Lecture 9
D3: Colour, Animation & Interaction

DTS204TC Data Visualisation

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

Outline

- Colour (p3 – p17)
    - opacity
    - APIs
    - d3.scaleOrdinal()
- Animation (p18 - p36)
    - transition()
        - duration
        - ease
        - delay
- Interaction (p37 – p46)
    - event Listener
    - tip

Colour

- Colour in D3: “fill”
  - Define the colour by ourselves
  - Use the colour APIs in D3
  - Sampling

Colour

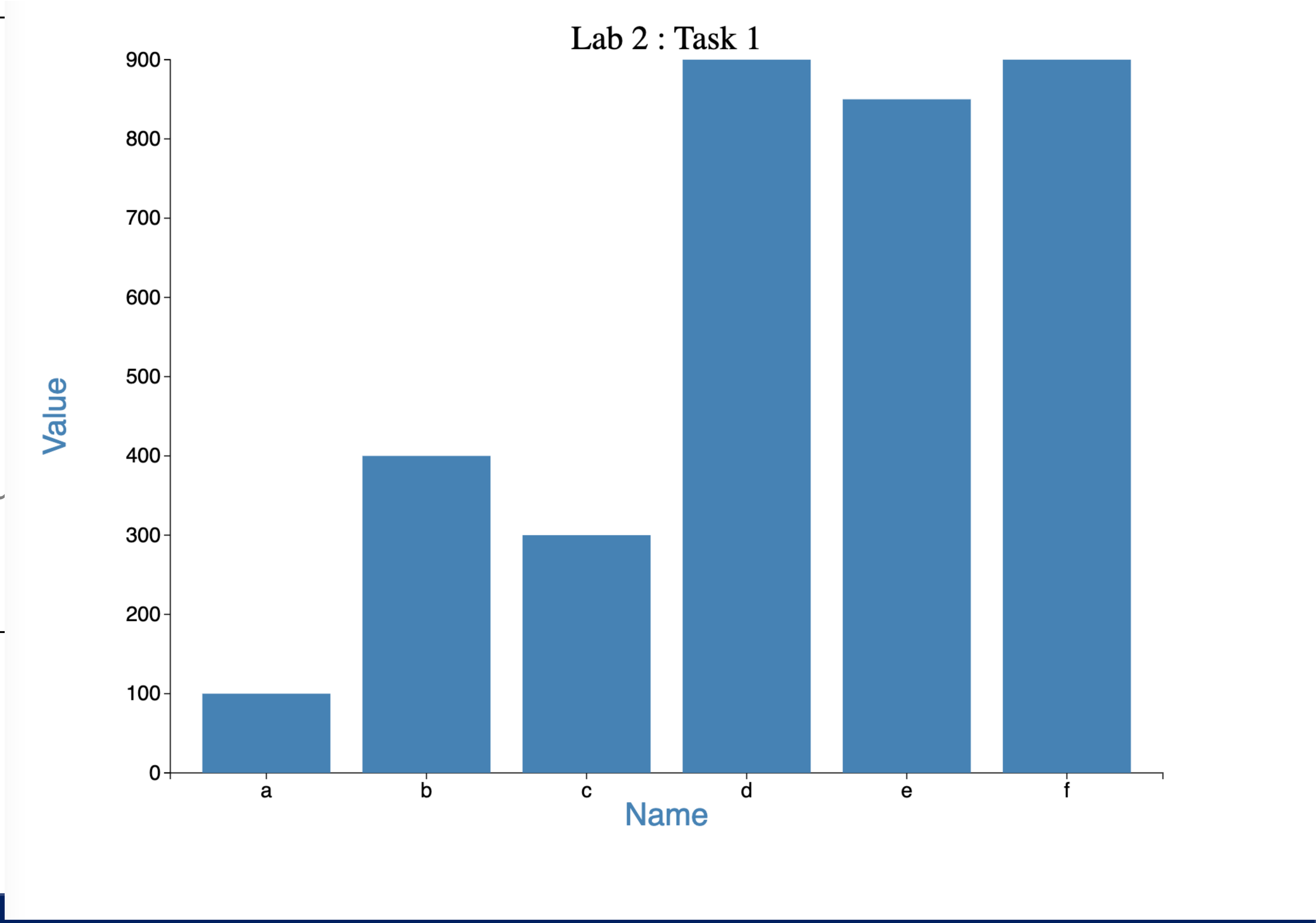
- Define the colour by ourselves: Name

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.value))
.attr("fill", "steelblue");
```

Colour

- Define the colour by ourselves: Name

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.valu
.attr("fill", "steelblue");
```



Colour

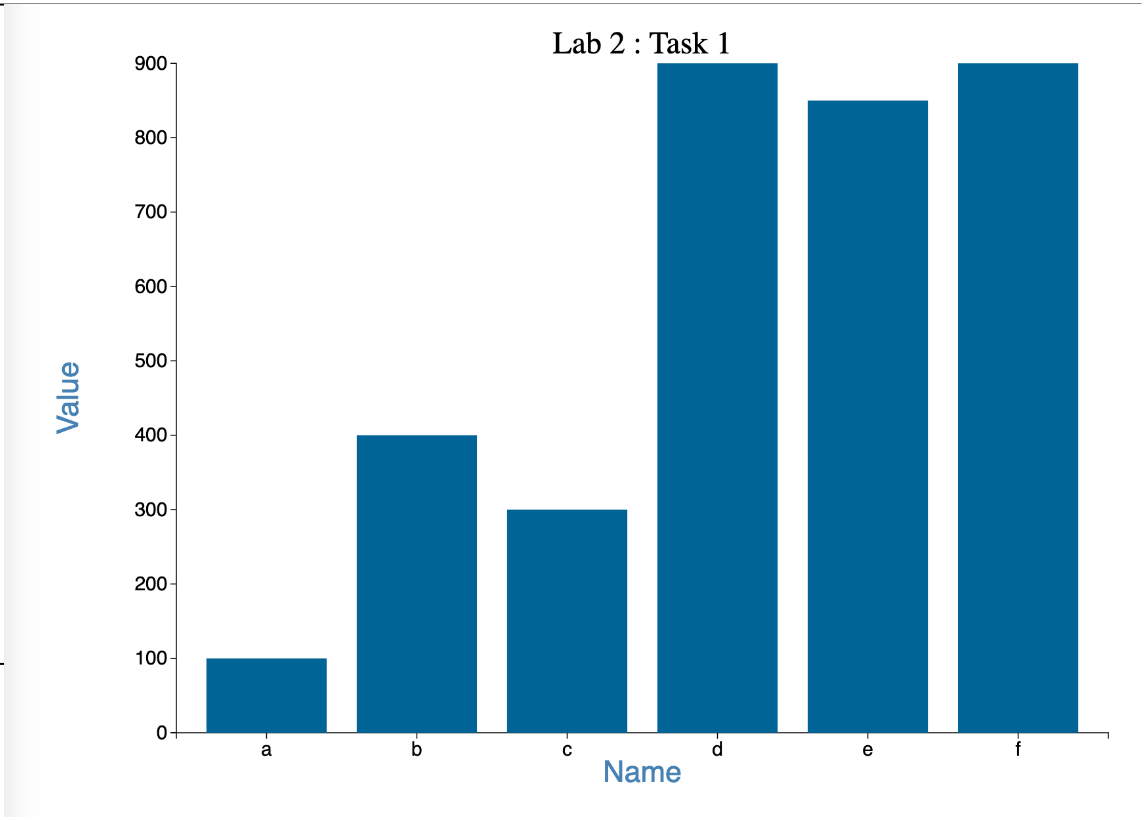
- Define the colour by ourselves: RGB

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.value))
.attr("fill", d3.rgb(0,100,150));
```

Colour

- Define the colour by ourselves: RGB

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.value))
.attr("fill", d3.rgb(0,100,150));
```



Colour

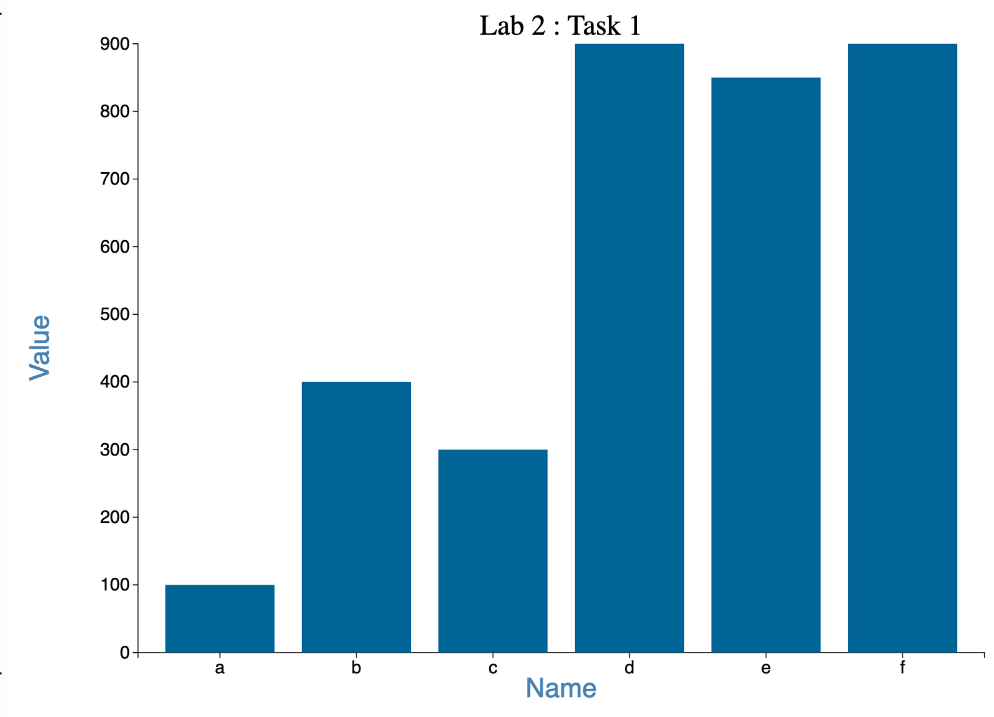
- Define the colour by ourselves: HEX codes
  - https://htmlcolorcodes.com

