



D3.js II

Graphic History and
Concepts of Visualization Technology

D3 라이브러리 다운 및 기본 설정

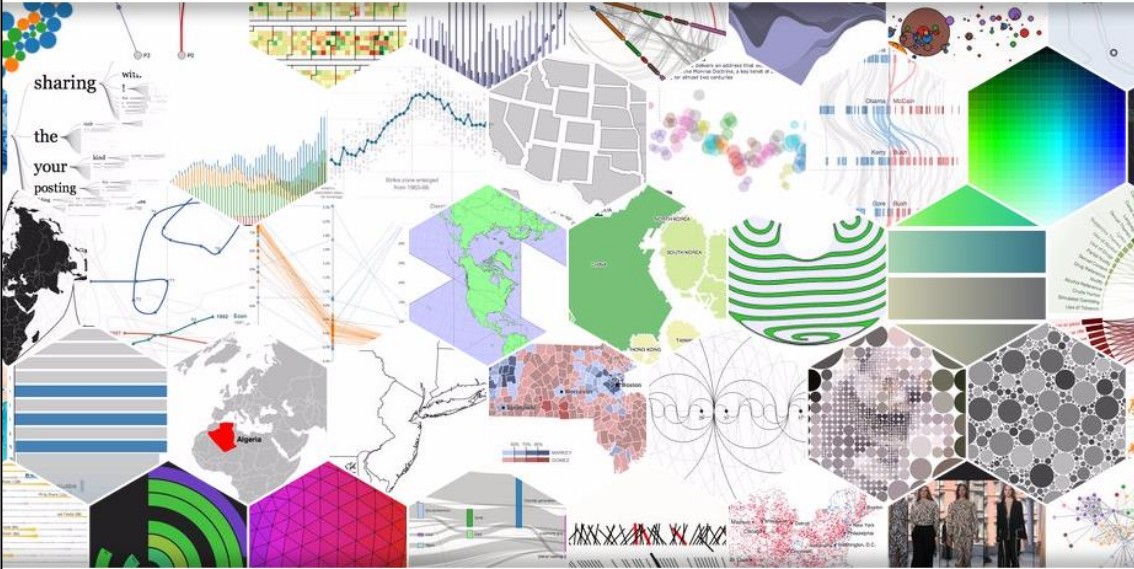


다운로드 방법

- 홈페이지 들어가서 직접 다운
- <https://d3js.org/>

Overview Examples Documentation Source

Data-Driven Documents



D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

Download the latest version (4.2.5) here:

- [d3.zip](#)

To link directly to the latest release, copy this snippet:

```
<script src="https://d3js.org/d3.v4.min.js"></script>
```

[See more examples.](#)

D3 라이브러리 다운 및 기본 설정

- API.md : D3 API 문서
- CHANGES.md : D3 업데이트 기록 문서
- d3.js : d3 시각화 자바스크립트 라이브러리 (개발용 = 디버그용)
- d3.min.js : d3 시각화 자바스크립트 라이브러리 (배포용 = 실제 서비스용)
- LICENSE : 오픈소스 라이브러리에 대한 내용 작성
- README.md : d3 사용에 대한 전반적인 소개 문서

테스트 및 개발 시에는 d3.js로 사용할 것을 권장

이유 : 문제가 생겼을 시 상세 오류 내용을 콘솔 창에 출력하도록 설정되어 있음

실제 서비스 시에는 d3.min.js를 사용할 것으로 권장

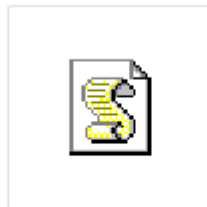
이유 : 자바스크립트 용량을 최소화한 것으로 서비스 시 안정적으로 활용 가능



API.md



CHANGES.md



d3.js



d3.min.js



LICENSE



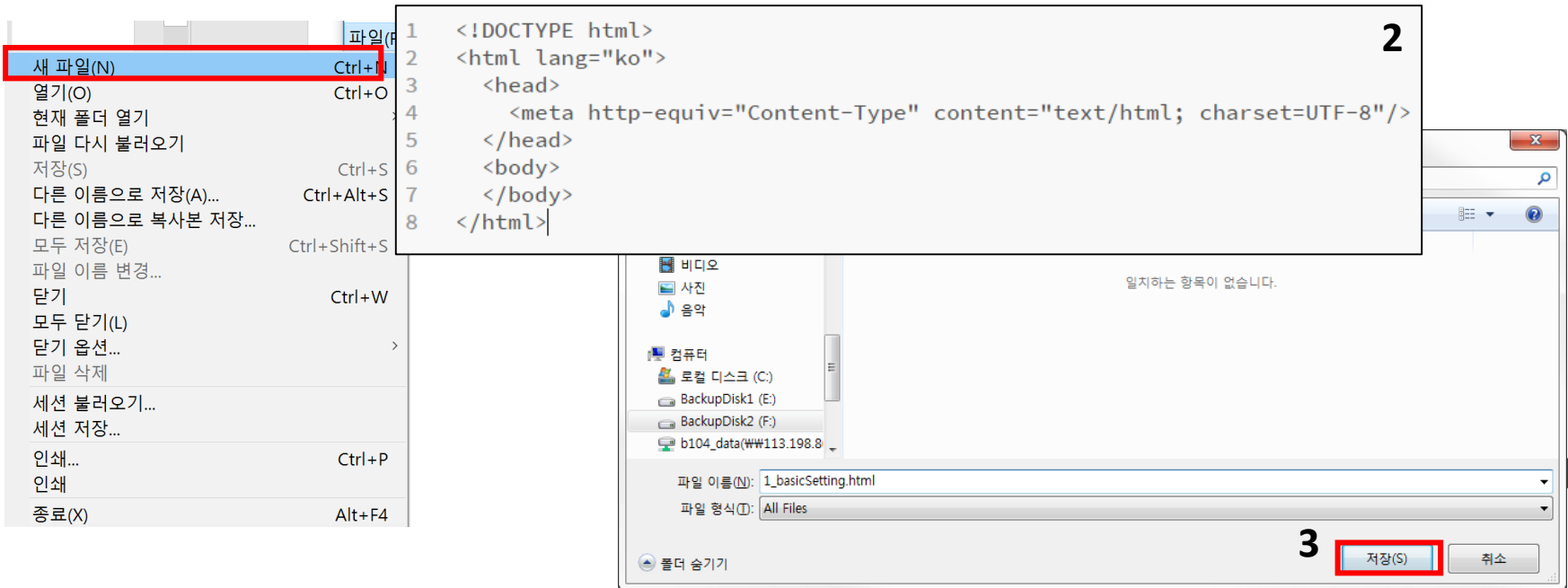
README.md

D3 라이브러리 다운 및 기본 설정

D3 시각화 개발 환경 세팅

1. 새 파일 생성
2. HTML5 기반 템플릿 적용
3. 파일 저장(HTML 형식 파일) 파일이름.html
4. D3 자바스크립트 라이브러리 삽입

1



D3 라이브러리 다운 및 기본 설정



D3 시각화 개발 환경 세팅

1. 새 파일 생성
2. HTML5 기반 템플릿 적용
3. 파일 저장(HTML 형식 파일) 파일이름.html
4. D3 자바스크립트 라이브러리 삽입 (저장 위치에 다운 받은 자바스크립트 파일 복사 후)

```
1  <!DOCTYPE html>
2  <html lang="ko">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
5      <script type="text/javascript" src="d3.js"></script>
6  </head>
7  <body>
8      <script type="text/javascript">
9          // D3를 활용한 코드가 들어갈 위치.
10         </script>
11 </body>
12 </html>
```

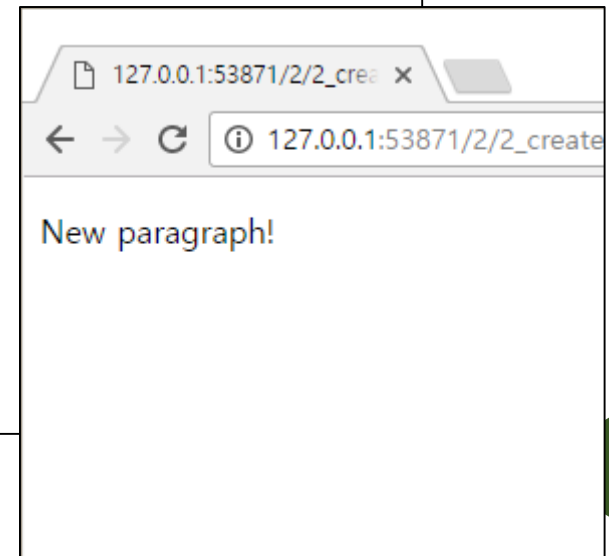


페이지 문서 요소 생성하기

D3를 사용해서 DOM 문서요소를 새로 생성 =
데이터를 표현하기 위한 사각형, 원 같은 시각적 요소

- D3를 이용한 문단을 의미하는 **p 문서 요소** 생성

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <script type="text/javascript" src="d3.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      d3.select("body").append("p").text("New paragraph!");
    </script>
  </body>
</html>
```



페이지 문서 요소 생성하기

- `d3.select("body").append("p").text("New paragraph!");`
> 구두점(.)과 함께 다음 메서드를 계속 연결하는 것을 메서드 체인이라고 함
- 주요 소스 코드 설명

`d3.select("body")`

body 태그 연결
(이 때 body 외 CSS선택자 또는 id를 집어넣을 수 있음)

`.append("p")`

Body를 연결한 상태에서 p 문서요소를 추가

`.text("New paragraph!")`

추가 된 p 요소 안에 문자열을 작성

;

자바스크립트 문단 마지막에
무조건 작성 해당 연결 끝

페이지 문서 요소 생성하기

- 메서드 체인을 사용하지 않을 경우 자바스크립트 변수에 넣어 작성
- 메서드 체인을 사용하지 않은 p 문서 요소 생성 소스코드

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <script type="text/javascript" src="d3.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      var body = d3.select("body");
      var p = body.append("p");
      p.text("New paragraph!");
    </script>
  </body>
</html>
```


