소프트웨어시스템실습

최종 프로젝트

2021년 12월 14일

2021년 2학기 소프트웨어시스템실습

담당교수: 최성종

고유정

kokoo99@office.uos.ac.kr

서울시립대학교 전자전기컴퓨터공학부

차 례

[1. 요약 1](#_Toc54003187)

[2. 요구사항 도출 2](#_Toc54003187)

[2.1. 스케치&사용시나리오 2](#_Toc54003188)

[2.2. 주요 요구사항(기능/비기능) 2](#_Toc54003188)

[3. 설계 4](#_Toc54003187)

[3.1. 사용 기술 4](#_Toc54003188)

[3.2. 소프트웨어 구조도&사용자 입력에 따른 흐름도 5](#_Toc54003188)

[4. 구현 7](#_Toc54003187)

[4.1. 함수 설명 1](#_Toc54003188)

[4.2. 필수 라이브러리 1](#_Toc54003188)

[5. 토론 1](#_Toc54003187)

[5.1. 완성도 분석 1](#_Toc54003188)

[6. 결론 1](#_Toc54003187)

[6.1. 과제 구현 결과&향후방향 1](#_Toc54003188)

[7. 부록 1](#_Toc54003187)

[7.1. 전체 소스코드 1](#_Toc54003188)

[7.2. 출처 1](#_Toc54003188)

# 요약

네이버에서 지도를 불러오고, 공공데이터 포털에서 행정안전부\_재난문자방송 발령현황을 조회하여, 지도의 광역자치단체 영역을 클릭 시 해당 지역에 대한 문자 발령 현황을 테이블로 표시한다.

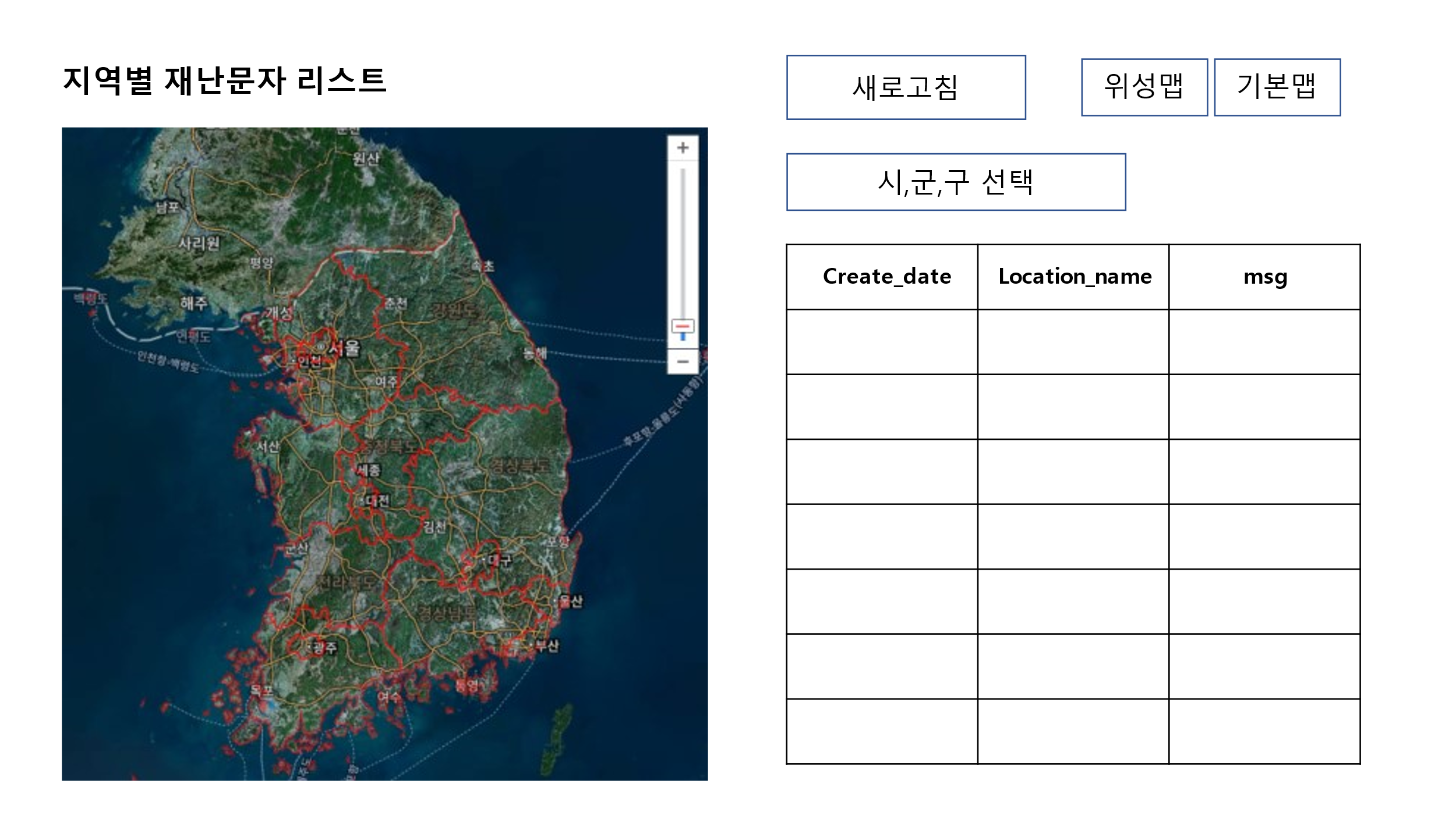
지도 유형을 인공위성 맵과 기본 맵 중 선택 가능하도록 한다.

