오픈소스 자바스크립트 라이브러리인 leaflet map(leaflet 라이브러리)을 활용하면 간단한 코드로 네이버지도나, 구글 지도와 같이 식상해진 지도 레이어가 아닌, 독특한 컨셉을 갖고 있는 레이어 테마를 활용하거나 직접 템플릿을 만들어 적용해볼 수 있다.

[leaflet 기본설정하기]

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.5.1/dist/leaflet.css"
   integrity="..."  crossorigin=""/>
<script src="https://unpkg.com/leaflet@1.5.1/dist/leaflet.js"
   integrity="..."  crossorigin=""></script>

<div id="mapid" style="width: 600px; height: 400px;"></div>
```

1. map 객체 생성하기
2. 타일 레이어 추가하기
3. 마커 추가하기(팝업 윈도우나 툴팁 추가하기)

1. map 객체 생성하기

```
var my map = L.map('mapid').setView([37.566, 126.978], 18);
```

map 객체를 생성하는 코드이다. setView()메서드는 지도의 기준점을 잡고, 초기 줌레벨을 어느 정도로 설정할지를 결정하는 부분이다.

2. 타일 레이어 추가하기

```
L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token=pk.eyJ1IjoibWFwYm94IiwiYSI6ImNpejY4NXVycTA2emYycXBndHRqcmZ3N3gifQ.rJcFIG214AriISLbB6B5aw', {
        maxZoom: 18,
        attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, ' +
                '<a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, ' +
                'Imagery <a href="https://www.mapbox.com/">Mapbox</a>',
                id: 'mapbox.streets'
        }).addTo(mymap);
```

leaflet map의 타일 레이어 형식은 URL템플릿의 세팅을 포함하고 있다.
생성한 'mymap'에 타일 레이어를 추가해주는 부분이다. 가장 첫 줄에 삽입되어 있는 부분이 '타일

레이어 URL 템플릿 값'을 불러오는 부분이다. 나머지 부분은 하단에 함께 표시할 '출처'부분이다. 예제 코들을 실행해보면, 우측하단에 RL템플릿을 제작한 사람, 그리고 OSM(OpenStreetMap)과 CC 정보 등의 정보가 표시되는 것을 볼 수 있다.

3. 마커 추가하기

```
L.marker([37.566, 126.978]).addTo(mymap).bindPopup(
"<b>안녕하세요! 난 팝업이야..</b><br />여기가 서울시청입니다...").openPopup();
```

leaflet with GeoJSON : leaflet map 레이어위에 다른 데이터(GeoJSON 형식)를 얹는다.

leaflet map에 GeoJSON을 활용하여 매핑하는 과정에 대해서 살펴본다. GeoJSON은 d3에서도 많이 활용되는 지오데이터 형식이다. 이 형식은 매우 유연해서, leaflet 맵을 처리하는데 있어서도 유연하게 작동한다.

GeoJSON이란? - https://geojson.org/

GeoJSON은 위치정보를 갖는 점을 기반으로 하여 체계적으로 다양한 지리적 데이터 구조를 표현하기 위해 설계된 개방형 공개 표준 형식이다. 이것은 JSON인 자바스크립트 오브젝트 노테이션을 사용하는 파일 포맷이다. GeoJSON객체는 Point, Line, Polygon MultiLineString, multiPolygon 등을 표현한다. FeatureCollection은 features의 리스트를 뜻하며 Features는 기하학적 객체와 해당 객체에 대한 속성값을 포함한다.


```
{
"type": "FeatureCollection",
"name": "HangJeongDong_ver20190403",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
{ "type": "Feature", "properties": { "OBJECTID": 13, "adm_nm": "서울특별시 종로구 창신3동", "adm_cd":
"1101069", "adm_cd2": "1111069000" }, "geometry": { "type": "MultiPolygon", "coordinates":
[ [ [ [ 127.014950496477127, 37.582130731356358 ], [ 127.014950961590728, 37.581411919002839 ],
[ 127.015477694466341, 37.57681045639643 ], [ 127.012649293720969, 37.575381346683614 ],
[ 127.011731809193805, 37.576182106359312 ], [ 127.011109504623903, 37.578910610965657 ],
[ 127.010480065311697, 37.58033548115759 ], [ 127.012155024459105, 37.58150517845035 ],
[ 127.012801219237048, 37.581317479457063 ], [ 127.013122386420051, 37.58143931902778 ],
[ 127.013827792100869, 37.581708123465212 ], [ 127.014685884830357, 37.582226547406286 ],
[ 127.014950496477127, 37.582130731356358 ] ] ] ] } }
]
}
```

map에 GeoJSON추가하기(1)

