

# | D3.js Tutorial

조재민

Human-Computer Interaction Lab

Dept. of Computer Science & Engineering

Seoul National University

Find me on Github (<http://github.com/e->) or  
visit my blog (<http://www.jaeminjo.com>)



version 0.1.zip  
66.37KB | 05.16까지

열기 | 폴더열기



version 0.1 - 수정.zip  
66.37KB | 05.16까지

열기 | 폴더열기

오후 4:42



장유리



version 0.1 - 수정 - 유리  
수정.zip  
66.37KB | 05.16까지

오후 4:43

저장 | 다른이름 저장



version 1.0 최종.zip  
66.37KB | 05.16까지

열기 | 폴더열기

오후 4:43



장유리



version 1.0 - 최종 최종.zi  
p  
66.37KB | 05.16까지

오후 4:43

저장 | 다른이름 저장



version 1.0 최종 최종 진짜  
최종.zip  
66.37KB | 05.16까지

열기 | 폴더열기

오후 4:44



장유리



version 1.0 - 최종 최종  
진짜 최종 final.zip  
66.37KB | 05.16까지

오후 4:44

저장 | 다른이름 저장

### Git

- Git: 형상 관리 도구
  - 왜 갑자기 작동이 안되지? 어제 저녁 버전으로 돌아가고 싶은데..
  - 네가 A 고치는 동안 난 B 고칠테니 끝나고 합치자.
  - 팀 프로젝트를 하는데 왜 이 친구는 코딩을 하나도 안 했지?
- 협업의 기본

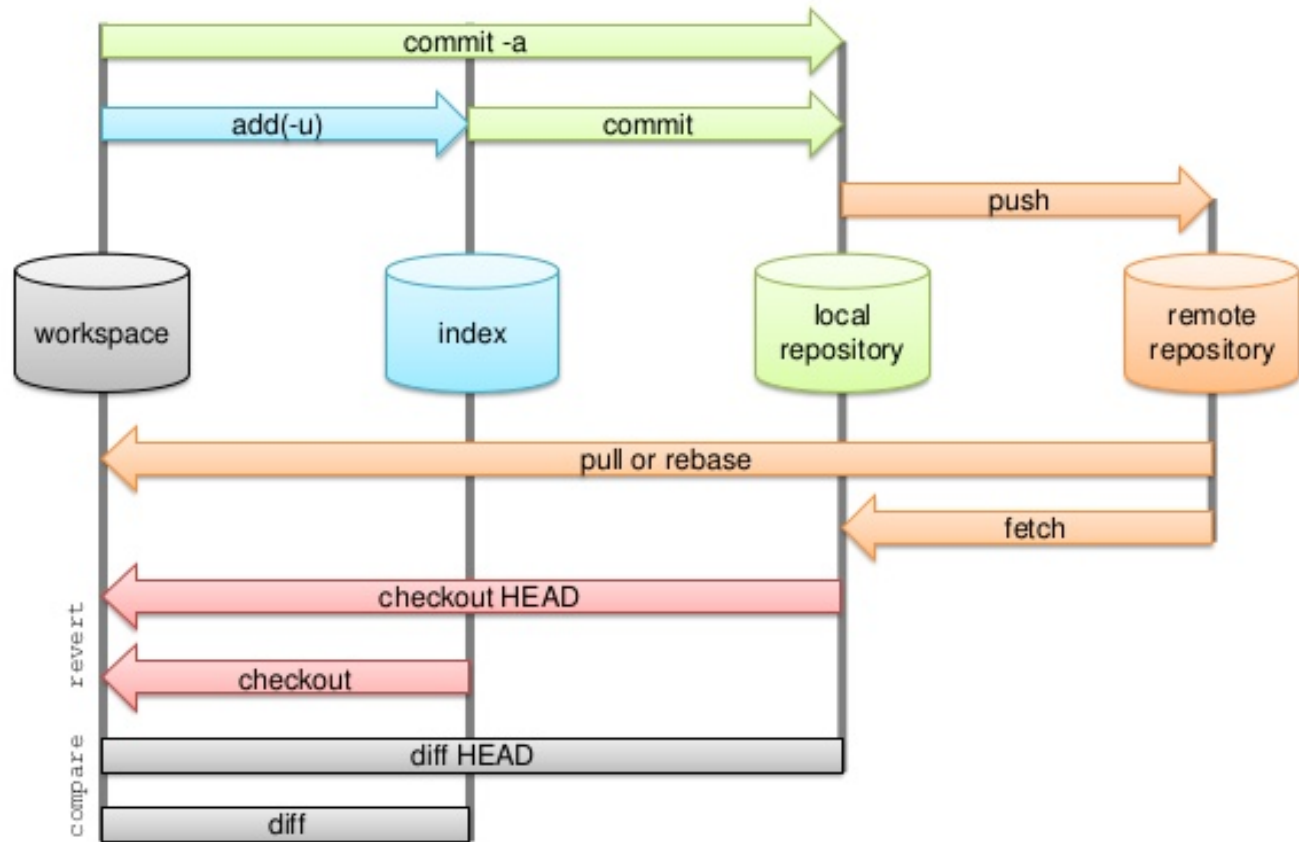
### Github

- Github: 세계 최대의 오픈 소스 커뮤니티
  - <https://github.com/>
  - Git 프로젝트를 무료로 호스팅 해줌
  - diff, github pages, wiki, issue tracking 등 여러 편의 기능 제공
- <https://github.com/google>
- <https://github.com/microsoft>
- <https://github.com/apache>

# Git

e.g., git commit -a

Git Data Transport Commands  
<http://osteele.com>



### Hello, Git!

- `mkdir my_project`
- `cd my_project`
- `echo "Hello, Git!" >> README.md`
- `git init`
- `git add README.md`
- `git commit -m "first commit"`
- `git remote add origin git@github.com:e-/dummy.git`
- `git push -u origin master`

### 기본 작업 패턴

- git pull
  - vim edit.cpp (기존 파일 수정)
  - vim new.cpp (새로 만듦)
  - git add .
  - git commit -m "Don't leave a commit message like this"
  - git push origin master
- 
- 내가 pull 하고 edit.cpp 를 수정하는 동안 팀원이 edit.cpp를 수정한다면? => conflict

