hci+d lab.

Lab 7: Pie Chart & Line Graph

정보 비주얼라이제이션 2015 Fall

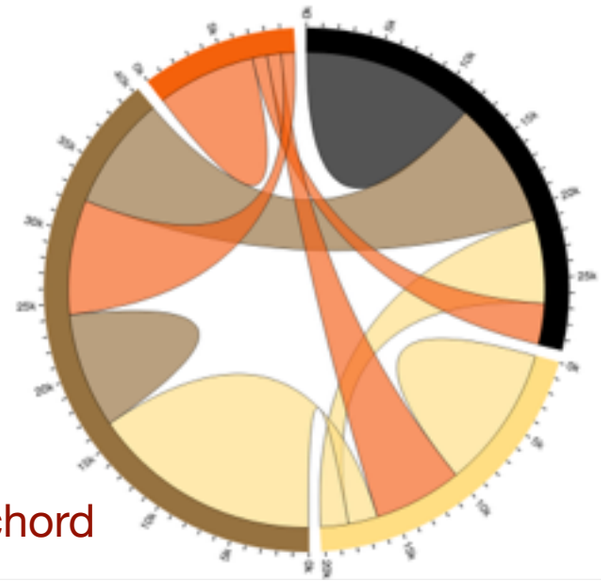
human-computer interaction + design lab.
Joonhwan Lee

Pie Chart

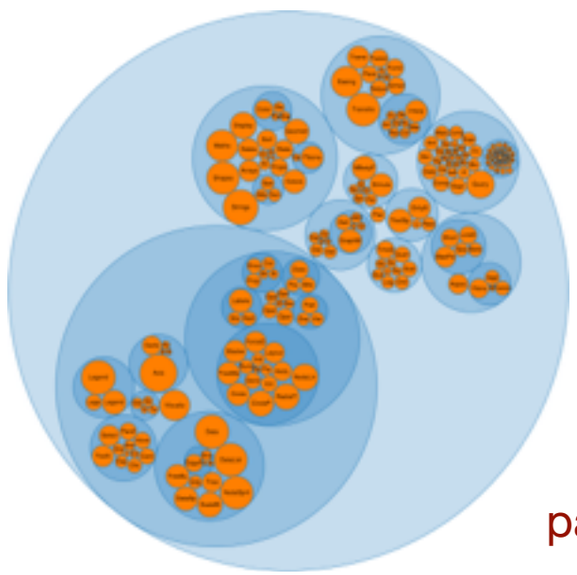
d3.js의 레이아웃 기능

- d3.js는 그리기에 필요한 레이아웃 제공

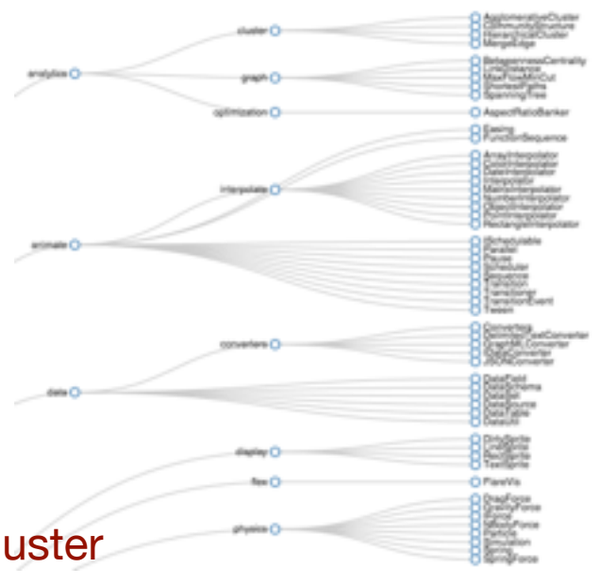
bundle



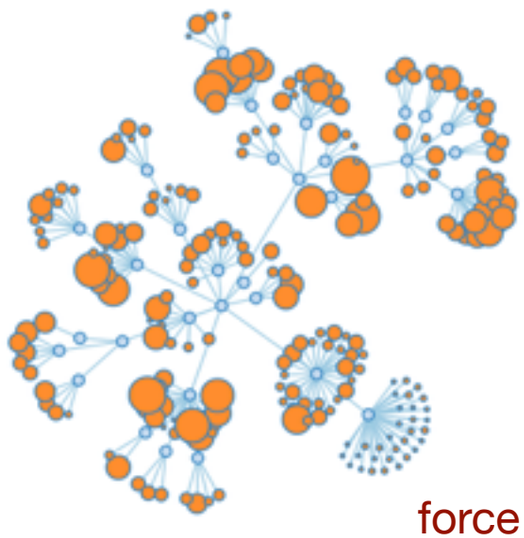
chord



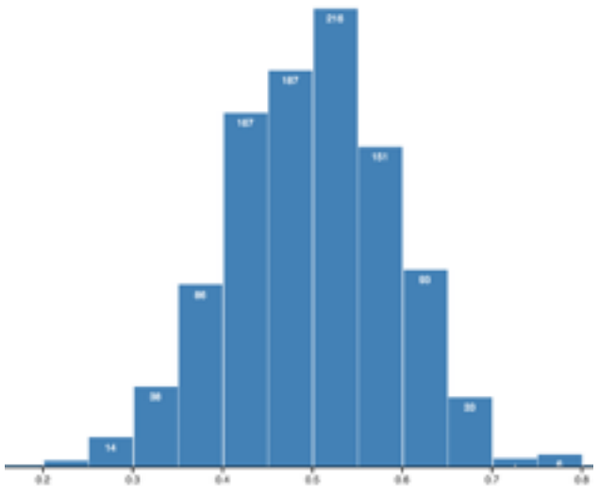
pack



cluster



force



histogram

d3.js의 레이아웃 기능

- Layout의 종류
  - Bundle - apply Holten's hierarchical bundling algorithm to edges.
  - Chord - produce a chord diagram from a matrix of relationships.
  - Cluster - cluster entities into a dendrogram.
  - Force - position linked nodes using physical simulation.
  - Hierarchy - derive a custom hierarchical layout implementation.
  - Histogram - compute the distribution of data using quantized bins.

d3.js의 레이아웃 기능

- Layout의 종류
  - Pack - produce a hierarchical layout using recursive circle-packing.
  - Partition - recursively partition a node tree into a sunburst or icicle.
  - Pie - compute the start and end angles for arcs in a pie or donut chart.
  - Stack - compute the baseline for each series in a stacked bar or area chart.
  - Tree - position a tree of nodes tidily.
  - Treemap - use recursive spatial subdivision to display a tree of nodes.