광역자치단체가 선택된 경우, 해당 자치 단체에 속한 시, 군 선택상자를 제공하여, 사용자가 선택 시 해당 시/군에 발령된 재난 문자만을 보여준다.

새로고침 버튼을 만들어 누르면 새로운 정보를 알아오도록 한다.

# 요구사항 도출

## 스케치&사용시나리오



* 사용 시나리오

1. 지도 유형을 인공위성 맵, 기본 맵 중 선택 가능하도록 하기.
2. Css를 이용하여 table 꾸미기
3. 해당 자치단체에 속한 시, 군 선택상자를 제공해 원하는 지역의 정보를 볼 수 있도록 한다.

## 주요 요구사항(기능/비기능)

사용자에게 제공하는 화면 구성 및 인터페이스에 대해 설명한다.

화면 구성은 다음과 같다.

화면 상단에 네이버 지도를 보여준다.

그 아래, 사용자에게 제시하는 메시지를 보여준다.

그 아래, 사용자가 선택한 광역자치단체에 소속된 시/군 선택상자를 보여준다.

그 아래, 현재 선택한 영역에 발령된 재난 메시지를 테이블로 보여준다.

사용자가 지도상에서 마우스를 움직이면, 해당 지역의 지역명을 툴팁으로 보여준다.

사용자가 지도상에서 마우스를 클릭하면, 해당 지역에 발령된 재난 문자를 보여준다.

광역단체가 선택된 경우, 해당 광역단체에 속한 시/군을 선택할 수 있는 콤보박스가 주어진다.

콤보박스의 내용은 광역단체의 선택이 변경될 때 마다 이에 맞게 다르게 채워진다.

# 설계

## 사용 기술

아래 형식으로 HTML을 작성한다.

|  |
| --- |
| <tag id="id1" class="class1" ....>  <button id=buttonRetrieve class=buttonClass1 type="button" title="공공 데이터 포탈에서 발령현황을 조회합니다.">재난문자방송 발령현황 조회</button> |

위의 경우, tag는 button, id는 buttonRetrieve, class는 buttonClass1으로 지정한 경우이다. 이 값들은 <style>태그내 작성하는 css에서 <script>태그내 작성하는 javascript에서 HTML 요소를 지정하는데 사용된다.

### CSS에서 html 요소 지정하기

|  |
| --- |
| #divMessage {  font-size : 24px ;  color : blue ;  } |

단어앞에 #을 붙이면 id를 나타낸다.

id를 divMessage로 지정한 HTML요소들의 폰트크기, 색상을 위와 같이 적용한다.

|  |
| --- |
| .buttonClass1 {  background-color: #4CAF50; /\* Green \*/  ...  } |

단어 앞에 .를 붙이면 클래스를 나타낸다.

class에 buttonClass1이 포함된 HTML 요소에 대해 위 스타일을 적용한다.

단어앞에 아무 글자도 없으면 태그를 나타낸다.

|  |
| --- |
| th,td {  padding: 5px;  } |

페이지내 모든 th, td 태그의 패딩속성을 5px로 적용한다.

### javascript에서 html 요소 지정하기

브라우저 자체가 인식하는 자바스크립트만으로 프로그래밍하는 것을 보완하여, jQuery를 사용하였다.

jQuery는 자바스크립트 라이브러리로 HTML 요소를 선택하거나 이벤트 핸들러를 지정하는 손쉬운 함수들을 제공한다.

jQuery를 사용하면 css 지정과 동일하게 tag, #id, .class의 형식으로 원하는 html요소들을 지정할 수 있다.

## 소프트웨어 구조도&사용자 입력에 따른 흐름도

### 프로그램 실행 요약

네이버 지도를 로드 한다.

