Final Project

2021년 12월 14일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

이승연

atishoo2387@office.uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

차 례

[1. 요약 1](#_Toc90345952)

[2. 요구사항 도출 2](#_Toc90345953)

[2.1. 사용 시나리오 2](#_Toc90345954)

[2.2. 주요 요구사항 3](#_Toc90345955)

[3. 구현 4](#_Toc90345956)

[3.1. 코드의 주요 구조 4](#_Toc90345957)

[3.2. 필수 라이브러리 7](#_Toc90345958)

[4. 토론 8](#_Toc90345959)

[4.1. 도출한 요구사항의 완성도 8](#_Toc90345960)

[4.2. 선택할 수 있는 다양한 기술 분석 8](#_Toc90345961)

[5. 결론 9](#_Toc90345962)

[5.1. 과제 구현 결과 9](#_Toc90345963)

[5.2. 향후 방향 9](#_Toc90345964)

[6. 부록 11](#_Toc90345965)

[6.1. 활용한 기술 설명 11](#_Toc90345966)

[6.2. 전체 소스코드 11](#_Toc90345967)

[7. 참고자료 리스트 18](#_Toc90345968)

# 요약

지도이(가) 표시된 사진

자동 생성된 설명

지도의 오른쪽 위에 컨트롤 버튼을 이용하여 클릭만으로 지도를 줌 인, 줌 아웃 할 수 있다. 지도를 클릭하면 마커를 생성하고 그 지점의 위도/경도를 정보를 출력한다. 또한 지도 하단에서는 클릭한 해당 행정구역의 재난문자 내용을 확인할 수 있다. 마우스 오른쪽 클릭 시 지도에 마커를 생성하지 않고 그 지점의 위도와 경도 정보를 확인할 수 있다. 지도의 가시성을 높이기 위하여 행정구역의 경계를 표시하였다.