Lab 1: Simple Pie Graph

- Step1: Pie 레이아웃 설정

```
var pie = d3.layout.pie()
```

- Step2: 원의 크기 설정

```
var arc =
d3.svg.arc().innerRadius(0).outerRadius(100)
```

Lab 1: Simple Pie Graph

- Step3: Pie chart 그리기 준비

```
var pieElements = d3.select("#myGraph")
 .selectAll("path")
 .data(pie(dataSet))
 .enter()
```

- Step4: 원의 크기 설정

```
pieElements
 .append("path")
 .attr("class", "pie")
 .attr("d", arc) // 데이터 크기대로 arc를 그림
 .attr("transform", "translate(" + svgWidth/2
            + ", " + svgHeight/2 + ")")
```

Lab 2: Color의 지정

- Step1: 색을 지정하려면,
  - Option1: 색의 array를 만들어 style 메소드에 적용

```
.style("fill", function(d, i) {
        return ["red", "orange",
            "yellow", "cyan", "#3f3"][i]
})
```

  - Option2: d3에 미리 준비된 색을 사용

```
var color = d3.scale.category10()
...
.style("fill", function(d, i) {
      return color(i)
})
```

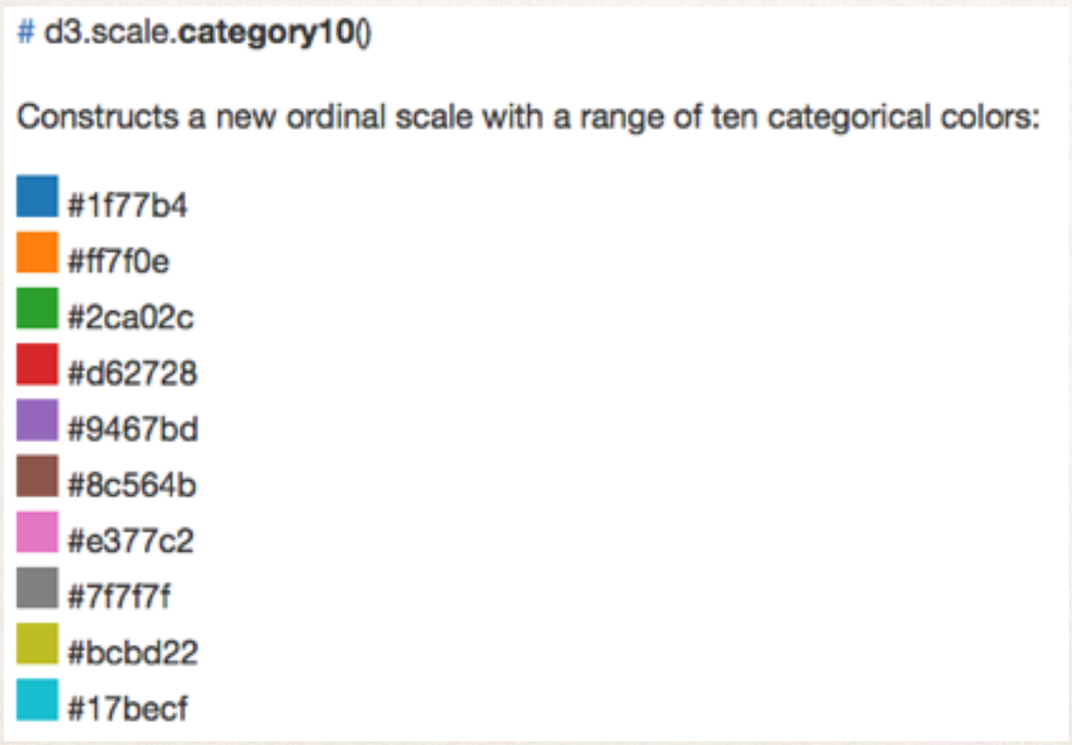
Lab 2: Color의 지정

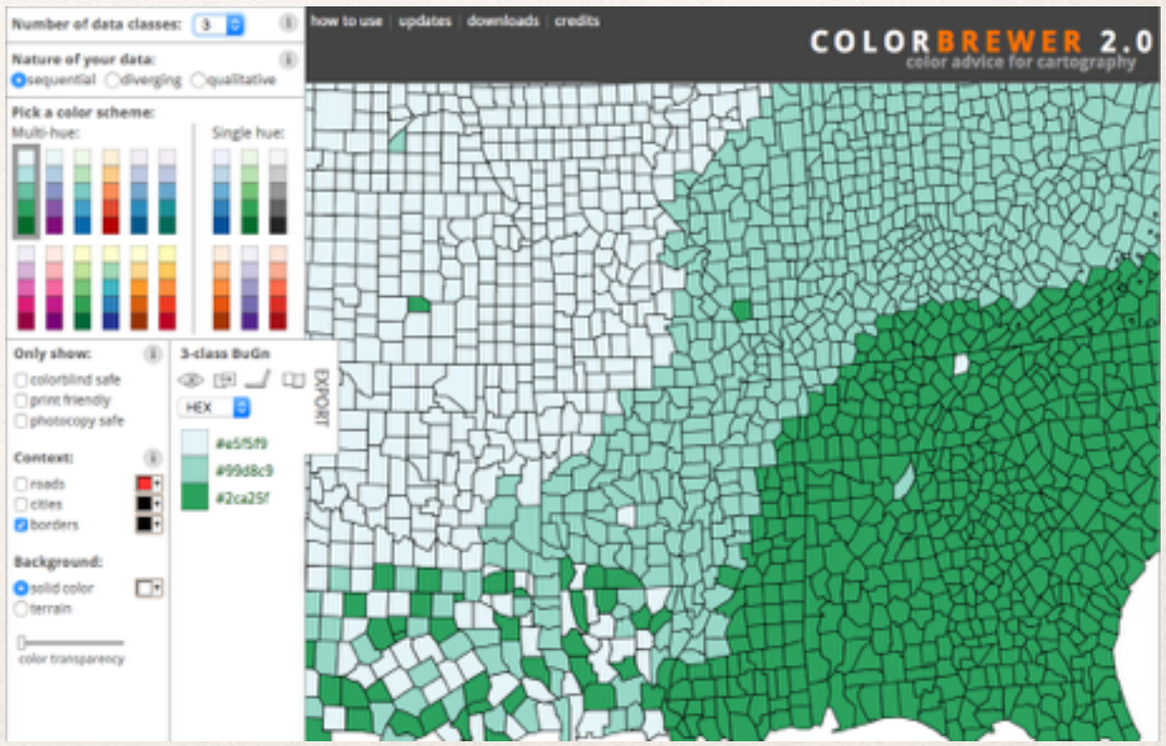
- Step2: CSS 추가
  - pie의 기본 속성 지정

```
.pie {
  stroke: white ;
  stroke-width: 1px;
}
```

