# Transition

## Data Visualization

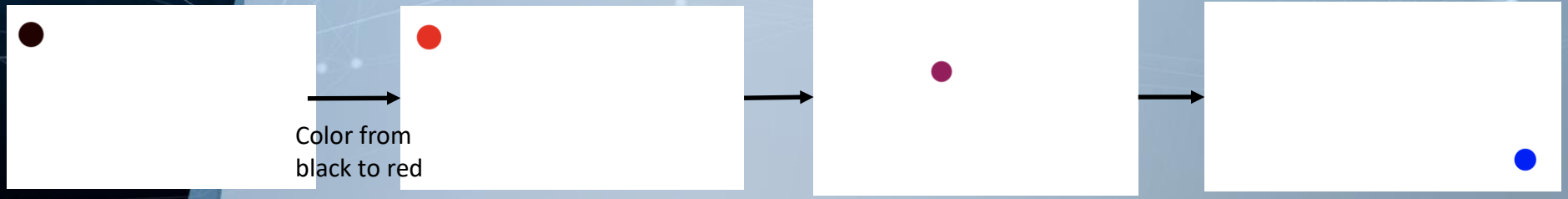
# Transitions

- When the state of an element changes from current state to desired state, transition helps to apply the change smoothly by interpolating the states between the two end states
  - https://www.d3-graph-gallery.com/graph/interactivity_transition.html#mostBasic
- Applications
  - Assist interactions
  - Story-telling a series of visualization connected by interaction and animation
  - A storytelling demonstration: http://vallandingham.me/scroll_demo/

# d3.transition()  (Ex07-01)

```
var circle = d3.select("svg")
                .append('circle')
                .attr('cx', 50)
                .attr('cy', 50)
                .attr('r', 20)
                .attr('fill', 'black');
circle
// Transition 1
.transition()
      .duration(2000)
      .attr("fill", "red")
// Transition 2
.transition()
      .duration(2000)
      .attr("fill", "blue")
      .attr('cx', 600)
      .attr('cy', 300)
      ;
```

Add a black circle at (20,20)

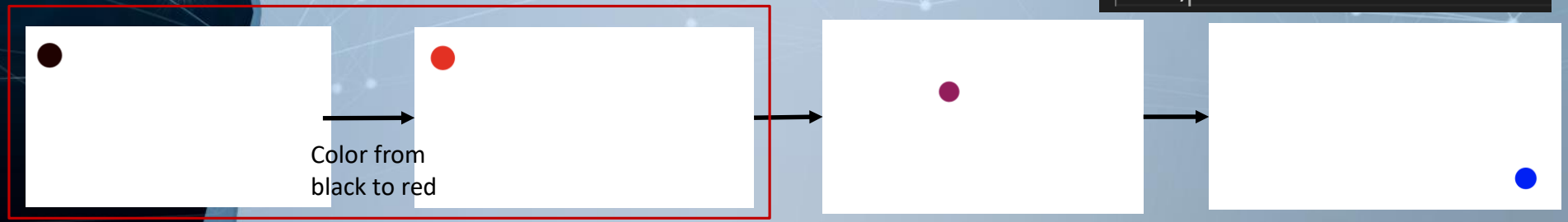



Move and change color to blue smoothly

# d3.transition() (Ex07-01)

- add "**.transition()**" before modify the attributes of the item, D3 can automatically calculate how to smoothly move from one state to another state
  - This example: from black to red color
- **.duration()**: this transition should spend 2000milliseconds (2seconds)

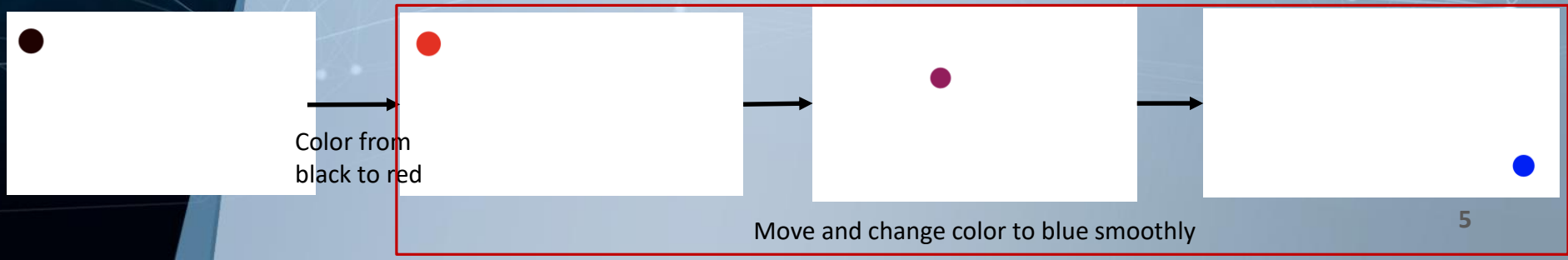
```
var circle = d3.select("svg")
                .append('circle')
                .attr('cx', 50)
                .attr('cy', 50)
                .attr('r', 20)
                .attr('fill', 'black');
circle
// Transition 1
.transition()
      .duration(2000)
      .attr("fill", "red")
// Transition 2
.transition()
      .duration(2000)
      .attr("fill", "blue")
      .attr('cx', 600)
      .attr('cy', 300)
      ;
```