### Conflict가 발생하면?

- 한 파일의 같은 부분을 여러 명이 수정하면 자동 병합 불가
- 이 경우 pull 시 conflict가 발생





## Conflict가 발생하면?

- Conflict가 발생한 파일을 열어 내 코드와 서버에 있는 코드 (HEAD) 중 어떤 코드를 사용할 지 수동으로 수정
- Commit & Push

```
1 <<<<<< HEAD
2
3 Here is the original change.
4 =====
5 Here is the modified change.
6 >>>>>> 58326c301d09b58f3ac23d616e73f7b478424cc5
7
```

## Git

- 참고 자료

- <https://help.github.com/articles/set-up-git/>
- <https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>
- <https://rogerdudler.github.io/git-guide/index.ko.html>
- <https://nolboo.kim/blog/2013/10/06/github-for-beginner/>

In case of fire



1. git commit



2. git push



3. leave building



# Data-Driven Documents

[illegible]

## 준비물

### 1. 실습 자료 다운로드 받기

<https://github.com/SNU-HCIL/2016-SNUBDA-Summer-Engineering/archive/master.zip>

### 2. 텍스트 에디터 및 웹 브라우저 준비하기

- 텍스트 에디터: Sublime, Vim, Emacs, Visual Studio, ...
- <http://www.sublimetext.com/2>
- 윈도우 기본 메모장은 인코딩에 주의
- 웹 브라우저: Chrome, Firefox, Edge, ...
- IE의 경우 로컬 스크립트 실행여부를 묻는 알람창이 뜨면 “차단된 콘텐츠 허용” 클릭

# HTML

- 차트를 그리는 태그는?
  - 캔버스 태그 (<canvas>)
  - SVG 태그 (<svg>)
- D3.js 의 경우 SVG 를 기본 그리기 요소로 사용
  - 그러나 성능을 위해 실제 연구에서는 canvas를 이용하는 경우도 있음!

## HTML

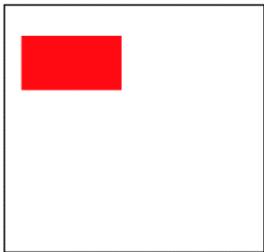
- SVG (Scalable Vector Graphics)

- 텍스트 기반의 벡터 이미지 포맷
- 대부분의 현대 웹 브라우저들이 SVG를 지원

예) Chrome, Safari, Firefox, IE, ...

- 기본적인 시각적 요소들 제공

예) 사각형 <rect>, 원 <circle>, 타원 <ellipse>, Path <path>, ...



```
<html><body>
  <svg width="150" height="150">
    <rect x="10" y="20" width="60" height="30"
fill="red"/>
  </svg>
</body></html>
```

## Basic SVG Elements (geometric attributes)



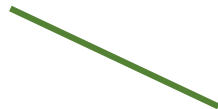
rect (x,y,width,height)



circle (cx, cy, r)



ellipse (cx, cy, rx, ry)



line (x1, y1, x2, y2)



path (d)



polygon (points)

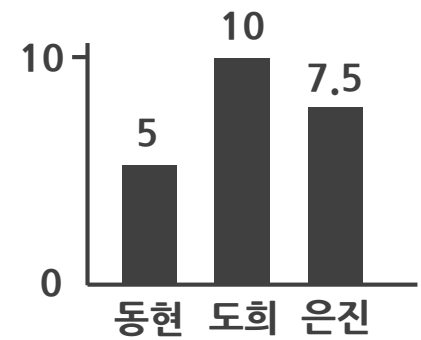
# D3.js 기초



## Binding Data to Visualization

이름	값
동현	5
도희	10
은진	7.5

Table

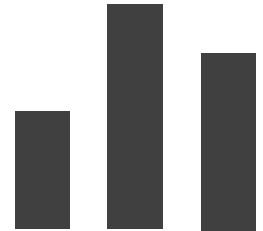


Visualization

## Binding Data to Visualization (Cont'd)

이름	값
동현	5
도희	10
은진	7.5

Table



Visualization

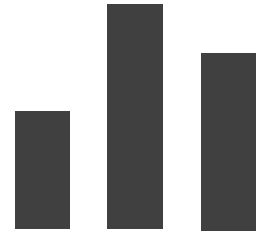
## Binding Data to Visualization (Cont'd)

이름	값
동현	5
도희	10
은진	7.5

Table

```
[  
  {"이름": "동현",  
   "값": 5},  
  
  {"이름": "도희",  
   "값": 10},  
  
  {"이름": "은진",  
   "값": 7.5}  
]
```

JSON



Visualization

## Binding Data to Visualization (Cont'd)

이름	값
동현	5
도희	10
은진	7.5

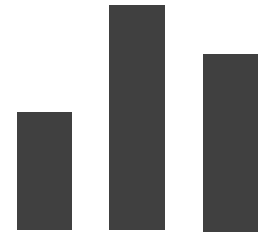
Table

```
[  
  {"이름": "동현",  
   "값": 5},  
  {"이름": "도희",  
   "값": 10},  
  {"이름": "은진",  
   "값": 7.5}  
]
```

JSON

```
<svg>  
  <rect ...></rect>  
  <rect ...></rect>  
  <rect ...></rect>  
</svg>
```

SVG



Visualization

## Binding Data to Visualization (Cont'd)

이름	값
동현	5
도희	10
은진	7.5

Table

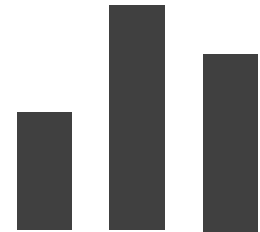
```
[  
  {"이름": "동현",  
   "값": 5},  
  {"이름": "도희",  
   "값": 10},  
  {"이름": "은진",  
   "값": 7.5}  
]
```

JSON

*D3.js*

```
<svg>  
<rect ...></rect>  
<rect ...></rect>  
<rect ...></rect>  
</svg>
```

SVG



Visualization

### 데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



## 데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```

화면 상에는 그대로

## 데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="150"></rect>  
</svg>
```

- Update
- Enter
- Exit



## 데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5}  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="150"></rect>  
  <rect height="75"></rect>  
</svg>
```

- Update
- Enter
- Exit

## 데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="150"></rect>  
  <rect height="75"></rect>  
</svg>
```

- Update
- Enter
- Exit

## 데이터와 노드간의 조인 (join)

