실습 코드 분석

2021년 11월 30일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

이름 : 국 승 주

tmdwn5430@office.uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

차 례

[1. 분석 코드 1](#_Toc89181097)

[1.1. 데이터레이어 표현하기 1](#_Toc89181098)

[1.2. 지난 주 실습 코드 1](#_Toc89181099)

[2. 주요 개념 정리 2](#_Toc89181100)

[2.1. jQuery-AJAX 2](#_Toc89181101)

[2.2. JSON 2](#_Toc89181102)

[2.3. GeoJSON 2](#_Toc89181103)

[3. 코드에서 사용한 클라스 3](#_Toc89181104)

[3.1. 클라스의 주된 목적 3](#_Toc89181105)

[3.2. 사용한 프로퍼티 설명 3](#_Toc89181106)

[3.3. 사용한 메소드 설명 3](#_Toc89181107)

[4. 실습 코드 분석 4](#_Toc89181108)

[4.1. 클라스의 주된 목적 4](#_Toc89181109)

# 분석 코드

## 데이터레이어 표현하는 코드

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no">

    <title>간단한 지도 표시하기</title>

*<!-- jquery -->*

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

*<!-- naver map -->*

    <script type="text/javascript" src="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=3j74nj8hha"></script>

</head>

<body>

<div id="map" style="width:100%;height:800px;"></div>

<script>

var HOME\_PATH = "https://navermaps.github.io/maps.js/docs",

    urlPrefix = HOME\_PATH +'/data/region',

    urlSuffix = '.json',

    regionGeoJson = [],

    loadCount = 0;

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 18,

    mapTypeId: 'normal',

    center: new naver.maps.LatLng(37.3586524, 127.1060678)

});

*// 지도가 로드된 다음에 일회성으로 실행되는 이벤트*

var HOME\_PATH = window.HOME\_PATH || '.';

naver.maps.Event.once(map, 'init\_stylemap', function () {

    $.ajax({

        url: HOME\_PATH +'/data/naver.json',

        dataType: 'json',

        success: startDataLayer

    });

});

var tooltip = $('<div style="position:absolute;z-index:1000;padding:5px 10px;background-color:#fff;border:solid 2px #000;font-size:14px;pointer-events:none;display:none;"></div>');

tooltip.appendTo(map.getPanes().floatPane);

function startDataLayer(geojson) {

    map.data.addGeoJson(geojson);

    map.data.setStyle(function(feature) {

        var color = 'red';

*if* (feature.getProperty('isColorful')) {

            color = feature.getProperty('color');

        }

*return* {

            fillColor: color,

            strokeColor: color,

            strokeWeight: 2,

            icon: null

        };

    });

    map.data.addListener('click', function(e) {

        e.feature.setProperty('isColorful', true);

    });

    map.data.addListener('dblclick', function(e) {

        var bounds = e.feature.getBounds();

*if* (bounds) {

            map.panToBounds(bounds);

        }

    });

    map.data.addListener('mouseover', function(e) {

        map.data.overrideStyle(e.feature, {

            strokeWeight: 8,

            icon: HOME\_PATH +'/img/example/pin\_spot.png'

        });

    });

    map.data.addListener('mouseout', function(e) {

        map.data.revertStyle();

    });

}

</script>

</body>

</html>

## 지난 주 실습 코드

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no">

    <title>간단한 지도 표시하기</title>

    <style>

        table {

          width: 100%;

        }

        th, td {

          border: 1px solid black;

        }

    </style>

*<!-- jquery -->*

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

*<!-- naver map -->*

    <script type="text/javascript" src="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=3j74nj8hha"></script>

    <script>

        function loadDoc() {

            var xhr = new XMLHttpRequest();

            var loc1 = document.getElementById("loc").innerHTML;

            var url = 'http://apis.data.go.kr/1741000/DisasterMsg4/getDisasterMsg2List'; */\*URL\*/*

            var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'R%2F5f0YzRaGQ7XNm6bvA5bZYpaxbTcHGPQOUAjpPfo9IGZgtjZw9iuZkTHeBVjCKgIhi%2BJlUgCZVagxNUnlrOPw%3D%3D'; */\*Service Key\*/*

            queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); */\*\*/*

            queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); */\*\*/*

            queryParams += '&' + encodeURIComponent('type') + '=' + encodeURIComponent('xml'); */\*\*/*

            queryParams += '&' + encodeURIComponent('create\_date') + '=' + encodeURIComponent('2021/06/22 00:00:00'); */\*\*/*

            queryParams += '&' + encodeURIComponent('location\_name') + '=' + encodeURIComponent(loc1); */\*\*/*

            xhr.open('GET', url + queryParams);

            xhr.onreadystatechange = function () {

*if* (this.readyState == 4) {

                    myFunction(this);

