Final Project

2021년 12월 14일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

이름 이미리내

dkfs1324@office.uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

차 례

[1. 요약 1](#_Toc54003187)

[1.1. 왜 이 주제를 선택했는가? 1](#_Toc54003188)

[2. 요구사항 도출 1](#_Toc54003187)

[2.1. 초기 스케치 1](#_Toc54003188)

[2.2. 주요 요구사항 1](#_Toc54003188)

[3. 설계 1](#_Toc54003187)

[3.1. 사용 기술 1](#_Toc54003188)

[3.2. 설계 구조도 1](#_Toc54003188)

[4. 구현 코드 상세 설명 1](#_Toc54003187)

[4.1. OPEN API 내용 전송 받기 1](#_Toc54003188)

[4.2. 테이블 형성하기 1](#_Toc54003188)

[4.3. 지도 출력 및 Geojson으로 Outline 그리기 1](#_Toc54003188)

[4.4. 지도 부가 서비스\_크기 조정 도구 삽입하기 1](#_Toc54003188)

[4.5. 마커 삽입하기 1](#_Toc54003188)

[4.6. 마커 클릭 시, 그에 관한 정보 출력 1](#_Toc54003188)

[5. 결론 1](#_Toc54003187)

[5.1. 과제 구현 결과 1](#_Toc54003188)

[5.2. 문제점 고찰 1](#_Toc54003188)

[6. 토론 1](#_Toc54003187)

[6.1. 과제의 완성도 1](#_Toc54003188)

[6.2. 사용된 기술의 분석 1](#_Toc54003188)

[7. 부록 1](#_Toc54003187)

[7.1. 전체 소스코드 1](#_Toc54003188)

[7.2. 참고자료 출처 1](#_Toc54003188)

# 요약

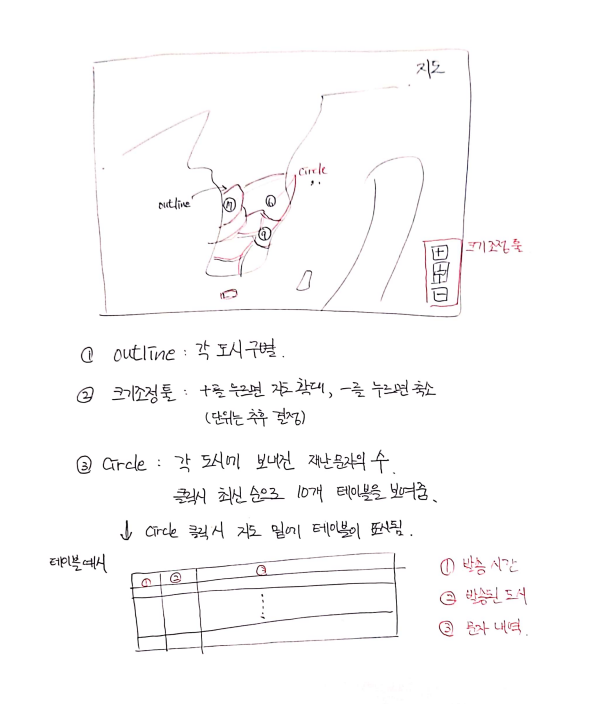
## 왜 이 주제를 선택했는가?

현재, 전 세계적으로 유행 중인 COVID-19으로 인해 우리는 일반적인 생활을 하던 과거보다는 훨씬 많은 재난문자를 받게 되었다. 그 전에는 태풍이나 홍수 등의 실질적인 재난과 관련된 문자를 받았기에 재난문자는 그다지 많이 오지 않았고, 우리의 일상 속에 스며들어 활성화되어 진지 그다지 오래되지 않았다. COVID-19의 유행으로 인해 현재 재난문자의 대부분은 COVID-19 확진자의 동선을 이야기하거나 혹은 그 전날의 확진자 수를 알려주고는 한다.