```
[  
  {"이름": "동헌", "값": 5},  
  {"이름": "도희", "값": 10},  
]
```

```
<svg>  
  <rect height="50"></rect>  
  <rect height="100"></rect>  
</svg>
```



데이터가 업데이트 됨

```
[  
  {"이름": "도희", "값": 15},  
  {"이름": "은진", "값": 7.5},  
]
```

```
<svg>  
  <rect height="150"></rect>  
  <rect height="75"></rect>  
</svg>
```

- Update
- Enter
- Exit

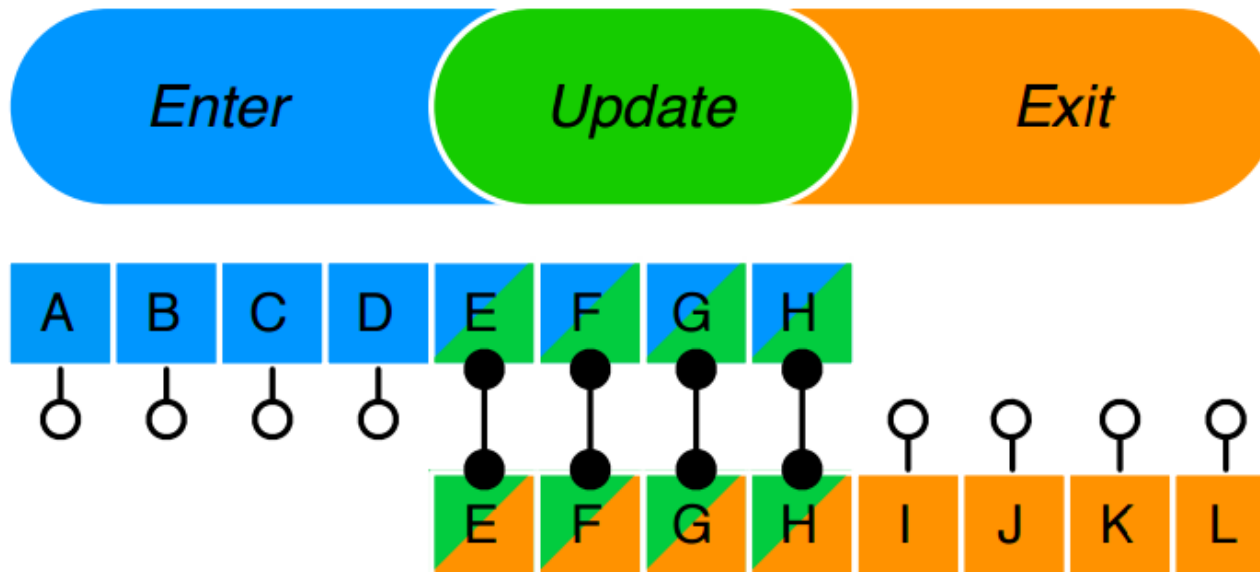
화면상의 차트와 데이터가 일치

## 데이터와 노드간의 조인 (join)

자바스크립트 상의  
데이터 (JSON)HTML (SVG) 상의  
노드

Data

Nodes



### D3.js의 기본 패턴

1. 화면상의 SVG 요소 선택
2. 데이터와 연결
3. 화면에 없는 SVG 요소 만들기
4. 화면에 있는 SVG 요소 업데이트

### D3.js의 기본 패턴

화면상의 SVG 요소 선택

데이터와 연결

화면에 없는 SVG 요소 만들기

화면에 있는 SVG 요소 업데이트

### D3.js의 기본 패턴

`d3.selectAll('circle')` – 화면상의 SVG 요소 선택

데이터와 연결

화면에 없는 SVG 요소 만들기

화면에 있는 SVG 요소 업데이트

## 기본적인 API

- `d3.select(selector)` => *selection*
  - *selector* 에 해당하는 요소들을 하나 선택한 셀렉션을 반환
  - `d3.select('body')` => body 태그를 선택
- `d3.selectAll(selector)` => *selection*
  - *selector* 에 해당하는 요소들을 모두 선택한 셀렉션을 반환
  - `d3.selectAll('rect')` => 페이지내 모든 rect를 선택
  - `d3.select('svg').selectAll('rect')` => svg 태그내에 있는 모든 rect를 선택



### D3.js의 기본 패턴

`d3.selectAll('circle')` – 화면상의 SVG 요소 선택

데이터와 연결

화면에 없는 SVG 요소 만들기

화면에 있는 SVG 요소 업데이트

### D3.js의 기본 패턴

`d3.selectAll('circle')` – 화면상의 SVG 요소 선택

데이터와 연결

화면에 없는 SVG 요소 만들기

화면에 있는 SVG 요소 업데이트

### D3.js의 기본 패턴

```
var my_data = [{x: 10, y:20}, {x: 30, y: 40}];
```

d3.selectAll('circle') – 화면상의 SVG 요소 선택

데이터와 연결

화면에 없는 SVG 요소 만들기

화면에 있는 SVG 요소 업데이트

### D3.js의 기본 패턴

```
var my_data = [{x: 10, y:20}, {x: 30, y: 40}];
```

d3.selectAll('circle') – 화면상의 SVG 요소 선택

**.data(my\_data)** – 데이터와 연결

화면에 없는 SVG 요소 만들기

화면에 있는 SVG 요소 업데이트

## 기본적인 API

- `selection.data(array)`
  - `selection`에 선택된 요소들과 *array*를 조인 (join)
  - 기본적으로 `update` 셀렉션을 가져옴

### D3.js의 기본 패턴

```
var my_data = [{x: 10, y:20}, {x: 30, y: 40}];
```

d3.selectAll('circle') – 화면상의 SVG 요소 선택

.data(my\_data) – 데이터와 연결

화면에 없는 SVG 요소 만들기

화면에 있는 SVG 요소 업데이트

### D3.js의 기본 패턴