# 요구사항 도출

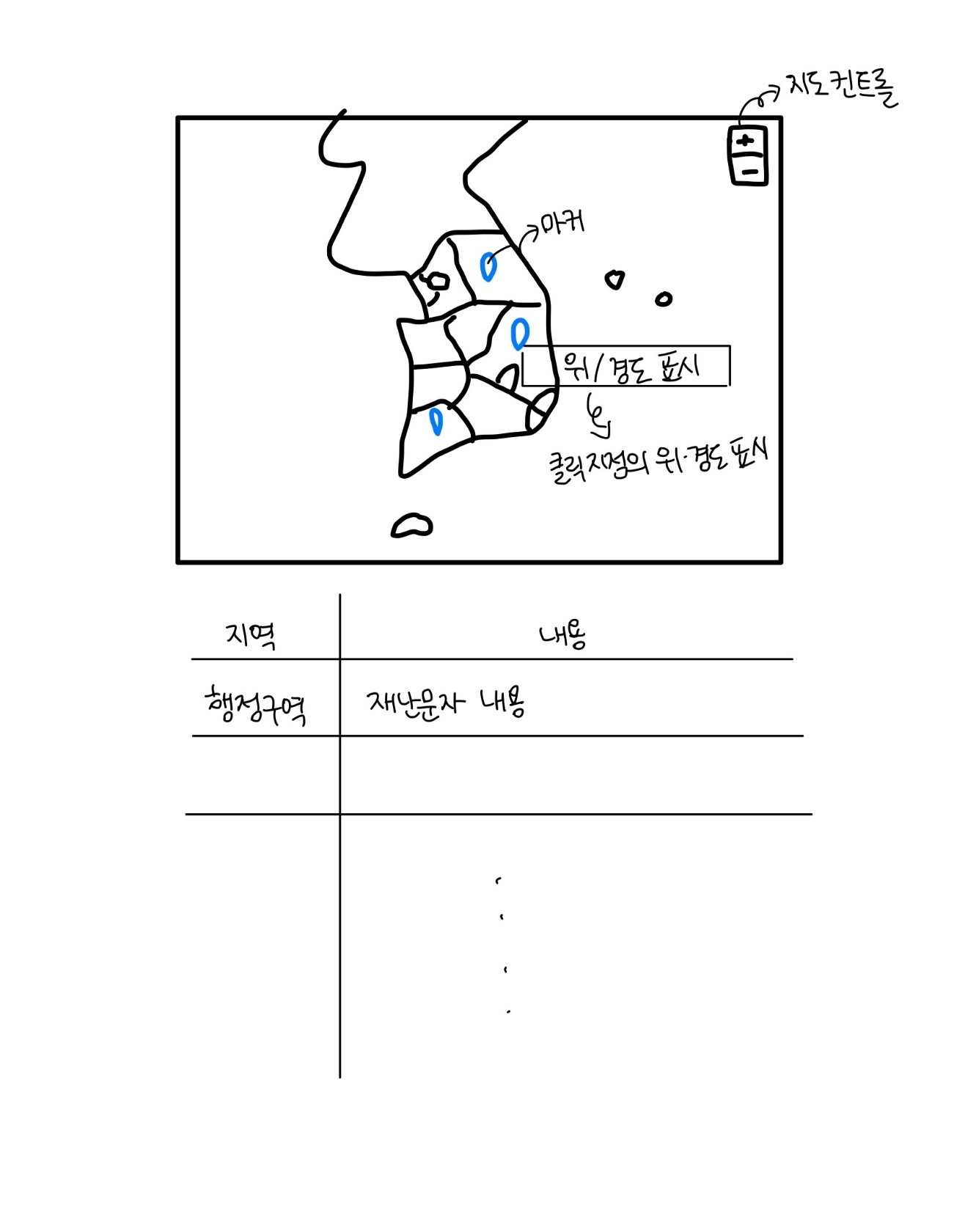
## 사용 시나리오

1. Map Control

2. Marker

3. Latitude & Longitude

4 Sketch



## 주요 요구사항

Map Control - 휠 사용 없이 클릭으로만 지도를 줌 인/아웃

Marker - 클릭한 부분에 마커 생성

Latitude & Longitude - 클릭한 부분의 위/경도 표시

지도 클릭 시 해당 행정 구역의 재난문자 내용 표시

# 구현

## 코드의 주요 구조

Map control 코드로 네이버 지도의 zoomControl객체를 사용하고 절대위치는 오른쪽 위로 지정

var mapOptions = {

    zoom: 7,

    zoomControl: true,

    zoomControlOptions: {

        style: naver.maps.ZoomControlStyle.SMALL,

        position: naver.maps.Position.TOP\_RIGHT

    }

};

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

마커 삽입과 클릭 시 위도와 경도를 나타내는 코드로 마커는 좌 클릭 시 발생, 위도와 경도 정보는 좌/우 상관없이 클릭 시 발생

var markerList = [];

    var menuLayer = $('<div style="position:absolute;z-index:10000;background-color:#fff;border:solid 1px #333;padding:10px;display:none;"></div>');

    map.getPanes().floatPane.appendChild(menuLayer[0]);

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

        var feature = e.feature;

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

            var marker = new naver.maps.Marker({

            position: e.coord,

            map: map

            });

            markerList.push(marker);

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

            queryParams += '&' + encodeURIComponent('location\_name') + '=' + encodeURIComponent(feature.getProperty('area1')); /\*\*/

            var coordHtml =

            '클릭 지점 위/경도 좌표' + '<br />' +

            'Point: ' + e.point;

            menuLayer.show().css({

                left: e.offset.x,

                top: e.offset.y

            }).html(coordHtml);

            console.log('Coord: ' + e.coord.toString());

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

            xhr.onreadystatechange = function () {

                if (this.readyState == 4) {

                    document.getElementById("table").innerHTML = 'Status: '+this.status+'nHeaders: '+JSON.stringify(this.getAllResponseHeaders())+'nBody: '+this.responseText;

                    xhr.onload = function(){

                    myFunction(this);

                    }

                    function myFunction(xml){

                    const xmlDoc = xml.responseXML;

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

                    let table = "<tr><th><p id='me'>지역</p></th><th>내용</th></tr>";

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

                        table += "<tr><td><p id='me'>" + x[i].getElementsByTagName("location\_name")[0].childNodes[0].nodeValue +"</p></td><td>"+ x[i].getElementsByTagName("msg")[0].childNodes[0].nodeValue + "</td></tr>";

                    }

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

                    }

                }

            };

            xhr.send('');

        } else {

            feature.setProperty('focus', false);