Lab 2: Color의 지정

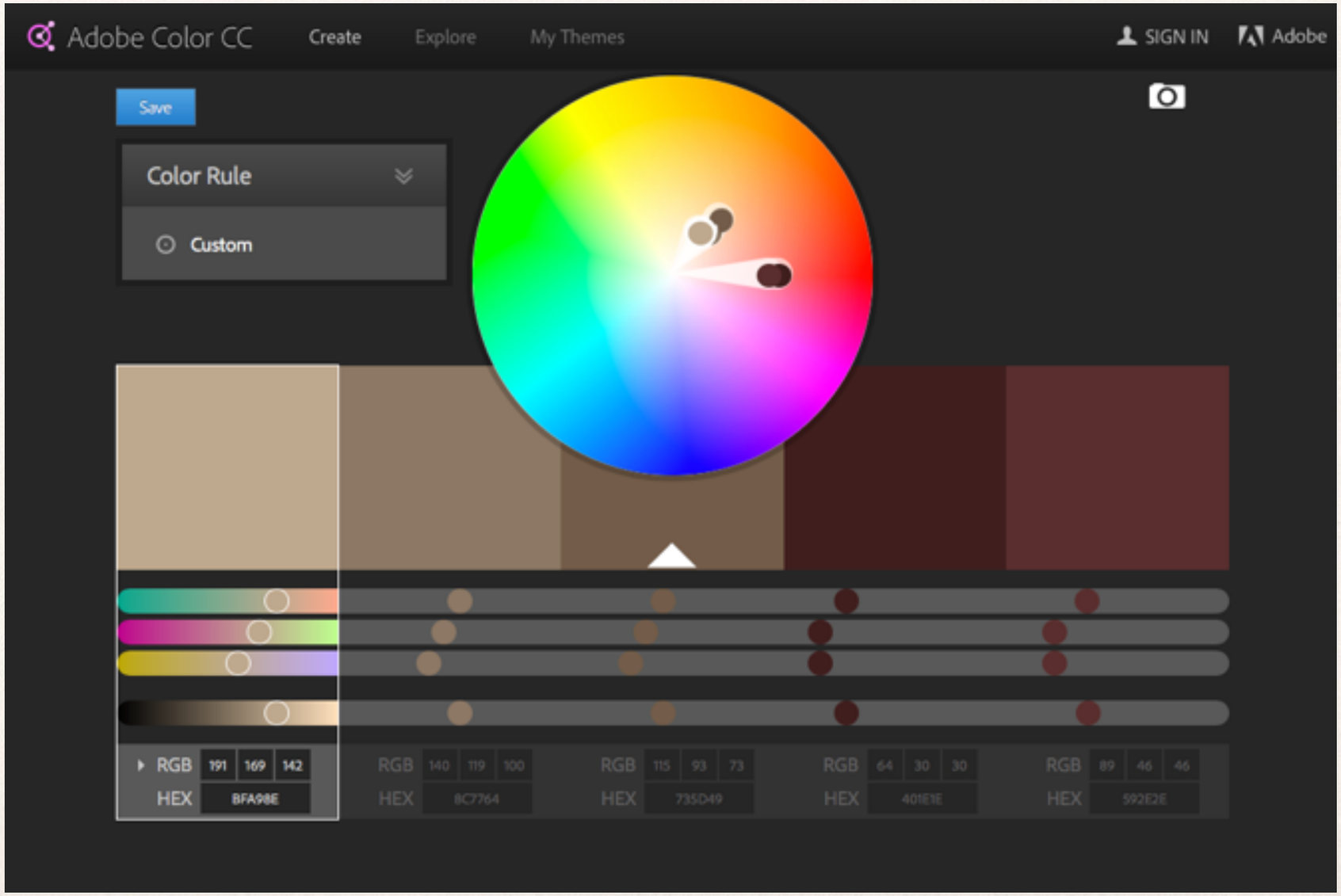
- Categorical Colors
  - https://github.com/mbostock/d3/wiki/Ordinal-Scales#categorical-colors
- ColorBrewer
  - https://github.com/mbostock/d3/wiki/Ordinal-Scales#colorbrewer





Lab 2: Color의 지정

- Adobe Kuler
  - https://color.adobe.com/



Lab 3: Animation

- 기본적으로 animation 은 bar를 그릴 때와 동일하게 `transition()`, `duration()`과 `delay()`를 사용
- 하지만, 단순히 bar의 width나 height를 증가시키는 것과는 달리 pie chart에서는 부채꼴의 좌표를 계산해서 조금씩 커지도록 그려줘야 함
  - `.attrTween()` 메서드를 사용