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.value))
.attr("fill", "#006496");
```

Colour

- Define the colour by ourselves: HEX codes

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.value))
.attr("fill", "#006496");
```



Colour

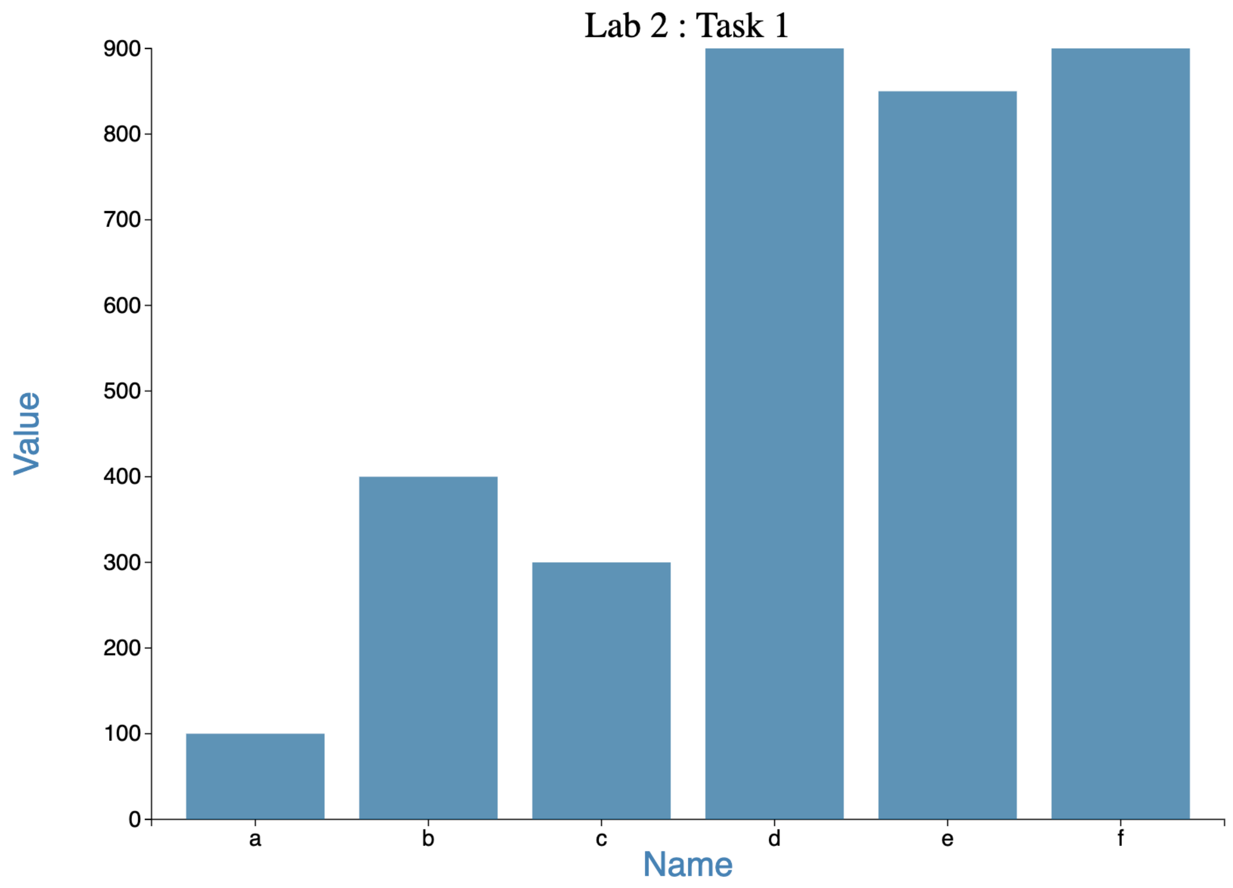
- Define the colour by ourselves: “opacity”

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.value))
.attr("fill", "#006496")
.attr("opacity", "0.7");
```

Colour

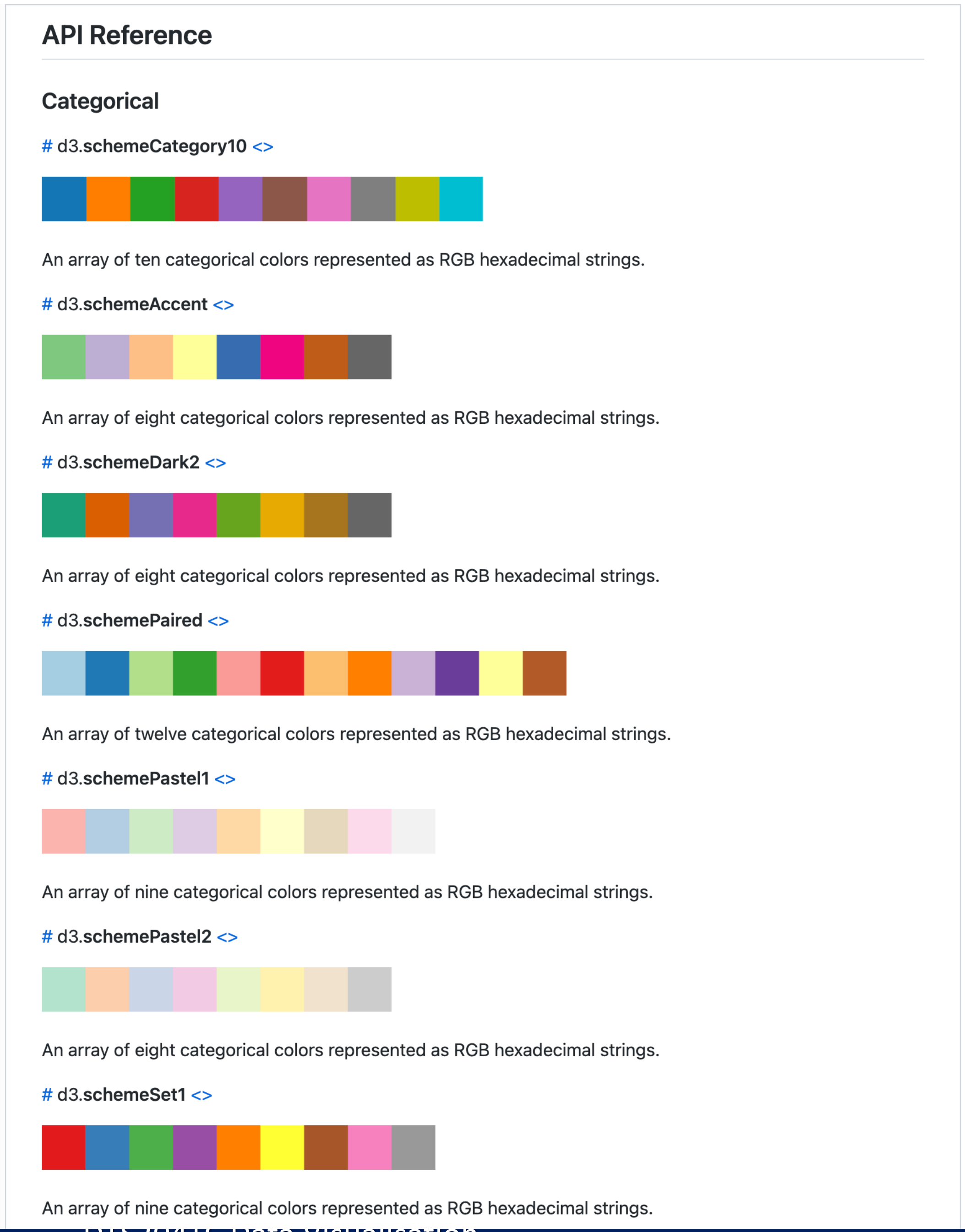
- Define the colour by ourselves: “opacity”

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.valu
.attr("fill", "#006496")
.attr("opacity", "0.7");
```



Colour

- Colour APIs:
  - **d3.scaleOrdinal()**
  - https://github.com/d3/d3-scale-chromatic



API Reference

Categorical

d3.schemeCategory10 <>

An array of ten categorical colors represented as RGB hexadecimal strings.

d3.schemeAccent <>

An array of eight categorical colors represented as RGB hexadecimal strings.

d3.schemeDark2 <>

An array of eight categorical colors represented as RGB hexadecimal strings.

d3.schemePaired <>

An array of twelve categorical colors represented as RGB hexadecimal strings.

d3.schemePastel1 <>

An array of nine categorical colors represented as RGB hexadecimal strings.

d3.schemePastel2 <>

An array of eight categorical colors represented as RGB hexadecimal strings.

d3.schemeSet1 <>

An array of nine categorical colors represented as RGB hexadecimal strings.