```
var my_data = [{x: 10, y:20}, {x: 30, y: 40}];
```

d3.selectAll('circle') – 화면상의 SVG 요소 선택

.data(my\_data) – 데이터와 연결

.enter() – 화면에 없는 SVG 요소 만들기

.append('circle')

화면에 있는 SVG 요소 업데이트

속성 변경

## 기본적인 API

- `selection.data(array).enter()`
  - `enter` 선택션을 가져옴
  - `selection.data(array).enter().append('rect')`
- `selection.data(array).exit()`
  - `exit` 선택션을 가져옴
  - `selection.data(array).exit().remove()`
- `selection.append(tag) => selection`
  - 선택션에 자식으로 `tag`를 추가
  - `d3.select('body').append('div')`



### D3.js의 기본 패턴

```
var my_data = [{x: 10, y:20}, {x: 30, y: 40}];
```

d3.selectAll('circle') – 화면상의 SVG 요소 선택

.data(my\_data) – 데이터와 연결

.enter() – 화면에 없는 SVG 요소 만들기

.append('circle')

화면에 있는 SVG 요소 업데이트

### D3.js의 기본 패턴

```
var my_data = [{x: 10, y:20}, {x: 30, y: 40}];
```

d3.selectAll('circle') – 화면상의 SVG 요소 선택

.data(my\_data) – 데이터와 연결

.enter() – 화면에 없는 SVG 요소 만들기

.append('circle')

.attr('cx', function(d){return d.x;})

– 화면에 있는 SVG 요소 업데이트

## 기본적인 API

- `selection.attr(key, value) => selection`
  - selection에 선택된 요소들의 속성을 변경
  - `<rect height="200"></rect>`
  - `d3.select('rect').attr('height', '300')`
  - `<rect height="300"></rect>`
- `selection.text(text) => selection`
  - selection에 선택된 요소들의 내용을 변경
  - `<text>hello</text>`
  - `d3.select('text').text('world')`
  - `<text>world</text>`

### D3.js의 기본 패턴

```
var my_data = [{x: 10, y:20}, {x: 30, y: 40}];
```

```
d3.selectAll('circle')
```

```
.data(my_data)
```

```
.enter()
```

```
  .append('circle')
```

```
  .attr('cx', function(d){return d.x;})
```

# D3.js로 시각화 만들기

### Bar chart 만들기 (1)

1. 1\_barchart 디렉토리 안에 있는 barchart\_skeleton.js를 텍스트 에디터로 열기
  2. barchart\_skeleton.html을 브라우저에서 열기
  3. js 파일을 수정한 후 저장하고 브라우저를 새로고침하여 결과를 확인
- 모든 html파일은 실습 내내 수정하실 필요가 없습니다.
  - 완성된 코드는 barchart.js에 있습니다.

### Bar chart 만들기 (2)

```
var data = [  
  {"name": "Sally", "value": 5},  
  {"name": "Tom", "value": 10},  
  {"name": "John", "value": 7.5}  
]
```

```
<script src="https://d3js.org/d3.v3.min.js"></script>
```

```
<svg>
```

```
</svg>
```

```
// <html>, <body>는 생략
```

Javascript

HTML

### Bar chart 만들기 (3)

```
var data = ...
```

```
var svg = d3.select('svg')
```

```
<svg>
```

```
</svg>
```

Javascript

HTML



### Bar chart 만들기 (4)

```
var data = ...
```

```
<svg>
```

```
</svg>
```

```
var svg = d3.select('svg')
```

```
svg
```

- svg 내부에는 <rect>가 하나도 없으므로 빈 선택션을 돌려줌

```
.selectAll('rect')
```

Javascript

HTML

### Bar chart 만들기 (5)

```
var data = ...
```

```
var svg = d3.select('svg')
```

```
svg
```

```
.selectAll('rect') // 빈 선택션
```

```
.data(data)
```

```
<svg>
```

```
</svg>
```

Javascript

HTML

### Bar chart 만들기 (6)

```
var data = ...
```

```
<svg>
```

```
</svg>
```

```
var svg = d3.select('svg')
```

```
svg
```

```
  .selectAll('rect')
```

```
  .data(data)
```

```
  .enter()
```

Javascript

HTML

## Bar chart 만들기 (7)

```
var data = ...  
    { "name": "Sally", "value": 5 }  
    { "name": "Tom", "value": 10 }  
var svg = d3.select('svg')  
    { "name": "John", "value": 7.5 }  
  
    <svg>  
    .  
    .  
    .  
    </svg>  
  
svg  
    .selectAll('rect')  
    .data(data)  
    .enter()
```

Javascript

HTML

## Bar chart 만들기 (8)

```
var data = ...
```

```
{ "name": "Sally", "value": 5 }
```

```
{ "name": "Tom", "value": 10 }
```

```
var svg = d3.select('svg')
```

```
{ "name": "John", "value": 7.5 }
```

```
svg
```

```
.selectAll('rect')
```

```
.data(data)
```

```
.enter()
```

```
.append('rect')
```

Javascript

```
<svg>
```

```
<rect></rect>
```

```
<rect></rect>
```

```
<rect></rect>
```

```
</svg>
```

자바스크립트로 HTML 요소를 생성  
(코드 입력 X)

HTML

## Bar chart 만들기 (9)

```
var data = ...
```

```
var svg = d3.select('svg')    {"name": "Sally", "value": 5}
```

```
    {"name": "Tom", "value": 10}
```

```
svg
```

```
    {"name": "John", "value": 7.5}
```

```
  .selectAll('rect')
```

```
  .data(data)
```

```
  .enter()
```

```
    .append('rect')
```

```
    .attr('width', 100)
```

```
    .attr('height', 20)
```

```
    .style('fill', 'steelblue')
```

```
<svg>
```

```
  <rect width="100" height="20"
  style="fill:steelblue"></rect>
```

```
  <rect width="100" height="20"
  style="fill:steelblue"></rect>
```

```
  <rect width="100" height="20"
  style="fill:steelblue"></rect>
```

```
</svg>
```



## Bar chart 만들기 (10)

```
var data = ...
```

```
var svg = d3.select('svg')
```

```
svg
  .selectAll('rect')
  .data(data)
  .enter()
    .append('rect')
      .attr('width', function(d, i){
        return d.value * 10;
      })
      .attr('height', 20)
      .style('fill', 'steelblue')
```

```
<svg>
```

```
  <rect width="50" height="20"
    style="fill:steelblue" ></rect>
```

```
  <rect width="100" height="20"
    style="fill:steelblue" ></rect>
```

```
  <rect width="75" height="20"
    style="fill:steelblue"></rect>
```

```
</svg>
```



## Bar chart 만들기 (11)