```
$.getJSON("sido.geojson",function(data){
	L.geoJson(data).addTo(mymap).bindPopup(function (layer) {
		return layer.feature.properties.CTP_KOR_NM;
	});
});

$.getJSON("sigungu.geojson",function(data){
	L.geoJson(data).addTo(mymap).bindPopup(function (layer) {
		return layer.feature.properties.SIG_KOR_NM;
	});
});

$.getJSON("20190403.geojson",function(data){
	L.geoJson(data).addTo(mymap).bindPopup(function (layer) {
		return layer.feature.properties.adm_nm;
	});
});

$.getJSON("countries.geojson",function(data){
	L.geoJson(data).addTo(mymap).bindPopup(function (layer) {
		return layer.feature.properties.name;
	});
});
```

map에 GeoJSON추가하기(2)

빈 GeoJSON 레이어를 먼저 생성한 뒤 변수에 할당한 다음 나중에 GeoData 정보를 설정한다.

```
geojsonFeature = "{ .... }";
var myLayer = L.geoJSON().addTo(map); // 빈 geojson 레이어를 'map'위에 추가한다.
myLayer.addData(geojsonFeature); // 생성한 geojson레이어에 geojsonFeature데이터 추가

function myPopup(feature, layer) { layer.bindPopup(feature.properties.XXX); }
L.geoJson(geojsonFeature, { onEachFeature : myPopup}).addTo(mymap);
```

스케일러블 벡터 그래픽스(Scalable Vector Graphics, SVG)는 2차원 벡터 그래픽을 표현하기 위한 XML 기반의 파일 형식으로, 1999년 W3C(World Wide Web Consortium)의 주도하에 개발된 백터 이미지 표준이다. SVG 형식의 이미지와 그 작동은 XML 텍스트 파일로 정의 되어 검색화·목록화·스크립트화가 가능하며 필요하다면 압축도 가능하다.

