최종 보고서

2021년 12월 13일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

오창윤

ogaoga5959@office.uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

차 례

[1. 요약](#_Toc54003187)

2. 요구사항 도출

3. 설계

4. 구현

5. 토론

6. 결론

7. 부록

8. 참고자료

# 요약

공공 데이터 포털의 지역별/종합 재난 문자 데이터를 이용하여 재난문자를 지역별로 분류하여 제공하고 위도, 경도를 입력받아 맵에 검색해 보는 기능을 추가하였습니다. 또한, 숫자를 입력 받아서 몇개 재난 문자를 조회할 수 있는지 알 수 있게 하였습니다.

웹사이트를 구현하기 위해서 javaScript, html, css를 사용하였습니다.

# 요구사항 도출

## 문제점

다양한 방식의 입력을 받아들이지 못하는 점이 불편함.   
사용자가 몇개 재난 문자를 조회할 수 있는 지 알 수 없음.

## 해결책 제시

지도를 클릭하여 표현하는 방식 외에 원하는 조회 개수를 입력 받아 맵에 검색해보는 기능을 추가하여 입력의 다양성을 확보함.

# 설계

## 사용 기술

지도이(가) 표시된 사진

자동 생성된 설명

입력의 다양성을 높이려고 하였다.

첫번째로는 지도를 클릭하여 해당 지역의 안전문자를 보여주게 하였다.

두번째로는 숫자를 입력받아서 몇개의 재난 문자를 조회할 수 있는지 알 수 있게 하였다.

## 사용자 입력에 따른 흐름도

사용자 입력 - > 지도클릭 -> 재난문자 표시 -> 다른 지역 클릭 -> 해당 지역의 재난문자 표시

# 구현

## 코드의 주요 구조

<style>

table,th,td {

border : 1px solid black;

border-collapse: collapse;

}

th,td {

padding: 5px;

}

</style>

Css로 기본적인 스타일을 잡아 주었다.

<form>

<table>

<tr>

<td>검색을 원하는 지역을 입력해 주세요 : </td>

<td colspan="2"><input type="text" id="location"></td>

<td><input type="button" value="검색" onclick="find()"></td>

<td id='cnt'></td>

</tr>

</table>

</form>

Form을 활용하여 table을 만들고 원하는 지역을 입력 받을 수 있도록 하였다.

var xhr = new XMLHttpRequest();

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

var queryParams = '?' + encodeURIComponent('serviceKey') + '='+'vgX4Mma%2BzsYIf0czI7rGsQF4dHCEkPRl5SKgVnaxikYp3iT2liuJp1ND4a3zRQrnrliriPbxOFifrU7RW8BIXw%3D%3D';

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');

url, 서비스 키를 받고, query를 생성하였다.

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

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)

});

GeoJSON을 통해 지도를 가져왔다.

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)

});

}

});

지도가 로드 된 다음 일회성으로 시도되는 이벤트이다.

function myFunction(xml) {

var xmlDoc = xml.responseXML; //xml DOM 객체생성

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

let table="<tr><th>지역</th><th>메시지 내용</th></tr>";

for (let 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("table").innerHTML = table;

}

함수를 통해 지역 이름과 메시지가 나타나게끔 설계하였다.

# 토론

## 도출한 요구사항의 완성도

지도를 클릭할 경우 해당 지역의 재난 문자 정보가 나타나게 설계 되었으며, 위도와 경도를 사용자가 입력할 수 있는 form도 만들었다.

또한, 숫자를 입력받아서 몇개의 재난 문자 조회 할 수 있는 지 알 수 있게 하였다.

## 선택할 수 있는 다양한 기술 분석

Naver map api위에 해당 지역을 다양한 색상으로 표현하면 더욱 가시성이 좋은 웹 페이지가 될 수 있을 것이다.

# 결론

## 과제 구현 결과

과제 구현 결과 입력 폼 받기, 지역클릭 시 해당 지역의 재난관련 문자를 표시, 현위치를 표시하려 했으나, 현위치를 표시하지 못하였다.

## 향후 방향

현위치를 입력받아 해당 지역의 재난문자를 표시하고 싶었지만 잘 되지 않았다. 그래도 그 과정에서 상당부분 많이 알게 되었고, 시험기간 이후에 향후 과제로 현위치를 입력받아 해당지역의 재난 문자를 표시하는 것이 목표이다.

# 부록

## jQuery

jQuery(제이쿼리)는 HTML의 클라이언트 사이드 조작을 단순화 하도록 설계된 크로스 플랫폼의 자바스크립트 라이브러리입니다. jQuery는 AJAX의 사용을 단순화 하기 위한 메서드들을 제공합니다.