        }

    });

    naver.maps.Event.addListener(map, 'keydown', function(e) {

        var keyboardEvent = e.keyboardEvent,

            keyCode = keyboardEvent.keyCode || keyboardEvent.which;

        var ESC = 27;

        if (keyCode === ESC) {

            keyboardEvent.preventDefault();

            for (var i=0, ii=markerList.length; i<ii; i++) {

                markerList[i].setMap(null);

            }

            markerList = [];

            menuLayer.hide();

        }

    });

    naver.maps.Event.addListener(map, 'mousedown', function(e) {

        menuLayer.hide();

    });

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

        var coordHtml =

            '클릭 지점 위/경도 좌표' + '<br />' +

            'Point: ' + e.point;

        menuLayer.show().css({

            left: e.offset.x,

            top: e.offset.y

        }).html(coordHtml);

        console.log('Coord: ' + e.coord.toString());

    });

## 필수 라이브러리

Naver maps Api

naver.maps.Marker: 지도 위에 마커를 올릴 때 사용하는 클래스

naver.maps.Polyline: 지도 위에 선을 그릴 때 사용하는 클래스

Geojson

# 토론

## 도출한 요구사항의 완성도

지도를 클릭할 시 위도/경도 좌표를 나타내는 것이 아니라 주소 정보를 나타냈다면 더욱 편의성을 갖춘 지도가 되었을 것이다. 또한 마커 클러스터를 이용하여 지도를 줌인&아웃 시 더욱 편리하게 마커 정보를 확인할 수 있을 것이다.

## 선택할 수 있는 다양한 기술 분석

네이버 지도에서 제공하는 Geocoding과 Reverse Geocoding API를 이용하여 지도의 좌표 값을 주소로 나타낼 수 있다. Geocoder 서브 모듈의 Service 객체를 사용하여 검색 기능도 구현할 수 있다.

# 결론

## 과제 구현 결과

지도이(가) 표시된 사진

자동 생성된 설명

마우스 휠이 아닌 클릭으로 지도를 줌 인&아웃 할 수 있고 지도를 클릭 시 마커와 그 지점의 위도/경도가 표시된다. 또한 지도 하단에는 해당 행정구역의 재난문자 내용이 표시된다. 오른쪽 클릭을 할 시 마커는 생성되지 않고 그 위치의 위도와 경도를 확인할 수 있다.

## 향후 방향

공공 데이터 포털에서 재난문자와 관련된 내용 말고도 여러 정보들을 받아와 지도와 연동할 수 있을 것 같다. 예를 들면 Geolocation API를 활용하여 내 위치를 지도에 표기하고 공공데이터 포털의 데이터와 연동하여 내 주변의 코로나 진단소를 마커로 표기할 수 있을 것이다. 또한 줌아웃을 했을 경우 마커를 클러스터화 하여 데이터를 더욱 보기 좋게 나타낼 수 있다.

# 부록

## 활용한 기술 설명

Map: 행정구역을 표시하기 위해 GeoJson 파일을 받아오고 매서드는 GET을 사용한다.

Map control : 네이버 지도가 제공하는 ZoomControl 객체를 사용한다.

Marker: 마커를 생성하기 위한 배열 생성 후 클릭 시 배열에 생성된 마커 추가, 지도에 표시한다.

Latitude & Longitude : 화면에 띄울 menulayer를 생성한 후 네이버 지도에서 받아온 위/경도 값을 표시한다.

## 전체 소스코드

<!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>Final Project</title>

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

<script type="text/javascript" src="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=yje5mz1g6c"></script>

<style>

table,th,td {

    border : 1px solid black;

    border-collapse: collapse;

}

th,td {

    padding: 5px;

}

#me {

    width: 200px;

}

</style>

</head>

<body>

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

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

<script>

var xhr = new XMLHttpRequest();

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

var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'ZBsjF/dpDGy4ttro8MXcmk54MzIqx7MKRa8Hjl0empaGL2tMoQ23G2av09XPy8LYe875HB1T18/nyRtNdMwbbQ=='; /\*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');

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

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

    urlSuffix = '.json',

    regionGeoJson = [],

    loadCount = 0;

var mapOptions = {

    zoom: 7,

    zoomControl: true,

    zoomControlOptions: {

        style: naver.maps.ZoomControlStyle.SMALL,

        position: naver.maps.Position.TOP\_RIGHT

    }

};