Colour

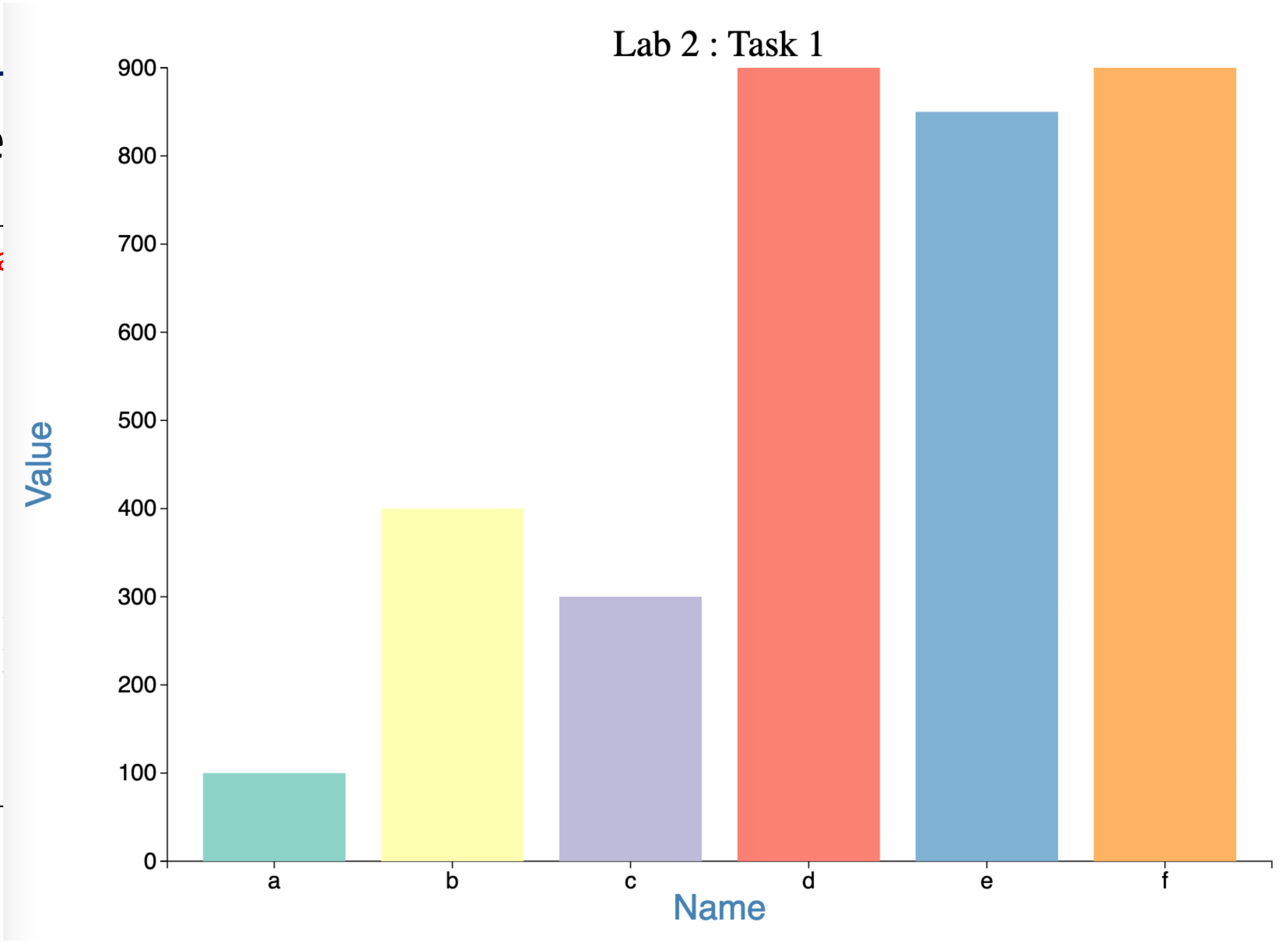
- Colour APIs: d3.scaleOrdinal()

```
const colorScale = d3.scaleOrdinal(d3.schemeSet3);
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth())
.attr("height",d => innerHeight - yScale(d.value))
.attr("fill", (d,i)=>colorScale(i));
```

Colour

- Colour APIs: d3.scale

```
const colorScale = d3.scaleOrdina
//draw bars
g.selectAll(".bar")
.data(data1)
.enter().append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("y",d => yScale(d.value))
.attr("width",xScale.bandwidth()
.attr("height",d => innerHeight -
.attr("fill", (d,i)=>colorScale(i));
```



Colour

- Colour APIs: d3.scaleOrdinal() more
    - const colourScale = d3.scaleOrdinal().domain().range();

Colour

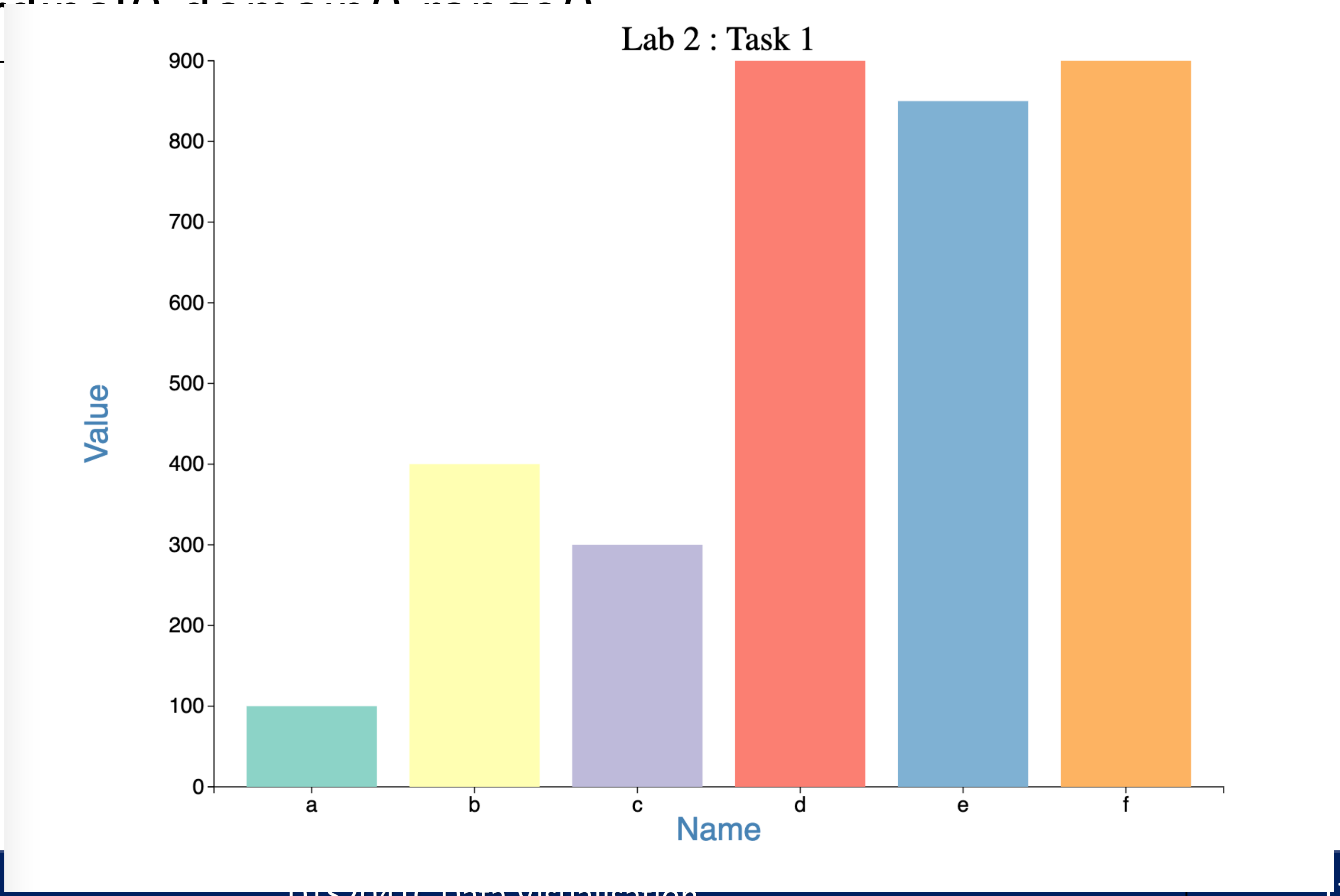
- Colour APIs: d3.scaleOrdinal() more
  - const colourScale = d3.scaleOrdinal().domain().range();

```
…
// Scale
…
const colourScale = d3.scaleOrdinal()
.domain(data1.map(d => d.name))
.range(d3.schemeSet3);

//axis
…

// draw bars
.attr("fill", d => colourScale(d.name))
…
```

Colour

- Colour APIs: d3.scaleOrdinal() more
  - const colourScale = d3.scaleOr

```
…
// Scale
…
const colourScale = d3.scaleOrdinal()
.domain(data1.map(d => d.name))
.range(d3.schemeSet3);

//axis
…

// draw bars
.attr("fill", d => colourScale(d.name))
…
```



Animation

- transition()
  - The **d3.selection.transition()** method indicates the start of transition and then different transition functions can be applied to the selected elements.