## AJAX

.AJAX는 A synchronous Javascript And XML의 약자입니다. 웹페이지에서 이벤트가 발생하면 자바스크립트에 의해 XMLHttpRequest 객체가 생성되고 그 객체는 웹서버에 요청을 보냅니다. 서버는 요청을 처리하고 웹페이지에 응답을 보내며 응답은 자바스크립트에 의해 읽어집니다.

## JSON

JSON은 JavaScript Object Notation의 약자로, 속성-값 쌍( attribute–value pairs and array data types (or any other serializable value)) 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 텍스트를 사용하는 개방형 표준 포맷입니다. 메소드를 사용하여 객체를 스트링으로, 스트링을 객체로 변환합니다.

## GeoJSON

GeoJSON은 JSON을 기반으로 하는 지리 공간 데이터 교환 형식입니다. GeoJSON은 여러 타입의 JSON 객체를 정의합니다. 그리고 그 바식은 지리적 특징에 대한 data를 표현하기위해 결합되어있습니다. 주요 Geometries 형태는 Point, LineString(Poly),Polygon,MultiPoint,MultiLineString, MultiPolygon등이 있습니다.

## 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>간단한 지도 표시하기</title>

<script type="text/javascript"

src="https://openapi.map.naver.com/openapi/v3/maps.js?ncpClientId=ms21qfhkka"></script>

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

<style>

table#table,

#table th,

#table td {

border: 1px solid black;

border-collapse: collapse;

}

th,

td {

padding: 5px;

}

</style>

</head>

<body>

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

<form>

<table>

<tr>

<td>원하는 조회개수를 입력하시오(전국에서 조회됩니다.) : </td>

<td colspan="2"><input type="text" id="Count"></td>

<td><input type="button" value="검색" onclick="find()"></td>

</tr>

</table>

</form>

<table id='table'></table>

<script>

var xhr = new XMLHttpRequest();

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

//지도 가져오는 부분

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 로드된다음 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;

}

$.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) {

//아니면 이거쓴다.

var styleOptions = {

fillColor: '#ff0000',

fillOpacity: 0.0001,

strokeColor: '#ff0000',

strokeWeight: 2,

strokeOpacity: 0.4

};

if (feature.getProperty('focus')) { //포커스이면 이걸로 바꾼다.

styleOptions.fillOpacity = 0.6;

styleOptions.fillColor = '#0f0';

styleOptions.strokeColor = '#0f0';

styleOptions.strokeWeight = 4;

styleOptions.strokeOpacity = 1;

}

return styleOptions;

});

regionGeoJson.forEach(function (geojson) {

map.data.addGeoJson(geojson);

});

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

var feature = e.feature;

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

//누르면 table에 table 만들려는 코드 전에 띄우기 부터하자.

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

queryParams = queryByClick(feature);

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;

// }

myFunction(this);

};

xhr.send('');

map.data.features.forEach(

function (feature) {

feature.setProperty('focus', false);

}

);

build\_table(feature.getProperty('area1'));

feature.setProperty('focus', true);

}

else {

clear\_table();

feature.setProperty('focus', false);

}

});

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

var feature = e.feature,

regionName = feature.getProperty('area1');

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) {

tooltip.hide().empty();

map.data.revertStyle();

});

} //startDataLayer함수 끝나는곳

function find() {

var num = document.getElementById('Count').value;

var queryParams = queryBynum(num);

var xhr = new XMLHttpRequest();

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

xhr.onreadystatechange = function () {

myFunction(this);

};

xhr.send('');

document.getElementById('Count').value = '';

}

function queryByClick(f) {

var queryParams = '?' + encodeURIComponent('serviceKey') + '=' + 'vgX4Mma%2BzsYIf0czI7rGsQF4dHCEkPRl5SKgVnaxikYp3iT2liuJp1ND4a3zRQrnrliriPbxOFifrU7RW8BIXw%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(f.getProperty('area1')); /\*\*/

return queryParams;

}

function queryBynum(num) {

var queryParams = '?' + encodeURIComponent('serviceKey') + '=' + 'vgX4Mma%2BzsYIf0czI7rGsQF4dHCEkPRl5SKgVnaxikYp3iT2liuJp1ND4a3zRQrnrliriPbxOFifrU7RW8BIXw%3D%3D'; /\*Service Key\*/

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

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

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

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

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

return queryParams;

}

function myFunction(xml) {

var xmlDoc = xml.responseXML; //xml DOM 객체생성

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

let table = "<tr><th>지역</th><th>메시지 내용</th></tr>";

for (let 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("table").innerHTML = table;

}

</script>

</body>

</html>

# 참고자료

Geocoder

<https://navermaps.github.io/maps.js.ncp/docs/tutorial-Geocoder-Geocoding.html>

Naver Maps API

<https://navermaps.github.io/maps.js.ncp/docs/index.html>

네이버 예제사이트

https://navermaps.github.io/maps.js.ncp/docs/tutorial-digest.example.html