이를 보며 나는 현재 거주하는 서울뿐만 아니라, 내 부모님이나 혹은 다른 친구들이 살고 있는 지역의 재난 문자를 보고 그들이 혹시나 동선이라도 겹치지 않았나 확인할 수 있는 프로그램을 제작하고 싶었고, 이를 구상하게 되었다.

2. 요구사항 도출

2.1. 초기 스케치



2.2. 주요 요구사항

1) 지도 위에 빨간 선으로 Outline이 그려져 있을 것.

2) 지도 크기 조절 도구를 지도 상에 배치할 것.

3) 지도 위 시 혹은 도에 마커를 띄워 클릭 시 그 도시의 명을 정보 창으로 보여주고, 지도 밑으로 해당 도시에 간 문자를 테이블 형태로 최신순 10개 제공할 것.

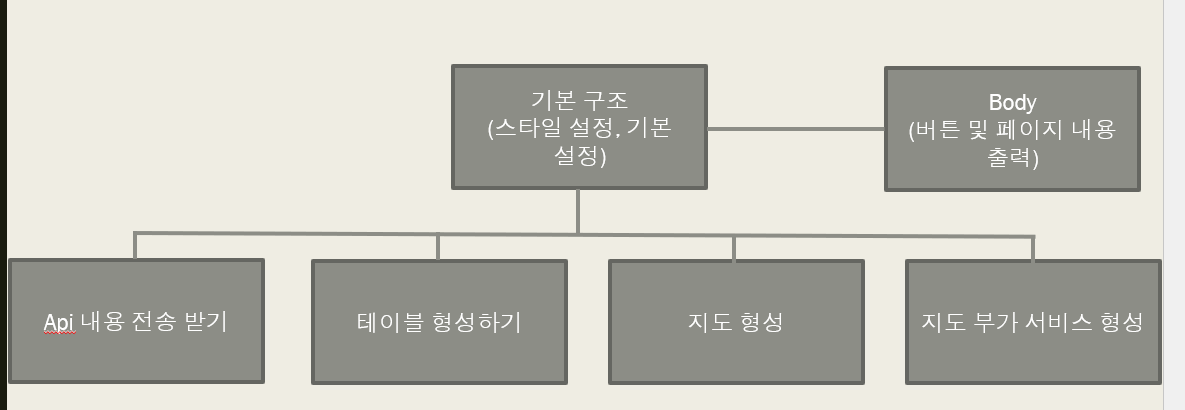
3. 설계

3.1. 사용 기술

-JavaScript?

: 자바 스크립트는 웹 [페이지](https://terms.naver.com/entry.nhn?docId=861087&ref=y)에서 [사용자](https://terms.naver.com/entry.nhn?docId=852026&ref=y)로부터 특정 [이벤트](https://terms.naver.com/entry.nhn?docId=850035&ref=y)나 입력 값을 받아 동적인 처리를 목적으로 고안된 객체 기반의 [스크립트](https://terms.naver.com/entry.nhn?docId=848218&ref=y) 프로그래밍 언어. 사용자 경험을 향상시키기 위한 방법으로 주로 사용된다. 일반적으로 HTML 문서에 내재되며, 브라우저에서 실행된다.

3.2. 설계 구조도



-프로그램의 구조도는 다음과 같다. 우선 첫번째로 HTML과 CSS로 기본 틀을 제작한다. 그 다음으로는 두 가지 부분으로 갈리는데, head 부분에 들어가는 4가지 파트, API 내용 받기, 테이블 형성, 지도 형성, 지도 부가 서비스 형성이 있다. 그리고 다음 부분은 body 부분에 들어가며 버튼 및 페이지 내용 출력으로 지도를 클릭 시, 표 출력이 어디 부분으로 될 지를 결정한다.

4. 구현 코드 상세 설명

4.1. OPEN API 내용 전송받기

텍스트이(가) 표시된 사진

자동 생성된 설명

-open api를 받기 위한 코드로, 모든 지역 별로 문자를 받기 위해 메소드 getProperty와 setProperty를 사용하여 함수를 구상하였다. 데이터 값을 설정하여 지역 당 10개의 문자를 받을 수 있도록 설정하였다.

