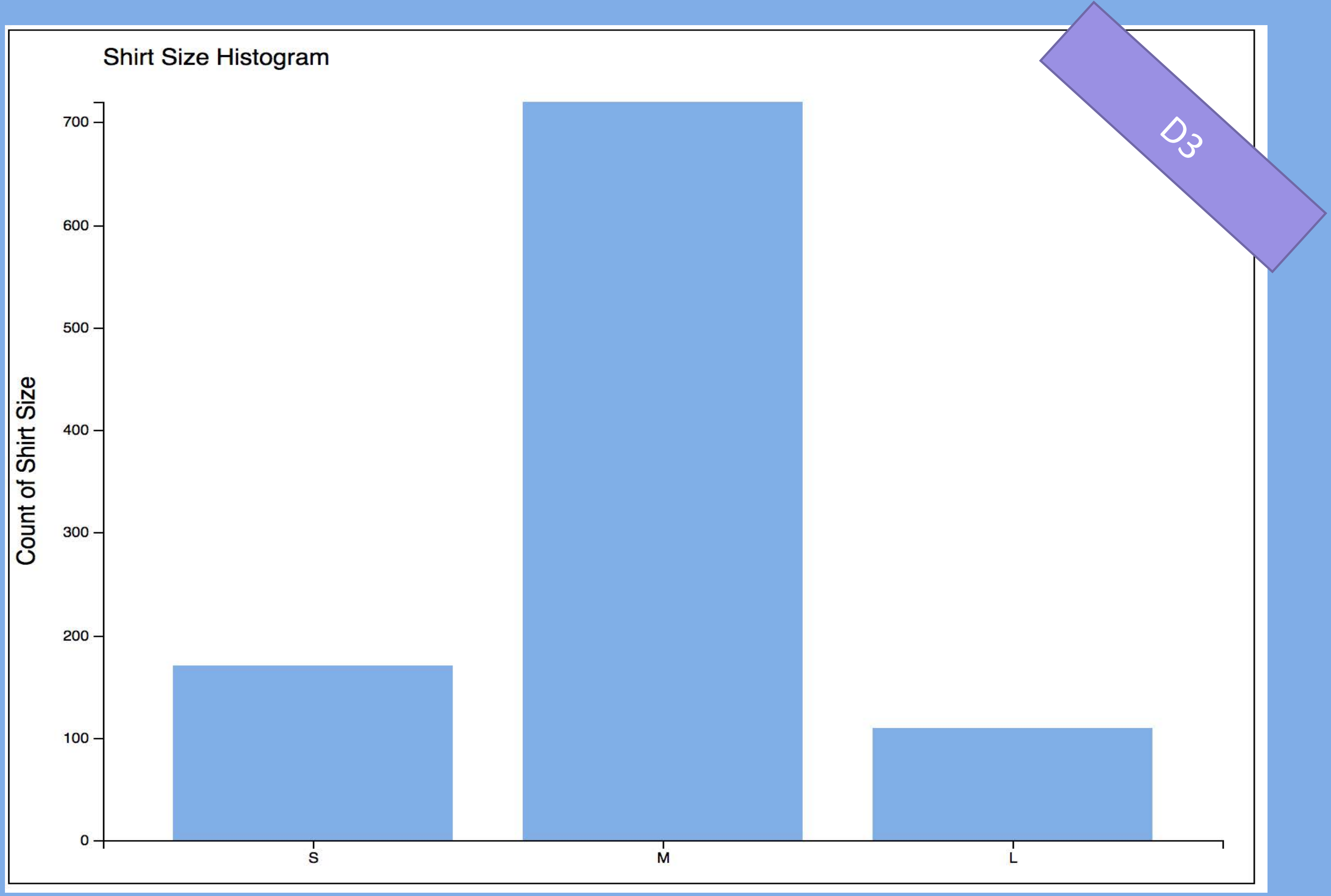
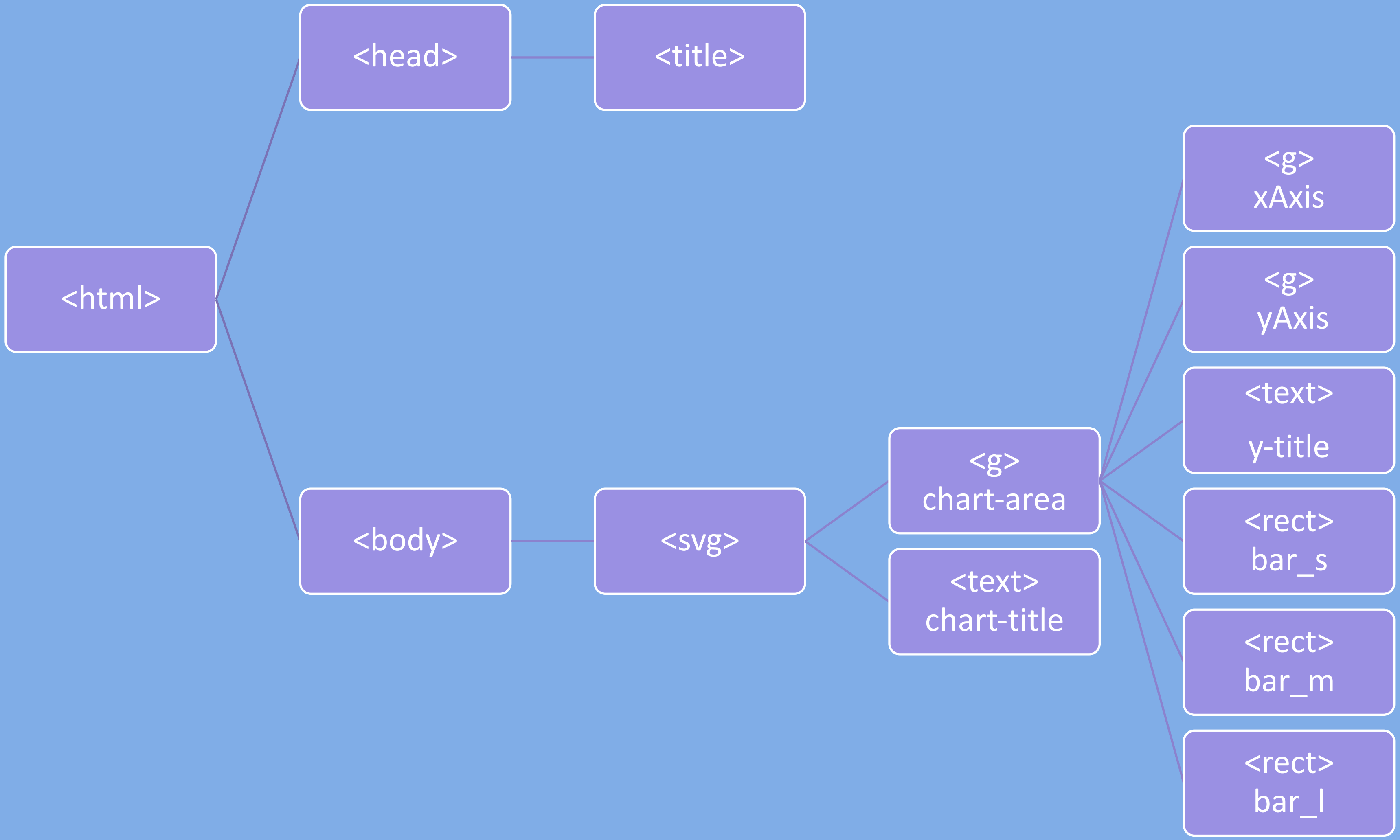
D3 Shirt Size Histogram