D3와 함께 사용되는 데이터

- D3 & 브라우저 기반의 시각화로 범위를 한정하면 데이터를 '텍스트 기반 데이터'로 정의

예) 자바스크립트 변수, 텍스트 파일, CSV 파일, JSON 파일

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
                                     charset=UTF-8"/>

    <script type="text/javascript" src="d3.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      d3.csv("ex.csv", function(data){
        console.log(data);
      });
    </script>
  </body>
</html>
```

CSV 데이터 예

Food, Delicious
Apples, 9
green beans, 5
egg salad, 4
Cookies, 10
Vegemite, 0.2
Burrito, 7

파일명: ex.csv

CSV란?

D3와 함께 사용되는 데이터



JSON 데이터

- csv와 동일한 방식으로 작동
- 하지만 메서드는 같지 않음
- d3.json()

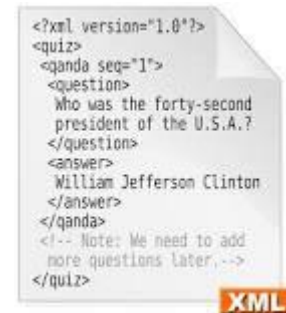


자바스크립트 변수

- 1차원, 2차원 배열로 작성된 데이터
- `var dataset = [5, 10, 15, 20, 25];`
- 일반적으로 많이 사용



JavaScript



XML



XML 데이터

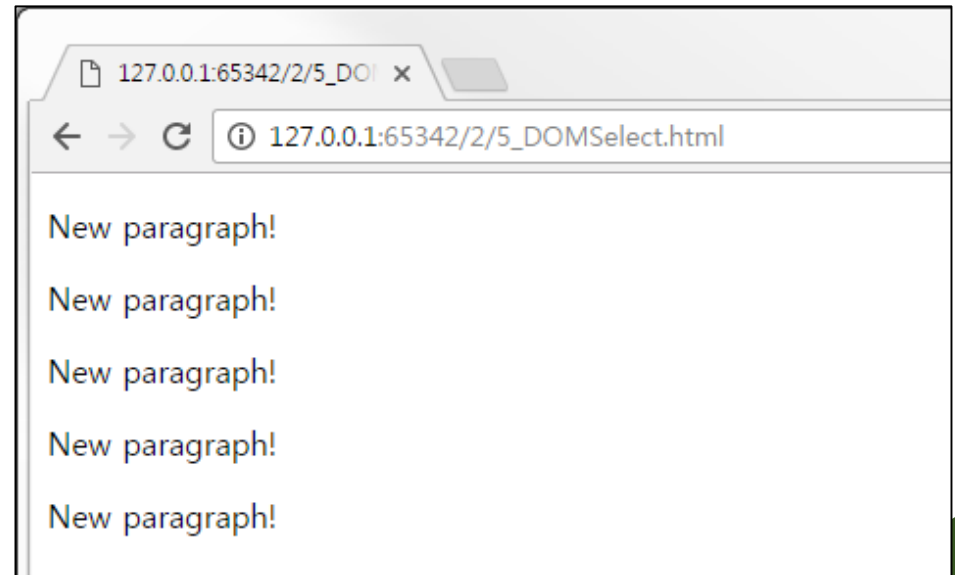
- W3C에서 개발된 다목적 마크업 언어
- 구조적이어서 데이터 교환식으로도 많이 사용

DOM 문서 요소 선택 및 데이터 엮기



자바스크립트 변수(배열) 데이터와 D3 엮기

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <script type="text/javascript" src="d3.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      var dataset = [5, 10, 15, 20, 25];
      d3.select("body").selectAll("p")
        .data(dataset);
        .enter()
        .append("p")
        .text("New paragraph!");
    </script>
  </body>
</html>
```



DOM 문서 요소 선택 및 데이터 엮기(binding)



주요 소스 코드 설명

`d3.select("body")`

선택자로 DOM에서 무엇을 선택하는 데
어떤 문서 요소가 데이터와 관련이 있는 지?

- HTML 문서에서 body를 찾고 메서드 체인을 위해 참조 값 반환

`.selectAll("p")`

- DOM에서 p 문서요소를 모두 선택
- p 요소는 DOM에 존재하지 않기 때문에 빈 선택물 반환

`.data(dataset);`

데이터 값이 몇 개인지 계산하고 파싱
이후 체인은 데이터 개수에 맞춰 반복

`.enter()`

- **DOM 문서 요소** 선택물이 있는 경우 선택 문서 요소 반환
- DOM 문서 요소 < 데이터 개수 일 경우 플레이스 홀더 역할을 할 문서 요소 생성 후 참조 값 반환
- **데이터와 엮인 새로운 문서 요소를 생성하려면 반드시 필요**

DOM 문서 요소 선택 및 데이터 엮기



주요 소스 코드 설명

```
var dataset = [5, 10, 15, 20, 25];
```

```
d3.select("body").selectAll("p")  
  .data(dataset)  
  .enter()  
  .append("p")  
  .text("New paragraph!");
```

→ P 문서요소보다 데이터 개수가 많으므로 빈 플레이스 홀더를 생성하여 다음 메서드에서 이를 활용

Dataset 개수가 5개이므로
총 5번 실행

데이터 사용

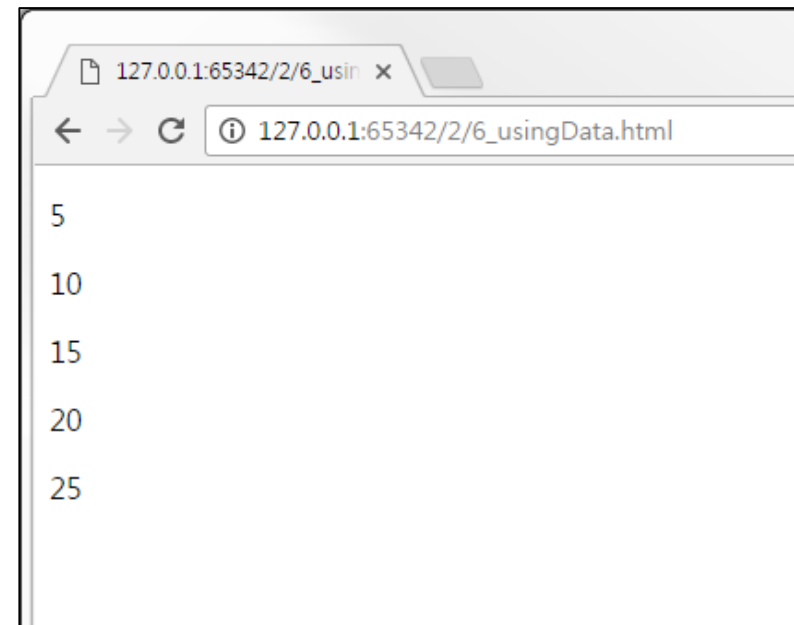


배열 데이터

- var dataset = [5, 10, 15, 20, 25];
- 5, 10, 15, 20, 25 를 실제 사용

```
...  
<body>  
  <script type="text/javascript">  
    var dataset = [5, 10, 15, 20, 25];  
    d3.select("body").selectAll("p")  
      .data(dataset)  
      .enter()  
      .append("p")  
      .text(function (d){  
        return d;  
      });  
  </script>  
</body>  
...
```

text 메서드 익명함수로 변경



D3에서 많이 활용되는 익명 함수



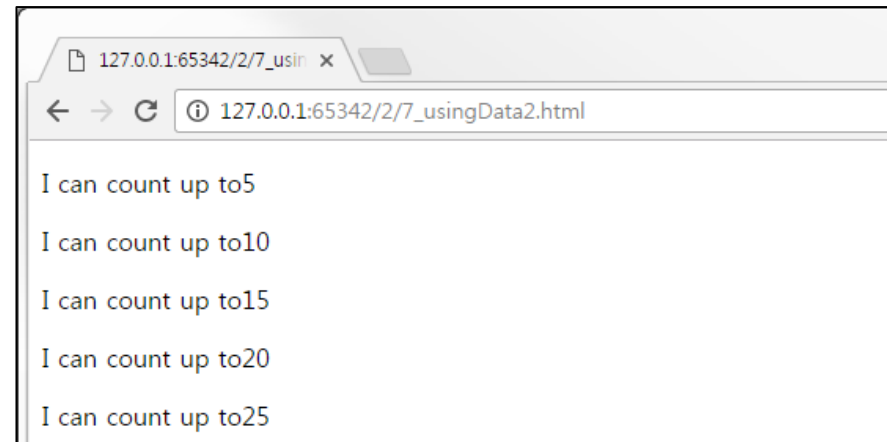
배열 데이터

- D3가 제공하는 메서드가 아닌 자신만의 함수 작성
- 익명함수 기본 구조 (함수 이름이 없음)

```
function (input_value) {  
    //원하는 연산  
    return output_value;  
}
```

- 함수명이 없기 때문에 익명 함수라고 불림
- 응용

```
...  
.text(function (d) {  
    return "I can count up to" + d;  
});  
...
```



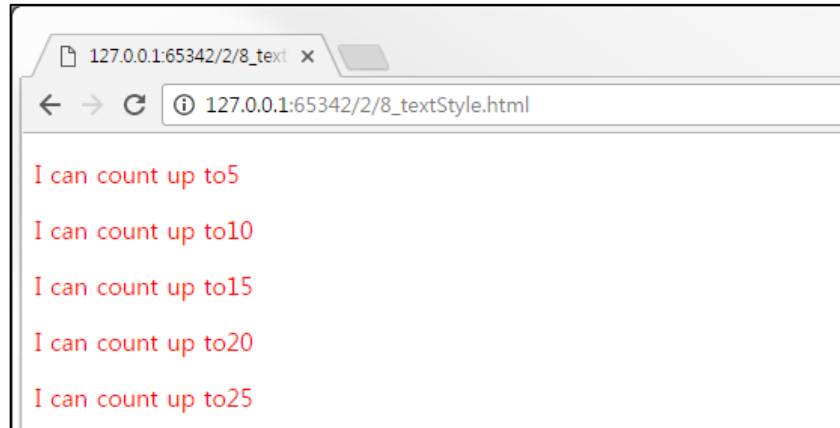
텍스트 스타일 및 속성 적용



배열 데이터

- text() 메서드 뒤 attr(), style()를 활용

```
var dataset = [5, 10, 15, 20, 25];  
d3.select("body").selectAll("p")  
  .data(dataset)  
  .enter()  
  .append("p")  
  .text(function (d){  
    return "I can count up to" + d;  
  })  
  .style("color", "red");
```