지도가 로드 되면 선택영역에 사용할 17개 geojson을 다운로드한다.

17개 geojson이 전부 다운로드 되면, 지도에서 해당 영역이 선택 가능하게 만들고, 재난문자를 추출한다.

준비상태가 되었다고 사용자에게 알림 메시지를 보여준다.

사용자가 지도에서 특정영역을 클릭하면, 추출한 재난문자 데이터 중 해당 영역과 관련된 데이터를 찾는다.

찾은 메시지의 시/군 영역으로 시/군 선택상자를 채운다.

메시지 테이블에 찾을 메시지를 보여준다.

사용자가 시/군 선택상자에서 특정 시/군을 선택하면, 현재 테이블의 row중 해당 시/군과 관련된 줄을 보이고, 관련 없는 줄을 감춘다.

### 사용자 입력에 따른 흐름도

## 텍스트이(가) 표시된 사진 자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

# 구현

## 함수 설명

* **startDataLayer**
* 받아온 GeoJSON 파일을 map.data에 넣음
* addListener를 통해

‘click’되면 onClickedDo 실행

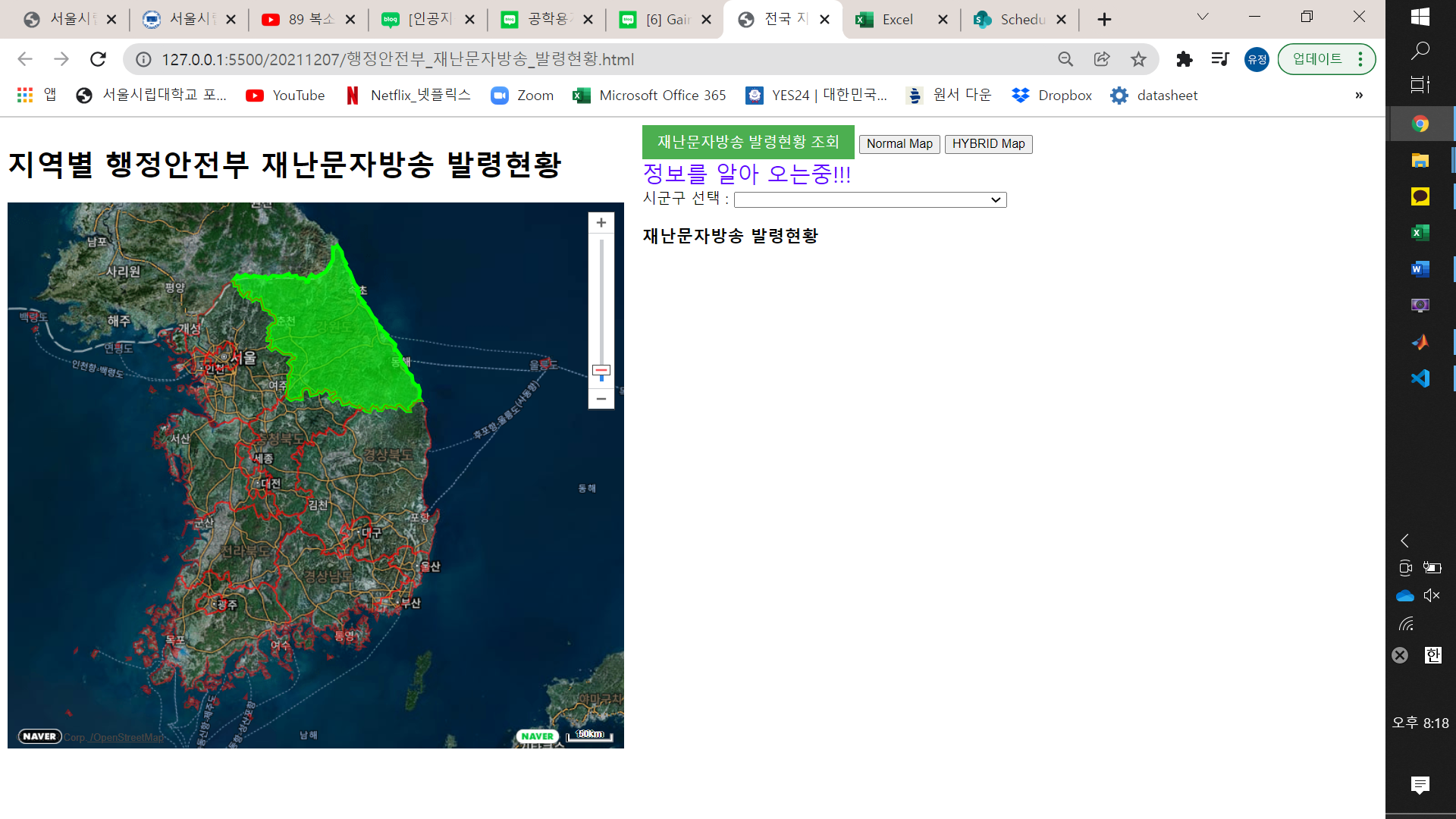
‘mouseover’되면 feature 변수에 area와 msg를 받아온 후 출력

‘mouseout’되면 툴팁을 hide(사라지게)하고, revertStyle을 통해 스타일 원상복귀 시키기.