SVG 형식의 파일은 어도비 일러스트레이터와 같은 벡터 드로잉 프로그램을 사용하여 편집이 가능하며 XML 형식으로 되어 있으므로 메모장과 같은 문서 편집기로도 편집이 가능하다.

[이미지를 디지털화 하는 두 가지 방식]

비트맵 VS 벡터

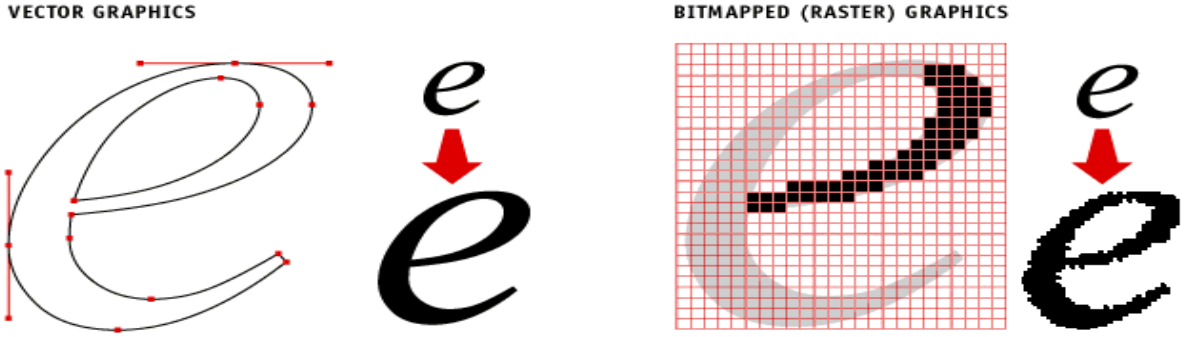
이미지를 디지털화 하는 방법은 크게, 전체 그림을 미세한 화소(pixel)로 분해, 각 화소의 색상과 위치를 기록해 저장한 비트맵 방식과 그림을 구성하고 있는 점이나 직선, 곡선 등의 위치와 기울기 등을 산술적인 데이터로 기록해 저장한 벡터 방식으로 나뉜다.

- 비트맵(bitmap)

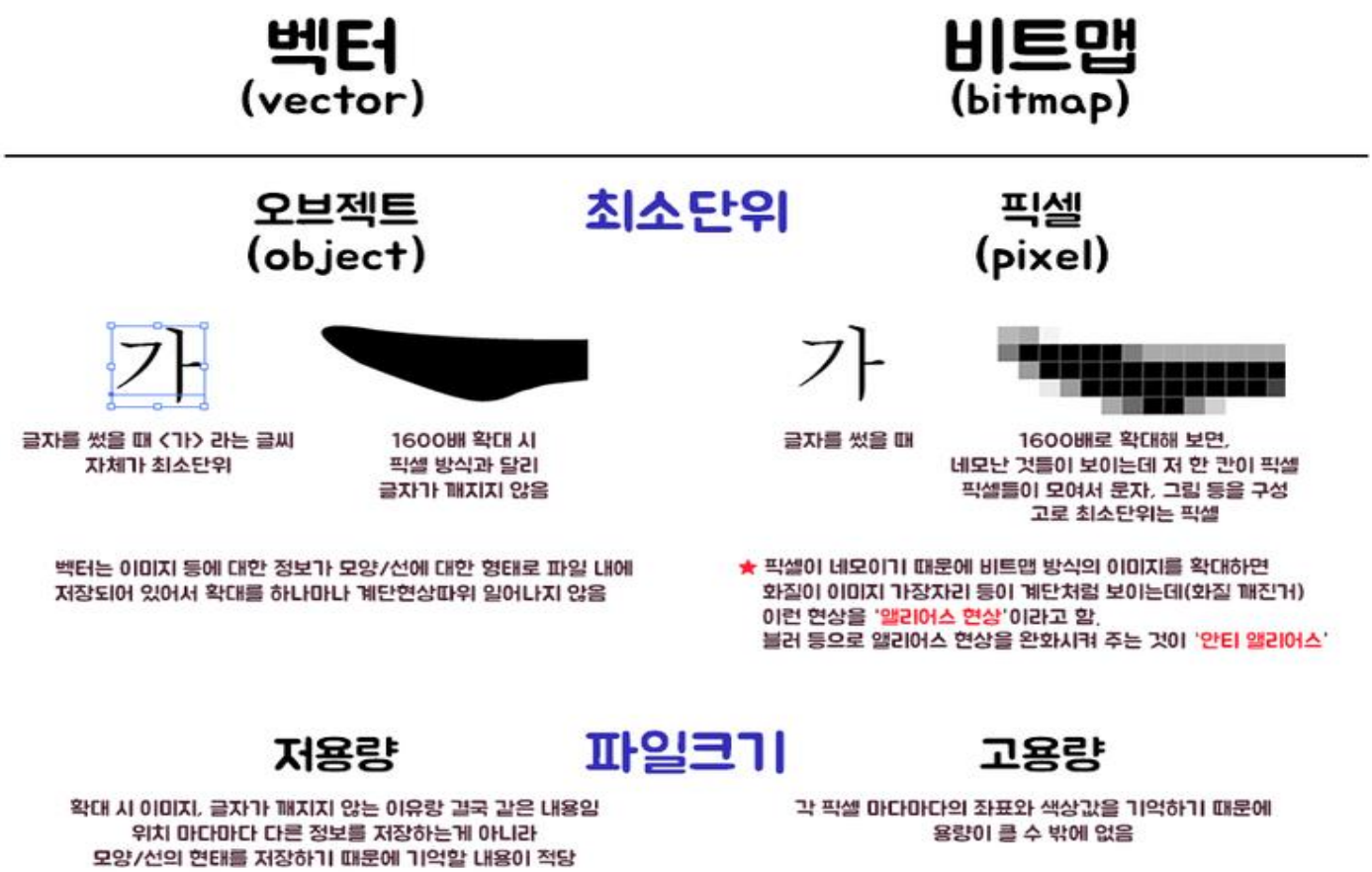
래스터 이미지라고도 불리는 비트맵 이미지는 각 픽셀을 0과 1의 이진법으로 처리해 1로 지정된 픽셀에만 특정 색을 넣어 형태를 구현한다. 일반적으로 컴퓨터에서 사용되는 디지털 이미지는 비트맵 방식이라 할 수 있다. 비트맵 방식은 복잡한 형태의 그림이나 사진도 문제 없이 표현할 수 있는 것이 장점이지만, 이미지를 구성하는 정보량이 많아 데이터의 용량이 크고 이미지를 확대했을 때 화질이 저하되는 것(계단 현상)이 단점이다.

- 벡터(Vector)

벡터 이미지는 특정한 형태를 수학적 함수로 표현해 구현한다. 예를 들어, 일정한 범위를 지정한 후 일차함수로 표현하면 직선이 만들어진다. 벡터 방식은 적은 데이터 용량으로 이미지를 구현할 수 있고, 확대해도 화질(선명도) 저하가 없는 장점이 있지만 복잡한 이미지는 표현할 수 없어 간단한 도형이나 로고 등을 그리는 경우나, 디자인 등의 영상용 이미지로 사용된다.



- 비트맵 : 거의 모든 사진 편집 및 일반적인 그래픽 편집, 픽셀의 조합으로 깊이 있는 색조와 부드러운 질감을 표현할 수 있다. 보정 정보를 포함해야 하므로 상대적으로 용량이 크다. 확대하면 확대에 대한 보정 정보가 없어서 계단 현상이 생긴다.
- 벡터 : 일러스트레이터, 인디자인, 코렐드로우, 플래시 등에 쓰임, 확대해도 원래의 품질과 특성을 유지한다. 형태를 저장하기 때문에 상대적으로 용량이 적다. 다양한 색상 표현에 한계가 있다.



[SVG 시작해보기]