Document Object Model



DOM Manipulation – append()

```
<g id="chart-area"></g>
```

append <rect>

```
g.append("rect")
  .attr("width", "300")
  .attr("height", "400");
```

```
<g id="chart-area">
  <rect width="300"
        height="400"/>
</g>
```

DOM Manipulation – attr()

```
<rect x="160" y="10" />
```

literal value

```
.attr("x", 42);
```

data item

```
.attr("x", function (d) {
  return d.weight;
});
```

data item, index

```
.attr("x", function (d, i) {
  return i * 10;
});
```

Arrow Functions

anonymous function

```
.attr("x", function (d) {
  return d.weight;
});
```

arrow function

```
.attr("x", d => d.weight);
.attr("x", (d, i) => { return d.weight; }
```

concise/readable!

Arrow Functions

anonymous function

```
.attr("x", function (d) {
  return xScale(d);
});
```

arrow function

```
.attr("x", d => xScale(d));
```

even simpler

```
.attr("x", xScale);
```

Information Visualization 2021 | Computer Science & iCompetence
Marco Soldati

March 2021

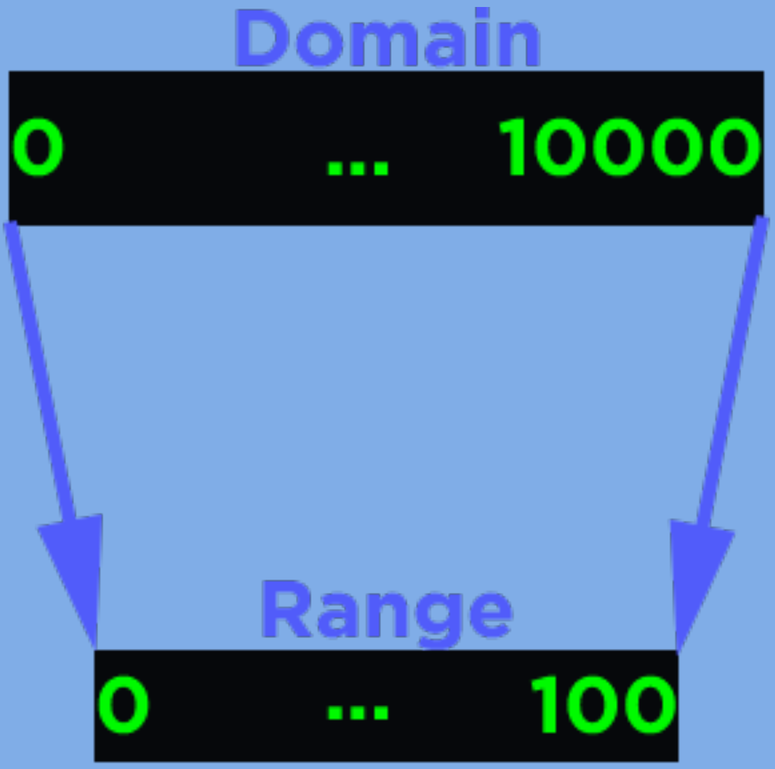
D3 Introduction

03

Selections, Databinding and Events

Recap – Scale

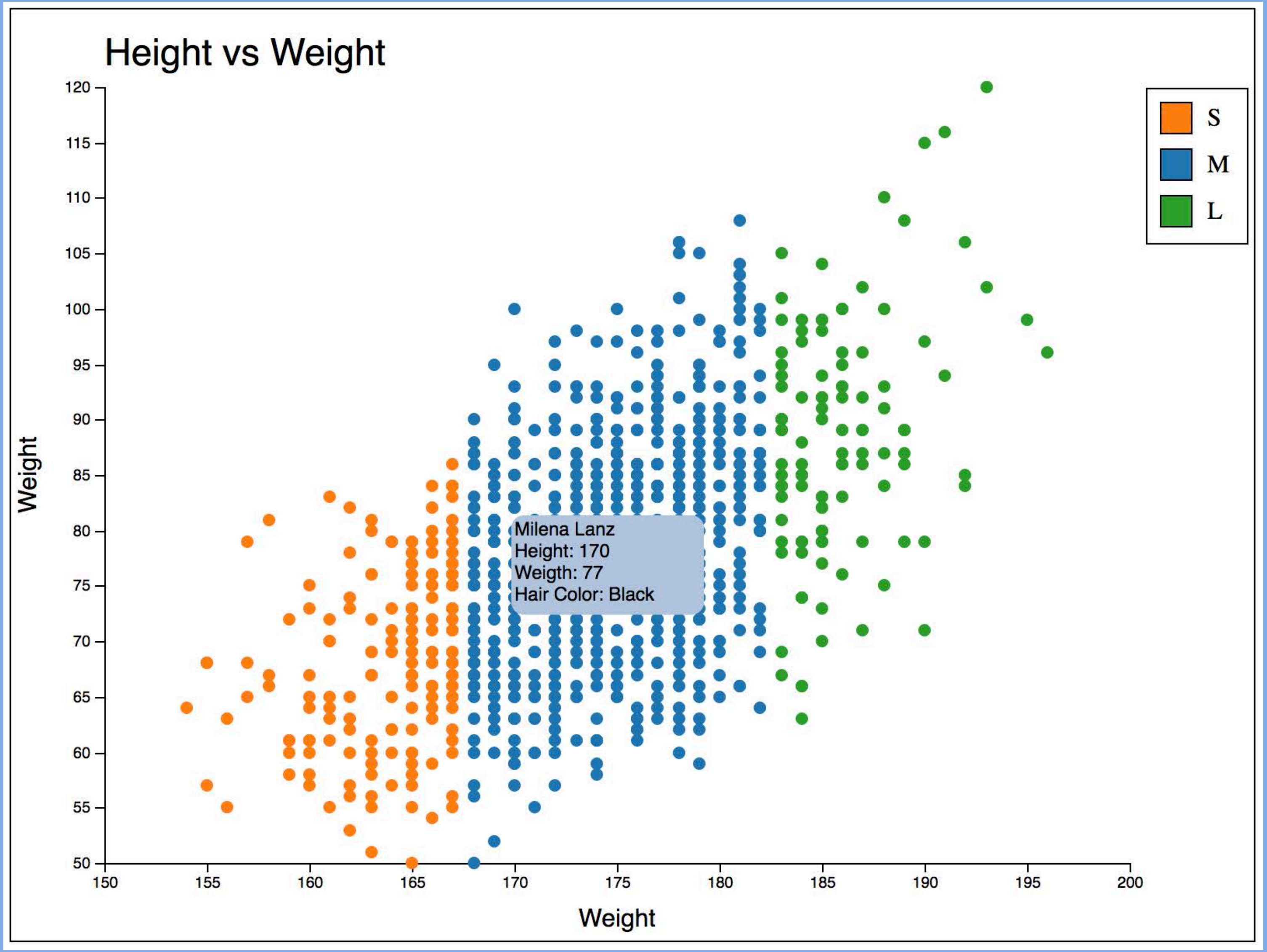
A mapping from values to values – aka a **function:**



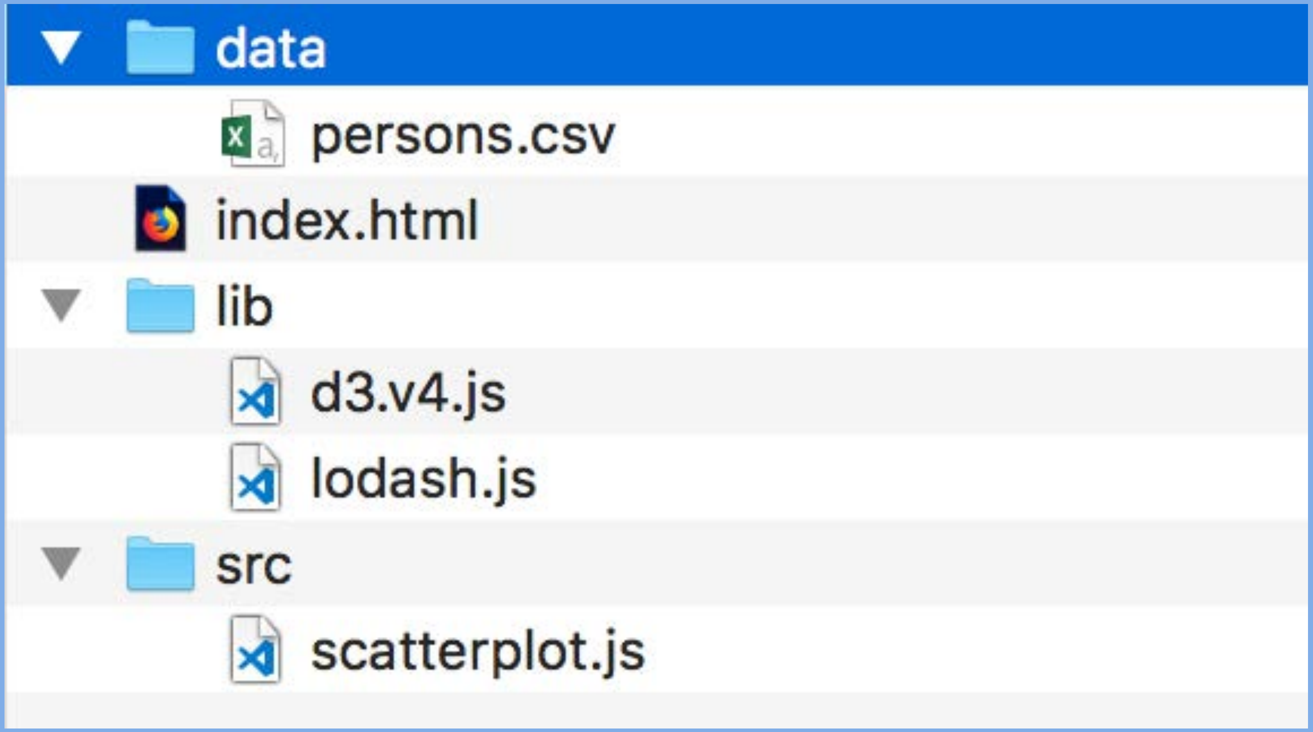
```
var x = d3.scaleLinear()
.domain([0, 10000])
.range([0, 100]);

x(2000); // 20
x(32000); // 320
```