Animation

Method	Description
selection.transition()	this schedules a transition for the selected elements
transition.duration()	duration specifies the animation duration in milliseconds for each element
transition.ease()	ease specifies the easing function, example: linear, elastic, bounce
transition.delay()	delay specifies the delay in animation in milliseconds for each element

Animation

- transition().duration() :

```
<body>
        <svg width="960" height="400" id="mainsvg"
        class="svgs" style='display: block; margin: 0 auto; '>
                <rect id="my_rect"
                x="10" y="200" width="100" height="30"
                stroke="black" fill="#69b3a2" stroke-width="1"
                > </rect>
        </svg>
        <script>
                d3.select("#my_rect")
                .transition().duration(4000)
                .attr("width", "400");
        </script>
</body>
```

Animation

- transition().duration() :

```
<body>
        <svg width="960" height="400" id="mainsvg"
        class="svgs" style='display: block; margin: 0 auto;'>
                <rect id="my_rect"
                x="10" y="200" width="100" height="30"
                stroke="black" fill="#69b3a2" stroke-width="1"
                > </rect>
        </svg>
        <script>
                d3.select("#my_rect")
                .transition().duration(4000)
                .attr("width", "400");
        </script>
</body>
```

Animation

- transition().duration() :

```
<body>
        <svg width="960" height="400" id="mainsvg"
        class="svg
        </svg>
        <script>
        </script>
</body>
```

Animation

- transition().ease()

```
<body>
        <svg width="960" height="400" id="mainsvg"
        class="svgs" style='display: block; margin: 0 auto;’>
                <rect id="my_rect"
                x="10" y="200" width="100" height="30"
                stroke="black" fill="#69b3a2" stroke-width="1"
                > </rect>
        </svg>
        <script>
                d3.select("#my_rect")
                .transition()
                .ease(d3.easeBounce)
                .duration(4000)
                .attr("width", "400");
        </script>
</body>
```

Animation

- transition().ease()
  - The ease() function is used to specify and control the motion of the transition.

Animation

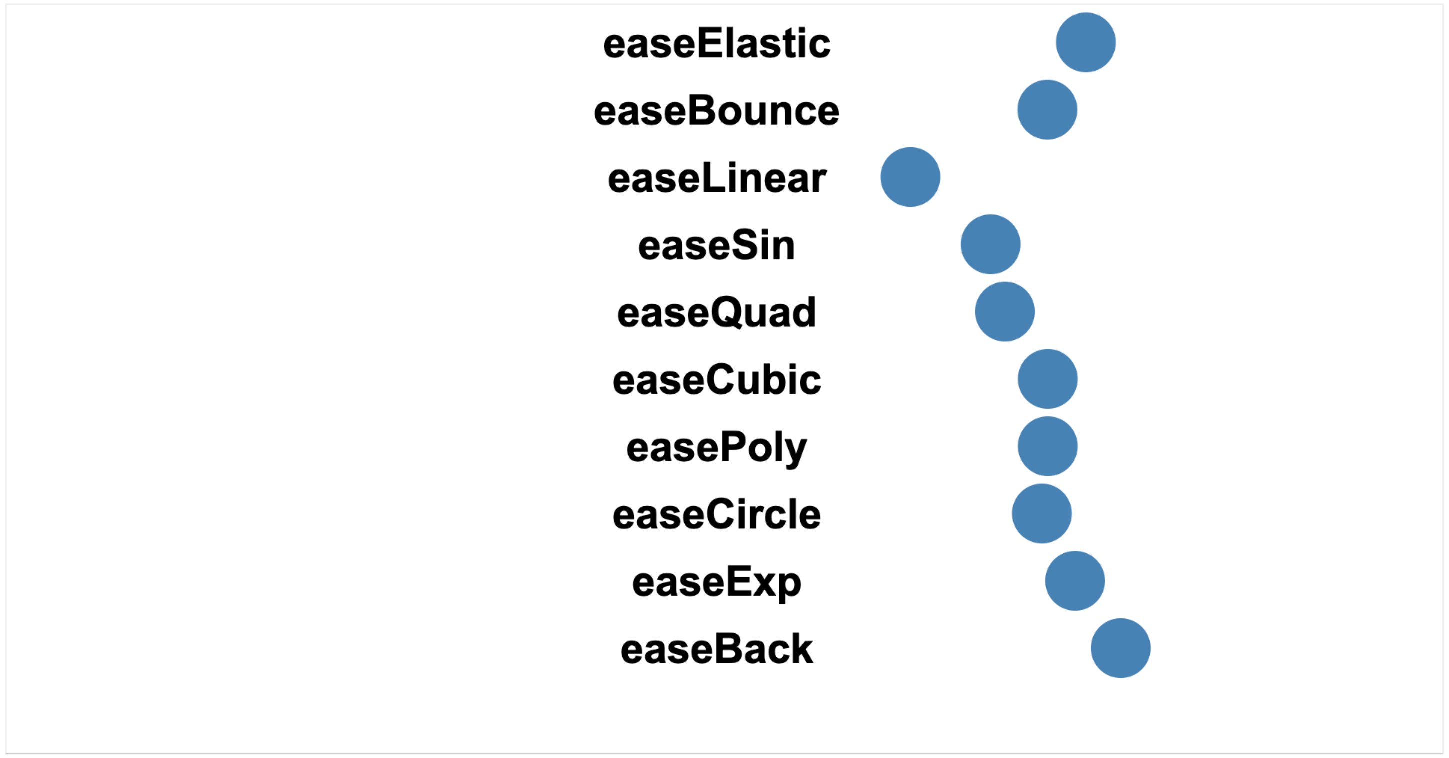
- **transition().ease()**

```
<body>
        <svg width="960" height="400" id="mainsvg"
        class="svg

        </svg>
        <script>

                .attr("width", "400");
        </script>
</body>
```

Animation

- transition().ease()
  - easeElastic
  - easeBounce
  - easeLinear
  - easeSin
  - easeQuad
  - easeCubic
  - easePoly
  - easeCircle
  - easeExp
  - easeBack
  - https://bl.ocks.org/d3noob/dcc534640631fee6ad32604b884f3856

Transition Easing Comparison in v7



Animation

- transition().delay()
  - The delay() function sets the delay parameter for each element in the selection on which the transition is applied. The transition will start after the specified delay value.

Animation

- transition().delay()

```
<body>
        <svg width="960" height="400" id="mainsvg"
        class="svgs" style='display: block; margin: 0 auto; '>
                <rect id="my_rect"
                x="10" y="200" width="100" height="30"
                stroke="black" fill="#69b3a2" stroke-width="1"> </rect>
                <rect id="my_rect1"
                x="10" y="250" width="100" height="30"
                stroke="black" fill="#69b3a2" stroke-width="1"> </rect>
        </svg>
…
```

Animation

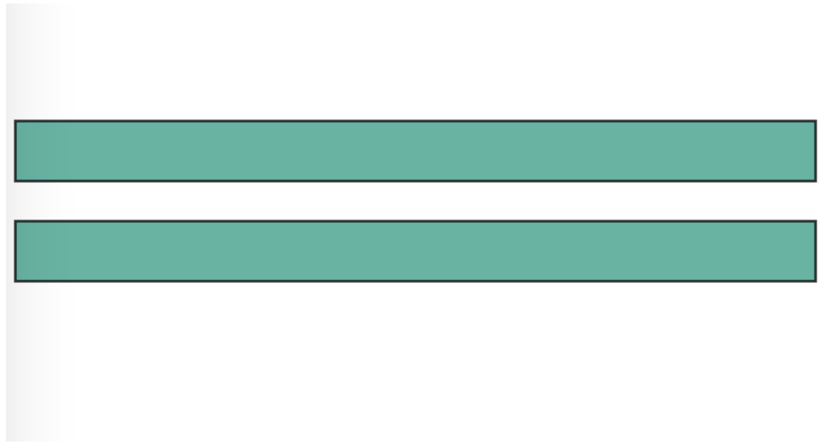
- transition().delay()

```
…
            <script>
                        d3.select("#my_rect")
                        .transition()
                        .ease(d3.easeBounce)
                        .duration(4000)
                        .attr("width", "400");

                        d3.select("#my_rect1")
                        .transition()
                        .ease(d3.easeBounce)
                        .duration(4000)
                        .delay(4000)
                        .attr("width", "400");
            </script>
</body>
```

Animation

- transition().delay()

```
…
        <script>
                d3.select("#my_rect")
                .transition()
                .ease(d3.easeBounce)
                .duration(4000)
                .attr("width", "400");

                d3.select("#my_rect1")
                .transition()
                .ease(d3.easeBounce)
                .duration(4000)
                .delay(4000)
                .attr("width", "400");
        </script>
</body>
```