- 마찬가지로 익명 함수 활용 자유롭게 스타일 변경 가능...



```
.style("color", function(d){  
  if (d > 15) {  
    return "red";  
  } else {  
    return "balck";  
  }  
});
```

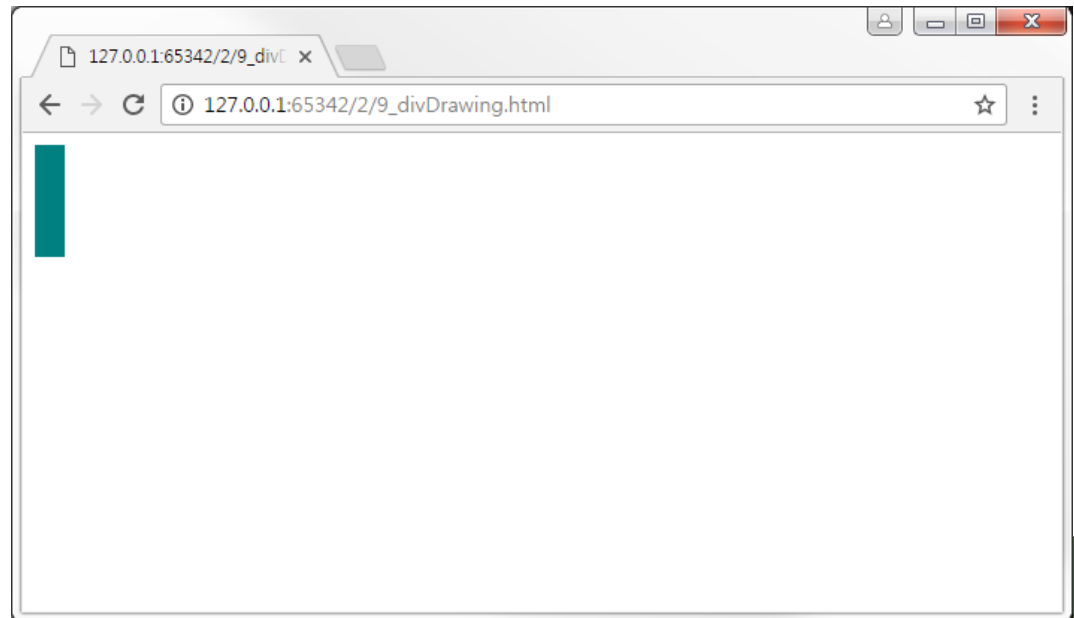

div 드로잉



HTML에서 사각형을 그리는 가장 쉬운 방법

- div를 추가 후 CSS 스타일 추가

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <style>
      div.bar{
        display: inline-block;
        width: 20px;
        height: 75px;
        background-color: teal;
      }
    </style>
  </head>
  <body>
    <div class="bar"></div>
  </body>
</html>
```



div 드로잉

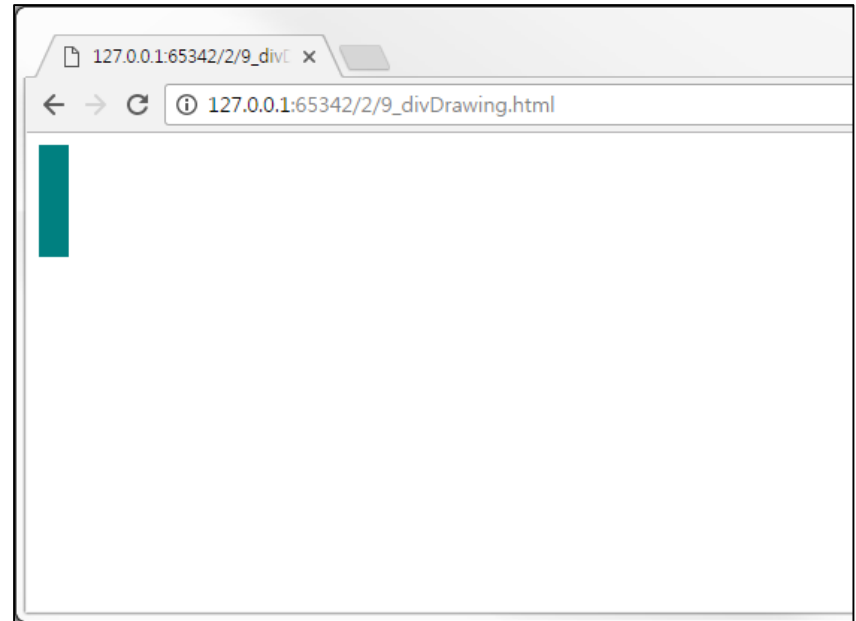


D3 활용 사각형 그리는 법

- attr 매서드로 CSS 클래스 추가

...

```
<style>
  .bar{
    display: inline-block;
    width: 20px;
    height: 75px;
    background-color: teal;
  }
</style>
</head>
<body>
  <div></div>
  <script>
    d3.select("body").selectAll("div").attr("class","bar");
  </script>
</body>
</html>
```



div 드로잉



D3 활용 사각형 그리는 법

- attr() 외 classed() 메서드 사용 가능

...

```
<style>
  .bar{
    display: inline-block;
    width: 20px;
    height: 75px;
    background-color: teal;
  }
</style>
</head>
<body>
  <div> </div>
  <script>
    d3.select("body").selectAll("div").classed("bar", true);
  </script>
</body>
</html>
```

**false 일 경우
bar 클래스 제거**

div 드로잉

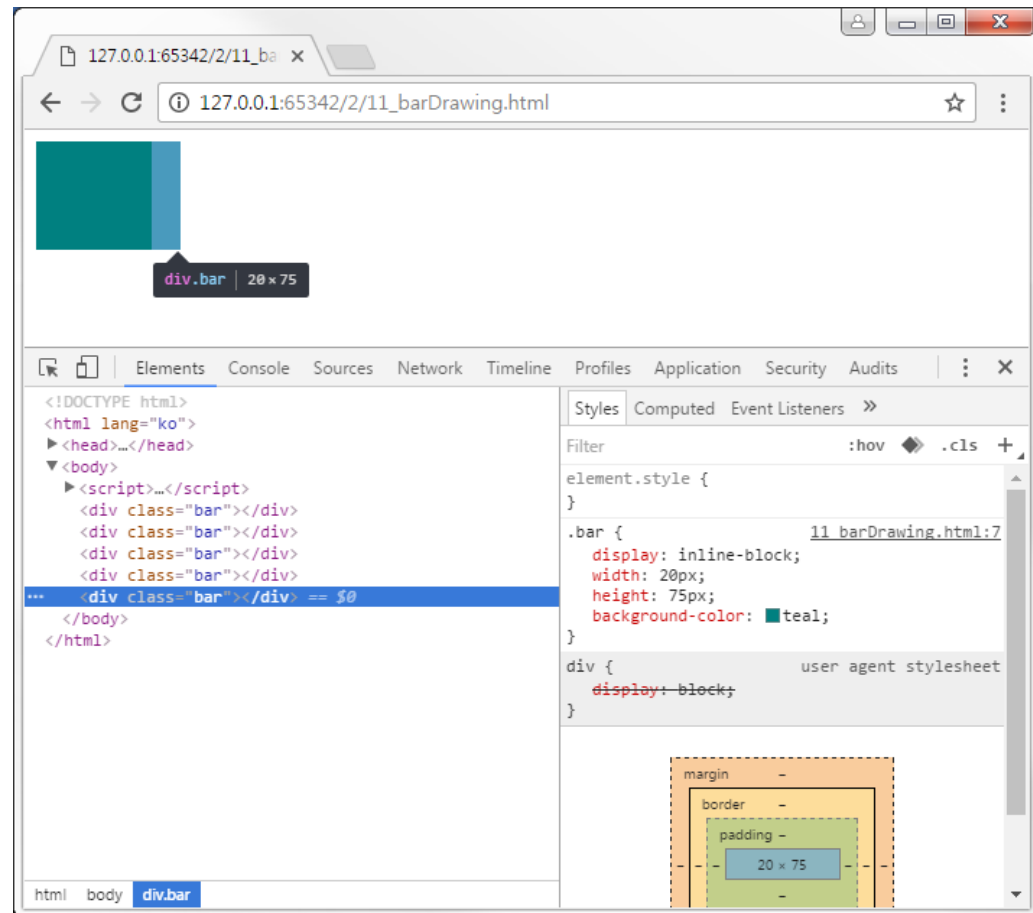


D3 활용 막대 차트

- 막대 차트 만들기 1 단계 - 데이터 삽입 `var dataset = [5, 10, 15, 25]`

...