# d3.transition() (Ex07-01)

- Transition chaining
  - We can create multiple transitions
  - When one transition finishes, next transition in the chain takes off

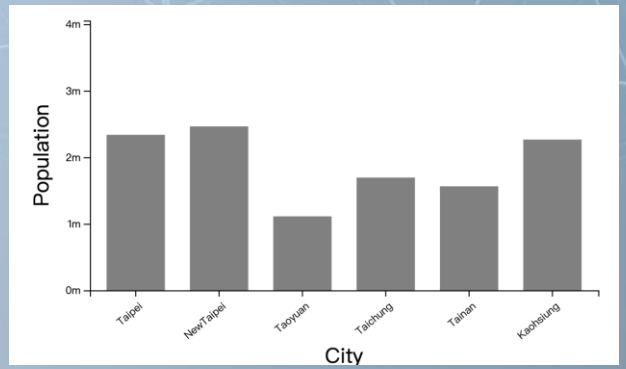
```
var circle = d3.select("svg")
                    .append('circle')
                    .attr('cx', 50)
                    .attr('cy', 50)
                    .attr('r', 20)
                    .attr('fill', 'black');
circle
// Transition 1
.transition()
      .duration(2000)
      .attr("fill", "red")
// Transition 2
.transition()
      .duration(2000)
      .attr("fill", "blue")
      .attr('cx', 600)
      .attr('cy', 300)
      ;
```



# Ex07-02

- Animate bar chart
  - Modify from Ex04-16
- cityPopulation.csv
  - Population of 6 Taiwan sspecial municipalities in 1956, 1966, 1980, 1990, 2000, 2010
- Update the bar length by the population in 1956, 1966, 1980, 1990, 2000, 2010

```
"City","1956","1966","1980","1990","2000","2010"
"Taipei",737029,1202952,2267584,2760475,2624257,2655515
"NewTaipei",668093,1143288,2364521,3065779,3722082,4054467
"Taoyuan",411575,660293,1067951,1377934,1808833,2190342
"Taichung",784475,1113291,1634820,2057857,2499527,2731056
"Tainan",992411,1333735,1541402,1695110,1846379,1840257
"Kaohsiung",899828,1445669,2225021,2512858,2756775,2777384
```



# Ex07-02

- main.js

Which year to display

All years

```
var count = 0;
var years = ['1956', '1966', '1980', '1990', '2000', '2010'];
d3.csv("cityPopulation.csv").then(drawPopulation);

function drawPopulation(cities) {
      const MARGIN = { LEFT: 100, RIGHT: 10, TOP: 10, BOTTOM: 130 }
      const WIDTH = 600 - MARGIN.LEFT - MARGIN.RIGHT
      const HEIGHT = 400 - MARGIN.TOP - MARGIN.BOTTOM

      const svg = d3.select("#chart-area").append("svg")
                  .attr("width", WIDTH + MARGIN.LEFT + MARGIN.RIGHT)
                  .attr("height", HEIGHT + MARGIN.TOP + MARGIN.BOTTOM)

      const g = svg.append("g")
                  .attr("transform", `translate(${MARGIN.LEFT}, ${MARGIN.TOP})`)
```

Load data. After loading is done, run "drawPopulation()"

Data from "cityPopulation.csv"

Rest of the code: very similar to Ex04-16 (draw the bar chart)

```
// X label
g.append("text")
      .attr("x", WIDTH / 2)
      .attr("y", HEIGHT + 70)
      .attr("font-size", "20px")
      .attr("text-anchor", "middle")
      .text("City")

// Y label
g.append("text")
      .attr("x", - (HEIGHT / 2))
      .attr("y", -40)
      .attr("font-size", "20px")
      .attr("text-anchor", "middle")
      .attr("transform", "rotate(-90)")
      .text("Population")

// X ticks
const x = d3.scaleBand()
            .domain(cities.map(d => d['City']))
            .range([0, WIDTH])
            .paddingInner(0.3)
            .paddingOuter(0.2)

const xAxisCall = d3.axisBottom(x)
g.append("g")
      .attr("transform", `translate(0, ${HEIGHT})`)
      .call(xAxisCall)
      .selectAll("text")
      .attr("y", "10")
      .attr("x", "-5")
      .attr("text-anchor", "end")
      .attr("transform", "rotate(-40)")

// Y ticks
const y = d3.scaleLinear()
      .domain([0, d3.max(cities, d => d['2010'])])
      .range([HEIGHT, 0])

const yAxisCall = d3.axisLeft(y)
                  .ticks(3)
                  .tickFormat(d => (d/1000000) + "m")
var yaxis = g.append("g").call(yAxisCall)
```