Lab 3: Animation

- Step 1: 기존의 코드에서 데이터가 그려지는 부분 삭제

```
// .attr("d", arc)
```

- Step 2: 에니메이션 코드 추가

```
.transition()
.duration(1000)
.delay(function(d, i) {
    return i * 1000
})
```

Lab 3: Animation

- Step 3: .attrTween 을 이용하여 arc 그리기

```
.attrTween("d", function(d, i) {
  var interpolate = d3.interpolate(
      { startAngle: d.startAngle,
        endAngle: d.startAngle },
      { startAngle: d.startAngle,
        endAngle: d.endAngle }
  )
  return function(t) {
      return arc(interpolate(t) )}
})
```

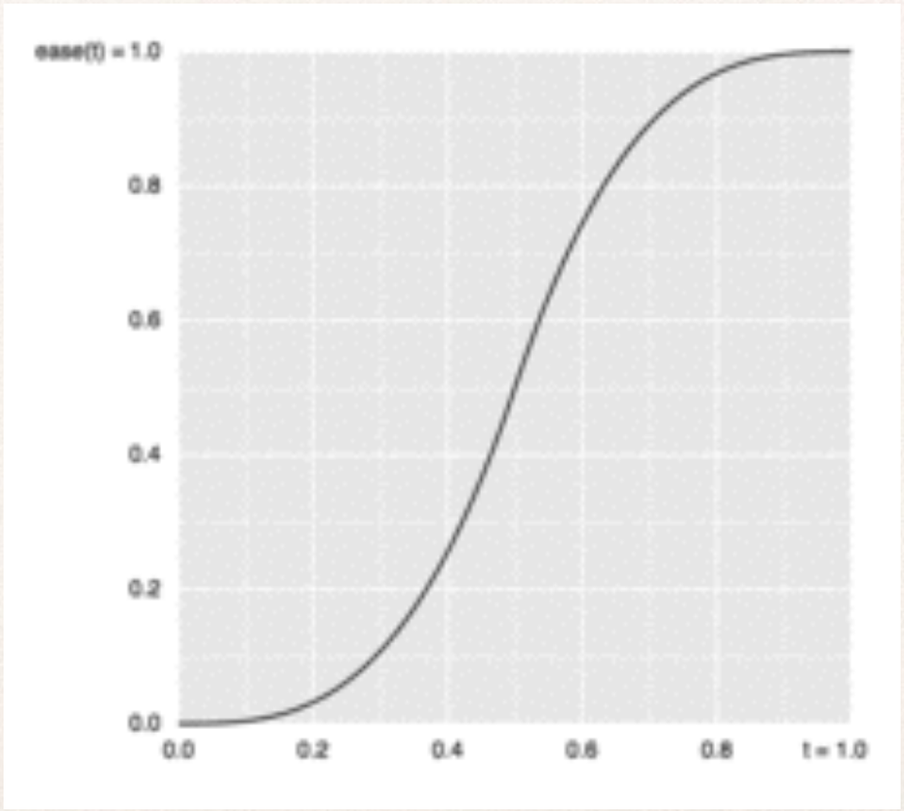
Lab 3: Animation

- Step 4: 데이터 순서대로 정렬해서 그리기
  - 현재는 큰 값 순서대로 정렬되어 있음

```
var pie = d3.layout.pie().sort(null)
```

Lab 3-1: Ease function in Animation

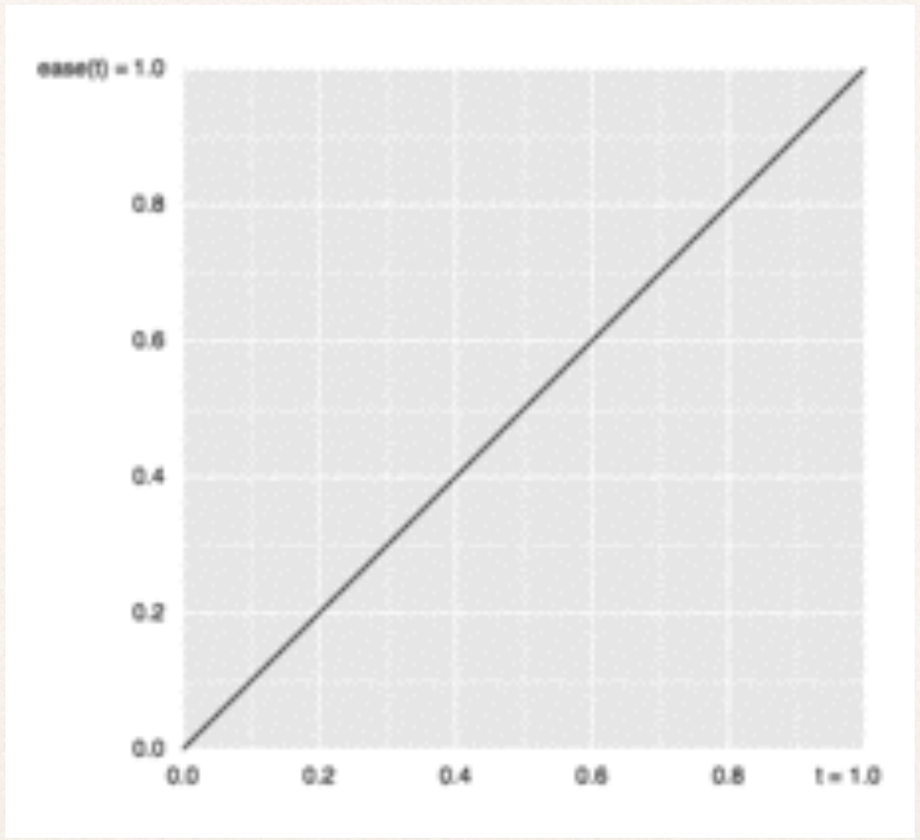
- 현재 animation 은 arc를 하나 그리고 잠깐 멈추는 듯한 느낌. 부드럽게 5개를 연속해서 그리지 않는다.
  - easing-in & easing-out 때문
  - d3 의 default easing function: cubic-in-out