```
<body>
  <div></div>
  <script>
    var dataset=[ 5, 10, 15, 20, 25];
    d3.select("body").selectAll("div")
      .data(dataset)
      .enter()
      .append("div")
      .attr("class", "bar");
  </script>
</body>
</html>
```



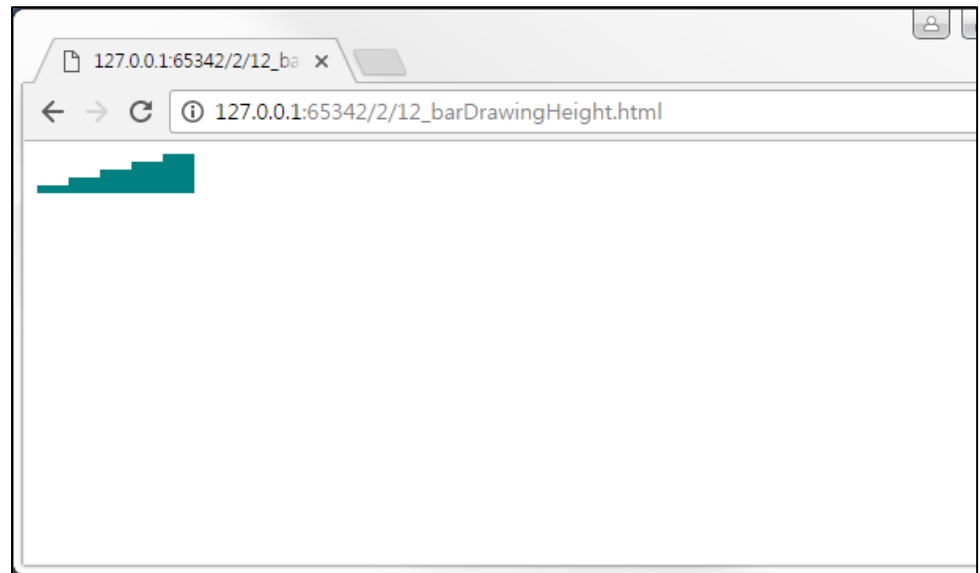
div 드로잉



D3 활용 막대 차트

- 막대 차트 만들기 2 단계 - 높이 지정 (스타일 설정)

```
...  
<body>  
  <script>  
    var dataset=[ 5, 10, 15, 20, 25];  
    d3.select("body").selectAll("div")  
      .data(dataset)  
      .enter()  
      .append("div")  
      .attr("class", "bar")  
      .style("height", function(d) {  
        return d + 'px';  
      });  
  </script>  
</body>  
</html>
```



스타일 높이(height) 값을 지정할 때
실제 데이터 값 + 'px' 이 들어가면서
5px, 10px, 15px, 20px, 25px 이 됨

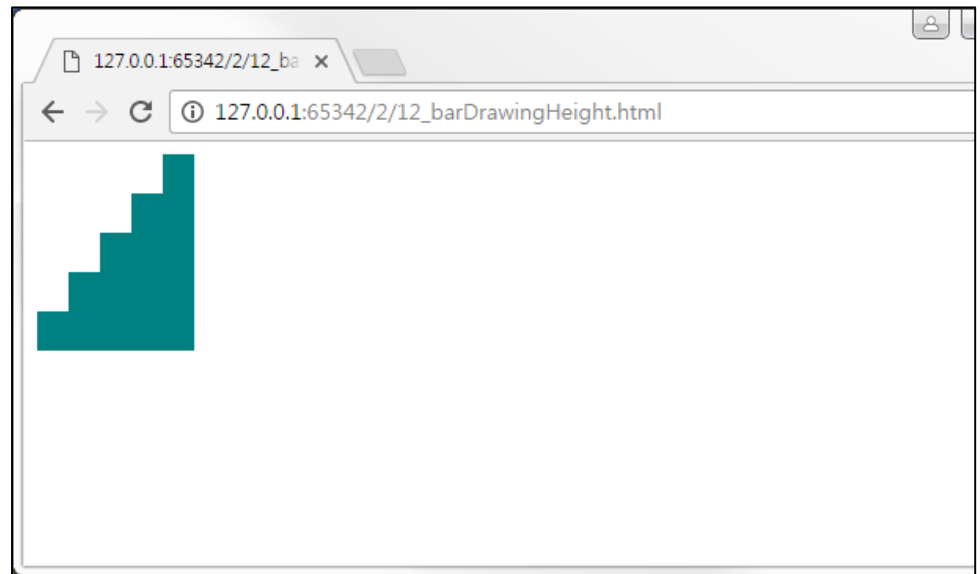
div 드로잉



D3 활용 막대 차트

- 막대 차트 만들기 2 단계 - 높이 지정 (스타일 설정)

```
...  
<body>  
  <script>  
    var dataset=[ 5, 10, 15, 20, 25];  
    d3.select("body").selectAll("div")  
      .data(dataset)  
      .enter()  
      .append("div")  
      .attr("class", "bar")  
      .style("height", function(d) {  
        return (d * 5) + 'px';  
      });  
  </script>  
</body>  
</html>
```



너무 낮아 높이 값을 높여 적절하게 시각화

div 드로잉

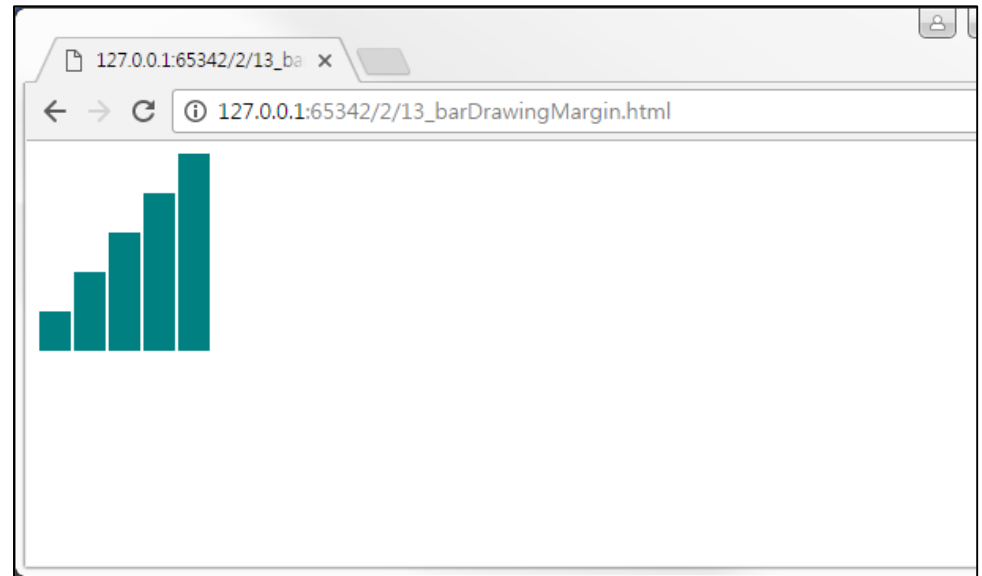


D3 활용 막대 차트

- 막대 차트 만들기 3 단계 - 스타일 지정으로 막대 사이 간격 조정

...

```
<body>
  <script>
    var dataset=[ 5, 10, 15, 20, 25];
    d3.select("body").selectAll("div")
      .data(dataset)
      .enter()
      .append("div")
      .attr("class", "bar")
      .style("height", function(d) {
        return (d*5) + 'px';
      })
      .style("margin-right", "2px");
  </script>
</body>
</html>
```



div 드로잉

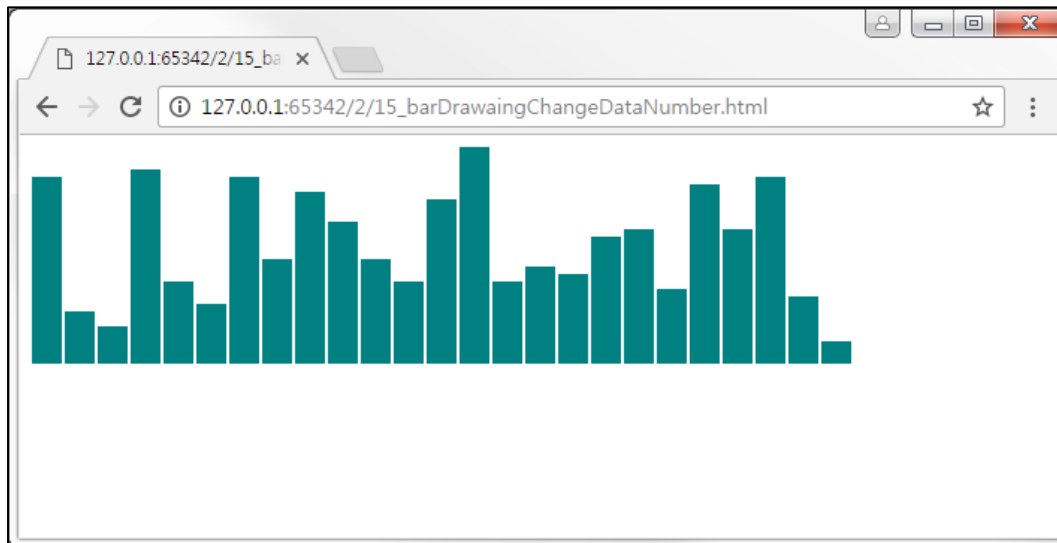
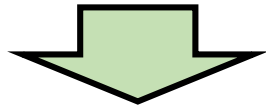
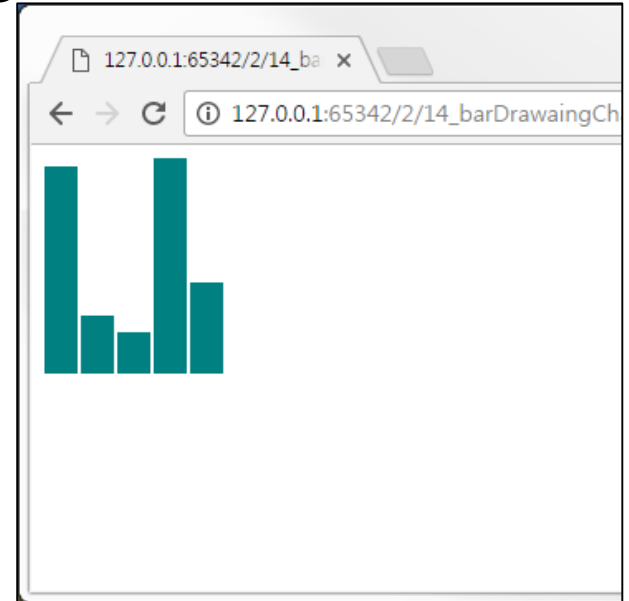
- 막대 차트 만들기 : 데이터 순서 및 개수 변경

데이터 변경: `var dataset=[25, 7, 5, 26, 11];`

데이터 개수 변경:

`var dataset =`

`[25, 7, 5, 26, 11, 8, 25, 14, 23, 19, 14,`
`11, 22, 29, 11, 13, 12, 17, 18, 10, 24,`
`18, 25, 9, 3];`





D3 활용 막대 차트 결론

- data()에 열 개 값을 주면 열 번 순회, 백만 개를 주면 백 만 번 순회
- 데이터 집합이 무엇이든 데이터 개수 만큼 알아서 순회
- 시각화가 데이터를 결정하는 것이 아니라 데이터가 시각화를 결정



응용 시각화 - 임의의 데이터 시각화

- 자바스크립트 사용법만 알면 이를 활용해 임의 데이터를 생성 가능

1. dataset이란 이름의 빈 배열 생성
2. 25번 반복하는 for 구문 생성
3. 반복문 안에서 0과 30 사이의 난수를 새로 생성
4. Dataset 배열에 새로 만든 난수를 추가한다

...