* **printMessage**
* 지역 명 나타내기
* **printClickedInfo**
* 클릭한 지역의 정보를 테이블 형태로 보여주기
* **onClickedArea**
* 지역이 클릭되었을 때 불리는 함수
* 만약 focus가 true가 아니면 이전에 선택되지 않았던 지역. 따라서 aDo에 그 지역 입력. Focus를 true로 바꾸고 printMsg, loadDoc을 이용해 table 출력
* 만약 focus가 true였다면 이전에 선택된 지역. 따라서 focus를 꺼줌.
* **NaverMapLoaded**
* 정보를 한번에 알아온다.
* **NaverMap**
* Map type을 정하는 함수.
* retrieveMessage
* 데이터 포털에서 재난문자 데이터를 가져온다.

# 토론

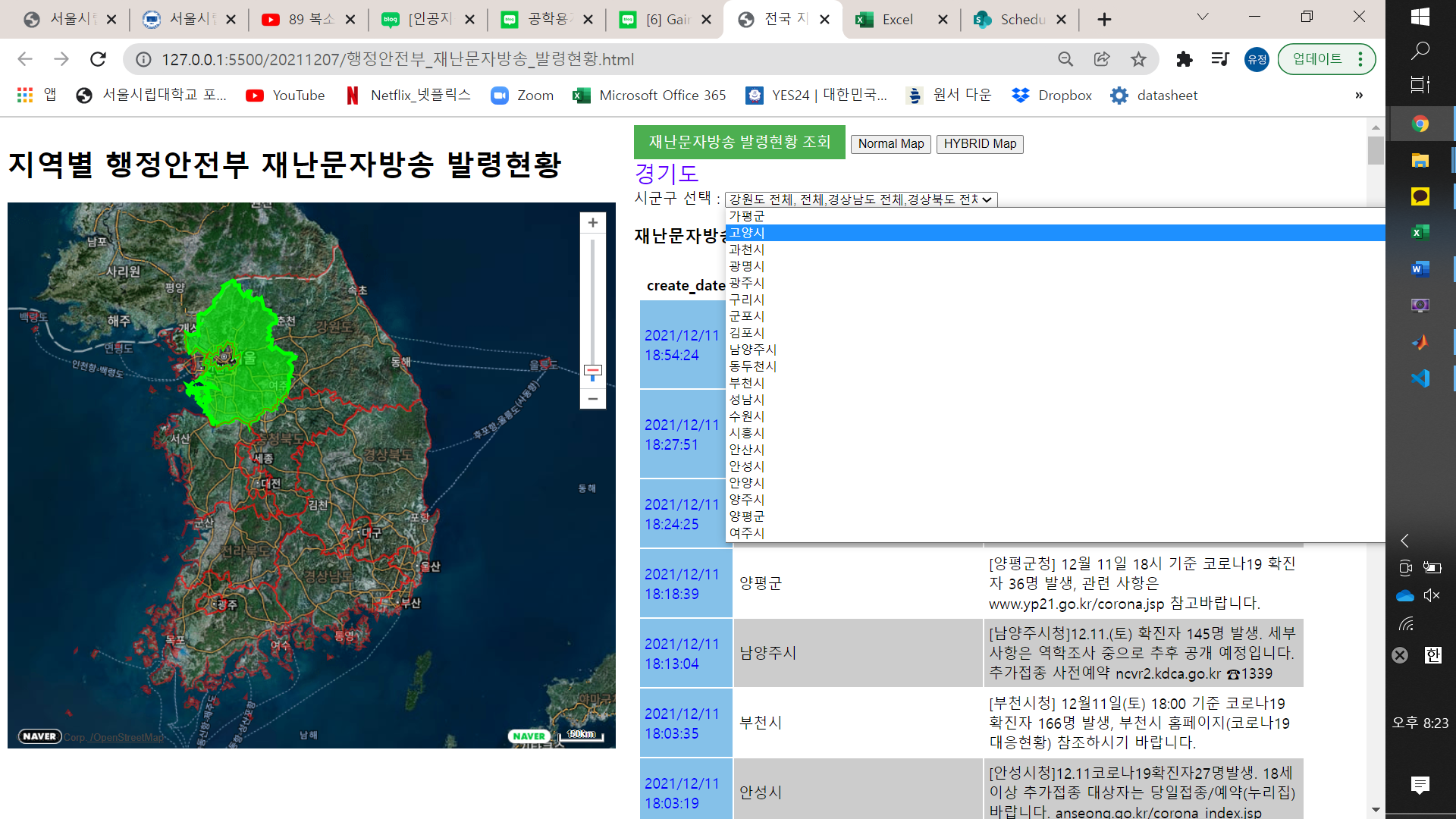
## 완성도 & 기술 분석



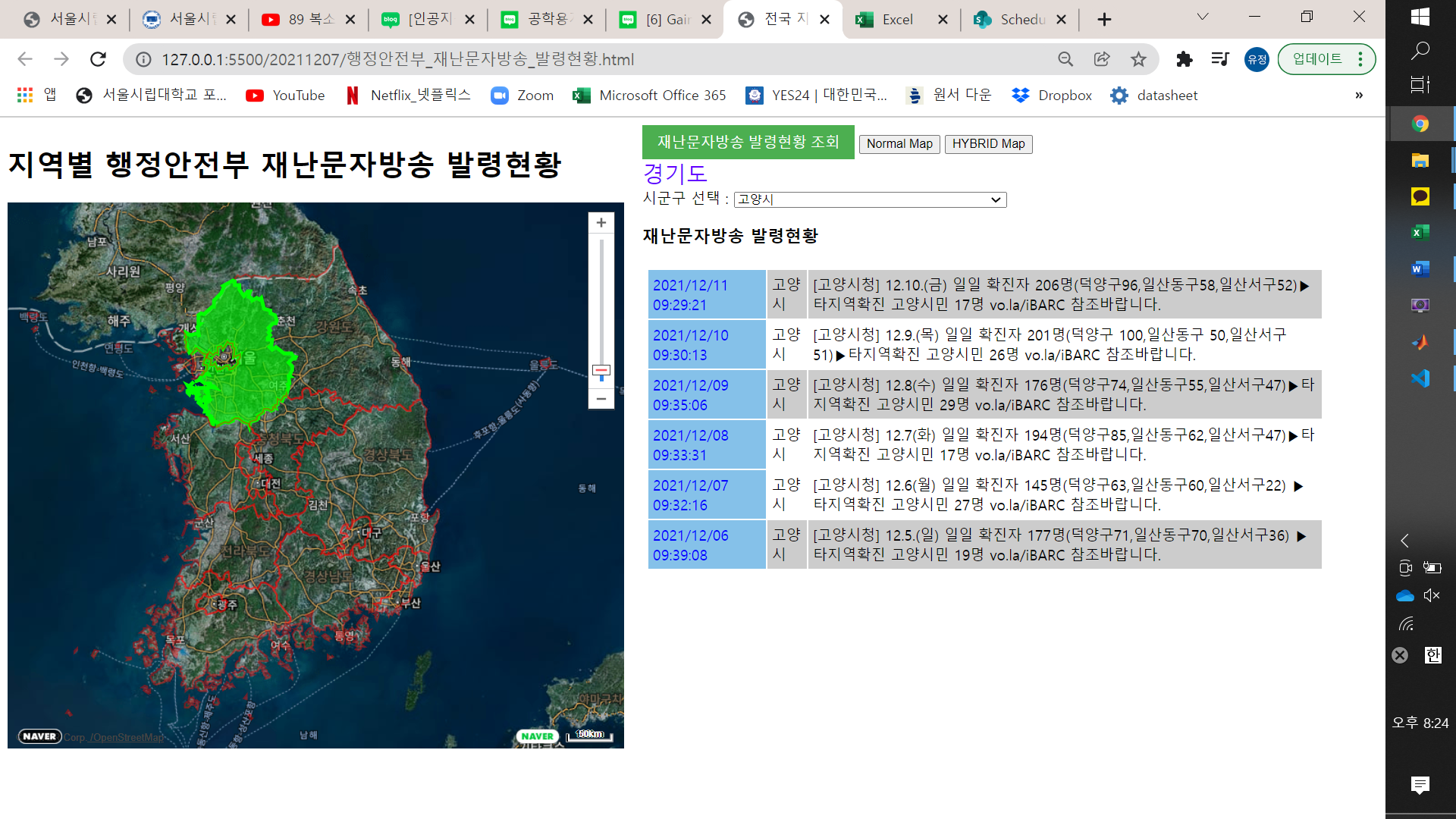
위의 그림은 실제 html파일을 실행시켰을 때 나오는 화면이다.

Normal Map / Hybrid Map 버튼을 통해 기본 지도와 인공 위성 지도 중 선택해 화면에 나타낼 수 있다.