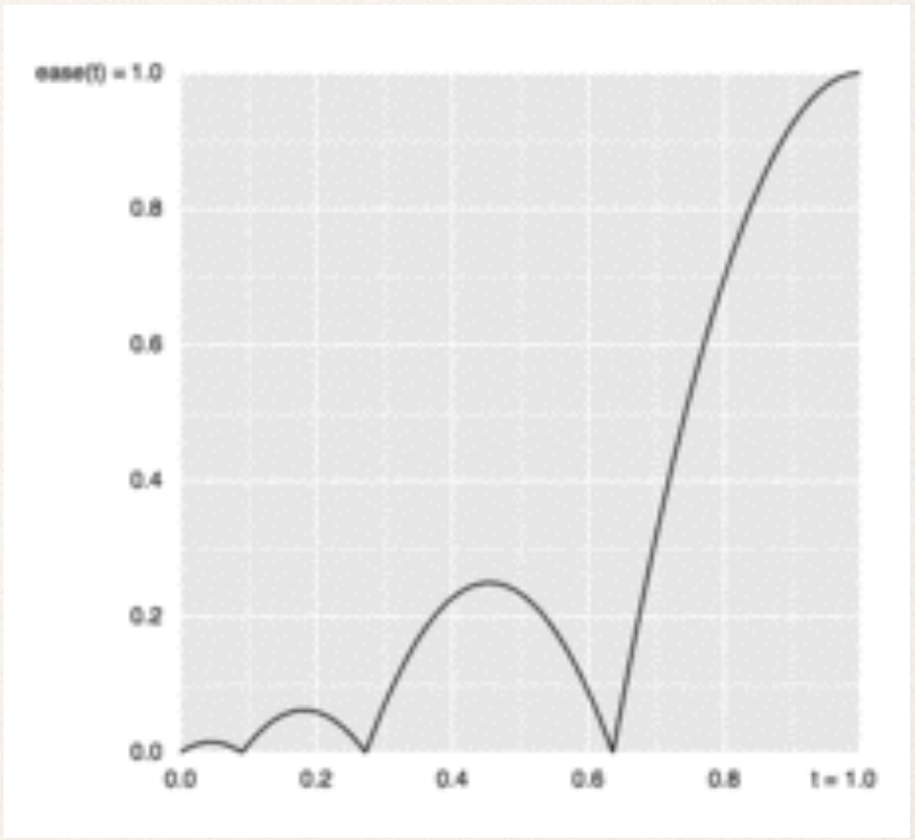
cubic-in-out

Lab 3-1: Ease function in Animation

- d3.js 는 ease 메서드를 통해 애니메이션의 easing-in & easing-out 을 지원
  - https://github.com/d3/d3-ease



linear



bounce

Lab 3-1: Ease function in Animation

- `.ease("linear")` 를 추가하여 움직임을 수정
- (참고) `.ease("bounce")` 로 고쳐보자.

Lab 4: 숫자와 텍스트 표시 1

- Pie chart 내부에 pie chart의 크기를 표시하고, 가운데에 총합을 표시

- Step 1: 총합을 표시할 Pie chart 내부 영역 준비

```
var arc =
d3.svg.arc().innerRadius(50).outerRadius(100)
```

Lab 4: 숫자와 텍스트 표시 1

- Step2: text 요소를 추가하여 숫자 표시
  - text 요소는 데이터셋에 따라 요소를 추가할 필요가 없기 때문에 `enter()`와 `data()`를 쓰지 않는다.
  - `d3.sum()`을 이용하면 데이터의 총 합을 구할 수 있다.

```
var textElements = d3.select("#myGraph")
  .append("text")
  .attr("class", "total")
  .attr("transform", "translate(" +
     svgWidth/2 + ", " + svgHeight/2 + ")")
  .text("Total: " + d3.sum(dataSet))
```

Lab 4: 숫자와 텍스트 표시 1

- Step3: 텍스트 스타일링

```
.total {
   font: 10pt "Helvetica Neue", Arial,
      Helvetica, Geneva, sans-serif;
   text-anchor: middle;
}
```

Lab 5: 숫자와 텍스트 표시 2

- g 요소를 추가한 후, g의 가운데 좌표에 텍스트를 표시

- Step1: g 요소 추가 후, path 를 g 요소 하위 요소로 추가

```
var pieElements = d3.select("#myGraph")
  .selectAll("g")
  .data(pie(dataSet))
  .enter()
  .append("g")
  .attr("transform", "translate(" + svgWidth/2
        + ", " + svgHeight/2 + ")")
```

Lab 5: 숫자와 텍스트 표시 2

- Step2: 숫자를 arc의 가운데에 표시

```
pieElements
  .append("text")// 데이터 수만큼 text 요소 추가
  .attr("class", "pieNum")
  .attr("transform", function(d, i){
    return "translate(" + arc.centroid(d) +
                          ")"
})
  .text(function(d, i){
    return d.value; // 값 표시
})
```

Lab 5: 숫자와 텍스트 표시 2

- `centroid()` 메서드
  - 도형의 중심 (무게중심)의 좌표를 반환
    - `arc.centroid()`
    - `rect.centroid()`
  - 복잡한 도형의 중심 좌표도 구할 수 있음

Lab 5: 숫자와 텍스트 표시 2

- Step3: 텍스트의 스타일링

```
.pieNum {
 font: bold 9pt "Helvetica Neue",
    Arial, Helvetica, Geneva, sans-serif;
 text-anchor: middle;
}
```

- Step3-1: 숫자 대신 레이블?

```
pleElements
...
.text(function(d, i){
        return dataLabel[i]
})
```

Lab 6: 여러 데이터의 로딩

- 2008~2014년 휴대폰 시장 점유율 데이터
  - pull-down menu 를 이용하여 여러 데이터를 선택하여 로딩

- Step1: 테스트 데이터 파일이름으로 변경
  - d3.csv("data/**mydata2008.csv**", function(error, data)

- Step2: 데이터 구조에 맞게 파일 로딩 부분 수정

```
for(var i = 0; i < data.length; i++) {
 dataSet.push(data[i].share)
 dataLabel.push(data[i].carrier)
}
```

Lab 6-1: 여러 데이터의 로딩

- Step1: d3 코드 전체를 함수화
  - 지금까지의 모든 코드를

    `function drawPie(filename) { … }`

    으로 묶어 준다.
  - **`drawPie("data/mydata2008.csv")`**로 만든 함수를 호출한다.
  - `drawPie()`의 패러미터를 바꾸어 주면 새로운 데이터가 로딩 됨.