```

var data = ...

var svg = d3.select('svg')

svg
    .selectAll('rect')
    .data(data)
    .enter()
    .append('rect')
    .attr('width', function(d, i){
        return d.value * 10;
    })
    .attr('height', 20)
    .style('fill', 'steelblue')
    .attr('transform', function(d, i){
        return 'translate(0,' + i * 30 + ')';
    });

```

```

<svg>
  <rect width="50" height="20"
    style="fill:steelblue"
    transform="translate(0, 0)"></rect>
  <rect width="100" height="20"
    style="fill:steelblue"
    transform="translate(0, 30)"></rect>
  <rect width="75" height="20"
    style="fill:steelblue"
    transform="translate(0, 60)"></rect>
</svg>

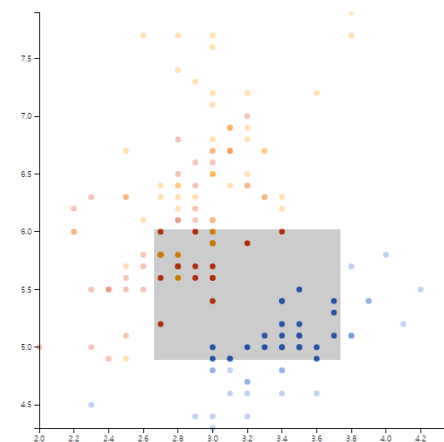
```





## 산점도 만들기 (1)

1. 2\_scatterplot 디렉토리 안에 있는 scatterplot\_skeleton.js를 텍스트 에디터로 열기
  2. scatterplot\_skeleton.html을 브라우저에서 열기
  3. js 파일을 수정한 후 저장하고 브라우저를 새로고침하여 결과를 확인
- 모든 html파일은 실습 내내 수정하실 필요가 없습니다.
  - 완성된 코드는 scatterplot.js에 있습니다.



MEAN(width) = 3.15, MEAN(length) = 5.42

## 산점도 만들기 (2)

- 사용할 데이터: data/iris.tsv
- 붓꽃의 종류별 꽃잎과 꽃받침의 길이와 너비가 탭으로 구분되어있음
  - setosa, versicolor, virginica
- [꽃받침 길이] [꽃받침 너비] [꽃잎 길이] [꽃잎 너비] [종류]
- 꽃받침의 길이와 너비로 산점도를 그리고 이를 개선해 봅시다.

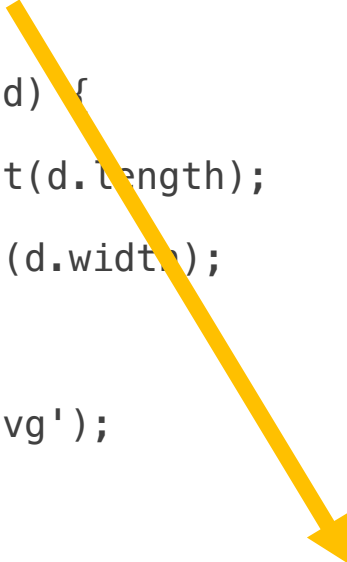
```
length width petalLength petalWidth species
5.1 3.5 1.4 0.2 setosa
4.9 3.0 1.4 0.2 setosa
4.7 3.2 1.3 0.2 setosa
4.6 3.1 1.5 0.2 setosa
5.0 3.6 1.4 0.2 setosa
5.4 3.9 1.7 0.4 setosa
```

## 산점도 만들기 (3)

```
d3.tsv('../data/iris.tsv',  
function(error, data) {  
  data.forEach(function(d) {  
    d.length = parseFloat(d.length);  
    d.width = parseFloat(d.width);  
  });  
  var svg = d3.select('svg');  
  
  ...  
  
});
```

```
<svg>  
</svg>
```

// <html>, <body> 및 d3.js를 로드하는 코드는 생략



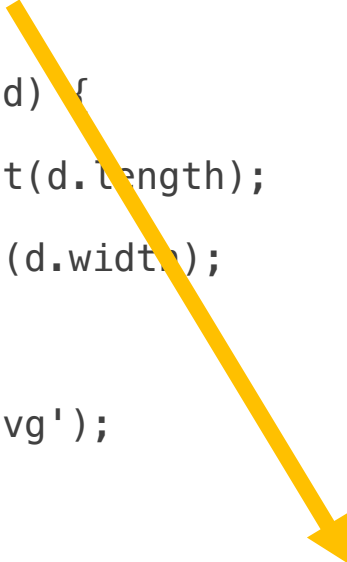
length	width	petalLength	petalWidth	species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

## 산점도 만들기 (4)

```
d3.tsv('../data/iris.tsv',  
function(error, data) {  
  data.forEach(function(d) {  
    d.length = parseFloat(d.length);  
    d.width = parseFloat(d.width);  
  });  
  var svg = d3.select('svg');  
  
  ...  
  
});
```

```
<svg>  
</svg>
```

// <html>, <body> 및 d3.js를 로드하는 코드는 생략



length	width	petalLength	petalWidth	species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

브라우저의 보안 정책으로 인해 로컬 스크립트  
파일에서 외부 서버 파일 불러오기가 불가능

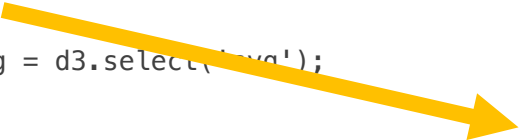
데이터 로딩 코드는 미리 작성되어 있으니 아래부터  
작성하시면 됩니다.

## 산점도 만들기 (5)

```
data.forEach(function(d) {  
  d.length = parseFloat(d.length);  
  d.width = parseFloat(d.width);  
});  
var svg = d3.select('svg');
```

```
<svg>  
</svg>
```

// <html>, <body> 및 d3.js를 로드하는 코드는 생략



```
[  
  {"length": 5.1, "width": 3.5, "species": "setosa"},  
  ...  
]
```

## 산점도 만들기 (6)

```
data.forEach(function(d) {  
  d.length = parseFloat(d.length);  
  d.width = parseFloat(d.width);  
});  
var svg = d3.select('svg');
```

svg