Code to setup and display y axis.

We will animate y-axis in Ex07-03

# Ex07-02

- main.js

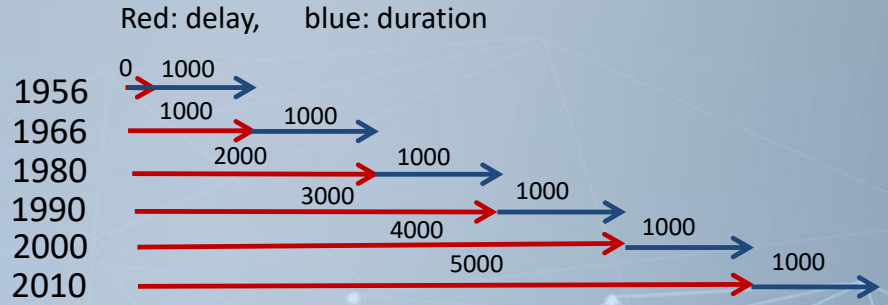
Count is 0 in the beginning, so years[count] here is '1956' (display data in 1956)

Draw bars

```
var rects = g.selectAll("rect").data(cities);

var r = rects.enter().append("rect")
            .attr("y", d => y(d[years[count]]))
            .attr("x", (d) => x(d['City']))
            .attr("width", x.bandwidth)
            .attr("height", d => HEIGHT - y(d[years[count]]))
            .attr("fill", "grey");

years.forEach(function(year, i){
      var t= d3.transition()
                  .delay(1000*i)
                  .duration(1000);

      r.transition(t)
            .attr("y", d => y(d[year]))
            .attr("height", d => HEIGHT - y(d[year]));
})
```

# Ex07-02

- main.js




Iterate through (year, i) =
(1956, 0),
(1966, 1),
(1980, 2),
(1990, 3),
(2000, 4),
(2010, 5),

This forEach loop finishes and setup all transition from 1956-2010 immediately

```
var r = rects.enter().append("rect")
        .attr("y", d => y(d[years[count]]))
        .attr("x", (d) => x(d['City']))
        .attr("width", x.bandwidth)
        .attr("height", d => HEIGHT - y(d[years[count]]))
        .attr("fill", "grey");

years.forEach(function(year, i){
    var t= d3.transition()
            .delay(1000*i)
            .duration(1000);

    r.transition(t)
        .attr("y", d => y(d[year]))
        .attr("height", d => HEIGHT - y(d[year]));
})
}
```

.delay(): how long to wait (milliseconds) before start the transition animation

Use the transition setting 't' to animate the change

# Ex07-03

- Animate the y-axis

This is usually how we add y axis

```
// Y ticks
const y = d3.scaleLinear()
      .domain([0, d3.max(cities, d => d['1956'])])
      .range([HEIGHT, 0])

const yAxisCall = d3.axisLeft(y)
                    .ticks(3)
                    .tickFormat(d => (d/1000000) + "m")
var yaxis = g.append("g").call(yAxisCall)
```

Create new yaxis using new y scale function

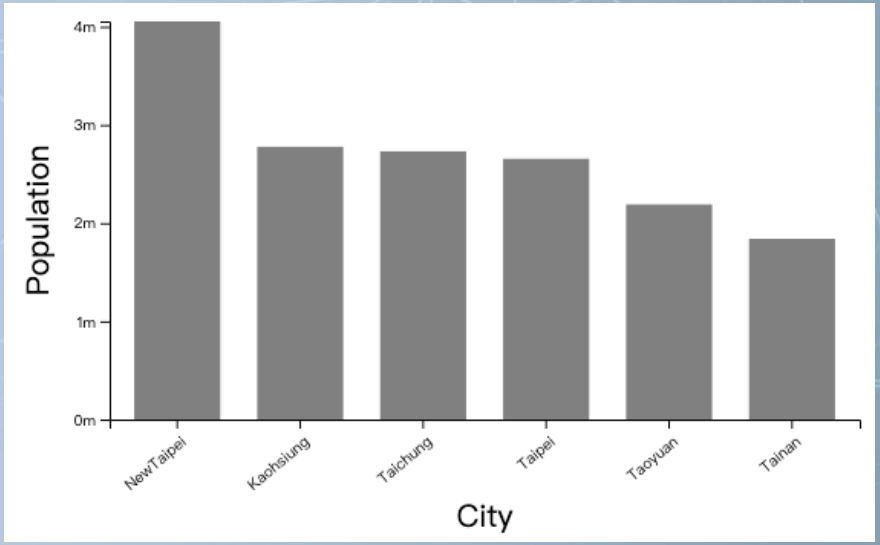
Before .call(newAxisCall), call transition(t) to update the y axis smoothly