                }

            };

            xhr.send('');

        }

        function myFunction(xhttp) {

            let i;

            let xmlDoc = xhttp.responseXML;

            let table="<tr><th>location</th><th>massage</th></tr>";

            let x = xmlDoc.getElementsByTagName("row");

*for* (i = 0; i <x.length; i++) {

                table += "<tr><td>" +

                x[i].getElementsByTagName("location\_name")[0].childNodes[0].nodeValue +

                "</td><td>"+

                x[i].getElementsByTagName("msg")[0].childNodes[0].nodeValue +

                "</td></tr>";

            }

            document.getElementById("list").innerHTML = table;

        }

    </script>

</head>

<body>

<div id="map" style="width:100%;height:400px;"></div>

<hr>

<div>현재 선택된 지역 : </div><div id="loc"></div>

<table id="list"></table>

<script>

var HOME\_PATH = "https://navermaps.github.io/maps.js/docs",

    urlPrefix = HOME\_PATH +'/data/region',

    urlSuffix = '.json',

    regionGeoJson = [],

    loadCount = 0;

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 7,

    mapTypeId: 'normal',

    center: new naver.maps.LatLng(36.4203004, 128.317960)

});

*// 지도가 로드된 다음에 일회성으로 실행되는 이벤트*

naver.maps.Event.once(map, 'init\_stylemap', function () {

*for* (var i = 1; i < 18; i++) {

        var keyword = i +'';

*if* (keyword.length === 1) {

            keyword = '0'+ keyword;

        }

*// 01, 02, 03 생성*

        $.ajax({

            url: urlPrefix + keyword + urlSuffix,

            success: function(idx) {

*return* function(geojson) {

                    regionGeoJson[idx] = geojson;

                    loadCount++;

*// 17개 다 받아오면 그리기 시작*

*if* (loadCount === 17) {

                        startDataLayer();

                    }

                }

            }(i - 1)

        });

    }

});

var tooltip = $('<div style="position:absolute;z-index:1000;padding:5px 10px;background-color:#fff;border:solid 2px #000;font-size:14px;pointer-events:none;display:none;"></div>');

tooltip.appendTo(map.getPanes().floatPane);

function startDataLayer() {

    map.data.setStyle(function(feature) {

*// 기본 라인*

        var styleOptions = {

            fillColor: '#ff0000',

            fillOpacity: 0.0001,

            strokeColor: '#ff0000',

            strokeWeight: 2,

            strokeOpacity: 0.4

        };

*// 마우스 올라갔을 떄 스타일*

*if* (feature.getProperty('focus')) {

            styleOptions.fillOpacity = 0.6;

            styleOptions.fillColor = '#0f0';

            styleOptions.strokeColor = '#0f0';

            styleOptions.strokeWeight = 4;

            styleOptions.strokeOpacity = 1;

        }

*return* styleOptions;

    });

*// 제가 받아온 Geojson파일을 map.data*

    regionGeoJson.forEach(function(geojson) {

        map.data.addGeoJson(geojson);

    });

    map.data.addListener('click', function(e) {

        var feature = e.feature;

*// 여기서 다른 선택된 지역 있는지 지워줘야 됩니당*

*// 지워주는 방법은*

*// map.data.\_features 를 foreach 하면서 focus 끄시면 됩니다.*

        map.data.forEach(reset);

        function reset() {

            feature.setProperty('focus', false);

        }

*if* (feature.getProperty('focus') !== true) {

            map.data.\_features.forEach(

                function (feature) {

                    feature.setProperty('focus', false);

                }

            );

            document.getElementById("loc").innerHTML = feature.getProperty('area1');

            feature.setProperty('focus', true);

            loadDoc();

        } *else* {

            feature.setProperty('focus', false);