Goal – Interactive Scatterplot



Template Folder



```
▼ data
    persons.csv
  index.html
▼ lib
    d3.v4.js
    lodash.js
▼ src
    scatterplot.js
```

Your turn

- Copy template from Active Directory
- Startup local web server

D3.js

- D3 modules
  - https://github.com/d3
- D3 Gallery
  - https://github.com/d3/d3/wiki/Gallery
- D3 API Reference
  - https://github.com/d3/d3/blob/master/API.md

HTML Template

```html
<!doctype html>
<html>
<head>
  <meta charset="utf-8"/>
</head>
<body>

<!-- put your stuff here -->

  <script src='lib/d3/d3.js'></script>
  <script src='lib/lodash.js'></script>
  <script src='src/scatterplot.js'></script>
</body>
</html>
```

scatterplot.js – SVG fragment

```javascript
// create svg canvas
const canvHeight = 600, canvWidth = 800;
const svg = d3.select("body").append("svg")
    .attr("width", canvWidth)
    .attr("height", canvHeight)
    .style("border", "1px solid");

// calc the width and height depending on margins.
const margin = {top: 50, right: 20, bottom: 50, left: 60};
const width = canvWidth - margin.left - margin.right;
const height = canvHeight - margin.top - margin.bottom;

// create parent group and add left and top margin
const g = svg.append("g")
    .attr("id", "chart-area")
    .attr("transform", "translate(" +margin.left + ","
      + margin.top + ")");
```

D3 – .select() and .selectAll()

Name	Behaviour	Example
.style()	Update the style (css)	d3.selectAll('circle').style('fill', 'red')
.attr()	Update an attribute (html)	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('.checkbox').property('checked', false)
.text()	Update the text content	d3.select('div.title').text('My new book')
.html()	Change the html content	d3.select(‘#chart1').html(‘<h1>A new chart</h1>')