Lab 6-1: 여러 데이터의 로딩

- Step1 (cont.): 코드 수정

```
drawPie("data/mydata2009.csv")

function drawPie(filename) {
 d3.csv(filename, function(error, data){
  ...
```

Lab 6-1: 여러 데이터의 로딩

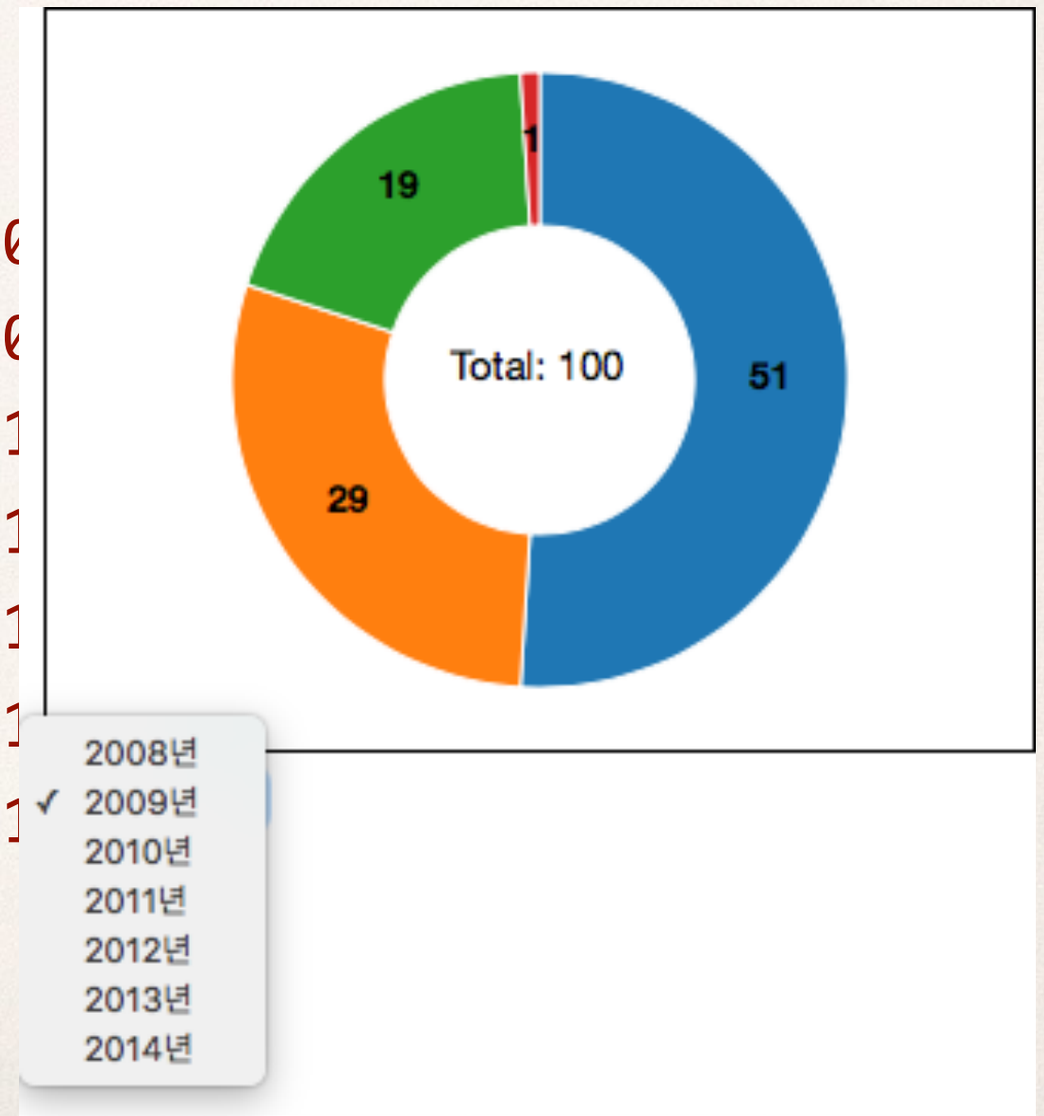
- Step2: HTML 파일 수정

```
<form>
  <select id="year">
    <option value="2008">2008년</option>
    <option value="2009">2009년</option>
    <option value="2010">2010년</option>
    <option value="2011">2011년</option>
    <option value="2012">2012년</option>
    <option value="2013">2013년</option>
    <option value="2014">2014년</option>
  </select>
</form>
```

Lab 6-1: 여러 데이터의 로딩

- Step2: HTML 파일 수정

```
<form>
 <select id="year">
    <option value="200
    <option value="201
    <option value="201
    <option value="201
    <option value="201
    <option value="201
 </select>
</form>
```