        }

    });

    map.data.addListener('mouseover', function(e) {

*// 이름을 받아*

        var feature = e.feature,

            regionName = feature.getProperty('area1');

*// 강원도*

*// 지역명 뜨는거*

        tooltip.css({

            display: '',

            left: e.offset.x,

            top: e.offset.y

        }).text(regionName);

        map.data.overrideStyle(feature, {

            fillOpacity: 0.6,

            strokeWeight: 4,

            strokeOpacity: 1

        });

    });

    map.data.addListener('mouseout', function(e) {

*// 툴팁 사라지게 하고*

        tooltip.hide().empty();

*// 스타일 원상복귀시키는*

        map.data.revertStyle();

    });

}

</script>

</body>

</html>

# 주요 개념 정리

## jQuery-AJAX

먼저 jQuery는 HTML의 클라이언트 사이드 조작을 단순화하도록 설계된 크크로스 플랫폼의 자바스크립트 라이브러리이다. jQuery를 사용하면 문서 객체 모델(DOM)과 이벤트에 관한 처리를 손쉽게 구현할 수 있다. 코드 보기, 문서 객체 모델 찾기, 애니메이션 만들기, 이벤트 제어, Ajax 개발을 쉽게 할 수 있도록 디자인되었다.

그 중에서 Ajax(Asynchronous JavaScript and XML)는 비동기적인 웹 애플리케이션의 제작을 위해 아래와 다음과 같은 조합을 이용하는 웹 개발 기법이다.

* 표현 정보를 위한 HTML (또는 XHTML)과 CSS
* 동적인 화면 출력 및 표시 정보와의 상호작용을 위한 DOM, JavaScript
* 웹 서버와 비동기적으로 데이터를 교환하고 조작하기 위한 XML, XSLT, XMLHttpRequest (Ajax 애플리케이션은 XML/XSLT 대신 미리 정의된 HTML이나 일반 텍스트, JSON, JSON-RPC를 이용할 수 있다.)

Ajax를 사용하면 페이지 이동없이 고속으로 화면을 전환할 수 있으며, 서버 처리를 기다리지 않고 비동기 요청이 가능하다. 또한 수신하는 데이터 양을 줄일 수 있고, 클라이언트에게 처리를 위임할 수 있다. 플러그인 없이도 인터렉티브한 웹페이지 구현할 수 있다는 장점이 있다. 하지만, 여러 가지 단점도 존재하는데, Ajax를 쓸 수 없는 브라우저에 대한 문제가 있으며 HTTP 클라이언트의 기능이 한정되어 있다. 페이지 이동 없는 통신으로 인하여 보안상의 문제가 존재하며 지원하는 Charset이 한정되어 있다. 스크립트로 작성되므로 디버깅이 용이하지 않고, 요청을 남발하면 역으로 서버 부하가 늘 수 있다는 단점이 있다.

## JSON

JavaScript에서는 key를 저장하는 것을 다양한 방법으로 활용할 수 있다. 하지만 그 중 1가지 방법으로만 사용할 수 있는데, 이를 통일시키기 위해 나온 것이 JSON(JavaScript Object Notation)이다. JSON은 “속성-값” 쌍 또는 “키-값” 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML(AJAX가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수 값을 표현하는 데 적합하다.

## GeoJSON

GeoJSON은 JSON의 한 종류이며, 위치정보를 갖는 점을 기반으로 체계적으로 지형을 표현하기 위해 설계된 개방형 공개 표준 형식이다. 지리좌표계의 점을 기반으로 Geocoding된 지형지물(주소 및 위치), 라인스트링(LineString – 거리, 고속도로 및 경계등 정보를 담고 있는 문자열) 또는 폴리라인, 다각형 (국가, 도시, 토지) 및 이러한 유형의 여러 부분으로 구성된 모음을 특징으로 한다.

# 코드에서 사용한 클라스

## 클라스의 주된 목적

클라스는 객체 지향 프로그래밍에서 특정 객체를 생성하기 위해 변수와 메소드를 정의하는 일종의 틀로, 객체를 정의하기 위한 상태(멤버 변수)와 메소드(함수)로 구성된다. 실무에서는 동일한 종류의 객체를 여러 개 생성해야 하는 경우가 많은데, 이 때 클라스를 사용한다면 더욱 깔끔하게 정의할 수 있다. 따라서 클라스를 사용해 여러 객체를 정의하는 것이다.