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

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;

        }

        $.ajax({

            url: urlPrefix + keyword + urlSuffix,

            success: function(idx) {

                return function(geojson) {

                    regionGeoJson[idx] = geojson;

                    loadCount++;

                    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) {

        var styleOptions = {

            fillColor: '#ff0000',

            fillOpacity: 0.0001,

            strokeColor: '#ff0000',

            strokeWeight: 2,

            strokeOpacity: 0.4

        };

        return styleOptions;

    });

    regionGeoJson.forEach(function(geojson) {

        map.data.addGeoJson(geojson);

    });

    var markerList = [];

    var menuLayer = $('<div style="position:absolute;z-index:10000;background-color:#fff;border:solid 1px #333;padding:10px;display:none;"></div>');

    map.getPanes().floatPane.appendChild(menuLayer[0]);

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

        var feature = e.feature;

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

            var marker = new naver.maps.Marker({

            position: e.coord,

            map: map

            });

            markerList.push(marker);

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

            queryParams += '&' + encodeURIComponent('location\_name') + '=' + encodeURIComponent(feature.getProperty('area1')); /\*\*/

            var coordHtml =

            '클릭 지점 위/경도 좌표' + '<br />' +

            'Point: ' + e.point;

            menuLayer.show().css({

                left: e.offset.x,

                top: e.offset.y

            }).html(coordHtml);

            console.log('Coord: ' + e.coord.toString());

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

            xhr.onreadystatechange = function () {

                if (this.readyState == 4) {

                    document.getElementById("table").innerHTML = 'Status: '+this.status+'nHeaders: '+JSON.stringify(this.getAllResponseHeaders())+'nBody: '+this.responseText;

                    xhr.onload = function(){

                    myFunction(this);

                    }

                    function myFunction(xml){

                    const xmlDoc = xml.responseXML;

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

                    let table = "<tr><th><p id='me'>지역</p></th><th>내용</th></tr>";

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

                        table += "<tr><td><p id='me'>" + x[i].getElementsByTagName("location\_name")[0].childNodes[0].nodeValue +"</p></td><td>"+ x[i].getElementsByTagName("msg")[0].childNodes[0].nodeValue + "</td></tr>";

                    }

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

                    }

                }

            };

            xhr.send('');

        } else {

            feature.setProperty('focus', false);

        }

    });

    naver.maps.Event.addListener(map, 'keydown', function(e) {

        var keyboardEvent = e.keyboardEvent,

            keyCode = keyboardEvent.keyCode || keyboardEvent.which;

        var ESC = 27;

        if (keyCode === ESC) {

            keyboardEvent.preventDefault();

            for (var i=0, ii=markerList.length; i<ii; i++) {

                markerList[i].setMap(null);

            }

            markerList = [];

            menuLayer.hide();

        }

    });

    naver.maps.Event.addListener(map, 'mousedown', function(e) {

        menuLayer.hide();

    });

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

        var coordHtml =

            '클릭 지점 위/경도 좌표' + '<br />' +

            'Point: ' + e.point;

        menuLayer.show().css({

            left: e.offset.x,

            top: e.offset.y

        }).html(coordHtml);

        console.log('Coord: ' + e.coord.toString());

    });

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

        tooltip.hide().empty();

        map.data.revertStyle();

    });

}

</script>

</body>

</html>

# 참고자료 리스트

- Map Control

<https://navermaps.github.io/maps.js/docs/tutorial-2-control-options.example.html>

- Marker

<https://navermaps.github.io/maps.js/docs/tutorial-1-marker-simple.example.html>

- Latitude & Longitude

<https://navermaps.github.io/maps.js/docs/tutorial-1-event-simple.example.html>

- My Location

<https://navermaps.github.io/maps.js/docs/tutorial-6-map-geolocation.example.html>

- Coordinate to Location

<https://navermaps.github.io/maps.js/docs/tutorial-3-geocoder-geocoding.example.html>

- DisasterMsg

<https://www.data.go.kr/data/15091495/openapi.do>

-Marker Cluster

<https://navermaps.github.io/maps.js/docs/tutorial-marker-cluster.example.html>