```
<svg width="180" height="200" xmlns="http://www.w3.org/2000/svg">
   // 여기에 SVG 태그를 작성한다.
</svg>
```

<svg> 태그는 svg의 루트 태그이며 svg는 XHTML 스펙을 따르고 있기 때문에 반드시 "xmlns" Attribute를 이용하여 NameSpace을 지정해줘야 한다. <svg> 태그를 HTML 문서 안에 작성하는 경우에는 xmlns 속성을 생략한다. version은 사용할 SVG 스펙의 버전을 말하는데 일반적으로는 1.1

을 사용하면 된다. width와 height는 svg Element의 크기를 지정해주며 생략하면 브라우저의 전체 크기가 적용된다.

기본 도형 그리기

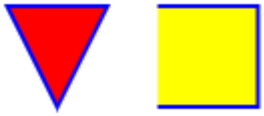
<rect> <circle> <ellipse> <line> <polyline> <polygon>

- 요소 및 속성

도형	요소명	속성
사각형	rect	x="<좌표값>" y="<좌표값>" width="<크기>" height="<크기>"
둥근 사각형	rect	x="<좌표값>" y="<좌표값>" width="<크기>" height="<크기>" rx="<크기>" ry="<크기>"
원	circle	cx="<좌표값>" cy="<좌표값>" r="<크기>"
타원	ellipse	cx="<좌표값>" cy="<좌표값>" rx="<크기>" ry="<크기>"
선	line	x1="<좌표값>" y1="<좌표값>" x2="<좌표값>" y2="<좌표값>"
연결선	polyine	points ="<점들의 리스트>"
다각형	polygon	points ="<점들의 리스트>"

```
<rect x="0" y="0" rx="10" ry="10" width="100" height="100" fill="red" stroke="blue" stroke-width="10"/>

<circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red"></circle>

<ellipse cx="50" cy="50" rx="40" ry="20" stroke="black" stroke-width="3" fill="red"></ellipse>

<line x1="0" y1="10" x2="30" y2="10" stroke="orange" stroke-width="3"></line>

<polygon points="x1,y1 x2,y2, x3,y3 ..." fill="black" stroke="" stroke-width=""></polygon>

<polyline points="x1,y1 x2,y2, x3,y3 ..." fill="black" stroke="" stroke-width=""></polyline>
```

<path> : 복잡한 도형



- **<path d="패스 데이터" pathlength="길이">**
  - 임의의 모양을 묘사 - 연속된 점으로 외각선을 표현 (직선, 곡선, 원호 등)
  - fill 과 stroke 등 스타일 지정이 가능 - 폐쇄형, 개방형
- Path Data
  - 대문자 기호는 절대좌표, 소문자 기호는 상대좌표
  - 좌표값 사이는 공백이나 컴마 등 구분문자 사용

명령	내용	기호	매개변수
moveto	(no drawing)	**M m**	(x y)+
closepath	(끝점-시작점 사이 line)	**Z z**	
lineto	General Line	**L l**	(x y)+
	Horizontal Line	**H h**	x+
	Vertical Lline	**V v**	y+
	cubic Bezier	**C c**	(x1 y1 x2 y2 x y)+