Animation

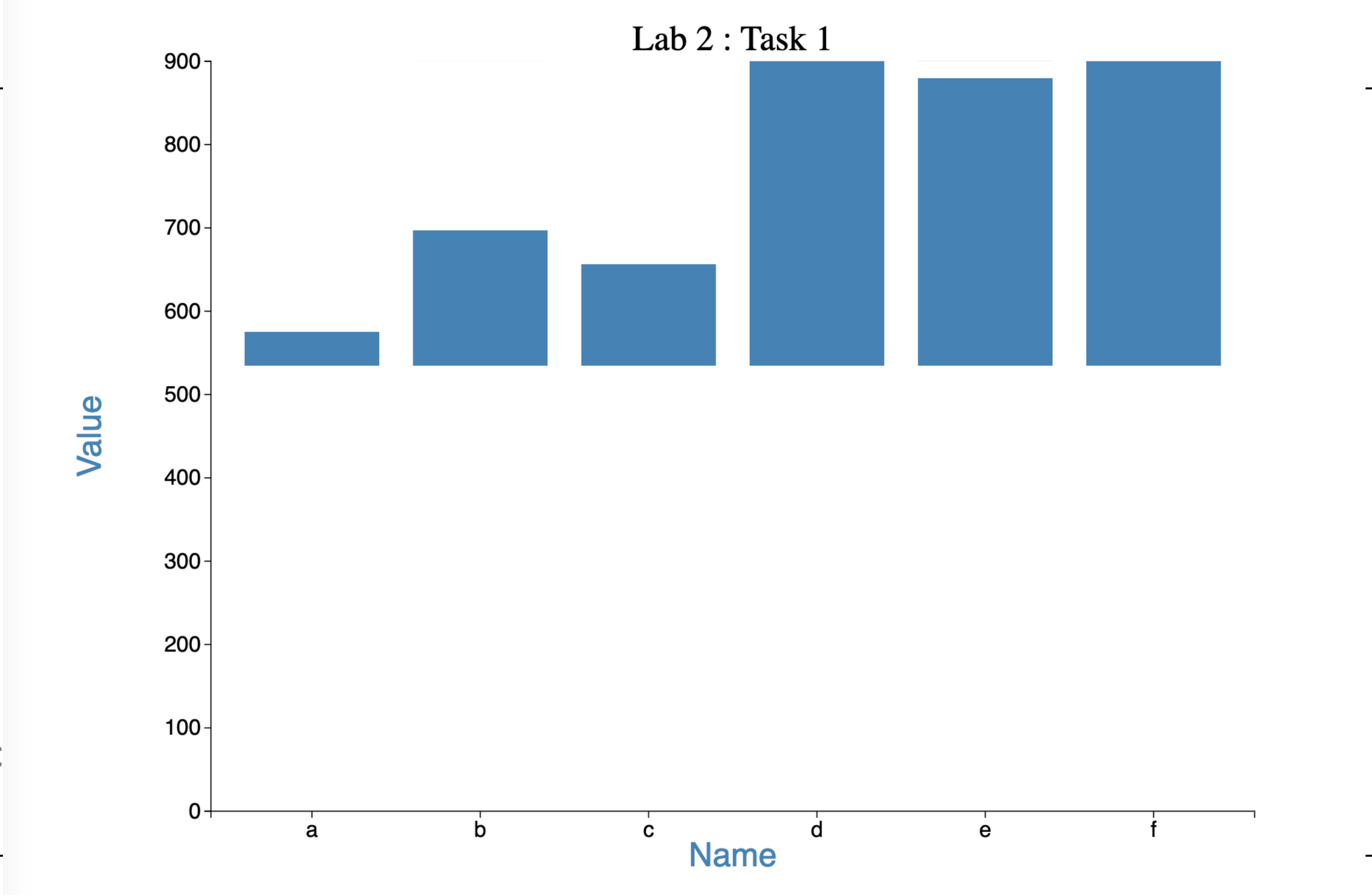
- Example: Lab 2 Task 1

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter()
.append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("width",xScale.bandwidth())
.attr("fill", "steelblue")
.transition().duration(2000)
.attr("y",d => yScale(d.value))
.attr("height",d => innerHeight - yScale(d.value));
```

Animation

- Example: Lab 2 Task 1

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter()
.append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("width",xScale.bandwidth())
.attr("fill", "steelblue")
.transition().duration(2000)
.attr("y",d => yScale(d.value))
.attr("height",d => innerHeight - ySc
```