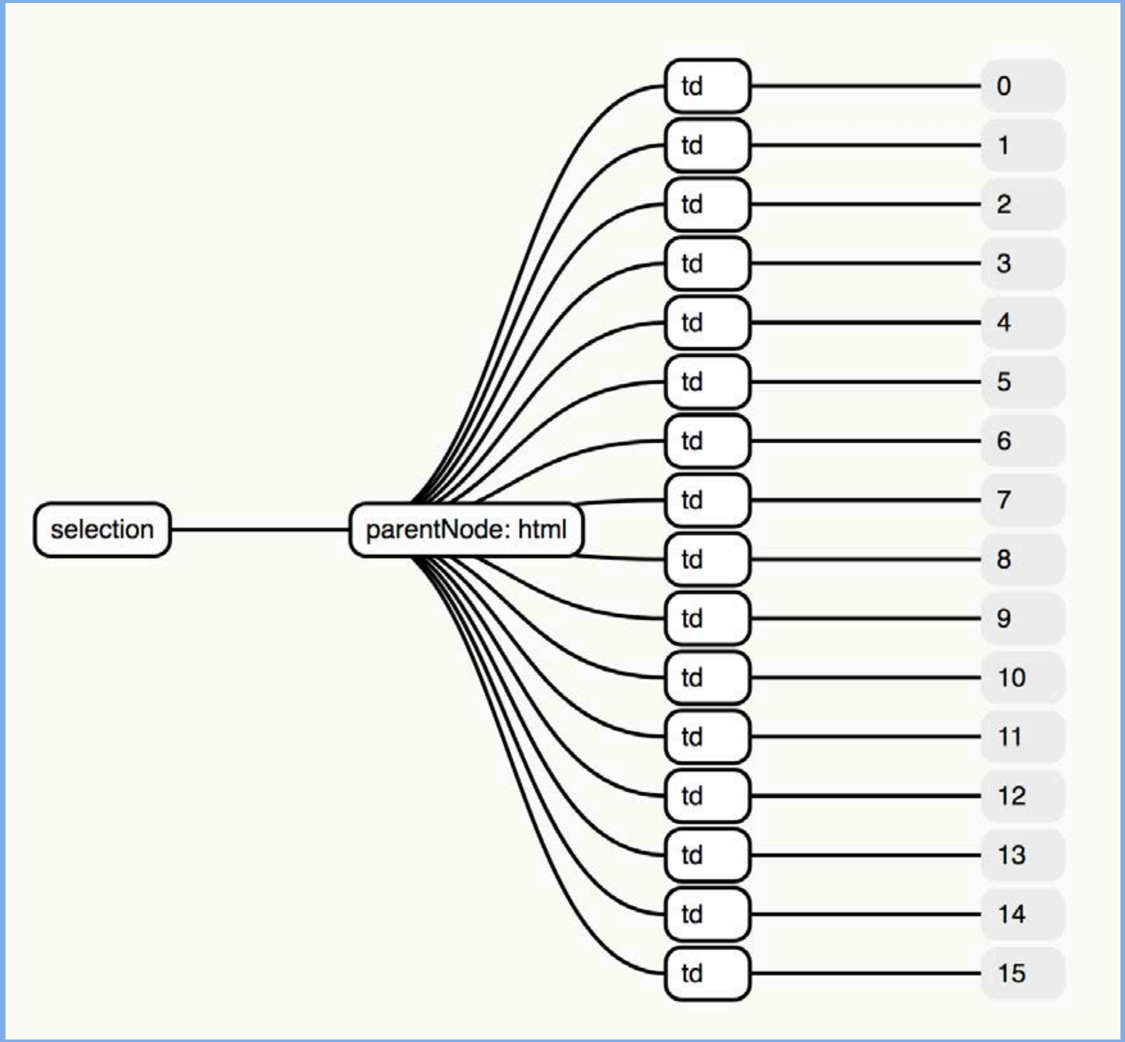
D3 – Nested Selections

```html
<table>
  <thead>
    <tr><td> A</td><td> B</td><td> C</td><td> D</td></tr>
  </thead>
  <tbody>
    <tr><td> 0</td><td> 1</td><td> 2</td><td> 3</td></tr>
    <tr><td> 4</td><td> 5</td><td> 6</td><td> 7</td></tr>
    <tr><td> 8</td><td> 9</td><td> 10</td><td> 11</td></tr>
    <tr><td> 12</td><td> 13</td><td> 14</td><td> 15</td></tr>
  </tbody>
</table>
```

```javascript
var td = d3.selectAll("tbody td");
var td = d3.select("tbody").selectAll("td");
var td = d3.selectAll("tbody tr").selectAll("td");
```

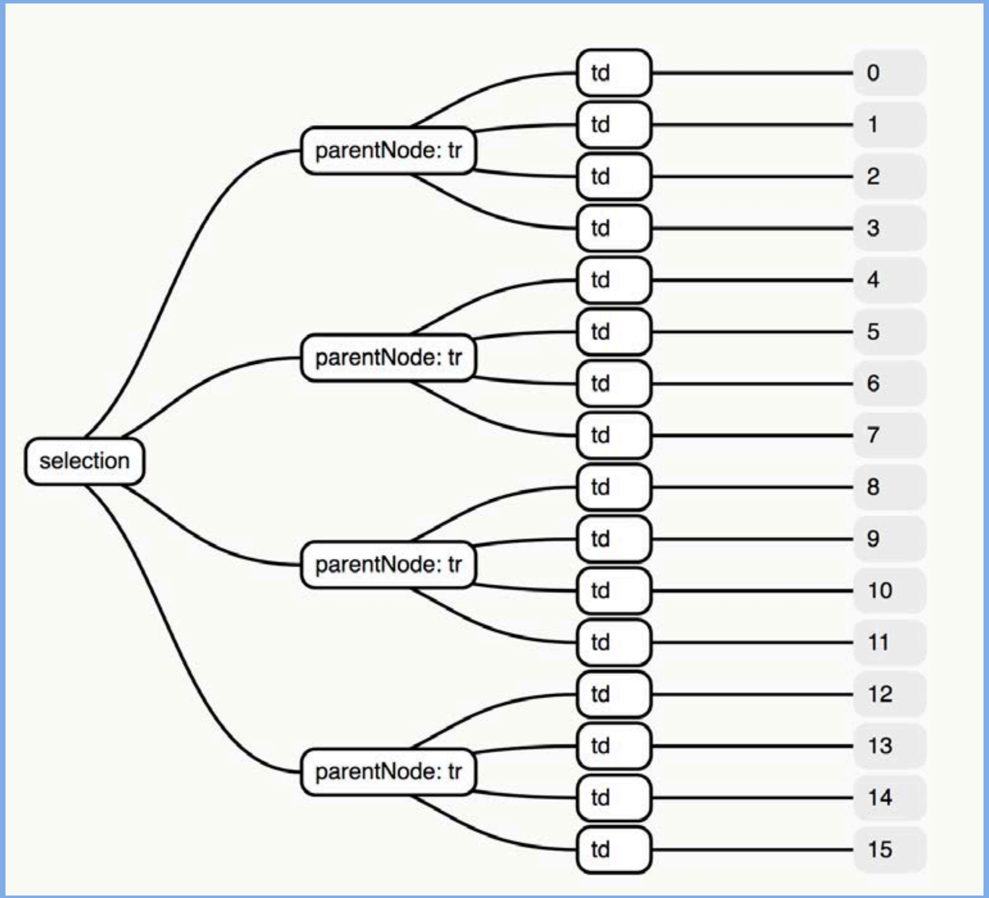
D3 – Nested Selections

```
var td = d3.selectAll("tbody td");
```