```
<path d="M 100 100 L 300 100 200 300 z" fill="red" stroke="blue" stroke-width="5" />
<path d="M 400 100 h 200 v 200 h -200"  fill="yellow" stroke="blue" stroke-width="5" />
```

Cubic Bezier Curve

양끝점 (시작점), (끝점)과 두개의 제어점 (x1 y1), (x2 y2)으로 구성

예) <path d="M100,200 C100,100 400,100 400,200" />

http://blogs.sitepointstatic.com/examples/tech/svg-curves/cubic-curve.html

[SVG 스타일 지정]

(1) 태그의 속성

```
<rect x="100" y="100" width="600" height="300" fill="red" stroke="blue"
  stroke-width="8"/>
```

(2) CSS

```
rect {
    fill: red; stroke: blue; stroke-width: 8
}
```

[페인팅(Painting)]

페인팅 대상

- 대상 객체 : 도형, 텍스트, 패스
- 페인팅 작업(operation) : fill & stroke
- 페인팅 방법 : single color, gradient (linear or radial), pattern

페인팅 속성의 값 : fill="..." (None : non fill, currentColor : 현재 색상으로 fill,
<color> : 지정한 색상, <URI> : 미리 정의된 fill 스타일)

[좌표시스템의 변환(Transformations)]

2D 기본 변환 : transform="..." (이동(translation) : translate(tx,ty),
회전(rotation) : rotate(a) 시계방향, 신축(scaling) : scale(sx,sy),
비틀림(shearing, skew) : skewX(a) 오른쪽, skewY(a) 아래쪽 축의 비틀림 각도)

[그래디언트(Gradient)]

SVG에서 지원되는 그래디언트

선형 그래디언트 <linearGradient id="..." 속성들 > 그래디언트 내용 </linearGradient>

원형 그래디언트 <radialGradient id="..." 속성들 > 그래디언트 내용 </radialGradient>

<defs> 에서 그래디언트 정의하고, fill 이나 stroke 에서 속성으로 참조(사용)

- linearGradient
그래디언트 내용의 표현 : <stop>, <animate>, <set>, ...
<stop offset="길이" stop-color="색상" stop-opacity="투명도"/>

- radialGradient
<radialGradient id="..." 속성들 >
gradientUnits = "userSpaceOnUse | objectBoundingBox"
cx, cy, r : 원형 그래디언트의 중심 및 크기, fx, fy : 그래디언트 변화의 시작점

[패턴 (Pattern)]
<defs> 에서 패턴을 정의하고, fill 이나 stroke 에서 속성으로 참조(사용)
<pattern id="..." 속성들 >
patternUnits = "userSpaceOnUse | objectBoundingBox"
x, y, width, height : 패턴의 뷰포트 크기
viewBox : 패턴의 뷰박스

[투명도의 특성]
객체와 그룹의 투명도 처리
각 객체의 투명도를 적용하여 렌더링 한 후 그룹의 렌더링을 적용한다

[SVG Animation 기본]
SVG Animation Elements
<animate ... /> 시간축에 속성이나 특성의 수치 값을 할당
<set ... /> 비수치값을 가지는 속성 애니메이션
<animateMotion ... /> 패스에 따라 이동
<animateColor ... /> 색상 애니메이션
<animateTransform ... /> 변환 애니메이션

- Animation 의 개념
어떤 객체가 어떻게 변화 => 무엇을, 얼마만큼, 언제 : attribute, from/to, begin/dur
<animate attributeName="속성이름" attributeType="속성유형(XML | CSS | auto)"
        from="시작값" to="종료값" begin="시작시간" dur="지속시간"
         values="변화값;변화값;…" calcMode="linear | discrete …">
        discrete : 앞의 값에서 다음 값으로 점프하도록 지정
        linear : 동일한 속도로 변화되도록 지정