```
<body>
```

```
<script>
```

```
var dataset = [ ];
```

```
for (var i = 0; i < 25; i++) {
```

```
    var newNumber = Math.random() * 30;
```

```
    dataset.push(newNumber);
```

```
}
```

```
d3.select("body").selectAll("div")
```

```
.data(dataset)
```

```
.enter()
```

```
.append("div")
```

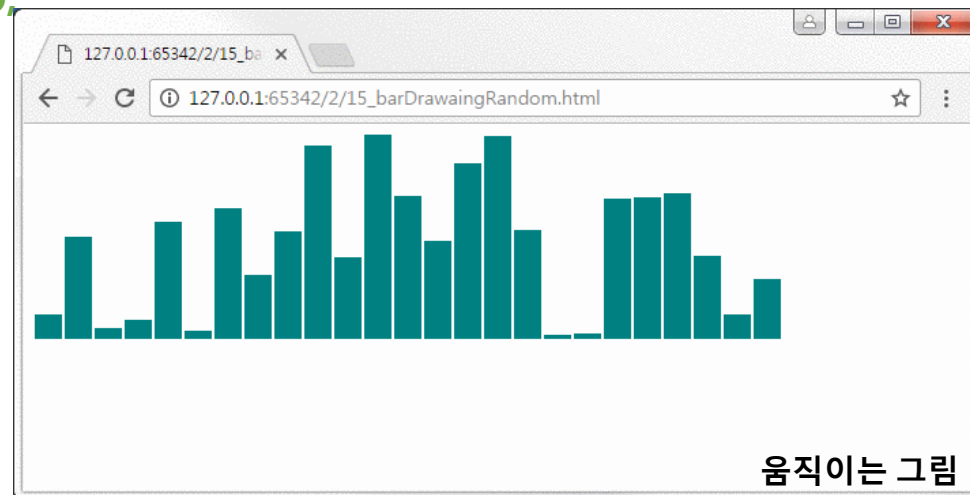
```
.attr("class", "bar")
```

```
.style("height", function(d) {  
    return (d*5) + 'px';
```

```
});
```

```
.style("margin-right", "2px");
```

F5 누를 때 마다 데이터 변경

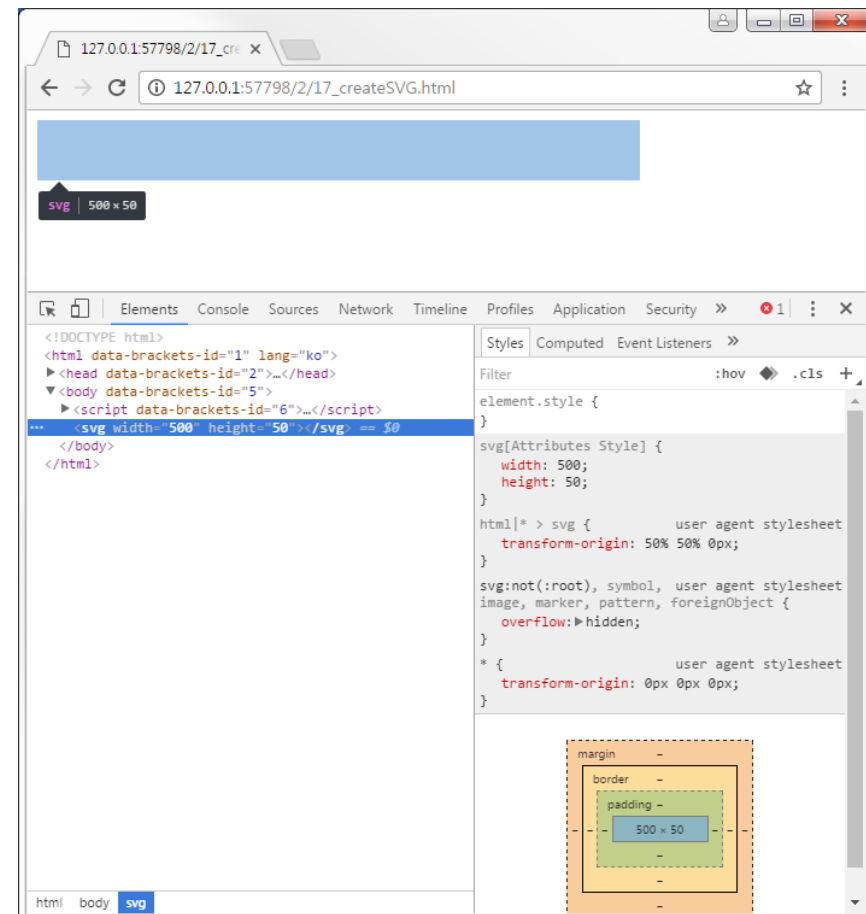


움직이는 그림

SVG 드로잉

- SVG 모든 프로퍼티(Property)는 속성으로 지정
`<element property="value"> </element>`
- D3 사용 SVG 기본 생성

```
...  
<body>  
  <script>  
    var svg =  
    d3.select("body").append("svg");  
    svg.attr("width", 500).attr("height",  
      50);  
  </script>  
...
```

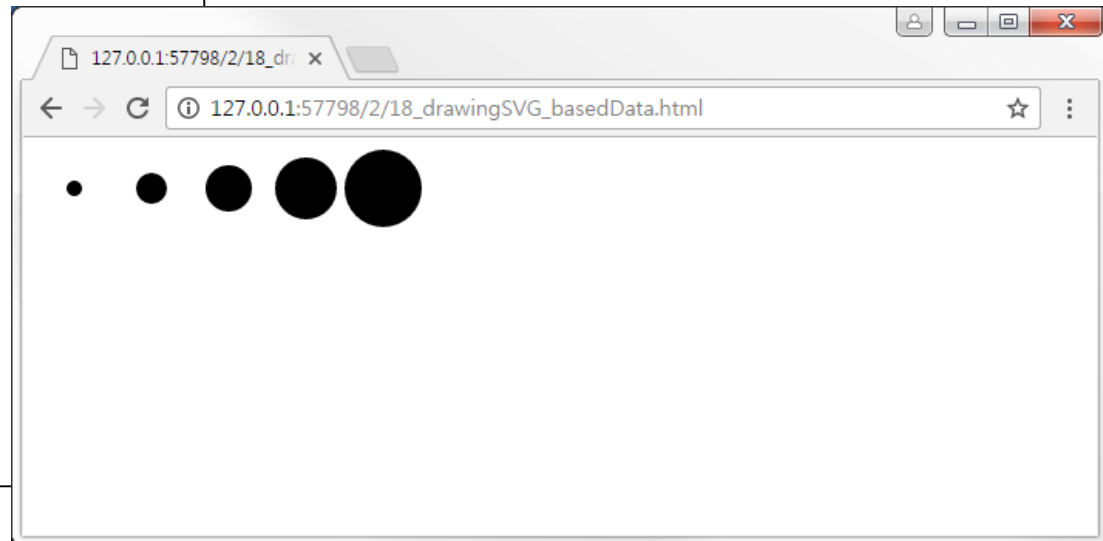


SVG 드로잉

데이터 기반 도형(원)

```
var dataset = [5, 10, 15, 20, 25];
```

```
...  
<body>  
  <script>  
    var dataset = [5, 10, 15, 20, 25];  
    circles.attr("cx", function(d, i) {  
      return (i*50) + 25;  
    })  
    .attr("cy", 50/2)  
    .attr("r", function(d) {  
      return d;  
    });  
  </script>  
</body>  
</html>
```



SVG 드로잉



데이터 기반 도형(원)

```
circles.attr("cx", function(d, i) {  
    return (i*50) + 25;  
})
```

- SVG 원 태그 속성 `cx` : 원의 중심 x좌표
- 파라미터 `d` : 데이터 `([5, 10, 15, 20, 25])`
- 파라미터 `i` : 해당 문서요소 색인 값 (0 부터 시작)

```
.attr("cy", 50/2)
```

- SVG 원 태그 속성 `cy` : 원의 중심 y좌표

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

- SVG 원 태그 속성 `r` : 원 반지름

$$(0 * 50) + 25$$

$$(1 * 50) + 25$$

$$(2 * 50) + 25$$

$$(3 * 50) + 25$$

$$(4 * 50) + 25$$

SVG 드로잉



데이터 기반 도형(원) - 색 입히기

```
var svg = d3.select("body").append("svg");  
svg.attr("width", 500).attr("height", 50);
```

```
var dataset = [5, 10, 15, 20, 25];  
var circles = svg.selectAll("circle")  
    .data(dataset)
```

```
    .enter()  
    .append("circle");
```

```
circles.attr("cx", function(d, i) {  
    return (i*50) + 25;  
})
```

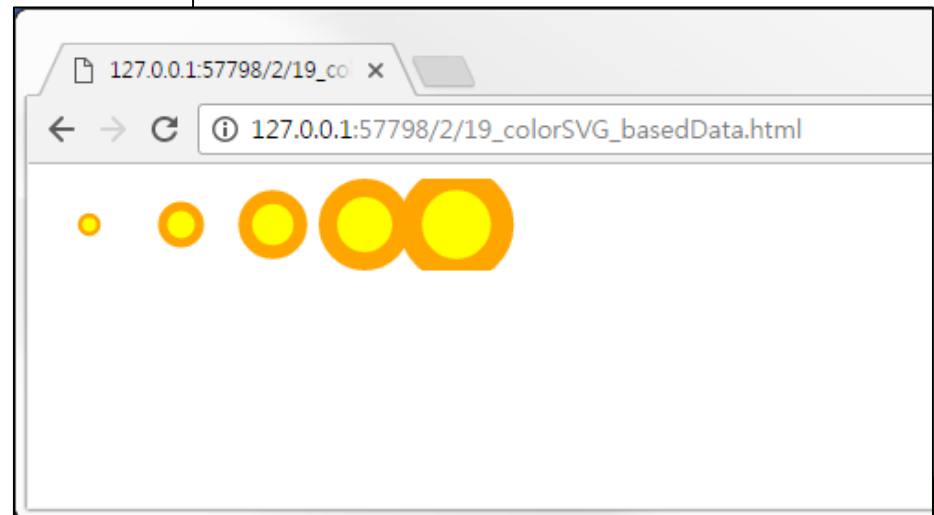
```
.attr("cy", 50/2)
```

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