## 사용한 프로퍼티 설명

### 프로퍼티의 주된 기능

프로퍼티는 해당 object의 특징을 의미한다. 크게 2가지 종류가 있는데 인스턴스 프로퍼티(Instance Property)와 정적 프로퍼티(Static Property)가 있다. 인스턴스 프로퍼티는 특정 object 인스턴스의 특정한 데이터를 갖고 있으며, 정적 프로퍼티는 모든 object 인스턴스들에게 공유된 데이터를 가지고 있다. 즉, 프로퍼티는 object의 특징을 저장한 것이라고 할 수 있다.

### 실습 코드에서 기능

1. 데이터레이어 표현하는 코드

: 해당 코드에서는 다음과 같은 부분으로 클라스를 지정하였다. (29번째 줄)

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 18, *//초기 배율 설정*

    mapTypeId: 'normal', *// 초기 지도 유형의 id*

    center: new naver.maps.LatLng(37.3586524, 127.1060678) *//초기 지도 가운데 지정*

});

‘map’이라는 매개변수 객체를 생성하였는데, ‘Map’이라는 이름의 class로 값을 지정한 것이다. 이 때 naver.maps.라는 부분은 namespace로 여러 개의 javascript 코드를 사용할 때에 중복되는 클라스 이름이 생성되지 않도록 작성해주는 것이다.

다음은 naver에서 제공하는 Map 클라스의 프로퍼티 목록이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 1. Map 프로퍼티 종류

1. 지난 주 실습 코드

: 해당 코드에서는 다음과 같은 부분으로 클라스를 지정하였다. (75번째 줄)

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 7, *//초기 배율 설정*

    mapTypeId: 'normal', *// 초기 지도 유형의 id*

    center: new naver.maps.LatLng(36.4203004, 128.317960)

});

## 사용한 메소드 설명

### 메소드의 주된 기능

프로퍼티는 해당 object의 특징을 의미한다. 크게 2가지 종류가 있는데 인스턴스 프로퍼티(Instance Property)와 정적 프로퍼티(Static Property)가 있다. 인스턴스 프로퍼티는 특정 object 인스턴스의 특정한 데이터를 갖고 있으며, 정적 프로퍼티는 모든 object 인스턴스들에게 공유된 데이터를 가지고 있다. 즉, 프로퍼티는 object의 특징을 저장한 것이라고 할 수 있다.

### 실습 코드에서 기능

1. 데이터레이어 표현하는 코드

: 해당 코드에서는 다음과 같은 부분으로 메소드를 지정하였다. (32번째 줄)

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 18, *//초기 배율 설정*

    mapTypeId: 'normal', *// 초기 지도 유형의 id*

    center: new naver.maps.LatLng(37.3586524, 127.1060678) *//초기 지도 가운데 지정*

});

‘Map’이라는 이름의 class에 ‘center’이라는 메소드를 지정해 준 것이다. 이를 통해서 초기 지도를 로딩할 때 가운데는 (37.3586524, 124.1060678)이 되는 것이다.

1. 지난 주 실습 코드

: 해당 코드에서는 다음과 같은 부분으로 메소드를 지정하였다. (78번째 줄)

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 7, *//초기 배율 설정*

    mapTypeId: 'normal', *// 초기 지도 유형의 id*

    center: new naver.maps.LatLng(36.4203004, 128.317960)

});

# 실습 코드 분석

## 데이터레이어 표현하는 코드

*// 지도 로드하기*

var HOME\_PATH = "https://navermaps.github.io/maps.js/docs",

    urlPrefix = HOME\_PATH +'/data/region',

    urlSuffix = '.json',

    regionGeoJson = [],

    loadCount = 0;

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 18, *//초기 배율 설정*

    mapTypeId: 'normal', *// 초기 지도 유형의 id*

    center: new naver.maps.LatLng(37.3586524, 127.1060678) *//초기 지도 가운데 지정*

});

각종 변수들의 초기 값을 지정해 준다.