4.2. 테이블 형성하기

텍스트이(가) 표시된 사진

자동 생성된 설명

-getElementById를 이용하여 id의 속성값을 문자열로 전달해 table을 출력할 수 있도록 하였고, ggetElementByTagname으로 표의 첫번째 가로축에 들어갈 분류 기준을 입력해줄 수 있게 되었다. 또한 이 모든 정보는 xml의 형태로 받게 되며, 조금의 로딩을 거쳐 출력하게 된다.

4.3. 지도 출력 및 Geojson으로 Outline 그리기

텍스트이(가) 표시된 사진

자동 생성된 설명

-네이버 클라우드 플랫폼으로 지도를 불러온 후, 센터를 설정하기 위해 스트링으로 center에 위도와 경도를 저장시켰다. 그리고 geojson으로 outline을 따기 위해 ajax로 함수를 설정하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

-addListener 메소드를 사용하여 지도를 클릭하였을 때와 안 하였을 때를 구분하여 효과가 일어나도록 했다. 마우스로 클릭 시 해당 지역이 붉게 변하고 마우스를 떼면 붉게 변한 것이 사라지도록 구상하였다.

4.4. 지도 부가 서비스\_크기 조정 도구 삽입하기

텍스트이(가) 표시된 사진

자동 생성된 설명

-map 변수를 사용하여 지도 생성 시 옵션을 따로 지정하였다. Zoom level은 13, 최소 Zoom level은 7, 컨트롤 표시 여부 또한 수락했다. setOption 메소드를 사용하여 옵션을 조정하였는데, zoom의 정도와 지도의 상호작용을 조절하였다. getOption 메소드를 이용하여 어떠한 경우에 해당 조건이 실행되는지를 설정하였다.

4.5. 마커 삽입하기

텍스트이(가) 표시된 사진

자동 생성된 설명

-makeMarker 메소드를 이용하여 총 16개의 지역을 표시하도록 지정하였으며, Marker 클래스에 저장된 스트링으로 해당 마커가 어떤 크기와 클리핑 위치, offset까지 설정하였다.

4.6. 마커 클릭 시, 그에 관한 정보 출력 텍스트이(가) 표시된 사진

자동 생성된 설명

-infowindow 위에 div로 해당 정보창에 들어갈 내용을 작성하고 addListner로 정보창이 열릴 때와 안 열리도록 할 때를 설정하였다.

5. 결론

5.1. 과제 구현 결과

지도이(가) 표시된 사진

자동 생성된 설명

-해당 사진과 같이 지도와 outline은 잘 표현되었으나, 해당 지역을 클릭하였을 때, 마커와 지도 크기 설정 도구가 화면에 표시되지 않았다. 하지만 지도를 클릭했을 때 표가 나올 수 있도록 설정해 놓았기 때문에 지도 또한 클릭하였으나 표가 출력되진 않았다.

5.2. 문제점 고찰

- 첫째, open api 정보를 불러오는 코드에 오류가 있었는지, 정보가 오지 않기 때문에 테이블을 생성하는 코드를 삽입하여 실행하여도 정보가 담긴 표가 작성되지 않는다.

둘째, 마커의 출력에도 문제가 있다. 마커의 이미지 파일과 함께 소스 파일을 같은 폴더에 저장하여도 마커가 출력되지 않았다. 이미지 파일의 이름까지 맞췄으니, 아마 마커 출력을 위한 코드에 문제가 있었던 듯하다.

셋째, 지도 설정 툴을 출력하는 코드를 삽입하였을 때, 마우스 휠로는 지도의 확대 정도가 조절되었으나, 도구가 화면상에 노출되지 않았다. 추가적인 함수가 필요한 듯싶은데 이를 더 고찰해야 한다.

6. 토론

6.1. 과제의 완성도

-사실 과제의 완성도는 30퍼센트 정도로 주요한 기능들이 작동하지 않았다. 분명 코드를 합치면서 어디선가 오류가 있거나 혹은 내가 중간에 실수해 빠뜨린 함수로 인해 돌아가지 않는 것일수도 있다.