D3 – Nested Selections

```
var td = d3.selectAll("tbody tr").selectAll("td");
```



```
td.style("color", function(d, i) => i  % 2 ? null : "red"; }); //col
td.style("color", function(d, i, j) => j  % 2 ? null : ”blue"; }); //row
```

scatterplot.js – Text labels

```
// chart title
svg.append("text")
  .attr("y", 0)
  .attr("x", margin.left)
  .attr("dy", "1.5em")
  .attr("font-family", "sans-serif")
  .attr("font-size", "24px")
  .style("text-anchor", "left")
  .text("Height vs Weight");
```

as HTML

```
<text y="0" x="60" dy="1.5em"
      font-family="sans-serif" font-size="24px">
  Height vs Weight
</text>
```

scatterplot.js – Fetch Data

v3	d3-request
v4	d3-fetch

scatterplot.js – Fetch Data

```
// load the data from the cleaned csv file.
// note: the call is done asynchronous.
// That is why you have to load the data inside of a
// callback function.
d3.csv("./data/persons.csv").then(function(data) {
  const heightDomain = d3.extent(data, d => Number(d.Height));
  const weightDomain = d3.extent(data, d => Number(d.Weight));
  ...
}
```

https://github.com/d3/d3-fetch

scatterplot.js – Data Statistics

Methods for computing basic summary statistics.

- d3.min - compute the minimum value in an array.
- d3.max - compute the maximum value in an array.
- d3.extent - compute the minimum and maximum value in an array.
- d3.sum - compute the sum of an array of numbers.
- d3.mean - compute the arithmetic mean of an array of numbers.
- d3.median - compute the median of an array of numbers (the 0.5-quantile).
- d3.quantile - compute a quantile for a sorted array of numbers.
- d3.variance - compute the variance of an array of numbers.
- d3.deviation - compute the standard deviation of an array of numbers.

https://github.com/d3/d3-array

scatterplot.js – Data Conversion

JavaScript data types

- string ← String() converts to string
- number ← Number() converts to number
- boolean ← Boolean () converts to boolean
- object
  - Object
  - Date
  - Array
- function

scatterplot.js – Scales

```
// create scales for x and y direction
const xScale = d3.scaleLinear()
  .domain(heightDomain)
  .rangeRound([0,width])
  .nice(5);

const yScale = d3.scaleLinear()
  .domain(weightDomain)
  .rangeRound([height,0])
  .nice(5);

const colorScale = d3.scaleOrdinal(d3.schemeCategory10);
```

scatterplot.js – Axis

```
// create xAxis
const xAxis = d3.axisBottom(xScale);
g.append("g") // create a group and add axis
  .attr("transform", "translate(0," + height + ")").call(xAxis);

// create yAxis
const yAxis = d3.axisLeft(yScale);
g.append("g") // create a group and add axis
  .call(yAxis);
```

scatterplot.js – Data Points

```
// add circle
g.selectAll("circle") // this results in an empty selection
  .data(data) // which is joined with the data
  .enter() // and a selection of new elements is created
  .append("circle")
  .attr("cx", d => xScale(d.Height))
  .attr("cy", d => yScale(d.Weight))
  .attr("r", 4)
  .style("fill", d => colorScale(d["Shirt Size"]));
```

Data Joins – .data()

DOM elements

```
<circle r="40" />
<circle r="40" cx="120" />
<circle r="40" cx="240" />
<circle r="40" cx="360" />
<circle r="40" cx="480" />
```

Data

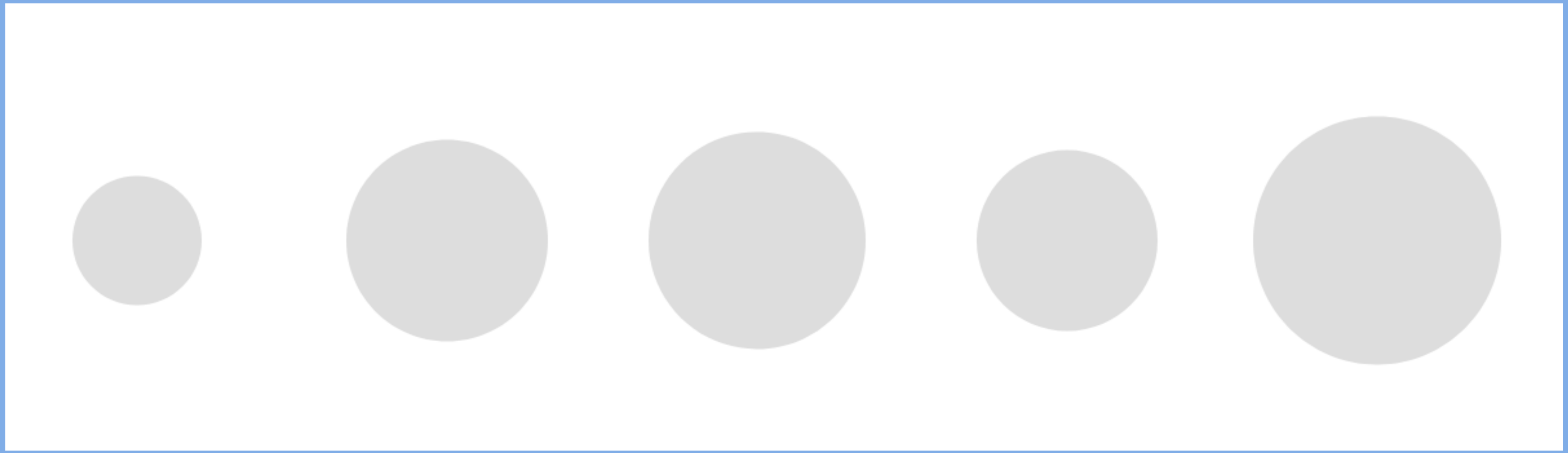
```
var scores = [
  { "name": "Andy", "score": 25},
  { "name": "Beth", "score": 39},
  { "name": "Craig", "score": 42},
  { "name": "Diane", "score": 35},
  { "name": "Evelyn", "score": 48}
]
```