또한 연두색 버튼(재난문자발송 발령현황 조회) 버튼을 통해 새로고침도 실제로 구현할 수 있었다.



지역을 선택하면 다음과 같이 create\_date와 location\_name, msg가 함께 테이블 형태로 출력된다. 그리고 선택박스를 통해 시군구를 선택 가능하다.



실제로 경기도 → 고양시를 선택했을 때의 화면이다. 고양시에서 발령한 문자를 모아서 보여주고 있다.

# 결론

## 과제 구현 결과&향후방향

5번의 사진에서 볼 수 있듯이, 처음 계획했던 대로 모든 요구사항을 충족시킬 수 있었다.

이 프로그램을 더욱 발전시키기 위해 다음의 기능들을 추가하면 좋을 것 같다.

1. 선택 영역으로 확대하기.
2. 더 세부 지역으로 나누기
3. 검색어로 메시지 찾는 기능 추가하기

# 부록

## 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>전국 지역별 행정안전부 재난문자방송 발령현황</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=6hw7de51b7"></script>      </head>      <style>          /\* 타이틀을 보여주는 class \*/          #divMessage {              font-size: 24px;              color: rgb(98, 0, 255);          }          .buttonClass1 {              background-color: #4CAF50;              /\* Green \*/              border: none;              color: white;              padding: 8px 16px;              text-align: center;              text-decoration: none;              display: inline-block;              font-size: 16px;              cursor: pointer;          };          .buttonClass2 {              background-color: #AF504C;              /\* Green \*/              border: none;              color: white;              padding: 8px 16px;              text-align: center;              text-decoration: none;              display: inline-block;              font-size: 16px;              cursor: pointer;          };          /\* css 태그 \*/          table,          td,          th {              border: 1px solid black;              border-collapse: collapse;          }          td,          th {              padding: 5px;          }          select {              width: 300px;          }          /\* 짝수행 \*/          tr:nth-child(even) {              background: #CCC;          }          /\* 홀수행 \*/          tr:nth-child(odd) {              background: #FFF;          }          /\* table 첫번재 컬럼 \*/          td:first-child {              background: #85C1E9;              color: blue;          };          /\* table 첫줄 제목 \*/          tr:first-child {              background: #F8C471;          };      </style>      <body>          <div style="width:45%;float:left;height:100%;margin-right:20px;">              <h1>지역별 행정안전부 재난문자방송 발령현황</h1>              <!-- 네이버 지도 영역 -->              <div id="map" style="width:100%;height:600px;"></div>          </div>          <div style="width:50%;float:left;height:100%;">              <div>                  <button                      id="buttonRetrieve"                      class="buttonClass1"                      type="button"                      title="공공 데이타 포탈애서 발령현황을 조회합니다.">재난문자방송 발령현황 조회</button>                  <button                      id="buttonNormapMap"                      class="buttonClass2"                      type="button"                      title="Normap Map으로 변경합니다.">Normal Map</button>                  <button                      id="buttonHybridMap"                      class="buttonClass2"                      type="button"                      title="HYBRID Map으로 변경합니다.">HYBRID Map</button>              </div>              <div id="divMessage">여기에 클릭된 지역의 정보가 표시됩니다.</div>              <div>                  <label>시군구 선택 :                  </label>                  <select id="selectCity">                      <option value="도시1">도시1</option>                      <option value="도시2">도시2</option>                  </select>              </div>              <h3>재난문자방송 발령현황</h3>              <div style='padding:5px;'>                  <table id="disaster"></table>              </div>          </div>          <script>              var regionGeoJson = [];              var map = naverMap(1);              function naverMap(mapType) {                  if (mapType == 0) {                      return new naver                          .maps                          .Map(document.getElementById('map'), {                              // zoom: 7, center: new naver.maps.LatLng(36.4203004, 128.317960),                              zoom: 5,                              center: new naver                                  .maps                                  .LatLng(37.5666805, 126.9784147),                              mapTypeId: 'normal'                          })                  }                  if (mapType == 1) {                      return new naver                          .maps                          .Map('map', {                              zoom: 5,                              center: new naver                                  .maps                                  .LatLng(37.5666805, 126.9784147),                              mapTypeId: naver.maps.MapTypeId.HYBRID,                              minZoom: 6,                              zoomControl: true,                              zoomControlOptions: {                                  position: naver.maps.Position.TOP\_RIGHT                              },                              mapDataControl: false,                              logoControlOptions: {                                  position: naver.maps.Position.LEFT\_BOTTOM                              },                              disableKineticPan: false                          })                  }              }              //jQuery를 이용, 툴팁 html요소를 만들고, map에 추가              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);              // 지도가 로드된 다음에 일회성으로 실행되는 이벤트              naver                  .maps                  .Event                  .once(map, 'init\_stylemap', function () {                      naverMapLoaded(); // 선택영역 다운로드 및 핸들러 설정                      retrieveMessage(); // 재난문자 현황을 조회                  });              function naverMapLoaded() {                  {                      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);                  }                  var HOME\_PATH = "https://navermaps.github.io/maps.js/docs";                  var urlPrefix = HOME\_PATH + '/data/region';                  var urlSuffix = '.json';                  var loadCount = 0;                  // 17개 영역의 지도 다운을 호출                  for (var i = 1; i < 18; i++) {                      var keyword = i + '';                      if (keyword.length === 1)                          keyword = '0' + keyword;                        // 01, 02, 03 생성                      var s = urlPrefix + keyword + urlSuffix;                      // if ( i==1 )  alert(s) ;                      $.ajax({                          url: urlPrefix + keyword + urlSuffix,                          success: function (idx) {                              return function (geojson) {                                  // 각 영역이 다운완료되었을 때 마다 호출된다.                                  regionGeoJson[idx] = geojson;                                  loadCount++; // 영역이 다운될 때 마다 증가, 17개 다 받아오면 그리기 시작                                  if (loadCount === 17)                                      startDataLayer();                                  }                              }(i - 1)                      });                  }              }              // callback : 지도 영역 17개가 전부 다운되었을 때 호출된다.              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;                      });                  regionGeoJson.forEach(function (geojson) {                      map                          .data                          .addGeoJson(geojson);                  });                  // 지도 영역이 클릭되었을 때 실행할 함수를 등록                  map                      .data                      .addListener('click', function (e) {                          onClickedArea(e);                      });                  // 지도 영역에 마우스 커서가 위치했을 때 실행할 함수를 등록                  map                      .data                      .addListener('mouseover', function (e) {                          // 이름을 받아                          var feature = e.feature,                              regionName = feature.getProperty('area1');                          regionMSG = feature.getProperty('msg1');                          // 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                              });                      });                  // 지도 영역에서 마우스가 빠져 나갈때 실행할 함수를 등록                  map                      .data                      .addListener('mouseout', function (e) {                          tooltip                              .hide()                              .empty(); // 툴팁 사라지게                          map                              .data                              .revertStyle(); // 스타일 원상복귀                      });              }          </script>          <script>              var areaPreviousClicked = ''; // 이전에 클릭한 지역              function printMessage(s) {                  document                      .getElementById("divMessage")                      .innerHTML = s;              }              // 클릭한 지역의 정보를 테이블 형태로 보여준다.              function printClickedInfo(s) {                  document                      .getElementById("disaster")                      .innerHTML = s;              }              //지역이 클릭되었을 때 불리는 함수              function onClickedArea(e) {                  var feature = e.feature;                  if (feature.getProperty('focus') !== true) {                      var aDo = feature.getProperty('area1');                      feature.setProperty('focus', true);                      printMessage(aDo);                      retrieveMessage(aDo);                      // 이전에 선택한 지역이 있었으면 focus를 끈다.                      if (areaPreviousClicked) {                          areaPreviousClicked.setProperty('focus', false);                          areaPreviousClicked = null;                      }                      areaPreviousClicked = feature;                  } else {                      feature.setProperty('focus', false);                  }              }              var allMessages = [];              function retrieveMessage(area1) {                  if (area1) {                      processMessage(area1);                      return;                  }                  printMessage('정보를 알아 오는중!!!');                  $('#selectCity').html(''); // 도시 콤보박스를 clear                  // 처음 모든 지역에 대한 정보를 알아온다.                  var numOfRows = '100'; // 가져올 갯수(전체 개수만큼 반복해서 가져와야 할 듯)                  numOfRows = '1000'; // 단순히 한꺼번에 적당히 많이 가져온다.                  var xhr = new XMLHttpRequest();                  var url = 'http://apis.data.go.kr/1741000/DisasterMsg3/getDisasterMsg1List';/\*URL\*/                  var pm = '?' + encodeURIComponent('serviceKey') + '=nzYSW6L2EXIDbdSRn%2Fn%2F1YF' +                          'kHomGLYQB8CgUlbyV1N0E3%2F9AqbBgz6Os3rxQB7AUQiKf%2B8F9mcx5jzIQZjoJSQ%3D%3D';/\*Service Key\*/                  pm += '&' + encodeURIComponent('pageNo') + '=' + encodeURIComponent('1');/\*\*/                  pm += '&' + encodeURIComponent('numOfRows') + '=' +                          encodeURIComponent(numOfRows);/\*\*/                  pm += '&' + encodeURIComponent('type') + '=' + encodeURIComponent('xml');/\*\*/                  xhr.onload = function () {                      var s = '정보를 알아 왔습니다.!!!