Animation

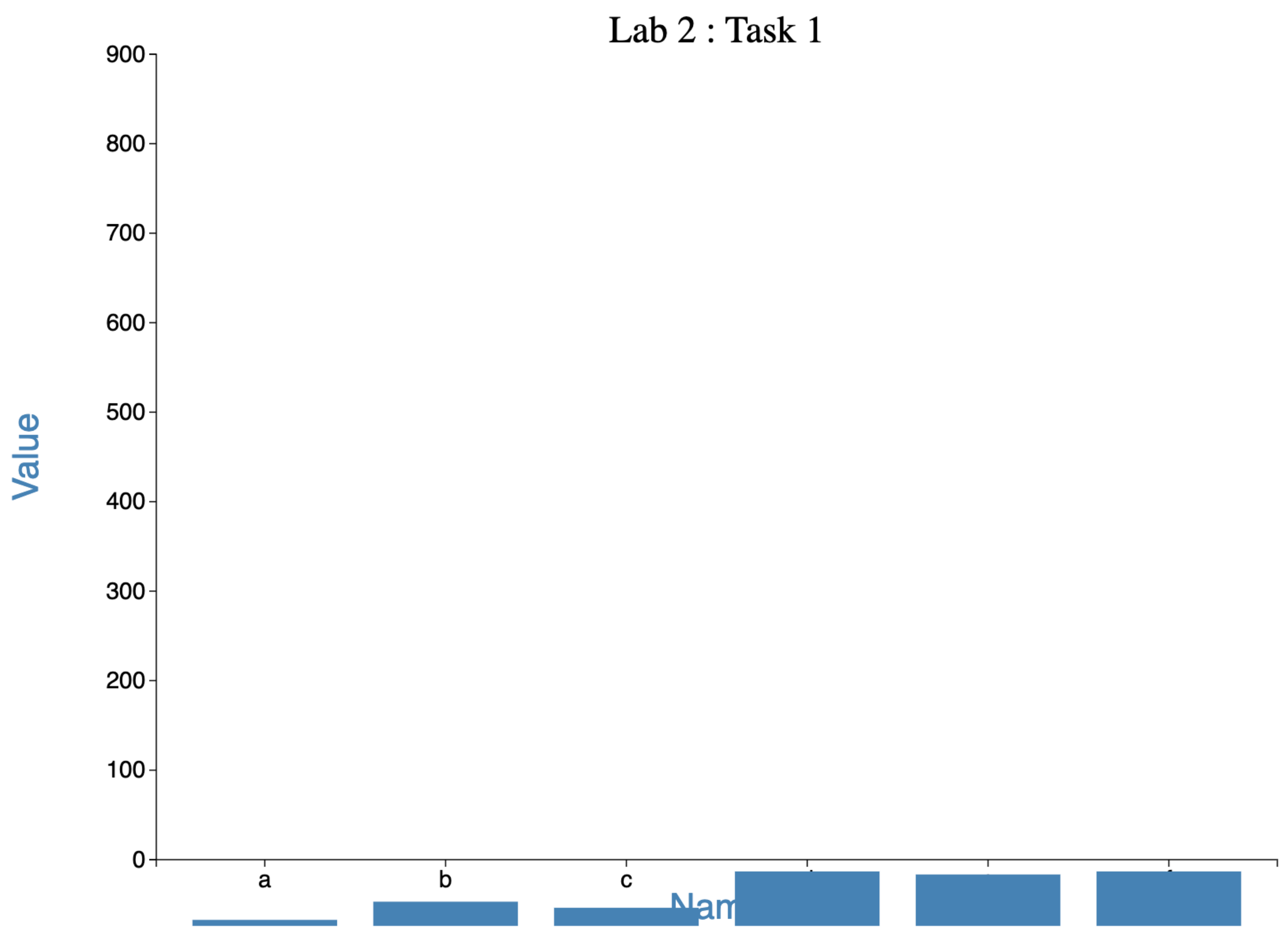
- Example: Lab 2 Task 1

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter()
.append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("width",xScale.bandwidth())
.attr("fill", "steelblue")
.attr("height",0)
.attr("y",height-margin.bottom)
.transition().duration(2000)
.attr("y",d => yScale(d.value))
.attr("height",d => innerHeight - yScale(d.value));
```

Animation

- Example: Lab 2 Task 1

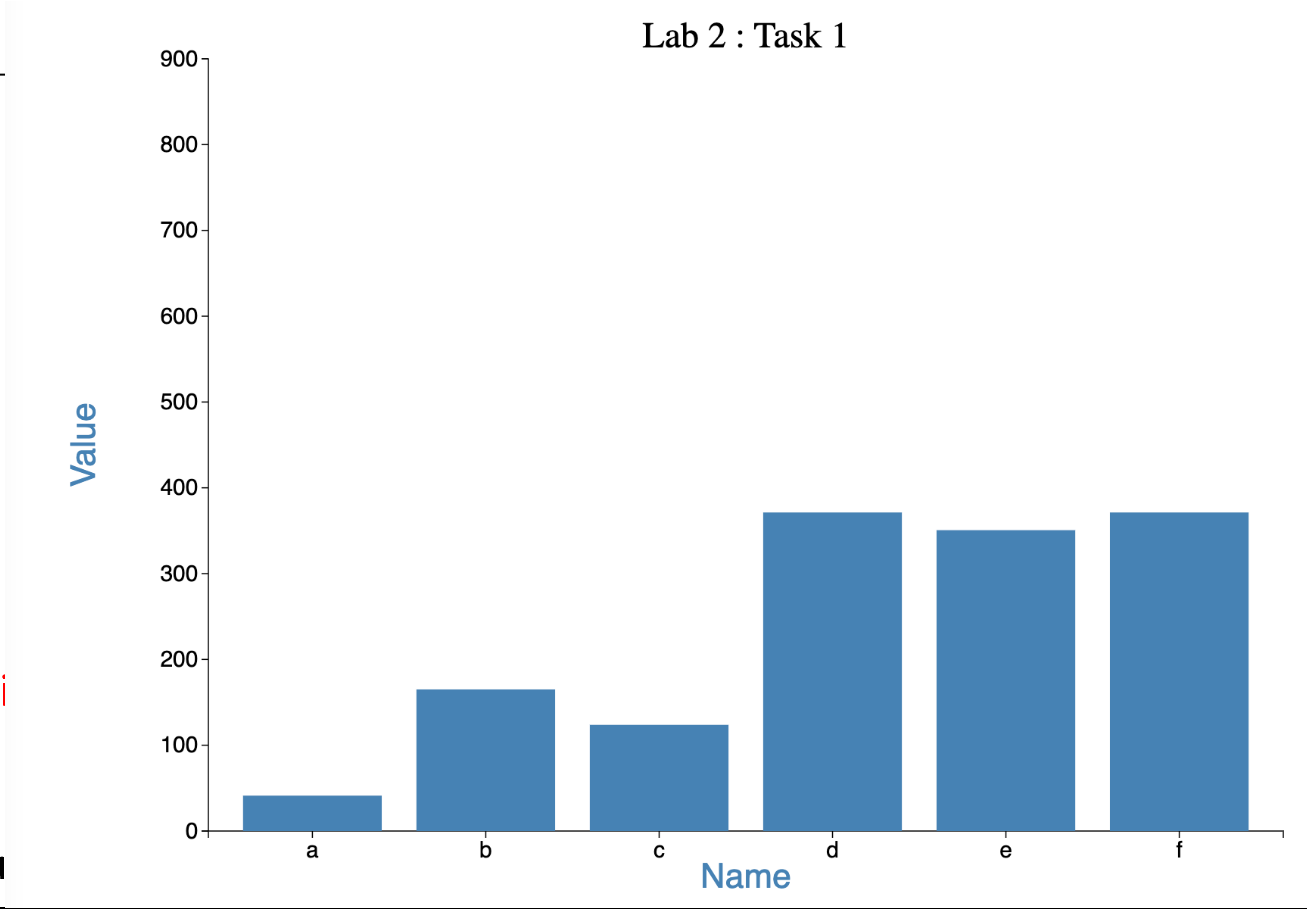
```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter()
.append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("width",xScale.bandwidth())
.attr("fill", "steelblue")
.attr("height",0)
.attr("y",height-margin.bottom)
.transition().duration(2000)
.attr("y",d => yScale(d.value))
.attr("height",d => innerHeight - ySca
```



Animation

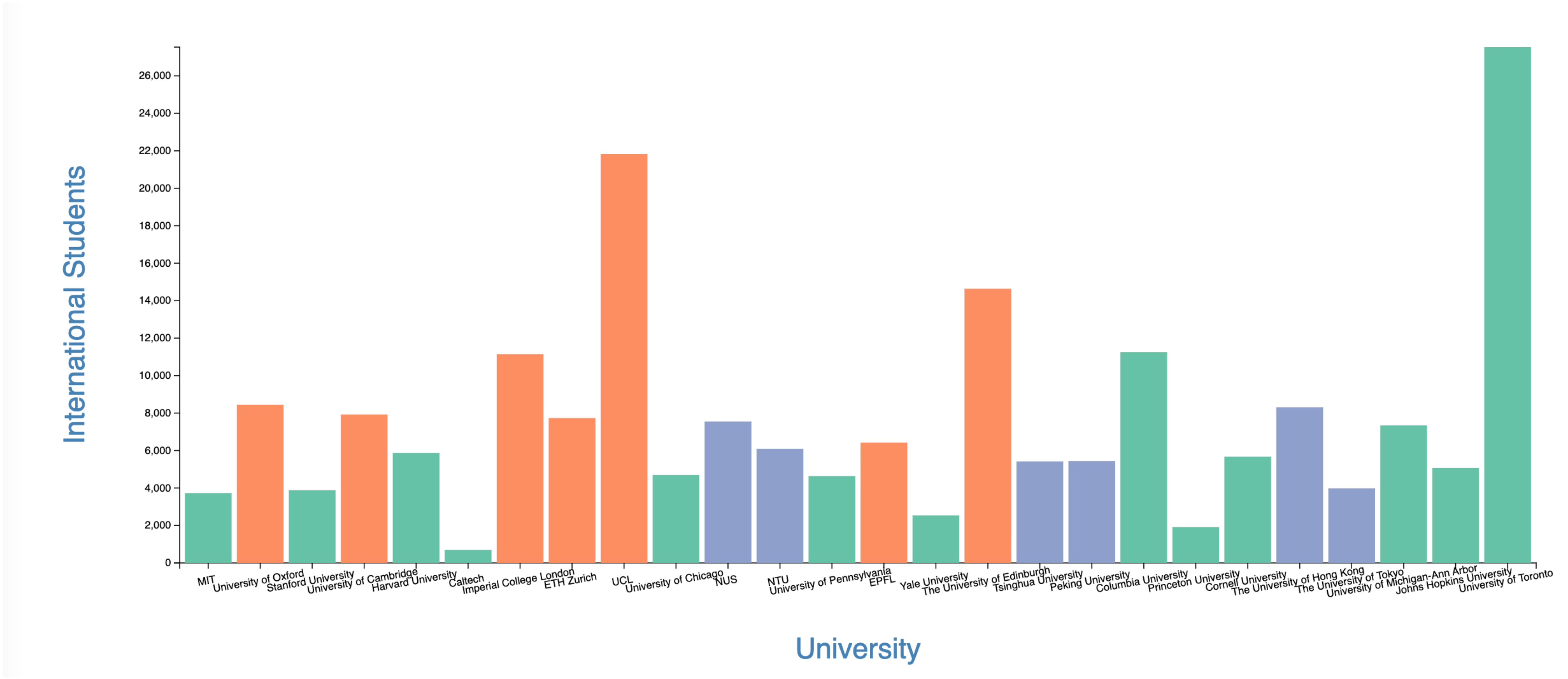
- Example: Lab 2 Task 1