Join

```
d3.selectAll('circle’)
  .data(scores);
```

Data Joins – .data()

```
d3.selectAll('circle’)
  .data(scores)
  .attr('r', d => d.score);
```



.enter() and .exit()

If the data array is **longer** than the DOM selection
there is a **shortfall** of DOM elements and
we need to **add** elements
→ **.data().enter(**) is called

If the data array is **shorter** than the DOM selection
there is a **surplus** of DOM elements and
we need to **remove** elements
→ **.data().exit()** is called

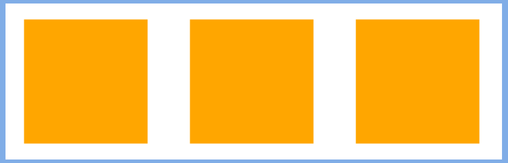
.enter()

Data

```
var myData = ['A', 'B', 'C', 'D', 'E'];
```

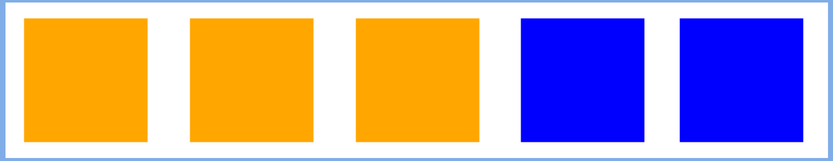
DOM

```
<div id="content">
  <div></div>
  <div></div>
  <div></div>
</div>
```



Join

```
d3.select('#content’)
  .selectAll('div’)
  .data(myData)
  .enter()
  .append('div’)
  .style("background-color", "blue");
```



https://bl.ocks.org/d3indepth/1450ba61704293ce528646a1cf02fd85

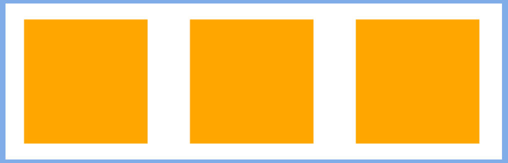
.exit()

Data

```
var myData = ['A'];
```

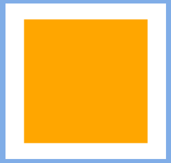
DOM

```
<div id="content">
  <div></div>
  <div></div>
  <div></div>
</div>
```



Join

```
d3.select('#content’)
  .selectAll('div’)
  .data(myData)
  .exit()
  .remove();
```



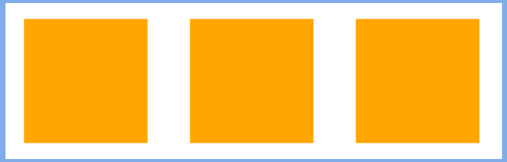
.merge()

Data

```
var myData = ['A', 'B', 'C', 'D', 'E'];
```

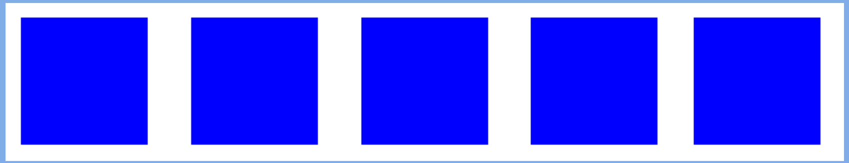
DOM

```
<div id="content">
  <div></div>
  <div></div>
  <div></div>
</div>
```



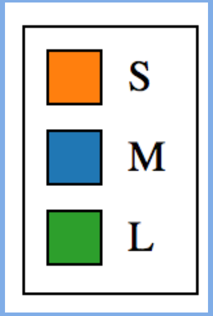
Join

```
var u = d3.select('#content’)
  .selectAll('div’)
  .data(myData);
u.enter()
  .append('div’)
  .merge(u)
  .style("background-color", "blue");
```



scatterplot.js - Legend

```
legendDomain = ["S", "M", "L"];
createLegend(legendDomain, colorScale);
```

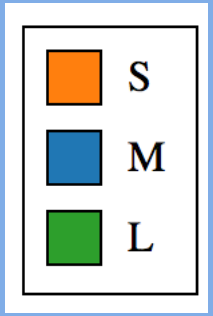


```
function createLegend(legendDomain, colorScale) {
  // 1. create a group to hold the legend
  // 2. create the legend boxes and the text label
  //    use .data(legendDomain) on an empty DOM selection
  // 3. create the main border of the legend
}
```

Your turn

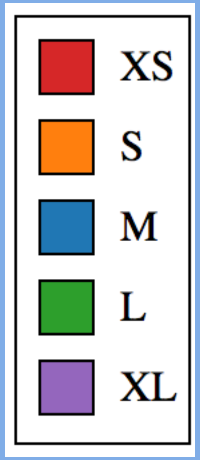
- Implement createLegend()

```
function createLegend(legendDomain, colorScale) {
  // 1. create a group to hold the legend
  // 2. create the legend boxes and the text label
  //    use .data(legendDomain) on an empty DOM selection
  // 3. create the main border of the legend
}
```