```
.selectAll('circle')  
.data(data)  
.enter()  
  .append('circle')
```

```
<svg>  
  <circle></circle>  
  <circle></circle>  
  <circle></circle>  
  <circle></circle>  
  ...  
</svg>
```

## 산점도 만들기 (7)

```

svg
  .selectAll('circle')
  .data(data)
  .enter()
    .append('circle')
    .attr('r', 3.5)
    .attr('cx', function(d) { return d.width * 100; })
    .attr('cy', function(d) { return d.length * 100; })

```

```

<svg>
  <circle r="3.5" cx="350" cy="510"></circle>
  ...
</svg>

```

```

[
  { "length": 5.1, "width": 3.5, "species": "setosa" },
  ...
]

```



### 산점도 만들기 (8)

1. 점의 위치를 상수 (100)을 곱하는 것이 아니라 조금 더 일반적으로 구할 수 있지 않을까?
2. X, Y 축을 보여주면 더 좋지 않을까?
3. 데이터에는 3가지 꽃의 종류가 있는데 종류에 따라 점의 색깔을 다르게 할 수 있지 않을까?





## 산점도 만들기 (9)

## • Scale?

- 데이터의 관측 값과 화면 상의 픽셀 값을 연결해주는 함수

Length (mm)

5.2  
3.8  
3.6

정의역 (domain)



Position (pixel)

260  
190  
180

치역 (range)

## 산점도 만들기 (10)

```
var width = 500, height = 500;
var x = d3.scale.linear()
    .domain([
        d3.min(data, function(d){return d.width;}),
        d3.max(data, function(d){return d.width;})
    ])
    .range([0, width]);

svg
    .selectAll('circle')
    .data(data)
    .enter()
    .append('circle')
    .attr('r', 3.5)
    .attr('cx', function(d) { return x(d.width); })
    .attr('cy', function(d) { return d.length * 100; })
    .exit();
```

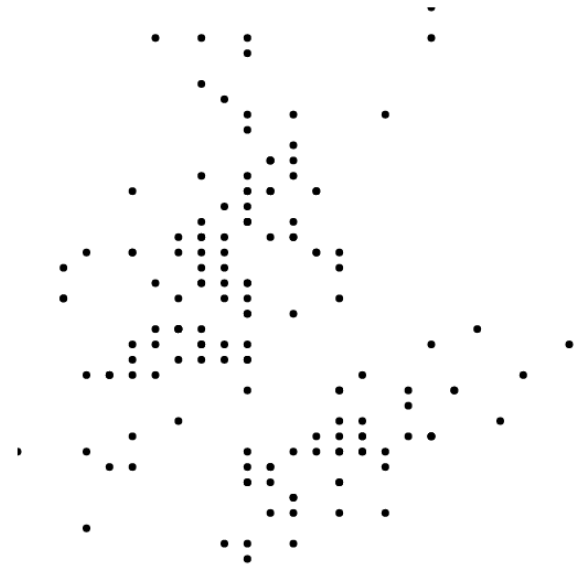


## 산점도 만들기 (11)

```
var width = 500, height = 500;
var x = d3.scale.linear()....;
var y = d3.scale.linear()
    .domain([
        d3.min(data, function(d){return d.length;}),
        d3.max(data, function(d){return d.length;})
    ])
    .range([height, 0]);

svg
    .selectAll('circle')
    .data(data)
    .enter()
    .append('circle')
    .attr('r', 3.5)
    .attr('cx', function(d) { return x(d.width); })
    .attr('cy', function(d) { return y(d.length); })
    });
```

\* 왼쪽 아래를 원점으로 만들기 위해  
range를 거꾸로 넣었음에 주의



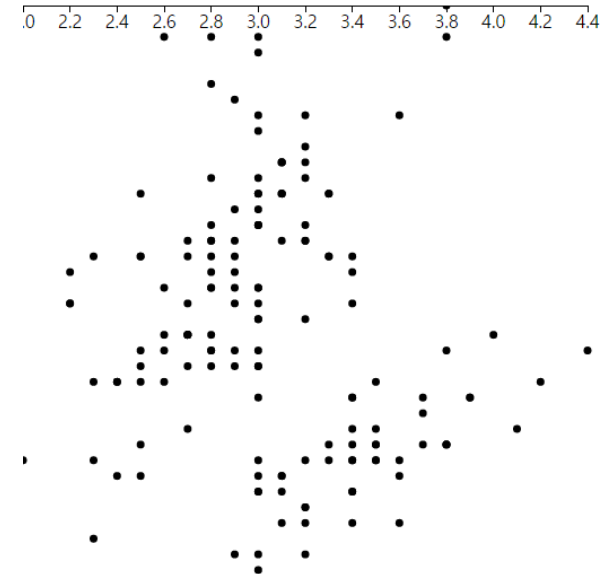
## 산점도 만들기 (12)

```
// .. 앞서 만든 코드에 추가
```

```
var x = d3.scale.linear()...;
```

```
var xAxis = d3.svg.axis()  
    .scale(x)  
    .orient('bottom');
```

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



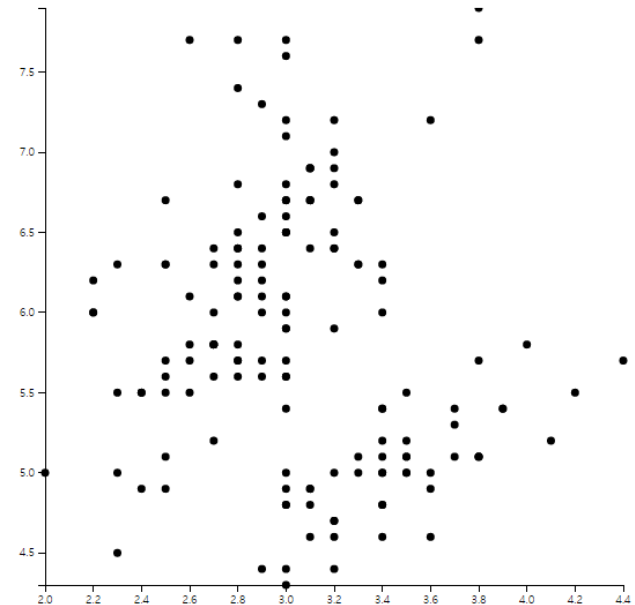
## 산점도 만들기 (13)