<br>지도에서 원하는 지역을 클릭하면, 해당 지역에 대한 정보를 보입니다.';                      printMessage('원하는 지역을 클릭하세요.');                      processMessage(this);                      alert(s);                  }                  xhr.open('GET', url + pm);                  xhr.send('');                  printClickedInfo(''); // 이전 정보를 지운다.              }              function nodeValue(o, key) {                  return o                      .getElementsByTagName(key)[0]                      .childNodes[0]                      .nodeValue;              }              function processMessage(xmlOrArea) {                  let s = "<tr>",                      o;                  // s += "<th>md101\_sn</th>";                  s += "<th>create\_date</th>";                  s += "<th>location\_name</th>";                  s += "<th>msg</th>";                  // s += "<th>location\_id</th>";                  s += "</tr>"; // alert('a');                  // 지역을 클릭했을 때, 불리는 경우이다.                  if (typeof xmlOrArea == typeof '') {                      if (allMessages.length == 0) {                          alert('정보를 가져오는 중입니다.\n잠시 후 클릭하세요.');                          return;                      }                      // 저장해 둔, 메시지 배열에서, 지역명을 포함한 메시지를 찾는다.                      var cnt = 10,                          c = 0,                          ss = []; // 화면에 보일 갯수, 현재 찾은 갯수                      for (let i = 0; i < allMessages.length; i++) {                          o = allMessages[i];                          var j = o                              .location\_name                              .indexOf(xmlOrArea);                          if (j < 0)                              continue;                          s += "<tr>";                          s += "<td>" + o.create\_date + "</td>";                          s += "<td>" + o                              .location\_name                              .replace(xmlOrArea, '')                              .trim() + "</td>";                          s += "<td>" + o.msg + "</td>";                          s += "</tr>";                          c++; // 화면에 보일 갯수만큼만 찾는다.                          // 지역역 선택상자에 넣을 지역명                          for (var j = 0; j < ss.length; j++) {                              if (ss[j] == o.location\_name)                                  break;                              }                          if (j >= ss.length) // 새로운 지역명                              ss.push(o.location\_name);                              // if ( c>= cnt )   break ;                          }                      // alert(ss.join(','));                      printClickedInfo(s);                      ss.sort();                      var t = ''; // '<option>세부지역 선택</option>' ;  도시 선택 콤보박스를 초기화한다.                      for (var i = 0; i < ss.length; i++)                          t += '<option>' + ss[i]                              .replace(xmlOrArea, '')                              .trim() + '</option>';                        // alert(t);                      $('#selectCity').html(t);                      return;                  }                  // xml : 처음 시작할 때, 전체 메시지를 가져오면서 불리는 경우                  var xmlDoc = xmlOrArea.responseXML; //xml DOM 객체생성                  var x = xmlDoc.getElementsByTagName("row");                  allMessages = []; // 여기에 가져온 전체 메시지를 저장해 두고, 지역 클릭시 지역명으로 찾는다.                  for (let i = 0; i < x.length; i++) {                      o = {                          // md101\_sn : nodeValue(x[i], "md101\_sn"),                          create\_date: nodeValue(x[i], "create\_date"),                          location\_id: nodeValue(x[i], "location\_id"),                          location\_name: nodeValue(x[i], "location\_name"),                          msg: nodeValue(x[i], "msg")                      };                      allMessages.push(o);                      s += "<tr>";                      s += "<td>" + o.create\_date + "</td>";                      s += "<td>" + o.location\_name + "</td>";                      s += "<td>" + o.msg + "</td>";                      s += "</tr>";                  }              }              // 시/군 콤보박스에서 선택이 변했을 때              $('#selectCity').on('change', function () {                  var e = $(this),                      city = e.val();                  $('#disaster > tbody  > tr').each(function (index, tr) {                      tr = $(tr);                      var s = tr.text();                      if (s.indexOf(city) >= 0)                          tr.show();                      else                          tr.hide();                      }                  );              });              $('#buttonRetrieve').on('click', function () {                  retrieveMessage();              });              $('#buttonNormapMap').on('click', function () {                  map = naverMap(0);                  naverMapLoaded();              });              $('#buttonHybridMap').on('click', function () {                  map = naverMap(1);                  naverMapLoaded();              });          </script>      </body>  </html> |

## 출처

Naver Maps API : https://navermaps.github.io/maps.js/docs/

공공데이터 포털 : https://www.data.go.kr/data/3058822/openapi.do

jQuery : https://jquery.com/html/css/javascript : https://www.w3schools.com/