*// 지도가 로드된 다음에 일회성으로 실행되는 이벤트*

var HOME\_PATH = window.HOME\_PATH || '.';

naver.maps.Event.once(map, 'init\_stylemap', function () {

    $.ajax({

        url: HOME\_PATH +'/data/naver.json',

        dataType: 'json',

        success: startDataLayer*//성공 시  실행 할 함수*

    });

});

ajax를 이용해 지정된 url(네이버 서버)에서 geojson 데이터를 받아온다. 만약 성공하면 startDataLayer 함수를 실행시킨다.

function startDataLayer(geojson) {

    map.data.addGeoJson(geojson); *//map.data에 geojson 추가*

    map.data.setStyle(function(feature) {*//스타일 지정*

        var color = 'red';

*if* (feature.getProperty('isColorful')) {

            color = feature.getProperty('color');*//원래 기존 색*

        }

*return* {

            fillColor: color, *//내부 색*

            strokeColor: color, *//외곽선 색*

            strokeWeight: 2, *//외곽선 두께*

            icon: null

        };

    });

map의 data객체에 geojson 정보를 저장한다. 그 후 setStyle을 통해서 지도에 표현되는 각종 정보의 초기값을 지정한다.

    map.data.addListener('click', function(e) {

        e.feature.setProperty('isColorful', true);

    });

    map.data.addListener('dblclick', function(e) {

        var bounds = e.feature.getBounds(); *//객체 좌표 경계*

*if* (bounds) {

            map.panToBounds(bounds); *//지도 부드럽게 연결*

        }

    });

    map.data.addListener('mouseover', function(e) {

        map.data.overrideStyle(e.feature, {

            strokeWeight: 8, *//외곽선 두께 8 지정*

            icon: HOME\_PATH +'/img/example/pin\_spot.png'

        });

    });

    map.data.addListener('mouseout', function(e) {

        map.data.revertStyle(); *//원래 스타일로*

    });

}

이벤트 발생 시 스타일을 지정하는 부분인데, addListner를 통해서 지정할 수 있다. Click의 경우 내부 색을 변경하고, 마우스가 지정된 범위 위에 올라가면 외곽선 두께를 8로 변경하며, 마우스가 지정된 범위 위에서 사라지면 원래 스타일로 돌아오는 것이다.

## 지난 주 실습 코드

function loadDoc() {

            var xhr = new XMLHttpRequest();

            var loc1 = document.getElementById("loc").innerHTML;

            var url = 'http://apis.data.go.kr/1741000/DisasterMsg4/getDisasterMsg2List'; */\*URL\*/*

            var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'R%2F5f0YzRaGQ7XNm6bvA5bZYpaxbTcHGPQOUAjpPfo9IGZgtjZw9iuZkTHeBVjCKgIhi%2BJlUgCZVagxNUnlrOPw%3D%3D'; */\*Service Key\*/*

            queryParams += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1'); */\*\*/*

            queryParams += '&' + encodeURIComponent('numOfRows') + '=' + encodeURIComponent('10'); */\*\*/*

            queryParams += '&' + encodeURIComponent('type') + '=' + encodeURIComponent('xml'); */\*\*/*

            queryParams += '&' + encodeURIComponent('create\_date') + '=' + encodeURIComponent('2021/06/22 00:00:00'); */\*\*/*

            queryParams += '&' + encodeURIComponent('location\_name') + '=' + encodeURIComponent(loc1); */\*\*/*

            xhr.open('GET', url + queryParams);

            xhr.onreadystatechange = function () {

*if* (this.readyState == 4) {

                    myFunction(this);

                }

            };

            xhr.send('');

        }

재난문자를 로딩하는 함수이다. 제대로 로딩이 되었다면 myFunction 함수를 실행시킨다.

function myFunction(xhttp) {

            let i;

            let xmlDoc = xhttp.responseXML;

            let table="<tr><th>location</th><th>massage</th></tr>";

            let x = xmlDoc.getElementsByTagName("row");

*for* (i = 0; i <x.length; i++) {

                table += "<tr><td>" +

                x[i].getElementsByTagName("location\_name")[0].childNodes[0].nodeValue +

                "</td><td>"+

                x[i].getElementsByTagName("msg")[0].childNodes[0].nodeValue +

                "</td></tr>";

            }

            document.getElementById("list").innerHTML = table;

        }

재난문자를 표 형태로 나타내는 함수이다. For문을 통해서 재난문자를 하나씩 구별하였다.