```
// .. 앞서 만든 코드에 추가
```

```
var yAxis = d3.svg.axis()  
    .scale(y)  
    .orient('left');
```

```
svg  
    .append('g')  
    .attr('class', 'y axis')  
    .attr('transform', 'translate(50, 0)')  
    .call(yAxis)
```

Y축을 표시할 공간을 확보하기 위해 x 스케일의 range를 [0, width] 가 아닌 [50, width + 50] 으로 바꿔준다

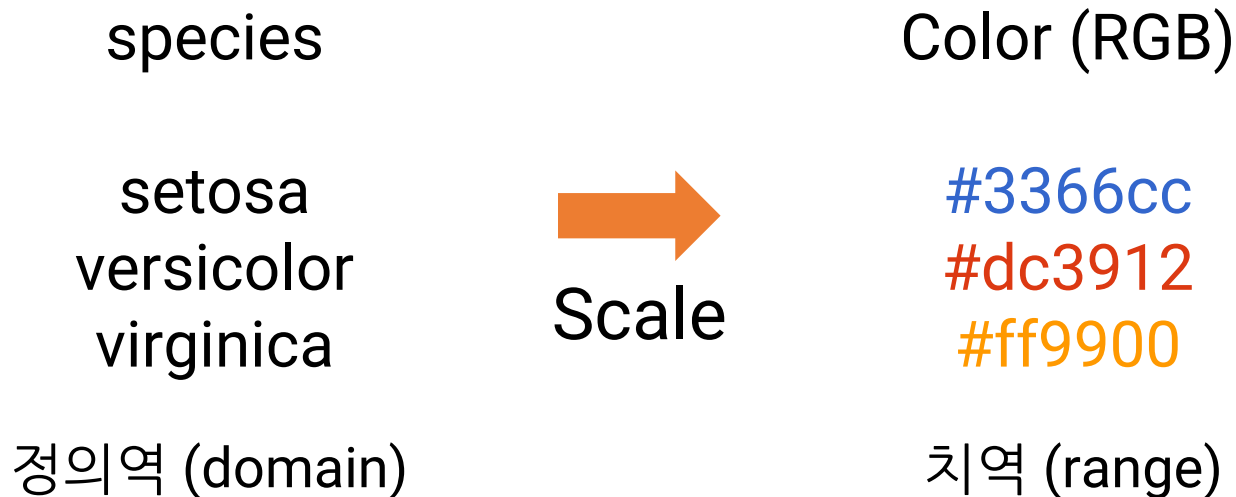


```
var x = d3.scale.linear()  
    .range([50, width + 50]) // range([0, width])
```

## 산점도 만들기 (14)

## • Color Scale?

- 데이터의 관측 값과 화면 상의 픽셀 값컬러코드를 연결해주는 함수



## 산점도 만들기 (15)

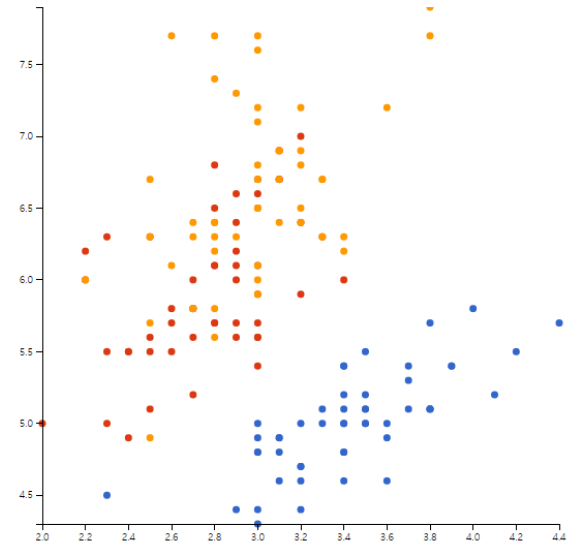
```

var width = 500, height = 500;
var x = d3.scale.linear()...;
var y = d3.scale.linear()...;

var color = d3.scale.ordinal()
    .domain(['setosa', 'versicolor', 'virginica'])
    .range(['#3366cc', '#dc3912', '#ff9900'])

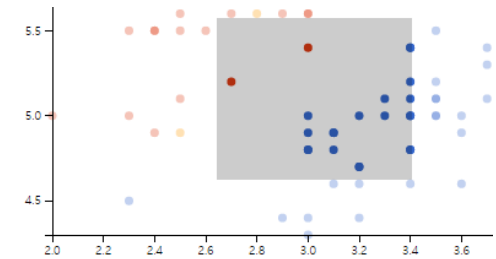
svg
    .selectAll('circle')
    .data(data)
    .enter()
        .append('circle')
        .attr('r', 3.5)
        .attr('cx', function(d) { return x(d.width); })
        .attr('cy', function(d) { return y(d.length); })
        .style('fill', function(d) { return color(d.species); })
    });

```



## 산점도 만들기 (16)

- 브러싱이란 분석자가 시각화의 관심있는 일부를 선택할 수 있도록 하여 보다 심화된 분석을 도와주는 인터랙션
- 산점도에서 관심있는 점들만 선택하고 선택된 점들의 평균 너비와 길이를 보이도록 해봅시다.



MEAN(width) = 3.20, MEAN(length) = 5.00





## 산점도 만들기 (17)

```
var brush = d3.svg.brush()
```

```
  .x(x)
```

```
  .y(y)
```

- 브러시 생성 후 x축 및 y축 스케일 지정
- 이를 통해 d3에서 스크린 좌표를 데이터 값으로 변환 가능

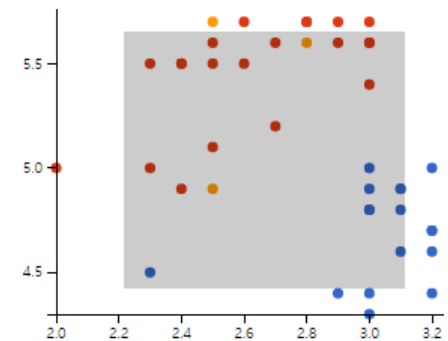
```
svg
```