6.2. 사용된 기술의 분석

-여기서 사용된 가장 많이 사용된 테크닉의 경우, addListener 메소드를 이용해 여러 조건을 만들어 작동할 수 있도록 하는 것이다. 대부분 두 가지, 마우스를 올리던가 내리던가 하는 조건으로 기능할 수 있도록 설정하였다.

7. 부록

7.1. 전체 소스코드

<!DOCTYPE html>

<html lang="en">

    <style>

        table,th,td {

          border : 1px solid black;

          border-collapse: collapse;

        }

        th,td {

          padding: 5px;

        }

        </style>

<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=08bksss6fl"></script>

    <script>

        function loadDoc() {

        var url = 'http://apis.data.go.kr/1741000/DisasterMsg3/getDisasterMsg1List'; /\*URL\*/

        var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'lbpH%2BY2gpZ0eZg3DeCGhk6AoNq99KDSuQBWEr5QwpxfsIVcrk3w4R53AP6t1nPhwogngpPDx5AbCbPxexpLf8A%3D%3D'; /\*Service Key\*/

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

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

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

        var xhttp = new XMLHttpRequest();

        xhttp.onreadystatechange = function () {

            if (this.readyState == 4 && this.status == 200) {

                myFunction(this);

            }

        };

        xhttp.open("GET", url + queryParams, true);

        xhttp.send();

        }

        function myFunction(xml) {

          var i;

          var xmlDoc = xml.responseXML;

          var table="<tr><th>지역</th><th>문자내용</th></tr>";

          var 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("tb").innerHTML = table;

        }

</script>

<script>

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

        center: new naver.maps.LatLng(37.3595704, 127.105399), //지도의 초기 중심 좌표

        zoom: 13, //지도의 초기 줌 레벨

        minZoom: 7, //지도의 최소 줌 레벨

        zoomControl: true, //줌 컨트롤의 표시 여부

        zoomControlOptions: { //줌 컨트롤의 옵션

            position: naver.maps.Position.TOP\_RIGHT

        }

    });

//setOptions 메서드를 이용해 옵션을 조정할 수도 있습니다.

map.setOptions("mapTypeControl", true); //지도 유형 컨트롤의 표시 여부

naver.maps.Event.addListener(map, 'zoom\_changed', function (zoom) {

    console.log('zoom:' + zoom);

});

// 지도 인터랙션 옵션

$("#interaction").on("click", function(e) {

    e.preventDefault();

    if (map.getOptions("draggable")) {

        map.setOptions({ //지도 인터랙션 끄기

            draggable: false,

            pinchZoom: false,

            scrollWheel: false,

            keyboardShortcuts: false,

            disableDoubleTapZoom: true,

            disableDoubleClickZoom: true,

            disableTwoFingerTapZoom: true

        });

        $(this).removeClass("control-on");

    } else {

        map.setOptions({ //지도 인터랙션 켜기

            draggable: true,

            pinchZoom: true,

            scrollWheel: true,

            keyboardShortcuts: true,

            disableDoubleTapZoom: false,

            disableDoubleClickZoom: false,

            disableTwoFingerTapZoom: false

        });

        $(this).addClass("control-on");

    }

});

// 관성 드래깅 옵션

$("#kinetic").on("click", function(e) {

    e.preventDefault();

    if (map.getOptions("disableKineticPan")) {

        map.setOptions("disableKineticPan", false); //관성 드래깅 켜기

        $(this).addClass("control-on");

    } else {

        map.setOptions("disableKineticPan", true); //관성 드래깅 끄기

        $(this).removeClass("control-on");

    }

});

// 타일 fadeIn 효과

$("#tile-transition").on("click", function(e) {

    e.preventDefault();

    if (map.getOptions("tileTransition")) {

        map.setOptions("tileTransition", false); //타일 fadeIn 효과 끄기

        $(this).removeClass("control-on");

    } else {

        map.setOptions("tileTransition", true); //타일 fadeIn 효과 켜기

        $(this).addClass("control-on");

    }

});