```
.attr("fill", "yellow")
```

```
.attr("stroke", "orange")
```

```
.attr("stroke-width", function(d){  
    return d/2;  
});
```



SVG 드로잉

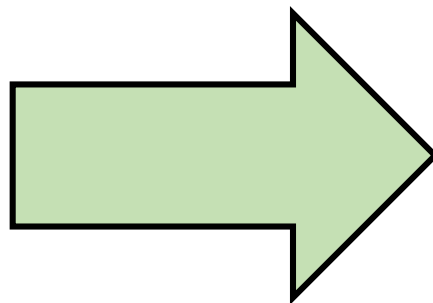


데이터 기반 막대 차트 생성

```
var dataset = [5, 10, 13, 19, 21, 25, 22, 18, 15, 13, 11, 12, 15, 20, 18, 17, 16, 18, 23, 25];
```

div 막대 생성

```
...  
<style>  
  .bar{  
    display: inline-block;  
    width: 20px;  
    height: 75px;  
    background-color: teal;  
    margin-right : 2px;  
  }  
</style>  
...  
<script>  
  d3.select("body").selectAll("div")  
    .data(dataset)  
    .enter()  
    .append("div")  
    .attr("class", "bar");  
</script>  
...
```



SVG?

SVG 드로잉



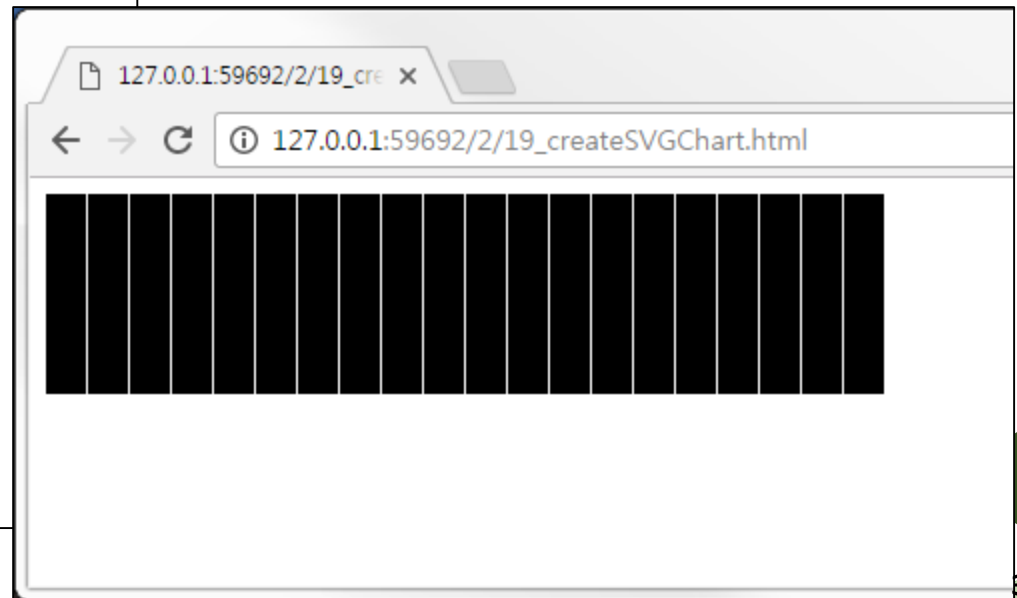
SVG 차트 (1단계)

- 너비, 높이 지정
- 위치: x, y 값

너비 높이 : width, height (임시 값 20, 100)

```
var svg = d3.select("body").append("svg");  
var w = 500;  
var h = 100;  
svg.attr("width", w).attr("height", h);
```

```
svg.selectAll("rect")  
  .data(dataset)  
  .enter()  
  .append("rect")  
  .attr("x", function (d, i){  
    return i * 21;  
  })  
  .attr("y", 0)  
  .attr("width", 20)  
  .attr("height", 100);
```



SVG 드로잉

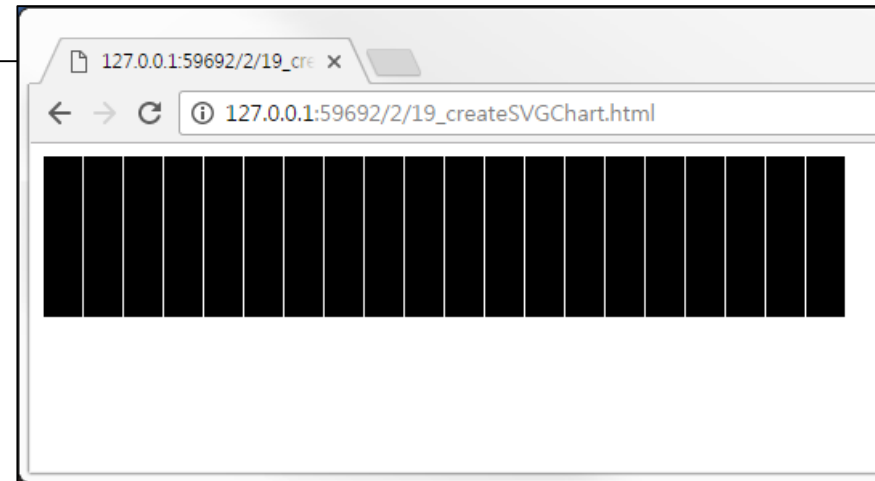


SVG 차트 (2단계)

- x 값 및 width 값을 데이터 개수와 비례하도록 지정
- 데이터 늘어나면 가격 조밀, 데이터 수가 줄면 막대 간격 넓어짐

```
var svg = d3.select("body").append("svg");
var w = 500;
var h = 100;
var barPadding = 1;
svg.attr("width", w).attr("height", h);

svg.selectAll("rect")
  .data(dataset)
  .enter()
  .append("rect")
  .attr("x", function (d, i){
    return i * (w / dataset.length); // i * 21;
  })
  .attr("y", 0)
  .attr("width", w / dataset.length - barPadding) // 21
  .attr("height", 100);
```



SVG 드로잉



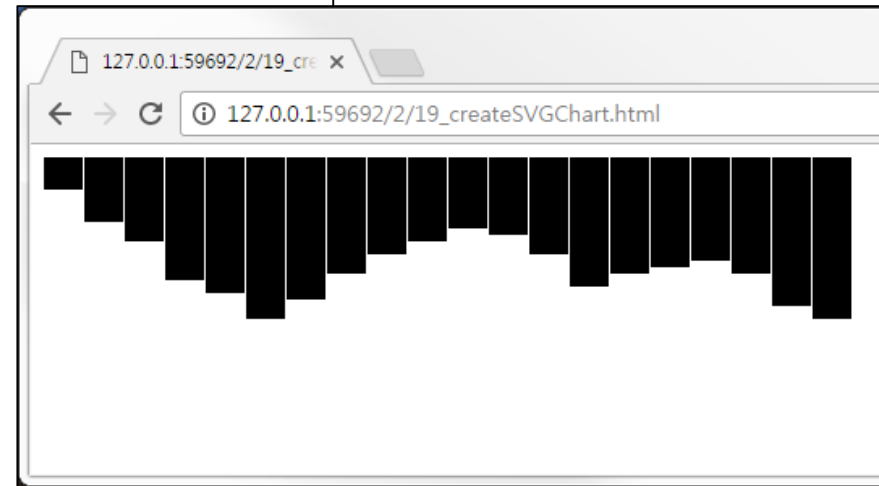
svg 차트 (3단계)

- 막대별 높이 지정

```
var svg = d3.select("body").append("svg");
var w = 500;
var h = 100;
var barPadding = 1;
svg.attr("width", w).attr("height", h);

svg.selectAll("rect")
  .data(dataset)
  .enter()
  .append("rect")
  .attr("x", function (d, i){
    return i * (w / dataset.length); // i * 21;
  })
  .attr("y", 0)
  .attr("width", w / dataset.length - barPadding) // 21
  .attr("height", function (d) {
    return d * 4;
  });
```

svg 좌표 체계로 인한 시각화



SVG 드로잉

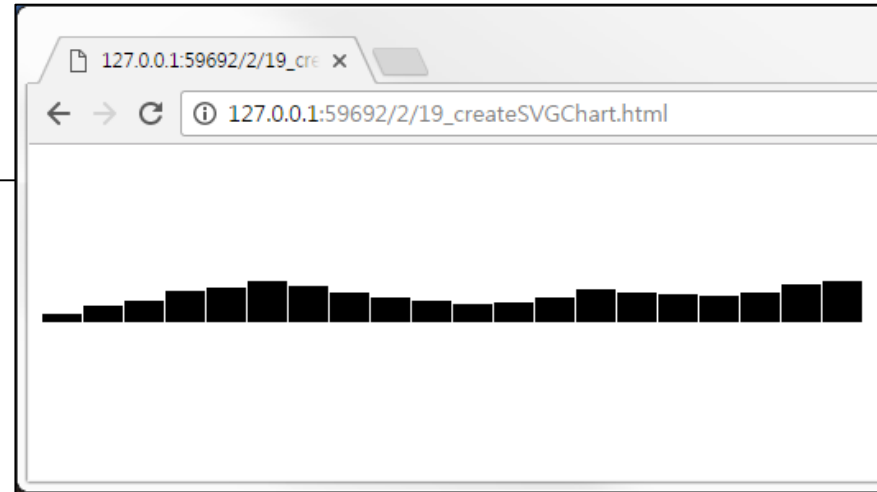


SVG 차트 (4단계)

- 차트 회전

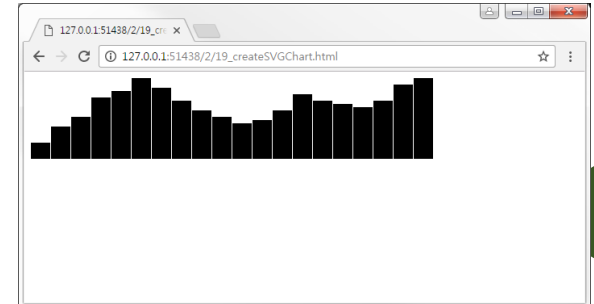
```
var svg = d3.select("body").append("svg");
var w = 500;
var h = 100;
var barPadding = 1;
svg.attr("width", w).attr("height", h);

svg.selectAll("rect")
  .data(dataset)
  .enter()
  .append("rect")
  .attr("x", function (d, i){
    return i * (w / dataset.length); // i * 21;
  })
  .attr("y", function (d) {
    return h - d;
  })
  .attr("width", w / dataset.length - barPadding) // 21
  .attr("height", function (d) {
    return d * 4;
  });
```



회전 시 기존 *4 한 높이 조절이
적용 되지 않음. 아래와 같이 수정