```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter()
.append("rect")
.attr("class","bar")
.attr("x",d => xScale(d.name))
.attr("width",xScale.bandwidth())
.attr("fill", "steelblue")
.attr("height",0)
.attr("y",height-margin.bottom-margin.top)//i
.transition().duration(2000)
.attr("y",d => yScale(d.value))
.attr("height",d => innerHeight - yScale(d.valu
```



Animation & Colour

- Lab 3 Task 1



Interaction

- Event:
  - Events are fired to notify code of "interesting changes" that may affect code execution.
  - click
  - mouseover
  - mouseout
  - keydown
  - contextmenu

Interaction

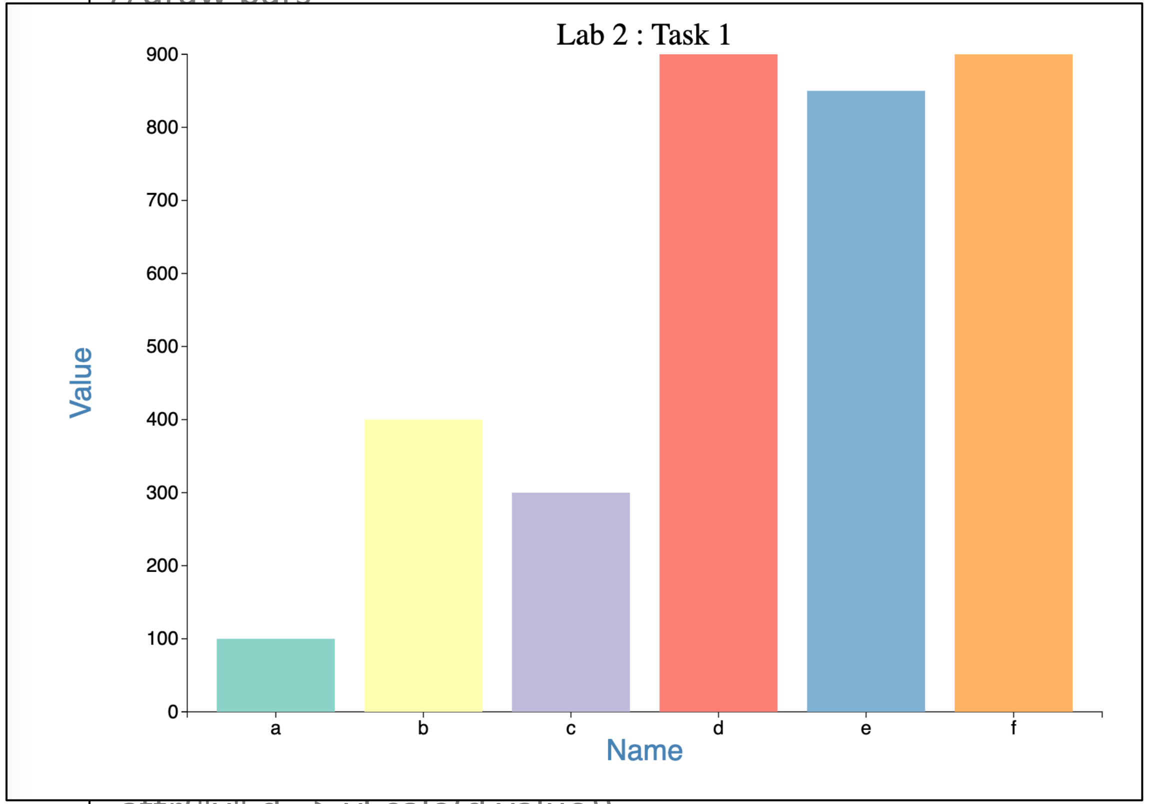
- Event Listener: .on()

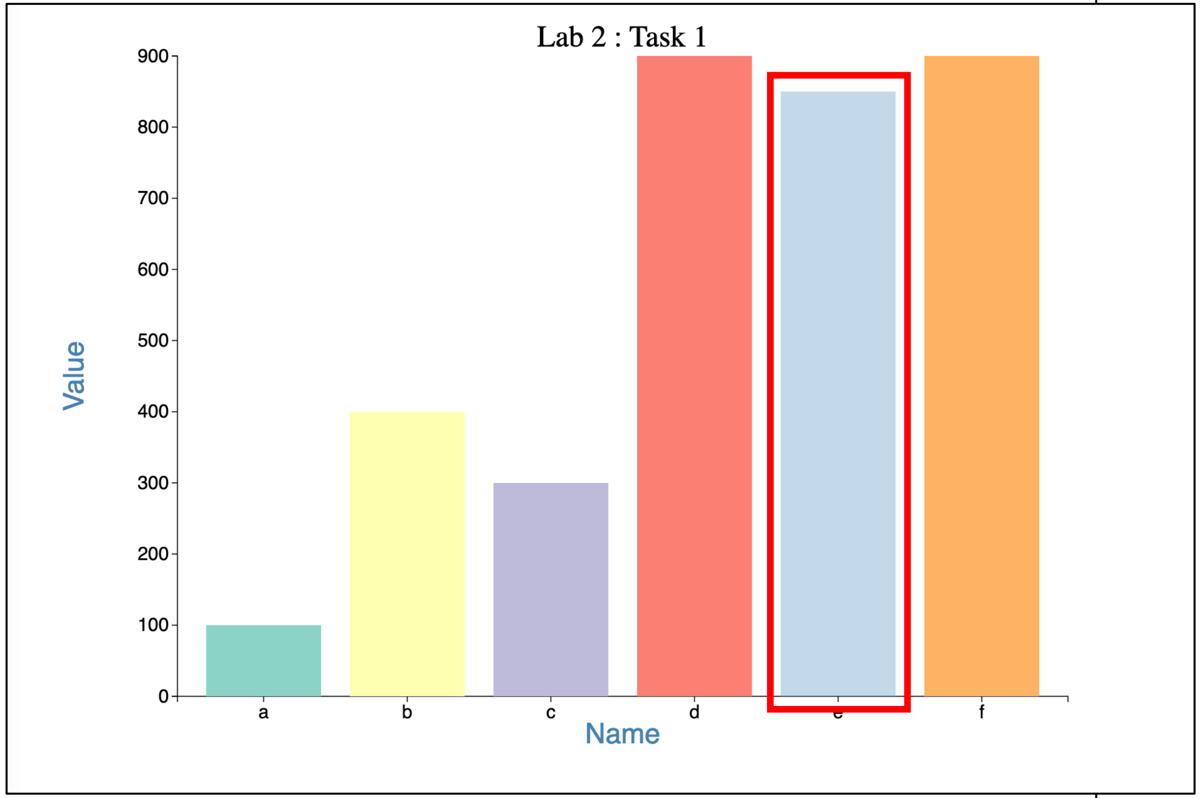
```
//draw bars
g.selectAll(".bar")
.data(data1)
.enter()
.append("rect")
.attr("class","bar")
.on("mouseover", function(d){
                d3.select(this)
                .attr("opacity",0.5);
})
.attr("x",d => xScale(d.name))
.attr("width",xScale.bandwidth())
.attr("fill", d => colourScale(d.name))
.attr("height",0)
.attr("y",height-margin.bottom-margin.top)
.transition().duration(2000)
.attr("y",d => yScale(d.value))
.attr("height",d => innerHeight - yScale(d.value));
```

Interaction

- Event Listener: .on()

```
//draw bars
```



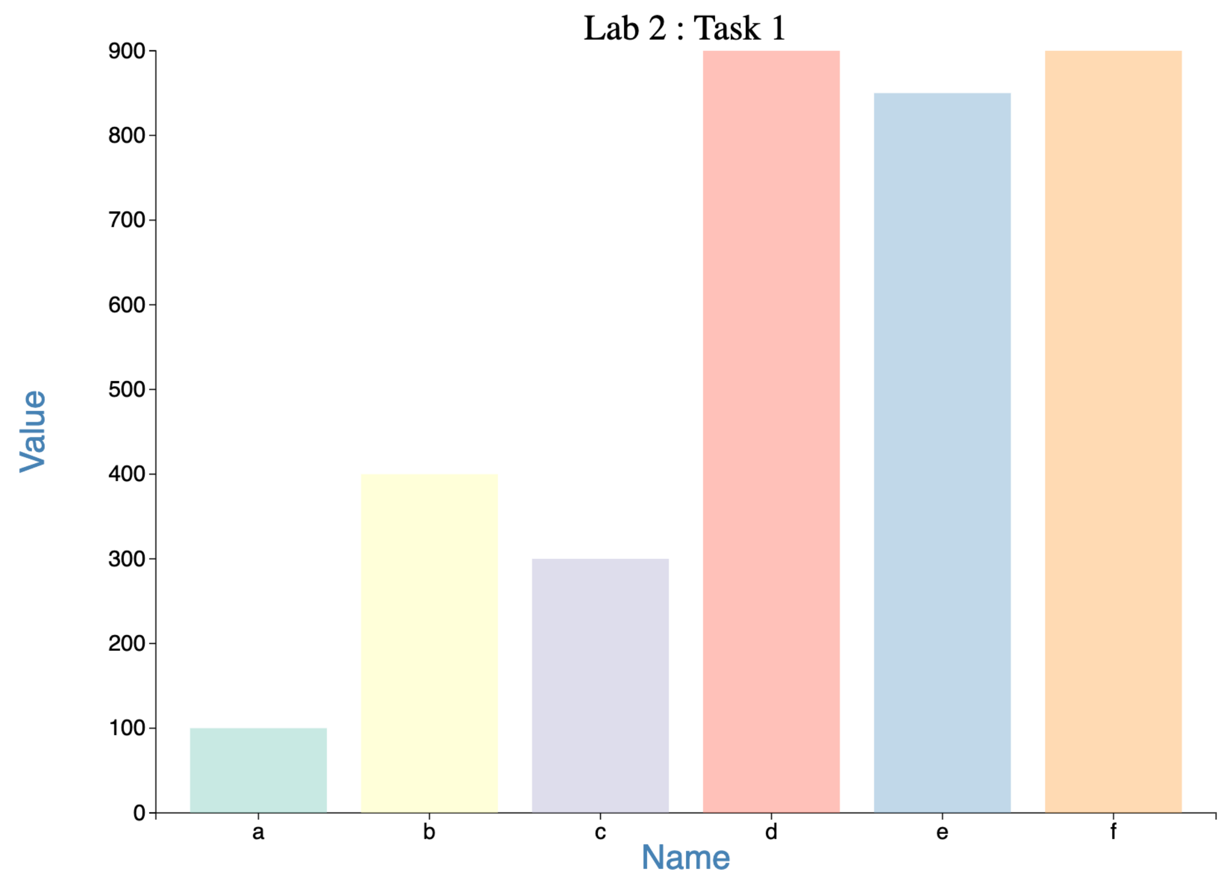


```
.attr("y",d => yScale(d.value))
.attr("height",d => innerHeight - yScale(d.value));
```

Interaction

- Event Listener: .on()

Problem ?



Interaction

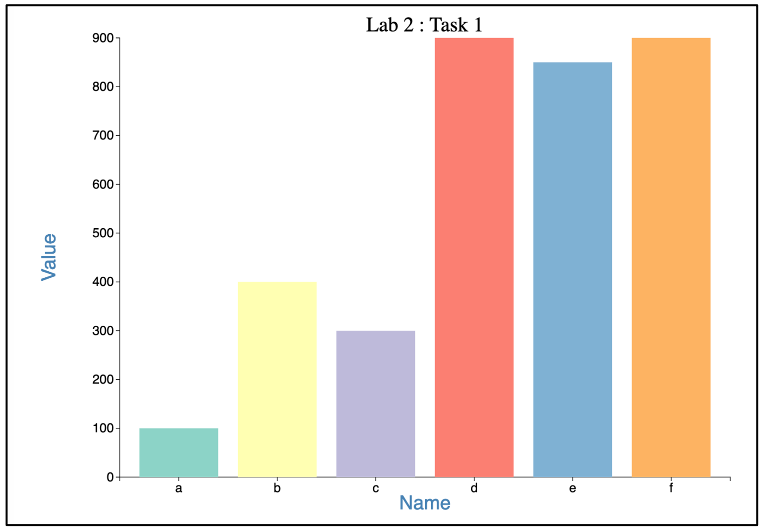
- mouseover & mouseout