- Test with

```
legendDomain = ["XS", "S", "M", "L", "XL"];
```



Events – .on()

```
d3.selectAll('circle’)
  .on('click', function(event, d) {

      d3.select('.status’)
        .text('You clicked on ‘ + d.Name);
  });
```

Events

Event name	Description
click	Element has been clicked
mouseenter	Mouse pointer has moved onto the element
mouseover	Mouse pointer has moved onto the element or its children
mouseleave	Mouse pointer has moved off the element
mouseout	Mouse pointer has moved off the element or its children
mousemove	Mouse pointer has moved over the element

See https://developer.mozilla.org/en-US/docs/Web/Events#Standard_events for a full list of events

Your turn

Create a Tooltip with contextual information

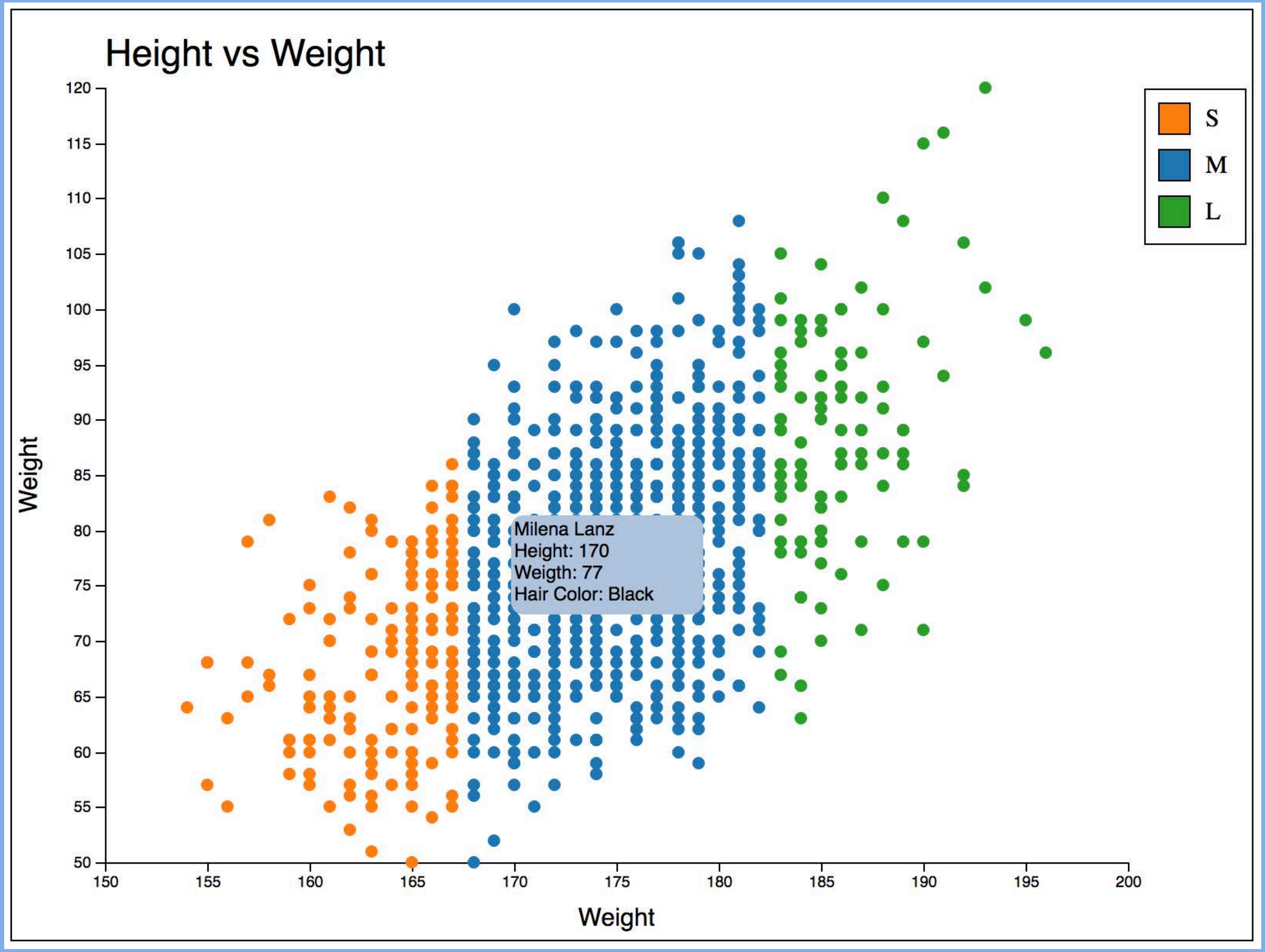
1. Create a tooltip div (style it with CSS from next slide)
2. Add “mouseover” event to every circle
    1. Display tooltip at mouse position
       (use .style(“left”, x) and .style(“top”, y)
    2. Create tooltip content (use .html())
3. Add “mouseout” event to every circle
    1. Hide tooltip (use attr(“visibility”, “hidden”))

Your turn

CSS for Tooltip <div class="“tooltip”>

```
div.tooltip {
  position: absolute;
  text-align: left;
  width: 80px;
  height: 60px;
  padding: 2px;
  font: 12px sans-serif;
  background: lightsteelblue;
  border: 0px;
  border-radius: 8px;
  pointer-events: none;
}
```

Interactive Scatterplot



Milena Lanz
Height: 170
Weigth: 77
Hair Color: Black