```
.attr("y", function (d) {
  return h - (d * 4);
})
```



SVG 드로잉

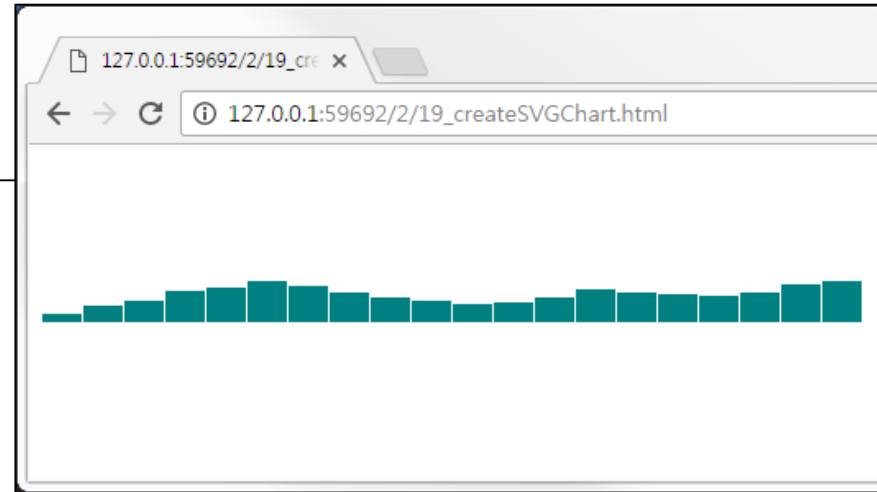


SVG 차트 (5단계)

- 차트 색상

```
var svg = d3.select("body").append("svg");
var w = 500;
var h = 100;
var barPadding = 1;
svg.attr("width", w).attr("height", h);

svg.selectAll("rect")
  .data(dataset)
  .enter()
  .append("rect")
  .attr("x", function (d, i){
    return i * (w / dataset.length); // i * 21;
  })
  .attr("y", function (d) {
    return h - d;
  })
  .attr("width", w / dataset.length - barPadding) // 21
  .attr("height", function (d) {
    return d * 4;
  })
  .attr("fill", "teal");
```



SVG 드로잉



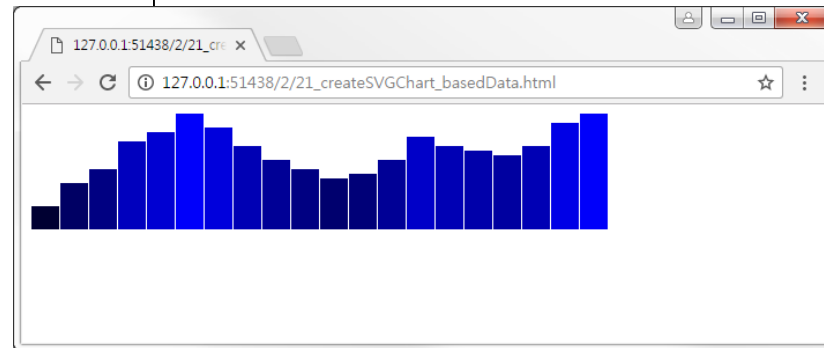
SVG 차트 (응용)

- 차트 색상

```
var dataset = [5, 10, 13, 19, 21, 25, 22,  
18, 15, 13, 11, 12, 15, 20, 18, 17, 16, 18, 23, 25];  
var svg = d3.select("body").append("svg");  
var w = 500;  
var h = 100;  
var barPadding = 1;  
svg.attr("width", w).attr("height", h);
```

```
svg.selectAll("rect")  
  .data(dataset)  
  .enter()  
  .append("rect")  
  .attr("x", function (d, i){  
    return i * (w / dataset.length);  
  })  
  .attr("y", function (d) {  
    return h - (d * 4);  
  })
```

```
  .attr("width", w / dataset.length - barPadding)  
  .attr("height", function (d) {  
    return d * 4;  
  })  
  .attr("fill", function(d) {  
    return "rgb(0, 0, " + (d*10) + ")";  
  });
```



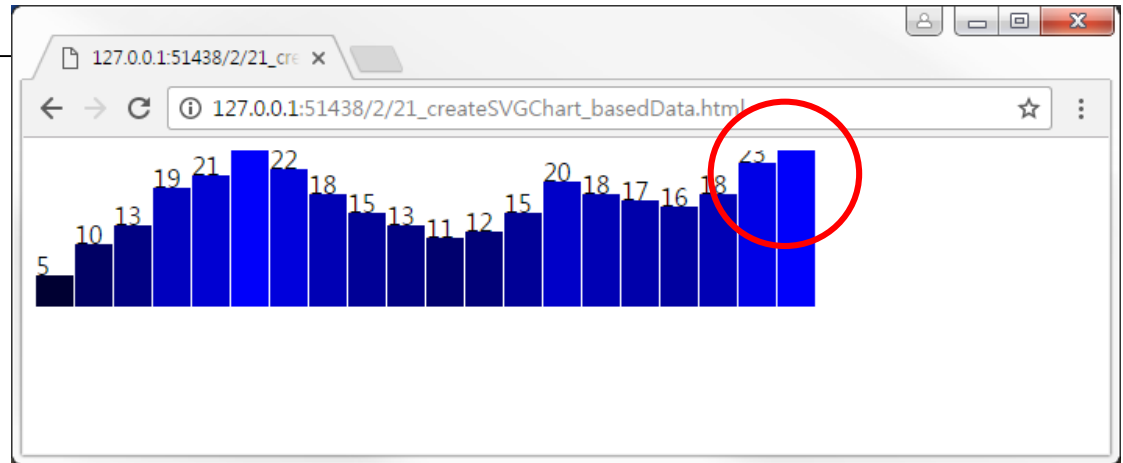
SVG 드로잉



SVG 차트 (SVG 차트 라벨 추가)

- 텍스트 데이터 값을 같이 보아야 할 때가 있을 때 추가
- 기존 막대 그래프 소스코드 아래 추가

```
svg.selectAll("text")
  .data(dataset)
  .enter()
  .append("text")
  .text(function(d){
    return d;
  })
  .attr("x", function (d, i){
    return i * (w / dataset.length)
  })
  .attr("y", function (d){
    return h - (d * 4);
  });
```



※ 문제

막대가 너무 높으면 라벨이 가려짐

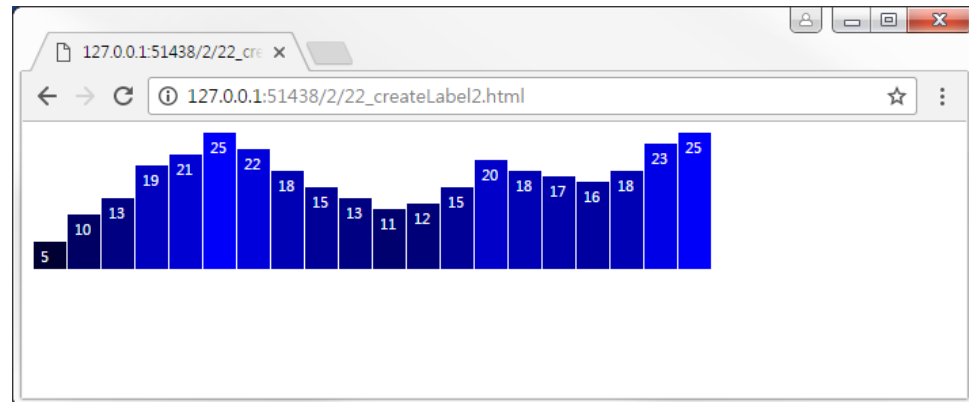
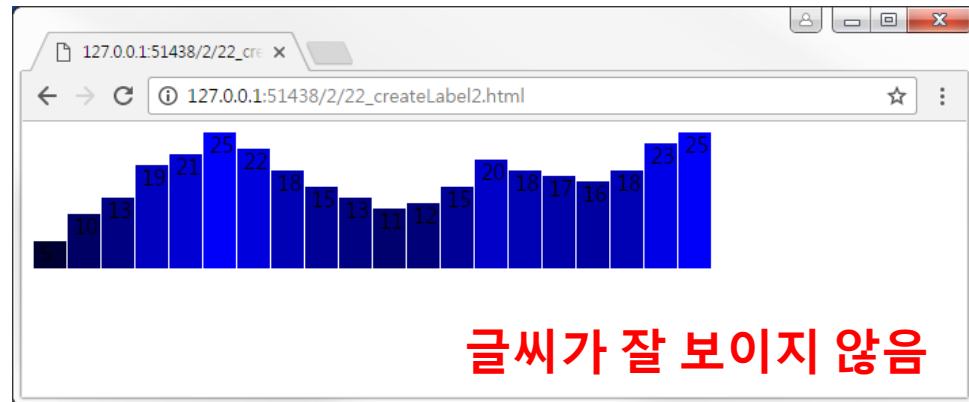
SVG 드로잉



SVG 차트 (SVG 차트 라벨 추가 II)

- x와 y값을 조금 늘려서 막대 안쪽으로 라벨 이동