```
…
.on("mouseover", function(d){
d3.select(this)
.attr("opacity",0.5);
})
.on("mouseout", function(d){
d3.select(this)
.attr("opacity",1);
})
…
```

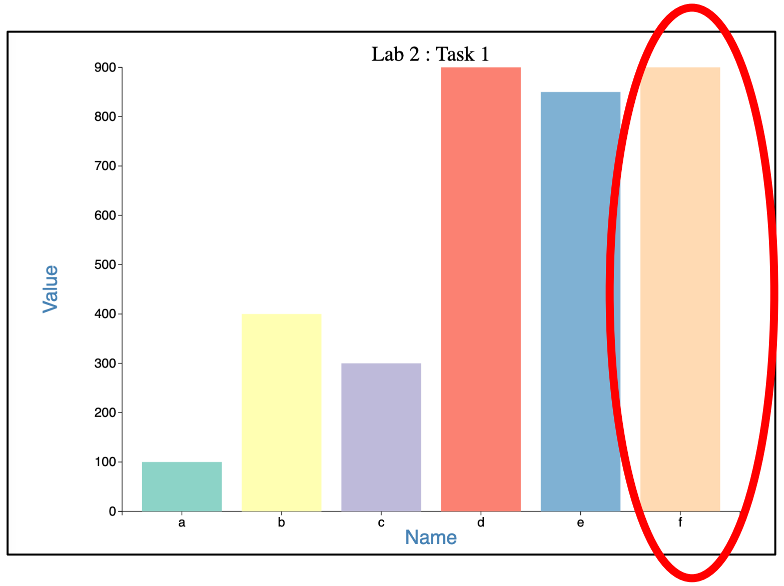
Interaction

- mouseover & mouseout

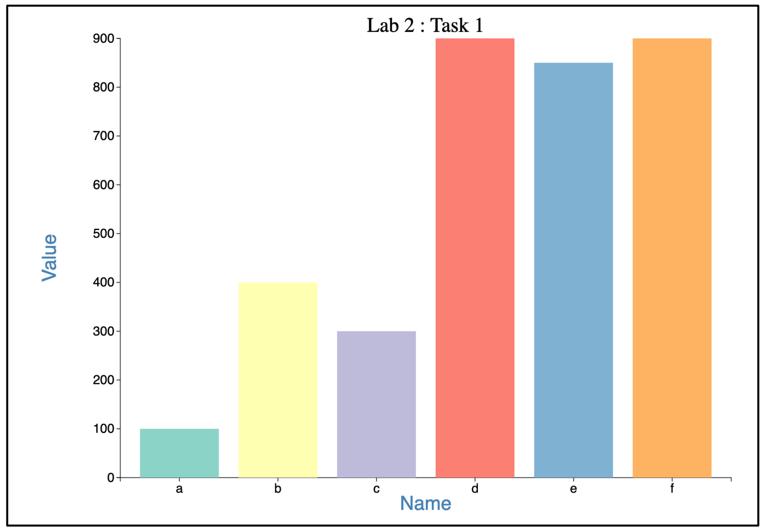
Original Chart



mouseover



mouseout



Interaction

- Tip
  - D3-tip.js
    - `<script src="d3-tip.js"></script>`

Interaction

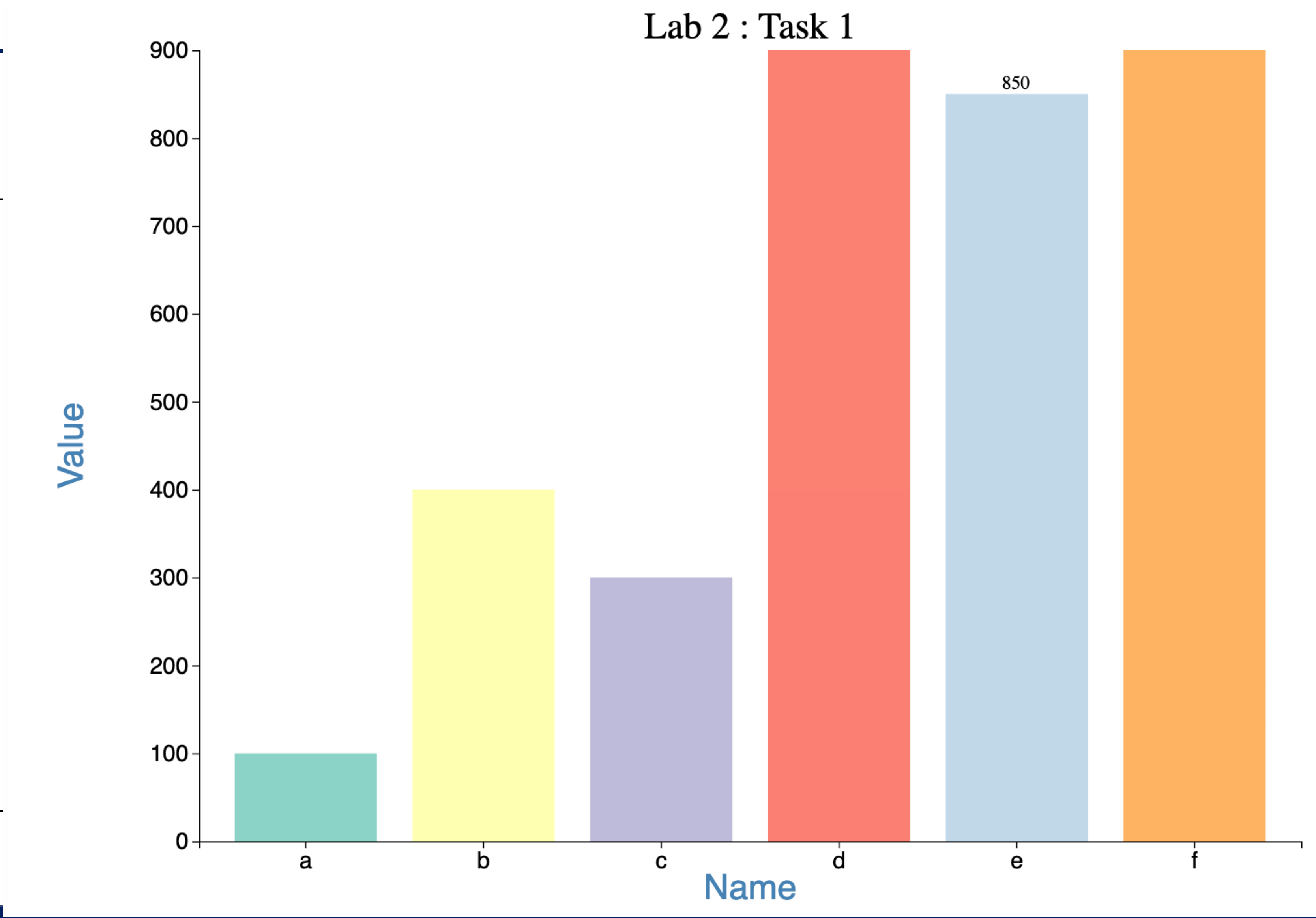
- Tip

```
const tip = d3.tip()
.attr("class", "d3-tip")
.html(function(d) {
                return d.value;
})
svg.call(tip);

…

.on("click",function(d){
tip.show(d);
})
```

Interaction

- Tip

```
const tip = d3.tip()
.attr("class", "d3-tip")
.html(function(d) {
return d.value;
})
svg.call(tip);

…

.on("click",function(d){
tip.show(d);
})
```



Interaction

- Tip

```
.on("click",function(d){
tip.show(d);
})
.on("mouseout", function(d){
d3.select(this)
.attr("opacity",1)
tip.hide(d);
})
```

Summary

- Colour
  - opacity
  - APIs
  - d3.scaleOrdinal()
- Animation
  - transition()
    - duration
    - ease
    - delay
- Interaction
  - event Listener
  - tip