Lab 6-1: 여러 데이터의 로딩

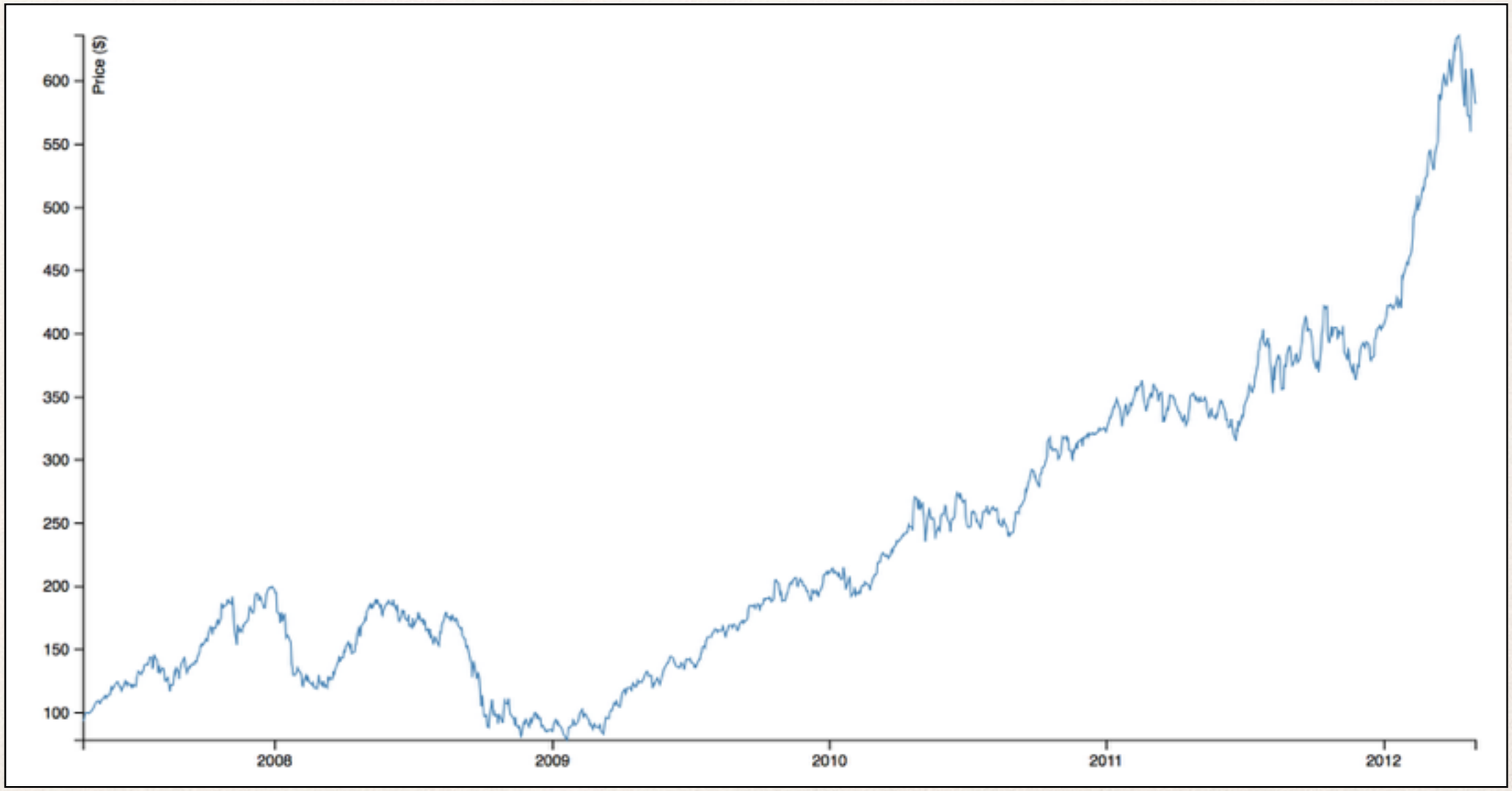
- Step3: 선택된 연도에 해당되는 파일 이름 처리

```
d3.select("#year").on("change", function() {
 d3.select("#myGraph").selectAll("*").remove()
 drawPie("data/mydata" + this.value + ".csv",
          this.value)
})
```

Line Graph

Line Graph

- Pie chart 처럼 준비된 layout은 없음
- SVG의 path 를 사용해서 line graph 생성

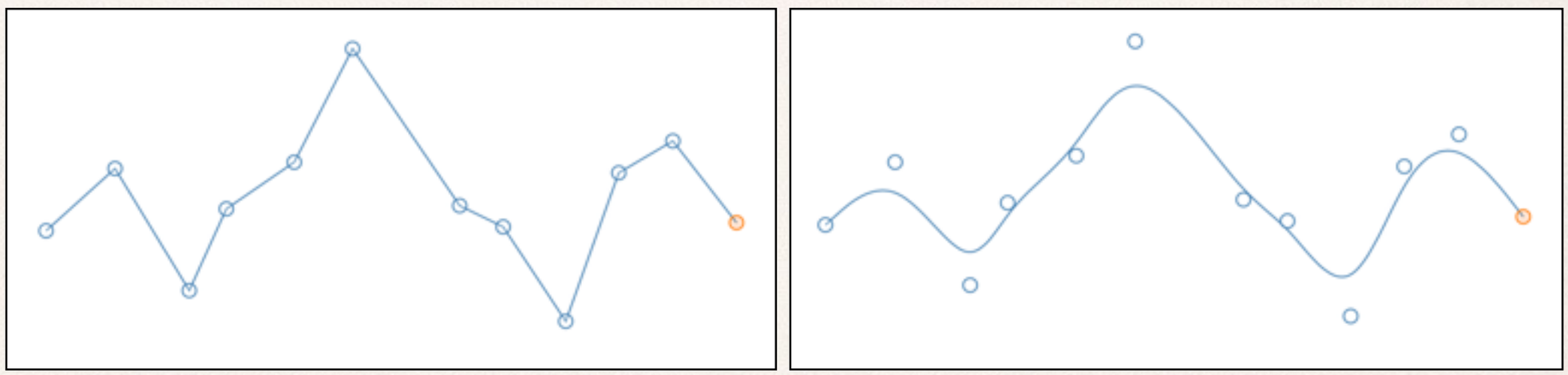


Line Graph

- `d3.svg.line()`
  - 두 점 간의 좌표를 연결하는 처리를 수행
  - x()와 y() 메서드의 내용을 정의해서 사용
    - **`line = d3.svg.line()`**
      `.x(…)`
      `.y(…)`
  - line 오브젝트가 만들어지면 path 의 속성으로 지정
    - **`.append("path")`**
      `.attr("d", line(data))`

