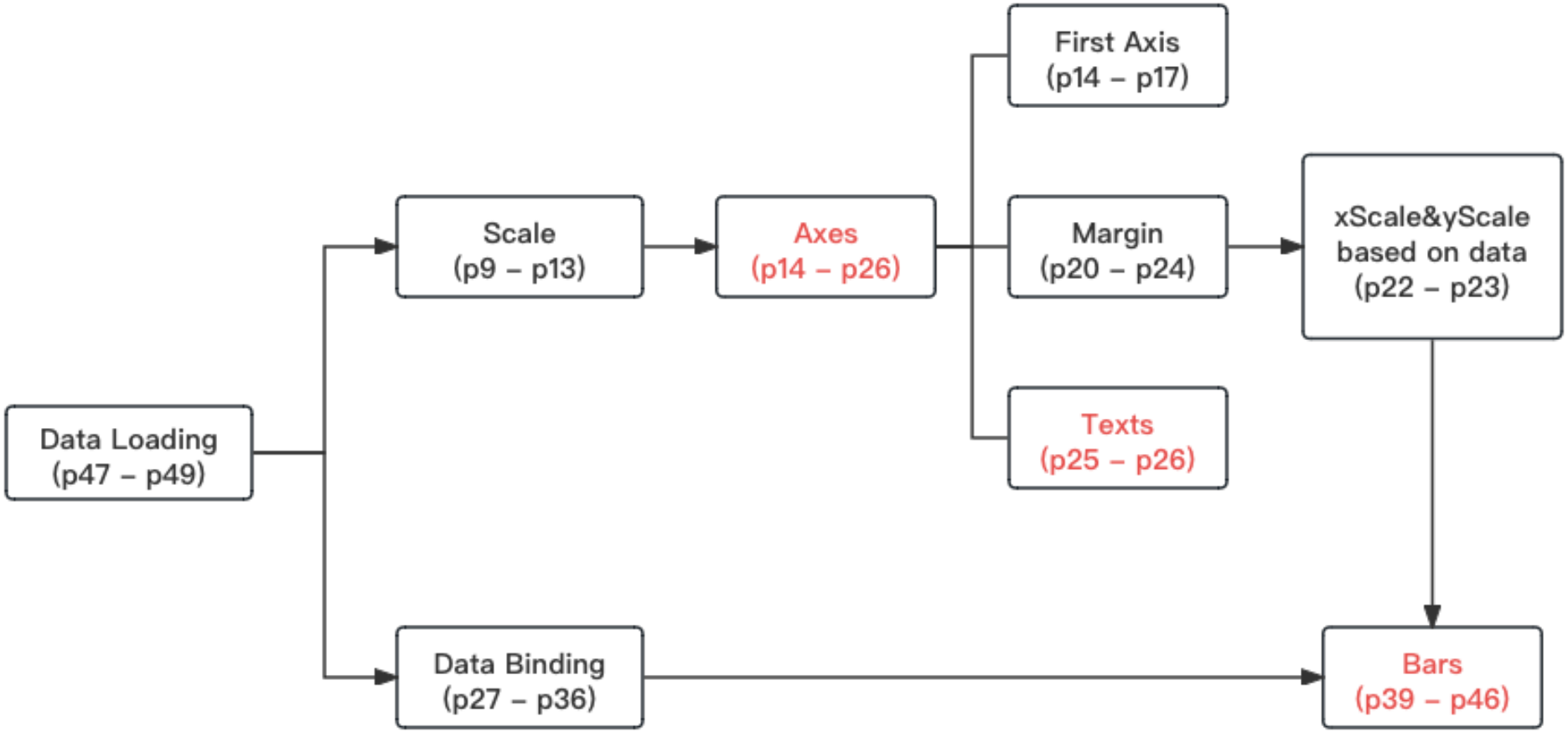
- d3.csv()
  - d3.csv(”path/to/data.csv”).then(data => {…})



Summary

- Scales in D3
- Axes in D3
- Data Binding
- Bar Chart
- Data Loading

Summary


First Axis
(p14 – p17)
Scale
(p9 – p13)
Axes
(p14 – p26)
Margin
(p20 – p24)
xScale&yScale
based on data
(p22 – p23)
Data Loading
(p47 – p49)
Texts
(p25 – p26)
Data Binding
(p27 – p36)
Bars
(p39 – p46)