```
svg.selectAll("text")
  .data(dataset)
  .enter()
  .append("text")
  .text(function(d){
    return d;
  })
  .attr("x", function (d, i){
    return i * (w / dataset.length) + 5;
  })
  .attr("y", function (d){
    return h - (d * 4) + 15;
  })
  .attr("font-family", "sans-serif")
  .attr("font-size", "11px")
  .attr("fill", "white");
```



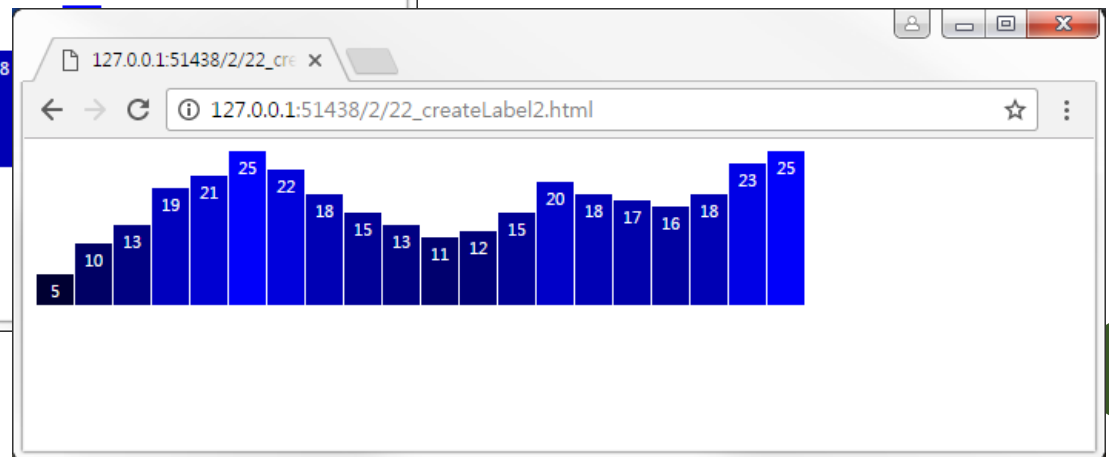
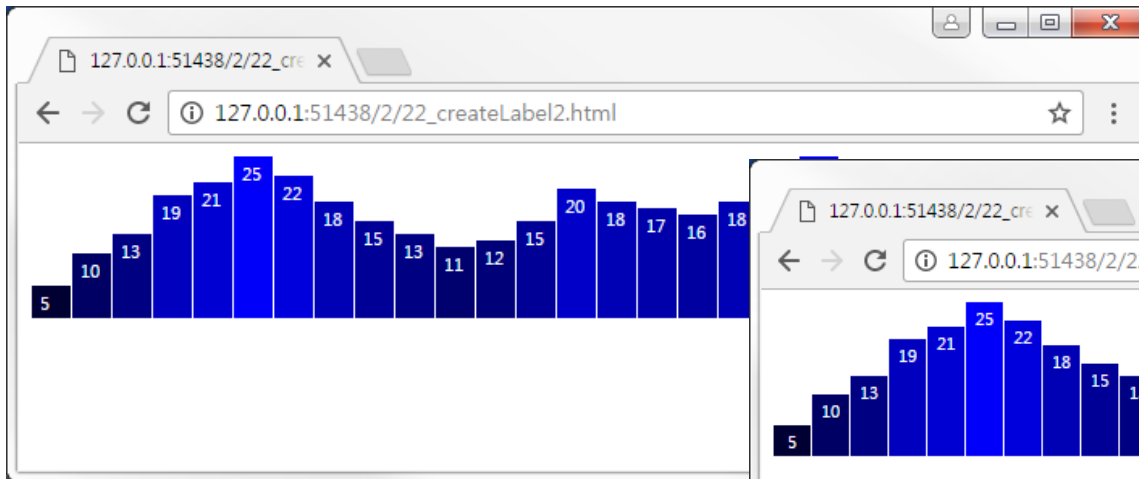
SVG 드로잉



SVG 차트 (SVG 차트 라벨 추가 II)

- 텍스트 막대 정중앙에 정렬되어 있지 않음
(처음 데이터인 5를 보면 확인 가능)

```
.attr("x", function (d, i){  
  return i * (w / dataset.length) + (w / dataset.length - barPadding) / 2;  
})  
.attr("text-anchor", "middle");
```



SVG 드로잉



산포도 만들기

- 막대 차트 그리기 : 1차원 데이터
- 산포도 : 2차원 데이터

var dataset =

```
[5, 10, 13, 19, 21, 25, 22, 18, 15, 13, 11,  
12, 15, 20, 18, 17, 16, 18, 23, 25];
```

var dataset = [

```
[5, 20], [480, 90], [250, 50], [100, 33],  
[330, 95], [410, 12], [475, 44], [25, 67],  
[85, 21], [220, 88]
```

];

SVG 드로잉

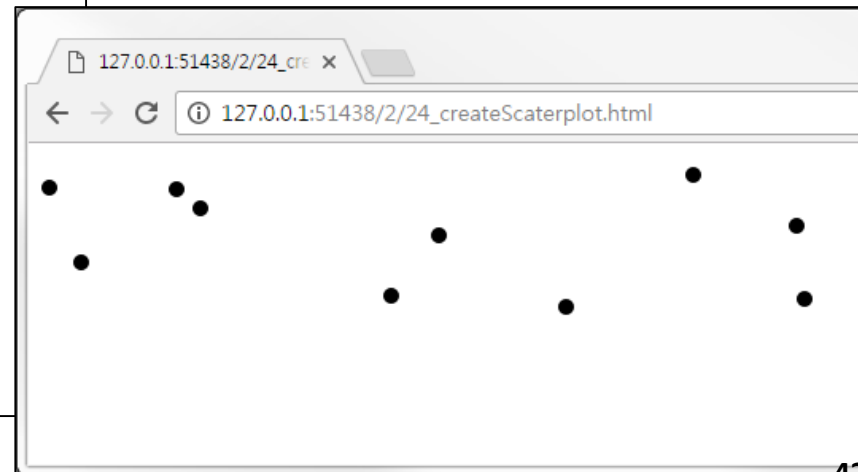


산포도 만들기

- SVG 원 위치를 이동해가면서 산포도 드로잉

```
<script>
  var dataset = [
    [5, 20], [480, 90], [250, 50], [100, 33],
    [330, 95], [410, 12], [475, 44], [25, 67],
    [85, 21], [220, 88]
  ];
  var svg =
d3.select("body").append("svg");
  var w = 500;
  var h = 100;
  svg.attr("width", w).attr("height", h);
  svg.selectAll("circle")
    .data(dataset)
    .enter()
    .append("circle")
```

```
.attr("cx", function(d) {
  return d[0]
})
.attr("cy", function(d) {
  return d[1]
})
.attr("r", 5);
</script>
```



SVG 드로잉



산포도 만들기 (소스코드 해석)

```
var dataset = [  
  [5, 20], [480, 90], [250, 50], [100, 33],  
  [330, 95], [410, 12], [475, 44], [25, 67],  
  [85, 21], [220, 88]  
];  
var svg = d3.select("body").append("svg");  
var w = 500;  
var h = 100;  
svg.attr("width", w).attr("height", h);  
svg.selectAll("circle")  
  .data(dataset)  
  .enter()  
  .append("circle")
```

var dataset = [

[5, 20],
[480, 90],
[250, 50],
[100, 33],
[330, 95],
[410, 12],
[475, 44],
[25, 67],
[85, 21],
[220, 88]

];

SVG 드로잉



산포도 만들기 (소스코드 해석)

...

```
var svg = d3.select("body").append("svg");
var w = 500;
var h = 100;
svg.attr("width", w).attr("height", h);
svg.selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle")
  .attr("cx", function(d) {
    return d[0]
  })
  .attr("cy", function(d) {
    return d[1]
  })
```

```
var dataset = [
  [5, 20],
  ...
];
      d[0]      d[1]
```

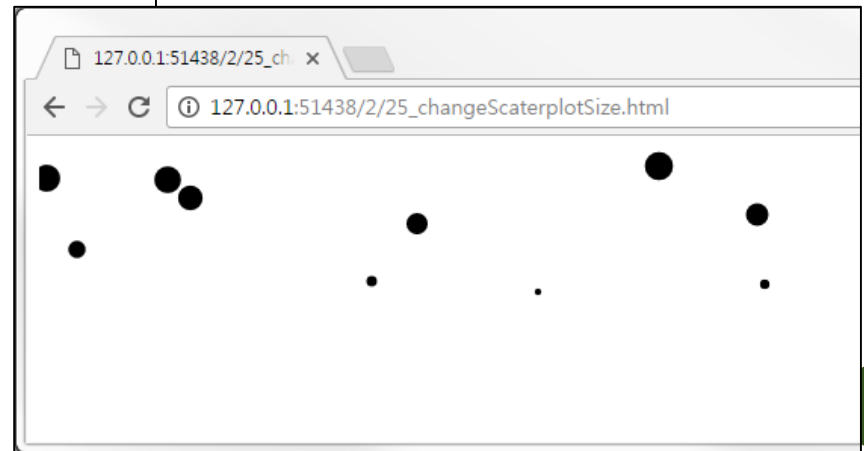
SVG 드로잉

산포도 크기 조절 (각 배열 인덱스 1 데이터에 맞게)

- 각 원의 넓이($A=\pi r^2$)와 y 값을 연결 (반지름 아님)

```
...  
d3.select("body").append("svg");  
  var w = 500;  
  var h = 100;  
  svg.attr("width", w).attr("height", h);  
  svg.selectAll("circle")  
    .data(dataset)  
    .enter()  
    .append("circle")  
    .attr("cx", function(d) {  
      return d[0];  
    })  
    .attr("cy", function(d) {  
      return d[1];  
    })
```

```
.attr("r", function(d){  
  return Math.sqrt(h -  
    d[1]);  
});  
</script>
```



SVG 드로잉



산포도 라벨

- 막대 차트 코드와 동일하게 각 원 옆에 라벨 삽입 가능

```
svg.selectAll("text")
  .data(dataset)
  .enter()
  .append("text")
  .text(function(d){
    return d[0] + "," + d[1];
  })
  .attr("x", function(d){
    return d[0];
  })
  .attr("y", function(d){
    return d[1];
  })
  .attr("font-family", "sans-serif")
  .attr("font-size", "11px")
  .attr("fill", "red");
```