Line Graph

- Interpolation
  - 두 점을 연결할 때의 보간법
    - ```
      line = d3.svg.line()
          .x(…)
          .y(…)
          .interpolate("linear")
      ```



linear　　basis

http://bl.ocks.org/mbostock/4342190
      ```

Lab 7: Line Graph

http://bl.ocks.org/mbostock/3883245

- Step1: 그래프의 크기와 margin, offset 등 설정

```
var margin = { top: 20, right: 20, bottom: 30,
left: 50 }
var width = 960 - margin.left - margin.right
var height = 500 - margin.top - margin.bottom

var svg = d3.select("body").append("svg")
  .attr("width", width + margin.left +
      margin.right)
  .attr("height", height + margin.top +
      margin.bottom)
  .append("g")
  .attr("transform", "translate(" + margin.left
      + "," + margin.top + ")")
```

Lab 7: Line Graph

- Step2: Data Loading

```
var parseDate = d3.time.format("%d-%b-
%y").parse
...
d3.tsv("data/mydata.tsv", function(error,
data) {
 data.forEach(function(d) {
    d.date = parseDate(d.date)
    d.close = +d.close
    console.log(d.date)
 })
...
})
```

Lab 7: Line Graph

- Step2: Data Loading

```
var parseDate = d3.time.format("%d-%b-
%y").parse
...
d3.tsv("data/my                    (error,
data) {
 data.forEach(function(d) {
    d.date = parseDate(d.date)
    d.close = +d.close
    console.log(d.date)
 })
...
})
```

텍스트 데이터를 date 오브젝트로 변환

텍스트 데이터를 int로 변환 => parseInt()

Lab 7: Line Graph

- Step3: d3.svg.line() 설정

```
var line = d3.svg.line()
    .x(function(d) { return x(d.date) })
    .y(function(d) { return y(d.close) })

...

d3.tsv(...) {
  svg.append("path")
    .datum(data)
    .attr("class", "line")
    .attr("d", line)
```

Lab 7: Line Graph

- Step3: d3.svg.line() 설정

```
var line = d3.svg.line()
    .x(function(d) { return x(d.date) })
    .y(function(d) { return y(d.close) })

...

d3.tsv(.
  svg.append("path")
    .datum(data)
    .attr("class", "line")
    .attr("d", line)
```

line에서 사용할 데이터를 불러옴

Lab 7: Line Graph

- Step4: x, y 축 설정

```
var x = d3.time.scale().range([0, width])
var y = d3.scale.linear().range([height, 0])

var xAxis =
    d3.svg.axis().scale(x).orient("bottom")
var yAxis =
    d3.svg.axis().scale(y).orient("left")

d3.tsv(...) {
  x.domain(d3.extent(data, function(d) {
    return d.date }))
  y.domain(d3.extent(data, function(d) {
    return d.close }))
```

Lab 7: Line Graph

- Step4: x, y 축 설정

domain과 range를 시간으로 설정

```
var x = d3.time.scale().range([0, width])
var y = d3.scale.linear().range([height, 0])

va                                    ale(x).orient("bottom")
va
    d3.svg.axis().scale(y).orient("left")

d3.tsv(...) {
  x.domain(d3.extent(data, function(d) {
    return d.da e }))
                                      data, function(d) {
                                      )
```

domain과 range를 linear 한 값으로 설정

date가 보여지는 range를 데이터 최소/최대로 설정

Lab 7: Line Graph

- Step5: 축 그리기

```
svg.append("g")
   .attr("class", "x axis")
   .attr("transform", "translate(0," +
      height + ")")
   .call(xAxis)

svg.append("g")
   .attr("class", "y axis")
   .call(yAxis)
```

Lab 7: Line Graph

- Step5 (cont.): 축 그리기

```
svg.append("g")
.attr("class", "y axis")
.call(yAxis)
.append("text")
.attr("transform", "rotate(-90)")
.attr("y", 6)
.attr("dy", ".71em")
.style("text-anchor", "end")
.text("Price ($)")
```

Lab 7: Line Graph

- Step6: CSS Styling

```
.axis path,
.axis line {
    fill: none;
    stroke: #000;
    shape-rendering: crispEdges;
}
.x.axis path {
 stroke: #000;
}
.line {
    fill: none;
    stroke: steelblue;
    stroke-width: 1px;
}
```

Assignment 4: Pie Chart + Line Graph

- 제출: 11/1 (자정)

Assignment 4: Pie Chart + Line Graph

- 공공데이터를 활용한 비주얼라이제이션
- 기본 데이터: 교통사고 현황
  - http://115.84.165.91/jsp/WWS7/WWSDS7100.jsp?re_stc_cd=10754&re_lang=kor
- 시각화 사례
  - 연도별 교통사고 발생 변화 추이
  - 지역별 교통사고 발생 빈도
  - 연령층 별 교통사고 발생 빈도

Assignment 4: Pie Chart + Line Graph

- 분석을 위한 자료 추가 가능
- 제출: 다음주 월요일 (11/1) 자정
- 파일 제출 방법
  - 본인의 firstname 으로 폴더를 만들고
  - html 파일은 반드시 index.html 로 이름을 부여한 후, 관련 파일을 모두 같은 폴더에 넣고 (주의: 한글 파일 이름 x)
  - zip 으로 압축해서 제출

Assignment 4: Pie Chart + Line Graph

- 참고 사이트
  - Pie and Line Chart: http://codepen.io/stefanjudis/pen/gkHwJ
  - Life Expectancy: http://projects.flowingdata.com/life-expectancy/
  - 60 Years of French First Names: http://dataaddict.fr/prenoms/
  - Dashboard: http://bl.ocks.org/NPashaP/96447623ef4d342ee09b
  - Pie charts labels: http://bl.ocks.org/dbuezas/9306799
  - Multi-Series Line Chart: http://bl.ocks.org/mbostock/3884955
  - X-Value Mouseover: http://bl.ocks.org/mbostock/3902569

Questions...?