<div id="map" style="width:100%;height:400px;"></div>

<hr>

<div>현재 선택된 지역 : </div><div id="loc"></div>

<table id="list"></table>

Html의 body부분이다. 지도와 현재 선택된 지역을 알 수 있도록 하였다.

var HOME\_PATH = "https://navermaps.github.io/maps.js/docs",

    urlPrefix = HOME\_PATH +'/data/region',

    urlSuffix = '.json',

    regionGeoJson = [],

    loadCount = 0;

var map = new naver.maps.Map(document.getElementById('map'), {

    zoom: 7, *//초기 배율 설정*

    mapTypeId: 'normal', *// 초기 지도 유형의 id*

    center: new naver.maps.LatLng(36.4203004, 128.317960)

});

각종 변수들의 초기값을 지정하였다.

*// 지도가 로드된 다음에 일회성으로 실행되는 이벤트*

naver.maps.Event.once(map, 'init\_stylemap', function () {

*for* (var i = 1; i < 18; i++) {

        var keyword = i +'';

*if* (keyword.length === 1) {

            keyword = '0'+ keyword;

        }

*// 01, 02, 03 생성*

        $.ajax({

            url: urlPrefix + keyword + urlSuffix,

            success: function(idx) {

*return* function(geojson) {

                    regionGeoJson[idx] = geojson;

                    loadCount++;

*// 17개 다 받아오면 그리기 시작*

*if* (loadCount === 17) {

                        startDataLayer();

                    }

                }

            }(i - 1)

        });

    }

});

지도가 로드된 후 실행되는 함수이다. ajax를 이용해 geojson 데이터를 받아온다. 만약 성공하면 function(idx)를 실행시킨다.

function startDataLayer() {

    map.data.setStyle(function(feature) {

*// 기본 라인*

        var styleOptions = {

            fillColor: '#ff0000',

            fillOpacity: 0.0001,

            strokeColor: '#ff0000',

            strokeWeight: 2,

            strokeOpacity: 0.4

        };

*// 마우스 올라갔을 떄 스타일*

*if* (feature.getProperty('focus')) {

            styleOptions.fillOpacity = 0.6;

            styleOptions.fillColor = '#0f0';

            styleOptions.strokeColor = '#0f0';

            styleOptions.strokeWeight = 4;

            styleOptions.strokeOpacity = 1;

        }

*return* styleOptions;

    });

초기 스타일과 마우스가 올라갔을 때 스타일을 지정해준다. 내부 색, 외곽선, 외곽선 두께 등을 확인할 수 있다.

*// 제가 받아온 Geojson파일을 map.data*

    regionGeoJson.forEach(function(geojson) {

        map.data.addGeoJson(geojson);

    });

Geojson 파일을 map.data에 저장하는 것이다.

    map.data.addListener('click', function(e) {

        var feature = e.feature;

*// 여기서 다른 선택된 지역 있는지 지워줘야 됩니당*

*// 지워주는 방법은*

*// map.data.\_features 를 foreach 하면서 focus 끄시면 됩니다.*

        map.data.forEach(reset);

        function reset() {

            feature.setProperty('focus', false);

        }

*if* (feature.getProperty('focus') !== true) {

            map.data.\_features.forEach(

                function (feature) {

                    feature.setProperty('focus', false);

                }

            );

            document.getElementById("loc").innerHTML = feature.getProperty('area1');

            feature.setProperty('focus', true);

            loadDoc();

        } *else* {

            feature.setProperty('focus', false);

        }

    });

지도의 지정된 영역을 1번 클릭했을 때 실행되는 이벤트이다.

    map.data.addListener('mouseover', function(e) {

*// 이름을 받아*

        var feature = e.feature,

            regionName = feature.getProperty('area1');

*// 강원도*

*// 지역명 뜨는거*

        tooltip.css({

            display: '',

            left: e.offset.x,

            top: e.offset.y

        }).text(regionName);

        map.data.overrideStyle(feature, {

            fillOpacity: 0.6,

            strokeWeight: 4,

            strokeOpacity: 1

        });

    });

    map.data.addListener('mouseout', function(e) {

*// 툴팁 사라지게 하고*

        tooltip.hide().empty();

*// 스타일 원상복귀시키는*

        map.data.revertStyle();

    });

}

마우스가 지정된 영역에 올라가거나 내려왔을 때 실행되는 이벤트이다.