// min/max 줌 레벨

$("#min-max-zoom").on("click", function(e) {

    e.preventDefault();

    if (map.getOptions("minZoom") === 10) {

        map.setOptions({

            minZoom: 7,

            maxZoom: 21

        });

        $(this).val(this.name + ': 7 ~ 21');

    } else {

        map.setOptions({

            minZoom: 10,

            maxZoom: 21

        });

        $(this).val(this.name + ': 10 ~ 21');

    }

});

//지도 컨트롤

$("#controls").on("click", function(e) {

    e.preventDefault();

    if (map.getOptions("scaleControl")) {

        map.setOptions({ //모든 지도 컨트롤 숨기기

            scaleControl: false,

            logoControl: false,

            mapDataControl: false,

            zoomControl: false,

            mapTypeControl: false

        });

        $(this).removeClass('control-on');

    } else {

        map.setOptions({ //모든 지도 컨트롤 보이기

            scaleControl: true,

            logoControl: true,

            mapDataControl: true,

            zoomControl: true,

            mapTypeControl: true

        });

        $(this).addClass('control-on');

    }

});

$("#interaction, #tile-transition, #controls").addClass("control-on");

</script>

<script>

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

var map = new naver.maps.Map('map', {

    center: new naver.maps.LatLng(37.3595704, 127.105399),

    zoom: 15

});

makeMarker(map, new naver.maps.LatLng(37.3595704, 127.105399), 0);

makeMarker(map, new naver.maps.LatLng(37.4018025, 127.1153248), 1);

makeMarker(map, new naver.maps.LatLng(37.2334567, 128.0983706), 2);

makeMarker(map, new naver.maps.LatLng(37.9765432, 128.0989846), 3);

makeMarker(map, new naver.maps.LatLng(37.5446778, 129.0959706), 4);

makeMarker(map, new naver.maps.LatLng(37.9067436, 129.0989823), 5);

makeMarker(map, new naver.maps.LatLng(37.7654394, 129.0824332), 6);

makeMarker(map, new naver.maps.LatLng(37.7894376, 129.0554346), 7);

makeMarker(map, new naver.maps.LatLng(37.2345877, 128.0565842), 8);

makeMarker(map, new naver.maps.LatLng(37.8094376, 127.0098765), 9);

makeMarker(map, new naver.maps.LatLng(37.6723980, 127.0438734), 10);

makeMarker(map, new naver.maps.LatLng(37.3234769, 128.0090998), 11);

makeMarker(map, new naver.maps.LatLng(37.3234598, 128.0547674), 12);

makeMarker(map, new naver.maps.LatLng(37.7562367, 130.0785636), 13);

makeMarker(map, new naver.maps.LatLng(37.7843678, 131.0867356), 14);

makeMarker(map, new naver.maps.LatLng(37.2434650, 130.0095435), 15);

function makeMarker(map, position, index) {

    var ICON\_GAP = 31;

    var ICON\_SPRITE\_IMAGE\_URL = HOME\_PATH +'/img/example/sp\_pin\_hd.png';

    var iconSpritePositionX = (index \* ICON\_GAP) + 1;

    var iconSpritePositionY = 1;

    var marker = new naver.maps.Marker({

        map: map,

        position: position,

        icon: {

          url: ICON\_SPRITE\_IMAGE\_URL,

          size: new naver.maps.Size(26, 36), // 이미지 크기

          origin: new naver.maps.Point(iconSpritePositionX, iconSpritePositionY), // 스프라이트 이미지에서 클리핑 위치

          anchor: new naver.maps.Point(13, 36), // 지도상 위치에서 이미지 위치의 offset

          scaledSize: new naver.maps.Size(395, 79)

        }

    });

    return marker;

}

</script>

</head>

<body>

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

<div id="table"></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: 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.

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

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

        } else {

            clear\_table();

            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>

<table id="tb"></table>

</body>

</html>

7.2. 참고자료 출처

-https://navermaps.github.io/maps.js.en/

(네이버 클라우드 플랫폼 활용 예시를 참고하였음.)