```
  .append('g')
```

```
  .attr('class', 'brush')
```

```
  .call(brush)
```

- 새로운 g 엘리먼트를 추가하고 앞서 만든 brush 를 적용



MEAN(width) = , MEAN(length) =

## 산점도 만들기 (18)

```
var brush = d3.svg.brush()  
  .x(x)  
  .y(y)  
  .on('brush', update)  
  .on('brushend', update)
```

```
function update() {  
  // ...  
}
```

- 사용자가 드래그를 하거나 끝내는 이벤트를 감지하여 이때 선택된 점들의 평균 계산

## 산점도 만들기 (19)

```
function update() {  
    var extent = brush.extent();  
    var widthRange = [extent[0][0], extent[1][0]];  
    var lengthRange = [extent[0][1], extent[1][1]];  
    var widthSum = 0, lengthSum = 0, n = 0;  
  
    // ...  
}
```

- 브러싱 된 영역을 가져옴
- X축상의 선택된 꽃받침 너비의 범위
- Y축상의 선택된 꽃받침 길이의 범위
- 평균을 구하기 위해 변수 초기화

## 산점도 만들기 (20)

```
function update() {  
  // ...  
  svg  
    .selectAll('circle')           • 전체 점 선택  
    .style('opacity', 0.3)         • 우선 모든 점을 반투명하게 함  
    .filter(function(d){  
      return widthRange[0] <= d.width && d.width <= widthRange[1] &&  
        lengthRange[0] <= d.length && d.length <= lengthRange[1];  
    })                               • 현재 브러시에 속하는 점들만 선택  
    .style('opacity', 1)           • 선택된 점들은 불투명하게 함  
    .each(function(d){            • 선택한 점들의 개수 및 길이와 너비의 합을 구함  
      n++;  
      widthSum += d.width;  
      lengthSum += d.length;  
    })  
  // ...  
}
```

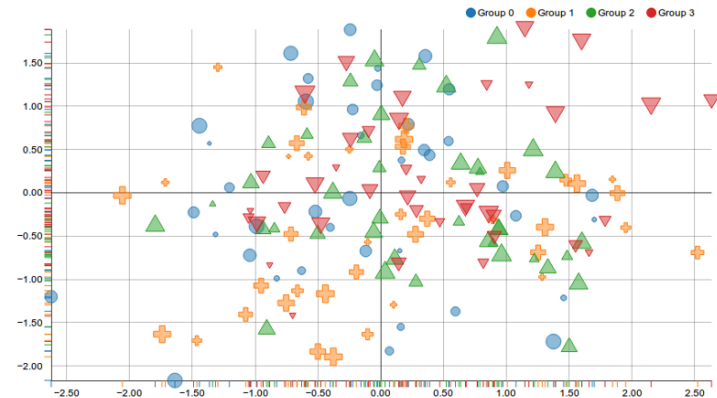
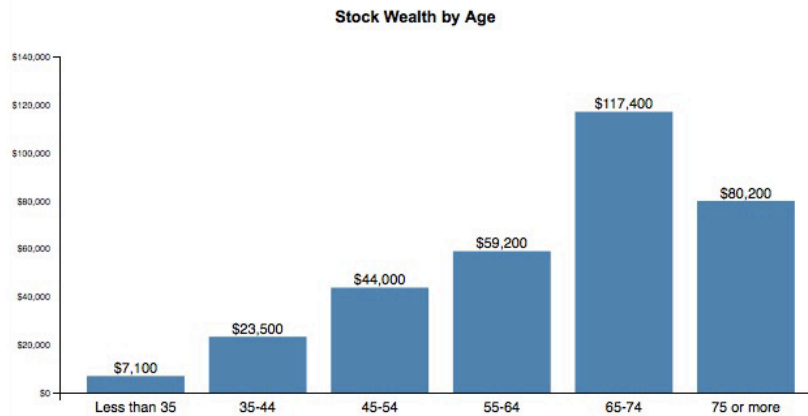
## 산점도 만들기 (21)

```
function update() {  
  // ...  
  
  if(n > 0){  
    d3.select('#mean-width').text(d3.format('.2f')(widthSum / n));  
    d3.select('#mean-length').text(d3.format('.2f')(lengthSum / n));  
  }  
}
```

- 하나 이상의 점이 선택되었다면 너비와 길이의 평균을 소수 둘째 점 자리까지 표시

## 과제 개요

- 본인이 기말 프로젝트에 사용 할 데이터에 대한 간단한 시각화 만들기
  - 데이터를 구하지 못한 수강생들을 위해 예제 데이터 제공
  - 오늘 배운 바 차트와 산점도를 그려보기



## 과제 상세

- 본인의 데이터에서 원하는 차원을 자유롭게 선택
- (필수) 바 차트 하나와 산점도 하나
  - (필수) 올바른 길이 혹은 위치 인코딩을 사용 해야 함
  - (필수) 차트 및 두 축의 제목, 축 상의 눈금 및 레이블
  - (선택) 색 채널 혹은 산점도 상에서 크기 및 모양 채널 사용 가능
- 화면에 한 번에 하나씩 차트를 그리되 버튼을 통해 차트 간 전환이 가능해야 함
  - 차트가 화면에 그려질 때 해당 차트에 대한 설명 및 해석 포함

## 과제 마감

- 마감: 2017년 5월 17일 수요일 23시 55분
- 소스 코드를 .zip 으로 압축하여 eTL을 통해 제출
- 최신 버전의 Chrome에서 채점



## 팁

- 로컬에서 작업 시 보안 정책으로 인해 d3.json 또는 d3.csv 함수로 파일 읽기가 안될 것임
1. 크롬 보안 정책 비활성화
    1. Windows: <http://stackoverflow.com/a/4208455>
    2. Mac: <http://stackoverflow.com/a/2791483>
  2. Firefox 이용
  3. (예제 코드에서 했듯이) 데이터를 코드에 json 혹은 string으로 넣기

## 참고자료

- 다양한 예제 및 소스코드
  - <https://github.com/mbostock/d3/wiki/Gallery>
- API 명세
  - <https://github.com/d3/d3-3.x-api-reference/blob/master/API-Reference.md> (3.x)
- 튜토리얼 및 읽을거리
  - <http://alignedleft.com/tutorials/d3>
  - <https://www.dashingd3js.com/table-of-contents>
  - <http://www.jaeminjo.com/>