Create a new y scale function by the maximal value of that year

```
var r = rects.enter().append("rect")
            .attr("y", d => y(d[years[count]]))
            .attr("x", (d) => x(d['City']))
            .attr("width", x.bandwidth)
            .attr("height", d => HEIGHT - y(d[years[count]]))
            .attr("fill", "grey");

years.forEach(function(year, i){
      var t= d3.transition()
                  .delay(1000*i)
                  .duration(1000);

      var newy = d3.scaleLinear()
            .domain([0, d3.max(cities, d => parseInt(d[year]))])
            .range([HEIGHT, 0]);
      var newyAxisCall = d3.axisLeft(newy)
                           .ticks(3)
                           .tickFormat(d => (d/1000000) + "m")
      yaxis.transition(t).call(newyAxisCall);

      r.transition(t)
            .attr("y", d => newy(d[year]))
            .attr("height", d => HEIGHT - newy(d[year]));
})
```

# Ex07-04

- Animate x-axis and always sort cities by the population



# Ex07-04

When we create the first frame, we sort the data by ‘1956’

and retrieve the city name

to create x-axis scale function (scaleBand)

```
// X label
g.append("text")
      .attr("x", WIDTH / 2)
      .attr("y", HEIGHT + 70)
      .attr("font-size", "20px")
      .attr("text-anchor", "middle")
      .text("City")

// Y label
g.append("text")
      .attr("x", - (HEIGHT / 2))
      .attr("y", -40)
      .attr("font-size", "20px")
      .attr("text-anchor", "middle")
      .attr("transform", "rotate(-90)")
      .text("Population")

cities = cities.sort((a,b)=>b['1956']-a['1956']);
var cityName = cities.map(d => d['City']);
// X ticks
const x = d3.scaleBand()
            .domain(cityName)
            .range([0, WIDTH])
            .paddingInner(0.3)
            .paddingOuter(0.2)

const xAxisCall = d3.axisBottom(x)
var xaxis = g.append("g")
            .attr("transform", `translate(0, ${HEIGHT})`)
            .call(xAxisCall);
xaxis.selectAll("text")
      .attr("y", "10")
      .attr("x", "-5")
      .attr("text-anchor", "end")
      .attr("transform", "rotate(-40)");
```

# Ex07-04

- do the similar thing when we switch to next year
  - Sort data by the year
  - Retrieve city name array
  - Create new x-axis scale function
  - Draw new x-axis by the new x-axis scale function

Update the x position of bars by new x-axis scale function

```
var rects = g.selectAll("rect").data(cities);

var r = rects.enter().append("rect")
            .attr("y", d => y(d[years[count]]))
            .attr("x", (d) => x(d['City']))
            .attr("width", x.bandwidth)
            .attr("height", d => HEIGHT - y(d[years[count]]))
            .attr("fill", "grey");

years.forEach(function(year, i){
      var t= d3.transition()
                  .delay(1000*i)
                  .duration(1000);

      var newy = d3.scaleLinear()
            .domain([0, d3.max(cities, d => parseInt(d[year]))])
            .range([HEIGHT, 0]);
      var newyAxisCall = d3.axisLeft(newy)
                        .ticks(3)
                        .tickFormat(d => (d/1000000) + "m")
      yaxis.transition(t).call(newyAxisCall);

      cities = cities.sort((a,b)=>b[year]-a[year]);
      var cityName = cities.map(d => d['City']);
      var newx = d3.scaleBand()
                  .domain(cityName)
                  .range([0, WIDTH])
                  .paddingInner(0.3)
                  .paddingOuter(0.2);
      var newxAxisCall = d3.axisBottom(newx);
      xaxis.transition(t).call(newxAxisCall);

      r.transition(t)
            .attr("x", (d) => newx(d['City']))
            .attr("y", d => newy(d[year]))
            .attr("height", d => HEIGHT - newy(d[year]));
})
```